Synthesizing Traffic Scenarios from Formal Specifications for Testing Automated Vehicles

Moritz Klischat and Matthias Althoff

Abstract— **Virtual testing plays an important role in the validation and verification of automated vehicles. State-of-the-art approaches first generate a huge amount of test scenarios through simulations or test drives, which are later filtered to obtain relevant scenarios for a given set of specifications. However, only few works exist on synthesizing scenarios directly from specifications. In this work, we present an optimization-based approach to synthesize scenarios only from formal specifications and a given map. To concretize the specifications, we formulate predicates, which are subsequently converted to a mixed-integer quadratic optimization problem. We demonstrate how our method can generate scenarios for maps featuring merging lanes and intersections given a variety of specifications.**

I. Introduction

Proving the safety of automated vehicles is still a major challenge due to the variety of situations that can be possibly encountered in the real world. To cope with this variety, virtual testing is essential in the development phase. Especially in industry, virtual testing is oftentimes still based on data recorded from real test drives or variations of it[1]. Not only is this expensive and time-consuming, but also the availability of large datasets to suppliers or public research institutions is limited.

In scenario-based testing, automated vehicles are virtually subjected to short sequences of traffic data, i.e., *scenarios*, which are representative of real-world traffic. For the verification of automated vehicles, test engineers are additionally interested in 1) using scenarios to test the software against selected requirements and 2) a large variety of traffic scenarios where some are only rarely found in datasets recorded from real test drives.

Requirements can include formal specifications of a scenario, i.e., the behavior of surrounding vehicles. These specifications can also be utilized to set a large variety of scenarios in order to create a diverse test suite. In this paper, we present an optimization-based algorithm that synthesizes concrete scenarios that fulfill a given formal specification.

A. Related Work

A commonly-used method to generate test scenarios is the parametrization of scenarios and combinations of all possible values within defined parameter ranges[1]. In particular, for complex scenarios with many parameters, the exploding number of parameter combinations quickly results in an unmanageable computational effort. At the same time, many parameter combinations might be unrealizable.

Approaches focusing on parameterizing and modifying trajectories from real traffic data are presented, e.g., in [1], [2]. The derived variations of a given scenario still resemble the high-level characteristics of the original data. While these approaches consider only one scenario at a time, the stochastic properties of the traffic behavior from an entire database of scenarios can be considered when sampling new scenarios from a Bayesian network whose parameters are learned from that database [3]. This method has been further extended from highways to complex intersections in [4]. With the application of importance sampling, scenarios with interesting, rarely occurring behavior can be generated more efficiently [5], [6].

In verification, software components are typically tested against their functional requirements in scenarios defined by formal specifications. Works on falsification aim at falsifying planning algorithms of automated vehicles against a given logic formula, e.g., formulated in signal temporal logic (STL) [7], [8]. However, these works do not answer the question of how to ensure that the behavior of surrounding vehicles conforms to assumptions the test specification is based on.

While the above approaches for scenario generation can be used to easily generate a large number of scenarios, they cannot explicitly consider specifications for the scenarios. Instead, classification techniques, e.g. [9], [10], would have to be applied subsequently to find scenarios that conform with a desired specification. This process is proposed in [11], [12]. However, these data-driven approaches require large amounts of data to find scenarios that conform with a specification. In particular, for rarely occurring specifications, the required amount of scenarios, and correspondingly the computational effort, increases disproportionately [11]. A more efficient approach is to generate scenarios directly from specifications. The resulting amount of scenarios scales only linearly with respect to the number of specifications.

In [13], specifications of scenarios are first derived from police reports on crashes using natural language processing from which waypoints connected by trajectory planners are generated. While this is demonstrated for scenarios with two vehicles, it remains unclear how the motion of more vehicles could be coordinated reliably. In the context of search-based scenario generation [14], the implicit conformance with a specification can be obtained through dedicated cost functions [15]. With these functions, the search is directed to regions in the space of scenario parameters where the

All authors are with the Technische Universität München, Fakultät für Informatik, Lehrstuhl für Robotik, Künstliche Intelligenz und Echtzeitsysteme, Boltzmannstraße 3, 85748, Garching, Germany. {moritz.klischat, althoff}@in.tum.de

[1]https://waymo.com/safety

specification is fulfilled.

Another representation of specifications is given by ontologies, which can also be generated automatically [16]. Highway scenario descriptions conforming with such ontologies are generated in [17] and combinatorial test generation from ontologies focusing on road infrastructure is proposed in [18].

A formal representation of specifications is provided by temporal logic. In [19], [20], control problems for linear systems subjected to linear temporal logic (LTL) specifications are solved using mixed-integer formulations.

Lately, two scenario description languages and respective scenario generators have been developed, which focus on a realistic visualization of the environment for vision-based algorithms [21], [22]. Another high-level description of traffic scenarios is presented in [9]; so-called traffic patterns describe the traffic flow at intersections, and an algorithm to match scenarios to the patterns is presented.

For the formalization of specifications with a focus on motion planning, a specification language based on constraints is proposed [23]. Valid parameter ranges that satisfy the specifications can be obtained using SAT solvers [24].

In summary, existing works do not provide methods to efficiently generate concrete scenarios from complex specifications in terms of the road network or the coordination of motion of surrounding vehicles.

B. Overview and Contributions

In this work, we present an optimization-based method to synthesize traffic scenarios that conform to a scenario specification. Our specification language is formally defined in Section II: We extend a high-level description similar to [9] by a detailed formal description of spatio-temporal relations of vehicles. The main part of our work is subsequently presented in Section III, where we use for the first time a framework based on mixed-integer quadratic optimization (MIQP) that enables the generation of concrete trajectories satisfying a scenario specification. To consider the specification in the optimization, we formulate them as mixed-integer convex constraints. In Section IV, we demonstrate how our approach can be used to efficiently generate diverse traffic scenarios by applying it to a large set of specifications. The following are the main contributions of our work:

- We present a general scenario specification suited for complex road network topologies.
- Our approach is the first based on MIQP for synthesizing concrete traffic scenarios that explicitly consider formal specifications.
- By including a feasibility check for specifications, our scenarios are guaranteed to be executable.

The advantage of our optimization-based scenario synthesis is that it does not rely on recorded or simulated data and we can generate scenarios based on a road network and an abstract specification only. Furthermore, it opens the possibility of incorporating additional objectives, such as criticality measures, into the cost function to obtain scenarios with desired properties.

