








# Post-hoc Scenario-based Testing of Automated Driving Systems: Classification of Driving Scenarios and Checking of Functional Requirements in Recorded Data

Till Schallau , Dominik Schmid , Nick Pawlinorz , Harun Teper ,  
Stefan Naujokat , Jian-Jia Chen <sup>†</sup>, Falk Howar <sup>‡</sup>  
Email: till.schallau@tu-dortmund.de

<sup>\*</sup>TU Dortmund University, 44227 Dortmund, Germany

<sup>†</sup>Lamarr Institute for Machine Learning and Artificial Intelligence, 44227 Dortmund, Germany

<sup>‡</sup>Fraunhofer Institute for Software and Systems Engineering, 44147 Dortmund, Germany

**Abstract**—We present a post-hoc approach for scenario-based testing of automated driving systems, enabling the analysis of safety and correctness for (cooperative) automated driving systems in many scenarios without conducting tests for individual scenarios. The system under test is operated in its physical environment, and data is recorded during operation. Then, driving scenarios are identified in this data and functional requirements are checked, yielding pass or fail verdicts for individual scenarios. We validate the envisioned post-hoc approach in a single-case mechanism experiment by the example of a platooning controller, identifying a previously unknown bug in the tested system, as well as a functional insufficiency concerning the intended operational design domain.

**Index Terms**—Scenario Classification, Requirement Monitoring, Automated Driving, Real-World Testing, Scenario-based Testing, Temporal Logics

## I. INTRODUCTION

The development of *automated driving systems* (ADS) has made significant progress in recent years. Such systems are now able to drive autonomously in various environments and scenarios. Before the widespread adoption of these systems, it is crucial to ensure they are safe and reliable. Assuring the safety of these systems, however, is still an open challenge [1]. For limiting the risk to a reasonable level, safety assurance has to document the safe behavior of an ADS in varied driving scenarios in its intended *operational design domain* (ODD).

The ISO 21448 norm [2], e.g., addresses specifically the safety of the intended functionality of automated driving systems, as opposed to their functional safety. The norm mandates that the safety of a system needs to be tested under all intended environmental conditions and concerning possible (environmental) triggering conditions that may lead to hazardous behavior [3]. To achieve this, a catalog of *known* scenarios must first be identified and used in testing. However, recognizing that exhaustive testing of all possible scenarios is unfeasible, ISO 21448 further mandates the exploration and estimation of *unknown* scenarios. The norm recommends a range of methods, including fleet tests, field experience, simulation, and real-world scenario exploration, to evaluate an ADS's performance under untested hazardous conditions and to assess the residual risk from these unknowns.

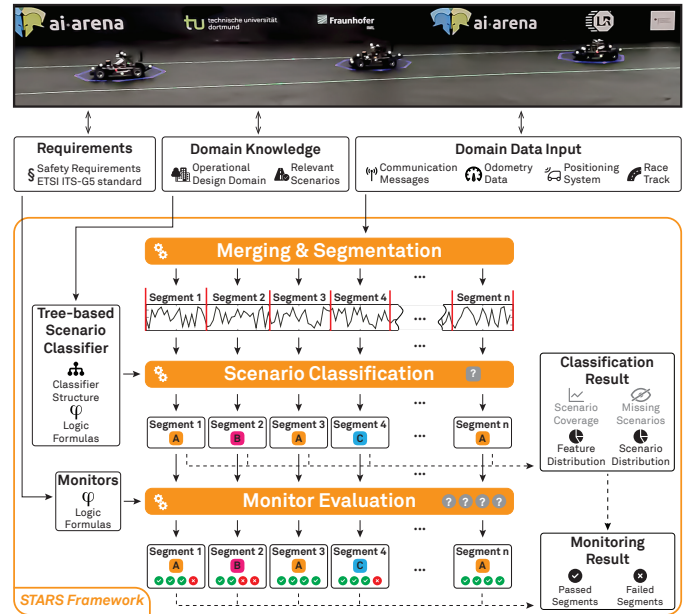


Fig. 1. Post-hoc scenario-based testing: Data is recorded from physical system during (test) operation. Features on recorded data are used to model scenarios and functional requirements. The STARS framework classifies scenarios and computes test verdicts of monitors.

Despite the norm's mandate for the estimation of residual risk, little research has specifically addressed the real-world testing of unknown scenarios [4]. Much of the existing academic work has focused on developing methods for the identification of triggering conditions [5] and for scenario-based testing [6], [7], [8] of *known* scenarios. Most of these works envision simulation as a testing method since it is relatively easy to simulate different driving scenarios and triggering conditions. While simulations can test the software of a system, real world testing is needed to ensure that the whole system behaves safely in the physical environment.

Runtime verification [9] has been suggested as one method for checking functional requirements during operation. While it certainly is helpful to monitor functional correctness during the operation, this alone is not sufficient for ensuring a

system’s safety and requires additions to a deployed system which would also require safety assurance.

In this paper, we present a method for gaining insights about the intended functionality in unknown scenarios in real word tests. Building on concepts from scenario-based testing, our approach implements system-level, post-hoc analysis that monitors safety and correctness in real-world settings, such as test fleets. The approach, sketched in Fig. 1, involves a system under test that is operated in its physical environment (e.g., as part of a test fleet or field test) while data is recorded during operation (e.g., sensor data, communication messages, actuator signals). Then, driving scenarios are identified in this data and functional requirements are monitored offline, yielding pass or fail verdicts.

This process not only reveals system insufficiencies during actual operation but also provides detailed insights into the conditions under which these issues arise. By continuously monitoring both encountered scenarios and the system’s intended functionality, our method enables a proactive exploration of unknown scenarios and the identification of unexpected hazardous conditions in accordance with ISO 21448.

In previous works, we have developed a conceptual and technical framework for the specification of scenario features on sequences of recorded data, the feature-based classification of scenarios, and the analysis of scenario coverage [10], [11]. In this paper, we extend this framework by monitoring of functional requirements.

