Tree-Based Scenario Classification

A Formal Framework for Measuring Domain Coverage when Testing Autonomous Systems

Till Schallau[1][0000−0002−1769−3486], Stefan Naujokat[1][0000−0002−6265−6641], Fiona Kullmann[1][0000−0001−5858−0659], and Falk Howar[1,2][0000−0002−9524−4459]

[1] TU Dortmund University, Dortmund, Germany
till.schallau@tu-dortmund.de
[2] Fraunhofer ISST, Dortmund, Germany

Abstract. Scenario-based testing is envisioned as a key approach for the safety assurance of automated driving systems. 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 recorded test drives into such scenarios and measuring scenario coverage in these test drives. Technically, we specify features in logic formulas over complex data streams and construct tree-based classifiers for scenarios from these feature specifications. For such specifications, we introduce CMFTBL, a new logic that extends existing linear-time temporal logics with aspects that are essential for concise specifications that work on field-recorded data. We demonstrate the expressiveness and effectiveness of our approach by defining a family of related scenario classifiers for different aspects of urban driving.

1 Introduction

One of the open challenges in developing automated driving systems is assuring their safety [20]. For several years, research has focused on structured approaches to assure the safety of autonomous driving systems [9, 21]. The recently published ISO 21448 [14] norm (Safety of the Intended Functionality) transfers the conceptual framework of system safety approaches (e.g., ISO 26262 [13]) to the assurance of a vehicle's safety under all environmental conditions and possible faults that are triggered by the environment [25]. 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 [33], formalizing what constitutes scenarios [31, 35], and on testing safety in specified scenarios [23].

Recent standardization efforts target the specification of so-called operational design domains (ODDs) [34] that define the anticipated environmental conditions for autonomous driving systems 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 driving system 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 [2].

In this paper, we present an approach to the specification of scenario sets through the definition and recognition of scenario features. These features can then be used to classify 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 specify features in logic formulas over complex data streams and construct tree-based classifiers from these feature specifications 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 [31], and in the classification of driving maneuvers (e.g., intersection, sunny, oncoming traffic, left-turn maneuver, etc.).

For the demonstration of the expressiveness and effectiveness of our approach, we conduct 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 demonstrate how to decompose and analyze coverage for individual features or layers of the 6-layer model.

Summarizing, the contribution of this paper is threefold:

1. We introduce the Counting Metric First Order Temporal Binding Logic (CMFTBL), 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 us to express imprecise specifications (in the spirit of "most of the time"); secondly, it is defined over complex structured domains for capturing scenes, and thirdly, it allows binding term evaluations to variables for access in nested parts of formulas (cf. Sect. 3).
2. We present a method for specifying and 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 [30], where features of scenarios are specified using the CMFTBL logic (cf. Sect. 4.1).
3. The specification of features for sets of scenarios enables 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 and not on the parameters of one scenario (cf. Sect. 4.2).

Related Work. The presented approach is related to various existing works on the safety of automated driving systems, as well as works that utilize logic for specifying properties of safety-critical systems.

Amersbach and Winner [1] present an approach for estimating the required number of concrete scenarios for achieving scenario coverage based on real-world driving and accident data. Their analysis shows that more scenarios are required than would be feasible to test. Therefore, they propose to group concrete parameter combinations in so-called functional (i.e., abstract) scenario specifications (e.g., lane-change, following). Similar to this, Jafer et al. [15] introduce the Aviation Scenario Definition Language (ASDL), which allows for graphically specifying abstract and concrete flight scenarios. The conformance of the defined scenarios to predefined rules is verified using state charts and a model checker [4]. Our work provides classifiers for such abstract features of a scenario.

Li et al. [17] generate abstract scenarios with the goal of maximizing the coverage of k-way combinatorial testing for various scenario categories (e.g., weather, road type, ego-action). While their scenario categories are related to our scenario features, they do not provide classifiers for categories but aim to generate concrete instances of abstract scenarios.

Variants of temporal logics are commonly used for specifying properties of autonomous systems and their operational environments: Esterle et al. [7] formalize traffic rules for highway situations using Linear Temporal Logic (LTL). Rizaldi et al. [24] also formalize German overtaking rules using the same logic. Others formalize similar traffic rules, such as interstate traffic [19] or intersections [18], using the Metric Temporal Logic (MTL), as LTL is not capable of modeling duration. In a different direction of works, Schumann et al. [32] use MTL and LTL in their R2U2 framework for runtime System Health Management. Johannsen et al. [16] demonstrate how they extend this approach to transform previously intractable "first order" specifications (i.e., self-reference, unboundedness, and explicit counting) into Mission-time Linear Temporal Logic (MLTL) formulas without the need for extensions. This is done by parameterizing specifications based on restrictions of the evaluation domain. Dokhanchi et al. [5] introduce a *freeze frame quantifier* in their Timed Quality Temporal Logic (TQTL) to refer to objects in some time frame. We use a similar operator for modeling relations of objects over time (e.g., lane changes). Finally, the FRET framework [10] internally also works with LTL formulas. It can be used to define and analyze formal requirements in the restricted natural language FRETISH [11]. Similar to our goal of providing concise specification mechanisms, this work aims at making formal specification languages accessible to domain experts.

Outline. The paper is structured as follows. Section 2 outlines an example that motivates our approach. The formal logic for defining scenario properties is introduced in Sect. 3. Section 4 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. 5. The paper concludes in Sect. 6.

Extended Version, Implementation, and Reproduction Package. We provide an extended version of this paper [28], the open-source framework implementation[3], and a reproduction package on Zenodo [29]. For more technical introductions to the framework, please refer to [26] and [27].

[3] `https://github.com/tudo-aqua/stars`

2 Motivational Example

We illustrate our approach to analyzing test drives in an urban environment. For this, we utilize a database of recorded test drives as the foundation. Recordings consist of sequences of *scenes* that are split into meaningful *segments*, e.g., based on regions of a map. A single scene is the snapshot of the state and the 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. We record segments (i.e., sequences of scenes) at fixed intervals.

