Electronic Communications of the EASST
Volume 82 (2022)

11th International Symposium
on Leveraging Applications of Formal Methods, Verification
and Validation
-
Doctoral Symposium, 2022

Validating Behavioral Requirements, Conditions, and Rules of Autonomous Systems with Scenario-Based Testing

Till Schallau and Stefan Naujokat

21 pages

Guest Editors: Sven Jörges, Salim Saay, Steven Smyth
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Validating Behavioral Requirements, Conditions, and Rules of Autonomous Systems with Scenario-Based Testing

Till Schallau[1] and **Stefan Naujokat**[2]

[1] till.schallau@tu-dortmund.de, [2] stefan.naujokat@tu-dortmund.de
Chair for Software Engineering
TU Dortmund University, Dortmund, Germany

Abstract: Assuring the safety of autonomous vehicles is more and more approached by using scenario-based testing. Relevant driving situations are utilized here to fuel the argument that an autonomous vehicle behaves correctly. Many recent works focus on the specification, variation, generation, and execution of individual scenarios. However, it is still an open question if operational design domains, which describe the environmental conditions under which the system under test has to function, can be assessed with scenario-based testing. In this paper, we present open challenges and resulting research questions in the field of assuring the safety of autonomous vehicles. We have developed a toolchain that enables us to conduct scenario-based testing experiments based on scenario classification with temporal logic and driving data obtained from the CARLA simulator. We discuss the toolchain and present first results using analysis metrics like class coverage or distribution.

Keywords: Autonomous Vehicles, Domain-Specific Languages, Scenario-based Testing

1 Introduction

Assuring the safety of autonomous vehicles is still an open challenge [Mar18]. Especially, considering the complex environment they are operating in and the underlying vaguely defined traffic rules they have to obey. In previous works, it was shown that a statistical argumentation on the safety and performance of autonomous systems (e.g., caused fatalities per million miles, crashes due to wrongly classified environmental objects [BZMS16], etc.) are not feasible in practice [JWKW18, KP16]. Especially, as billions of miles have to be driven to evaluate the functional correctness of subsystems or the whole autonomous system for each new vehicle model or software update. Therefore, the focus of research in the last years concentrated on assuring the safety of driving functions instead [MHR16, FKL$^+$19].

The ISO 26262 standard [ISO18] has set the state of the art for the correct operation of autonomous systems [SHFG20]. The more recent ISO 21448 standard [ISO22] describes the assurance of a vehicle's safety under all environmental conditions and their triggering conditions. Triggering conditions describe specific conditions of a scenario that may initialize hazardous situations or prevent the system from correctly detecting required overrides of the system (e.g., driver input is ignored). To be more cost-efficient and not endanger traffic participants, such as pedestrians, these conditions are implemented with concrete values in specialized scenarios [NMB+20]. These scenarios can be described [SWT+21, UMR+15] and built [SSLM13] using stacked layers that contain distinct and separate sets of information. Instead of testing the system under real-world conditions, the driving functions are first evaluated using such scenario-based testing environments. After successfully operating in these scenarios, those systems are then undergoing real-world testing. The utilized scenarios allow for concrete situation definitions that are reproducible as well as predictable. Additionally, test engineers are capable of defining expected maneuvers and specific outcomes that can then automatically be validated. Nevertheless, real-world tests are mandatory as they can be utilized to find relevant and critical scenarios [KKHK19, WLFM18].

Some of the observed environmental conditions and traffic situations might not be relevant for the driving function under test. Therefore, when developing autonomous systems, a first step is to define a so-called operational design domain (ODD) [Cen20], in which the system's operating environment is defined. This includes, but is not limited to, behavioral requirements, environmental conditions (e.g., weather, time of day, etc.), roadway features (e.g., count of lanes, road types, traffic signs, etc.), traffic laws, regulations, and other domain concerns.

Previous works already utilized formal logic to define traffic rules for (uncontrolled) intersections [MMA22, KD20], interstate traffic [MRMA20], overtaking maneuvers [RKH+17] and for the safe distance between vehicles [EGK20].

Extending on these approaches, we analyze the question of whether it is possible to describe all ODD requirements logically and machine-interpretable by utilizing formal specifications of features. This is of great importance as it would allow for automatic analyses of predefined scenarios. It is an open question on how suitable scenario-based testing is for evaluating autonomous systems based on their corresponding ODD. We aim to analyze and validate this applicability and propose to create a domain-specific language (DSL) that is capable of describing the ODD and its logical descriptions of features, but on a more abstract level suitable for domain experts.

In this paper, these goals are formulated into three research questions, which we discuss and bring into context. In a previous research effort [SNKH23], we introduced an extension to existing formal logics that is more expressive and presented several metrics that measure the coverage and representativeness of a set of test scenarios based on a given specification. Here, we relate the results from this work to our research questions, and, in particular, detail our prototype implementation of a simulation and analysis toolchain to address the first two of our questions. The implemented project is discussed and future adaptions to solve the third question are summarized.

Outline The paper is structured as follows. Section 2 formulates the research questions and reasons about them in the context of our research roadmap. Section 3 motivates our experiment setup and outlines technical aspects of our toolchain. A short summary of the results from the case study in [SNKH23] is given in Section 4 alongside their impact towards answering our research questions. Future work and next steps are described in Section 5. We give an overview of related works in Section 6 and conclude the paper in Section 7.

2 Problems and Research Questions

In this section, we discuss open problems in the field of assuring the safety of autonomous vehicles. Focusing on operational design domains, scenario-based testing, and domain-specific languages, we sketch a research roadmap along with three research questions derived from the discussed problems.

2.1 Formalizing Operational Design Domains

An operational design domain describes the environmental conditions under which an autonomous system has to operate properly. When developing new driving tasks, domain experts develop suitable ODDs based on their domain knowledge and related traffic laws. The resulting ODD documents are currently designed for the software and hardware engineers and may include informal instructions and traffic laws, such as "No motor vehicle must, without good reason, travel so slowly as to impede the flow of traffic." (§3(2) German Traffic Law - STVO[1]). As this kind of requirement is too vague for automatic evaluation, machine-interpretable specification mechanisms for operational design domains need to be developed. This results in the first research question:

RQ 1: Is it possible to model and validate autonomous systems' behavioral requirements, conditions, and rules – as defined in an operational design domain – using formal logic?

Answering this research question should give insight into the value of operational design domains and how they are designed for specific driving functions and whole autonomous systems. Furthermore, we expect to gain a more detailed understanding of how far formal logics can be utilized to define behavioral requirements and traffic rules. In this process, additions to existing logical formalization approaches are required to fully define traffic laws and conditions. In a first step, we imagine a manual conversion step between predefined ODDs and their corresponding formal representations.

2.2 Utilization of Scenario-Based Testing

Given a set of machine-interpretable logical formulas describing the environmental conditions, behaviors, and traffic rules, the question arises of how these formal specifications can be automatically validated to assure the compliance of the autonomous system with the underlying ODD. Today, it is widely assumed that safety assurance will have to be conducted iteratively for different driving scenarios through a combination of assurance efforts that for a particular scenario can document with some certainty the safety of an autonomous vehicle [FFS+23]. It is an open question, however, how scenario-based testing of autonomous systems in combination with predefined ODDs is possible. From these considerations, the second research question arises:

RQ 2: Is it possible to utilize scenario-based testing to ensure that an autonomous system fulfills the claims defined by its operational design domain?

[1] https://germanlawarchive.iuscomp.org/?p=1290

To be able to conclude whether an autonomous system complies with its intended domain, some form of numerical analysis may be implemented. We imagine a set of metrics that represent the coverage of a given set of scenarios in consideration of predefined requirements.

