International Journal on Software Tools for Technology Transfer
https://doi.org/10.1007/s10009-025-00827-9

GENERAL

Special Issue: FMICS 2024

Correct-by-construction instantiation of abstract scenarios

Jan Steffen Becker[1] · **Christian Neurohr**[1]

Accepted: 7 October 2025
© The Author(s) 2025

Abstract

In the automotive domain, scenario-based development is the answer to the increasing complexity of highly automated driving functions. Scenario-based methods cluster the large scenario space by so-called abstract scenarios which can be used to sample an infinite number of concrete scenarios. Because abstract scenarios constrain driving maneuvers by excessive use of additional constraints, constraint solving techniques may be required to find concrete scenario instances. In order to guarantee correctness of simulation runs with respect to the abstract scenario, realistic vehicle dynamics need to be considered during the instantiation process. This work proposes an encoding scheme for abstract scenarios into linear constraint systems for the generation of correct-by-construction concrete scenarios. It covers both temporal and spatial aspects of the abstract scenario, as well as simple, but realistic vehicle dynamics. The applicability of the method is shown by demonstrating it on a set of basic driving maneuvers. This paper extends the previous version by presenting further experimental results and giving an overview over practical applications.

Keywords Scenario-based testing · Simulation · Vehicle dynamics · Constraint solving · Abstract scenario

1 Introduction

In the scenario-driven development of automated driving systems (ADSs) [47], scenarios are used at different levels of abstraction [33, 36]. This paper looks at two abstraction levels: *abstract* and *concrete* scenarios. While a concrete scenario [33] describes one specific event sequence in a fixed scenery, abstract scenarios [36] focus on the relations between traffic participants and the environment. Thereby, an abstract scenario may cover a variety of scenes and event sequences – it subsumes an infinite number of concrete scenarios. In a scenario-based testing process [9, 35], for example, abstract scenarios are used to cluster the test space into adequate scenario classes. Scenario class mining may be data-driven or based on expert knowledge, and supported by criticality analysis [36]. Test execution, and experimental evaluation (e.g., in the context of criticality analysis), however, requires sampling of concrete scenarios from the scenario classes.

Formal languages for abstract scenarios, such as OpenSCENARIO DSL [44], SCENIC [21], or Traffic Sequence Charts (TSCs) [13], allow the user to specify complex spatial and temporal relations between traffic participants. These various constraints may interact in a way that turns instantiation of abstract scenarios into a tough challenge. As a simple example, speed and distance constraints later on in a scenario can restrict valid placements of traffic participants at the beginning of the scenario.

This work is an extended version of the paper "Safe Linear Encoding of Vehicle Dynamics for the Instantiation of Abstract Scenarios" [4] presented at the FMICS 2024 conference. As a quite generic solution, it proposes to use constraint solving to generate concrete scenarios that satisfy all the constraints. Given an abstract scenario, the constraint system must guarantee that

This paper is supported by European Union's Horizon 2020 research and innovation programme under grant agreement No. 101146542 (SYNERGIES). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Climate, Infrastructure and Environment Executive Agency (CINEA). Neither the European Union nor CINEA can be held responsible for them.

✉ J.S. Becker
jan.becker@dlr.de

C. Neurohr
christian.neurohr@dlr.de

[1] German Aerospace Center (DLR) e.V., Institute of Systems Engineering for Future Mobility, Oldenburg, Germany

Published online: 17 October 2025

- the concrete scenario is executable on a simulator with realistic vehicle dynamics, and
- the simulation conforms to the abstract scenario, i.e., all the constraints in the abstract scenario are fulfilled.

This requires that knowledge about vehicle dynamics is incorporated into the constraint solving problem. In this work an approach is presented, how both abstract scenarios and vehicle dynamics can be encoded into a linear constraint solving problem. The focus is on the encoding of vehicle dynamics.

This paper extends the original work [4] by providing – in addition to the encoding of the vehicle dynamics – extended constructions for encoding abstract scenarios into linear constraint solving problems. Furthermore, it adds new experimental results (Sect. 6) and gives an overview over potential applications (Sect. 7).

The remainder of the paper is structured as follows: Sect. 2 summarizes related work, and Sect. 3 gives an overview on the scenario instantiation approach. In Sect. 4, the complete construction for encoding vehicle dynamics is presented, followed by an investigation into its mathematical properties in Sect. 5. An experimental evaluation is described in Sect. 6. Potential applications of the proposed approach are put forward in Sect. 7. The paper concludes with a short discussion in Sect. 8.

2 Related work

A range of description languages for traffic scenarios has been proposed so far [32]. Most of them follow the concepts defined by Ulbrich et al. [52], who define a scenario as a sequence of scenes, where the temporal evolution between the scenes is described by events and actions. Bach et al. [2] present a model-based scenario description language in a graphical modeling environment. Zhang et al. [57] decide for a textual language for scripting driving scenarios on two different abstraction levels. Menzel et al. [34] propose to derive so-called logical scenarios from natural language scenario descriptions in an engineering process. Logical scenarios are equipped with parameter ranges and probability distributions that allow to sample test cases that follow well-known characteristics. Schuldt et al. [50] propose testing strategies based on logical scenarios. ASAM OpenSCENARIO XML has evolved to a widely accepted industry standard for logical and concrete driving scenarios and has been adopted by academic and industrial driving simulators such as ESMINI, Vires VTD, and CARLA. These description languages all use a similar set of driving maneuvers as scenario building blocks. This allows an efficient execution by simulation software, but only a limited level of abstraction.

Neurohr et al. [36] therefore propose the use of formalized abstract scenario descriptions. The authors present concepts for exploiting abstract scenarios for criticality analysis [36], hazard analysis and risk assessment (HARA) [31], and testing [35] of automated driving functions.

Textual domain-specific languages for abstract scenarios are M-DSL [18], OpenSCENARIO DSL [44], and Scenic [21]. These languages combine maneuvers and textual constraints. Because they still rely on maneuver actions, their interpretation is simulator-dependent. While Scenic comes with an execution engine that implements a guided simulation approach, the support of OpenSCENARIO DSL is still limited. Since version 0.9.15, the CARLA ScenarioRunner[1] implements a subset of OpenSCENARIO DSL.

Zone graphs [11] and TSCs [13, 14] are graphical specification languages for abstract scenarios. TSCs have mathematically defined semantics [15], and have been proposed for a wide range of applications, including simulation [5] and monitoring [23]. In contrast to the aforementioned languages, TSCs rely exclusively on seamless declarative scene descriptions. Therefore, simulation requires constraint solving [3, 5]. The approach for scenario instantiation presented in this paper has already been used in the authors' previous works [5, 9], but without giving details about the constructions. Similar approaches are followed by, e.g., Eggers et al. [17], Klischat and Althoff [28], and Goyal et al. [22].

There are different approaches for solving constraint systems arising from traffic scenarios. A purely discrete approach is presented by Schwammberger [51]. Above cited work [3, 5, 17, 28] sketches some solutions that consider vehicle dynamics to some extent. General-purpose numeric solvers for hybrid systems such as PHAVer [19] and SpaceEx [20] generate precise solutions, but scale bad for the large state space induced by traffic scenarios. Plaku et al. [40, 41] overcome this limitation by combining simulation techniques with temporal model checking.

Bézier curves are widely used for path planning (e.g., in [12, 24, 43]). The cited works use them in combination with a single track vehicle dynamics model (sometimes also called bicycle model). Kong et al. [29] show experimentally that a more complex dynamic model (that considers tire forces) has no advantages over the simple model with respect to control design.

3 Formal description of abstract scenarios

According to Ulbrich et al., "a scenario describes the temporal development between several scenes in a sequence of scenes" [52]. This is realized differently in the different scenario description languages. In this paper, a simplified form of TSCs [13] is used as an example specification language.

[1] https://github.com/carla-simulator/scenario_runner/tree/v0.9.15 accessed on 2024-04-19.

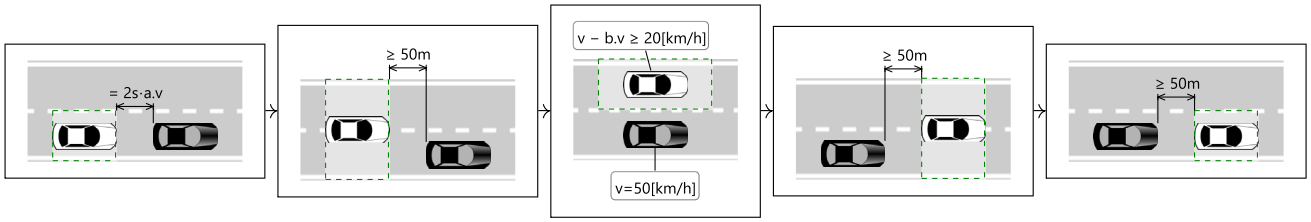


Fig. 1 An overtaking maneuver as TSC

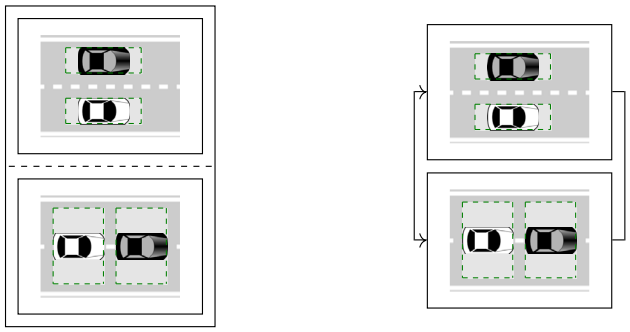


Fig. 2 Parallel composition (left) and choice (right) of two invariant nodes

A TSC describes an abstract scenario as an acyclic graph of so-called *spatial views*. A spatial view is a graphical representation of a scene with a well-defined mathematical semantics. Figure 1 shows a TSC describing an overtake maneuver. The placement of symbols (roads, crossings, vehicles) in a spatial view describes spatial relations between the depicted objects in both an x and y direction. TSCs combine spatial views to so-called *basic charts*. The simplest form of a basic chart is an *invariant node* that contains a single spatial view. More complex basic charts can be formed by combining them in sequence (the second chart starts when the first chart ends), parallel (both charts start and end at the same time), or forming a choice (one of the two charts is executed). The TSCs considered in this paper each consist of a single basic chart. The TSC in Fig. 1 is a sequence five invariant nodes. The syntax of parallel composition and choice is shown in Fig. 2

In this paper, only a subset of the TSC language is used. The interested reader is referred to [15] for the complete syntax and semantics.

3.1 Syntax and semantics of spatial views

The TSC language itself does not define a domain ontology. Instead, the attributes and dynamics of the depicted objects are defined by the so-called *world model* of the TSC. A so-called *symbol dictionary* then links the used graphical representations (called *object symbols*) to object classes. Decoupling the syntactical elements from the domain ontology in this way allows to easily cover new concepts (e.g., e-scooters) in the system's operational design domain without a need to extend the formalism itself. In this work, we assume that the world model defines vehicles, obstacles, straight roads, and orthogonal crossings.

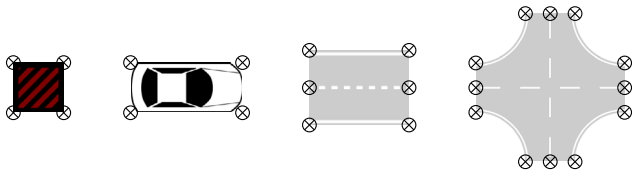


Fig. 3 Anchor points (marked by ⊗) defined by the symbol dictionary

Besides declaring the used graphical representations, the symbol dictionary lays out the foundations for interpretation of spatial relations by declaring *anchor points* on the object symbols. In this work, we assume that each object symbol has anchor points that stand for the corners of the object's axis-aligned bounding box. Roads and crossings have additional anchor points at the lane borders, as depicted in Fig. 3. As a consequence, spatial relations are always defined with respect to vehicle bounding boxes (in a Cartesian coordinate system) and lane boundaries. A spatial view translates to a set of spatial and non-spatial constraints consisting of

1. a spatial relation between each pair of anchor points, based on their relative position in the spatial view (called implicit relations)
2. distance constraints between anchor points, graphically represented by *distance lines* in the spatial view
3. non-spatial constraints denoted by textual annotations in the spatial view

Note, that the anchor points are not shown in the spatial view. The dashed boxes are so-called somewhere boxes and are used to finer control the implicit spatial relations from item 1—the objects outside the box are put in relation to the box borders, instead of the object inside the box. Each somewhere box, and the spatial view itself, spans a so-called *frame*.