II. Scenario Specification

In this work, we use discretized time $t_k = k\Delta t$ with time steps Δt and k as the time index. To simplify the notation, we denote the time-step sequence $(\underline{k}, \underline{k}+1, ..., \overline{k})$ by $[\![\underline{k}, \overline{k}]\!]$. We generate traffic scenarios, which are defined by vehicles V_i following trajectories $x_i(k) \in \mathbb{R}^n, k \in [\![0, h]\!]$, where $i \in \{1, \dots, n_{\text{veh}}\}$ refers to the index of the vehicle, and $h \in \mathbb{N}$ represents the considered time horizon.

Subsequently, we introduce a two-level scenario specification consisting of route patterns and scene sequences. The route patterns are defined in Section II-A and scene sequences in Section II-C.

A. Route Patterns

Before introducing the route patterns, let us define road networks consisting of lanelets [25].

Definition 1 (Lanelet): A lanelet L_{id} with an identifier id is composed of right and left borders defined by polylines and attributes describing its spatial relations to other lanelets: *successors*, *predecessors*, *adjacent_right*, and *adjacent_left*.□

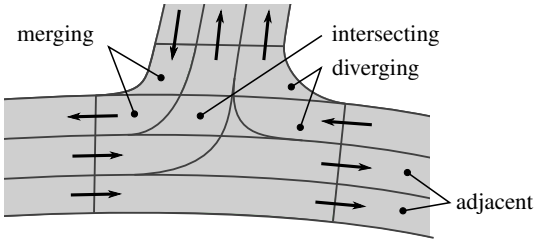


Fig. 1: Example of relations between lanelets.

A lanelet is said to be *adjacent_right* to a lanelet L, when its left lanelet border intersects with L across its entire length. Compared to the original work [25], we additionally use the relations *merging*, *diverging*, and *intersecting*. *Merging (diverging)* lanelets share the same successors (predecessors). *Intersecting* lanelets comprise all pairs of lanelets where the borders intersect and no adjacency or merging relation can be found. Fig. 1 shows an example of these relations.

To further structure the lanelet network, we introduce lanelet sections C_{i_c} combining lanelets that are coupled laterally through the relations *adjacent_left* and *adjacent_right*, as illustrated in Fig. 2. The ID of such a section is denoted by $i_\text{c} \in \mathbb{N}$. For convenience, we denote the IDs of the lanelets by tuples $id = (i_\text{c}, i_l)$, where $i_l \in \mathbb{N}$ enumerates the lanelets of a section in the lateral direction from right to left. Using lanelet sections, we define routes in the lanelet network.

Definition 2 (Route): A route R_κ with route index κ is defined as a tuple of connected lanelet sections. □

For instance, the route of vehicle V_i in Fig. 2 is given by $R_0 = (C_0, C_1)$. The trajectory of each vehicle V_i can be mapped to a route. For this mapping, we introduce the operator $\text{route}(V_i)$, which maps a vehicle to a route R_κ.

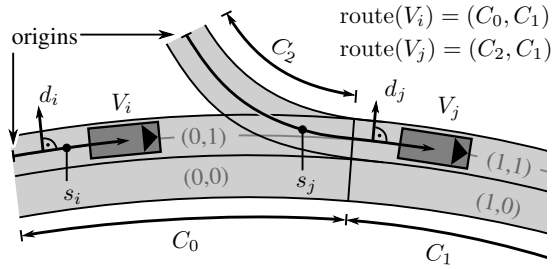
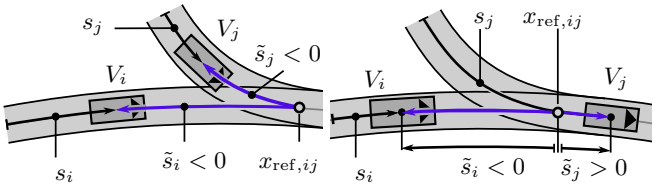


Fig. 2: Example of lanelet sections and lane-based coordinate systems..



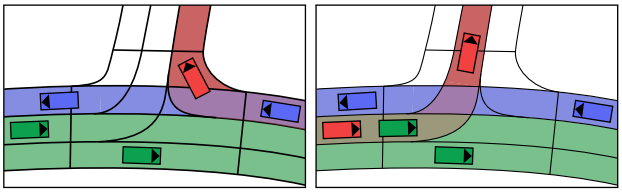
(a) Before merging. (b) After merging.

Fig. 4: Coupling of lane-based coordinate systems of two vehicles V_i, V_j using $x_{\mathrm{ref},ij}$ for computing the distance $\mathrm{dist}(V_i, V_j) = \tilde{s}_j - \tilde{s}_i$.

Definition 3 (Route Pattern): A route pattern of a scenario is the union of the routes of all vehicles in the scenario:

$$T_{\mathrm{route}} = \bigcup_{i=1}^{n_{\mathrm{veh}}} \mathrm{route}(V_i). \qquad \square$$

With these patterns, a scenario can be described on an abstract level. Fig. 3 shows an example of two different route patterns for the same intersection.



(a) (b)

Fig. 3: Examples of two route patterns at an intersection, vehicles are colored based on their assigned routes.

B. Lane-based Coordinate Systems

Henceforth, we consider trajectories in lane-based coordinate systems that are aligned to a reference path. The reference path is represented by a polyline and constructed for each route $R_\kappa \in T_{\mathrm{route}}$ by concatenating the center lines of consecutive lanelets from each section of the route. Since every vehicle is assigned to a route, every vehicle is also assigned to a lane-based coordinate system. The longitudinal state of a vehicle V_i in the coordinate system of $\mathrm{route}(V_i)$ is given by $x_{\mathrm{s},i} = [s_i, \dot{s}_i, \ddot{s}_i]$, where s_i is the longitudinal position, and the lateral state is given by $x_{\mathrm{d},i} = [d_i, \dot{d}_i, \ddot{d}_i]$ with d_i being the lateral position (see Fig. 2). We define a projection operator $\mathrm{lon}_i(x)$ to project a Cartesian coordinate to a longitudinal position of $\mathrm{route}(V_i)$.

For two vehicles V_i and V_j, our specification can involve the longitudinal distance between these vehicles from possibly different coordinate systems. As in our previous work [26], we couple the coordinates s_i and s_j of a pair of vehicles by computing a common reference point $x_{\mathrm{ref},ij}$ at the first intersection of their reference paths. We introduce auxiliary coordinates $\tilde{s}_i = s_i - \mathrm{lon}_i(x_{\mathrm{ref},ij})$ and $\tilde{s}_j = s_j - \mathrm{lon}_j(x_{\mathrm{ref},ij})$ relative to the reference point. Using these coordinates, we define the operator $\mathrm{dist}(V_i, V_j) \coloneqq \tilde{s}_j - \tilde{s}_i$ for computing the longitudinal distance between V_i and V_j. As shown in Fig. 4, this distance is also defined for vehicles before their lanelets merge.