This benefits the testing of autonomous vehicles and their implemented driving functions, as the expressiveness of sets of scenarios can be evaluated quantitatively and test scenarios can, ultimately, be developed more elaborately.

2.3 Domain-Specific Rule Formalization

By finding solutions for RQ 1 and RQ 2, we can formalize the requirements of predefined ODD specifications, which are then automatically analyzed by a scenario-based test approach. This workflow still requires the domain experts to informally express the operational design domain and then a logics expert to convert it into its equivalent formal specification and formulas. To counteract this kind of error-prone workflow, which often results in multiple roundtrips between domain experts and logics experts, we propose a third research question:

RQ 3: Is it possible to build a domain-specific language for behavioral requirements, conditions, and rules that is usable by domain experts and has the required formal logic as a generation target?

A specialized domain-specific language allows for the simultaneous development of operational design domains by domain experts and the automatic generation into a corresponding representation as a set of logical formulas. The difficulty of this approach lies in the development of a suitable DSL that is capable of being understandable by domain experts and expressive enough to be formally generatable.

Furthermore, with such a domain-specific language, we enable an iterative approach of analyzing and testing an autonomous system against a given ODD. By analyzing missing conditions or violated laws, it might even be possible to automatically generate new test cases for the scenario-based test approach.

3 Experiment Setup

Our general approach towards answering the research questions is to conduct experiments in which we formally specify a vehicle's requirements relative to its ODD and analyze those specifications on recordings of (for now, only simulated) scenarios. On the one hand, we aim for a specification that is expressive enough to formally capture all relevant properties. On the other hand, analyzing test scenarios requires some form of *coverage* quantification on features and other qualitative analyses to reason about having tested the autonomous vehicle sufficiently. Each scenario consists of a time-stamped sequence of scenes that hold all information about the ego vehicle (i.e., the vehicle under evaluation) and its surroundings. We started building a theoretical framework and a corresponding proof-of-concept tool pipeline to conduct our experiments. We focus here more on the technical aspects of this tool pipeline, while the formal background of this framework is beyond the scope of this paper. We do, however, briefly present an informal overview of the framework's concepts and formalisms to provide an intuition for understanding the tool pipeline. For the full formal elaboration, please refer to [SNKH23].

- We introduce Counting Metric First-Order Temporal Binding Logic (CMFTBL) as an extension to Metric First-Order Temporal Logic (MFOTL) on finite traces [Cho95, Mül09]. In particular, our extension (a) defines a *minimum prevalence operator* to express that a property (or sub-formula) holds for a certain fraction of all future states (within the finite trace), and (b) introduces a *binding operator* that stores an evaluation of a term into a variable, so that the result of this evaluation can be accessed in operator contexts of future states.

- We introduce Tree-Based Scenario Classifiers (TSCs) as a structured way to create semantical groups of autonomous systems' features. With these classifiers, we extend the concept of defining a system's capabilities in form of an operational design domain by introducing structural properties reminiscent of feature model diagrams [SHT06] (e.g., introducing optional and mandatory nodes). Each valid sub-structure of the classifier tree forms a scenario class and a recording of a scenario is classified as this class if all nodes' conditions (expressed as CMFTBL formulas) hold.

- We define multiple metrics and analyses to quantify and analyze if, and to which degree, the recorded data covers possible scenario classes. The primary metric we are considering is *scenario class coverage*, expressing the ratio between the number of observed classes and the number of classes modeled by a TSC. To measure the individual rarity of the modeled environmental conditions, we introduce a metric for *absolute feature occurrence*. Furthermore, the *scenario instance count* metric determines how often a certain class has been encountered. Complementing this information, we also identify *class instance missings*. However, as gaining meaningful insights from large sets of missing classes is difficult, we also analyze *feature pair misses*, i.e., pairs of TSC nodes that do not exist together in any of the observed classes.

3.1 Tool Pipeline Overview

Our toolchain[2] for generating, analyzing, and evaluating driving data is sketched in Figure 1. Recorded driving data of simulation runs is generated by the CARLA simulator [DRC$^+$17]. Using the spatial properties of the actors (i.e., vehicles and pedestrians) during simulation and map data (for which CARLA relies on OpenDrive maps[3]), we generate two types of JSON files: a set of *driving data* files (one for each simulation run) and a set of *map data* files (one for each map). For each simulation run, those two types are then merged into a combined data format that enables the direct evaluation of temporal properties. Afterwards, the merged data is – based on the road the ego vehicle is driving on – segmented into *analysis segments* which are then ready to be classified as certain scenario classes.

[2] Our Kotlin framework, CARLA monitors, and experiments are available on GitHub: https://github.com/tudo-aqua/stars, https://github.com/tudo-aqua/stars-export-carla, and https://github.com/tudo-aqua/stars-carla-experiments
[3] https://www.asam.net/standards/detail/opendrive/

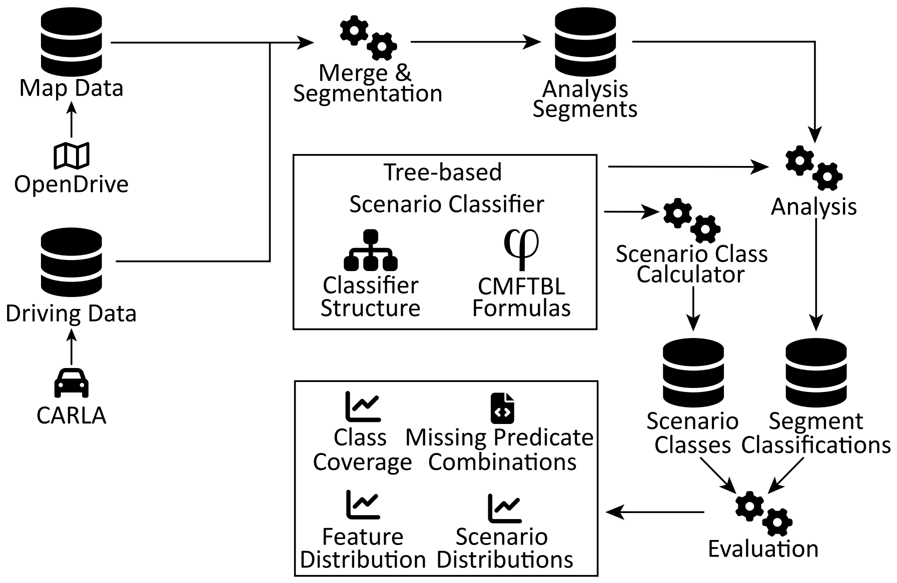

Figure 1: Overview of the tool pipeline for analyzing the coverage of test scenario sets[4]

To facilitate the definition of scenario classifiers and their analysis, we built a Kotlin library which allows for the definition of the classifier structure as well as their edge condition functions and monitor functions. For those functions, we also provide a library that allows us to express and evaluate CMFTBL formulas in Kotlin.

Given such a tree-based classifier, on the one hand, we calculate all possible scenario classes and on the other hand, each analysis segment is classified into one of those classes according to whether edge conditions are satisfied or not. In this case, we say that an analysis segment results in an *instance* of a scenario class. Based on the observed instances, a subsequent evaluation step calculates the class coverage statistics, identifies coverage gaps, as well as provides distributions of scenarios and individual features.