We demonstrate the approach in a single-case mechanism experiment with model-scale vehicles and a (cooperative) automated driving system. The goal of the experiment is twofold: we aim to validate (a) the feasibility and (b) the usefulness of determining success or failure with respect to requirements for scenario-based tests extracted from recorded data.

Our results show that we can identify driving scenarios in recorded data and establish a meaningful connection to requirements. This allowed us to find a previously unknown bug in the studied system as well as a functional insufficiency with respect to its intended operational design domain. The experiment data is publicly available on Zenodo [12] and the open-source Kotlin implementation on GitHub<sup>1</sup>.

**Related Works.** Validating requirements through testing has been the subject of only a few studies, underscoring an ongoing research challenge, as highlighted by Fakhfakh et al. [13]. André et al. [14] introduce a method for testing service-based component models. Similar to our case study, vehicle platooning primarily serves as a demonstration of the method rather than focusing extensively on testing the platooning system itself.

Ponn et al. [15] present an approach to identify challenging scenarios for automated vehicles using real driving data by applying a hierarchical clustering and rule-based classification. Their work focuses on finding a reduced set of appropriate test scenarios based on their complexity. This is in contrast to

our work, as we are identifying scenarios based on triggering conditions.

Khastgir et al. [16] state that focusing on testing ADS in hazard-based scenarios is a fundamental consideration for the safety-assurance of these systems. While their focus is on identifying hazardous test scenarios, our approach similarly identifies scenarios where monitors detect violations of functional and safety requirements.

Peng et al. [17] employ timed automata, emphasizing crash prevention and safe distance determination between vehicles of different speeds. Similarly, we utilize temporal logic to validate safety properties.

**Outline.** The paper is structured as follows. Section II provides an overview of preliminary concepts. The platooning case study is described in Sect. III. We outline the experiment setup in Sect. IV, present the results in Sect. V, and discuss them in Sect. VI. The paper is concluded in Sect. VII and future work is outlined in Sect. VIII.

## II. PRELIMINARIES

We use temporal logic to model *features of scenarios* and functional requirements of vehicles as properties on finite traces of recorded data. *Scenarios* represent specific driving situations (e.g., “Vehicle enters curve with high speed”) identified by a set of observed *features* (e.g., “entering curve” and “high speed”). *Monitors* validate the system behavior by checking requirements (e.g., “the vehicle maintains a safe distance to the preceding vehicle”). We provide a short introduction to the temporal logic used in this paper (with only an informal sketch of its semantics) and the concepts of traces, features, scenarios, and scenario classification.

**Traces of Recorded Data.** The presented analysis splits the recorded data taken from the model vehicle experiment setup into finite sequences called *segments*. The data has global timestamps provided by ROS2 [18] that are used for the communication of the sensors. For every timestamp, the data contains lists of objects with properties. These objects are described using a fixed set of domain-specific concepts (e.g., vehicles with positions and velocities, and communication messages) represented as functions and relations.

**Counting Metric First-Order Temporal Binding Logic.** We model properties over segments in Counting Metric First-Order Temporal Binding Logic (CMFTBL) [10], an extension to MFOTL (Metric First-Order Temporal Logic) [19] and LTL.

In this paper we use the LTL operators *Globally*  $\Box$  and *Eventually*  $\Diamond$ . From the CMFTBL extensions we require the *Binding* operator  $\Downarrow$  and *MinPrevalence*  $\nabla$ . These allow us to compare values between different states and to reason about partial satisfaction durations of (sub-)formulas. An interval  $I := [a, b)$ , with  $a \in \mathbb{N}_0$ ,  $b \in \mathbb{N} \cup \{\infty\}$  and  $a < b$  introduced in MTL further restricts the timestamps in the sequence, that are evaluated by the operator. If an operator has not specified an interval  $I$ , we analyze the formula for the remaining timestamps in the sequence, assuming the interval  $[0, \infty)$ . The semantics of the logic is formally defined in [10]. Here,

<sup>1</sup><https://github.com/tudo-aqua/stars-auna-experiments>

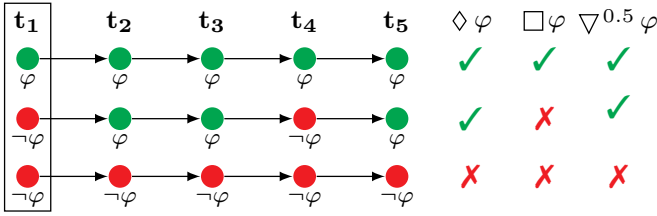


Fig. 2. Exemplary traces and evaluations of CMFTBL operators at  $t_1$

we describe the semantics of the operators needed for this paper informally. Let  $\varphi$  be a formula:

- $(\Box_I \varphi)$   $\varphi$  holds for every timestamp in the interval  $I$ .
- $(\Diamond_I \varphi)$   $\varphi$  holds for at least one timestamp in the interval  $I$ .
- $(\nabla_I^p \varphi)$   $\varphi$  holds in at least fraction  $p$  of the timestamps within interval  $I$ .
- $(\downarrow_v^t \varphi)$  Evaluates term  $t$  in the current timestamp, *binding* (“saving”) the result to variable  $v$  for future reference.

We also require the Boolean operators  $\wedge$ ,  $\vee$ ,  $\neg$ , the implication  $\implies$ , and the arbitrary tautology  $\top$ .

