# Tree-Based Scenario Classification: A Formal Framework for Coverage Analysis on Test Drives of Autonomous Vehicles

Till Schallau, Stefan Naujokat, Fiona Kullmann, and Falk Howar

***Abstract*—Scenario-based testing is envisioned as a key approach for the safety assurance of autonomous vehicles. In scenario-based testing, relevant (driving) scenarios are the basis of tests. Many recent works focus on specification, variation, generation, and execution of individual scenarios. In this work, we address the open challenges of classifying sets of scenarios and measuring coverage of these scenarios in recorded test drives. Technically, we define logic-based classifiers that compute features of scenarios on complex data streams and combine these classifiers into feature trees that describe sets of scenarios. We demonstrate the expressiveness and effectiveness of our approach by defining a scenario classifier for urban driving and evaluating it on data recorded from simulations.**

***Index Terms*—temporal logic, metric, scenario classification, scenario-based testing, autonomous vehicles**

## I. Introduction

One of the open challenges in the development of autonomous driving software is assuring its safety [1]. It has long been established that statistical arguments on the performance of the complete system (e.g., caused fatalities per million miles) are not attainable in practice [2], [3]. The billions of miles that would have to be driven without failures are simply not feasible for every new vehicle or software update. For several years now, the focus of research has been on structured approaches to assuring the safety of autonomous driving functions instead [4], [5].

The recently published ISO 21448 [6] norm (Safety of the Intended Functionality) transfers the conceptual framework of system safety approaches (e.g., ISO 26262 [7]) to the assurance of a vehicle's safety under all environmental conditions and possible faults that are triggered by the environment [8]. Basically, the idea is to identify relevant driving situations and potential triggers and then use these as a basis for testing the safety of a vehicle or its driving software. Many recent works focus on defining notions of safety [9], [10], formalizing what constitutes scenarios [11]–[14], and on testing safety in specified scenarios [15].

Recent standardization efforts target the specification of so-called operational design domains (ODDs) [16] that define the anticipated environmental conditions for autonomous vehicles at a high level (e.g., weather conditions, road types and parameters, etc.). To combine works and results on testing individual scenarios into compelling arguments about the safety of an autonomous vehicle in its operational design domain, we need tools for describing sets of relevant scenarios in some ODD and methods for analyzing coverage of these scenarios in driving tests as, e.g., stated in the ASAM OpenODD concept [17].

In this paper, we present an approach to the specification of sets of scenarios through classifiers for features of scenarios in the set. These classifiers can then be used to identify observed scenarios in recorded test drives. Moreover, we can compute the set of possible combinations of features from our specification. This enables us to provide coverage metrics and to identify counterfactual scenarios, i.e., scenarios that were not observed but could be observed. Technically, we use logic-based classifiers that identify features of scenarios on complex data streams and combine these classifiers into feature trees that describe sets of scenarios, emerging from the combinatorial combination of features. We extend an existing modal logic to express features that can be found in ODD standards, in the 6-layer model of driving scenarios [12], [14], and in the classification of driving maneuvers (e.g., intersection, traffic light present, light rain, oncoming traffic, left-turn maneuver, etc.).

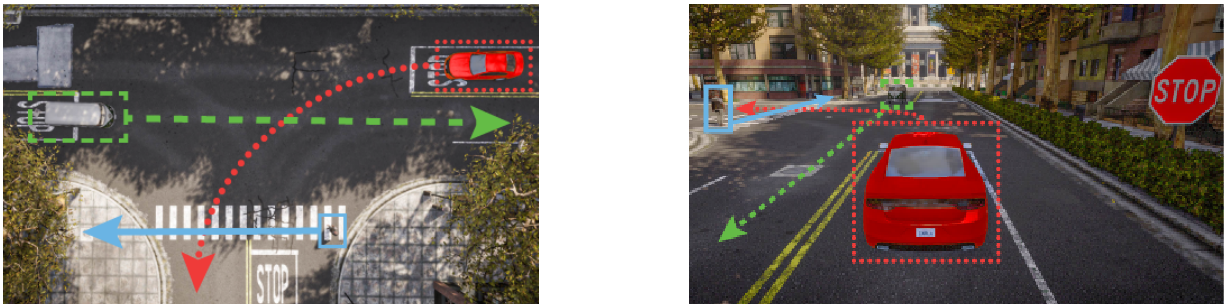
We demonstrate the expressiveness and effectiveness of our approach in a case study: We specify a small set of features and use test drives in a randomized simulation to analyze the observed scenario classes and the coverage that can be achieved in this setup. We also show how coverage can be decomposed and only analyzed for individual features or layers of the 6-layer model.

Summarizing, the contribution of this paper is threefold:

1) We present a formal logic for describing properties over recorded sequences of scenes. The logic extends upon existing temporal logics in multiple aspects that are essential for concise specifications that work on field-recorded data: firstly, the logic allows fuzzy specifications (in the spirit of: "most of the time"); secondly, it is defined over complex structured domains for capturing scenes (cf. Sect. III).
2) We present a method for classifying sets of scenarios that is conceptually inspired by recent works and standardization efforts around operational design domains (ODDs) and technically inspired by feature models [19], where features of scenarios are specified using the presented logic over sequences of scenes. To the best of our knowledge, this

T. Schallau, S. Naujokat and F. Kullmann are with TU Dortmund University, Dortmund, Germany (e-mail: till.schallau@tu-dortmund.de; stefan.naujokat@tu-dortmund.de; fiona.kullmann@tu-dortmund.de)

F. Howar is with TU Dortmund University, Dortmund, Germany and is also with Fraunhofer ISST, Dortmund, Germany (e-mail: falk.howar@tu-dortmund.de)




(a) The traffic situation as seen from a birds eye view

(b) The traffic situation as seen from ego's perspective

Fig. 1: Screenshots taken in CARLA [18] showing a traffic situation in which the ego vehicle (red, dotted) stops at a stop sign. The trajectory of the planned left turn is crossed by an oncoming vehicle (green, dashed) and by a pedestrian (blue, solid) crossing the street.

is one of the first approaches that addresses classification and analyses on sets of different scenarios (cf. Sect. IV-A).

3) The specification of features and formal models for sets of scenarios enable several quantitative and qualitative analyses on recorded driving data, e.g., scenario coverage, missing scenarios, missing combination of features, and distribution of combinations of features. In contrast to other works, these metrics focus on sets of scenarios instead of on parameter ranges within one scenario (cf. Sect. IV-B).

The ultimate goal of this work is to get a handle on the task of specifying, selecting, and prioritizing relevant scenarios and representative combinations of environmental conditions across all scenarios.

**Outline.** The paper is structured as follows. Section II outlines an example that motivates our approach. The formal logic for defining scenario properties is introduced in Sect. III. Section IV then introduces the formalism for scenario classifier trees and the calculation of coverage metrics and analyses on such trees. The results of our case study are presented in Sect. V, which is followed by a discussion on related work in Sect. VI. The paper concludes in Sect. VII.

**Reproduction Package.** For the experiments conducted in our case study, a reproduction package is available on Zenodo [20][1].

## II. Motivational Example

We illustrate our approach for the task of analyzing test drives in an urban environment. We assume to have a database of recorded test drives. Recordings consist of sequences of *scenes* and are split into meaningful *segments*, e.g., based on regions of a map. A single scene is the snapshot of the state and observed environment of the ego vehicle, comprising map data, position and velocity of the ego vehicle, stationary objects, and moving objects around the ego vehicle. Segments (i.e., sequences of scenes) are recorded at fixed (e.g., $0.5$-second) intervals.

Our task is now to decide if this database contains test drives that cover enough relevant *scenarios* (i.e., archetypes of driving situations), or at least to identify and classify the encountered scenarios. A scenario, in this case, would be a basic driving task, like making an unprotected left turn on a three-way intersection, and it could have variants, (e.g., presence of oncoming traffic or pedestrians).

Figure 1 shows an example of a scene from a segment in which three road users are situated on a T-junction. The ego vehicle, which is marked by the red box, is planning to turn left. It is currently stopped at the stop line of the stop sign on the ego vehicle's lane. The car marked with the green box is following its lane, going straight over the junction and crossing the trajectory of the ego vehicle. The destination lane of ego contains a crosswalk on which a pedestrian, marked in solid blue, is currently crossing the road. The pedestrian is also crossing the trajectory of the ego vehicle.

When analyzing the segment, specific maneuvers, environmental properties, and features can be observed from the viewpoint of the ego vehicle: road type is *T-junction*; ego is *turning left*; there is *oncoming traffic*; a *stop sign* is present; the ego vehicle does *stop at the stop line*, since it *must yield* to another vehicle; a *pedestrian is crossing* the destination lane; the weather is *sunny* during *daytime*. These features can be formally described by formulas in a temporal first-order logic over sequences of scenes. A set of features can then be used to classify segments: the combination of features that hold defines the scenario class. The segment then is one concrete instance of this scenario class.

Assuming that features are not entailed by other features, we generate $2^n$ scenario classes with $n$ features. For the more realistic case that some dependencies exist between features (e.g., no overtaking without multiple lanes), we can use trees to model taxonomies of features and still compute possible scenario classes and check if they exist in our data. Possible variations of features in the example could be: *the ego vehicle drives straight instead of turning left*, *there is no oncoming traffic*, or *no pedestrian is crossing the road*. For the sake of simplicity, we neglect the other properties for the following calculation. Based on these three variations, a total of $2^3 = 8$ possible scenario classes are observable. We can use this information to compute missed scenario classes or to measure scenario class coverage for our database of test drives. In our example, one scenario class was observed. Given the eight possible scenario classes, we obtain a scenario class coverage

[1]Please note: an arXiv preprint of this article is also referenced there.

of 12.5%. The following two sections formalize these concepts.

## III. A Temporal Logic for Properties of Scenarios

We base our classifiers for scenarios on the environment representation that is usually produced by the perception subsystem of an autonomous vehicle: a map of the road network and typed objects with positions, velocities, and observed states. To express properties of recorded sequences of scenes (i.e., momentary snapshots of the environment of the ego vehicle), we need a formal logic that can express properties in individual scenes as well properties between objects in multiple different scenes. Examples are distances between objects in one scene, or the fraction of scenes in a sequence in which a leading vehicle is present. We introduce such a logic and then use it for defining classifier trees that express sets of scenarios in terms of features in the scenarios (cf. Sect. IV-A).