3.2 Data Exchange Format

In our toolchain, scenario classifiers' features are evaluated on an abstract representation of the world, i.e., the ego vehicle and its surroundings. We have *segments*, each of which holds a sequence of timestamps (called *ticks*) on which formulas expressed in CMFTBL are evaluated. Our primary concept is the notion of *actors* (e.g., vehicles and pedestrians) that move along *lanes* on *roads*. For each tick, we know the actors' positions on their lanes (measured in distance from the start of the lane), current velocities, etc., as well as additional static and dynamic information for the lanes, like speed limits on certain parts of a lane, yield priorities between lanes, current states of traffic lights, etc.

Figure 2 shows an excerpt from the data classes provided by our Kotlin library. Most of the structure should be self-explanatory, so we only highlight some important concepts: as in-

[4] Icons used in the figures are licensed under CC BY 4.0 https://fontawesome.com/license/free

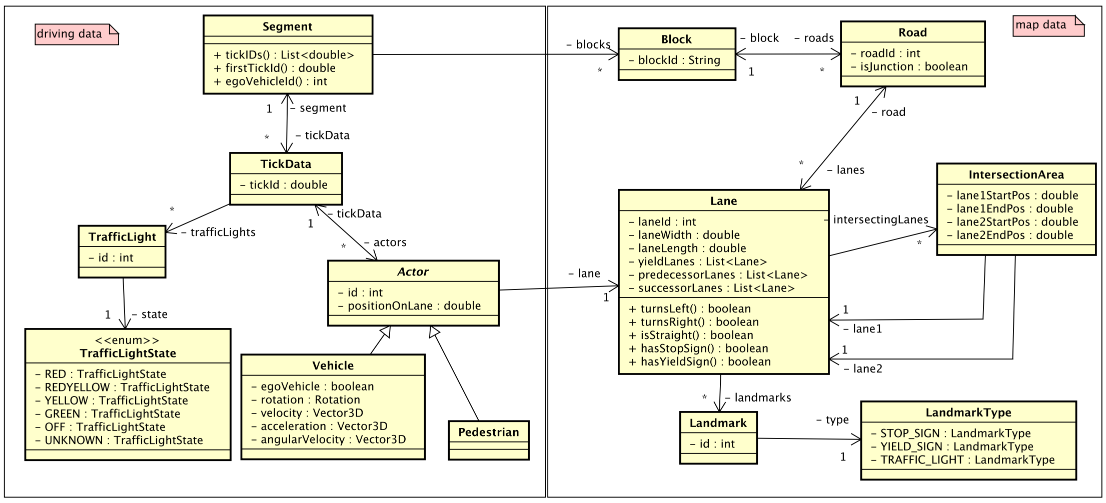

Figure 2: Abstract world representation used in our formulas for the TSC evaluations (excerpt)

troduced before, static and dynamic data is weaved together and segments are built by slicing simulation runs according to the road the ego vehicle drives on. Furthermore, an actor's position on the lane is measured from the start of the lane, i.e., `actor.positionOnLane` $\in [0,$ `actor.lane.laneLength`$]$. Each segment is classified by the TSC as one scenario class instance. The `tickId` is a timestamp measured in seconds. The list of `TickData` in `Segment` is strictly increasing with regards to the `tickId`. The static data is the same for all ticks in a segment. Traffic lights are modeled as landmarks belonging to a lane. Their current state (i.e., which lights are on) needs to be looked up in the dynamic data according to the `id`. Yield/stop signs and traffic lights are interpreted as belonging to (the end of) a lane before entering the junction where yielding must be obeyed.

We defined various helper functions (like `Lane.hasYieldSign()` to check if an according `Landmark` exists for this lane) to enable using those directly as conditions within our formulas. As most of them are simple and straightforward accessors on the data structure, Fig. 2 omits several of those functions for brevity.

This data structure is designed to potentially be constructed in various ways. Real-world test drives using sensors, GPS information, and high-definition maps are equally addressed as synthetic data from different simulation environments. Our evaluation methodology abstracts from the individual challenges one faces in providing such data – in particular from real-world test drives, which are already widely researched. We regard our generation of driving data based on the CARLA simulator as a proof-of-concept for our approach.

3.3 Definition of Tree-Based Scenario Classifiers

We provide an internal Kotlin DSL[5] to define the tree structure of a scenario classifier as well as a framework for expressing hierarchical predicates with CMFTBL formulas that are used in the classifier as condition and monitor functions. Please note that the developed Kotlin DSL is

```
/** pedestrian p is on the same lane as vehicle v, and v
* is driving towards p with a distance of < 10m **/
val inReach = predicate(Pedestrian::class to Vehicle::class)
{ ctx, p, v ->
  onSameLane.holds(ctx, p, v)
    && (p.positionOnLane - v.positionOnLane) in 0.0..10.0
}
/** true if at any one time stamp in the future there exists
* a pedestrian that crosses the lane right before v **/
val pedestrianCrossed = predicate(Vehicle::class)
{ ctx, v ->
  eventually(v) { v ->
    v.tickData.pedestrians.any { p ->
      inReach.holds(ctx, p, v)
    }
  }
}
```

Listing 1: Hierarchical predicate for crossing pedestrians

```
exclusive("Road Type") {
  all("Junction")        {
    condition = { ctx -> isInJunction.holds(ctx) }
    optional("Dynamic Relation") {
      leaf("Pedestrian Crossed") {
        condition = { ctx ->
          pedestrianCrossed.holds(ctx)
        }
      }
      leaf("Must Yield") {
        condition = { ctx ->
          ctx.vids.any { otherId ->
            mustYield.holds(ctx, actor2Id = otherId)
          }
        }
        monitorFunction = { ctx ->
          ctx.vids.any { otherId ->
            hasYielded.holds(ctx, actor2Id = otherId)
          }
        }
      }
    }
  }
}
```

Listing 2: TSC definition excerpt with our Kotlin DSL

currently only used for implementation means and is not intended as a proposal related to RQ 3.

Listing 1 shows an example predicate that checks whether eventually, a pedestrian crosses the vehicle's lane within a distance of 10.0 meters in front of the vehicle. The hierarchical nature of the predicates can be seen with `inReach` being used within `pedestrianCrossed`. Also, the predicate `onSameLane` is used in `inReach`, but omitted in the code excerpt for brevity. The function `predicate` is a builder provided by our framework to create predicates from a lambda function that implements its evaluation based on actor states that change over time.

Listing 2 shows an excerpt from a TSC definition containing (among others) a node for *Pedestrian Crossed* that makes use of the predicate previously discussed. For a given predicate evaluation context (`ctx`), which is provided by the framework during analysis and contains the currently analyzed segment of tick data, the predicate can be evaluated. The keywords `exclusive`, `all`, `optional`, and `leaf` define node types inspired by feature models and indicate how many children the evaluation condition holds in valid classes: exactly one, all children, any subset of children, and zero, respectively.

3.4 Scenario Class Calculator

For a given scenario classifier, to evaluate our metrics, we need to explicitly calculate all possible classes (i.e., valid sub-structures of the classifier tree) modeled by it. We have, so far, no results regarding the decidability of whether two CMFTBL formulas are mutually exclusive (i.e., intersection emptiness). We thus currently require by construction of the TSC that the evaluation conditions of a node's children only yield a valid amount of children concerning the node type (exclusive, all, optional, leaf). Given this restriction, we can calculate all valid classes modeled by a TSC by combinatorically building a scenario class for each subset of the TSC's nodes for

[5] The internal Kotlin DSL for TSC definition is realized as a type-safe builder. See https://kotlinlang.org/docs/type-safe-builders.html.

which the children's cardinalities induced by the node type are met. Actually, in our implementation, the node types introduced before are special cases of the more generic *bounded* node type which explicitly defines upper and lower bounds for the children of a TSC node when building valid classes.