Our task is to decide whether this database of test drives covers sufficiently many relevant *scenarios* (i.e., archetypes of driving situations), or at least classify the test drives to identify 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 with three road users on a T-junction. The ego vehicle is planning to turn left. It currently stops at the stop line of the stop sign on the ego vehicle's lane. The other car follows its lane, going straight over the junction, and crossing the trajectory of the ego vehicle. The ego vehicle's destination lane contains a crosswalk with a pedestrian currently crossing the road. Thus, the pedestrian is crossing the trajectory of the ego vehicle.

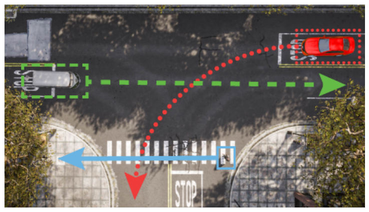


Fig. 1: Simulation in CARLA [6]: the ego vehicle (red, dotted) stops at a stop sign. The trajectory of its planned left turn is crossed by an oncoming vehicle (green, dashed) and by a pedestrian (blue, solid).

When analyzing the segment, we can observe specific maneuvers, environmental properties, and features 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 logic over sequences of scenes. Then, we can use a set of features to classify segments: the combination of identified features (i.e., corresponding formulas that hold) defines the scenario class. The segment is one concrete *instance* of this scenario class. Assuming that features are not entailed by other features, we obtain 2^n scenario classes from 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. 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*. Based on these three variations (neglecting the other properties for the sake of simplicity), a total of $2^3 = 8$ possible scenario classes are observable. We can use this information to compute missing scenario classes or to measure scenario class coverage on a set of test drives. In our example, one scenario class was observed. Given the eight possible scenario classes, we obtain a scenario class coverage of 12.5%. Section 4 formalizes these concepts.

3 A Temporal Logic for Properties of Scenarios

We base scenario classifiers on the environment representation that is usually produced by the perception sub-system of an automated vehicle: a map of the road network and typed objects with positions, velocities, and observed states. We express properties of recorded sequences of scenes (i.e., momentary snapshots of the environment of the ego vehicle) in a formal logic. 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 extend Metric First-Order Temporal Logic (MFOTL) [3] with a *minimum prevalence operator* that allows to express that a property holds for a certain fraction of all future states (within a finite trace), a corresponding past time version, and a *binding operator* that binds the value of a term at the current time to a variable. While the prevalence operators extend the expressiveness of MFOTL, binding is a shorthand for existentially quantified formulas of a certain form. We also introduce evaluation over complex data structures (in the form of further notational shorthands and conventions) to more easily capture the properties of scenes. We name our extension Counting Metric First-Order Temporal Binding Logic (CMFTBL).

A signature σ 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 σ-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 $I(f) : \mathcal{D}^{\mathrm{ar}(f)} \to \mathcal{D}$ for $f \in \mathcal{F}$, and $I(r) \subseteq \mathcal{D}^{\mathrm{ar}(r)}$ for $r \in \mathcal{R}$. We write an interval of the set of non-empty intervals $\mathcal{I}$ over $\mathbb{N}$ as $[b, b') := \{a \in \mathbb{N} | b \leq a < b'\}$, where $b \in \mathbb{N}, b' \in \mathbb{N} \cup \{\infty\}$ and $b < b'$.

Formulas in CMFTBL over the signature σ, intervals $\mathcal{I}$, and the countably infinite set of variables $\mathcal{V}$ $\big($assuming $\mathcal{V} \cap (\mathcal{C} \cup \mathcal{F} \cup \mathcal{R}) = \emptyset\big)$ 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, φ and ψ are formulas, then $(\neg\varphi)$, $(\varphi \vee \psi)$, $(\exists x : \varphi)$, and $(\downarrow^t_x \varphi)$ are formulas, where $\downarrow^t_x$ 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 φ and ψ are formulas, then next $(\bigcirc_I \varphi)$, previously $(\bullet_I \varphi)$, until $(\varphi\ U_I\ \psi)$, since $(\varphi\ S_I\ \psi)$, min. prevalence $(\triangledown^p_I \psi)$, and past min. prevalence $(\blacktriangledown^p_I \psi)$ are formulas.

The pair $\langle \boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}\rangle$ is a *finite temporal structure* over the signature σ, where $\boldsymbol{\mathfrak{D}} = (\mathfrak{D}_0, \mathfrak{D}_1, \cdots, \mathfrak{D}_n)$ is a finite sequence of structures (i.e., scenes) over σ and

$\boldsymbol{\tau} = (\tau_0, \tau_1, \cdots, \tau_n)$ is a finite sequence of non-negative rational numbers $\tau_i \in \mathbb{Q}^+$ with length n. The elements in the sequence $\boldsymbol{\tau}$ are (increasing) *timestamps*. Furthermore, the interpretations of relations $r^{\mathfrak{D}_0}, r^{\mathfrak{D}_1}, \cdots, r^{\mathfrak{D}_n}$ in a temporal structure $\langle \boldsymbol{\mathfrak{D}}, \boldsymbol{\tau} \rangle$ 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 $v : \mathcal{V} \to \mathcal{D}$ 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 a 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 the 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 $(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, i) \models \varphi$ inductively, as shown in Table 1, where $|\boldsymbol{\tau}|$ denotes the number of timestamps in the temporal structure $\langle \boldsymbol{\mathfrak{D}}, \boldsymbol{\tau} \rangle$ and is mostly used as an upper bound for intervals of temporal operators. The temporal structure $\langle \boldsymbol{\mathfrak{D}}, \boldsymbol{\tau} \rangle$ satisfies formula φ iff $(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, \emptyset, 0) \models \varphi$.