We use logic structures to describe scenes over a given signature of domain-specific functions and relations (e.g., positions, lanes, vehicles, velocities, etc.). We introduce *CMFTBL* (Counting Metric First-Order Temporal Binding Logic), a metric first-order logic for modeling time that extends *MFOTL* (Metric First-Order Temporal Logic) [21], [22], while focusing on finite traces of states. In particular, we extend MFOTL by a *minimum prevalence operator* that allows us to express that a property (or sub-formula) holds for a certain fraction of all future states (within the finite trace). We also introduce 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. While the former extends the expressiveness of MFOTL, the second one is a shorthand for existentially quantified formulas of a certain form.

A signature $\sigma$ is a tuple $\langle \mathcal{C}, \mathcal{F}, \mathcal{R}, \mathrm{ar}\rangle$, where $\mathcal{C}$ is a set of named constants, $\mathcal{F}$ is a set of function symbols, $\mathcal{R}$ is a set of relation symbols, and $\mathrm{ar} : (\mathcal{F} \cup \mathcal{R}) \mapsto \mathbb{N}_0$ defines an arity for each function symbol $f \in \mathcal{F}$ and relation symbol $r \in \mathcal{R}$. A $\sigma$-structure $\mathfrak{D}$ is a pair $\langle \mathcal{D}, I\rangle$ of a domain $\mathcal{D}$ and interpretations of constants, functions, and relations with $I(c) \in \mathcal{D}$ for $c \in \mathcal{C}$, $\mathrm{ar}(f)$-ary function $I(f) : \mathcal{D}^{\mathrm{ar}(f)} \rightarrow \mathcal{D}$ for $f \in \mathcal{F}$, and $I(r) \subseteq \mathcal{D}^{\mathrm{ar}(r)}$ for $r \in \mathcal{R}$.

An interval of the set of non-empty intervals $\mathcal{I}$ over $\mathbb{N}$ can be written as $[b, b') := \{a \in \mathbb{N} | b \leq a < b'\}$, where $b \in \mathbb{N}, b' \in \mathbb{N} \cup \{\infty\}$ and $b < b'$.

*CMFTBL formulas* over the signature $\sigma$, intervals $\mathcal{I}$, and the countably infinite set of variables $\mathcal{V}$ (assuming $\mathcal{V} \cap (\mathcal{C} \cup \mathcal{F} \cup \mathcal{R}) = \emptyset$) are inductively defined as follows:

(i) A *term* $t$ is either a constant $c$, a variable $v$, or for $f \in \mathcal{F}$ and terms $t_1, \cdots, t_{\mathrm{ar}(f)}$ the application $f(t_1, \cdots, t_{\mathrm{ar}(f)})$.

(ii) For $r \in \mathcal{R}$ and terms $t_1, \cdots, t_{\mathrm{ar}(r)}$, the predicate $r(t_1, \cdots, t_{\mathrm{ar}(r)})$ is a *formula*.

(iii) For $x \in \mathcal{V}$ and $d \in \mathcal{D}$, if $t$ is a term, $\varphi$ and $\psi$ are formulas, then $(\neg\varphi), (\varphi \vee \psi)$, $(\exists x : \varphi)$, and $(\downarrow_x^t \varphi)$ are formulas, where $\downarrow_x^t$ evaluates $t$ in the current state and binds the result to variable $x$.

(iv) For $I \in \mathcal{I}$ and $p \in \mathbb{R}$, if $\varphi$, and $\psi$ are formulas then next $(\circ_I \varphi)$, until $(\varphi U_I \psi)$, and min. prevalence $(\nabla_I^p \psi)$ are formulas.

While the semantics of *MFOTL* is defined over infinite sequences, we restrict our attention and definitions to finite sequences. The pair $\langle \vec{\mathfrak{D}}, \vec{\tau}\rangle$ is a *finite temporal structure* over the signature $\sigma$, where $\vec{\mathfrak{D}} = (\mathfrak{D}_0, \mathfrak{D}_1, \cdots, \mathfrak{D}_n)$ is a finite sequence of structures (i.e., scenes) over $\sigma$ and $\vec{\tau} = (\tau_0, \tau_1, \cdots, \tau_n)$ is a finite sequence of non-negative rational numbers $\tau_i \in \mathbb{Q}^+$ with length $n$. The elements in the sequence $\vec{\tau}$ are (increasing) *time stamps*. Furthermore, the interpretations of relations $r^{\mathfrak{D}_0}, r^{\mathfrak{D}_1}, \cdots, r^{\mathfrak{D}_n}$ in a temporal structure $\langle \vec{\mathfrak{D}}, \vec{\tau}\rangle$ corresponding to a predicate symbol $r \in \mathcal{R}$ may change over time. The same is true for functions. Constants $c \in \mathcal{C}$ and the domain $\mathcal{D}$, on the other hand, do not change over time. More formally, we assume for all $0 \leq i < n$ that $\tau_i < \tau_{i+1}$ and for $\mathfrak{D}_i = \langle \mathcal{D}_i, I_i\rangle$ and $\mathfrak{D}_{i+1} = \langle \mathcal{D}_{i+1}, I_{i+1}\rangle$ that $\mathcal{D}_i = \mathcal{D}_{i+1}$. Moreover, $c^{\mathfrak{D}_i} = c^{\mathfrak{D}_{i+1}}$ for each constant symbol $c \in \mathcal{C}$.

A *valuation* is a mapping $v : \mathcal{V} \rightarrow \mathcal{D}$ from variables to domain elements. We write $v[x \mapsto d]$ for the valuation $v$ that maps $x$ to $d$. All other variables are not affected in the valuation $v$. We abuse notation by applying a valuation $v$ also to constant symbols $c \in \mathcal{C}$, with $v(c) = c^{\mathfrak{D}}$.

We evaluate term $t$ for valuation $v$ and structure $\mathfrak{D}$, denoted by $\beta[t, v, \mathfrak{D}]$ as follows. For constants and variables $x$, let $\beta[x, v, \mathfrak{D}] = v(x)$. For function application $a = f(t_1, \cdots, t_{\mathrm{ar}(f)})$, let

$$\beta[a, v, \mathfrak{D}] = f^{\mathfrak{D}}(\beta[t_1, v, \mathfrak{D}], \cdots, \beta[t_{\mathrm{ar}(f)}, v, \mathfrak{D}]).$$

We define the semantics of CMFTBL in terms of the relation $(\vec{\mathfrak{D}}, \vec{\tau}, v, i) \models_{CMFTBL} \varphi$ inductively in Table I, where $|\vec{\tau}|$ denotes the count of time stamps and is mostly used as an upper bound for intervals of temporal operators. The temporal structure $\langle \vec{\mathfrak{D}}, \vec{\tau}\rangle$ satisfies formula $\varphi$ iff $(\vec{\mathfrak{D}}, \vec{\tau}, \emptyset, 0) \models_{CMFTBL} \varphi$.

For $I \in \mathbb{I}$ and the common Boolean constant $\top$ (for true), we define the usual syntactic shorthands and non-metric versions of operators as follows.

| | | | |
|---|---|---|---|
| $(\varphi \wedge \psi)$ | $:=$ | $(\neg((\neg\varphi) \vee (\neg\psi)))$ | logical and |
| $(\varphi \Rightarrow \psi)$ | $:=$ | $((\neg\varphi) \vee \psi)$ | implication |
| $(\forall x : \varphi)$ | $:=$ | $(\neg(\exists x : \neg\varphi))$ | all quantifier |
| $(\Diamond_I \varphi)$ | $:=$ | $(\top\ U_I\ \varphi)$ | eventually |
| $(\Box_I \varphi)$ | $:=$ | $(\neg(\Diamond_I(\neg\varphi)))$ | always |
| $(\Delta_I^p \varphi)$ | $:=$ | $(\nabla_I^{1-p} \neg\varphi)$ | max. prevalence |

We obtain non-metric variants of the temporal operators for interval $[0, \infty)$. The past-time operators from MFOTL (*previously*, *since*, *once*, and *historically*) are not required in the study presented in this paper. They could equally be defined for CMFTBL, but are omitted for brevity.

To enhance the readability (and also the writing) of CMFTBL formulas, we introduce several notational conventions. Let $isVehicle \in \mathcal{R}$ be a unary relation. We define the set of all vehicles $\mathcal{V} \subseteq \mathcal{D}$ as follows:

$$\mathcal{V} := \{d \in \mathcal{D} \mid isVehicle(d)\}$$

Analogously, we define the set of pedestrians as $\mathcal{P}$, and the set of actors $\mathcal{A} := \mathcal{P} \cup \mathcal{V}$ with $\mathcal{P} \cap \mathcal{V} = \emptyset$. For our domain elements, we furthermore introduce a notation reminiscent of object-relational associations in programming languages. For some

| | | |
|---|---|---|
| $(\vec{\mathfrak{D}}, \vec{\tau}, v, i) \models_{CMFTBL} r(t_1, ..., t_{a(r)})$ | iff | $(\beta(t_1, v, \mathfrak{D}_i), \cdots, \beta(t_{\mathrm{ar}(r)}, v, \mathfrak{D}_i)) \in r^{\mathfrak{D}_i}$ |
| $(\vec{\mathfrak{D}}, \vec{\tau}, v, i) \models_{CMFTBL} (\neg\psi)$ | iff | $(\vec{\mathfrak{D}}, \vec{\tau}, v, i) \not\models_{CMFTBL} \psi$ |
| $(\vec{\mathfrak{D}}, \vec{\tau}, v, i) \models_{CMFTBL} (\psi \vee \psi')$ | iff | $(\vec{\mathfrak{D}}, \vec{\tau}, v, i) \models_{CMFTBL} \psi$ or $(\vec{\mathfrak{D}}, \vec{\tau}, v, i) \models_{CMFTBL} \psi'$ |
| $(\vec{\mathfrak{D}}, \vec{\tau}, v, i) \models_{CMFTBL} (\exists x : \psi)$ | iff | $(\vec{\mathfrak{D}}, \vec{\tau}, v[x \mapsto d], i) \models_{CMFTBL} \psi$, for some $d \in \mathcal{D}$ |
| $(\vec{\mathfrak{D}}, \vec{\tau}, v, i) \models_{CMFTBL} (\circ_I \psi)$ | iff | $\tau_{i+1} - \tau_i \in I$ and $(\vec{\mathfrak{D}}, \vec{\tau}, v, i+1) \models_{CMFTBL} \psi$ and $(\vec{\mathfrak{D}}, \vec{\tau}, v, k) \models_{CMFTBL} \psi$, for all $k \in \mathbb{N}$ with $j < k \leq i$ |
| $(\vec{\mathfrak{D}}, \vec{\tau}, v, i) \models_{CMFTBL} (\psi U_I \psi')$ | iff | for some $j \geq i, \tau_j - \tau_i \in I, (\vec{\mathfrak{D}}, \vec{\tau}, v, j) \models_{CMFTBL} \psi'$, and $(\vec{\mathfrak{D}}, \vec{\tau}, v, k) \models_{CMFTBL} \psi$, for all $k \in \mathbb{N}$ with $i \leq k < j$ |
| $(\vec{\mathfrak{D}}, \vec{\tau}, v, i) \models_{CMFTBL} \nabla_I^p \psi$ | iff | $(\vec{\mathfrak{D}}, \vec{\tau}, v, j) \models_{CMFTBL} \psi$, for at least fraction $p$ of indices $i \leq j \leq \lvert\vec{\tau}\rvert$ with $\tau_j - \tau_i \in I$ |
| $(\vec{\mathfrak{D}}, \vec{\tau}, v, i) \models_{CMFTBL} (\downarrow_x^t \psi)$ | iff | $(\vec{\mathfrak{D}}, \vec{\tau}, v[x \mapsto \beta(t, v, \mathfrak{D}_i)], i) \models_{CMFTBL} \psi$ |