3.5 Analysis of Segments

The analysis phase iterates through all segments and, for each, calculates a segment classification in form of a TSC instance according to whether the node conditions (written as CMFTBL formulas) hold. Instances of the same class are grouped, so that the subsequent evaluation can easily find missings, count occurrences, etc.

3.6 Evaluation of Segment Classifications

In the evaluation phase, the previously calculated classes and gathered segment classifications are processed. Being our main goal for the evaluation, the coverage for the TSC can now simply be calculated as the ratio between the number of observed classes and the number of possible classes.

Additionally, we gather various further insights from the collected data:

- **Class coverage over time:** We actually collect if a new scenario class has been observed for each analyzed segment, which provides us with a curve on coverage increase over time.
- **Class instance missings:** As we know the set of all possible scenario classes, we can exactly determine which ones we did not observe in our test data.
- **Feature pair misses:** Gaining meaningful insights from large sets of instance missings is difficult. Therefore, we also pairwise analyze whether feature combinations of valid classes have been observed at all and output all combinations that were missed.
- **Violated monitors:** In addition to the information which combinations of environmental aspects have been observed, we check the adherence to traffic rules (such as speed limit violations or overrun stop sign) using the monitor functions of the TSC definition. In each TSC instance, the occurrence of monitor violations is recorded for each node.
- **Absolute feature occurrence:** This metric provides useful information on the rarity of the modeled environmental conditions. Naturally, gaining high coverage in TSCs with (potentially multiple combinations of) rare events requires an extremely high amount of test scenarios.
- **Distribution of instance counts:** As we collect all segment classifications, we can count for each class how often it was observed in the test data. This directly corresponds to the rarity of the individual events.

4 Preliminary Results

Using the framework described in Section 3, we formalized environmental conditions, rules, and requirements (that could be part of operational design domains) as a tree-based classifier and showed that high coverage is achievable using scenario-based testing. In the following, we summarize the results of the case study, which is described in detail in [SNKH23].

To construct a tree-based classifier for our case study, we evaluated the 6-layer approach of Scholtes et al. [SWT$^+$21]. The six layers are summarized as: Layer 1 - *Road Network and Traffic Guidance Objects*, Layer 2 - *Roadside Structures*, Layer 3 - *Temporary Modifications of L1 and L2*, Layer 4 - *Dynamic Objects*, Layer 5 - *Environmental Conditions* and Layer 6 - *Digital Information*. This resulted in the TSC displayed in Figure 3. Here, each path from the root to a leaf node represents a specific environmental condition. As described in Section 3.4, valid classes are calculated combinatorically, so that combining all nodes of the proposed TSC yields a large set of possible classes. However, some combinations of nodes (i.e., conditions) may never occur or are not of interest. We, therefore, split this TSC into several *projections* defining interesting node combinations. As the primary criterion for these projections, we chose the specific layers of the 6-layer approach. To fit all projections into one Figure, we introduce colored markings attached to nodes. Each classifier is marked by its own color. Full circles ● include this node and all child nodes recursively. Consequently, the *full TSC* classifier contains all nodes of the TSC in Figure 3. Half-filled circles ◐ only include the node they are attached to without recursively including child nodes automatically (i.e., if certain child nodes need to be included, they require a marking on their own). The classifier ◐ *layer (4)+5* therefore only includes *Weather*, *Traffic Density* and *Time of Day* and all their children. All edges define logical formulas as conditions on whether to include their target nodes. Formula φ in Figure 3, for example, is used to detect a lane change for a given vehicle v, by utilizing our binding operator. We bind the lane of vehicle v at the first evaluation time stamp to a new variable l. As the vehicle v progresses in time and might change its lane, we can compare its lane value to l to detect a lane change. The formula is defined as: $changedLane(v) := \downarrow_l^{v.lane} \left(\Diamond(l \neq v.lane)\right)$. As usual, $\top$ represents the boolean *true* constant, and unlabeled edges' conditions were implemented, but are not presented here in detail.

To obtain test scenario data for evaluating the TSCs from our case study, we generated driving data by recording CARLA's autonomous driving simulation vehicles. Figure 4 shows our results for the scenario class coverage for each classifier throughout analyzed segments. In our experiments, we generated driving data from 100 simulations of 5 minutes, each with 200 autonomous vehicles and 30 pedestrians. Subsequently treating each of the 200 vehicles as *the ego vehicle* and segmentation according to the road the ego drives on, this resulted in $113,767$ analyzable segments of at least 5 seconds in length. For our experiments, in addition to six classifiers that are based on the layers of information described in [SWT$^+$21], we define an additional classifier that is dedicated to pedestrians as well as a full combinatorial classifier containing all properties (i.e., ◐ *full TSC*). Analysis of the data behind Figure 4 shows that the classifier ◐ *layer 1+2* reaches full coverage after $12,233$ analyzed segments. The next closely fully covered classifier reaches a coverage of 97% after $59,409$ analyzed segments. The next three classifiers reach coverages of 90%, 72%, and 48%, respectively, and show a similar flattening curve as the others. As already discussed above, the full combination of all TSC nodes creates a large set of classes. The feature

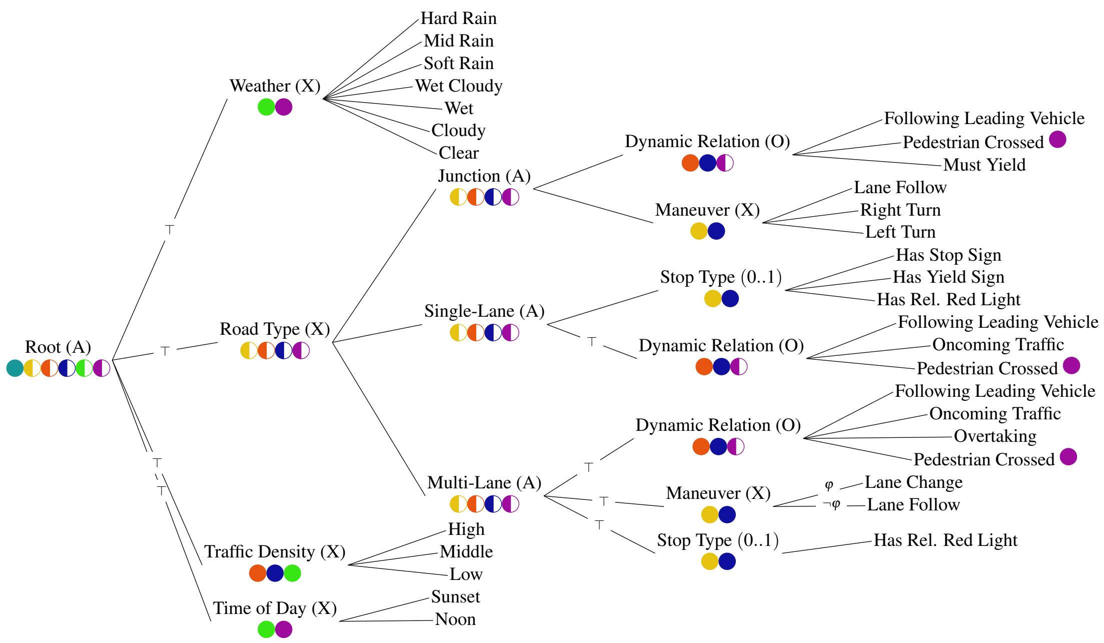

Figure 3: This figure shows the full classifier structure used for the analyses in [SNKH23]. Hereby, edge labels reference the related logical formula that is deciding whether an edge is taken. We use the following classifiers:
◐ *full TSC*, ◐ *layer 1+2*, ◐ *layer 4*, ◐ *layer 1+2+4*, ◐ *layer (4)+5* and ◐ *pedestrian*.