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

$(\varphi \wedge \psi)$	$:= \Big(\neg\big((\neg\varphi) \vee (\neg\psi)\big)\Big)$	logical and
$(\varphi \Rightarrow \psi)$	$:= \big((\neg\varphi) \vee \psi\big)$	implication
$(\forall x : \varphi)$	$:= \big(\neg(\exists x : \neg\varphi)\big)$	all quantifier
$(\Diamond_I \varphi)$	$:= (\top\ U_I\ \varphi)$	eventually
$(\Box_I \varphi)$	$:= \Big(\neg\big(\Diamond_I(\neg\varphi)\big)\Big)$	always
$(\blacklozenge_I \varphi)$	$:= (\top\ S_I\ \varphi)$	once
$(\blacksquare_I \varphi)$	$:= \Big(\neg\big(\blacklozenge_I(\neg\varphi)\big)\Big)$	historically

Analogously, we define maximum prevalence operators based on minimum prevalence:

$(\triangle_I^p \varphi)$	$:= (\triangledown_I^{1-p} \neg\varphi)$	max. prevalence
$(\blacktriangle_I^p \varphi)$	$:= (\blacktriangledown_I^{1-p} \neg\varphi)$	past max. prevalence

We obtain non-metric variants of the temporal operators for interval $[0, \infty)$. In addition to these commonly used patterns, we introduce several notational conventions to ease presentation.

$(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, i) \models r(t_1, ..., t_{a(r)})$	iff $\big(\beta(t_1, v, \mathfrak{D}_i), \cdots, \beta(t_{\mathrm{ar}(r)}, v, \mathfrak{D}_i)\big) \in r^{\mathfrak{D}_i}$
$(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, i) \models (\neg\psi)$	iff $(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, i) \not\models \psi$
$(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, i) \models (\psi \vee \psi')$	iff $(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, i) \models \psi$ or $(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, i) \models \psi'$
$(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, i) \models (\exists x : \psi)$	iff $(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v[x \mapsto d], i) \models \psi$, for some $d \in \mathcal{D}$
$(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, i) \models (\bigcirc_I \psi)$	iff $\tau_{i+1} - \tau_i \in I$ and $(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, i+1) \models \psi$
$(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, i) \models (\bullet_I \psi)$	iff $i > 0$, $\tau_i - \tau_{i-1} \in I$, and $(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, i-1) \models \psi$
$(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, i) \models (\psi\ U_I\ \psi')$	iff for some $j \geq i, \tau_j - \tau_i \in I, (\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, j) \models \psi'$,
	and $(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, k) \models \psi$, for all $k \in \mathbb{N}$ with $i \leq k < j$
$(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, i) \models (\psi\ S_I\ \psi')$	iff for some $j \leq i, \tau_i - \tau_j \in I, (\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, j) \models \psi'$,
	and $(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, k) \models \psi$, for all $k \in \mathbb{N}$ with $j < k \leq i$
$(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, i) \models (\triangledown_I^p \psi)$	iff $(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, j) \models \psi$, for at least fraction p
	of indices $i \leq j \leq \lvert\boldsymbol{\tau}\rvert$ with $\tau_j - \tau_i \in I$
$(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, i) \models (\blacktriangledown_I^p \psi)$	iff $(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, j) \models \psi$, for at least fraction p
	of indices $0 \leq j \leq i$ with $\tau_i - \tau_j \in I$
$(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, i) \models (\downarrow_x^t\ \psi)$	iff $(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v[x \mapsto \beta(t, v, \mathfrak{D}_i)], i) \models \psi$

Table 1: Definition of the semantics of the Counting Metric First-Order Temporal Logic (CMFTBL), given inductively by the relation $(\boldsymbol{\mathfrak{D}}, \boldsymbol{\tau}, v, i) \models \psi$. The newly introduced operators are *minimum prevalence* $(\triangledown_I^p \psi)$, *past minimum prevalence* $(\blacktriangledown_I^p \psi)$, and *binding* $(\downarrow_x^t\ \psi)$.

Let $isVehicle \in \mathcal{R}$ be a unary relation. We define the set of all vehicles $\mathcal{V} \subseteq \mathcal{D}$ as $\{d \in \mathcal{D} \mid isVehicle(d)\}$. Analogously, we define the set of pedestrians as $\mathcal{P}$, and the set of actors $\mathcal{A} := \mathcal{P} \cup \mathcal{V}$ with $\mathcal{P} \cap \mathcal{V} = \emptyset$. We also use a notation reminiscent of object-relational element associations. For instance, for some vehicle $v \in \mathcal{V}$ and the relations $\{isEgo, isLane, onLane\} \subseteq \mathcal{R}$, we use shorthand notations like, e.g., $v.isEgo$ for $isEgo(v)$ and $v.lane$ for the unique $l \in \mathcal{D}$ where $isLane(l) \wedge onLane(v, l)$.

Formulas used for specifying features in a tree-based classifier usually consist of one unary relation that needs to hold for the ego vehicle. For such an $r \in \mathcal{R}$ and $\varphi := \exists v \in \mathcal{V} : \Box(v.isEgo) \wedge r(v)$, we write $ego.r$ instead of the entire formula φ. As an example, consider *ego.obeyedSpeedLimit* to model that the ego vehicle obeys the speed limit at all times. Using our notational conventions, the relation is defined as follows:

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

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 concisely specify traffic rules and environmental features using CMFTBL formulas and evaluate those on sequences of scenes. We express each predicate of our case study (cf. Sect. 5) in this way. For comparison, consider the formula for "the ego vehicle obeys the speed limit at all times" without these syntactic conventions:

$$\begin{aligned}\varphi_{osl} := \exists v \in \mathcal{D} : \square\big(isVehicle(v) \wedge isEgo(v)\big)\wedge \\ \square\Big(\exists l \in \mathcal{D} : \exists p \in \mathcal{D} : isLane(l) \wedge onLane(v,l)\wedge \\ \wedge\, leq\big(speed(v), speedLimitAt(p,l)\big)\Big)\end{aligned}$$

We conclude this section with two example formulas from our case study that utilize the operators introduced by CMFTBL. Consider a relation *isInJunction* that decides whether a vehicle $v \in \mathcal{V}$ primarily drives through a junction during the analyzed segment. For this, we require the road the vehicle drives on to be categorized as a junction "for the most part" (e.g., 80%) of the segment. To express this, we need the newly introduced minimum prevalence operator.