C. Scene Specification

To further specify the behavior of vehicles, we introduce predicates. We focus on essential predicates that can already express a large variety of traffic scenarios. Nevertheless, any other predicate that can be formulated as a mixed-integer convex constraint in the MIQP, as in Section III, can be modeled as well. The predicates are introduced as follows:

Definition 4 (onLanelet): Let a set of lanelets $\mathcal{L}(i_c, [\![\underline{i}_l, \overline{i}_l]\!])$ be defined by the section ID i_c and a sequence of lateral IDs $[\![\underline{i}_l, \overline{i}_l]\!]$. Computing the longitudinal bounds of all lanelets in $\mathcal{L}(i_c, [\![\underline{i}_l, \overline{i}_l]\!])$ in the coordinate system of a vehicle V_i yields the interval $[\underline{s}_{\mathcal{L}(i_c,[\![\underline{i}_l,\overline{i}_l]\!])}, \overline{s}_{\mathcal{L}(i_c,[\![\underline{i}_l,\overline{i}_l]\!])}]$ and the lateral bounds at a longitudinal position s_i within that interval are determined as $[\underline{d}_{\mathcal{L}(i_c,[\![\underline{i}_l,\overline{i}_l]\!])}(s_i), \overline{d}_{\mathcal{L}(i_c,[\![\underline{i}_l,\overline{i}_l]\!])}(s_i)]$. Then the predicate `onLanelet` is defined as

$$\begin{aligned}\texttt{onLanelet}(V_i, i_c, [\![\underline{i}_l, \overline{i}_l]\!]) \iff & \\ \underline{s}_{\mathcal{L}(i_c,[\![\underline{i}_l,\overline{i}_l]\!])} \le s_i \le \overline{s}_{\mathcal{L}(i_c,[\![\underline{i}_l,\overline{i}_l]\!])} & \\ \underline{d}_{\mathcal{L}(i_c,[\![\underline{i}_l,\overline{i}_l]\!])}(s_i) \le d_i \le \overline{d}_{\mathcal{L}(i_c,[\![\underline{i}_l,\overline{i}_l]\!])}(s_i). \square &\end{aligned}$$

Note, that our definition of $\texttt{onLanelet}(V_i, i_c, [\![\underline{i}_l, \overline{i}_l]\!])$ considers the center of the vehicle and not the whole occupancy of the vehicle. By specifying multiple lanelets, we make lane changes possible.

Definition 5 (isBehind): If V_i and V_j are specified to move on the same lanelet or on merging, diverging, or succeeding lanelets, we define that vehicle V_i is behind vehicle V_j with a safety margin $r \in \mathbb{R}^+$ through the predicate

$$\texttt{isBehind}(V_i, V_j) \iff \mathrm{dist}(V_i, V_j) > r. \qquad (1)$$

$\square$

For a vehicle, which is specified to move on a lanelet L_i intersecting with another lanelet L_j, we use additional predicates to specify the crossing behavior. For formulating these predicates, the conflicting area of both vehicles needs to be defined first. To maximize the flexibility for generating scenarios, we aim at computing a small conflicting area. Hence, we use the intersecting area of both lanelets to avoid collisions. When traffic rules should be considered, one could alternatively use stop lines at intersections.

To determine the intersecting area of two lanelets L_i and L_j in lane-based coordinates, we first compute the intersecting points $\xi_z(L_i, L_j) \in \mathbb{R}^2$ with $z \in \{1, \dots, 4\}$ of the lanelet borders through a sweep-line algorithm [27]. After computing the longitudinal coordinates $s_{\xi,z}(L_i, L_j)$ of

each point $\xi_z(L_i, L_j)$, we can determine the longitudinal interval $[\underline{s}^{\text{ca}}_{i,j}, \overline{s}^{\text{ca}}_{i,j}]$ that bounds all $s_{\xi,z}(L_i, L_j)$ as illustrated in Fig. 5. Using this interval, we divide the lanelet L_i into three sections and introduce the corresponding predicates below. These predicates are also illustrated in Fig. 5.

Definition 6 (Conflict Area Predicates): The position of a vehicle V_i on a lanelet intersecting with the lanelet of another vehicle V_j is evaluated by the predicates

$$\begin{aligned}\texttt{beforeCA}(V_i, V_j) &\iff s_i < \underline{s}^{\text{ca}}_{i,j} - r,\\ \texttt{inCA}(V_i, V_j) &\iff \underline{s}^{\text{ca}}_{i,j} - r \leq s_i \leq \overline{s}^{\text{ca}}_{i,j} + r,\\ \texttt{behindCA}(V_i, V_j) &\iff s_i > \overline{s}^{\text{ca}}_{i,j} + r\end{aligned}$$

using a safety margin $r > l_i/2$ with l_i being the length of V_i. □

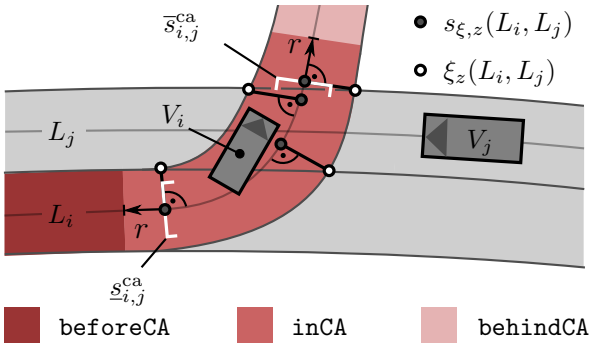


Fig. 5: Construction of longitudinal bounds $[\underline{s}^{\text{ca}}_{i,j}, \overline{s}^{\text{ca}}_{i,j}]$ of the conflict area for vehicle V_i crossing the lanelet of V_j. Furthermore, the intervals corresponding to each intersection predicate of V_i are shown.

To further structure the specification, we divide the set of predicates into scenes as illustrated for an example shown in Fig. 6.

Definition 7 (Scene Sequence): We define a scene S_l by the tuple $(\mathcal{P}_l, \underline{k}_l)$, which encodes the set $\mathcal{P}_l$ of predicates that hold true starting at the switching time $\underline{k}_l \in [\![0, h]\!]$ until the switching time of the subsequent scene S_{l+1}. Thus the complete scenario is specified by the sequence of n_{sc} scenes

$$T_{\text{sc}} = (S_0, \ldots, S_l, \ldots, S_{n_{\text{sc}}}).$$ □

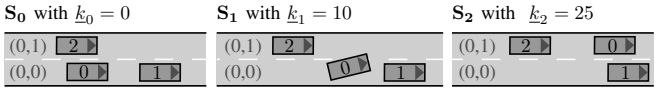


Fig. 6: Example of a lane-change maneuver defined by three scenes.