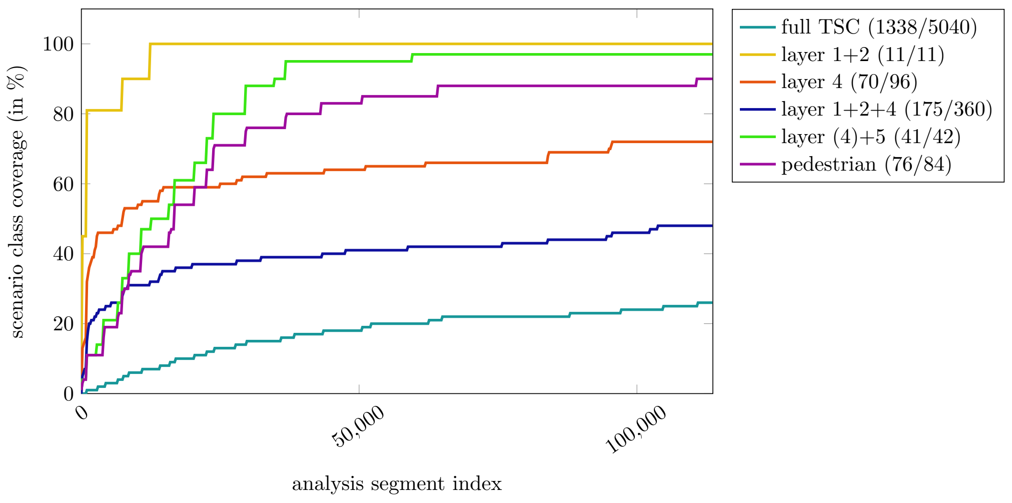

Figure 4: Scenario class coverage over the course of the 113,767 scene sequences. Figure taken from [SNKH23]

combinations that are not possible have an exponential impact on the missed classes. Thus, with 1.338 observed of 5.040 possible classes, the ◐ *full TSC* projection can only achieve a coverage of 26%.

These results have shown that our formal logic can be used to formalize relevant properties of common driving situations. Furthermore, we have shown that for sensibly constructed scenario classifiers, a high scenario class coverage is achievable. Additionally implemented analysis tools allowed us to analyze the evaluation results regarding property combination occurrences and overall missing property occurrences.

Considering our research questions, we are confident that it is possible to model environmental requirements, conditions, and traffic rules using formal logic. Furthermore, we have shown that scenario-based testing is capable of validating structured combinations of conditions which are formally defined. These two factors are already partly answering RQ 1 and RQ 2. Which parts are missing and how we aim to solve them, is outlined in the following section.

5 Next steps

With the developed framework at hand, we are capable of analyzing driving data which is generated by the CARLA simulator. Automatically generated driving data can be useful for a proof-of-concept evaluation. Nevertheless, the generated data is currently limited to urban traffic and we still have to consider interstate traffic and additional urban cases (e.g., support for additional traffic signs, recognition of jaywalking, etc.). Besides missing traffic scenarios, our framework is only evaluated on automatically generated driving data. Figure 5 shows some planned extensions to the landscape of our experimental setup to further improve and validate our framework. We plan to include support for additional map formats, like Lanelet 2 [PPJ$^+$18] or OpenStreet Map [Ben10]. Supporting these formats also allows us to analyze existing datasets of test scenarios, such as the INTERACTION dataset [ZSW$^+$19], which contains real-world driving data, or the CommonRoad scenario set [AKM17], a collection of several thousand handcrafted scenarios. By analyzing such datasets, we can improve our framework and yield insights into the datasets that might be useful for its users, as we are capable of automatically labeling the included scenarios by our classifiers. Generating driving data is done easily with simulators, but our framework should eventually be evaluated and tested on real-world driving data. In contrast to generated and simulation-based driving data, real-world driving data is fuzzy, as the installed sensors usually do not yield optimal measurements, or neural network-based object recognition might fail. It is important to test the automatic analysis of formally defined environmental conditions on imperfect data to be able to reason about the limits of the approach.

Currently, we have not started developing solutions regarding RQ 3. We plan to develop a specialized domain-specific language that is suitable for domain experts and uses CMFTBL formulas as its generation target. We envision including our DSL as a plugin to the *stiEF* methodology [BSH$^+$19] which is based on JetBrains MPS[6]. It aims at creating an iterative workflow for multilingual scenario descriptions and their generation into usable scenario definitions with concrete simulation parameters. After developing and deploying the DSL, we plan to research scenario synthesis to close the loop, as displayed in Figure 5: Based on the analysis of scenar-

[6] https://www.jetbrains.com/mps/

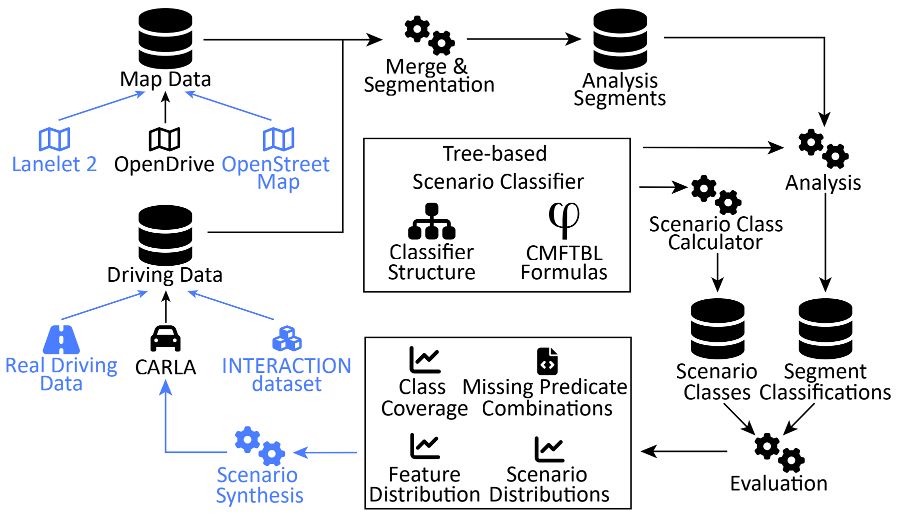

Figure 5: Overview of the next additions to the framework of Figure 1 marked in blue[7]

ios, we are capable of calculating missing combinations of conditions and can reason about the observed infrastructure and environmental conditions of the analyzed set of scenarios. With this information, it might be possible to automatically synthesize new scenario definitions, which can then be simulated using the CARLA simulator. One goal would be to create such a workflow loop to increase the diversity and expressiveness of sets of scenarios.

The last missing step in fully answering our research questions is the combination of the knowledge gained from RQ 2 and RQ 3 to test our framework with predefined operational design domains. To achieve this, we plan to adapt ODDs from official regulations, or use already defined ODDs, such as those defined by the European Union [The22].

6 Related Work

This section provides an overview of related works that pertain to our approach, including formalizing traffic scenarios, scenario-based testing, real-world data analysis, as well as coverages and metrics.