Definition 1 (Semantics of spatial views)
Assume we can express the position of each anchor point $o.a$ of each object o depicted in a spatial view as a pair of

expressions $(o.a_x, o.a_y)$ over the attributes of o defined in the world model. Each somewhere box (with index k) has four anchor points $(\underline{x}_k, \underline{y}_k)$, $(\overline{x}_k, \underline{y}_k)$, $(\underline{x}_k, \overline{y}_k)$, and $(\overline{x}_k, \overline{y}_k)$. Furthermore, each distance line is labeled with a predicate $\mathtt{D}_d$. A spatial view induces a set Φ of constraints consisting of the following:

- For each frame f, and each pair $o.a, o'.a'$ of anchor points in f, including anchor points of the spanning box, two constraints
  - one of $o.a_x < o'.a'_x$, $o.a_x > o'.a'_x$, or $o.a_x = o'.a'_x$ depending on $o.a$ being left of, right of, or on the same x-position as $o'.a'$, and
  - one of $o.a_y < o'.a'_y$, $o.a_y > o'.a'_y$, or $o.a_y = o'.a'_y$ depending on $o.a$ being below of, above of, or on the same y-position as $o'.a'$.
- A constraint $\mathtt{D}_d(|o.a_x - o'.a'_x|)$ for each horizontal distance line d between anchor points $o.a$ and $o'.a'$.
- Likewise, a constraint $\mathtt{D}_d(|o.a_y - o'.a'_y|)$ for each vertical distance line.
- The textually annotated non-spatial relations

A spatial view then translates to the constraint

$$\varphi_{sv} :\equiv \exists \underline{x}_1, \overline{x}_1 \underline{y}_1, \overline{y}_1, \ldots, \underline{x}_n, \overline{x}_n \underline{y}_n, \overline{y}_n \in \mathbb{R} : \bigwedge \Phi$$

where $\underline{x}_1, \overline{x}_1 \underline{y}_1, \overline{y}_1, \ldots, \underline{x}_n, \overline{x}_n \underline{y}_n, \overline{y}_n$ ranges over all somewhere box anchor points.

Example 1 (Semantics of first spatial view in Fig. 1)
The semantics of the first spatial view in Fig. 1 is

$$\begin{aligned}
&\exists \underline{\mathtt{x}}_1, \bar{\mathtt{x}}_1, \underline{\mathtt{y}}_1, \bar{\mathtt{y}}_1 \in \mathbb{R}.\ r.\underline{\mathtt{x}} < \underline{\mathtt{x}}_1 < \bar{\mathtt{x}}_1 < b.\underline{\mathtt{x}} < b.\bar{\mathtt{x}} < r.\bar{\mathtt{x}} \\
&\quad \wedge r.\underline{\mathtt{y}} = \underline{\mathtt{y}}_1 < \bar{\mathtt{y}}_1 < r.c < r.\bar{\mathtt{y}} \wedge r.\underline{\mathtt{y}} < b.\underline{\mathtt{y}} < b.\bar{\mathtt{y}} < r.c \\
&\quad \wedge \underline{\mathtt{x}}_1 = a.\underline{\mathtt{x}} < a.\bar{\mathtt{x}} = \bar{\mathtt{x}}_1 \wedge \underline{\mathtt{y}}_1 < a.\underline{\mathtt{y}} < a.\bar{\mathtt{y}} < \bar{\mathtt{y}}_1 \\
&\quad \wedge b.\underline{\mathtt{x}} - a.\bar{\mathtt{x}} = 2\,\mathrm{s} \cdot a.v\,,
\end{aligned}$$

where $\underline{\mathtt{x}}$, $\bar{\mathtt{x}}$, $\underline{\mathtt{y}}$, and $\bar{\mathtt{y}}$ denote bounding boxes (see Fig. 4) of the white vehicle a, the black vehicle b, and the road r. The variable $r.c$ denotes the y-coordinate of the road center line. Eliminating the quantifiers leads to

$$\begin{aligned}
&r.\underline{\mathtt{y}} < b.\underline{\mathtt{y}} < b.\bar{\mathtt{y}} < r.c \wedge r.\underline{\mathtt{y}} < a.\underline{\mathtt{y}} < a.\bar{\mathtt{y}} < r.c < r.\bar{\mathtt{y}} \\
&\quad \wedge r.\underline{\mathtt{x}} < a.\underline{\mathtt{x}} < a.\bar{\mathtt{x}} < b.\underline{\mathtt{x}} < b.\bar{\mathtt{x}} < r.\bar{\mathtt{x}} \\
&\quad \wedge b.\underline{\mathtt{x}} - a.\bar{\mathtt{x}} = 2\,\mathrm{s} \cdot a.v\ .
\end{aligned}$$

3.2 Semantics of the temporal structure

We define the semantics of a TSC in terms of *scenario instances*.

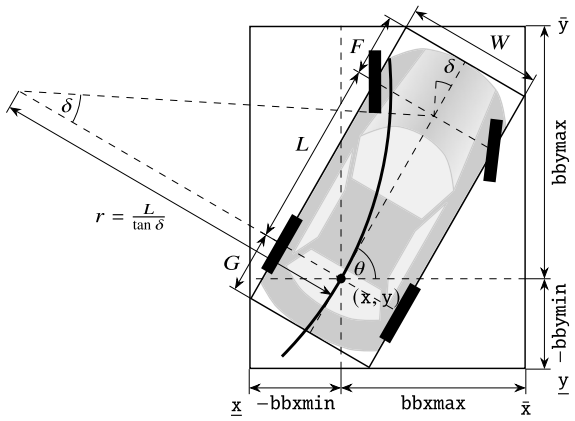


Fig. 4 The simple single track model

Definition 2 (Scenario instances)
A scenario instance is a function $\sigma : \mathbb{R}_{\geq 0} \to \mathcal{S}$ assigns a scene instance $\sigma(t) \in \mathcal{S}$ to every point in time. Each scene instance $\sigma(t) \in \mathcal{S}$ assigns a value to each attribute of each object in a scenario.

Basic charts are evaluated scenario instances σ and left-closed, right-open intervals $[b, e) \subseteq \mathbb{R}_{\geq 0}$. We define satisfaction of basic charts, written $\sigma, [b, e) \models \mathsf{BC}$ inductively as follows.

Definition 3 (Semantics of TSCs)
Given a formula φ over object attributes (i.e., the semantics of a spatial view), we write $\sigma(t) \models \varphi$ if φ evaluates under $\sigma(t)$ to *true*.

- An invariant node with spatial view sv is satisfied on σ and $[b, e)$ if $\sigma(t) \models \varphi_{sv}$ for all $t \in [b, e)$, where φ_{sv} is the semantics of sv according to Definition 1.
- A sequence of basic charts A and B is satisfied on σ and $[b, e)$ if there exists some time point m with $b < m < e$ such that $\sigma, [b, m) \models A$ and $\sigma, [m, e) \models B$.
- The parallel composition of basic charts A and B is satisfied on σ and $[b, e)$ if both $\sigma, [b, e) \models A$ and $\sigma, [b, e) \models B$.
- The choice between basic charts A and B is satisfied on σ and $[b, e)$ if $\sigma, [b, e) \models A$ or $\sigma, [b, e) \models B$ (or both).

A scenario instance σ is instance of a TSC consisting of basic chart BC if there exists a time point $e > 0$ such that $\sigma, [0, e) \models \mathsf{BC}$.

In this work, a simple single track vehicle dynamics model [49] is used. The single track model (Fig. 4) assumes that the longitudinal axis of the vehicle is tangential to the vehicle's trajectory, and touches the trajectory at the center of the (stable) rear axis, which is used as a reference point for the vehicle position $\mathbf{p}(t) = (x(t), y(t))^T$ at time t. The direction of the longitudinal axis is called the *heading angle* $\theta(t)$ and

is at the same time the direction of movement. In steady state cornering, the vehicle follows a circle with radius $r = \frac{L}{\tan\delta}$ depending on wheel base L and steering angle δ. In detail, we assume that in all scenario instances all vehicles adhere to the following equations of movement.

Definition 4 (Vehicle equations of movement)
The single track model's equations of movement are

$$\dot{x}(t) = v(t)\cos\theta(t)$$
$$\dot{y}(t) = v(t)\sin\theta(t)$$
$$\dot{\theta}(t) = v(t)\frac{\tan\delta(t)}{L}$$

and are subject to the constraints

$$|\delta(t)| \leq \delta_{\max}$$
$$a_{lat}(t) := |\dot{\theta}(t)v(t)| \leq a_{lat,\max} := 0.4\mathrm{g} \quad .$$

The bounding box extrema $\underline{x}(t) = x + \texttt{bbxmin}(\theta(t))$, $\overline{x} + \texttt{bbxmax}(\theta(t))$, $\underline{y} = y + \texttt{bbymin}(\theta(t))$, and $\overline{y} = y + \texttt{bbymax}(\theta(t))$ at time t are calculated from the reference points and some offsets. Note, that the offsets `bbxmin`, `bbxmax`, `bbymin`, and `bbymax` only depend on rotation and vehicle size. Since the latter is fixed, we consider the offsets being functions of the heading angle $\theta(t)$ only.

In the spatial views, we denote the heading angle by heading. Heading angles with a difference of a multiple of 2π describe in fact the same heading direction. Therefore, we only require that the heading attribute equals the heading angle $\theta(t)$ modulo a period of 2π, i.e., for all time points t,

$$\exists k \in \mathbb{Z}.\, \mathsf{heading}(t) = \theta(t) + 2k\pi \quad .$$

4 Encoding vehicle dynamics and scenes

The encoding process for an abstract scenario into a constraint solving problem (CSP) follows the idea of bounded model checking and is displayed in Fig. 5. First, we split the time domain of the scenario into N intervals $T_i = [t_{i-1}, t_i)$ (for $i = 1, 2, \ldots, N$) of fixed length $t_i - t_{i-1} = \Delta$, called slices. Note, that a scene may hold during more than one slice, so the number of slices can be larger than the number of scenes. Usually, one starts with the minimal number of slices, and adds further slices until a solution is found or an abortion criterion (e.g., maximum number of slices or timeout) reached. On each time slice $i = 1, 2, \ldots, N$, we describe each spatial view (indexed by j) in the scenario with a constraint ϕ^i_j. This constraint is further described later on in Sect. 4.3. Furthermore, we introduce constraints that describe the possible movement of vehicles according to the single track model. The temporal structure of the abstract scenario is encoded with boolean state variables.

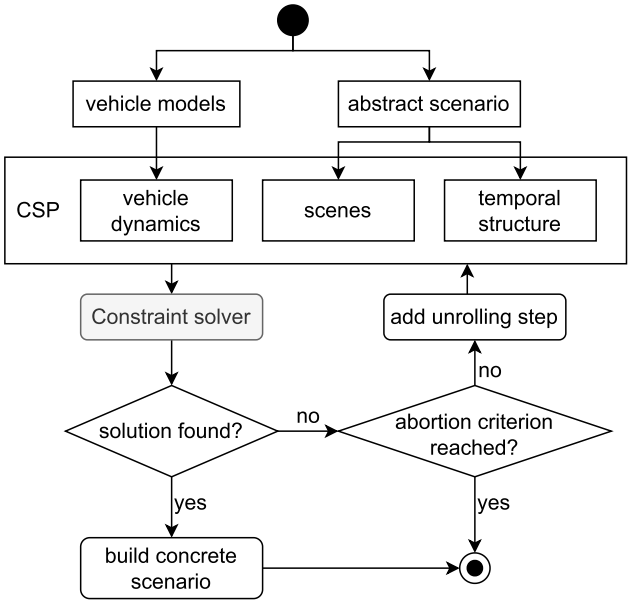


Fig. 5 Strategy for scenario instantiation

4.1 Encoding of the temporal structure

For TSCs that consist of a single sequence of invariant nodes, we can encode the temporal structure using a single Boolean variable s^i_j per invariant node (with index j) and time slice (with index i).