TABLE I: Inductive definition of the relation $(\vec{\mathfrak{D}}, \vec{\tau}, v, i) \models_{CMFTBL} \psi$

vehicle $v \in \mathcal{V}$ and the relations $\{isEgo, isLane, onLane\} \subseteq \mathcal{R}$ we use shorthand notations like:

$$v.isEgo := isEgo(v)$$
$$v.lane := l \mid l \in \mathcal{D} \wedge isLane(l) \wedge onLane(v, l)$$

All formulas used for the properties in our tree-based classifier need to be evaluated for the ego vehicle and usually depend on one unary relation. Thus, for most formulas $\varphi$ we can define a pattern for some $r \in \mathcal{R}$:

$$\varphi := \exists v \in \mathcal{V} : \Box(v.isEgo) \wedge r(v)$$

In such cases, we just define the relation $r$. For example, assuming $r = obeyedSpeedLimit$, we could validate that the ego vehicle at all times obeys the speed limit:

$$obeyedSpeedLimit(v) := \Box\big(v.speed \leq v.lane.speedLimitAt(v.pos)\big)$$

The associations *v.speed* and *v.pos* are functions as introduced before and *speedLimitAt* is a function from a position number and a lane to a speed limit number. For numbers, we assume the relations $\{eq, neq, lt, gt, leq, geq\} \in \mathcal{R}$ to represent the common mathematical comparators $\{=, \neq, <, >, \leq, \geq\}$, which we also allow as notation shortcuts.

With these notational conventions, we can quite straightforwardly define traffic rules and environmental features using CMFTBL formulas and evaluate those on sequences of scenes. Each predicate of our case study (cf. Sect. V) is expressed this way. For comparison, consider the *obeyedSpeedLimit* formula without these syntactic conventions:

$$\varphi_{oSL} := \exists v \in \mathcal{D} : \Box\big(isVehicle(v) \wedge isEgo(v)\big) \wedge \Box\Big(\exists l \in \mathcal{D} : \exists p \in \mathcal{D} : isLane(l) \wedge onLane(v, l) \wedge \wedge\, leq\big(speed(v), speedLimitAt(p, l)\big)\Big)$$

Assumptions on the data can furthermore be validated using dedicated formulas. For example, *only one ego vehicle may exist at all times and it does not change over time* could be expressed as:

$$uniqueEgo := \exists v \in \mathcal{V} : \Box\left(v.isEgo \wedge \forall v' \in \mathcal{V} : v'.isEgo \Rightarrow v = v'\right)$$

Other data checks – e.g., that each vehicle can only be on one lane at a time or that a vehicle's actual position on a lane can not be greater than the lane's length – can be added accordingly to ensure the single relation nature of the object associations as well as overall data sanity and consistency.

## IV. Classifiers for Scenarios and Metrics on Sets of Scenarios

We want to use CMFTBL formulas for expressing features of scenarios and for classifying recorded driving data into scenario classes. Formally, we assume recorded driving data to be given as temporal structures $\langle \vec{\mathfrak{D}}, \vec{\tau} \rangle$ over some fix basic signature. This basic signature is the set of properties that is provided as information in the data, i.e., objects with positions and classifications on a road network with information about lanes, signs, and signals.

For the scope of this paper, we additionally assume that the recorded data is already segmented into sequences in a meaningful way. We use $\mathfrak{S}$ to denote a set of segments of form $\langle \vec{\mathfrak{D}}, \vec{\tau} \rangle$. In practice, segmentation could either be done based on a map or based on classification, e.g., of driving maneuvers of the ego vehicle, or by some other sensible approach.[2]

We can then define classifiers that identify the scenario class of some observed data $\langle \vec{\mathfrak{D}}, \vec{\tau} \rangle$ and define metrics over observed classes of scenarios.

### A. Classifiers for Scenarios

Instead of simply using a set of features, we organize features hierarchically in trees to account for dependencies between features (a lane change, e.g., can only occur on a multi-lane road). This will enable us to capture the taxonomies of features found in the 6-layer model or in draft standards for specifying operational design domains.

*Definition 1 (Tree-Based Scenario Classifier):* A tree-based scenario classifier (TSC) $\mathbb{T}$ is a tuple $\langle \mathcal{Q}, q_r, \Gamma, \lambda_l, \lambda_u \rangle$ with

- set of nodes $\mathcal{Q}$ (i.e., the modeled features)
- root node $q_r \in \mathcal{Q}$

[2]In our case study, we will segment data with the help of a map into sequences that contain one main driving situation: driving through an intersection, driving along a section of a multilane road between two intersections, etc.

- set of edges $\Gamma$ of type $(q, q', \varphi)$ where
  - $q, q' \in \mathcal{Q}$ are source and destination,
  - CMFTBL formula $\varphi$ is the edge condition,
- lower bounds for sub-features of nodes $\lambda_l : \mathcal{Q} \to \mathbb{N}_0$
- upper bounds for sub-features of nodes $\lambda_u : \mathcal{Q} \to \mathbb{N}_0$

We write $q \xrightarrow{\varphi} q'$ for $(q, q', \varphi)$. We require $\mathbb{T}$ to be a tree rooted at $q_r$. For $q \in \mathcal{Q}$, let $c(q) = \{q' \mid q \xrightarrow{\varphi} q' \in \Gamma\}$ denote the children of $q$. Bounds must be $0 \leq \lambda_l(q) \leq \lambda_u(q) \leq |c(q)|$. A path of length $k$ in $\mathbb{T}$ is a sequence of $k$ transitions $q_{i-1} \xrightarrow{\varphi_i} q_i$ with $1 \leq i \leq k$ and $q_0 = q_r$.

Inspired by feature models, we name certain types of nodes $q \in \mathcal{Q}$ depending on their lower and upper bounds (abbreviated notation with parentheses):

| | |
|---|---|
| **All / (A)** | $\lambda_l(q) = \lambda_u(q) = \vert c(q)\vert$ |
| **Exclusive / (X)** | $\lambda_l(q) = \lambda_u(q) = 1$ |
| **Optional / (O)** | $\lambda_l(q) = 0 \wedge \lambda_u(q) = \vert c(q)\vert$ |
| **a/b-Bounded / (a..b)** | $\lambda_l(q) = a \wedge \lambda_u(q) = b$ |
| **Leaf / ( )** | $\lambda_l(q) = \lambda_u(q) = 0$ |

We introduce bounds on sub-features as a means of computing an upper bound on the number of combinatorial combinations, i.e., the number of observable scenario classes, in the next section. A more precise approach to computing feasible scenarios would be to compute satisfiable combinations of features. Such an approach, however, does not seem feasible or meaningful. Even if the satisfiability of some fragment of CMFTBL can be established, there is no mechanism to constrain acceptable models to realistic segments.

We can now describe individual scenario classes for a scenario classifier.

*Definition 2 (Scenario Class):* For a given tree-based scenario classifier $\mathbb{T} = \langle \mathcal{Q}^o q_r^o, \Gamma^o, \lambda_l^o, \lambda_u^o\rangle$, a scenario class is a tree $T = \langle \mathcal{Q}, q_r, \Gamma\rangle$ with

- set of nodes $\mathcal{Q} \subseteq \mathcal{Q}^o$
- root node $q_r = q_r^o$
- set of edges $\Gamma$ of type $(q, q')$ and such that $(q, q', \varphi) \in \Gamma^o$

We require the number of children $c(q)$ for every node $q \in \mathcal{Q}$ to be within the lower and upper bounds of $q$ in $\mathbb{T}$.

Let $\mathcal{T}_{\mathbb{T}}$ denote the (finite) set of all scenario classes for tree-based classifier $\mathbb{T}$, and let $\mathcal{W}$ denote the (infinite) set of observable segments of driving data $\langle \vec{\mathfrak{D}}, \vec{\tau}\rangle$. We denote the classification function that maps observed driving data $\langle \vec{\mathfrak{D}}, \vec{\tau}\rangle$ to a scenario class $T$ based on the tree-based scenario classifier $\mathbb{T}$ by $C_{\mathbb{T}} : \mathcal{W} \to \mathcal{T}_{\mathbb{T}}$. For recorded data segment $\mathcal{S} = \langle \vec{\mathfrak{D}}, \vec{\tau}\rangle$, we compute $C_{\mathbb{T}}(\mathcal{S}) = \langle \mathcal{Q}, q_r, \Gamma\rangle$ by computing the set $\mathcal{Q}$ of nodes, which uniquely determines the set of transitions. We initialize $\mathcal{Q}$ as $\{q_r^o\}$ and then add every node $q'$ for which $q \in \mathcal{Q}$ and $(q, q', \varphi) \in \Gamma^o$ with $\mathcal{S} \models \varphi$ until a fixed point is reached. We assume that bounds permit that a valid class can be computed for every realistic segment $\mathcal{S}$ and lift $C_{\mathbb{T}}$ to sets of segments by letting $C_{\mathbb{T}}(\mathfrak{S})$ denote the set of observed scenario classes for $\mathfrak{S}$.

### B. Coverage Metrics for Sets of Scenarios

Given a set $\mathfrak{S}$ of recorded segments and a classifier $\mathbb{T}$, we want to analyze and quantify *if and to which degree* the recorded data covers possible scenarios.