Formalizing traffic scenarios. In previous research, various formal logics have been used to define specific scenario rules for traffic regulations. For instance, Esterle et al. utilized Linear Temporal Logic to formalize traffic rules for highway situations [EGK20]. Rizaldi et al. [RKH+17] also employed Linear Temporal Logic to formalize German overtaking rules and provided verified checkers that can determine if a specific trace satisfies the defined Linear Temporal Logic formulas. Other researchers have utilized Metric Temporal Logic to formalize similar traffic rules, such as for interstate [MRMA20] and intersection traffic [MMA22]. These formalizations include traffic rules for general traffic, such as *safe distance to preceding vehicles*, *unnecessary braking*, *maximum speed limits*, and *traffic flow*. Additionally, Karimi and Duggirala described

[7] Icons used in the figures are licensed under CC BY 4.0 https://fontawesome.com/license/free

traffic rules for uncontrolled intersections using Answer Set Programming [KD20]. Their rules specify the expected behavior of traffic participants concerning right-of-way at unprotected intersections, which can be challenging as there are no yield or stop signs.

Scenario-based Testing. Recent studies have focused on scenario-based safety assurance for autonomous vehicles, primarily through testing. Scenario-based testing spans multiple different research aspects, such as defining [WBK+19], specifying, instantiating, executing [DRC+17], and generation of scenarios [RKKS17], as well as mining scenarios from real-world data, test automation [SZZ+22], creating similarity notions between scenarios [ZFYZ21], and identifying critical test scenarios [ANBS18]. To support these aspects, Steimle et al. [SMM21] have provided a taxonomy and definitions of terms for scenario-based development and testing. Additionally, Ulbrich et al. [UMR+15] defined a scenario as a sequence of scenes, with a scene representing a snapshot of a vehicle's environment. Klischat and Althoff [KA19] generated critical tests that can be used for regression testing.

Generally, scenarios can be categorized as *functional*, *logical*, and *concrete* [MBM18]. Functional scenarios refer to the description of entities in a domain and their semantic relations, while logical scenarios represent entities and their relationships using parameter ranges. Concrete scenarios, on the other hand, are specific instances of tempo-spatial structures with fixed parameters. These concepts are broadly accepted in industry and academia and serve as a framework for defining objectives and challenges. Menzel et al. use these concepts to transform a keyword-based scenario description into simulation formats [MBI+19]. Elster et al. utilize sensor model knowledge for their definitions of logical scenarios [ELR+21].

Zhang et al. [ZWZZ20] and Medrano-Berumen and Akbas [MA19] have addressed the generation of scenarios from semantic primitives and the development of appropriate primitives. The former generates collision-free traffic scenarios by employing extracted traffic primitives to describe road shapes, which can be used to create additional scenarios. Similarly, the latter generates roadways by connecting building blocks, i.e., geometric primitives.

Analyzing Real-World Data. Scenario-based testing is widely accepted for testing autonomous systems, but it is important to complement it with evaluations on real-world data [WLFM18]. Real-world data can help to identify critical traffic scenarios that can be used to develop scenarios for scenario-based testing [BOS16] and provide the necessary parameter ranges [MBM18] . Additionally, real-world data can assist in understanding how human drivers perceive autonomous system failures in real-world situations [DB16]. Real-world data also has other applications, such as decision-making [FV11], pedestrian intention estimation [AMS+20], and object detection [LCW+22]. However, simulation data should always be used alongside real-world data, as it can model situations that may not be feasible in the real world, such as accidents [LBR+18].

Coverage and Metrics. When it comes to scenario-based testing, a notable challenge emerges in effectively attaining coverage and ensuring the relevance of a scenario set [DMPR18]. Laurent et al. [LKA+22] propose a coverage criterion for assessing the sufficiency of a test set by considering the parameters used in the decision-making process of autonomous systems. Langner et al. automatically detect novel traffic scenarios using a machine-learning approach [LBR+18]. Using this, they can reduce a given test set to unique test scenarios. A test-ending criterion aimed at establishing the safety of automated and autonomous driving systems is proposed by Hauer et al. [HSHP19].

Closest to ours are the following two works: Amersbach and Winner proposed an approach to calculate the necessary number of concrete scenarios based on the given parameter ranges to achieve scenario coverage. They emphasized the importance of developing a specification of functional scenarios, such as lane change and following, for validating highly automated vehicles [AW19].

Li et al. [LCS+22] present an approach for generating abstract scenarios to maximize the coverage of k-way combinatorial testing. The resulting abstract scenarios can be regarded as equivalence classes, each of which is associated with a set of concrete scenarios. The categories used to generate the abstract scenarios, such as weather, road type, and ego-action, are similar to the scenario classifiers employed in our previous work [SNKH23].

7 Conclusion

In this paper, we analyzed the current state of assuring the safety of autonomous vehicles, listed current open problems, and used these to derive three research questions. Following these questions, we introduced our framework to model and analyze classes of scenarios with a tree-based classifier. The framework automatically calculates scenario class coverage, feature occurrences, violated monitors, and distributions of these metrics for scenario-based test sets using temporal logic formulas that are based on a set of behavioral requirements, conditions, and rules. We were able to show that using simulation-based generation of test scenarios, high scenario class coverage is achievable. The development of our framework and its results already contribute to the first two research questions. Nevertheless, open questions are still remaining, especially regarding the evaluation of the potential of our proposed framework. Finally, we outlined our next steps towards fully answering the remaining parts of the research questions and proposed a workflow which allows for automatic generation of test scenarios to fulfill some coverage criterion.

Bibliography

[AKM17] M. Althoff, M. Koschi, S. Manzinger. CommonRoad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*. Pp. 719–726. IEEE, June 2017.
doi:10.1109/ivs.2017.7995802

[AMS+20] W. M. Alvarez, F. M. Moreno, O. Sipele, N. Smirnov, C. Olaverri-Monreal. Autonomous Driving: Framework for Pedestrian Intention Estimation in a Real World Scenario. In *2020 IEEE Intelligent Vehicles Symposium (IV)*. Pp. 39–44. IEEE, Oct. 2020.
doi:10.1109/iv47402.2020.9304624

[ANBS18] R. B. Abdessalem, S. Nejati, L. C. Briand, T. Stifter. Testing vision-based control systems using learnable evolutionary algorithms. In *Proceedings of the 40th International Conference on Software Engineering*. ICSE ’18, pp. 1016–1026. ACM, May 2018.
doi:10.1145/3180155.3180160

[AW19] C. Amersbach, H. Winner. Defining Required and Feasible Test Coverage for Scenario-Based Validation of Highly Automated Vehicles. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Pp. 425–430. IEEE, Oct. 2019. doi:10.1109/itsc.2019.8917534

[Ben10] J. Bennett. *OpenStreetMap*. Packt Publishing Ltd, 2010.