To provide an intuitive understanding of the operators used in this paper, we give concrete examples here. The formula  $\Diamond_{[0,5]} v.vel > vel_{max}$  evaluates whether the velocity of vehicle  $v$  exceeds the given maximum velocity  $vel_{max}$  at least once within the next 5s. Similarly, the formula  $\Box_{[0,5]} v.vel \leq vel_{max}$  evaluates whether the velocity of  $v$  stays below the maximum velocity in all timestamps in the next 5s. The formula  $\nabla^{0.5} v.vel > v_0$  requires the property  $v.vel > v_0$  (i.e. the velocity of vehicle  $v$  is greater than  $v_0$ ) to appear in at least 50% of timestamps in the analyzed segment. An illustration of these three operators is given in Fig. 2. Here, three example traces are provided that mark whether the predicate  $\varphi$  holds at each of the timestamps  $t_1$  through  $t_5$ . The example evaluation of the three operators is performed for timestamp  $t_1$ . The results are marked on the right side.

Lastly, the formula  $\downarrow_{p_0}^{v.pos} (\Diamond |p_0 - v.pos| > d)$  uses the binding operator, which freezes the position of vehicle  $v$  at the first timestamp to a new variable  $p_0$ . In the nested formula  $\Diamond |p_0 - v.pos| > d$ , which evaluates later timestamps, the position difference between the two timestamps is calculated and checked whether it eventually exceeds a threshold of  $d$ .

**Tree-based Scenario Classification.** We model and identify scenarios in segments of recorded data by using a set of *features* (specified in CMFTBL). These features describe conditions that are observable in the recorded data; for example, the feature “strong acceleration” holds if an acceleration value of over  $2 \text{ m/s}^2$  is observed anywhere in the analyzed segment.

Formally, we utilize a *tree-based scenario classifier* [10] (TSC) to organize features and their sub-features in a hierarchical tree structure. This tree reflects the taxonomy and semantic structure of the features within the ODD that the system is intended to operate in. Each node in the tree

represents a specific feature, and the edge connecting to a node is labeled with a CMFTBL formula that defines that feature. Additionally, each node is associated with bounds that specify the minimum and maximum number of child feature nodes that may occur simultaneously. These bounds are defined as follows:

<b>(A)ll</b>	=  child nodes	<b>E(X)clusive</b>	= 1
<b>(O)ptional</b>	= $0.. child \text{ nodes} $	<b>Leaf()</b>	= 0
<b>(a..b)-Bounded</b>	= a..b		

All possible feature combinations define the set of classifiable *scenarios*. Based on the TSC, we determine which edge formulas in a segment evaluate to *true*, resulting in one of these scenarios.

The key point here is that specific combinations of scenario features may form triggering conditions for the system to fail. It is therefore necessary to classify the recorded data into scenarios and then link them to system failures. By doing this, valuable insights are gained into the scenarios where failures occurred, helping to identify the source of failures.

As a basis for the work presented in this paper, we use the open-source framework STARS<sup>2</sup> that implements the concepts sketched above.

### III. CASE STUDY: THREE-VEHICLE PLATOONING

Our evaluation is based on the platooning setup by Krieger et al. [20], which uses the AuNa framework by Teper et al. [21]. In this setup, there are three F1/10 [22] vehicles (cf. Fig. 3), which form a platoon on an oval racing track, representing a scaled-down version of the Aldenhoven Testing Center race track. The resulting track has a lane width of 1.4 m. The vehicles have a width of 0.2 m, a length of 0.4 m, and a target velocity of 3 m/s. Each vehicle in the platoon uses a CACC controller [23] for longitudinal and lateral control. The control relies on two inputs.



Fig. 3. The F1/10 vehicle

First, each vehicle has a global map of the center line of the track to stay on the track and knows its global position using a motion capture system. Second, the vehicles broadcast their current state to the other vehicles in the platoon. This data includes the vehicle’s position, orientation, velocity, longitudinal acceleration, steering angle, and yaw rate. Each vehicle only uses the data of its immediate predecessor to calculate an appropriate acceleration and steering angle to stay on the lane while maintaining a safe distance to the vehicle in front. This distance is calculated based on a static distance headway and a time headway. Furthermore, for the leading F1/10 vehicle, a virtual leading vehicle is calculated that moves with the target velocity of the platoon. This virtual leading vehicle is calculated using the waypoint data and target velocity and also features an appropriate deceleration and acceleration in curves.

<sup>2</sup><https://github.com/tudo-aqua/stars/>

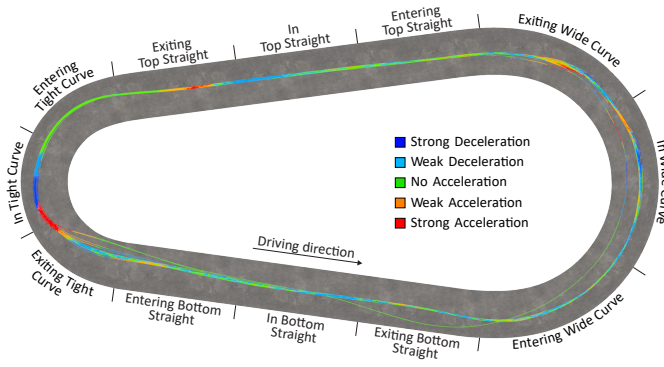


Fig. 4. Visualization of the acceleration values on the test track for the last vehicle in the platoon and marked lane segments around the track.

For the communication, the vehicles use Cooperative Awareness Messages (CAMs), as defined by the ETSI ITS-G5 standard [24]. This standard defines the format of the messages, as well as the rules for transmission. A new message must be transmitted when one of the following conditions is fulfilled:

- The position changed by more than 0.4 m. (C1)
- The velocity changed by more than 0.05 m/s. (C2)
- The heading changed by more than  $4^\circ$ . (C3)
- The elapsed time since last CAM is  $\geq 1$  s. (C4)

Here, C1 and C2 are modified from the ETSI ITS-G5 standard according to the 1/10 scale of the vehicles.

The test track consists of two straight lane sections and two curves of different radii. We split each of those into three subsections of equal length to distinguish between entering, exiting, and driving in the middle of that section (cf. Fig. 4). At any point in time, we know in which of those 12 sections a vehicle is currently driving.