We start by showing how to compute the number of scenario classes for a tree-based scenario classifier $\mathbb{T} = \langle \mathcal{Q}, q_r, \Gamma, \lambda_l, \lambda_u\rangle$. Let $\Gamma_q = \{(q, q', \varphi) \in \Gamma\}$ be the set of edges originating in $q$, and

$$[\Gamma_q]^{\lambda_l(q)..\lambda_u(q)} =_{def} \bigcup_{i=\lambda_l(q)}^{\lambda_u(q)} [\Gamma_q]^i$$

be the set of all subsets of these edges with size within lower bound and upper bound of $q$. We define the size $|\mathbb{T}| = |q_r|$ recursively as

$$|q| =_{def} \sum_{G \in [\Gamma_q]^{\lambda_l(q)..\lambda_u(q)}} \left( \prod_{(q,q',\varphi) \in G} |q'| \right).$$

The primary metric we are considering is **scenario class coverage** (SCC), expressing the ratio between the amount of observed scenario classes and the number of classes modeled by classifier $\mathbb{T} = \langle \mathcal{Q}, q_r, \Gamma, \lambda_l, \lambda_u\rangle$. For a set $\mathfrak{S}$ of recorded segments, we define

$$\mathrm{SCC}(\mathfrak{S}, \mathbb{T}) =_{def} \frac{|\mathcal{C}_{\mathbb{T}}(\mathfrak{S})|}{|\mathbb{T}|}$$

It can be expected that gaining high coverage on TSCs with (potentially multiple combinations of) rare events requires an increasingly high amount of test scenarios. To measure the individual rarity of the modeled environmental conditions, from which explanations for coverage gaps might be derived, we introduce a metric for **absolute feature occurrence** (afo). It counts the number of segments that are classified as scenarios containing a given node (i.e., feature).

$$\mathrm{afo}(\mathfrak{S}, q) =_{def} |\{\langle \mathcal{Q}, q_r, \Gamma\rangle \in \mathcal{C}_{\mathbb{T}}(\mathfrak{S}) \mid q \in \mathcal{Q}\}|$$

In addition to coverage, which only considers if a scenario class has been observed, we define **scenario instance count** (sic) to count how often a certain class has been encountered in a set of scenarios.

$$\mathrm{sic}(\mathfrak{S}, t) =_{def} |\{\mathcal{S} \in \mathfrak{S} \mid \mathcal{C}_{\mathbb{T}}(\mathcal{S}) = t\}|$$

Similar to calculating the size of a TSC, we can enumerate all possible scenario classes and use them to identify **class instance missings**, i.e., classes as which no $\mathcal{S} \in \mathfrak{S}$ is classified. However, gaining meaningful insights from large sets of missing classes is difficult. Therefore, we also analyze **feature pair misses**, i.e., pairs of TSC nodes that do not exist together in any observed class.

## V. Evaluation

Our evaluation is designed as a single case mechanism experiment [23] that validates the presented approach and our implementation. We develop a tree-based scenario classifier for an urban driving environment and use it for analyzing simulated urban traffic. Features are chosen to model the types of properties (or labels) that are envisioned for specifications of operational design domains (ODDs) as described in BSI 1883 [16] or OpenODD [17].

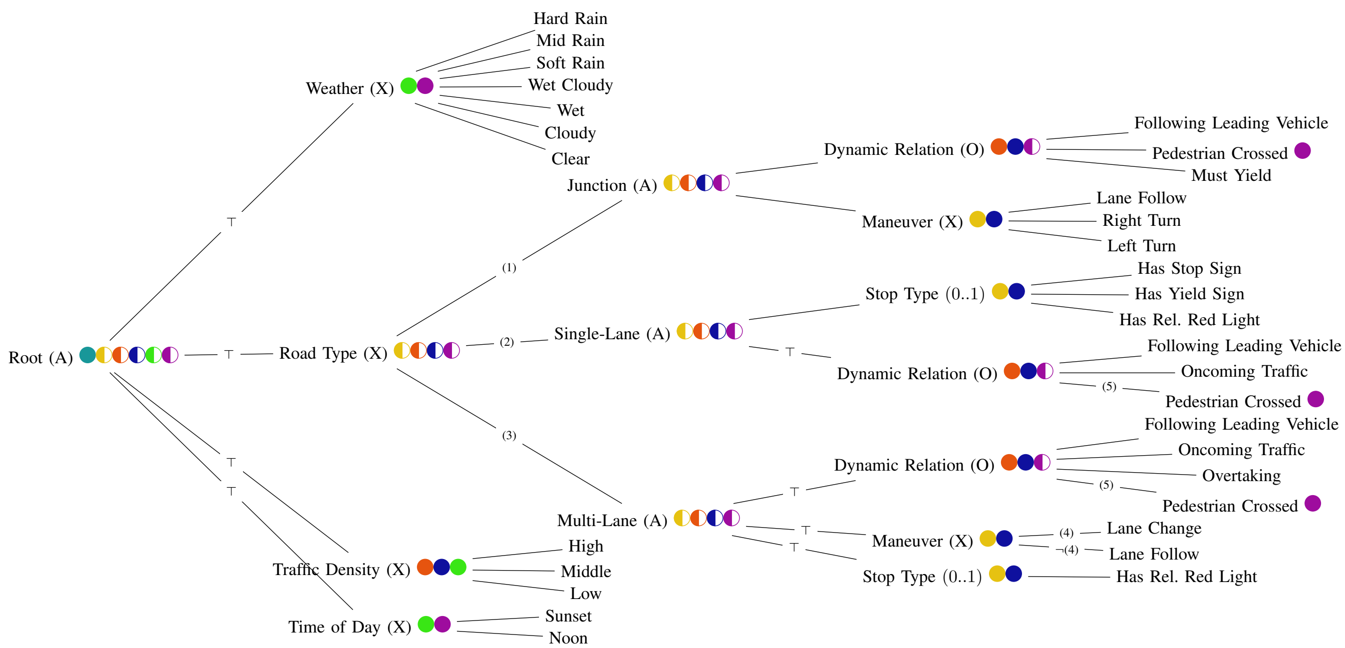



Fig. 2: This figure shows the full classifier structure used for the analyses in this paper. Hereby, edge labels reference the related logical formula that is deciding whether an edge is taken. We use the following classifier sets: ◐ *full TSC*, ◐ *layer 1+2*, ◐ *layer 4*, ◐ *layer 1+2+4*, ◐ *layer (4)+5*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

Since this is the first work on scenario class coverage, we aim at answering the following questions — mostly with qualitative data.

Q1. Can relevant properties of operational design domains be expressed in CMFTBL?

Q2. Is it computationally feasible to classify scenarios with a tree-based scenario classifier?

Q3. To which degree can scenario class coverage be achieved and can scenario class coverage generate useful insights (e.g., missing classes)?

The remainder of this section discusses the classifier developed for our case study, details the experimental setup, presents results from the simulated experiments, and provides initial answers to the above questions.

### A. Tree-Based Scenario Classifier Definition

To construct a tree-based scenario classifier (TSC) for our case study, we evaluated the 6-layer model of scenario classification by Scholtes et al. [12] and extracted observable features. We defined logical formulas using CMFTBL that are capable of identifying these features on segments (i.e., sequences of scenes). The hierarchic organization of all the features resulted in the TSC visualized in Fig. 2. We additionally define smaller TSCs by grouping features that we want to analyze together. This allows us to study coverage for smaller sets of features. For the sake of presentation in this paper, we introduce TSC projections. They combine related features into subsets of all features of an original *full TSC*. The following projections are based on the layers of information discussed in [12]:

- **full TSC**: the complete TSC that serves as the reference point for the comparison with the other projections
- **layer 1+2**: driving features in relation to static information (roads, lanes traffic signs, etc.) during the scenario run
- **layer 4**: driving features in relation to other objects that dynamically change during the scenario run (like other vehicles, pedestrians, etc.)
- **layer 1+2+4**: combination of static information and dynamically changing objects
- **layer (4)+5**: environmental features from layer 5 combined with the traffic density from layer 4
- **pedestrian**: example for a more specialized projection to analyze the coverage of pedestrians crossing the street in all possible environmental situations

We did not include Layer 3 *(Temporary Modifications of Layer 1 and Layer 2)* and Layer 6 *(Digital Information)*, as there were no elements of these layers available in our test environment.

We visualize our projection labels in Fig. 2 through colored circles. A filled circle ● indicates that the complete subtree rooted at this node is included in the projection. Half circles ◐ indicate that the subtree rooted at this node is partially (i.e., as labeled) included in the projection. All edges of the TSC have a corresponding logical condition. For readability reasons, Fig. 2 only depicts *always-true* edge conditions and formulas explicitly described in this paper. A more detailed overview of the set of implemented predicates is given in the next section. There, we discuss the total number of predicates, their definitions using our newly introduced operators, and their complexity.

### B. Predicates

For our case study, we defined all scenario class features as CMFTBL predicates and formulas. This section discusses a selection of these predicates, in particular to demonstrate our newly introduced prevalence and binding CMFTBL operators.

Let $\mathcal{V}$ be the set of all vehicles and $\mathcal{P}$ be the set of all pedestrians. As a basis for our data structure, we use the OpenDrive standard[3]. Therefore, we can reason about static and dynamic relations between vehicles and other entities (e.g., other vehicles, pedestrians, landmarks, etc.) by mapping entity positions to their respective OpenDrive lanes. Consequently, each entity is related to a lane which in itself contains additional information we use for our predicate definitions.