Construction 1 (Encoding of scene sequences)
A sequence of M scenes with indices $j = 1, 2, \ldots, M$ is encoded for N slices by introducing a Boolean variable s^i_j for each scene $j = 1, 2, \ldots, 5$ and slice index $i = 1, 2, \ldots, N$ together with a constraint

$$\overbrace{\bigwedge_{i=1}^{N-1}\left((\mathsf{s}^{i+1}_1 \rightarrow \mathsf{s}^i_1) \wedge \bigwedge_{j=1}^{M-1}(\mathsf{s}^{i+1}_{j+1} \rightarrow (\mathsf{s}^i_{j+1} \vee \mathsf{s}^i_j))\right)}^{(a)}$$
$$\wedge \overbrace{\left(\mathsf{s}^1_1 \wedge \bigwedge_{j=2}^{M}\neg\mathsf{s}^1_j\right)}^{(b)} \wedge \overbrace{\mathsf{s}^N_M}^{(c)} \quad .$$

Part (a) of the formula says that scene $j+1$ may be 'active' on slice $i+1$ only if itself or its predecessor was active on slice i. The sub-term $(\mathsf{s}^{i+1}_1 \rightarrow \mathsf{s}^i_1)$ handles this rule for the first scene which has no predecessor. Furthermore, only the first scene is active on the first slice (part (b)), and the last scene in the end (part (c)). The link between the spatial and temporal aspects of the scenario is done by adding a series of implications $\bigwedge_{j,i}(\mathsf{s}^i_j \rightarrow \phi^i_j)$ ranging over all slices and scenes.

Algorithm 1 Translation of temporal structure

function $\mathrm{BC2SAT}(\mathsf{BC}, (\mathsf{b}^i)_i)$
  if BC is an invariant node **then**
    introduce a family $(\mathsf{s}^i_{\mathsf{BC}})_i$ of variables
    $\Omega \leftarrow \bigwedge_{i=0}^{N-1}((\mathsf{s}^{i+1}_{\mathsf{BC}} \wedge \mathsf{b}^i) \rightarrow \mathsf{s}^i_{\mathsf{BC}})$
    $\phi^0 \leftarrow \mathit{false}$
    $\phi^i \leftarrow \mathsf{s}^{i-1}_{\mathsf{BC}} \wedge \mathsf{b}^{i-1}$ for $i = 1, \ldots, N-1$
    return $\Omega, (\phi^i)_i$
  else if BC is a sequence of A and B **then**
    declare a fresh family $(\mathsf{m}^i_{\mathsf{BC}})_i$ of variables
    $\Omega_A, (\phi^i_A)_i \leftarrow \mathrm{BC2SAT}(A, (\mathsf{b}^i)_i)$
    $\Omega_B, (\phi^i_B)_i \leftarrow \mathrm{BC2SAT}(B, (\mathsf{m}^i_{\mathsf{BC}})_i)$
    $\Omega \leftarrow (\mathsf{m}^0_{\mathsf{BC}} \rightarrow \phi^0_A) \wedge \bigwedge_{i=0}^{N-1}(\mathsf{m}^{i+1}_{\mathsf{BC}} \rightarrow (\mathsf{m}^i_{\mathsf{BC}} \vee \phi^{i+1}_A))$
    $\Omega' \leftarrow \bigwedge_{i=0}^{N-1}(\mathsf{m}^i_{\mathsf{BC}} \rightarrow \mathsf{m}^{i+1}_{\mathsf{BC}})$
    return $\Omega_A \wedge \Omega_B \wedge \Omega \wedge \Omega', (\phi^i_B)_i$
  else if BC is a choice of A and B **then**
    $\Omega_A, (\phi^i_A)_i \leftarrow \mathrm{BC2SAT}(A, (\mathsf{b}^i)_i)$
    $\Omega_B, (\phi^i_B)_i \leftarrow \mathrm{BC2SAT}(B, (\mathsf{b}^i)_i)$
    return $\Omega_A \wedge \Omega_B, (\phi^i_A \vee \phi^i_B)_i$
  else if BC is a parallel composition of A and B **then**
    $\Omega_A, (\phi^i_A)_i \leftarrow \mathrm{BC2SAT}(A, (\mathsf{b}^i)_i)$
    $\Omega_B, (\phi^i_B)_i \leftarrow \mathrm{BC2SAT}(B, (\mathsf{b}^i)_i)$
    return $\Omega_A \wedge \Omega_B, (\phi^i_A \wedge \phi^i_B)_i$

For TSCs that contain parallel compositions and/or choices, an invariant node may have multiple predecessors or successors. Here, the translation to a CSP must ensure that all branches of a parallel composition or choice are entered and left at the same time. Therefore, the function $\mathrm{BC2SAT}(\mathsf{BC}, (\mathsf{b}^i))$ presented in Algorithm 1 adds some modifications to the previous construction. As an input, it takes the basic chart BC to be encoded in the CSP, and a family $(\mathsf{b}^i)_{i=0,\ldots,N}$ of $N+1$ Boolean variables (or the vector $(\mathit{true})_{i=0,\ldots,N}$ containing the literal *true* in each place). The variables are used to indicate whether a sub-chart can be entered in a particular step—a subchart can be entered in step i only if the variable with index i evaluates to *true*, and the variables with index $j < i$ evaluate to false (this ensures that the subchart is entered only once). These variables are introduced at the switching points of sequences, and passed along the syntax tree of the TSC to the sub-charts of the second part of a sequence. The function returns an invariant Ω and a family $(\phi^i)_{i=0,\ldots,N}$ of constraints that describe if the chart has been completed in a particular step (the constraint ϕ^i is for step i). The function is called recursively on all subcharts of a TSC. Hereby, the invariant Ω collects the possible transitions between invariant nodes. The constraint families $(\phi^i)_i$ serve as intermediate results – for the toplevel chart, only ϕ^N (completeness in the last slice) is used. A complete TSC is then encoded as

$$\Omega \wedge \phi^N \quad \text{with} \quad \Omega, (\phi^i)_i := \mathrm{BC2SAT}(\mathsf{BC}, (\mathit{true})_{i=0,\ldots,N})$$

where BC is the top-level basic chart of the TSC and $(\mathit{true})_{i=0,\ldots,N}$ fixes the start of the TSC to the first time slice.

4.2 Encoding trajectories as splines

We model vehicle trajectories as Bézier splines of fixed degree $n \geq 2$. Bézier curves [42] of degree n are defined by interpolation between $n+1$ points $\mathbf{p}_0, \ldots \mathbf{p}_n$, where $\mathbf{p}_0$ and $\mathbf{p}_n$ are endpoints of the curve, and $\mathbf{p}_1, \ldots, \mathbf{p}_{n-1}$ are bendpoints.

Definition 5 (Bézier curve)
A Bézier curve of degree n with control points $\mathbf{p}_0, \ldots, \mathbf{p}_n \in \mathbb{R}^2$ is a parametric curve $\mathbf{p} : [0,1] \rightarrow \mathbb{R}^2$ given by

$$\mathbf{p}(s) := \sum_{k=0}^{n} \mathcal{B}^n_k(s)\mathbf{p}_k$$

$$\text{with} \quad \mathcal{B}^n_k(s) := \binom{n}{k}(1-s)^k s^{n-k}$$

called the Bernstein basis polynomial. A Bézier spline is a connected curve that is piecewise defined by Bézier curves.

Bézier curves have the following properties [42] that are of interest for this work:

(P1) A Bézier curve is always located within the polygon (convex hull) spanned by its control points.
(P2) Bézier curves are closed under linear combinations and affine transformations; Bézier curves of same degree can be combined/transformed by applying the combination/transformation (pairwise) to the control points.
(P3) The first derivative is again Bézier curve of degree $n-1$:

$$\dot{\mathbf{p}}(s) = \sum_{k=0}^{n-1} \mathcal{B}^{n-1}_k(s) n(\mathbf{p}_{k+1} - \mathbf{p}_k)$$

(P3) By applying *degree elevation*, we can express every Bézier curve of degree $n-1$ with control points $\mathbf{p}_0^{(n-1)}, \ldots, \mathbf{p}_{n-1}^{(n-1)}$ as a Bézier curve of degree n with control points $\mathbf{p}_0^{(n)} = \mathbf{p}_0^{(n-1)}$, $\mathbf{p}_n^{(n)} = \mathbf{p}_{n-1}^{(n-1)}$, and $\mathbf{p}_k^{(n)} = \frac{k}{n}\mathbf{p}_{k-1}^{(n-1)} + \frac{n-k}{n}\mathbf{p}_k^{(n-1)}$ for $k = 1, \ldots, n-1$.

We describe the trajectories of vehicles by Bézier splines, using one spline segment per time slice. The spline uniquely defines both the path and the speed profile of a vehicle. A trajectory $\mathbf{p} : [t_0, t_N] \rightarrow \mathbb{R}^2$ of N slices is described by a sequence of $N \in \mathbb{N}$ Bézier curves $\mathbf{p}^1, \ldots, \mathbf{p}^N$ such that

$$\mathbf{p}(t) = \mathbf{p}^i(s) \text{ for } t \in [t_{i-1}, t_i) \text{ and } s = \frac{t - t_{i-1}}{\Delta} \quad . \qquad (1)$$

Note that velocity and acceleration at time point $t \in [t_{i-1}, t_i)$ scale by Δ^{-1} (resp. Δ^{-2}) and are given by

$$\mathbf{v}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p}^i((t - t_{i-1})\Delta^{-1}) = \Delta^{-1}\dot{\mathbf{p}}^i(s) \tag{2}$$

$$\mathbf{a}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{v}(t) = \Delta_i^{-2}\ddot{\mathbf{p}}^i(s) \ . \tag{3}$$

4.3 Encoding of spatial views

In the following, we consider only a single time slice and vehicle. So, we can drop the superscripts and refer to the Bézier curve for the current slice as $\mathbf{p}$. It is described by $n+1$ control points $\mathbf{p}_0, \ldots, \mathbf{p}_n$. We encode the kth control point $\mathbf{p}_k$ by a pair of variables x_k and y_k. Other vehicle attributes that are used in the spatial views are bounding box dimensions $bound \in BB := \{\texttt{bbxmin}, \texttt{bbxmax}, \texttt{bbymin}, \texttt{bbymax}\}$, speed v, longitudinal acceleration acc, and heading angle θ. Expressing them exactly in terms of the x_k and y_k would result in a non-linear CSP. Instead, we approximate them by safe lower and upper bounds $\underline{\mathsf{bound}}$, $\overline{\mathsf{bound}}$ (for $bound \in BB$), $\underline{v}(s), \overline{v}(s), \underline{acc}(s), \overline{acc}(s)$, $\underline{\mathsf{head}}$, and $\overline{\mathsf{head}}$ (lower bounds are underlined and upper bounds overlined). In case of bounding box dimensions and heading angle, these are free variables in the CSP; in case of v and acc these are Bernstein polynomials of degree $n-1$ resp. $n-2$ over free variables $\underline{\mathsf{v}}_0, \ldots, \underline{\mathsf{v}}_{n-1}$ (resp. $\overline{\mathsf{v}}_0, \ldots, \overline{\mathsf{v}}_{n-1}$, $\underline{\mathsf{acc}}_0, \ldots, \underline{\mathsf{acc}}_{n-2}$, and $\overline{\mathsf{acc}}_0, \ldots, \overline{\mathsf{acc}}_{n-2}$). For clarity, this paper uses a sans-serif font for free variables in the CSP, and a serif font for expressions and polynomials over these variables.