#### IV. EXPERIMENT SETUP: TESTING THE PLATOONING CONTROLLER

We conduct the single-case mechanism experiment by using the approach sketched in Fig. 1 to test the platooning controller. For this, firstly, we merge all received messages into one ordered trace, then we apply a segmentation to the trace, identify features of driving scenarios that classify segments as scenarios, and finally formalize test oracles that monitor functional requirements.

First, the recorded data is synchronized and merged into a timely-ordered trace for later analysis (cf. Sect. IV-A). This trace is then segmented into semantic units based on acceleration phases (cf. Sect. IV-B). The driving scenarios are formalized and hierarchically structured using a Tree-based Scenario Classifier, with details on the TSC and associated predicates provided in Sect. IV-C. Finally, monitors addressing driving safety and communication requirements are introduced in Sect. IV-D.

##### A. Alignment of recorded data

The recorded data (30 laps, taking about 8 minutes) consists of unsynchronized ROS2 messages published by the vehicles and the Vicon motion capture system, each containing a timestamp. Ordering all messages by their timestamps yields a time-accurate ordered trace of vehicle states. With each new message, we update the global vehicle state with the newly sent information and keep all remaining property values from the previous vehicle state. Technically, the sent information is divided into four message types:

- VICON\_POSE (position and rotation),
- ODOMETRY (velocity and acceleration),
- CAM (Cooperative Awareness Messages),
- ACKERMANN\_CMD (steering angle).

The resulting data model can be found in the open-source repository for this paper<sup>1</sup>. The trace of vehicles states can then be sliced into reasonable analysis segments.

##### B. Segmentation of recorded data

Using the STARS framework, we evaluate the data recorded in the experiments. First, the trace needs to be sliced into semantic segments, in order to identify the scenario in the current driving situation. Since the vehicle's current acceleration is the central control point of the platooning controller, we segment the data based on this value. We provide a simple segmentation function that slices the data into alternating sections of acceleration and deceleration phases. Figure 5 exemplarily illustrates the acceleration values of the last vehicle for about two seconds and the resulting three partially overlapping segments, including the slicing points. This overlap ensures that there remains enough analyzable data around each tick of data, avoiding a loss of context around possible triggering conditions. We use  $\pm 0.4 \text{ m/s}^2$  as an overlapping interval, which has experimentally proven to be a fitting value for this scenario, so that ticks inside this window belong to both adjoining segments. That means, in each acceleration phase, there are no ticks with  $acceleration < -0.4 \text{ m/s}^2$  and in each deceleration phase, there are no ticks with  $acceleration > 0.4 \text{ m/s}^2$ . The acceleration values of the last vehicle for the whole experiment is visualized in Fig. 4.

This segmentation method is based on Peters et al. [25], where data slicing was learned using automata learning. Slicing the 377,568 ticks of data for the second and third vehicle by the acceleration window yields a total of 673 and 622 partially overlapping segments containing 481,009 and 492,696 ticks respectively. We ignore the first vehicle since it only follows the virtual leading vehicle and is not constrained by the driving behavior of an actual preceding vehicle.

##### C. Features of Driving Scenarios

To classify the scenarios encountered by the vehicles during the experiment, we define various scenario features which we grouped into two categories: on the one hand, we distinguish the *driving situations* which describe what the vehicle is confronted with. On the other hand, we identify *driving*



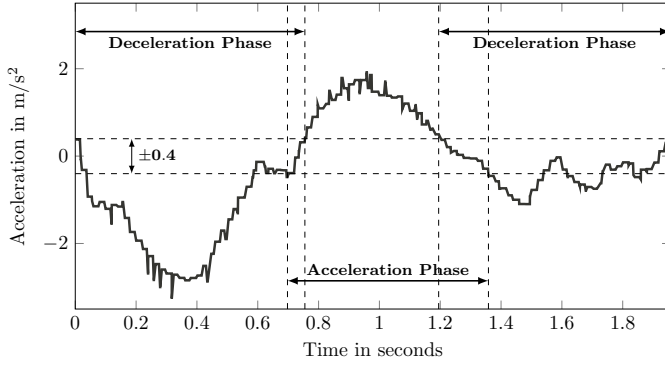


Fig. 5. Example segmentation of three segments for the last vehicle, showing deceleration and acceleration phases and their overlaps based on the  $\pm 0.4 \text{ m/s}^2$  overlapping interval.

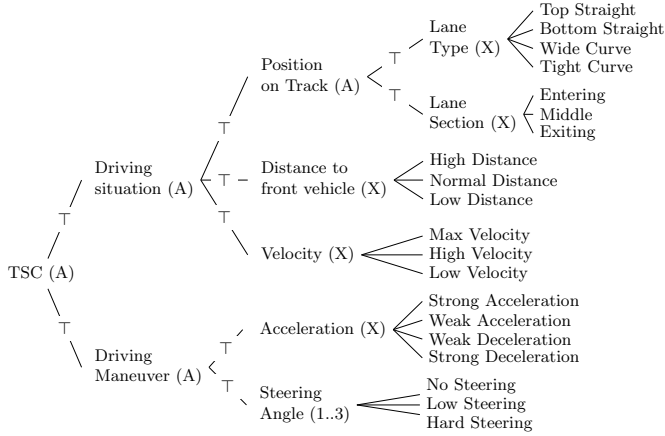


Fig. 6. The TSC used in our experiments. The bounds attached to the nodes are: (A)ll and E(X)clusive. All edges labeled with  $\top$  have the formula "true". Here, only leaf nodes represent relevant features. *Position on Track* features are derived from Fig. 4, while all other features are formalized in Tables I and II.

*maneuvers* that the vehicle decides to take. Figure 6 shows these two groups with their corresponding features hierarchically organized in a TSC. A specific scenario that could, for instance, be encountered is that the vehicle drives through the *exit* of the *wide curve* with a *high distance to the front vehicle* while driving with a *low steering angle* at *low velocity*, but with *strong acceleration*. All edge conditions of the TSC that are not depicted in Fig. 6 are described below.