To decide whether a vehicle $v \in \mathcal{V}$ was primarily driving through a junction during the analyzed segment, we require the vehicles' road to be categorized as a junction in at least 80% of the time stamps.

$$isInJunction(v) := \nabla^{0.8}(v.lane.road.isJunction) \quad (1)$$

Similarly, to determine whether a vehicle $v \in \mathcal{V}$ is driving on a single-lane road, we require $v$'s road to have only one lane for at least 80% of the observed scenes. Additionally, we require the road to not be classified as a junction.

$$\begin{aligned} onSingleLaneRoad(v) := &\neg isInJunction(v) \wedge \\ &\nabla^{0.8}(sameDirectionLaneCount(v.lane) = 1) \end{aligned} \quad (2)$$

We define the predicate for deciding if a vehicle $v \in \mathcal{V}$ is on a multi-lane road by combining the predicates (1) and (2).

$$\begin{aligned} onMultiLaneRoad(v) :=&\neg onSingleLaneRoad(v) \wedge \\ &\neg isInJunction(v) \end{aligned} \quad (3)$$

To be able to detect a lane change for a given vehicle $v \in \mathcal{V}$, our binding operator is utilized. 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.

$$changedLane(v) := \downarrow_l^{v.lane} \big(\Diamond(l \neq v.lane)\big) \quad (4)$$

Pedestrians crossing a lane are (at some timestamp) identified as being *on* this lane. Therefore, we can detect if for a pedestrian $p \in \mathcal{P}$ and vehicle $v \in \mathcal{V}$ the predicate `onSameLane(v,p)` holds. For the vehicle $v$, a crossing of pedestrian $p$ is only relevant if it happens 'closely in front of $v$'. This is defined by `inReach(p,v)` which checks if $p$'s position on the (same) lane is in front of and at most 10 meters away from $v$.

$$\begin{aligned} &pedestrianCrossed(v) := \\ &\quad \Diamond\big(\exists p \in \mathcal{P} : onSameLane(p,v) \wedge inReach(p,v)\big) \\ &onSameLane(a_0, a_1) := a_0.lane = a_1.lane \\ &\quad inReach(a_0, a_1) := 0 \leq a_0.pos - a_1.pos \leq 10 \end{aligned} \quad (5)$$

All predicates defined for the tree-based scenario classifier in Fig. 2 are of similar complexity as the ones presented above. In total, we defined 51 predicates (including sub-predicates) to completely express the detection of the modeled features for our experiments. We used the min. prevalence operator in 18 predicates to model that some feature is present for most of the time covered by a segment. The binding operator was used once directly, for specifying the change of lanes, and once indirectly by using the negation of the change of lanes.

[3]https://www.asam.net/standards/detail/opendrive/

## C. Experimental Setup

As the basis for our experiments, we built a toolchain using the CARLA simulator [18] and an analysis framework written in the Kotlin programming language[4] that classifies recorded scenario runs according to a tree-based classifier. As a proof-of-concept implementation, it is not optimized for performance. However, it is already sufficient to run our experiments within a few hours. Thus, we left a proper algorithmic approach to CMFTBL evaluation for future work.

Based on the classifications of recorded runs, a subsequent analysis step calculates the coverages and analyses introduced in Sect. IV-B. Additionally, by iteratively analyzing the set of recorded segments, we can measure class coverage over time by counting newly observed scenario classes. This provides us with an increasing curve on coverage.

In our toolchain, TSCs are evaluated on an abstract representation of the scene, i.e., the ego vehicle and its surroundings. This data structure is designed to be constructed in various ways, and we provide an implementation for CARLA. Static as well as dynamic data is exported into JSON files during simulation, read by the Kotlin framework, and weaved together forming the consistent abstract view of the world.

The data for each simulation run is then segmented to be classified. The primary factor for this segmentation is the ego vehicles' road. This results in each simulation run being cut into individual segments of either driving through a junction or following a (potentially multi-lane) road section without crossed lanes. After this segmentation, we discard all segments too short for a meaningful analysis, i.e., segments containing only 10 or fewer scenes.

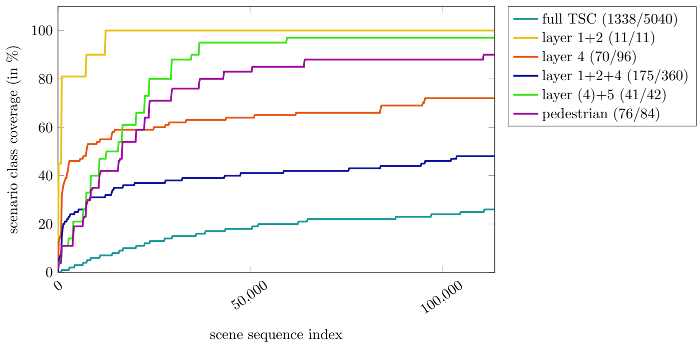
For our experiments, we recorded 100 simulation runs of 5 minutes each. In every run, a random map, daytime, and weather were chosen and up to 200 vehicles and 30 pedestrians were spawned randomly on the map. For maps that do not specify enough spawn points, we spawned as many actors as possible.

During the simulation, all vehicles drove around the map using CARLA's autopilot. We analyzed each simulation run multiple times: once with each vehicle being considered to be *the ego vehicle*. This enabled us to increase the amount of encountered situations (and therefore coverage) without the need to record about 200 times as many simulation runs. Overall, this resulted in $113,767$ analyzed segments representing $1,104$ hours of driving data. The analysis of this data with the classifiers and predicates introduced in Sects. V-A and V-B takes about 118 minutes on a single core of a 2021 Apple M1 Pro SoC. A reproduction package for our experiments – a virtual machine image that contains our recorded driving data, the framework, specifications, and analysis code – is archived on Zenodo [20].

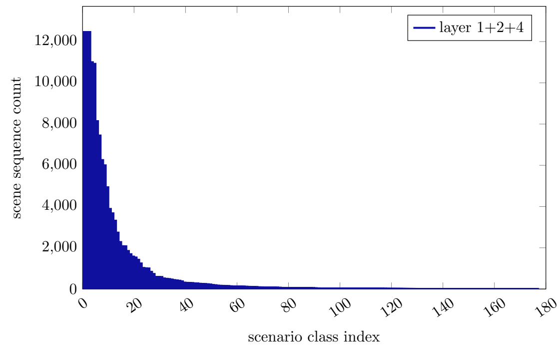
## D. Experimental Results

In this section, we present the application of our coverage metrics and analyses for scenario classes based on our data set of $113,767$ classified segments.

[4]https://kotlinlang.org




(a) Scenario class coverage over the course of the 113,767 analyzed segments




(b) Distribution of the 175 observed scenario classes for the *layer 1+2+4* projection

Fig. 3: Coverage of scenario classes and distribution of segments over classes

**Class Coverage.** We visualize our results for scenario class coverage over the course of analyzed segments in Fig. 3a. Each colored curve represents the coverage result of one classifier projection, as defined in Sect. V-A. The legend also shows for each projection the final count of observed classes after analysis of all $113,767$ segments as well as the number of possible classes. The *layer 1+2* projection covers $100\%$ of scenario classes after $12,233$ analyzed segments. Furthermore, *layer (4)+5* almost fully covers the possible scenario classes after around $59,409$ segments, but misses one scenario class and therefore only reaches $97\%$. The *pedestrian* projection covers over $90\%$ of relevant scenario classes. The projections *layer 4* and *layer 1+2+4* cover $72\%$ and $48\%$ of relevant scenario classes, respectively. Finally, the reference projection *full TSC* reaches a coverage of $26\%$.

**Scenario Instance Count.** In Fig. 3b, we exemplarily demonstrate the scenario instance count metric of the 175 observed scenario classes for the *layer 1+2+4* projection. The plot shows a long-tail distribution in which $85,120$ segments of the total $113,767$ segments are classified into only $15$ scenario classes. The remaining $28,647$ segments are classified into the remaining $160$ scenario classes. The three most common scenario classes are each observed about $11,000$ times.

**Test Scenario Set Analysis.** Table II gives an overview of the statistical values of our generated test scenario set. We used three maps shipped with the CARLA simulator with an average lane length of 37.4 meters. Note that especially *Town 01* has some long lane sections with the maximum length being 310 meters. In contrast, some lane sections are only $4$ meters long. In total, we generated $1,104.2$ hours of data with a segment count of $113,767$. On average, there are over $1,000$ segments per road section of each map, while the segments have an average length of over $50$ scenes with a maximum scene count of 592.

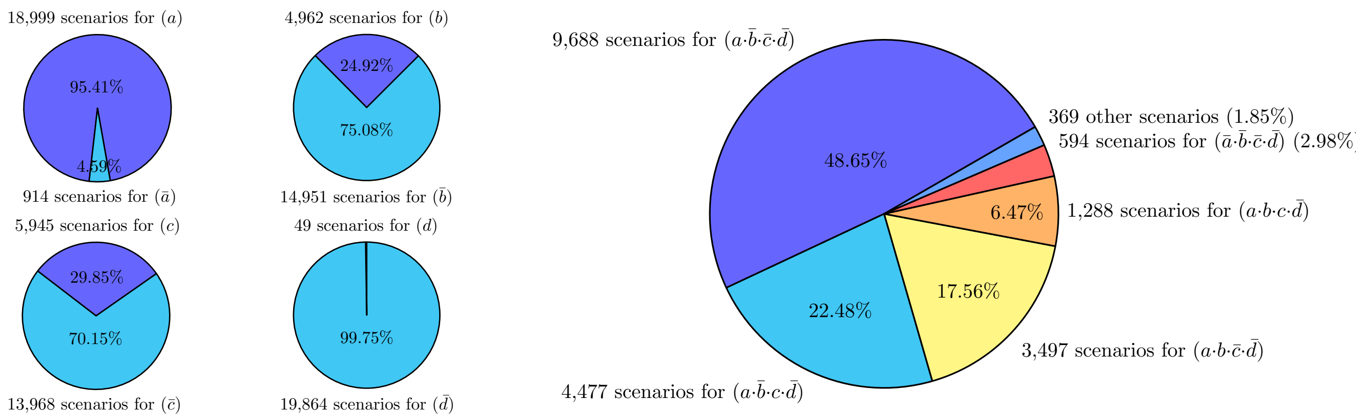
**Absolute Feature Occurrence.** Our analysis provides detailed insights into specific scenario classes regarding the underlying features and their combinations. To demonstrate the results, Fig.4 exemplarily shows analyses on the *Dynamic Relation* features of the *Multi-Lane* node of our TSC (cf. Fig. 2). For better readability in the figure, we label the observable features as *a*="*Oncoming Traffic*", *b*="*Pedestrian Crossed*", *c*="*Following Leading Vehicle*" and *d*="*Overtaking*". We also write $(x)$ or $(\bar{x})$ if feature *x* was observed or not observed, respectively. For example, the combination $(a{\cdot}b{\cdot}c{\cdot}\bar{d})$ describes the scenario classes in which *Oncoming Traffic*, *Pedestrian Crossed* and *Following Leading Vehicle* are observed, while *Overtaking* is not observed. Figure 4a visualizes the individual absolute occurrences of each observable feature for the dynamic relations on multi-lane roads. The percentages are based on the $19,913$ analyzed segments classified as containing the *Multi-Lane* feature. Here, *Oncoming Traffic* (a) appears in $95.41\%$ of the total occurrences. *Pedestrian Crossed* (b) and *Following Leading Vehicle* (c) are similarly present with a coverage of $24.92\%$ and $29.85\%$, whereas *Overtaking* (d) is only encountered in $0.25\%$ of the analyzed segments.