Definition 6 (Free variables in the CSP)
Given a fixed integer $n \geq 2$, the encoding of spatial views (Construction 2) and vehicle dynamics (Construction 4) uses the following variables for each object o in the scenario:

- If o is a vehicle:
  - $o.\underline{\mathsf{bound}}$, $o.\overline{\mathsf{bound}}$ for each $bound \in BB$
  - $o.\underline{\mathsf{head}}$, $o.\overline{\mathsf{head}}$
  - $o.\underline{\mathsf{v}}_k, o.\overline{\mathsf{v}}_k$ for each $k = 0, 1, \ldots, n-1$ (describing polynomials $\underline{v}(s) := \sum_{k=0}^{n-1} \underline{\mathsf{v}}_k \mathcal{B}_k^{n-1}(s)$ and $\overline{v}(s) := \sum_{k=0}^{n-1} \overline{\mathsf{v}}_k \mathcal{B}_k^{n-1}(s)$)
  - $o.\underline{\mathsf{acc}}_k, o.\overline{\mathsf{acc}}_k$ for each $k = 0, 1, \ldots, n-2$ (describing polynomials $\underline{acc}(s) := \sum_{k=0}^{n-2} \underline{\mathsf{acc}}_k \mathcal{B}_k^{n-2}(s)$, and $\overline{acc}(s) := \sum_{k=0}^{n-2} \overline{\mathsf{acc}}_k \mathcal{B}_k^{n-2}(s)$)
  - a Boolean variable rws indicating reverse movement.

  These variables are introduced for every time slice.
- If o is a static object, introduce a free variable for each attribute of o (the same variable is used on all time slices).

The variable rws is needed later on in Construction 4. Section 4.4 describes the constraints that are added to the CSP to ensure that the values assigned to these variables are consistent with the vehicle trajectory.

Now, recall the overall approach presented in this paper. The variables introduced above are part of a CSP that encodes the possible scenario instances of a TSC. From each solution of the CSP, we can extract a scenario instance σ by interpolating between the Bézier control points that describe the vehicle trajectories. The control points are represented as free variables in the CSP, so the solution contains their positions directly. The exact construction for doing this will be shown later in this paper. When building the CSP, we cut the time domain into slices and represent each invariant node Inv (containing a spatial view sv) on each slice (with index i) by a free Boolean variable $\mathsf{s}^i_{\mathsf{Inv}}$. We have to add constraints to our CSP which ensure that $\mathsf{s}^i_{\mathsf{Inv}}$ can be assigned *true* only if the spatial view holds on the whole slice. When applying quantifier elimination, a spatial view can be expressed as a quantifier-free linear formula φ over the object attributes, which are encoded by the real-valued free variables in the CSP. Given some scene as a monotone Boolean combination of linear constraints, we can utilize properties (P1) and (P2) of Bézier splines and conservatively approximate each linear constraint using the Bézier control points and lower/upper bounds introduced above.

Construction 2 (Safe approximations of scenes)
Assume a formula φ that describes a scene as a monotone Boolean combination of linear inequalities of the form $\sum_i a_i A_i \lhd b$, where $a_i, b \in \mathbb{R}$ are constants, the A_i are object attributes $o_i.attr$ defined in the world model, and $\lhd \in \{\leq, <\}$. It is translated recursively to a linear constraint $tr(\varphi)$ over the free variables given in Definition 6 as follows:

- *For linear inequalities, it is*

$$tr\left(\sum_i a_i A_i \lhd b\right) :\equiv \bigwedge_{k=0}^{n} \sum_i a_i \tilde{A}_{i,k} \lhd_k b$$

where $\lhd_0 = \lhd$, $\lhd_k = \leq$ for $k > 0$, and the A_i are replaced by linear terms $\tilde{A}_{i,k}$ as follows:
  - *If A_i is an attribute of some vehicle o_i, and $a_i \geq 0$:*

$$\tilde{A}_{i,k} = \begin{cases} o_i.\mathsf{x}_k + o_i.\overline{\texttt{bbxmin}} & \text{if } A_i = o_i.\underline{x} \\ o_i.\mathsf{x}_k + o_i.\overline{\texttt{bbxmax}} & \text{if } A_i = o_i.\overline{x} \\ o_i.\mathsf{y}_k + o_i.\overline{\texttt{bbymin}} & \text{if } A_i = o_i.\underline{y} \\ o_i.\mathsf{y}_k + o_i.\overline{\texttt{bbymax}} & \text{if } A_i = o_i.\overline{y} \\ o_i.\overline{\mathsf{head}} & \text{if } A_i = o_i.\mathsf{heading} \\ o_i.\overline{v}_k^{(n)} & \text{if } A_i = o_i.v \\ o_i.\overline{acc}_k^{(n)} & \text{if } A_i = o_i.acc \end{cases}$$

where $o_i.v_k^{(n)}$ and $o_i.acc_k^{(n)}$ are derived from $o_i.\overline{\mathsf{v}}_0, \ldots, o_i.\overline{\mathsf{v}}_{n-1}$, resp. $o_i.\overline{\mathsf{acc}}_0, \ldots, o_i.\overline{\mathsf{acc}}_{n-2}$, by degree elevation, i.e., (we drop the prefix o_i)

$$\overline{v}_0^{(n)} := \overline{\mathsf{v}}_0$$

$$\overline{v}_n^{(n)} := \overline{\mathsf{v}}_{n-1}$$

$$\overline{v}_k^{(n)} := \frac{k}{n}\overline{\mathsf{v}}_{k-1} + \frac{n-k}{n}\overline{\mathsf{v}}_k \text{ for } k = 1,\dots,n-1 \ .$$

For the acceleration, degree elevation is applied twice, i.e., $\overline{acc}_k^{(n)} := \overline{\mathsf{acc}}_0$ *if* $n = 2$*, and, if* $n > 2$*,*

$$\overline{acc}_0^{(n)} := \overline{\mathsf{acc}}_0$$

$$\overline{acc}_n^{(n)} := \overline{\mathsf{acc}}_{n-2}$$

$$\overline{acc}_1^{(n)} := \frac{2}{n}\overline{\mathsf{acc}}_0 + \frac{n-2}{n}\overline{\mathsf{acc}}_1$$

$$\overline{acc}_{n-1}^{(n)} := \frac{n-2}{n}\overline{\mathsf{acc}}_{n-3} + \frac{2}{n}\overline{\mathsf{acc}}_{n-2}$$

$$\overline{acc}_k^{(n)} := \frac{(k-1)k}{(n-1)n}\overline{\mathsf{acc}}_{k-2} + \frac{2(n-k)k}{(n-1)n}\overline{\mathsf{acc}}_{k-1} + \frac{(n-k)(n-k-1)}{(n-1)n}\overline{\mathsf{acc}}_k \text{ for } k = 2,\dots,n-2.$$

  - *If* A_i *is an attribute of some vehicle* o_i*, and* $a_i < 0$*, then* $\tilde{A}_{i,k}$ *is defined the same as above, except that lower bounds are used instead of upper bounds (e.g.,* $o_i.\underline{\mathtt{bbxmin}}$ *instead of* $o_i.\overline{\mathtt{bbxmin}}$*).*
  - *For attributes of static objects,* $\tilde{A}_{i,k}$ *is the variable introduced for the attribute.*
- *Conjunctions and disjunctions are propagated through, i.e.,* $tr(\phi_1 \wedge \phi_2) \equiv tr(\phi_1) \wedge tr(\phi_2)$ *and* $tr(\phi_1 \vee \phi_2) \equiv tr(\phi_1) \vee tr(\phi_2)$*.*

Example 2 (Encoding of 2-seconds distance)
A strict 2-seconds distance

$$b.\underline{x} - a.\overline{x} > 2\,\mathrm{s} \cdot (a.v - b.v)$$

between two vehicles a and b is encoded with quadratic Bézier splines as a conjunction of the following constraints:

- $(b.\mathsf{x}_0 + b.\underline{\mathtt{bbxmin}}) - (a.\mathsf{x}_0 + a.\overline{\mathtt{bbxmax}}) > 2\,\mathrm{s} \cdot (a.\overline{\mathsf{v}}_0 - b.\underline{\mathsf{v}}_0)$
- $(b.\mathsf{x}_1 + b.\underline{\mathtt{bbxmin}}) - (a.\mathsf{x}_1 + a.\overline{\mathtt{bbxmax}}) \geq 2\,\mathrm{s} \cdot (\frac{a.\overline{\mathsf{v}}_0 + a.\overline{\mathsf{v}}_1}{2} - \frac{b.\underline{\mathsf{v}}_0 + b.\underline{\mathsf{v}}_0}{2})$
- $(b.\mathsf{x}_2 + b.\underline{\mathtt{bbxmin}}) - (a.\mathsf{x}_2 + a.\overline{\mathtt{bbxmax}}) \geq 2\,\mathrm{s} \cdot (a.\overline{\mathsf{v}}_1 - b.\underline{\mathsf{v}}_1)$

4.4 Encoding of vehicle dynamics

The CSP constructed from the abstract scenario must ensure that

- the trajectory is possible within the single track model, and
- on each time slice, the variables, resp. polynomials, $\underline{\mathsf{bound}}$, $\overline{\mathsf{bound}}$ (for $bound \in BB$), $\underline{\mathsf{head}}$, $\overline{\mathsf{head}}$, $\underline{v}(s) := \sum_{k=0}^{n-1} \underline{\mathsf{v}}_k \mathcal{B}_k^{n-1}(s)$, $\overline{v}(s) := \sum_{k=0}^{n-1} \overline{\mathsf{v}}_k \mathcal{B}_k^{n-1}(s)$, $\underline{acc}(s) := \sum_{k=0}^{n-2} \underline{\mathsf{acc}}_k \mathcal{B}_k^{n-2}(s)$, and $\overline{acc}(s) := \sum_{k=0}^{n-2} \overline{\mathsf{acc}}_k \mathcal{B}_k^{n-2}(s)$ are valid lower/upper bounds for the actual bounding box extrema, heading angle, speed, and acceleration.

This is done by constructing some constraint system denoted Φ in the following, that is instantiated for each vehicle and time slice. It includes a special linear approximation of the single track model equations of motion.

The construction for Φ uses a discretization of the heading angle θ. We choose a finite set $\mathcal{I}$ of heading intervals. Each interval $I = [\hat{\theta} - \eta, \hat{\theta} + \eta] \in \mathcal{I}$ is characterized by a *primary heading angle* $\hat{\theta} \in [-\pi, \pi]$ and a *heading angle deviation* $\eta \in [0, \pi/2)$. The selection of heading intervals depends on the abstract scenario. For example, a U-turn scenario requires that the union of heading intervals contains at least a half circle, while in an overtaking scenario a small heading range may be sufficient. The to-be-constructed constraint Φ ensures that on each slice the vehicle heading is within one of the intervals. From this interval, linear constraints for the vehicle movement within the slice are derived. In order to simplify the construction, we project the vehicle trajectory into a rotated coordinate system where the x-axis points into the primary heading direction.

Construction 3 (Normalized rotation)
Given some interval $I = [\hat{\theta} - \eta, \hat{\theta} + \eta]$ *and variables* $\mathsf{x}_k, \mathsf{y}_k$ *(for* $k = 0,\dots,n$*) describing Bézier curve control points, define*

$$X(I)_k := \mathsf{x}_k \cos\hat{\theta} + \mathsf{y}_k \sin\hat{\theta}$$

$$Y(I)_k := -\mathsf{x}_k \sin\hat{\theta} + \mathsf{y}_k \cos\hat{\theta} \ .$$

Analogously, denote the control points of the projected first and second derivatives (see property (P3)) by $(\dot{X}(I)_k, \dot{Y}(I)_k)^T$ *resp.* $(\ddot{X}(I)_k, \ddot{Y}(I)_k)^T$*.*

In the rotated coordinate system, the vehicle moves roughly in X-direction, with a deviation of $\pm\eta$. Because the speed on a curve segment is determined by the distance between control points, the curvature increases when the vehicle accelerates (or brakes) on a curved segment (visually, the segment is skewed which leads to a sharper bend). In order to make handling of the curvature easier, the following construction distinguishes between straight and curved segments. Acceleration and deceleration along the rotated $X(I)$ axes are allowed on straight ($\eta = 0$) segments only, otherwise $\ddot{X}(I)$ is kept constant. Furthermore, a Boolean variable rws differentiates forward ($\mathsf{rws} = 0$) from reverse ($\mathsf{rws} = 1$) driving.