**Driving situations.** We classify the driving situations mainly by the information considered by the CACC controller to calculate the next action. This includes the road curvature, the distance to the preceding vehicle, and the current velocity. The threshold values for the classification (e.g., *hard steering*) as well as the corresponding CMFTBL formulas to identify this feature in a segment are given in Table I.

**Driving maneuvers.** The CACC controller only controls the current acceleration and steering angle. The threshold values and formulas for the classifications are given in Table II. Due to the segmentation method, exactly one of the two predicates,

TABLE I  
THRESHOLD VALUES AND PREDICATES FOR DISTANCE AND VELOCITY

Distance	Threshold values	CMFTBL predicate
High	(3 m, $\infty$ )	$\downarrow \frac{v.pos}{p_0} \diamond d(p_0, v.pos) > 3$
Normal	[1.5 m, 3 m]	$\downarrow \frac{v.pos}{p_0} \square d(p_0, v.pos) \in [1.5, 3]$
Low	[0 m, 1.5 m]	$\downarrow \frac{v.pos}{p_0} \diamond d(p_0, v.pos) < 1.5$

Velocity	Threshold values	CMFTBL predicate
Max	(2.75 m/s, $\infty$ )	$\nabla^{0.5} v.vel > 2.75$
High	[2 m/s, 2.75 m/s]	$\nabla^{0.5} v.vel \in [2, 2.75]$
Low	[0 m/s, 2 m/s]	$\nabla^{0.5} v.vel < 2$

TABLE II  
THRESHOLD VALUES AND PREDICATES FOR ACCELERATION AND STEERING ANGLE

Acceleration	Threshold values	CMFTBL predicate
Strong	(2 m/s <sup>2</sup> , $\infty$ )	$\diamond v.acc > 2$
Weak	(0.4 m/s <sup>2</sup> , 2 m/s <sup>2</sup> ]	$\diamond v.acc > 0.4 \wedge \neg \text{Strong}(v)$

Deceleration	Threshold values	CMFTBL predicate
Strong	( $-\infty$ , -2 m/s <sup>2</sup> )	$\diamond v.acc < -2$
Weak	[-2 m/s <sup>2</sup> , -0.4 m/s <sup>2</sup> )	$\diamond v.acc < -0.4 \wedge \neg \text{Strong}(v)$

Steering	Threshold values	CMFTBL predicate
Hard	(10°, $\infty$ )	$\diamond  v.angle  > 10$
Low	[2.5°, 10°]	$\diamond  v.angle  \geq 2.5 \wedge \neg \text{Hard}(v)$
No	[0°, 2.5°)	$\square  v.angle  < 2.5$

either acceleration or deceleration, holds for each segment. For the classification of the steering angle, only the maximum observed steering is considered. For example, we define *low steering* if, in this segment, there eventually is a steering angle of over 2.5° in the given interval, and *hard steering* is not observed in the whole segment.

#### D. Monitors for Functional Requirements

To validate safety-critical properties of the platooning controller, we use global monitors in the STARS framework. These observe the behavior throughout the whole experiment and report violations of the defined properties including the timestamp, current segment, and the classified scenario. In contrast to classical safety monitoring, this enables the categorization of critical instances that lead to a system failure or misbehavior, and therefore helps to identify critical combinations of environmental conditions and the current control state. With this, we can identify triggering conditions that may lead to a violation of the safety requirements. For the definition of the monitors, we apply the same logic as for the scenario classification.

The monitors are grouped into two categories: *driving safety* and *communication safety*.

**Driving safety monitors.** The first group contains monitors regarding the safe operation of the system concerning the driving maneuvers. The platooning vehicles are following the waypoints of the lane. First, we check for the lateral offset of

the vehicles to the middle of the lane to ensure that the vehicles do not leave the track. As the track has a width of 0.7 m in both directions, we choose 0.4 m as the threshold for lateral offset, as exceeding this value would bring the vehicles too close to the edge of the track. Therefore, we monitor the lateral offset for all vehicles  $v \in \mathcal{V}$  using the following formula:

$$\text{maxLateralOffset}(v) := \square v.\text{lateralOffset} \leq 0.4$$

Second, in a platooning case, vehicles must maintain a consistent distance to the preceding vehicle. The distance should not be too short, as this might cause accidents, but also not too far, as the vehicle then might lose connection. The function  $d(p_0, p_1)$  calculates the Euclidean distance between positions  $p_0$  and  $p_1$ . Let  $v_p$  be the preceding vehicle of  $v \in \mathcal{V}$  for the following formulas.

For the minimum distance to the preceding vehicle, we take the braking distance based on the current velocity of vehicle  $v$  as the safety-critical threshold, conservatively assuming the velocity of  $v_p$  to be 0 m/s.

$$\text{minDistance}(v) := \square d(v.\text{pos}, v_p.\text{pos}) > (0.36 \times v.\text{vel})^2$$

For the maximum distance, we take the threshold for *high distance* (3 m) plus 25% as the safety-critical value. The monitor is similarly defined as the minimum distance:

$$\text{maxDistance}(v) := \square d(v.\text{pos}, v_p.\text{pos}) < (3 \times 1.25)$$

Third, especially in platooning scenarios, emergency brakes are dangerous. Therefore, we check that the vehicles never brake harder than 3 m/s<sup>2</sup>.