**Full Combinatorial Analysis.** As defined in our TSC (cf. Sect. V-A) the *Dynamic Relation* node is marked as *Optional*, i.e., all combinations of the four children form valid scenario classes. Consequently, there are 16 possible combinations of features. Figure 4b visualizes the distribution for all combinations of features. It can be seen that $95.16\%$ of observed scenarios are covered by the following four feature combinations: $(a{\cdot}\bar{b}{\cdot}\bar{c}{\cdot}\bar{d})$, $(a{\cdot}\bar{b}{\cdot}c{\cdot}\bar{d})$, $(a{\cdot}b{\cdot}\bar{c}{\cdot}\bar{d})$ and $(a{\cdot}b{\cdot}c{\cdot}\bar{d})$. Of the five feature combinations that never occurred, each includes feature $(d)$, which directly stems from the overall low occurrence of only $0.25\%$ of feature $(d)$. Additionally, the three observed combinations that include feature $(d)$ are among the four combinations with lowest occurrence.

**Feature Pair Misses.** As discussed before, our method yields precise information on which scenario classes never occurred. But as the full TSC analysis resulted in 3,702 unseen classes, a detailed analysis is unfeasible. With the analysis of feature pair misses, we instead focus on predicate combinations that never occurred. This results in the information that the following five predicate combinations were never observed together: (Overtaking & Lane Change), (Overtaking & Has Red Light), (Has Stop Sign & High Traffic), (Has Yield Sign & High Traffic), (Has Yield Sign & Middle Traffic).

| Map | Data [h] | Segments [#] | Road Sections [#] | Seg./R.S. [#] | | Seg. length [#] | | | | Lane length [m] | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | AVG | SD | Avg | SD | Min | Max | Avg | SD | Min | Max |
| Town 01 | 467.4 | 38,190 | 38 | 1,005.0 | 696.8 | 89.1 | 68.1 | 11 | 580 | 52.7 | 68.8 | 16 | 310 |
| Town 02 | 357.1 | 38,702 | 28 | 1,382.2 | 856.4 | 67.4 | 51.8 | 11 | 592 | 34.2 | 32.5 | 16 | 178 |
| Town 10 | 279.7 | 36,875 | 32 | 1,152.3 | 815.7 | 55.6 | 69.7 | 11 | 584 | 36.5 | 27.0 | 4 | 122 |
| Total | 1,104.2 | 113,767 | 98 | | | | | | | | | | |

TABLE II: Analysis of simulated test drives per map: driving time, road sections, segments per road sections, segment lengths, and lane lengths




(a) Distributions of individual feature occurrences

(b) Distribution of all feature combinations

Fig. 4: Distributions of "dynamic relations" for "multi-lane" roads. We define *a*="Oncoming Traffic", *b*="Pedestrian Crossed", *c*="Following Leading Vehicle" and *d*="Overtaking". The 369 other scenarios are composed of the following combinations: 152 scenarios for $(\bar{a}{\cdot}\bar{b}{\cdot}c{\cdot}\bar{d})$, 143 scenarios for $(\bar{a}{\cdot}b{\cdot}\bar{c}{\cdot}\bar{d})$, 37 scenarios for $(a{\cdot}\bar{b}{\cdot}\bar{c}{\cdot}d)$, 25 scenarios for $(\bar{a}{\cdot}b{\cdot}c{\cdot}\bar{d})$, 9 scenarios for $(a{\cdot}b{\cdot}\bar{c}{\cdot}d)$ and 3 scenarios for $(a{\cdot}\bar{b}{\cdot}c{\cdot}d)$. The remaining combinations $(a{\cdot}b{\cdot}c{\cdot}d)$, $(\bar{a}{\cdot}b{\cdot}c{\cdot}d)$ $(\bar{a}{\cdot}b{\cdot}\bar{c}{\cdot}d)$, $(\bar{a}{\cdot}\bar{b}{\cdot}c{\cdot}d)$ and $(\bar{a}{\cdot}\bar{b}{\cdot}\bar{c}{\cdot}d)$ never occurred.

### *E. Discussion*

In the previous section, we visualized and described various methods of analyzing test drives regarding a given specification. We demonstrated the expressiveness of our approach with coverage metrics for scenario classes and predicate combinations. This section discusses these findings in the context of the three questions formulated on page 5 and closes with a discussion on threats to validity.

**Q1 (Expressivity).** Using the CMFTBL logic, we were able to express many relevant properties for common driving situations considered in the proposals for operational design domains [16] and approaches like the 6-layer model [12]. In particular, the prevalence operator was helpful to detect properties where it is natural to formulate 'majority of the time' constraints (like environmental conditions or traffic density). The binding operator adds an intuitive mechanism for value storage that can be used to include 'remembered information from the past' in the evaluation of a state. The notational conventions (like dedicated sets for vehicles/pedestrians or object-relational element associations) furthermore facilitate a mapping from a more human-readable presentation to CMFTBL formula syntax. Properties we did not include in our case study were usually left out not because it was impossible (or even particularly inconvenient) to be expressed using CMFTBL, but because we were not able to automatically extract – with a reasonable amount of effort – the required information from our simulation setup with CARLA (e.g., yield priorities in roundabouts, behavior on highway entries, or temporary modifications like construction work). We are confident that the logic can express most of the features required of a scenario classifier for an ODD.

**Q2 (Analysis Time).** We analyzed a total of $1,104$ hours of data from simulated test drives, which took a little over $118$ minutes. With a total of $113,767$ segments we have on average $34.93$ seconds of driving data per segment and $62.23$ milliseconds of computation time per segment evaluation. While a more comprehensive scenario classifier would contain more features, due to the tree-based structure of our classifier, whole sub-trees get cut off from evaluation if a condition does not hold (e.g., none of the *single-lane* features of Fig. 2 are evaluated when the segment is recognized as a *junction*). The obtained results thus indicate that our approach is generally feasible with regard to computation time. Even online monitoring of properties while driving (i.e., after completing a segment) seems possible.

**Q3 (Scenario Coverage).** Our experiments demonstrate that scenario coverage can be achieved with our concept of hierarchical classifiers. Even though the features evaluated with our classifiers are limited in scope, they cover a sensible amount of situations for urban driving. With our approach, it is possible to automatically classify test drives based on a predefined specification. Our detailed analyses proved particularly useful for the interpretation of the coverage levels our projections converge to. All five feature combinations not encountered at all throughout our data combine a *layer 1+2* feature with a *layer 4* feature. Due to the combinatorial nature of our classifier concept, about half of all combinations in the *layer 1+2+4* projection remain undetected. We can utilize this information and investigate why certain feature pairs are missed. For instance, in the three maps we included in our experiments, only a single junction on a small side road has a yield sign. We are less likely to detect middle or high traffic density on this road. These insights can be used to plan test drives or as a basis for analyzing the relevance of specified scenarios in some real environment.

### Threats to Validity

*Internal Validity.* To test our approach, we generated data with CARLA, as this allowed us to produce a large set of test drives using automated scripts. We have not tested our approach on a set of ground truth data to check our predicates against pre-labeled data. However, we manually inspected rendered videos of the generated data set to match the actual driving situations we addressed with our formulas. Combined with manually written test cases for each predicate and all implemented logical operators, we are confident that the predicates are capable of detecting the scenarios they are designed for. Even though the binding operator was not used pervasively in our case study, it would not have been possible to express a change of lanes without the operator and we expect that many more complex predicates pertaining to driving require the operator. Two examples are *change of heading* and *giving the right of way* in an all-way stop situation.

*External Validity.* We were able to define all relevant predicates for our experiments, but this is not fully independent of our selection of maps and the behavior of CARLA's autopilot. As stated in Sect. II, we focused on analyzing urban driving scenarios, but the available maps in the CARLA simulator also include interstate traffic. As other works already formalize interstate traffic (see [24]), we are confident that we are also capable of analyzing new types of maps and traffic situations using our introduced approach. Furthermore, we can also record human-controlled driving behavior using the CARLA simulator and a hardware setup containing a steering wheel, pedals and multiple screens. These recordings produce the same file format as our generated scenarios and can therefore easily be analyzed.

*Concept Validity.* When analyzing more complex situations, the specification might get too large for our approach to be practically usable. Especially, as data from the real world can contain errors and deviations, various complex predicates and classification trees might become necessary. Our experiments use the perfect world perception provided by CARLA, which removes the fuzziness of sensor data. Analyzing real-world data requires the intelligent handling of such fuzzy sensor data streams. Predicates then need to take into consideration that the environmental perception, such as object tracking, might be incorrect or imprecise. Previous works show that current research is already addressing certain problems in regards to environmental perceptions [25], such as sensor fusion [26], or object reference generation [27]. We are confident that with further results and insights, we can use our formal specifications to include fuzzy perception data.

## VI. Related Work

Our approach is related to various existing works on the safety of autonomous vehicles.

**Formalizing traffic scenarios.** Previous work formalizes traffic rules using different formal logics to define specific scenario rules. Esterle et al. formalize traffic rules for highway situations [28] by using Linear Temporal Logic (LTL). The same logic is also used by Rizaldi et al. [29] to formalize German overtaking rules. Additionally, they provide verified checkers that are able to calculate the satisfaction of a specific trace against the defined LTL formulas. Other works formalize similar traffic rules using the Metric Temporal Logic such as interstate traffic [24] or intersections [30]. Additional traffic rules regarding uncontrolled intersections are formally described by Karimi and Duggirala [31] using Answer Set Programming. Most of their rules specify the expected behavior of traffic participants in regards of the right of way at unprotected intersections.