Instead of specifying the exact switching times, it is more convenient to set lower and upper bounds for the duration δ_l of each scene, i.e., $h_{\min} \leq \delta_l \leq h_{\max}$, $h_{\max} \in \mathbb{N}$, $\forall l \in [\![0, n_{\text{sc}}]\!]$. This also reduces the number of specifications, because scenarios with the same predicates $\mathcal{P}_l$, but different switching times $\underline{x}_l$ can be described by the same specification. The exact duration is determined in the scenario synthesis presented in the next section.

Our definition of a *scene* can be considered as a formalized version of the frequently-used definition in [28]. When creating the specification, collision-free scenarios can be specified by respecting simple rules. For instance, collisions on intersections are avoided when the specification satisfies

$$\begin{aligned}\forall i, j, l\colon i \neq j \wedge \neg(\texttt{inCA}(V_i, V_j) \in \mathcal{P}_l \wedge \texttt{inCA}(V_j, V_i) \in \mathcal{P}_l)\\ \vee \neg(\texttt{inCA}(V_i, V_j) \wedge \texttt{inCA}(V_j, V_i)).\end{aligned} \quad (2)$$

Test cases for automated vehicles can be formulated using our specification by including an ego vehicle in the specification and finally deleting it from the synthesized scenario.

III. Synthesis of Trajectories through Optimization

To synthesize concrete trajectories that comply with a given scenario template, we formulate a mixed logical dynamical (MLD) system and generate a combined MIQP whose solution yields the trajectories for all vehicles. As commonly done in motion planning for road vehicles, we solve the longitudinal and lateral planning problems separately [29].

Similar approaches are proposed for the control of general MLD systems [30], cooperative planning of trajectories [31]–[33] or for trajectory planning of lane changes for single vehicles [34]. Unlike in previous work, our optimization problem combines planning of longitudinal and lateral motion for multiple vehicles while handling intersections.

A. Representation of Switching Times

Initially, the switching times $\underline{k}_l$ of each scene are unknown and need to be determined as part of the optimization problem. To facilitate formulating the optimization problem, we encode each $\underline{k}_l$ through a binary vector $\beta_l \in \{0, 1\}^h$, $\forall l \in [\![0, n_{\text{sc}}]\!]$ that switches at $\underline{k}_l$

$$\beta_l(k) = \begin{cases} 0, & k < \underline{k}_l \\ 1, & k \geq \underline{k}_l \end{cases}.$$

The temporal order of scenes is ensured through the linear constraints

$$\begin{aligned}&\beta_0(0) = 1 \qquad (3)\\ \forall k \in [\![1, h]\!], \forall l \in [\![0, n_{\text{sc}} - 1]\!]\colon\quad &\beta_l(k) \leq \beta_{l+1}(k).\end{aligned}$$

For constraining the duration δ_l of each scene, we need to determine whether a scene is active, i.e., whether $\underline{k}_l \leq k < \underline{k}_{l+1}$. For convenience, we denote this by auxiliary binary variables $\alpha_l \in \{0, 1\}^h$ (see Fig. 7 for an example):

$$\forall k \in [\![0, h]\!]\colon \alpha_l(k) = \begin{cases} \beta_l(k) - \beta_{l+1}(k), & 0 \leq l < n_{\text{sc}} \\ \beta_l(k), & l = n_{\text{sc}} \end{cases}.$$

Hence, the durations are bounded through

$$\forall k \in [\![0, h]\!]\colon \quad h_{\min} \leq \sum_{k=0}^{h} \alpha_l(k) \leq h_{\max}. \qquad (4)$$

	$\underline{k}_0 = 0$		$\underline{k}_1 = 2$			$\underline{k}_2 = 5$	
$\beta_0(k)$	1	1	1	1	1	1	1
$\beta_1(k)$	0	0	1	1	1	1	1
$\beta_2(k)$	0	0	0	0	0	1	1
$\alpha_0(k)$	1	1	0	0	0	0	0
$\alpha_1(k)$	0	0	1	1	1	0	0
$\alpha_2(k)$	0	0	0	0	0	1	1

Fig. 7: Example for binary variables $\beta_l(k)$ and $\alpha_l(k)$ with three scenes.

B. System Dynamics

We now define the linear system dynamics for the longitudinal (denoted by subset s) and lateral motions (denoted by subset d) of all vehicles using the state matrices $A_s, A_d \in \mathbb{R}^{p,p}$ and the input matrices $B_s, B_d \in \mathbb{R}^{p,q}$ by

$$x_\mathrm{s}(k+1) = A_\mathrm{s} x_\mathrm{s}(k) + B_s u_\mathrm{s}(k). \tag{5}$$

The lateral motion is defined by

$$x_\mathrm{d}(k+1) = A_\mathrm{d} x_\mathrm{d}(k) + B_d u_\mathrm{d}(k). \tag{6}$$

The state vector x_s comprises the concatenated states of all vehicles $x_\mathrm{s} = [x_{\mathrm{s},1}^T, \dots, x_{\mathrm{s},n_\mathrm{veh}}^T]^T$ and the input vector is given by the jerk of all vehicles to obtain continuous acceleration: $u_\mathrm{s} = [\dddot{s}_1, \dots, \dddot{s}_{n_\mathrm{veh}}]^T$ and $u_\mathrm{d} = [\dddot{d}_1, \dots, \dddot{d}_{n_\mathrm{veh}}]^T$. In combination with the binary variables $\beta_l(k)$ we obtain an MLD system.

C. Converting Predicates to Mixed-Integer Constraints

To consider the specification T_sc in the MIQP motion planning problem, it needs to be written as mixed-integer linear inequality constraints. The formulation of such constraints from predicates for temporal logic has been shown, e.g., for STL [35], [36].

Since our predicates are intentionally formulated as linear inequality constraints, see Definitions 4 to 6, we make use of the big-M method [37] to only activate them during their corresponding scene, i.e., when $\alpha_l(k) = 1$. In this well-known method for mixed-integer programming, a term involving a vector $M \in \mathbb{R}_+^m$, which has sufficiently large values and is of correct dimensions, is added to a linear inequality constraint to control its activation. In our case, adding the binary term $M(1-\alpha_l(k))$ to a constraint as in (7), ensures that a constraint is always fulfilled if $\alpha_l(k) = 0$.

We convert each predicate of each scene S_l, $l \in \{0, \dots, n_\mathrm{sc}\}$ separately and finally combine all constraints to two inequalities in the required big-M form

$$\begin{aligned} \underline{g}_l(x_\mathrm{s}, k) &> \underline{c}_l(k) - M(1-\alpha_l(k)) \\ \overline{g}_l(x_\mathrm{s}, k) &< \overline{c}_l(k) + M(1-\alpha_l(k)) \end{aligned} \tag{7}$$

for lower and upper bounds $\underline{c}_l(k), \overline{c}_l(k) \in \mathbb{R}^{n_\mathrm{pred}}$ using linear functions $\underline{g}_l(x_\mathrm{s}, k)$, $\overline{g}_l(x_\mathrm{s}, k)$.