$$\text{maxDeceleration}(v) := \square v.\text{acc} > -3.0$$

**Communication monitors.** The vehicles use CAM messages and should, therefore, comply with the format and rules of the underlying standard. For each of the conditions (C1)-(C4) defined at the beginning of this section, we implemented a corresponding monitor. For each communication message, we assert an arrival within 5 ms from the time they should have been sent. First, a position change by more than 0.4 m should send a new CAM message (C1). This monitor builds upon multiple predicates:

$$\begin{aligned} \text{camMessagePositionChange}(v) := \\ \square \downarrow_{p_0}^{v.\text{pos}} \left( \text{isCAM}(v) \implies (\text{noMorePositionChanges}(v) \right. \\ \left. \vee (\diamond d(p_0, v.\text{pos}) \geq 0.4 \wedge \text{nextCAMInTime}_{t=5}(v))) \right) \end{aligned}$$

The first part of the monitor checks whether a position change occurs in the remaining data stream.

$$\text{noMorePositionChanges}(v) := \downarrow_{p_0}^{v.\text{pos}} \square d(p_0, v.\text{pos}) < 0.4$$

Then we check whether there is another CAM message within tms or the stream of data ends.

$$\text{nextCAMInTime}_t(v) := \diamond_{[0,t]} v.\text{source} = \text{CAM} \vee \neg \diamond_{[t,\infty]} \top$$

Similarly, we define a monitor for velocity changes (C2)

$$\begin{aligned} \text{camMessageSpeedChange}(v) := \\ \square \downarrow_{v_0}^{v.\text{vel}} \left( \text{isCAM}(v) \implies (\text{noMoreSpeedChanges}(v) \right. \\ \left. \vee (\diamond |v_0 - v.\text{vel}| \geq 0.05 \wedge \text{nextCAMInTime}_{t=5}(v))) \right) \end{aligned}$$

$$\text{noMoreSpeedChanges}(v) := \downarrow_{v_0}^{v.\text{vel}} \square |v_0 - v.\text{vel}| < 0.05$$

and heading changes (C3).

$$\begin{aligned} \text{camMessageHeadingChange}(v) := \\ \square \downarrow_{h_0}^{v.\text{heading}} \left( \text{isCAM}(v) \implies (\text{noMoreHeadingChanges}(v) \right. \\ \left. \vee (\diamond \text{headingChange}(h_0, v.\text{heading}) \geq 4 \right. \\ \left. \wedge \text{nextCAMInTime}_{t=5}(v))) \right) \end{aligned}$$

$$\begin{aligned} \text{noMoreHeadingChanges}(v) := \\ \downarrow_{h_0}^{v.\text{heading}} \square \text{headingChange}(h_0, v.\text{heading}) < 4 \end{aligned}$$

$$\text{headingChange}(h_0, h_1) := ((h_0 - h_1 + 180) \bmod 360) - 180$$

Finally, the controller has to send a CAM message at least every 1 s (C4) (+ 5 ms slack).

$$\begin{aligned} \text{nextCAMMessageWithin1000ms}(v) := \\ \square (\text{isCAM}(v) \implies \text{nextCAMInTime}_{t=1005}(v)) \end{aligned}$$

## V. EVALUATION:

### MONITOR VERDICTS FOR TEST DATA

The monitor for the maximum deceleration value failed 64 times, indicating that the vehicles had to perform this many emergency braking maneuvers. Utilizing the associated scenarios from our recorded data, we pinpointed the triggering conditions responsible for these critical situations. As demonstrated in Fig. 7 and Fig. 4, the majority of these hard braking events occurred within or immediately before the tight curve section. Analyzing the driving data further revealed that the vehicles consistently entered this curve segment at high speeds. Due to this, the vehicles followed trajectories with turning circles too large for the track geometry, resulting in emergency braking to avoid leaving the track.

The intense braking maneuvers resulted in three occurrences of the third vehicle losing the connection to its preceding

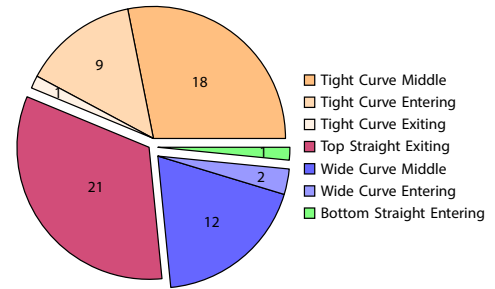


Fig. 7. Lane sections with observed *maxDeceleration*

vehicle, as captured by the failing monitors for *maxDistance*. Simultaneously, the monitor for *maxLateralOffset* also failed, indicating that the vehicle had nearly exited the track by cutting across the tight curve directly into the subsequent straight lane section, as depicted in Fig. 4. The aggregated data and subsequent analysis clearly highlight that the vehicles consistently followed the trajectories of their immediate predecessors without independently adjusting their trajectory according to the specific track geometry. This explicit behavioral pattern was observable in the recorded data, even without prior knowledge of the programmed controller logic. Such shortcomings in trajectory planning and environmental awareness could lead to critical or hazardous situations, especially in real-world driving scenarios with dynamic traffic conditions. These findings were subsequently visually inspected through additional simulations to confirm the identified critical conditions.

Additionally, the monitor for the CAM messages on *speed changes* failed 126 times. Detailed analysis of this issue allowed us to pinpoint a specific bug within the controller software: a missing absolute value calculation in the logic handling speed change events. Consequently, the controller erroneously triggered CAM messages only upon increases in speed, neglecting necessary notifications when speed reductions occurred. This not only violated the ETSI ITS-G5 standard for Cooperative Awareness Messages but this may also have significantly degraded communication reliability within the vehicle platoon.

The insights gained through this extended evaluation demonstrate the value of the combination of scenario analysis and monitoring.

## VI. DISCUSSION

We applied global safety monitoring combined with scenario classification to identify bugs and undesired behaviors of the analyzed platooning controller. In contrast to per-scenario testing, no explicit test setup is required to expose the system to various situations. Instead, the system is evaluated in its natural environment, and the recorded data is then classified post-hoc.