**Scenario-based Testing.** In the past few years, research has started to focus on scenario-based safety assurance (mainly testing) of autonomous vehicles [32], [33], exploring definition, specification, instantiation, execution [18], and generation of scenarios for scenario-driven development [34], for regression testing [35], for autonomous driving testing [36], and for accident scenarios [37], mining scenarios from data, test automation [38], notions of similarity between scenarios [39], and on finding critical test scenarios [40]. Steimle et al. [41] provide a taxonomy and definitions of terms for scenario-based development and testing. Ulbrich et al. [13] define a scenario to be a sequence of scenes and a scene to be a snapshot of a vehicle's environment, including all actors, observers, self-representations, and relationships between them. Klischat and Althoff [42] generate critical test scenarios using evolutionary algorithms by minimizing the solution space of the vehicle under test. Menzel et al. [43] define that scenarios can be functional, logical, or concrete. A functional scenario describes the entities of the domain and their relation at a semantic level (different levels of abstraction are deemed possible). Logical scenarios represent entities and relations with the help of parameter ranges, i.e., provide an interpretation of the semantic signature on the tempo-spatial structures that represent scenes. Concrete scenarios, finally, are individual instances of tempo-spatial structures with their semantic meaning. These notions are widely accepted in industry and academia today and provide a framework for formulating goals and challenges [44], [45].

Generating scenarios from semantic primitives and developing adequate semantic primitives is approached by Zhang et al. [46] and Medrano-Berumen and Akbas [47]. The first work generates collision-free traffic scenarios by describing road shapes using extracted traffic primitives. Similarly, the second work generates roadways by connecting building blocks (i.e., geometric primitives). In contrast to our work, these works describe scenarios using geometric shapes and calculations to build scenarios, while we describe scenarios with logically defined higher-level predicates.

**Analyzing Real-World Data.** Besides scenario-based testing being widely accepted for testing autonomous systems, evaluations on real-world data are nevertheless mandatory [48], [49]. Especially, as real-world data can be used to find relevant or critical traffic scenarios, which can then be applied to develop scenarios for scenario-based testing [43], [50]. Real-world data can also be utilized to help to understand how human drivers perceive autonomous system failures in real-world situations [51]. Other applications for real-world data are: decision making [52], pedestrian intention estimation [53] or object detection [54]. Nevertheless, real-world data should always be accompanied by exhaustive simulation data [55], as it can model situations that are not feasible to test in the real-world (e.g., accidents).

**Coverage and Metrics**. To check whether a test set is sufficient, coverage criteria are developed. For this, Laurent et al. [56] introduce a coverage criterion for the parameters that are utilized in the decision process of autonomous systems.

Langner et al. [55] automatically detect novel traffic scenarios using a machine-learning approach. Using this, they are able to reduce a given test set to unique test scenarios. Their future work includes labeling of scenarios to further improve the classification of novel scenarios.

A metric for the driveability of scenes is first introduced by Guo et al. [57]. The term *driveability* describes how easy an autonomous vehicle can navigate through a scene. The authors collected explicit and implicit factors that contribute to the driveability from different studies, reports and standards.

Hauer et al. [58] introduce a test ending criterion for testing automated and autonomous driving systems that should help arguing over the safety of autonomous vehicles.

Closest to ours are the following two works: Amersbach and Winner [59] introduce a first approach for scenario coverage by calculating the required number of concrete scenarios regarding specified parameter ranges. They argue that for validating highly automated vehicles a specification of functional scenarios (e.g., lane-change, following, etc.) has to be developed. Li et al. [60] generate abstract scenarios while maximizing the coverage of $k$-way combinatorial testing. Each abstract scenario can be seen as an equivalence class for which a set of concrete scenarios is generated. The categories used for generating the abstract scenarios are similar to the scenario classifiers used in this paper (e.g., weather, road type, ego-action).

## VII. Conclusion

We have presented a logic for expressing features of driving scenarios in a temporal logic and for combining classifiers for features into tree-based scenario classifiers that structure the operational design domain of an autonomous vehicle into relevant scenario classes. Tree-based scenario classifiers enable an analysis of scenario class coverage for recorded driving data. We have evaluated our technique in simulated urban driving experiments. The results show that we are capable of achieving full coverage for some scenario classifiers and can reason about the observed features of the analyzed set of recorded test drives.

## References


[1] R. Mariani, "An overview of autonomous vehicles safety," in *2018 IEEE International Reliability Physics Symposium (IRPS)*. New York: IEEE, 2018, doi: 10.1109/irps.2018.8353618, IRPS 2018.

[2] P. Junietz, W. Wachenfeld, K. Klonecki, and H. Winner, "Evaluation of different approaches to address safety validation of automated driving," in *21st International Conference on Intelligent Transportation Systems, ITSC 2018, Maui, HI, USA, November 4-7, 2018*. New York: IEEE, 2018, pp. 491–496, doi: 10.1109/itsc.2018.8569959, ITSC 2018.

[3] N. Kalra and 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*, vol. 94, pp. 182–193, 2016, doi: 10.1016/j.tra.2016.09.010.

[4] M. Mauritz, F. Howar, and 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*. Springer International Publishing, 2016, pp. 672–687, doi: 10.1007/978-3-319-47169-3_52.

[5] H. Felbinger, F. Kluck, Y. Li, M. Nica, J. Tao, F. Wotawa, and M. Zimmermann, "Comparing two systematic approaches for testing automated driving functions," in *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*. New York: IEEE, 2019, doi: 10.1109/iccve45908.2019.8965209, ICCVE 2019.

[6] ISO Central Secretary, "Road vehicles - safety of the intended functionality," International Organization for Standardization, Geneva, CH, Standard ISO 21448:2022, 2022. [Online]. Available: https://www.iso.org/standard/77490.html

[7] ISO Central Secretary, "Road vehicles - functional safety - part 1: Vocabulary," International Organization for Standardization, Geneva, CH, Standard ISO 26262-1:2018, 2018. [Online]. Available: https://www.iso.org/standard/68383.html

[8] 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)*. New York: IEEE, 2020, pp. 1–5, doi: 10.1109/SysCon47679.2020.9275888, SysCon 2020.

[9] B. Schütt, M. Steimle, B. Kramer, D. Behnecke, and E. Sax, "A taxonomy for quality in simulation-based development and testing of automated driving systems," *IEEE Access*, vol. 10, pp. 18 631–18 644, 2022, doi: 10.1109/ACCESS.2022.3149542.

[10] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," arXiv, Tech. Rep. arXiv:1708.06374, 2018, doi: 10.48550/arXiv.1708.06374.

[11] F. Schuldt, F. Saust, B. Lichte, and M. Maurer, "Effiziente systematische Testgenerierung für Fahrerassistenzsysteme in virtuellen Umgebungen," doi: 10.24355/dbbs.084-201307101421-0, 2013. [Online]. Available: https://nbn-resolving.org/urn:nbn:de:gbv:084-13071014254

[12] 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, 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, 2021, doi: 10.1109/access.2021.3072739.

[13] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, 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*. New York: IEEE, 2015, doi: 10.1109/itsc.2015.164, ITSC 2015.

[14] D. Nalic, T. Mihalj, M. Bäumler, M. Lehmann, A. Eichberger, and S. Bernsteiner, "Scenario based testing of automated driving systems: A literature survey," 2020, doi: 10.46720/f2020-acm-096, FISITA Web Congress 2020 ; Conference date: 24-11-2020 Through 24-11-2020. [Online]. Available: https://www.fisita.com/library/fisita-world-congress/2020/f2020-acm-096

[15] 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, 2020, doi: 10.1109/ACCESS.2020.2993730.

[16] The British Standards Institution, "Operational Design Domain (ODD) taxonomy for an automated driving system (ADS) – Specification," Specification PAS 1883:2020, 2020. [Online]. Available: https://www.bsigroup.com/globalassets/localfiles/en-gb/cav/pas1883.pdf

[17] Association for Standardization of Automation and Measuring Systems, "ASAM OpenODD: Concept Paper," 2021, version 1.0. [Online]. Available: https://www.asam.net/index.php?eID=dumpFile&t=f&f=4544&token=1260ce1c4f0afdbe18261f7137c689b1d9c27576

[18] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., vol. 78. PMLR, 13–15 Nov 2017, pp. 1–16. [Online]. Available: https://proceedings.mlr.press/v78/dosovitskiy17a.html

[19] P.-Y. Schobbens, P. Heymans, and J.-C. Trigaux, "Feature diagrams: A survey and a formal semantics," in *14th IEEE International Requirements Engineering Conference (RE'06)*. New York: IEEE, 2006, doi: 10.1109/re.2006.23, RE 2006.

[20] T. Schallau, S. Naujokat, F. Kullmann, and F. Howar, "Tree-Based Scenario Classification: A Formal Framework for Coverage Analysis on Test Drives of Autonomous Vehicles - Replication Artifact," doi: 10.5281/zenodo.8131947, jul 2023.

[21] S. Müller, "Theory and applications of runtime monitoring metric first-order temporal logic," Ph.D. dissertation, ETH Zurich, 2009.

[22] D. Basin, F. Klaedtke, S. Müller, and E. Zălinescu, "Monitoring metric first-order temporal properties," *J. ACM*, vol. 62, no. 2, may 2015, doi: 10.1145/2699444.

[23] R. J. Wieringa, "Single-case mechanism experiments," in *Design Science Methodology for Information Systems and Software Engineering*. Springer, 2014, pp. 247–267, doi: 10.1007/978-3-662-43839-8_18.

[24] S. Maierhofer, A.-K. Rettinger, E. C. Mayer, and M. Althoff, "Formalization of interstate traffic rules in temporal logic," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. New York: IEEE, 2020, pp. 752–759, doi: 10.1109/IV47402.2020.9304549, IV 2020.

[25] G. Velasco-Hernandez, D. J. Yeong, J. Barry, and J. Walsh, "Autonomous driving architectures, perception and data fusion: A review," in *2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP)*. New York: IEEE, 2020, doi: 10.1109/iccp51029.2020.9266268, ICCP 2020.

[26] S. Fadadu, S. Pandey, D. Hegde, Y. Shi, F.-C. Chou, N. Djuric, and C. Vallespi-Gonzalez, "Multi-view fusion of sensor data for improved perception and prediction in autonomous driving," in *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. New York: IEEE, 2022, doi: 10.1109/wacv51458.2022.00335, WACV 2022.

[27] R. Philipp, Z. Zhu, J. Fuchs, L. Hartjen, F. Schuldt, and F. Howar, "Automated 3d object reference generation for the evaluation of autonomous vehicle perception," in *2021 5th International Conference on System Reliability and Safety (ICSRS)*. New York: IEEE, 2021, doi: 10.1109/icsrs53853.2021.9660660, ICSRS 2021.

[28] K. Esterle, L. Gressenbuch, and A. C. Knoll, "Formalizing traffic rules for machine interpretability," in *2020 IEEE 3rd Connected and Automated Vehicles Symposium (CAVS)*. IEEE, 2020, pp. 1–7, doi: 10.1109/CAVS51000.2020.9334599, CAVS 2020.