Construction 4 (Single track model constraints)
Given a set $\mathcal{I}$ *of heading intervals, the simple single track model is approximated by a constraint*

$$\Phi :\equiv \Phi_{cont} \wedge \bigvee_{I \in \mathcal{I}} ((\neg\mathsf{rws} \rightarrow \Phi_{fwd}(I)) \wedge (\mathsf{rws} \rightarrow \Phi_{rws}(I)))$$

with $\Phi_{fwd}(I) :\equiv \Phi_H(I) \wedge \Phi_{BB}(I) \wedge \Phi_{valid}(I) \wedge \Phi_v(I) \wedge \Phi_{acc}(I)$ *where*

$$\Phi_{cont} :\equiv \mathbf{p}_n{=}\mathbf{p}'_0 \wedge \dot{\mathbf{p}}_n{=}\dot{\mathbf{p}}'_0 \wedge (\dot{\mathbf{p}}_n{=}\mathbf{0} \rightarrow \underline{\mathsf{head}}{=}\overline{\mathsf{head}}{=}\underline{\theta'}{=}\overline{\theta'}) \wedge (\dot{\mathbf{p}}_n{\neq}\mathbf{0} \rightarrow (\mathsf{rws}{\leftrightarrow}\mathsf{rws}'))$$

$$\Phi_{BB}(I) :\equiv \bigwedge_{bound \in BB} \left(\underline{\mathsf{bound}} \leq \min_{\theta \in I} bound(\theta) \wedge \overline{\mathsf{bound}} \geq \max_{\theta \in I} bound(\theta)\right)$$

$$\Phi_H(I) :\equiv [\underline{\mathsf{head}}, \overline{\mathsf{head}}] \supseteq I \wedge \bigwedge_{k=0}^{n-1} \left(\dot{X}(I)_k \geq 0 \wedge |\dot{Y}(I)_k| \leq \dot{X}(I)_k \tan\eta\right)$$

$$\Phi_{valid}(I) :\equiv \bigwedge_{k=0}^{n-2} \left(\underbrace{\ddot{X}(I)_k = 0}_{\eta > 0 \text{ only}} \wedge \left|\frac{\ddot{Y}(I)_k}{\Delta^2}\right| \leq a_{lat,\max} \right) \wedge \bigwedge_{k=0}^{n-1} \left(\dot{X}(I)_k \geq \frac{2(n-1)L\tan(\eta)}{\tan(\delta_{\max})} \right)$$

in case $\eta = 0$*:*

$$\Phi_v(I) :\equiv \bigwedge_{k=0}^{n-1} \underline{\mathsf{v}}_k \leq \Delta^{-1}\dot{X}(I)_k \leq \overline{\mathsf{v}}_k$$

$$\Phi_{acc}(I) :\equiv \bigwedge_{k=0}^{n-2} \underline{\mathsf{acc}}_k \leq \Delta^{-2}\ddot{X}(I)_k \leq \overline{\mathsf{acc}}_k$$

in case $\eta > 0$*:*

$$\Phi_v(I) :\equiv \bigwedge_{k=0}^{n-1} \left(\underline{\mathsf{v}}_k \leq \Delta^{-1}\dot{X}(I)_k \wedge \overline{\mathsf{v}}_k \geq \frac{\dot{X}(I)_k + \tan\left(\frac{\eta}{2}\right)|\dot{Y}(I)_k|}{\Delta} \right)$$

$$\Phi_{acc}(I) :\equiv \bigwedge_{k=0}^{n-2} (\underline{\mathsf{acc}}_k \leq -|\Delta^{-2}\ddot{Y}(I)_k| \wedge \overline{\mathsf{acc}}_k \geq |\Delta^{-2}\ddot{Y}(I)_k|)$$

The constraint $\Phi_{rws}(I)$ *is the same as* $\Phi_{fwd}(I)$ *except for the following replacements:* $\dot{X}(I)_k$ *by* $-\dot{X}(I)_k$*,* $\underline{\mathsf{v}}_k$ *by* $-\overline{\mathsf{v}}_k$*, and* $\overline{\mathsf{v}}_k$ *by* $-\underline{\mathsf{v}}_k$*.*

In the construction, Φ_{cont} ensures that the next segment starts with the heading and speed at the end of the current segment. On each segment, $\Phi_{fwd}(I)$ and $\Phi_{rws}(I)$ encode forward resp. backwards movement within a heading interval I. Here, $\phi_H(I)$ ensures that the movement direction on the current slice is within the given interval. The remaining constraints ensure that curvature and lateral acceleration are within validity range of the single track model ($\phi_{valid}(I)$), and that bounding box ($\phi_{BB}(I)$), velocity ($\phi_v(I)$), and acceleration ($\phi_{acc}(I)$) bounds are valid.

Now we have everything together to encode a TSC into a CSP:

Construction 5 (CSP for a TSC)
For a TSC consisting of a basic chart BC*, and some unrolling depth* $N \in \mathbb{N}$*, let*

- $\Omega, (\phi^i)_i = BC2SAT(\mathsf{BC}, (true)_{i=0,\ldots,N})$ *using Algorithm 1*
- $\phi^i_{sv} :\equiv tr(\varphi_{sv}, i)$ *for each spatial view and time slice* $i = 0, \ldots, N-1$*, where* φ_{sv} *is the semantics of sv according to Definition 1, and* $tr(\varphi_{sv}, i)$ *is the encoding* $tr(\varphi_{sv})$ *(Construction 2) of* φ_{sv} *using variables for slice* i
- Φ^i_o *the encoding of the single track model (Construction 4) for each vehicle* o *and each time slice* $i = 0, 1, \ldots, N-1$

The CSP that encodes the TSC is then

$$\mathsf{CSP} :\equiv \Omega \wedge \phi^N \wedge \bigwedge_{i=0}^{N-1} \bigwedge_{sv} (\mathsf{s}^i_{sv} \rightarrow \phi^i_{sv}) \wedge \bigwedge_{i=0}^{N-1} \bigwedge_{o} \Phi^i_o \ .$$

5 Correctness

This section presents the mathematical properties of the construction that together ensure formal correctness of the approach. The proofs can be found in the Appendix. Recall that the constraint Φ from Construction 4 is instantiated for every slice and vehicle. Each solution of the resulting constraint system contains control point values that can be used to construct a vehicle trajectory according to Equation (1).

The first lemma describes the relation between the Bézier splines and the single track model differential equations given in Definition 4.

Lemma 1 (Trajectories from CSP solutions)
Assume Bézier control points $\mathbf{p}^i_k = (\mathsf{x}^i_k, \mathsf{y}^i_k)^T$ *(for* $k = 0, \ldots, n$ *and* $i = 0, \ldots, N-1$*), construct a trajectory* $\mathbf{p}$ *as defined in Equation (1), and let*

$$v(t) = |\dot{\mathbf{p}}(t)| \quad \text{and} \quad \delta(t) = \arctan(\kappa'(t)L)$$

where

$$\kappa'(t) = \begin{cases} \frac{\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)}{(\dot{x}(t)^2 + \dot{y}(t)^2)^{3/2}} & \text{if } v(t) \neq 0 \\ 0 & \text{else.} \end{cases}$$

If Φ *(as defined in Construction 4) holds for all consecutive pairs of time slices (i.e., after setting* $\mathsf{x}_k = \mathsf{x}^i_k$*,* $\mathsf{x}'_k = \mathsf{x}^{i+1}_k$*,* $\mathsf{y}_k = \mathsf{y}^i_k$*,* $\mathsf{y}'_k = \mathsf{y}^{i+1}_k$*, and likewise for all other variables in* Φ*,*

for $i = 0, \dots, N-1$), then there exists a continuous function $\theta : \mathbb{R}_{\geq 0} \to \mathbb{R}$ such that the single track equations of movement (Definition 4) are satisfied.

This is mainly achieved by the constraint Φ_{cont} that ensures the required smoothness of the trajectory, even in the case of vanishing speed. The above lemma gives straightforward instructions how to extract a vehicle trajectory and speed profile from the solutions of the CSP: the solution directly contains the Bézier spline control points that describe the vehicle trajectories, and speed, acceleration, and steering angle can be derived from their first and second derivatives. Only deriving the heading angle is a bit more tricky. If the vehicle speed is nonzero, the heading angle (at least modulo 2π, which is sufficient for most practical applications) can be derived from the velocity vector, e.g.

$$\mathsf{heading}(t) = \mathtt{atan2}(\dot{y}(t), \dot{x}(t))$$

using the `atan2` function that is avaliable in most programming language and converts a vector into an angle.[2] In standstill the result of the above function is undefined. In this case, however, Φ_{cont} (see Construction 4) asserts that (on the time slice that contains t) $\underline{\mathsf{head}} = \overline{\mathsf{head}} = \theta(t) + 2k\pi$ for some integer k. The relation between $\underline{\mathsf{head}}$, $\overline{\mathsf{head}}$ and the actual heading angle $\theta(t)$ is also described in Lemma 3 later in this section.

For the following lemmas, we fix some Bézier spline trajectory and construct $v(t)$ and $\delta(t)$ as in Lemma 1. Furthermore, we fix some time slice $[t_{i-1}, t_i)$ and some heading interval $I = [\hat{\theta} - \eta, \hat{\theta} + \eta]$ with $\hat{\theta} \in \mathbb{R}, \eta \in [0, \pi/2)$. Again, we drop the superscripts.

The next lemma states that the generated trajectory also fulfills the steering angle and lateral acceleration constraints imposed by the single track model.

Lemma 2 (Trajectories adhere to vehicle dynamics)
If both the heading constraint $\Phi_H(I)$ and the constraint $\Phi_{valid}(I)$ hold, then $|\kappa(t)| \leq \frac{\tan(\delta_{\max})}{L}$ and $|a_{lat}(t)| \leq a_{lat,\max}$ for all $t \in [t_{i-1}, t_i]$.

The last two lemmas finally state that the used approximations for speed and acceleration are correct.

Lemma 3 (Correctness of upper and lower bounds)
If both the heading constraint $\Phi_H(I)$ and $\Phi_v(I)$ hold, then $\underline{v}(s) \leq v(t) \leq \overline{v}(s)$, $\underline{acc}(s) \leq acc(t) \leq \overline{acc}(s)$, $\underline{\mathsf{bound}} \leq bound(\theta(t)) \leq \overline{\mathsf{bound}}$ for $bound \in BB$, and $\exists k \in \mathbb{Z}. \underline{\mathsf{head}} \leq \theta(t) + 2k\pi \leq \overline{\mathsf{head}}$ for all $t \in [t_{i-1}, t_i]$ and $s = \frac{t - t_{i-1}}{t_i - t_{i-1}}$.

[2] The order of arguments and the output range may vary between programming languages.

Both lemmas 2 and 3 above describe a relationship between the heading constraint $\Phi_H(I)$ and the other constraints Φ_{valid}, Φ_v, and Φ_{acc} that are part of Construction 4. Because the constructed CSP asserts that spline segment fulfills both $\Phi_H(I)$, $\Phi_{valid}(I)$, $\Phi_v(I)$, and $\Phi_{acc}(I)$ for some heading interval $I \in \mathcal{I}$, the generated trajectories are correct.

Corollary 1 (Correctnes of Construction 4)
Bézier splines satisfying Construction 4 are valid trajectories for the simple single track model, with valid bounds for velocity and acceleration.

The last two lemmas state how a scenario instance φ derived from a solution to the CSP relate to an abstract scenario. The first one describes the properties of $tr(\varphi)$ (Construction 2).

Lemma 4 (Correctness of Construction 2)
Assume a scenario instance σ where all vehicle trajectories have been constructed as described in Lemma 1, and a linear constraint φ over object attributes from the scenario instance.

If the constraint Φ (Construction 4) holds for all vehicles, and the constraint $tr(\varphi_{sv})$ (Construction 2) holds, then $\sigma(t) \models sv$ for all $t \in [t_{i-1}, t_i)$.

The second one states that if the spatial views are correctly encoded in the scenario and linked to the Boolean state variables for the invariant nodes, then the constraint system generated by `BC2SAT` (Algorithm 1) correctly encodes the abstract scenario.

Lemma 5 (Correctness of Algorithm 1)
For some basic chart BC and a family $(\mathsf{b}^i)_{i=0,\dots,N}$ of $N+1$ Boolean variables let $\Omega, (\phi^i)_i = BC2SAT(\mathsf{BC}, (true^i)_i)$.