[BOS16] J. Bach, S. Otten, E. Sax. Model based scenario specification for development and test of automated driving functions. In *2016 IEEE Intelligent Vehicles Symposium (IV)*. Pp. 1149–1155. IEEE, June 2016. doi:10.1109/ivs.2016.7535534

[BSH+19] F. Bock, C. Sippl, A. Heinz, C. Lauer, R. German. Advantageous Usage of Textual Domain-Specific Languages for Scenario-Driven Development of Automated Driving Functions. In *2019 IEEE International Systems Conference (SysCon)*. Pp. 1–8. IEEE, Apr. 2019. doi:10.1109/syscon.2019.8836912

[BZMS16] M. Bunting, Y. Zeleke, K. McKeever, J. Sprinkle. A Safe Autonomous Vehicle Trajectory Domain Specific Modeling Language for Non-Expert Development. In *Proceedings of the International Workshop on Domain-Specific Modeling*. DSM 2016, pp. 42–48. ACM, Oct. 2016. doi:10.1145/3023147.3023154

[Cen20] Centre for Connected & Autonomous Vehicles. Operational Design Domain (ODD) taxonomy for an automated driving system (ADS) - Specification. Specification PAS 1883:2020, The British Standards Institution, Aug. 2020. https://www.bsigroup.com/globalassets/localfiles/en-gb/cav/pas1883.pdf

[Cho95] J. Chomicki. Efficient checking of temporal integrity constraints using bounded history encoding. *ACM Transactions on Database Systems* 20(2):149–186, June 1995. doi:10.1145/210197.210200

[DB16] M. Dikmen, C. M. Burns. Autonomous Driving in the Real World: Experiences with Tesla Autopilot and Summon. In *Proceedings of the 8th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. Automotive'UI 16, pp. 225–228. ACM, Oct. 2016. doi:10.1145/3003715.3005465

[DMPR18] W. Damm, E. Möhlmann, T. Peikenkamp, A. Rakow. A Formal Semantics for Traffic Sequence Charts. In *Principles of Modeling*. Lecture Notes in Computer Science, pp. 182–205. Springer International Publishing, 2018. doi:10.1007/978-3-319-95246-8_11

[DRC+17] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*. Proceedings of Machine Learning Research 78, pp. 1–16. PMLR, Oct.

2017.
https://proceedings.mlr.press/v78/dosovitskiy17a.html

[EGK20] K. Esterle, L. Gressenbuch, A. Knoll. Formalizing Traffic Rules for Machine Interpretability. In *2020 IEEE 3rd Connected and Automated Vehicles Symposium (CAVS)*. Pp. 1–7. Nov. 2020.
doi:10.1109/CAVS51000.2020.9334599

[ELR$^+$21] L. Elster, C. Linnhoff, P. Rosenberger, S. Schmidt, R. Stark, H. Winner. Fundamental Design Criteria for Logical Scenarios in Simulation-based Safety Validation of Automated Driving Using Sensor Model Knowledge. In *2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*. Pp. 209–214. IEEE, July 2021.
doi:10.1109/ivworkshops54471.2021.9669207

[FFS$^+$23] F. Favaro, L. Fraade-Blanar, S. Schnelle, T. Victor, M. Peña, J. Engstrom, J. Scanlon, K. Kusano, D. Smith. Building a Credible Case for Safety: Waymo's Approach for the Determination of Absence of Unreasonable Risk. Technical report, arXiv, June 2023.
doi:10.48550/arXiv.2306.01917

[FKL$^+$19] H. Felbinger, F. Kluck, Y. Li, M. Nica, J. Tao, F. Wotawa, M. Zimmermann. Comparing two systematic approaches for testing automated driving functions. In *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*. Pp. 1–6. IEEE, Nov. 2019.
doi:10.1109/iccve45908.2019.8965209

[FV11] A. Furda, L. Vlacic. Enabling Safe Autonomous Driving in Real-World City Traffic Using Multiple Criteria Decision Making. *IEEE Intelligent Transportation Systems Magazine* 3(1):4–17, Apr. 2011.
doi:10.1109/mits.2011.940472

[HSHP19] F. Hauer, T. Schmidt, B. Holzmüller, A. Pretschner. Did We Test All Scenarios for Automated and Autonomous Driving Systems? In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Pp. 2950–2955. IEEE, Oct. 2019.
doi:10.1109/itsc.2019.8917326

[ISO18] ISO Central Secretary. Road vehicles - Functional safety - Part 1: Vocabulary. Standard ISO 26262-1:2018, International Organization for Standardization, Dec. 2018.
https://www.iso.org/standard/68383.html

[ISO22] ISO Central Secretary. Road vehicles - Safety of the intended functionality. Standard ISO 21448:2022, International Organization for Standardization, June 2022.
https://www.iso.org/standard/77490.html

[JWKW18] P. Junietz, W. Wachenfeld, K. Klonecki, H. Winner. Evaluation of Different Approaches to Address Safety Validation of Automated Driving. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Pp. 491–496.

IEEE, Nov. 2018.
doi:10.1109/itsc.2018.8569959

[KA19] M. Klischat, M. Althoff. Generating Critical Test Scenarios for Automated Vehicles with Evolutionary Algorithms. In *2019 IEEE Intelligent Vehicles Symposium (IV)*. Pp. 2352–2358. IEEE, June 2019.
doi:10.1109/ivs.2019.8814230

[KD20] A. Karimi, P. S. Duggirala. Formalizing traffic rules for uncontrolled intersections. In *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*. Pp. 41–50. IEEE, Apr. 2020.
doi:10.1109/iccps48487.2020.00012

[KKHK19] L. Klitzke, C. Koch, A. Haja, F. Köster. Real-world Test Drive Vehicle Data Management System for Validation of Automated Driving Systems. In *Proceedings of the 5th International Conference on Vehicle Technology and Intelligent Transport Systems - VEHITS*. Pp. 171–180. SciTePress, May 2019.
doi:10.5220/0007720501710180

[KP16] N. Kalra, S. 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, Dec. 2016.
doi:10.1016/j.tra.2016.09.010

[LBR+18] J. Langner, J. Bach, L. Ries, S. Otten, M. Holzapfel, E. Sax. Estimating the Uniqueness of Test Scenarios derived from Recorded Real-World-Driving-Data using Autoencoders. In *2018 IEEE Intelligent Vehicles Symposium (IV)*. Pp. 1860–1866. IEEE, June 2018.
doi:10.1109/ivs.2018.8500464

[LCS+22] C. Li, C.-H. Cheng, T. Sun, Y. Chen, R. Yan. ComOpT: Combination and Optimization for Testing Autonomous Driving Systems. In *2022 International Conference on Robotics and Automation (ICRA)*. Pp. 7738–7744. IEEE, May 2022.
doi:10.1109/icra46639.2022.9811794

[LCW+22] K. Li, K. Chen, H. Wang, L. Hong, C. Ye, J. Han, Y. Chen, W. Zhang, C. Xu, D.-Y. Yeung, X. Liang, Z. Li, H. Xu. CODA: A Real-World Road Corner Case Dataset for Object Detection in Autonomous Driving. In *Computer Vision - ECCV 2022*. Lecture Notes in Computer Science, pp. 406–423. Springer Nature Switzerland, Oct. 2022.
doi:10.1007/978-3-031-19839-7_24

[LKA+22] T. Laurent, S. Klikovits, P. Arcaini, F. Ishikawa, A. Ventresque. Parameter Coverage for Testing of Autonomous Driving Systems Under Uncertainty. *ACM Transactions on Software Engineering and Methodology* 32(3):58:1–58:31, Apr. 2022.
doi:10.1145/3550270

[MA19] C. Medrano-Berumen, M. I. Akbas. Abstract Simulation Scenario Generation for Autonomous Vehicle Verification. In *2019 SoutheastCon*. Pp. 1–6. IEEE, Apr. 2019. doi:10.1109/southeastcon42311.2019.9020575

[Mar18] R. Mariani. An overview of autonomous vehicles safety. In *2018 IEEE International Reliability Physics Symposium (IRPS)*. Pp. 6A.1–1–6A.1–6. IEEE, Mar. 2018. doi:10.1109/irps.2018.8353618

[MBI+19] T. Menzel, G. Bagschik, L. Isensee, A. Schomburg, M. Maurer. From Functional to Logical Scenarios: Detailing a Keyword-Based Scenario Description for Execution in a Simulation Environment. In *2019 IEEE Intelligent Vehicles Symposium (IV)*. Pp. 2383–2390. IEEE, June 2019. doi:10.1109/ivs.2019.8814099