To give an example, we can directly write (1) of `isBehind`(V_i, V_j) as a constraint using the big-M method:

$$\underbrace{\tilde{s}_j(k) - \tilde{s}_i(k)}_{\underline{g}_l(x_\mathrm{s},k)} > \underbrace{r}_{\underline{c}_l(k)} - M(1-\alpha_l(k)). \tag{8}$$

D. Longitudinal Optimization Problem

We solve an optimization problem for a user-defined convex cost function $J_\mathrm{s}(x_s, u_s, w)$, which can incorporate terms for desired properties, such as efficiency or criticality. An example is given later for the evaluation in Section IV. By introducing weights $w \in \mathbb{R}^\rho$ for selected terms, we can utilize the cost functions for parameterizing the scenarios in order to obtain variations of the scenario. The complete optimization problem is given by

$$\underset{x_\mathrm{s}(0), u_\mathrm{s}}{\arg\min} \quad \sum_{k=0}^{h} J_\mathrm{s}(x_\mathrm{s}(k), u_\mathrm{s}(k), w) \tag{9}$$

subjected to dynamic constraints, $\forall k \in [\![0, h]\!]$:

$$\begin{aligned} x_\mathrm{s}(k+1) &= A_\mathrm{s} x_\mathrm{s}(k) + B_\mathrm{s} u_\mathrm{s}(k) \\ u_{\mathrm{s},\min} &\le u_\mathrm{s}(k) \le u_{\mathrm{s},\max} \\ x_{\mathrm{s},\min} &\le x_\mathrm{s}(k) \le x_{\mathrm{s},\max}, \end{aligned} \tag{10}$$

predicate constraints, $\forall l \in \{0, \dots, n_{st}\}$, $\forall k \in [\![0, h]\!]$:

$$\begin{aligned} \underline{g}_l(x_\mathrm{s}, k) &< \underline{c}(k) + M(1-\alpha_l(k)) \\ \overline{g}_l(x_\mathrm{s}, k) &> \overline{c}(k) - M(1-\alpha_l(k)), \end{aligned} \tag{11}$$

and logical constraints from (3) and (4):

$$\begin{aligned} \beta_0(0) &= 1 \\ \forall k \in [\![0, h]\!], \forall l \in [\![0, n_\mathrm{sc}-1]\!]&: \beta_l(k) \le \beta_{l+1}(k), \\ \forall k \in [\![0, h]\!]&: h_\mathrm{min} \le \sum_{k=0}^{h} \alpha_l(k) \le h_\mathrm{max}. \end{aligned} \tag{12}$$

E. Lateral Motion

After solving the longitudinal motion problem, we can compute the lateral trajectory of each vehicle. Since the switching times can be obtained from the previously computed $\beta_l(k)$, the active predicates at every time k are known. Hence, only a quadratic program without binary variables needs to be solved. At the longitudinal positions $s(k)$, the lateral lanelet bounds $[\underline{d}_L(s(k)), \overline{d}_L(s(k))]$ specified through `onLanelet` are extracted. Furthermore, the lateral reference path $d_\mathrm{ref}(k) \in \mathbb{R}$ is computed. The trajectory is obtained by solving

$$\underset{u_\mathrm{d}(\cdot)}{\arg\min} \quad \sum_{k=0}^{h} J_\mathrm{d}(x_\mathrm{d}(k), u_\mathrm{d}(k), d_\mathrm{ref}(k), w) \tag{13}$$

subjected to dynamic constraints, $\forall k \in [\![0, h]\!]$:

$$\begin{aligned} x_\mathrm{d}(k+1) &= A_\mathrm{d} x_\mathrm{d}(k) + B_\mathrm{d} u_\mathrm{d}(k) \\ u_{\mathrm{d},\min}(k) &\le u(k) \le u_{\mathrm{d},\max}(k) \\ x_{\mathrm{d},\min}(k) &\le x_d(k) \le x_{\mathrm{d},\max}(k) \end{aligned}$$

and lane constraints, $\forall k \in [\![0, h]\!]$:

$$\underline{d}_L(s(k)) \le x_d(k) \le \overline{d}_L(s(k)).$$

F. Feasibility Checking

Formulating the scenario synthesis as an MIQP problem enables us to check the satisfiability of a specification by the given system dynamics through checking the feasibility of the MIQP problem. For more insights into the cause of a possible infeasibility, the feasibility check can be performed in two steps:

1) The logical checking of the specifications T_{sc} detects a contradiction in the specifications in each scene: We introduce the feasible set $\mathcal{D}_{\mathrm{pred}} \subset \mathbb{R}^{\sum_l n_{\mathrm{pred},l}} \times \{0,1\}^{hn_{\mathrm{sc}}}$, denoting the mixed-integer set fulfilling predicate constraints (11) and binary constraints (12) of the longitudinal problem (9). An empty set $\mathcal{D}_{\mathrm{pred}}$ implies the contradiction of at least two specifications.
2) The satisfiability of the specification by the given system dynamics can be checked by additionally considering the dynamic constraints in (10) for the MIQP feasibility check. This problem can be solved by a branch and bound algorithm and the decidability of this problem has been proven already in [19] for general MLD systems.

IV. Evaluation

In this section, we demonstrate our approach by means of two maps: Map A features two merging lanes on a highway and map B shows an urban T-intersection. To test our approach with a large variety of specifications, we generate them through an exhaustive search, as described in Section IV-A. For the system matrices A_{s} and A_{d}, we use triple integrators for each vehicle and as cost functions we choose $J_{\mathrm{s}}(x_{\mathrm{s}}(k), u_{\mathrm{s}}(k), w) = x_{\mathrm{s}}(k)^T Q_{\mathrm{s}} x_{\mathrm{s}}(k) + u_{\mathrm{s}}(k)^T R_{\mathrm{s}} u_{\mathrm{s}}(k)$ and $J_{\mathrm{d}}(x_{\mathrm{d}}(k), u_{\mathrm{d}}(k), w) = x_{\mathrm{d}}(k)^T Q_{\mathrm{d}} x_{\mathrm{d}}(k) + u_{\mathrm{d}}(k)^T R_{\mathrm{d}} u_{\mathrm{d}}(k)$ to obtain efficient motions. Table I lists the chosen parameters of our algorithm.