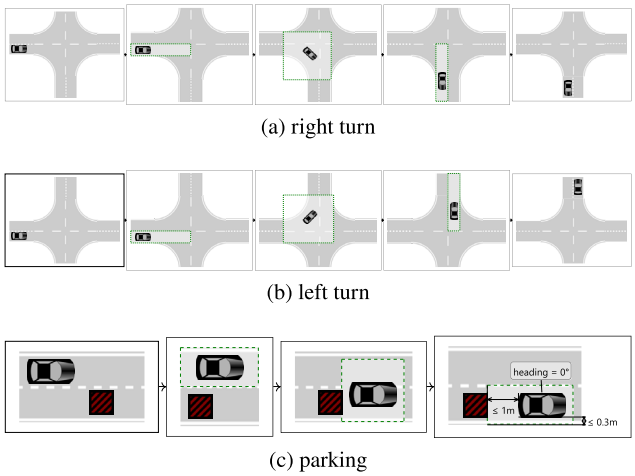
If Ω and ϕ^N hold, then $\sigma, [p\Delta, q\Delta) \models \mathsf{BC}$.

The proof idea is to show by induction over subcharts BC of the TSC, respectively sub-calls $\Omega, (\phi^i)_i = \mathrm{BC2SAT}(\mathsf{BC}, (\mathsf{b}^i)_i)$, that

- there is a unique index q with $\mathsf{b}^i = false$ for $i < q$ and $\mathsf{b}^i = true$ for $i \geq q$, marking the start of the interval where BC is evaluated on, and
- ϕ^q evaluates to *true* only if $q > p$ and $\sigma, [p\Delta, q\Delta) \models \mathsf{BC}$.

All together this, implies correctness of the encoding of TSCs.

Corollary 2 (Correctnes of solutions of the CSP)
When encoding a TSC into a CSP as described in Construction 5, each solution of the CSP yields a scenario instance of the TSC.



(a) right turn

(b) left turn

(c) parking

Fig. 6 Additional demonstration scenarios

6 Demonstration

In order to demonstrate the applicability of the approach, this section presents experimental results with four different abstract scenarios: an overtaking (Fig. 1), a right turn (Fig. 6a), a left turn (Fig. 6b), and a parking scenario (Fig. 6c). The four scenarios have been selected in order to contain a majority of the basic driving maneuvers that commonly occur in urban traffic scenarios. In particular, they map to the basic maneuvers identified by Hartjen et al. [25] as follows:

- Overtaking: keep velocity, follow lane, change lane, follow object,[3] passing
- Left/right turn: follow lane, approach/cross junction, left/right turn[4]
- Parking: driveaway,[5] halt, approach object, reversing, park

For each of the scenarios, several trajectories are instantiated. The trajectories are then used as test cases for a prototypical controller implementation. With the help of the CARLA driving simulator, it is evaluated how close the controller can follow the provided trajectories.

6.1 Trajectory generation

The vehicle dynamics encoding given in this paper has been integrated into the scenario instantiation framework for TSCs that has been presented in previous work [5, 9]. As a CSP solver, Z3 [16] is used. The TSCs (temporal structure and spatial views) given in Figs. 1 and 6 are translated to BMC problems using the encoding scheme sketched in Sect. 3 and the related work [5, 9]. Vehicle dynamics are encoded with quadratic Bézier splines ($n = 2$) using Construction 4 given in this paper. Table 1 lists the heading intervals $\mathcal{I}$, unrolling depth N, and step size Δ used in each scenario. All scenarios use a fictive vehicle model with a maximum speed of 120 km/h, (20 km/h when reversing), maximum acceleration of 1 ms^{-2} (both forward and reverse), and braking deceleration of $-5\,\mathrm{ms}^{-2}$. The single track model vehicle parameters (except heading ranges which are scenario-dependent) are listed in Table 2. All scenarios use a lane width of 3 m. The crossing in the turning scenario has a corner radius of 6 m as shown in Fig. 7.

For each of the SMT problems, the solver can find a satisfying model within less than 10 seconds.[6] In Fig. 8, one of the generated trajectories is plotted for each abstract scenario. The shaded areas show the outer bounding box approximation calculated by the solver. Note that the inequalities $\Phi_{BB}(I)$ in Construction 4 allow the solver to assume any bounding approximation box that fits the vehicle, so the bounding boxes are not necessarily tight. It can be observed that the solver tends to choose the maximum bounding box that satisfies the current spatial view.

6.2 Scenario simulation

The presented concrete scenarios show that trajectories can be found for all basic maneuvers. The trajectories are correct by construction with respect to the chosen vehicle model and abstract scenario. However, the single track vehicle model simplifies actual vehicle dynamics. For example, side slip of the tires is neglected. As a consequence, when a simulator executes the scenario with a more accurate vehicle physics model, the simulated trajectory may differ from the ideal one calculated by the CSP solver. In order to give an upper bound for the simulation error for the demonstration scenarios, the generated trajectories have been executed with the CARLA 0.9.14 driving simulator. As a vehicle, the CARLA built-in model of an Audi A2 has been chosen, which has nearly the same wheel base as our fictive vehicle. In the experiment, a custom controller implementation has been used that operates in the simulation loop. In each simulation step, the controller calculates throttle/brake and steering values by comparing the current actual and planned vehicle states.

For the description of the control algorithm, we use the following input variables:

- $\mathbf{p}_k$, $\mathbf{v}_k$, $\mathbf{h}_k$: actual position, velocity, and heading vectors of the CARLA actor in step k
- $\mathbf{p}'_k$, $\mathbf{v}'_k$, $\mathbf{a}'_k$, $\mathbf{h}'_k$, κ'_k: position, velocity, acceleration, heading vector, and curvature of the reference trajectory at time $t_k = kT$

[3] keep distance to other vehicle.

[4] 90° turn on junction.

[5] acceleration from standstill.

[6] The results have been produced using Z3 v4.8.14 running on a Windows 10 notebook with Intel Core i7-1185G7 CPU @ 3.00 GHz and 16 GB RAM.

Table 1 Heading intervals, number of slices (N), and slice length (Δ) for the demonstration scenarios

scenario	intervals $\mathcal{I}$	#slices N	slice length Δ
overtaking	$[0,0]°$, $[-10,10]°$	20	2 s
parking	$[0,0]°$, $[-15,15]°$, $[15,15]°$, $[0,30]°$	8	3 s
left turn	$[0,0]°$, $[0,45]°$, $[45,90]°$, $[90,90]°$	8	3 s
right turn	$[0,0]°$, $[0,-45]°$, $[-45,-90]°$, $[-90,-90]°$	8	3 s

Table 2 Vehicle parameters for the demonstration scenarios

parameter	variable	value
max. steering angle	$\delta_{\max}$	45.0°
front	F	0.8 m
wheel base	L	2.8 m
rear	G	0.7 m
width	W	1.8 m

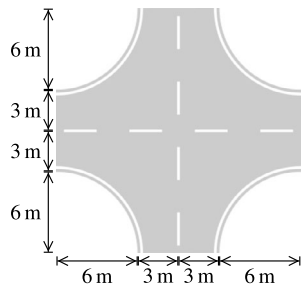
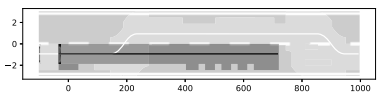
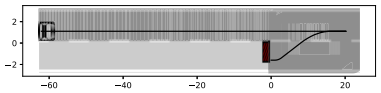


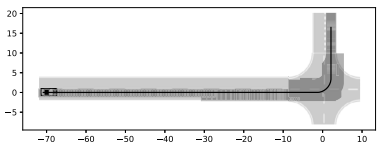
Fig. 7 Dimensions of the crossing in the demonstration scenarios



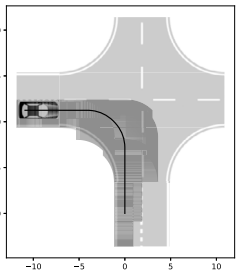
(a) overtaking



(b) parking



(c) left turn



(d) right turn

Fig. 8 Generated trajectories; the solid lines show vehicle trajectories, and the shaded areas the bounding box approximation; axes are in Meters

- $\mathbf{n}_k$ and $\mathbf{n}'_k$ are unit vectors orthogonal to $\mathbf{h}_k$ respectively $\mathbf{h}'_k$

Table 3 Parameter values for Algorithm 2

parameter	variable	value
simulation step size	T	0.01 s
position error weight	T_2	0.1 s
acceleration weight	T_1	0.05 s
minimum lookahead distance	D	1.0 m
PID proportional weight	K_p	$1.2\ \frac{1}{\mathrm{m}\cdot\mathrm{s}^{-1}}$
PID integral weight	K_i	$0.2\ \frac{1}{\mathrm{m}\cdot\mathrm{s}^{-1}}$
PID differential weight	K_d	$0.1\ \frac{1}{\mathrm{m}\cdot\mathrm{s}^{-1}}$
wheel base	L	2.8 m;
max. steering angle	$\delta_{\max}$	60.0°

- rws_k: Boolean flag indicating whether the vehicle drives in reverse in step k

All other constants are listed in Table 3. Algorithm 2 shows the complete control algorithm for the CARLA vehicle.

For longitudinal control (lines 3–15), a simple discrete PID controller is used. It uses a target speed v_{tgt} as input which is calculated from the current position error and the velocity on the generated trajectory. The method

$$\mathrm{clamp}(x) := \begin{cases} -1 & \text{if } x \le -1 \\ 1 & \text{if } x \ge 1 \\ x & \text{else} \end{cases}$$

clamps a value to the interval $[-1,1]$.

For lateral control, a custom strategy is used. The idea is to calculate some Bézier curve that leads the vehicle back to the reference trajectory. This is done by constructing two quadratic Bézier curves with common end point $\hat{\mathbf{p}}_k$ as shown in Fig. 9. The first curve starts with the current position $\mathbf{p}_k$ and heading direction $\mathbf{h}_k$ of the vehicle. The second curve starts with the planned position $\mathbf{p}'_k$ and planned movement direction $\mathbf{h}'_k$. For both curves, the bend point has a distance d_1 to the start, so is located at $\mathbf{q}_k = \mathbf{p}_k + d_1\mathbf{h}_k$ resp. $\mathbf{q}'_k = \mathbf{p}'_k + d_1\mathbf{h}'_k$. We choose the common point $\hat{\mathbf{p}}_k$ such that the second curve (with control points $\mathbf{p}'_k$, $\mathbf{q}'_k$, and $\hat{\mathbf{p}}_k$) has, in point $\mathbf{p}'_k$, the same curvature κ'_k as the reference trajectory. This is satisfied for any point $\hat{\mathbf{p}}_k = \mathbf{p}'_k + 2d_1^2\kappa'_k\mathbf{n}'_k + d_2\mathbf{h}'_k$ with $d_2 \in \mathbb{R}$. For the control algorithm, we let $d := \max(D, |\mathbf{v}'| \cdot T_1)$ and

Algorithm 2 Control algorithm for the CARLA actor

1: $I_{-1} \leftarrow 0,\ e_{-1} \leftarrow 0$
2: **for** each step $k = 0, 1, 2, \ldots$ **do**
3: ▷ *longitudinal control:* ◁
4: $v_{tgt} \leftarrow \left(\frac{\mathbf{p}'_k - \mathbf{p}_k}{T_1} + \mathbf{v}'_k + T_1 \mathbf{a}'_k\right) \cdot \mathbf{h}_k$
5: $e_k \leftarrow v_{tgt} - |\mathbf{v}_k|$
6: **if** $\mathsf{rws}_k = true$ **then**
7: $e_k \leftarrow -e_k$
8: $I_k \leftarrow \mathrm{clamp}(I_{k-1} + K_i e_k T)$
9: $u \leftarrow \mathrm{clamp}(K_p e_k + I_k + \frac{K_d}{T}(e_k - e_{k-1}))$
10: **if** $u \geq 0$ **then**
11: $\mathsf{throttle}_k \leftarrow u$
12: $\mathsf{brake}_k \leftarrow 0$
13: **else**
14: $\mathsf{throttle}_k \leftarrow 0$
15: $\mathsf{brake}_k \leftarrow -u$
16: ▷ *lateral control:* ◁
17: $d \leftarrow \max(D, |\mathbf{v}'_k| \cdot T_1)$
18: **if** $\mathsf{rws}_k = true$ **then**
19: $d \leftarrow -d$
20: $\kappa'_k \leftarrow -\kappa'_k$
21: $\hat{\mathbf{p}} \leftarrow \mathbf{p}' + 2d\mathbf{h}' + \kappa' d^2 \mathbf{n}'$
22: $\kappa \leftarrow \frac{(\hat{\mathbf{p}} - \mathbf{p}) \cdot \mathbf{n}}{d^2}$
23: $\mathsf{steer}_k \leftarrow \mathrm{clamp}\left(\frac{\arctan(\kappa \cdot L)}{\delta_{\max}}\right)$

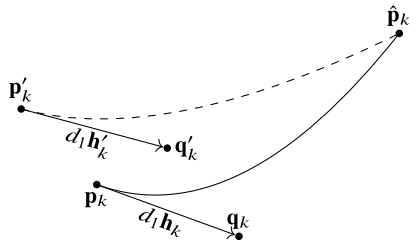


Fig. 9 Lateral control strategy: compute the steering angle from a quadratic spline (solid line) that starts from the current position $\mathbf{p}$ and meets another spline (dashed) which has the same curvature and heading as the reference trajectory

set $d_1 = \frac{d}{\sqrt{2}}$ and $d_2 = 2d$. The steering value is then derived from the curvature κ_k of the first curve in points $\mathbf{p}_k$.