[29] A. Rizaldi, J. Keinholz, M. Huber, J. Feldle, F. Immler, M. Althoff, E. Hilgendorf, and T. Nipkow, "Formalising and monitoring traffic rules for autonomous vehicles in isabelle/HOL," in *Lecture Notes in Computer Science*. Springer International Publishing, 2017, pp. 50–66, doi: 10.1007/978-3-319-66845-1_4.

[30] S. Maierhofer, P. Moosbrugger, and M. Althoff, "Formalization of intersection traffic rules in temporal logic," in *2022 IEEE Intelligent Vehicles Symposium (IV)*. New York: IEEE, 2022, doi: 10.1109/iv51971.2022.9827153, IV 2022.

[31] A. Karimi and P. S. Duggirala, "Formalizing traffic rules for uncontrolled intersections," in *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems ICCPS*. New York: IEEE, 2020, pp. 41–50, doi: 10.1109/iccps48487.2020.00012, ICCPS 2020.

[32] H. Weber, J. Bock, J. Klimke, C. Roesener, J. Hiller, R. Krajewski, A. Zlocki, and L. Eckstein, "A framework for definition of logical scenarios for safety assurance of automated driving," *Traffic Injury Prevention*, vol. 20, no. sup1, pp. S65–S70, 2019, doi: 10.1080/15389588.2019.1630827.

[33] B. Weng, L. Capito, U. Ozguner, and K. Redmill, "Towards guaranteed safety assurance of automated driving systems with scenario sampling: An invariant set perspective," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 638–651, 2022, doi: 10.1109/tiv.2021.3117049.

[34] F. Bock, C. Sippl, A. Heinz, C. Lauer, and R. German, "Advantageous usage of textual domain-specific languages for scenario-driven development of automated driving functions," in *2019 IEEE International Systems Conference (SysCon)*. New York: IEEE, 2019, doi: 10.1109/syscon.2019.8836912, SysCon 2019.

[35] E. Rocklage, H. Kraft, A. Karatas, and J. Seewig, "Automated scenario generation for regression testing of autonomous vehicles," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. New York: IEEE, 2017, doi: 10.1109/itsc.2017.8317919, ITSC 2017.

[36] A. Li, S. Chen, L. Sun, N. Zheng, M. Tomizuka, and W. Zhan, "SceGene: Bio-inspired traffic scenario generation for autonomous driving testing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 14 859–14 874, 2022, doi: 10.1109/tits.2021.3134661.

[37] I. R. Jenkins, L. O. Gee, A. Knauss, H. Yin, and J. Schroeder, "Accident scenario generation with recurrent neural networks," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. New York: IEEE, 2018, doi: 10.1109/itsc.2018.8569661, ITSC 2018.

[38] J. Sun, H. Zhang, H. Zhou, R. Yu, and 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*, vol. 23, no. 9, pp. 14 088–14 103, 2022, doi: 10.1109/tits.2021.3136353.

[39] J. Zhao, J. Fang, Z. Ye, and L. Zhang, "Large scale autonomous driving scenarios clustering with self-supervised feature extraction," in *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2021, doi: 10.1109/iv48863.2021.9575644.

[40] R. B. Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing vision-based control systems using learnable evolutionary algorithms," in *Proceedings of the 40th International Conference on Software Engineering*. ACM, 2018, doi: 10.1145/3180155.3180160.

[41] M. Steimle, T. Menzel, and 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*, vol. 9, pp. 147 828–147 854, 2021, doi: 10.1109/access.2021.3123504.

[42] M. Klischat and M. Althoff, "Generating critical test scenarios for automated vehicles with evolutionary algorithms," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. New York: IEEE, 2019, pp. 2352–2358, doi: 10.1109/ivs.2019.8814230, IV 2019.

[43] T. Menzel, G. Bagschik, and M. Maurer, "Scenarios for development, test and validation of automated vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. New York: IEEE, 2018, pp. 1821–1827, doi: 10.1109/IVS.2018.8500406, IV 2018.

[44] T. Menzel, G. Bagschik, L. Isensee, A. Schomburg, and 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)*. New York: IEEE, 2019, doi: 10.1109/ivs.2019.8814099, IV 2019.

[45] L. Elster, C. Linnhoff, P. Rosenberger, S. Schmidt, R. Stark, and 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)*. New York: IEEE, 2021, doi: 10.1109/ivworkshops54471.2021.9669207, IV Workshops, 2021.

[46] W. Zhang, W. Wang, J. Zhu, and D. Zhao, "Multi-vehicle interaction scenarios generation with interpretable traffic primitives and gaussian process regression," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. New York: IEEE, 2020, doi: 10.1109/iv47402.2020.9304568, IV 2020.

[47] C. Medrano-Berumen and M. I. Akbaş, "Abstract simulation scenario generation for autonomous vehicle verification," in *2019 SoutheastCon*. New York: IEEE, 2019, pp. 1–6, doi: 10.1109/SoutheastCon42311.2019.9020575.

[48] L. Klitzke, C. Koch, A. Haja, and 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*. SCITEPRESS - Science and Technology Publications, 2019, doi: 10.5220/0007720501710180.

[49] H. Winner, K. Lemmer, T. Form, and J. Mazzega, "PEGASUS—first steps for the safe introduction of automated driving," in *Lecture Notes in Mobility*. Springer International Publishing, 2018, pp. 185–195, doi: 10.1007/978-3-319-94896-6_16.

[50] J. Bach, S. Otten, and E. Sax, "Model based scenario specification for development and test of automated driving functions," in *2016 IEEE Intelligent Vehicles Symposium (IV)*. New York: IEEE, 2016, doi: 10.1109/ivs.2016.7535534, IV 2016.

[51] M. Dikmen and C. M. Burns, "Autonomous driving in the real world," in *Proceedings of the 8th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. ACM, 2016, doi: 10.1145/3003715.3005465.

[52] A. Furda and L. Vlacic, "Enabling safe autonomous driving in real-world city traffic using multiple criteria decision making," *IEEE Intelligent Transportation Systems Magazine*, vol. 3, no. 1, pp. 4–17, 2011, doi: 10.1109/mits.2011.940472.

[53] W. M. Alvarez, F. M. Moreno, O. Sipele, N. Smirnov, and C. Olaverri-Monreal, "Autonomous driving: Framework for pedestrian intention estimation in a real world scenario," in *2020 IEEE Intelligent Vehicles

*Symposium (IV)*. New York: IEEE, 2020, doi: 10.1109/iv47402.2020.9304624, IV 2020.
[54] 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, and H. Xu, "CODA: A real-world road corner case dataset for object detection in autonomous driving," in *Lecture Notes in Computer Science*. Springer Nature Switzerland, 2022, pp. 406–423, doi: 10.1007/978-3-031-19839-7_24.
[55] J. Langner, J. Bach, L. Ries, S. Otten, M. Holzapfel, and E. Sax, "Estimating the uniqueness of test scenarios derived from recorded real-world-driving-data using autoencoders," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. New York: IEEE, 2018, doi: 10.1109/ivs.2018.8500464, IV 2018.
[56] T. Laurent, S. Klikovits, P. Arcaini, F. Ishikawa, and A. Ventresque, "Parameter coverage for testing of autonomous driving systems under uncertainty," *ACM Transactions on Software Engineering and Methodology*, 2022, doi: 10.1145/3550270.
[57] J. Guo, U. Kurup, and M. Shah, "Is it safe to drive? an overview of factors, metrics, and datasets for driveability assessment in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, pp. 3135–3151, 2020, doi: 10.1109/tits.2019.2926042.
[58] F. Hauer, T. Schmidt, B. Holzmüller, and A. Pretschner, "Did we test all scenarios for automated and autonomous driving systems?" in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. New York: IEEE, 2019, doi: 10.1109/itsc.2019.8917326, ITSC 2019.
[59] C. Amersbach and H. Winner, "Defining required and feasible test coverage for scenario-based validation of highly automated vehicles," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. New York: IEEE, 2019, pp. 425–430, doi: 10.1109/itsc.2019.8917534, ITSC 2019.
[60] C. Li, C.-H. Cheng, T. Sun, Y. Chen, and R. Yan, "ComOpT: Combination and optimization for testing autonomous driving systems," in *2022 International Conference on Robotics and Automation (ICRA)*. New York: IEEE, 2022, doi: 10.1109/icra46639.2022.9811794, ICRA 2022.

**Till Schallau** received the M.Sc. degree in Computer Science from TU University, Germany, in 2019. Since then, he has been a research associate (Ph.D.) at the Chair for Software Engineering in Prof. Dr. Falk Howar's group, TU Dortmund University, Germany. His current research interests include the formal specification of scenario-based testing of autonomous systems as well as utilizing domain-specific languages.

**Stefan Naujokat** received the Dipl.-Inf. and Dr.-Ing. degrees from the Department of Computer Science at TU Dortmund University, Germany, in 2009 and 2017, respectively. He was a Ph.D. student and postdoc at the Chair for Programming Systems until 2019 and currently is senior researcher at the Chair for Software Engineering in Prof. Dr. Falk Howar's group, TU Dortmund University, Germany. His dissertation focused on simplifying language workbenches to make the development of visual domain-specific languages more accessible to a wider audience of language engineers. His research interests furthermore include metamodeling, code generation, and bridging the gap between formal specifications and domain-specific languages.

**Fiona Kullmann** received the B.Sc. degree in Computer Science from TU Dortmund University, Germany, in 2021. She started working for the Automated Quality Assurance group under Prof. Dr. Falk Howar in the same year. She is currently studying for her M.Sc. degree at TU Dortmund University, participating in its Formula Student racing team by developing autonomous driving capabilities.

**Falk Howar** is a professor for Software Engineering in the Department of Computer Science at TU Dortmund University and Coordinator of Software Engineering Research at Fraunhofer ISST. His research focuses on the safety and security of intelligent and autonomous software systems. He is particularly interested in the use of formal methods to analyze the behavior of such systems. After studying computer science and earning his doctorate at TU Dortmund University, Falk Howar first worked in the USA at Carnegie Mellon University (Silicon Valley) and at NASA Ames Research Center, where he developed methods for testing an autonomous air traffic control system. Subsequently, he was on the management board of the Institute for Applied Software Systems Engineering at TU Clausthal. There, he conducted research on safe autonomous driving functions together with partners from the automotive industry.