Our code is written in Python, and the optimization problems are solved using the solver Gurobi[2]. The computation times are measured on a system with an Intel i7-8650U 1.90 GHz processor. We uploaded all generated scenarios to our website[3] with IDs ZAM_Zip and ZAM_Tjunction. A selection of scenarios can also be found in the video attachment at https://mediatum.ub.tum.de/1537464.

TABLE I: Parameters of our algorithm used for both the maps.

Parameter	Value
Δt [s]	0.25
t_h [s]	*map A:* 8.5, *map B:* 15.0
$t_{h_{\min}}$ [s]	1.5
$a_{s,\min}$ [m/s^2]	-7.0
$a_{s,\max}$ [m/s^2]	3.0
$\lvert a_{\max}\rvert$ [m/s^2]	-7.0
$Q_{\mathrm{s}}, Q_{\mathrm{d}}$	$\mathrm{diag}([5,1,1])$
$R_{\mathrm{s}}, R_{\mathrm{d}}$	0.5

[2]http://www.gurobi.com

[3]https://commonroad.in.tum.de/scenarios

A. Generation of Specifications

To generate a large number of specifications, we use a simple exhaustive search. A more distinct automation of formulating specifications is a subject of future work, and the utilized approach satisfies mainly the purpose of evaluating our synthesis approach. To generate specifications of the scenes, we manually define an initial scene, which is subsequently evolved through an exhaustive tree search, until a defined number of scenes is reached. For map A, we evolve a given scene to the next scene by letting up to two vehicles either change to the adjacent lanelet or to the succeeding lanelet. When switching lanes or merging into one lane, we add nodes in the search tree for each possible order of vehicles in the target lane. For map B, we let one vehicle switch to the following intersection predicate in the order $\big(\texttt{beforeCA}(V_i, V_j), \texttt{inCA}(V_i, V_j), \texttt{behindCA}(V_i, V_j)\big)$. To avoid collisions in the conflict area, we additionally disregard all the specifications that violate the condition in (2). Furthermore, we only consider specifications, wherein at least three vehicles cross the intersection.

B. Map A: Merging Lanes

The highway map features a road with merging lanes as depicted in Fig. 8. According to our definition, this map contains one route and accordingly one lane-based coordinate system. As depicted in Table II, the initial scene S_0 consists of two vehicles driving behind each other on each lane. Using the previously introduced approach for generating four scenes, we obtain 120 specifications in total for which we subsequently synthesize scenarios using our presented approach.

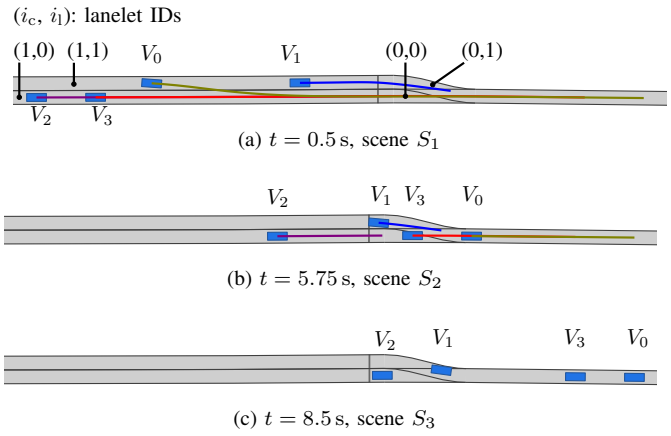
TABLE II: Specification of a scene sequence for Map A.

Scene	Predicates	Vehicle V_0,	Vehicle V_1	Vehicle V_2	Vehicle V_3
S_0	$\texttt{onLanelet}(V_i, i_c, i_l)$	V_0,1,1	V_1,1,1	V_2,1,0	V_3,1,0
	$\texttt{isBehind}(V_i, V_j)$	V_0, V_1	-	V_2, V_3	-
S_1	$\texttt{onLanelet}(V_i, i_c, i_l)$	V_0,1,{0,1}	V_1,1,1	V_2,1,0	V_3,1,0
	$\texttt{isBehind}(V_i, V_j)$	V_0,V_1	-	V_2, V_3	V_3, V_0
S_2	$\texttt{onLanelet}(V_i, i_c, i_l)$	V_0,0,0	V_1,1,1	V_2,1,0	V_3,0,0
	$\texttt{isBehind}(V_i, V_j)$	-	V_1, V_3	V_2, V_3	-
S_3	$\texttt{onLanelet}(V_i, i_c, i_l)$	V_0,0,0	V_1,0,1	V_2,0,0	V_3,0,0
	$\texttt{isBehind}(V_i, V_j)$	V_0, V_1	V_1, V_3	V_2, V_1	V_3, V_0

Out of all specifications, 71 scenarios can be generated, taking on average 0.21 s. For the remaining specifications, the satisfiability checking failed. These specifications mainly contained overtaking maneuvers, which could not be completed in the prescribed time horizon t_h. In Table II we show an exemplary specification and Fig. 8 depicts three frames of the synthesized scenario.

C. Map B: Intersection

We use a T-intersection (Map B) to demonstrate intersection-related predicates. To obtain complex scenarios, we generate specifications for three routes that each intersect with the two other routes, as depicted in Fig. 9. In the



(a) $t = 0.5\,\mathrm{s}$, scene S_1

(b) $t = 5.75\,\mathrm{s}$, scene S_2

(c) $t = 8.5\,\mathrm{s}$, scene S_3

Fig. 8: Synthesized scenario for Map A at different times t.

initial configuration S_0, we place two vehicles on the lanelets approaching the intersection. By generating specifications as described in Section IV-A with six scenes, we obtain in total 557 different specifications T_{sc}. Our algorithm could generate scenarios for all the specifications. In Table III we show an example of a specification and the resulting scenario is depicted in Fig. 10. The computation takes on average 2.29 s.

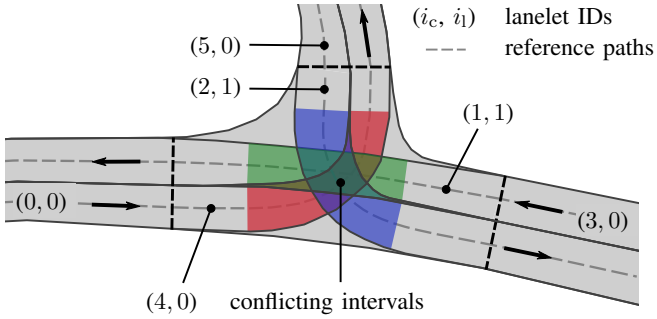


Fig. 9: Map B: reference paths of each route and conflicting intervals used in the experiments. Only lanelet IDs used in the specifications are shown.