Using this scenario-based testing approach, we successfully located a previously undiscovered software bug, affecting the correctly timed communication of the controllers. Furthermore, the tight curve radius of the track was revealed as a problematic driving situation for the platooning vehicles. The combination of failing monitors and scenario classification enabled us to identify this as a triggering condition for the system: entering the tight curve at high speed, caused emergency braking, which is then followed by loss of connection with the preceding vehicle. This behavior also led to vehicles leaving the track by short-cutting through the curve.

Moreover, we detected a functional insufficiency in the implementation of the CACC controller with respect to the requirements of the ETSI ITS-G5 standard for CAM messages.

For our experiments we used the criteria and their thresholds of the ETSI ITS-G5 standard (cf. C1-C4) and values of the platooning controller implementation (cf. Tabs. II and I).

It should be noted that the discretizations and thresholds applied must be defined by the respective operational design domain or be derived from applicable legislative texts. These thresholds critically influence the interpretation of scenario features and the validation of system behavior. In lack of concrete requirements for our use-case, we defined thresholds based on our experience, which may differ, when applying our approach to other domains.

A further limitation of our study stems from the use of a highly accurate motion capture system for sensor data acquisition. While this ensures precise measurements, it may not reflect the inherent uncertainties present in real-world sensor systems. Consequently, the high fidelity of our sensor data might limit the generalizability of our findings to systems operating under less controlled sensing conditions.

## VII. CONCLUSION

This paper presented a post-hoc evaluation of driving data from a platooning use-case, utilizing the STARS framework for analysis. We introduced global safety monitoring accompanying scenario classification to systematically investigate triggering conditions that result in system misbehavior. This methodology led to the identification of both a previously undetected software bug and a functional insufficiency concerning the intended operational design domain.

By leveraging existing data from the system's operation in a realistic environment, this approach minimizes the need for labor-intensive and explicit scenario design. This enables a more scalable and efficient way of validating automated driving systems. Our findings demonstrate that post-hoc scenario-based testing can serve as a method that supports classical approaches of uncovering both requirement violations and software insufficiencies in controller logic.

The experiment setup relied on the formal definition of predicates for driving situations, maneuvers, and functional requirements. These were derived directly from the ETSI ITS-G5 standard and from threshold values of the platooning controller implementation.

Our work introduced the value of combining scenario classification with safety monitoring in post-hoc evaluations, especially in contexts where real-world data collection is feasible, and explicit scenario design is impractical.

## VIII. FUTURE WORK

As future work, we intend to take scenario class coverage and the distribution of feature combinations into consideration in order to argue about the completeness of automated driving system testing. Furthermore, we plan to (1) infer scenario descriptions from missing feature combinations to automatically generate simulation setups for the system under test, and (2) plan optimal routing instructions to cover missing features in real-world testing. By enforcing exposure to various feature combinations, we aim to identify edge cases inducing triggering conditions that lead to system failures.

In ongoing research, we are also investigating the impact of *unknown* or *unclassifiable* data points on scenario classification and coverage metrics. This includes assessing how well

the system generalizes to new, unanticipated situations and how such data can be integrated into the scenario classification process.

Additionally, we plan to inject artificial software bugs and faults into the system under test to evaluate the sensitivity and effectiveness of our monitoring framework in detecting various classes of errors. This fault injection will help quantify the robustness of our approach in identifying functional and safety-critical issues.

Lastly, we aim to scale our methodology to real-world driving situations. For this, we are preparing experiments utilizing a sensor-equipped bicycle, representing sensor setups commonly found in automated driving systems. This will allow us to explore scalability challenges and apply our scenario-based testing approach to actual traffic data, providing insights into its practical deployment and limitations in more complex, uncontrolled environments.