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

We utilize the binding operator to detect a lane change for a given vehicle $v \in \mathcal{V}$. We bind the lane of vehicle v at the current timestamp to a new variable l. In nested formulas, which might evaluate other (here, later) timestamps, we can compare the vehicle's lane value to l to detect a lane change.

$$changedLane(v) := \downarrow_l^{v.lane} \big(\lozenge(l \neq v.lane)\big) \tag{2}$$

4 Classifiers for Scenarios and Metrics on Sets of Scenarios

We want to use CMFTBL formulas to express features of scenarios and classify recorded driving data into scenario classes. Formally, we assume recorded driving data to be given as temporal structures $\langle \boldsymbol{\mathfrak{D}}, \boldsymbol{\tau} \rangle$ over some fix basic signature $\langle \mathcal{C}, \mathcal{F}, \mathcal{R}, \mathrm{ar} \rangle$. This basic signature encodes 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. We use $\mathfrak{S}$ to denote a set of segments of form $\langle \boldsymbol{\mathfrak{D}}, \boldsymbol{\tau} \rangle$. In practice, segmentation could either be done based on a map or based on classification.

4.1 Classifiers for Scenarios

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

We write $q \xrightarrow{\varphi} q'$ for (q, q', φ). 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 $0 \leq \lambda_l(q) \leq \lambda_u(q) \leq |c(q)|$.

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):

(A)ll	$\lambda_l(q) = \lambda_u(q) = \lvert c(q)\rvert$	**(O)ptional**	$\lambda_l(q) = 0 \wedge \lambda_u(q) = \lvert c(q)\rvert$
E(X)clusive	$\lambda_l(q) = \lambda_u(q) = 1$	**(a..b)-Bounded**	$\lambda_l(q) = a \wedge \lambda_u(q) = b$
Leaf ()	$\lambda_l(q) = \lambda_u(q) = 0$		

We introduce these bounds on sub-features to compute the number of combinatorial combinations, i.e., the number of observable scenario classes (cf. Sect. 4.2). A more precise approach to computing possible 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. A much more useful analysis in the context of our approach (which it does indeed provide), is to determine which segments (if any) are classified with feature combinations that are invalid according to the explicit definitions in the TSC. This information can then be used to refine the tree or find errors in the data.

Figure 2 shows one of the TSCs developed for our case study (cf. Sect. 5) as a simple example. All nodes (except for the root node) model features of urban driving scenarios. In this example, we focus on different features corresponding to road types. The nodes are labeled with a short description of the feature, followed by the node type (that defines the upper and lower bounds for the number of children in classes modeled by this TSC) in parentheses. Each edge from source to destination is labeled with a formula to recognize the destination node's feature. For better readability of the figure, we omit all formulas except for the always true $\top$ as well as two references to the formulas (1) and (2) presented in Sect. 3.

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 $q_r = q_r^o$, and set of edges Γ of type (q, q') 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 $\langle \boldsymbol{\mathfrak{D}}, \boldsymbol{\tau} \rangle$. We

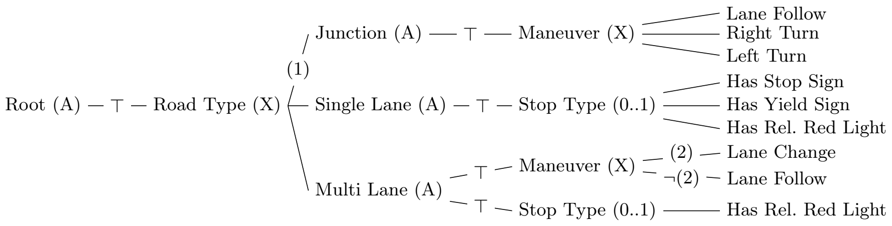


Fig. 2: Simple example classifier from our case study. Edge labels (1) and (2) refer to the formulas shown in Sect. 3.

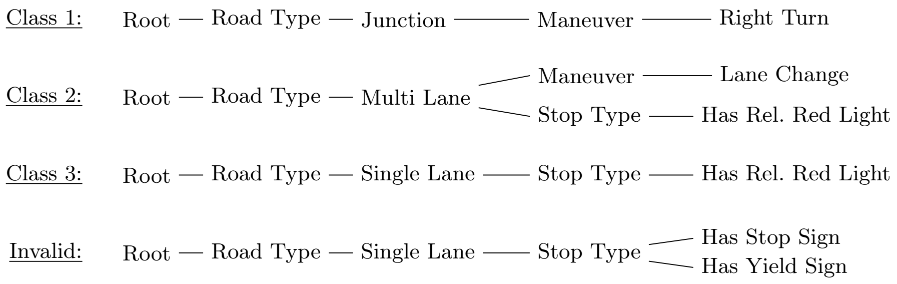

Class 1:
Root — Road Type — Junction — Maneuver — Right Turn
Class 2:
Root — Road Type — Multi Lane
Maneuver — Lane Change
Stop Type — Has Rel. Red Light
Class 3:
Root — Road Type — Single Lane — Stop Type — Has Rel. Red Light
Invalid:
Root — Road Type — Single Lane — Stop Type
Has Stop Sign
Has Yield Sign

Fig. 3: Scenario classes (3 of the total 11) modeled by the TSC shown in Fig. 2. The invalid class violates the bounds of the single lane *Stop Type (0..1)* node.

denote the classification function that maps observed driving data $\langle \mathfrak{D}, \boldsymbol{\tau} \rangle$ 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 $\mathcal{S} = \langle \mathfrak{D}, \boldsymbol{\tau} \rangle$, 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 a valid class to 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 $\mathfrak{S}$.

In total, the TSC presented in Fig. 2 models 11 different scenario classes, i.e., subsets of feature nodes that comply with the bounds given by the node types. As the main node *Road Type* is an (X)-node (with bounds $\lambda_l = \lambda_u = 1$), the possible scenario classes are the 3 junction maneuvers plus the 4 single lane stop type variants (stop sign, yield sign, relevant red light, or none of these) plus 4 possible combinations of multi-lane features (lane change with relevant red light, lane change without relevant red light, etc.). Figure 3 shows three example classes as well as one invalid class where stop sign and yield sign are both recognized at the same time, which violates the bounds of the single lane *Stop Type (0..1)* node. Finding such invalid classes in the analyzed data either indicates errors in the TSC modeling (i.e., in the formulas or bounds) or faulty data (e.g., errors in map data).