[MBM18] T. Menzel, G. Bagschik, M. Maurer. Scenarios for Development, Test and Validation of Automated Vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*. Pp. 1821–1827. IEEE, June 2018. doi:10.1109/IVS.2018.8500406

[MHR16] M. Mauritz, F. Howar, A. Rausch. Assuring the Safety of Advanced Driver Assistance Systems Through a Combination of Simulation and Runtime Monitoring. In *Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications*. Lecture Notes in Computer Science, pp. 672–687. Springer International Publishing, Oct. 2016. doi:10.1007/978-3-319-47169-3_52

[MMA22] S. Maierhofer, P. Moosbrugger, M. Althoff. Formalization of Intersection Traffic Rules in Temporal Logic. In *2022 IEEE Intelligent Vehicles Symposium (IV)*. Pp. 1135–1144. IEEE, June 2022. doi:10.1109/iv51971.2022.9827153

[MRMA20] S. Maierhofer, A.-K. Rettinger, E. C. Mayer, M. Althoff. Formalization of Interstate Traffic Rules in Temporal Logic. In *2020 IEEE Intelligent Vehicles Symposium (IV)*. Pp. 752–759. IEEE, Oct. 2020. doi:10.1109/iv47402.2020.9304549

[Mül09] S. Müller. *Theory and Applications of Runtime Monitoring Metric First-order Temporal Logic*. PhD thesis, ETH Zurich, 2009. doi:10.3929/ethz-a-005932651

[NMB+20] D. Nalic, T. Mihalj, M. Bäumler, M. Lehmann, A. Eichberger, S. Bernsteiner. Scenario Based Testing of Automated Driving Systems: A Literature Survey. In *Proceedings of the FISITA Web Congress 2020*. Oct. 2020. https://www.fisita.com/library/fisita-world-congress/2020/f2020-acm-096

[PPJ+18] F. Poggenhans, J.-H. Pauls, J. Janosovits, S. Orf, M. Naumann, F. Kuhnt, M. Mayr. Lanelet2: A high-definition map framework for the future of automated driving. In

2018 21st International Conference on Intelligent Transportation Systems (ITSC). Pp. 1672–1679. IEEE, Nov. 2018.
doi:10.1109/itsc.2018.8569929

[RKH+17] A. Rizaldi, J. Keinholz, M. Huber, J. Feldle, F. Immler, M. Althoff, E. Hilgendorf, T. Nipkow. Formalising and Monitoring Traffic Rules for Autonomous Vehicles in Isabelle/HOL. In *Integrated Formal Methods*. Lecture Notes in Computer Science, pp. 50–66. Springer International Publishing, Aug. 2017.
doi:10.1007/978-3-319-66845-1_4

[RKKS17] E. Rocklage, H. Kraft, A. Karatas, J. Seewig. Automated scenario generation for regression testing of autonomous vehicles. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. Pp. 476–483. IEEE, Oct. 2017.
doi:10.1109/itsc.2017.8317919

[SHFG20] A. K. Saberi, J. Hegge, T. Fruehling, J. F. Groote. Beyond SOTIF: Black Swans and Formal Methods. In *2020 IEEE International Systems Conference (SysCon)*. Pp. 1–5. IEEE, Aug. 2020.
doi:10.1109/SysCon47679.2020.9275888

[SHT06] P.-Y. Schobbens, P. Heymans, J.-C. Trigaux. Feature Diagrams: A Survey and a Formal Semantics. In *14th IEEE International Requirements Engineering Conference (RE'06)*. Pp. 139–148. IEEE, Sept. 2006.
doi:10.1109/re.2006.23

[SMM21] M. Steimle, T. Menzel, M. Maurer. Toward a Consistent Taxonomy for Scenario-Based Development and Test Approaches for Automated Vehicles: A Proposal for a Structuring Framework, a Basic Vocabulary, and Its Application. *IEEE Access* 9:147828–147854, Oct. 2021.
doi:10.1109/access.2021.3123504

[SNKH23] T. Schallau, S. Naujokat, F. Kullmann, F. Howar. Tree-Based Scenario Classification: A Formal Framework for Coverage Analysis on Test Drives of Autonomous Vehicles. *arXiv preprint*, July 2023.
doi:10.48550/arXiv.2307.05106

[SSLM13] F. Schuldt, F. Saust, B. Lichte, M. Maurer. Effiziente systematische Testgenerierung für Fahrerassistenzsysteme in virtuellen Umgebungen. Technical report, Institut für Regelungstechnik, TU Braunschweig, 2013.
doi:10.24355/dbbs.084-201307101421-0

[SWT+21] M. Scholtes, L. Westhofen, L. R. Turner, K. Lotto, M. Schuldes, H. Weber, N. Wagener, C. Neurohr, M. H. Bollmann, F. Kortke, J. Hiller, M. Hoss, J. Bock, L. Eckstein. 6-Layer Model for a Structured Description and Categorization of Urban Traffic and Environment. *IEEE Access* 9:59131–59147, 2021.
doi:10.1109/access.2021.3072739

[SZZ+22] J. Sun, H. Zhang, H. Zhou, R. Yu, Y. Tian. Scenario-Based Test Automation for Highly Automated Vehicles: A Review and Paving the Way for Systematic Safety Assurance. *IEEE Transactions on Intelligent Transportation Systems* 23(9):14088–14103, Sept. 2022.
doi:10.1109/tits.2021.3136353

[The22] The European Commission. Commission Implementing Regulation (EU) 2022/1426 of 5 August 2022 laying down rules for the application of Regulation (EU) 2019/2144 of the European Parliament and of the Council as regards uniform procedures and technical specifications for the type-approval of the automated driving system (ADS) of fully automated vehicles. *Offical Journal of the European Union* 65(L 221):1–64, Aug. 2022.
http://data.europa.eu/eli/reg_impl/2022/1426/oj

[UMR+15] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, M. Maurer. Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. Pp. 982–988. IEEE, Sept. 2015.
doi:10.1109/itsc.2015.164

[WBK+19] H. Weber, J. Bock, J. Klimke, C. Roesener, J. Hiller, R. Krajewski, A. Zlocki, L. Eckstein. A framework for definition of logical scenarios for safety assurance of automated driving. *Traffic Injury Prevention* 20(sup1):S65–S70, June 2019.
doi:10.1080/15389588.2019.1630827

[WLFM18] H. Winner, K. Lemmer, T. Form, J. Mazzega. PEGASUS—First Steps for the Safe Introduction of Automated Driving. In *Road Vehicle Automation 5*. Lecture Notes in Mobility, pp. 185–195. Springer International Publishing, June 2018.
doi:10.1007/978-3-319-94896-6_16

[ZFYZ21] J. Zhao, J. Fang, Z. Ye, L. Zhang. Large Scale Autonomous Driving Scenarios Clustering with Self-supervised Feature Extraction. In *2021 IEEE Intelligent Vehicles Symposium (IV)*. Pp. 473–480. IEEE, July 2021.
doi:10.1109/iv48863.2021.9575644

[ZSW+19] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kummerle, H. Konigshof, C. Stiller, A. de La Fortelle, M. Tomizuka. INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps. Technical report, arXiv, Sept. 2019.
doi:10.48550/arXiv.1910.03088

[ZWZZ20] W. Zhang, W. Wang, J. Zhu, D. Zhao. Multi-Vehicle Interaction Scenarios Generation with Interpretable Traffic Primitives and Gaussian Process Regression. In *2020 IEEE Intelligent Vehicles Symposium (IV)*. Pp. 1197–1204. IEEE, Oct. 2020.
doi:10.1109/iv47402.2020.9304568