## REFERENCES

- [1] R. Memon, K. Arezoo, K. Alipour, and M. Ghamari, "Autonomous driving systems: An overview of challenges in safety, reliability and privacy," in *2022 15th International Conference on Human System Interaction (HSI)*. IEEE, Jul. 2022, pp. 1–7, doi: 10.1109/HSI55341.2022.9869489.
- [2] International Organization for Standardization, "Road vehicles — safety of the intended functionality," ISO, ISO Standard 21448:2022, Jun. 2022. [Online]. Available: <https://www.iso.org/standard/77490.html>
- [3] A. K. Saberi, J. Hegge, T. Fruehling, and J. F. Groote, "Beyond SOTIF: Black swans and formal methods," in *2020 IEEE International Systems Conference (SysCon)*. IEEE, Aug. 2020, pp. 1–5, doi: 10.1109/SysCon47679.2020.9275888.
- [4] N. F. Salem, T. Kirschbaum, M. Nolte, C. Lalitsch-Schneider, R. Graubohm, J. Reich, and M. Maurer, "Risk management core - toward an explicit representation of risk in automated driving," *IEEE Access*, vol. 12, pp. 33 200–33 217, 2024, doi: 10.1109/ACCESS.2024.3372860.
- [5] Z. Zhu, R. Philipp, C. Hungar, and F. Howar, "Systematization and identification of triggering conditions: A preliminary step for efficient testing of autonomous vehicles," in *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, Jun. 2022, pp. 798–805, doi: 10.1109/IV51971.2022.9827238.
- [6] M. Scholtes, L. Westhofen, L. R. Turner, K. Lotto, M. Schuldes, H. Weber, N. Wagener, C. Neurohr, M. H. Bollmann, F. Körtke, J. Hiller, M. Hoss, J. Bock, and L. Eckstein, "6-layer model for a structured description and categorization of urban traffic and environment," *IEEE Access*, vol. 9, pp. 59 131–59 147, Apr. 2021, doi: 10.1109/ACCESS.2021.3072739.
- [7] S. Ulbrich, T. Menzel, A. Reschka, F. Schultdt, and M. Maurer, "Defining and substantiating the terms scene, situation, and scenario for automated driving," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, Sep. 2015, pp. 982–988, doi: 10.1109/ITSC.2015.164.
- [8] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles," *IEEE Access*, vol. 8, pp. 87 456–87 477, May 2020, doi: 10.1109/ACCESS.2020.2993730.
- [9] E. Zapridou, E. Bartocci, and P. Katsaros, "Runtime verification of autonomous driving systems in CARLA," in *Runtime Verification*, ser. Lecture Notes in Computer Science, J. Deshmukh and D. Ničković, Eds., Cham: Springer International Publishing, Oct. 2020, pp. 172–183, doi: 10.1007/978-3-030-60508-7\_9.
- [10] T. Schallau, S. Naujokat, F. Kullmann, and F. Howar, "Tree-based scenario classification," in *NASA Formal Methods*, ser. Lecture Notes in Computer Science, N. Benz, D. Gopinath, and N. Shi, Eds., vol. 14627. Cham: Springer Nature Switzerland, May 2024, pp. 259–278, doi: 10.1007/978-3-031-60698-4\_15.
- [11] T. Schallau, D. Mäkel, S. Naujokat, and F. Howar, "STARS: A tool for measuring scenario coverage when testing autonomous robotic systems," in *Dependable Computing – EDCC 2024 Workshops*, ser. Communications in Computer and Information Science, B. Sangchoolie, R. Adler, R. Hawkins, P. Schleiss, A. Arteconi, and A. Mancini, Eds., vol. 2078. Cham: Springer Nature Switzerland, Mar. 2024, pp. 62–70, doi: 10.1007/978-3-031-56776-6\_6.
- [12] H. Teper, C. Krieger, I. Priyanta, J. Freytag, P. Schulte, M. Roidl, C. Wietfeld, and J.-J. Chen, "F1/10 platooning with etsi its-g5 communication using ros 2," doi: 10.5281/zenodo.13710763, Nov. 2024.
- [13] F. Fakhfakh, M. Tounsi, and M. Mosbah, "Vehicle platooning systems: Review, classification and validation strategies," *International Journal of Networked and Distributed Computing*, vol. 8, no. 4, pp. 203–213, Sep. 2020, doi: 10.2991/ijndc.k.200829.001.
- [14] P. Andre, J.-M. Mottu, and G. Ardourel, "Building test harness from service-based component models," in *Proceedings of the 10th International Workshop on Model Driven Engineering, Verification and Validation Co-Located with 16th International Conference on Model Driven Engineering Languages and Systems (MODELS 2013)*, ser. CEUR Workshop Proceedings, F. Boulanger, M. Famelis, and D. Ratiu, Eds., vol. 1069, Miami, United States, Oct. 2013, pp. 11–20. [Online]. Available: <https://hal.science/hal-00918505>
- [15] T. Ponn, M. Breituß, X. Yu, and F. Diermeyer, "Identification of challenging highway-scenarios for the safety validation of automated vehicles based on real driving data," in *2020 Fifteenth International Conference on Ecological Vehicles and Renewable Energies (EVER)*. IEEE, Sep. 2020, pp. 1–10, doi: 10.1109/EVER48776.2020.9242539.
- [16] S. Khastgir, S. Brewerton, J. Thomas, and P. Jennings, "Systems approach to creating test scenarios for automated driving systems," *Reliability Engineering & System Safety*, vol. 215, p. 107610, Nov. 2021, doi: 10.1016/j.res.2021.107610.
- [17] C. Peng, M. M. Bonsangue, and Z. Xu, "Model checking longitudinal control in vehicle platoon systems," *IEEE Access*, vol. 7, pp. 112 015–112 025, 2019, doi: 10.1109/ACCESS.2019.2935423.
- [18] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, May 2022, doi: 10.1126/scirobotics.abm6074.
- [19] D. Basin, F. Klaedtke, S. Müller, and E. Zălinescu, "Monitoring metric first-order temporal properties," *Journal of the ACM (JACM)*, vol. 62, no. 2, pp. 15:1–15:45, May 2015, doi: 10.1145/2699444.
- [20] C. Krieger, H. Teper, J. Freytag, I. F. Priyanta, P. Schulte, M. Roidl, J.-J. Chen, and C. Wietfeld, "Integration of scaled real-world testbeds with digital twins for future AI-enabled 6g networks," in *2023 IEEE Globecom Workshops (GC Wkshps)*. IEEE, Dec. 2023, pp. 2037–2042, doi: 10.1109/GCWkshps58843.2023.10464605.
- [21] H. Teper, A. Bayuwindra, R. Riebl, R. Severino, J.-J. Chen, and K.-H. Chen, "AuNa: Modularly integrated simulation framework for cooperative autonomous navigation," arXiv, arXiv Preprint arXiv:2207.05544, Jul. 2022, doi: 10.48550/arXiv.2207.05544.
- [22] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "F1tenth: An open-source evaluation environment for continuous control and reinforcement learning," in *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*, ser. Proceedings of Machine Learning Research, H. J. Escalante and R. Hadsell, Eds., vol. 123. PMLR, Aug. 2020, pp. 77–89. [Online]. Available: <https://proceedings.mlr.press/v123/o-kelly20a.html>
- [23] A. Bayuwindra, J. Ploeg, E. Lefeber, and H. Nijmeijer, "Combined longitudinal and lateral control of car-like vehicle platooning with extended look-ahead," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 3, pp. 790–803, May 2020, doi: 10.1109/TCST.2019.2893830.
- [24] European Telecommunications Standards Institute, "Intelligent Transport Systems (ITS); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service," ETSI, ETSI EN 302 637-2 v1.4.1, Oct. 2019. [Online]. Available: <https://www.en-standard.eu/csn-etsi-en-302-637-2-v1-4-1-intelligent-transport-systems-its-vehicular-communications-basic-set-of-applications-part-2-specification-of-cooperative-awareness-basic-service/>
- [25] H. Peters, F. Howar, and A. Rausch, "Towards inferring environment models for control functions from recorded signal data," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 2. IEEE, Mar. 2016, pp. 1–4, doi: 10.1109/SANER.2016.83.