4.2 Coverage Metrics for Sets of Scenarios

Given a set of recorded segments $\mathfrak{S}$ and a classifier $\mathbb{T}$, we want to analyze and quantify *if and to what 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 $\mathbb{T} = \langle \mathcal{Q}, q_r, \mathit{\Gamma}, \lambda_l, \lambda_u \rangle$. Let $\mathit{\Gamma}_q = \{(q, q', \varphi) \in \mathit{\Gamma}\}$ be the set of edges originating in q, and $[\mathit{\Gamma}_q]^{\lambda_l(q)..\lambda_u(q)} =_{def} \bigcup_{i=\lambda_l(q)}^{\lambda_u(q)} [\mathit{\Gamma}_q]^i$ be the set of all subsets of these edges with size within the lower and upper bounds of q. We define the size $|\mathbb{T}| = |q_r|$ recursively as:

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

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

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

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):

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

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:

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

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 $\overline{\mathcal{S}} \in \mathfrak{S}$ 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.

5 Evaluation

We designed our evaluation as a single case mechanism experiment [37] that validates the presented approach. We aim at 1) demonstrating that we can express relevant properties of ODDs in CMFTBL, 2) evaluating to which degree we can achieve scenario class coverage, and 3) evaluating whether coverage analysis

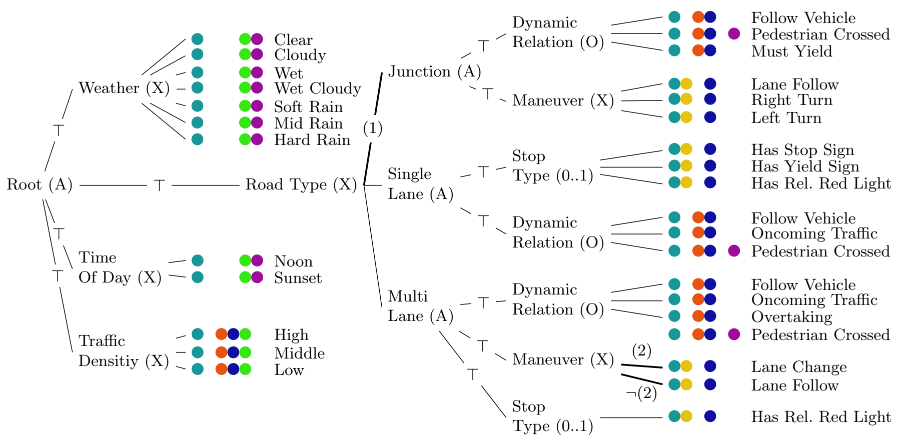
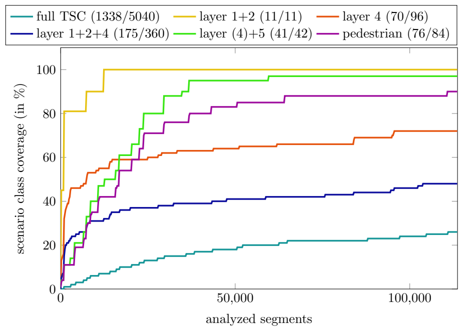


Fig. 4: Tree-Based Scenario Classifiers modeled for our experiments. Features (as well as their paths to the root node) are included in a TSC, if they are marked with a corresponding colored circle. Bold edges with labels (1) and (2) refer to the formulas shown in Sect. 3. Formulas for other features are omitted. Based on the *full TSC* classifier, which contains all evaluated features, we define multiple subsets based on the 6-layer model of scenario classification. In total, we define six classifiers: ● *full TSC*, ● *layer 1+2*, ● *layer 4*, ● *layer 1+2+4*, ● *layer (4)+5*, and ● *pedestrian*.

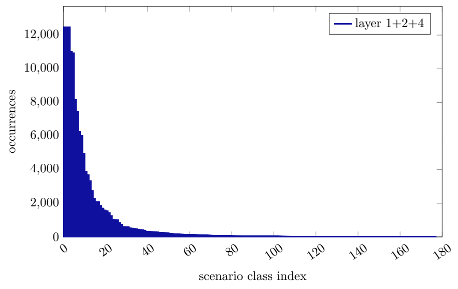
generates valuable insights about missing classes. To this end, we identify and model features for an urban driving environment, define a family of related tree-based scenario classifiers based on those features, and use these classifiers for analyzing simulated urban traffic. We chose features to model the types of properties (or labels) that are envisioned for specifications of operational design domains (ODDs) as described in BSI 1883 [34] or OpenODD [2]. The remainder of this section discusses the features and classifiers developed for our case study, details the experimental setup, presents results from the simulated experiments, and provides initial answers to the above questions.

Classifiers for the 6-Layer Model. To construct tree-based scenario classifiers (TSCs) for our case study, we evaluated the 6-layer model of scenario classification by Scholtes et al. [31] and extracted observable features. In Fig. 4, we visualize all TSCs developed for this case study at once, using colored circles to indicate the features included in each TSC. The hierarchic organization of the complete set of features extracted from the 6-layer model resulted in the *full TSC*, while the others focus on individual layers or combinations of layers. Please note: the *layer 1+2* TSC marked with ● is the one we already presented as an example in Sect. 4.1 (cf. Fig. 2).

We define features as CMFTBL predicates and formulas. In total, we define 51 predicates (including sub-predicates) to completely express the detection of the modeled features for our experiments. We use the min. prevalence operator



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



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

Fig. 5: Coverage of scenario classes and distribution of segments over classes

in 16 predicates to model that some feature holds for most of the time during a segment. For more examples of predicates, please refer to our implementation[4] and the preprint version of this paper [28].

Experimental Setup. For recording and classifying scenario runs, we built a toolchain based on the CARLA simulator [6] and our analysis framework [26].

Based on the classifications of recorded runs, we calculate coverage and the other metrics introduced in Sect. 4.2. Moreover, we analyze saturation by counting class coverage over time.