V. Conclusions and Future Work

We presented a method for synthesizing traffic scenarios from specifications. Using this approach, scenarios can be generated efficiently instead of searching large databases for scenarios with matching specifications. By tailoring dedicated specifications, one could generate scenarios in which vehicles obey or disregard certain traffic rules. Our method can be further extended to a reactive environment for testing of automated vehicles, where the specification-conforming motion of surrounding vehicles is adapted to the action of the ego vehicle. Future work also includes the automatic generation of specifications and the design of dedicated cost functions for the creation of critical scenarios.

Acknowledgements

The authors gratefully acknowledge financial support by the Central Innovation Programme of the German Federal Government under grant ZF4086007BZ8.

TABLE III: Specification of a scene sequence for Map B.

	Predicates	Vehicle V_i: V_0	V_1	V_2	V_3	V_4	V_5
S_0	onLanelet(V_i, i_c, i_l)	V_0,3,0	V_1,3,0	V_2,0,0	V_3,0,0	V_4,5,0	V_5,5,0
	isBehind(V_i, V_j)	V_0, V_1	-	V_2, V_3	-	V_4, V_5	-
S_1	onLanelet(V_i, i_c, i_l)	V_0,3,0	V_1,3,0	V_2,0,0	V_3,0,0	V_4,5,0	V_5,2,1
	isBehind(V_i, V_j)	V_0, V_1	-	V_2, V_3	-	V_4, V_5	-
	conflict area predicate	-	-	-	-	-	before
S_2	onLanelet(V_i, i_c, i_l)	V_0,1,1	V_1,1,1	V_2,0,0	V_3,4,0	V_4,5,0	V_5,2,1
	isBehind(V_i, V_j)	V_0, V_1	-	V_2, V_3	-	V_4, V_5	-
	conflict area predicate	-	before		before	-	in
S_3	onLanelet(V_i, i_c, i_l)	V_0,1,1	V_1,1,1	V_2,0,0	V_3,4,0	V_4,2,1	V_5,2,1
	isBehind(V_i, V_j)	V_0, V_1	-	V_2, V_3	-	V_4, V_5	-
	conflict area predicate	before	in	-	before	before	behind
S_4	onLanelet(V_i, i_c, i_l)	V_0,1,1	V_1,1,1	V_2,0,0	V_3,4,0	V_4,2,1	V_5,2,1
	isBehind(V_i, V_j)	V_0, V_1	-	V_2, V_3	-	V_4, V_5	-
	conflict area predicate	before	in	-	in	before	behind
S_5	onLanelet(V_i, i_c, i_l)	V_0,1,1	V_1,1,1	V_2,4,0	V_3,4,0	V_4,2,1	V_5,2,1
	isBehind(V_i, V_j)	V_0, V_1		V_2, V_3	-	V_4, V_5	-
	conflict area predicate	before	behind	before	behind	before	behind

References

[1] M. R. Zofka, F. Kuhnt, R. Kohlhaas, C. Rist, T. Schamm, and J. M. Zollner, "Data-driven simulation and parametrization of traffic scenarios for the development of advanced driver assistance systems," in *18th Int. Conf. on Information Fusion*, 2015, pp. 1422–1428.

[2] S. Wagner, K. Groh, T. Kuhbeck, and A. Knoll, "Towards cross-verification and use of simulation in the assessment of automated driving," in *Proc. of the IEEE Intelligent Vehicles Symposium*. IEEE, 2019, pp. 1589–1596.

[3] T. A. Wheeler, M. J. Kochenderfer, and P. Robbel, "Initial Scene Configurations for Highway Traffic Propagation," in *Proc. of the IEEE Conference on Intelligent Transportation Systems*, 2015, pp. 279–284.

[4] S. Jesenski, J. E. Stellet, F. Schiegg, and J. M. Zollner, "Generation of scenes in intersections for the validation of highly automated driving functions," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2019, pp. 502–509.

[5] D. Zhao, H. Lam, H. Peng, S. Bao, D. J. LeBlanc, K. Nobukawa, and C. S. Pan, "Accelerated Evaluation of Automated Vehicles Safety in Lane-Change Scenarios Based on Importance Sampling Techniques," *IEEE Trans. on Intelligent Transportation Systems*, vol. 18, no. 3, pp. 595–607, 2017.

[6] M. O'Kelly, A. Sinha, H. Namkoong, J. Duchi, and R. Tedrake, "Scalable end-to-end autonomous vehicle testing via rare-event simulation," in *Proc. of the 32nd Int. Conf. on Neural Information Processing Systems*, 2018, pp. 9849–9860.

[7] C. E. Tuncali, G. Fainekos, H. Ito, and J. Kapinski, "Simulation-based adversarial test generation for autonomous vehicles with machine learning components," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 1555–1562.

[8] C. Gladisch, T. Heinz, C. Heinzemann, J. Oehlerking, A. V. Vietinghoff, and T. Pfitzer, "Experience paper : search-based Testing in automated driving control Applications," in *Proc. of the 34th IEEE/ACM Int. Conf. on Automated Software Engineering*, 2019, pp. 26–37.

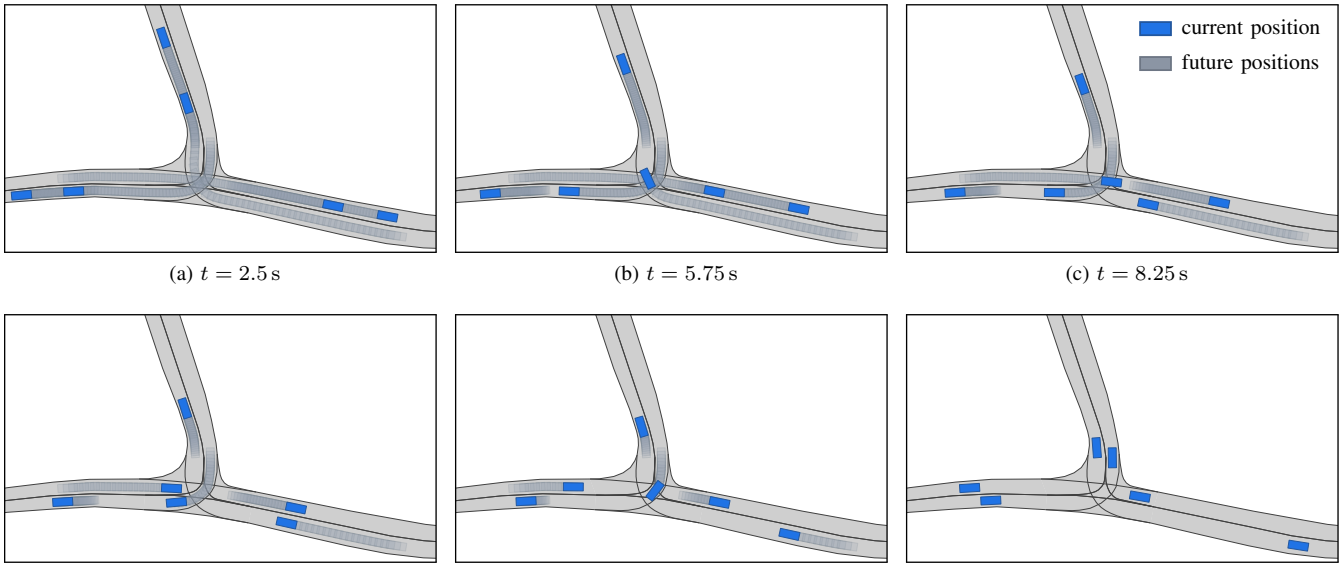
[9] H. Zhang, A. Geiger, and R. Urtasun, "Understanding high-level semantics by modeling traffic patterns," in *Proc. of the IEEE Int. Conf. on Computer Vision*, 2013, pp. 3056–3063.