Each simulation run then consists of three phases depicted in Fig. 10.

1. **Actor drops down.** The vehicle is spawned in the position $(0, 0)$. Due to possible bounding box collisions, vehicles in CARLA have to be spawned in the air above the ground. The first 5 simulated seconds are used to let the vehicle drop down to the road surface.
2. **Settling in.** After 5 s simulation time, the PID controller is given time to settle in: the PID controller is used to accelerate the vehicle from standstill to the initial velocity of the scenario. In this phase, only lines 4–14 of Algorithm 2 are executed using $v_{tgt} = |\mathbf{v}_0|$.
3. **Scenario execution.** After 10 s, the PID controller is assumed to be in a stable state and the target velocity reached. Now, the vehicle is teleported to the initial position of the scenario and execution of the scenario (Algorithm 2) starts with $k = 0$. During this phase, both the lateral and longitudinal control strategies are effective.

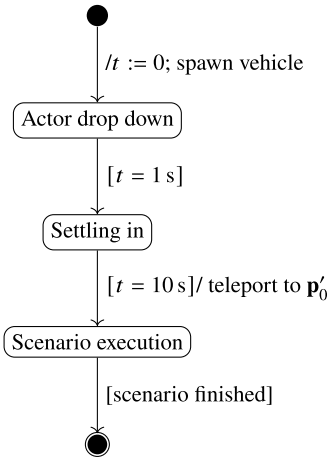


Fig. 10 Experiment execution

Table 4 Error between ideal and simulated trajectories, in meters

	bounds	lat	lon	heading
mean	0.197	0.027	0.182	1.156
min	0.053	< 0.001	0.040	0.001
25%	0.115	0.010	0.097	0.648
50%	0.156	0.031	0.115	0.991
75%	0.196	0.040	0.196	1.844
max	2.126	0.153	2.126	2.616

For experiment evaluation, only the third phase is considered.

6.3 Experiment evaluation

Table 4 shows the distribution of the error among the experiments, in terms of lateral (lat) and longitudinal (lon) distance of the actual reference point to the calculated trajectory, heading angle (heading), and absolute distance error in the corners of the vehicle bounding box (bounds). Recall that the trajectories which are used as a reference to calculate these errors are extracted from the solutions of the CSP. They are sound instances of the abstract scenario by construction. The calculated errors can therefore be seen as an upper bound for the deviation from the abstract scenario. In total, 100 instances have been generated and simulated for each abstract scenario, using the model enumeration technique described

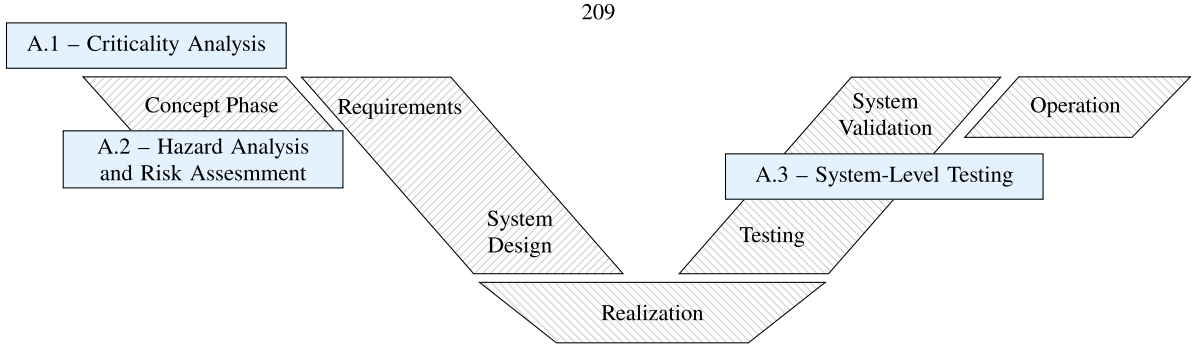


Fig. 11 Applications for SMT-based abstract scenario instantiation in the development process structured along the V-model of ISO 26262

by Borchers et al. [9]. Scalability and performance of the encoding have been evaluated in earlier work already [5, 9] and are therefore not addressed in the experiments.

The controller aims at minimizing the error in the vehicle reference point located at the rear axis. As a consequence, the bounding box error is dominated by the heading and longitudinal errors. Choosing another reference point in the simulation may reduce this effect.

Compared to the previously published results [4] achieved on a single instance of each abstract scenario, the extended results show that the generated trajectories are differently challenging for the controller, especially the speed profile. The maximum lateral deviation is quite moderate. In 75% of the experiments, it is below 4.0 cm. This is comparable to the performance achieved by Kong et al. [29] using a model predictive controller. The maximum lateral error is still below 16.0 cm, which indicates good stability of the lateral controller in the demonstration scenarios.

The longitudinal and overall error is below 20.0 cm in 75% of the experiments, which is acceptable, considering that the prototype PID controller is not optimally tuned. The maximum error of 2.12 m indicates that the acceleration profile in some of the generated trajectories brings the PID controller into an unstable state. Hence, the longitudinal performance can probably be improved, e.g., by replacing the used discrete PID controller by a model predictive control algorithm [29].

In summary, the experimental results show that the approach can successfully generate varying concrete scenarios involving different basic maneuvers. Overall, the simple single track model is adequate to model the vehicle physical behavior in those scenarios. In the abstract scenarios used for the experiments, the acceleration profiles are quite unrestricted, which leads to a wide variety in the generated instances. Note, that by the nature of the used solving techniques (cf. [6]) the constraint solver tends to push the solutions to the border of the solution space. Hence, the results are represent for the maximal errors, but cannot be expected to follow a meaningful distribution. Techniques to maximize and measure variety of the solutions are elaborated by Borchers et al. [9].

7 Application

In this section we discuss three possible applications for the abstract scenario instantiation using the safe linear encoding of vehicle dynamics, as presented in the previous sections.

Specifically, we consider its application within a criticality analysis, for hazard analysis and risk assessment (HARA), and for virtual system level testing. As to frame these applications adequately, we mapped them to the general development process for automotive systems in form of a V-model, cf. Fig. 11.

7.1 Criticality analysis for automated driving systems

A criticality analysis for automated driving systems seeks to structure the operational domain regarding the emergence of criticality [36, 37]. The overall concept is to identify and formalize safety-critical influencing factors [1, 55] in the operational domain, called *criticality phenomena*, followed by an analysis of their *causal relations* [30]. In contrast to a HARA, a criticality analysis abstracts from a concrete system but is performed for a whole class of systems. Therefore, it is located outside the V-model and can be performed before even the concept phase, cf. Fig. 11. Another difference is the focus on *criticality* – as a quantifiable risk surrogate – instead of risk itself which is often evaluated qualitatively based on expert judgment. The results of a criticality analysis can be used throghout the development process, for example, in a subsequent HARA, to derive valuable test cases, within a safety argumentation or for the derivation of requirements on simulation systems and models [38].

A criticality analysis interfaces the scenario-based approach in the so-called scenario branch, as depicted by Fig. 12. More precisely, the criticality phenomena and their

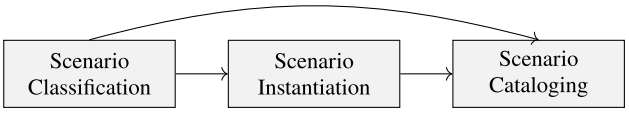


Fig. 12 Schematic view of the scenario branch in a criticality analysis for automated driving systems, adapted from [37, Fig. 2]

causal relations which are identified by a criticality analysis are used to form scenario classes that contain them (*scenario classification*). For scenario classification, we aim at a high abstraction level to keep specification efforts manageable. Therefore, the criticality analysis utilizes abstract scenarios here. These abstract scenarios are cataloged and contribute to the knowledge basis (*scenario cataloging*). They are built on a formal specification and can therefore be used at any point in the development process.

An integral part of a criticality analysis for automated driving systems is the quantitative evaluation of such scenarios using criticality metrics [56]. For this, the scenarios classes have to be instantiated as concrete scenarios (*scenario instantiation*). When the abstract scenarios in a criticality analysis are specified as TSCs, the instantiation approach presented in this work can be applied to obtain realistic concrete trajectories for criticality evaluation.

7.1.1 Example: 'reduced coefficient of friction'

Next, we consider an example how the proposed abstract scenario instantiation approach can contribute to a criticality analysis. Note that the results from Sect. 6.2 indicate that there may be a noticeable difference between a generated concrete scenario and its execution in a simulation. The prototype agent controller responsible for these observed deviations is presented in Sect. 6.2. As a criticality analysis does not need a specific model of the system under development, but rather a simple, generic controller this custom controller fits perfectly. This can be leveraged to analyze of criticality phenomena virtually.

Many criticality phenomena are related to the environment such as *heavy rain*, *glare*, or *limited global illumination*. Here, we consider the rather abstract criticality phenomenon *reduced coefficient of friction* which subsumes all conditions that reduce a vehicle's tire-to-road friction.

Assuming that environmental conditions are implemented validly in a simulator, we can evaluate the effect of these conditions on the scenario (in form of the actors' trajectories). In the experiments carried out so far, optimal road conditions (as assumed by the scenario instantiation approach) have been simulated. For most of the generated scenario instances, the simulation runs are close to the generated concrete scenario.

The idea for the analysis of the criticality phenomenon *reduced coefficient of friction* is to simulate a concrete scenario under different environmental conditions. From the abstract scenario depicted by Fig. 13, 100 different solutions of the corresponding CSP have been generated, leading to 25 different concrete scenario instances.[7] Each instance was simulated both with and without the area of reduced friction. In order to assess the influence of the reduced friction to the criticality of the scenario, we measure the *time to collision* (TTC) between the ego vehicle and the truck, cf. [56, Sect. 5.2.1]. The TTC estimates the remaining time until an imminent collision would occur provided both traffic participants keep their current direction and speed. Naturally, a low TTC indicates a high criticality of the situation. Figure 14 shows the minimal TTC in each simulation run without (x-axis) vs. with reduced friction area (y-axis). One experiment has been discarded due to obviously unrealistic physics in the simulation. The samples below the diagonal line show instances where the simulation with reduced friction area ended up in a lower TTC than the simulation without reduced friction. In 12 of the 24 remaining scenario instances, the TTC differs between the two simulations, all showing a notably smaller TTC in the simulation with reduced friction. The samples at the bottom of the figure have a zero TTC which indicates a collision, which is the case for 10 simulation runs. Overall, the simulations demonstrate a high impact of the reduced friction area on the criticality of the scenario.

7.2 Hazard analysis and risk assessment

For the development of safe automated driving system, a systematic hazards analysis and risk assessment (HARA) that accompanies the specification and design phase is indispensable, cf. ISO 26262 [27] and ISO 21448 [26]. This includes a systematic investigation of the systems functionality and its operational design domain. For exaple, Kramer et al. propose a comprehensive HARA approach – which emphasizes risks arising due to the increased level of automation – to uncover hazardous scenarios [8, 31]. Again, a formal specification of hazardous scenarios as abstract scenarios in the concept phase has many advantages: scenarios are unambigous, usuable throughout the development process, and executable in a simulator.