For our experiments, we record 100 simulation runs of 5 minutes each. In every run, a random map, daytime, and weather is chosen and up to 200 vehicles and 30 pedestrians are spawned randomly on the map. All vehicles are driven by CARLA's autopilot. We analyze each simulation run multiple times: once with each vehicle being considered to be *the ego vehicle*. Overall, this results in 113, 767 segments representing 1, 104 hours of driving data. The analysis of this data with all classifiers and predicates discussed above takes 118 minutes on a single core of a 2021 Apple M1 Pro SoC.

Experimental Results We visualize our results for scenario class coverage over the course of analyzed segments in Fig. 5a. Each colored curve represents the coverage result of one classifier from Fig. 4 and shows its class coverage percentage in relation to its respective observed scene sequences. The legend also shows the final count of observed scenario classes for each classifier after analyzing all 113, 767 segments and the number of possible classes. The *layer 1+2* classifier covers 100% of scenario classes after 12, 233 analyzed segments. Furthermore, *layer (4)+5* almost fully covers its possible scenario classes after around 59, 409 segments but misses one scenario class and therefore only reaches 97% coverage. The *pedestrian* classifier covers over 90% of relevant scenario

[4] `https://github.com/tudo-aqua/stars-carla-experiments/blob/main/src/main/kotlin/tools/aqua/stars/carla/experiments/predicates.kt`

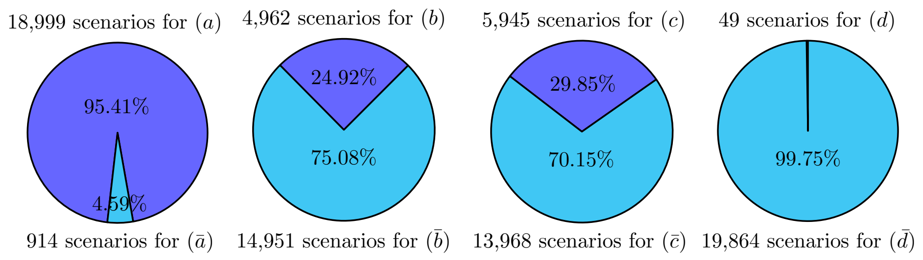


Fig. 6: Visualization of the absolute feature occurrence of "Dynamic Relations" for "Multi-Lane" roads. We define a="Oncoming Traffic", b="Pedestrian Crossed", c="Following Leading Vehicle" and d="Overtaking".

classes. The classifiers *layer 4* and *layer 1+2+4* cover 72% and 48% of relevant scenario classes, respectively. Finally, the reference classifier *full TSC* reaches a coverage of 26%.

Figure 5b visualizes the scenario instance count metric of the 175 observed scenario classes for the *layer 1+2+4* classifier. 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 classes. The three most common scenario classes are each observed about $11,000$ times. The other classifiers show similar long-tail distributions.

Our analysis provides detailed insights into specific scenario classes regarding the underlying features and their combinations. To demonstrate the results, Figs. 6 and 7 visualize analyses on the *Dynamic Relation* features of the *Multi-Lane* node of the TSC (cf. Fig. 4). For better readability in the figures, 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.

Absolute Feature Occurrence. Figure 6 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.

Feature Combinations. As the classifiers (cf. Fig. 4) define the *Dynamic Relation* node as *Optional*, all combinations of the four children form valid scenario classes. Figure 7 shows the distribution for these combinations. It can be seen that 98.14% of observed scenarios are covered by the following five feature combinations: $(a{\cdot}\bar{b}{\cdot}\bar{c}{\cdot}\bar{d})$, $(a{\cdot}\bar{b}{\cdot}c{\cdot}\bar{d})$, $(a{\cdot}b{\cdot}\bar{c}{\cdot}\bar{d})$, $(a{\cdot}b{\cdot}c{\cdot}\bar{d})$, and $(\bar{a}{\cdot}\bar{b}{\cdot}\bar{c}{\cdot}\bar{d})$. The 369 *other*

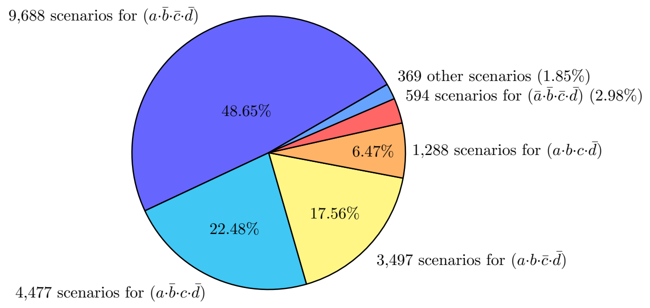


Fig. 7: Distribution of all feature combinations of “Dynamic Relations” for “Multi-Lane” roads. We define a=“Oncoming Traffic”, b=“Pedestrian Crossed”, c=“Following Leading Vehicle” and d=“Overtaking”.

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 five feature combinations that never occurred – $(a{\cdot}b{\cdot}c{\cdot}d)$, $(\bar{a}{\cdot}b{\cdot}c{\cdot}d)$, $(\bar{a}{\cdot}b{\cdot}\bar{c}{\cdot}d)$, $(\bar{a}{\cdot}\bar{b}{\cdot}c{\cdot}d)$, and $(\bar{a}{\cdot}\bar{b}{\cdot}\bar{c}{\cdot}d)$ – each include feature (d), which directly stems from the overall low occurrence of only 0.25% for feature (d).

Missing Feature Combinations. As discussed, our method yields precise information on which scenario classes never occurred. However, a detailed analysis is not feasible, as the full TSC analysis resulted in $3,702$ unseen classes. 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), and (Has Yield Sign & Middle Traffic).

5.1 Discussion

We discuss the experimental results in the context of the three contributions outlined in Sect. 1.

Specification of Scenario Classes: With our hierarchic structuring of properties into scenario classifier trees, we were able to express and organize many relevant features for common driving situations considered in the proposals for operational design domains [34] and approaches like the 6-layer model [31], as well as classify recorded driving data according to the different scenario classes defined by such trees. While our TSC approach (as well as the framework implementation) is generally independent of the logic used to evaluate driving data, CMFTBL proved particularly helpful. The prevalence operator could be used to detect properties where it is natural to formulate “majority of the time” constraints (like environmental conditions or traffic density). Features we did not include in our case study were not left out because it was impossible (or even particularly inconvenient) to express them 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).