[10] F. Kruber, J. Wurst, and M. Botsch, "An Unsupervised Random Forest Clustering Technique for Automatic Traffic Scenario Categorization," in *Proc. of the IEEE Conf. on Intelligent Transportation Systems*, 2018, pp. 2811–2818.

[11] J. Bach, J. Langner, S. Otten, E. Sax, and M. Holzapfel, "Test scenario selection for system-level verification and validation of geolocation-dependent automotive control systems," in *23rd Int. Conf. on Engineering, Technology and Innovation*, 2018, pp. 203–210.

[12] C. Sippl, F. Bock, D. Wittmann, H. Altinger, and R. German, "From simulation data to test cases for fully automated driving and ADAS," in *Lecture Notes in Computer Science*, 2016, pp. 191–206.

[13] A. Gambi, T. Huynh, and G. Fraser, "Generating effective test cases for self-driving cars from police reports," in *Proc. of the 27th ACM Joint*



(a) $t = 2.5\,\mathrm{s}$ (b) $t = 5.75\,\mathrm{s}$ (c) $t = 8.25\,\mathrm{s}$

(d) $t = 10.0\,\mathrm{s}$ (e) $t = 12.25\,\mathrm{s}$ (f) $t = 14.75\,\mathrm{s}$

Fig. 10: Synthesized scenario for Map B at different points in time t.

Meeting on European Software Engineering Conf. and Symposium on the Foundations of Software Engineering, 2019, pp. 257–267.

[14] S. Ali, L. C. Briand, H. Hemmati, and R. K. Panesar-Walawege, "A systematic review of the application and empirical investigation of search-based test case generation," *IEEE Transactions on Software Engineering*, vol. 36, no. 6, pp. 742–762, 2010.

[15] F. Hauer, A. Pretschner, and B. Holzmüller, "Fitness functions for testing automated and autonomous driving systems," in *SAFECOMP 2019: Computer Safety, Reliability, and Security*, pp. 69–84.

[16] G. Bagschik, T. Menzel, and M. Maurer, "Ontology based Scene Creation for the Development of Automated Vehicles," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 1813–1820.

[17] 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," *Proc. of the IEEE Intelligent Vehicles Symposium*, pp. 2383–2390, 2019.

[18] Y. Li, J. Tao, and F. Wotawa, "Ontology-based test generation for automated and autonomous driving functions," *Information and Software Technology*, vol. 117, Art. no. 106200, 2020.

[19] S. Karaman, R. G. Sanfelice, and E. Frazzoli, "Optimal control of mixed logical dynamical systems with Linear Temporal Logic specifications," in *Proc. of the IEEE Conf. on Decision and Control*, 2008, pp. 2117–2122.

[20] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based trajectory generation with linear temporal logic specifications," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2014, pp. 5319–5325.

[21] D. J. Fremont, X. Yue, T. Dreossi, A. L. Sangiovanni-Vincentelli, S. Ghosh, and S. A. Seshia, "Scenic: A language for scenario specification and scene generation," in *Proc. of the ACM SIGPLAN Conf. on Programming Language Design and Implementation*, 2019, pp. 63–78.

[22] R. Majumdar, A. Mathur, M. Pirron, L. Stegner, and D. Zufferey, "PARACOSM: A language and tool for testing autonomous driving systems," 2019, arXiv:1902.01084.

[23] A. Eggers, M. Stasch, T. Teige, T. Bienmüller, and U. Brockmeyer, "Constraint Systems from Traffic Scenarios for the Validation of Autonomous Driving," in *Third International Workshop on Satisfiability Checking and Symbolic Computation, Part of FLOC 2018*, pp. 95–109.

[24] K. Scheibler, A. Eggers, T. Teige, M. Walz, T. Bienmüller, and U. Brockmeyer, "Solving Constraint Systems from Traffic Scenarios for the Validation of Autonomous Driving," in *Proc. of the 4th SC-square Workshop*, 2019, pp. 2–12.

[25] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: efficient map representation for autonomous driving," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2014, pp. 420–425.

[26] M. Klischat and M. Althoff, "Generating Critical Test Scenarios for Automated Vehicles with Evolutionary Algorithms," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2019, pp. 2352–2358.

[27] M. I. Shamos and D. Hoey, "Geometric intersection problems," in *Proc. of the 17th Annual Symposium on Foundations of Computer Science*, 1976, pp. 208–215.

[28] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer, "Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving," in *Proc. of the IEEE Conf. on Intelligent Transportation Systems*, 2015, pp. 982–988.

[29] C. Katrakazas, M. Quddus, W. H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 416–442, 2015.

[30] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[31] J. Peng and S. Akella, "Coordinating Multiple Robots with Kinodynamic Constraints Along Specified Paths," *The International Journal of Robotics Research*, vol. 24, no. 4, pp. 295–310, 2005.

[32] F. Altché, X. Qian, and A. De La Fortelle, "Time-optimal coordination of mobile robots along specified paths," in *IEEE Int. Conf. on Intelligent Robots and Systems*, 2016, pp. 5020–5026.

[33] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *European Control Conference*, 2001, pp. 2603–2608.

[34] C. Miller, C. Pek, and M. Althoff, "Efficient Mixed-Integer Programming for Longitudinal and Lateral Motion Planning of Autonomous Vehicles," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 1954–1961.

[35] V. Raman, M. Maasoumy, A. Donze, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *Proc. of the IEEE Conf. on Decision and Control*, 2014, pp. 81–87.

[36] S. Sadraddini and C. Belta, "Robust temporal logic model predictive control," in *Proc. of the 53rd Annual Allerton Conference on Communication, Control, and Computing*, 2016, pp. 772–779.

[37] I. Griva, S. G. Nash, and A. Sofer, *Linear and Nonlinear Optimization*. SIAM, 2009.