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. 4 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 in-vehicle analysis of properties while driving (i.e., after completing a segment) seems possible.

Analyses/Metrics: Our experiments show that we can achieve scenario coverage with hierarchical classifiers. Even though the features evaluated with the classifiers are limited in scope, they already model more than 5.000 situations in urban driving. In our experiments, we achieve high coverage and saturation of observed scenario classes. This is in contrast to Hartjen et al. [12], who analyze saturation effects in observed maneuvers of multiple vehicles including and around the ego vehicle. In their analysis, the amount of observed unique sequences shows no significant saturation over time, indicating that individual maneuvers induce a space that is too big to be feasible for analyzing scenario coverage.

Our detailed analyses proved particularly useful for the interpretation of the coverage levels our classifiers converge to: all five feature combinations that are 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* classifier 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.

5.2 Threats to 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.

Internal Validity. 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.

External Validity. As stated in Sect. 2, we focused on analyzing urban driving scenarios. With our map selection, we could define all relevant predicates for our experiments. However, other typical urban driving situations comprise, e.g., interstate-like roads with on-ramps or roundabouts. In another set of experiments, we successfully adapted the formalizations of interstate traffic done by [19] and analyzed their coverage in simulated driving data. For roundabouts, the main challenge is that the underlying OpenDRIVE data (used by the maps in CARLA) does not explicitly specify a roundabout's lane layout.

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 uncertainties of real sensor data. Analyzing real-world data requires the intelligent handling of such uncertainties. Predicates then need to consider that the environmental perception, such as object tracking, has uncertainty. Current research is already addressing problems in regards to environmental perception [36], such as sensor fusion [8], or object reference generation [22]. We are confident that with further results and insights, we can use our *min. prevalence* operator to deal with uncertainties or sensor errors in the analyzed data.

6 Conclusion

We have presented a temporal logic for expressing features of driving scenarios over complex data streams and combining such features into tree-based scenario classifiers that structure the operational design domain of an autonomous driving system into relevant scenario classes. On recorded driving data, tree-based scenario classifiers enable an analysis of scenario class coverage, metrics, and qualitative analyses that uncover and explain misses. 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. Amersbach, C., Winner, H.: Defining required and feasible test coverage for scenario-based validation of highly automated vehicles. In: ITSC 2019. pp. 425–430. IEEE, New York (2019). https://doi.org/10.1109/itsc.2019.8917534
2. Association for Standardization of Automation and Measuring Systems: ASAM OpenODD: Concept Paper v1.0 (2021), https://www.asam.net/index.php?eID=dumpFile&t=f&f=4544&token=1260ce1c4f0afdbe18261f7137c689b1d9c27576
3. Basin, D., Klaedtke, F., Müller, S., Zălinescu, E.: Monitoring metric first-order temporal properties. J. ACM **62**(2) (2015). https://doi.org/10.1145/2699444
4. Chhaya, B., Jafer, S., Durak, U.: Formal verification of simulation scenarios in aviation scenario definition language (ASDL) **5**(1) (2018). https://doi.org/10.3390/aerospace5010010

5. Dokhanchi, A., Amor, H.B., Deshmukh, J.V., Fainekos, G.: Evaluating perception systems for autonomous vehicles using quality temporal logic. In: RV 2018. pp. 409–416. Springer (2018). https://doi.org/10.1007/978-3-030-03769-7_23
6. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: PMLR 2017. vol. 78, pp. 1–16. PMLR (2017), https://proceedings.mlr.press/v78/dosovitskiy17a.html
7. Esterle, K., Gressenbuch, L., Knoll, A.C.: Formalizing traffic rules for machine interpretability. In: CAVS 2020. pp. 1–7. IEEE (2020). https://doi.org/10.1109/CAVS51000.2020.9334599, CAVS 2020
8. Fadadu, S., Pandey, S., Hegde, D., Shi, Y., Chou, F.C., Djuric, N., Vallespi-Gonzalez, C.: Multi-view fusion of sensor data for improved perception and prediction in autonomous driving. In: WACV 2022. IEEE, New York (2022). https://doi.org/10.1109/wacv51458.2022.00335
9. Felbinger, H., Kluck, F., Li, Y., Nica, M., Tao, J., Wotawa, F., Zimmermann, M.: Comparing two systematic approaches for testing automated driving functions. In: ICCVE 2019. IEEE, New York (2019). https://doi.org/10.1109/iccve45908.2019.8965209
10. Giannakopoulou, D., Pressburger, T., Mavridou, A., Rhein, J., Schumann, J., Shi, N.: Formal requirements elicitation with FRET. In: REFSQ 2020 (2020), https://ceur-ws.org/Vol-2584/PT-paper4.pdf
11. Giannakopoulou, D., Pressburger, T., Mavridou, A., Schumann, J.: Generation of formal requirements from structured natural language. In: REFSQ 2020. pp. 19–35. Springer (2020). https://doi.org/10.1007/978-3-030-44429-7_2
12. Hartjen, L., Philipp, R., Schuldt, F., Friedrich, B.: Saturation effects in recorded maneuver data for the test of automated driving. 13. Uni-DAS e.V. Workshop Fahrerassistenz und automatisiertes Fahren pp. 74–83 (2020), https://www.uni-das.de/images/pdf/fas-workshop/2020/FAS_2020_HARTJEN.pdf
13. ISO Central Secretary: Road vehicles - functional safety - part 1: Vocabulary. Standard ISO 26262-1:2018, International Organization for Standardization, Geneva, CH (2018), https://www.iso.org/standard/68383.html
14. ISO Central Secretary: Road vehicles - safety of the intended functionality. Standard ISO 21448:2022, International Organization for Standardization, Geneva, CH (2022), https://www.iso.org/standard/77490.html
15. Jafer, S., Chhaya, B., Durak, U.: Graphical specification of flight scenarios with aviation scenario defintion language (ASDL). In: AIAA Modeling and Simulation Technologies Conference. American Institute of Aeronautics and Astronautics (2017). https://doi.org/10.2514/6.2017-1311
16. Johannsen, C., Kempa, B., Jones, P.H., Rozier, K.Y., Wongpiromsarn, T.: Impossible made possible: Encoding intractable specifications via implied domain constraints. In: FMICS 2023. pp. 151–169. Springer (2023). https://doi.org/10.1007/978-3-031-43681-9_9
17. Li, C., Cheng, C.H., Sun, T., Chen, Y., Yan, R.: ComOpT: Combination and optimization for testing autonomous driving systems. In: ICRA 2022. IEEE, New York (2022). https://doi.org/10.1109/icra46639.2022.9811794
18. Maierhofer, S., Moosbrugger, P., Althoff, M.: Formalization of intersection traffic rules in temporal logic. In: IV 2022. IEEE, New York (2022). https://doi.org/10.1109/iv51971.2022.9827153, IV 2022
19. Maierhofer, S., Rettinger, A.K., Mayer, E.C., Althoff, M.: Formalization of interstate traffic rules in temporal logic. In: IV 2020. pp. 752–759. IEEE, New York (2020). https://doi.org/10.1109/IV47402.2020.9304549, IV 2020

20. Mariani, R.: An overview of autonomous vehicles safety. In: 2018 IEEE International Reliability Physics Symposium (IRPS). IEEE, New York (2018). https://doi.org/10.1109/irps.2018.8353618, IRPS 2018
21. Mauritz, M., Howar, F., Rausch, A.: Assuring the safety of advanced driver assistance systems through a combination of simulation and runtime monitoring. In: ISoLA 2016, pp. 672–687. Springer (2016). https://doi.org/10.1007/978-3-319-47169-3_52
22. Philipp, R., Zhu, Z., Fuchs, J., Hartjen, L., Schuldt, F., Howar, F.: Automated 3d object reference generation for the evaluation of autonomous vehicle perception. In: ICSRS 2021. IEEE, New York (2021). https://doi.org/10.1109/icsrs53853.2021.9660660, ICSRS 2021
23. Riedmaier, S., Ponn, T., Ludwig, D., Schick, B., Diermeyer, F.: Survey on scenario-based safety assessment of automated vehicles. IEEE Access **8**, 87456–87477 (2020). https://doi.org/10.1109/ACCESS.2020.2993730
24. Rizaldi, A., Keinholz, J., Huber, M., Feldle, J., Immler, F., Althoff, M., Hilgendorf, E., Nipkow, T.: Formalising and monitoring traffic rules for autonomous vehicles in isabelle/HOL. In: iFM 2017, pp. 50–66. Springer (2017). https://doi.org/10.1007/978-3-319-66845-1_4
25. Saberi, A.K., Hegge, J., Fruehling, T., Groote, J.F.: Beyond SOTIF: Black swans and formal methods. In: SysCon 2020. pp. 1–5. IEEE, New York (2020). https://doi.org/10.1109/SysCon47679.2020.9275888, SysCon 2020
26. Schallau, T., Mäckel, D., Naujokat, S., Howar, F.: STARS: A tool for measuring scenario coverage when testing autonomous robotic systems. In: EDCC 2024. Springer (2024). https://doi.org/10.1007/978-3-031-56776-6_6
27. Schallau, T., Naujokat, S.: Validating behavioral requirements, conditions, and rules of autonomous systems with scenario-based testing **82** (2023). https://doi.org/10.14279/tuj.eceasst.82.1222
28. Schallau, T., Naujokat, S., Kullmann, F., Howar, F.: Tree-based scenario classification: A formal framework for coverage analysis on test drives of autonomous vehicles. https://doi.org/10.48550/arXiv.2307.05106, type: article
29. Schallau, T., Naujokat, S., Kullmann, F., Howar, F.: Tree-based scenario classification: A formal framework for coverage analysis on test drives of autonomous vehicles - replication artifact (2023). https://doi.org/10.5281/zenodo.8131947, reproduction artifact
30. Schobbens, P.Y., Heymans, P., Trigaux, J.C.: Feature diagrams: A survey and a formal semantics. In: RE 2006. IEEE, New York (2006). https://doi.org/10.1109/re.2006.23
31. Scholtes, M., Westhofen, L., Turner, L.R., Lotto, K., Schuldes, M., Weber, H., Wagener, N., Neurohr, C., Bollmann, M.H., Kortke, F., Hiller, J., Hoss, M., Bock, J., Eckstein, L.: 6-layer model for a structured description and categorization of urban traffic and environment. IEEE Access **9**, 59131–59147 (2021). https://doi.org/10.1109/access.2021.3072739
32. Schumann, J., Moosbrugger, P., Rozier, K.Y.: R2u2: Monitoring and diagnosis of security threats for unmanned aerial systems. In: RV 2015. pp. 233–249. Springer (2015). https://doi.org/10.1007/978-3-319-23820-3_15
33. Schütt, B., Steimle, M., Kramer, B., Behnecke, D., Sax, E.: A taxonomy for quality in simulation-based development and testing of automated driving systems. IEEE Access **10**, 18631–18644 (2022). https://doi.org/10.1109/ACCESS.2022.3149542
34. The British Standards Institution: Operational Design Domain (ODD) taxonomy for an automated driving system (ADS) – Specification. Specification PAS

1883:2020 (2020), https://www.bsigroup.com/globalassets/localfiles/en-gb/cav/pas1883.pdf
35. Ulbrich, S., Menzel, T., Reschka, A., Schuldt, F., Maurer, M.: Defining and substantiating the terms scene, situation, and scenario for automated driving. In: ITSC 2015. IEEE, New York (2015). https://doi.org/10.1109/itsc.2015.164
36. Velasco-Hernandez, G., Yeong, D.J., Barry, J., Walsh, J.: Autonomous driving architectures, perception and data fusion: A review. In: ICCP 2020. IEEE, New York (2020). https://doi.org/10.1109/iccp51029.2020.9266268, ICCP 2020
37. Wieringa, R.J.: Single-case mechanism experiments. In: Design Science Methodology for Information Systems and Software Engineering, pp. 247–267. Springer (2014). https://doi.org/10.1007/978-3-662-43839-8_18