Simulations in early concept phases can be helpful to analyze hazardous scenarios and to gauge parts of the associated risk: controllability and severity. When hazardous scenarios are specified using TSCs, the presented abstract scenario instantiation approach can be applied here to assess concrete instances virtually. Similar to criticality phenomena, hazards can be included in the scenario following instantiation and before execution in a simulator.

However, in the concept phase no complex ADS model might be available. If a preliminary ADS model based on the

[7] Many solutions differ only in auxiliary variables and therefore encode identical scenario instances. These duplicates have been dropped from the results.

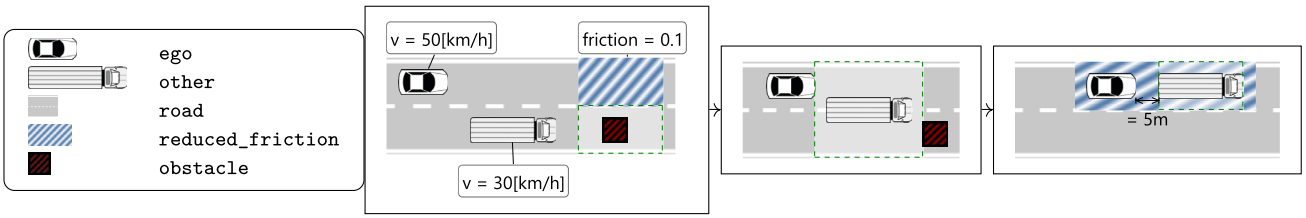


Fig. 13 Cut in scenario with an area of reduced friction

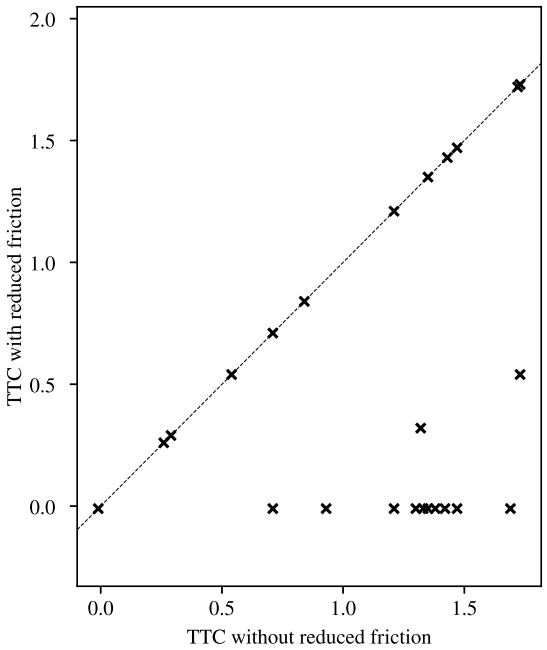


Fig. 14 Minimal TTC within each instance of the scenario "cut in with reduced friction", simulated without (x-axis) vs. with reduced friction area (y-axis). The diagonal line marks the 1:1 ratio between TTCs

item definition is already available, this early model should be inserted as controller for the ego vehicle – again after instantiation but prior to execution. In this case, parts of the ego's trajectory will have to removed after the SMT-based instantiation such that the ADS model's behavior in the hazardous scenario can be evaluated by the safety engineer. This could, for example, mean the removal of all points of the ego's trajectory except start- and end point (and some additional way points in between) [5, 48].

7.3 Virtual system-level testing of automated driving systems

It is a widely accepted fact that the number of test cases that are necessary to prove safety of an automated driving system is so large that it is impossible to execute all test cases physically [53]. Although physical testing cannot (yet) be skipped completely from the verification process, a safety argumentation can hardly be constructed without extensive simulation. In virtual testing, a simulator replaces the physical environment for the system under test (SuT). Depending on the aspects of the system that shall be tested (e.g. maneuver planning, sensor data processing, perception), it is possible in this setting to use only the software or an executable model of the SuT. This leads to a family of test setups that is commonly *X-in-the-loop* (XiL) testing, where *X* may be a model, software, or even hardware [46]. XiL testing saves costs and lowers risks to test drivers and test vehicle environment. Furthermore, it allows to perform much more tests in the same time than it would be possible with physical testing [7].

For a reliable safety argumentation, a test strategy is needed that ensures that sufficiently many valuable tests have been carried out [10, 54]. In general, a test case consists of a scenario and pass/fail criteria. For testing, so-called concrete scenarios are required that deterministically define all the inputs to the SuT. The state of the art of test case generation from functional scenarios uses a layered approach to detail down the scenarios. Data formats such as OpenSCENARIO [44, 45] drive state of the art simulation environments. They define behavior in terms of actions (e.g. a lane change or following the SuT) triggered by events and situations (e.g. distance to the SuT or simulation time). This requires behavior models for traffic participants that are usually provided by the simulation tool, so exchanging scenario descriptions between tools is problematic, even if they share the same data format. Furthermore, representativeness (wrt. real-world traffic participants behavior) of these models is questionable.

Borchers et al. present a prototypical toolchain for virtual verification of automated driving systems [9] which utilizes the abstract scenario instantiation presented in this work. Therein, concrete scenarios are sampled directly from an abstract test case. The result consists of explicit trajectories that do not depend on simulator-specific implementations of driving maneuvers. Again, these concrete scenarios cannot be used a test cases directly. Before executing the concrete scenario in the simulation, the SuT's trajectory has to be (partially) removed so that the evaluate the model's behavior.

8 Conclusion

This paper describes a method for encoding abstract scenarios to linear constraint solving/BMC problems. The encoding of vehicle dynamics, given in detail in Sect. 4, uses a simple single track model. The main ideas are to describe the vehicle trajectories as Bézier splines that are parameterized by time. So, the solution of the constraint problem determines the position of every vehicle to every point in time. Linear constraints give safe bounds for velocity and acceleration, by a discretization of the heading angle.

Due to the linearizations and over-approximations, heading intervals, number, and length of time slices must be appropriately chosen for the abstract scenario. While finer discretizations increase the covered scenario space, they also increase the complexity of the CSP problem. The practical demonstration in Sect. 6 shows that the approach can generate trajectories for a majority of basic driving maneuvers, and that the simple vehicle dynamics yields trajectories that can be executed under realistic driving physics. The approach may be combined with model enumeration techniques (e.g. all-SMT [39] or recursive blocking [6]) in order to systematically generate different scenarios. Some enumeration techniques are evaluated by Borchers et al. [9], who investigate into strategies for finding an optimal coverage. Recursive atom blocking [9], that is used in the experiments from Sects. 6 and 7.1.1, explores the Boolean part of the CSP's solution space and thereby splits it in convex regions. The returned scenarios tend to be located at the borders of these regions. Overall, this results in a high diversity of the generated scenarios, although no formal statement can be made about coverage of the scenario space [9]. Neither we can make assumptions about the distribution of generated samples. Although physically plausible, it is therefore unclear, how well the sampled scenarios represent real world behavior. This is an open challenge that needs to be addressed in future research. Three possible applications of the presented instantiation approach regarding automated driving were sketched in Sect. 7: criticality analysis, HARA, and system-level testing. So far, the approach has been used for consistency analysis of TSCs [3], for scenario instantiation within a criticality analysis [5] (where also first scalability results have been published), and for virtual system-level testing of ADSs [9]. Future work comprises the extension to other domains besides automotive such as maritime and railway scenarios – which will also address the current shortcomings of the approach.

Appendix: Proofs

The following proofs show the case for forward driving only. Proofs for reversing are analogous.

Proof of Lemma 1

Given some time point t, call the ith slice, with $t_{i-1} \le t \le t_i$, the *current slice*. Assuming that the constraint Φ holds, there is some heading range $I_i = [\hat{\theta}_i - \eta_i, \hat{\theta}_i + \eta_i] \in \mathcal{I}$ such that $\Phi_H(I_i)$ and $\Phi_{valid}(I_i)$ hold. Furthermore, Φ_{cont} holds. The ODE system for the single track model can also be formulated as a linear ODE

$$\dot{\mathbf{h}}(t) = \begin{bmatrix} 0 & -v(t)\kappa(t) \\ v(t)\kappa(t) & 0 \end{bmatrix} \mathbf{h}(t) \qquad \dot{\mathbf{p}}(t) = v(t)\mathbf{h}(t) \quad (4)$$

with the constraint $|\mathbf{h}(t)| = 1$. It can be easily verified that the single track model ODEs solve the above linear ODE if $\mathbf{h}(t) = (\cos\theta(t), \sin\theta(t))^T$ and $\kappa(t) = \frac{\tan\delta(t)}{L}$. We show that the ODEs in Equation (4) hold on the current slice when setting

$$\mathbf{h}(t) = \begin{cases} (\cos\hat{\theta}_i, \sin\hat{\theta}_i)^T & \text{if } v(t) = 0 \\ \dot{\mathbf{p}}(t)/v(t) & \text{else.} \end{cases} \quad (5)$$

and $\delta(t)$ as in the Lemma. In case $v(t) \neq 0$, this is true by some basic analysis:

$$\begin{aligned} \dot{\mathbf{h}}(t) &= \frac{d}{dt}\frac{\dot{\mathbf{p}}(t)}{v(t)} = \frac{d}{dt}\begin{bmatrix} \frac{\dot{x}(t)}{\sqrt{\dot{x}(t)^2+\dot{y}(t)^2}} \\ \frac{\dot{y}(t)}{\sqrt{\dot{x}(t)^2+\dot{y}(t)^2}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\ddot{x}(t)\dot{y}(t)^2 - \ddot{y}(t)\dot{x}(t)\dot{y}(t)}{(\dot{x}(t)^2+\dot{y}(t)^2)^{3/2}} \\ \frac{\ddot{y}(t)\dot{x}(t)^2 - \ddot{x}(t)\dot{x}(t)\dot{y}(t)}{(\dot{x}(t)^2+\dot{y}(t)^2)^{3/2}} \end{bmatrix} \\ &= \begin{bmatrix} -\kappa'(t)\dot{y}(t) \\ \kappa'(t)\dot{x}(t) \end{bmatrix} = \begin{bmatrix} 0 & -v(t)\kappa(t) \\ v(t)\kappa(t) & 0 \end{bmatrix} \mathbf{h}(t) \end{aligned}$$

(note that $\kappa'(t) = \kappa(t)$). In case $v(t) = 0$, we have to show that $\dot{\mathbf{h}}(t) = 0$. Constraint $\Phi_{valid}(I)$ allows $v(t) = 0$ only if $\eta_i = 0$. By $\Phi_H(I)$ and Construction 3, this implies that we are on a straight segment with $\dot{\mathbf{p}}(t') = (\cos\hat{\theta}_i, \sin\hat{\theta}_i)^T v(t')$, and therefore $\mathbf{h}(t') = (\cos\hat{\theta}_i, \sin\hat{\theta}_i)^T$ constant for all $t' \in [t_{i-1}, t_i]$. So, $\dot{\mathbf{h}}(t) = 0$.

The remaining part, showing that no discontinuities at time slice boundaries occur, is a direct consequence of the constraint Φ_{cont}. □

For the following proofs, fix some heading interval $I = [\hat{\theta} - \eta, \hat{\theta} + \eta]$ with $\hat{\theta} \in \mathbb{R}, \eta \in [0, \pi/2)$, and time slice $[t_{i-1}, t_i)$ as the current time slice. Define the Bernstein polynomials $X(s) := \sum_{k=0}^{n} \mathcal{B}_k^n(s) X_k$ and $Y(s) := \sum_{k=0}^{n} \mathcal{B}_k^n(s) Y_k$ as a rotation of the current trajectory segment by $-\hat{\theta}$, with control points $(X_k, Y_k)^T$ given in Construction 3.

Proof of Lemma 2

Fix some $t \in [t_{i-1}, t_i]$, and let $s = (t - t_{i-1})\Delta^{-1}$. By property (P1) of Bézier curves, the constraints $\Phi_{valid}(I)$ and $\Phi_H(I)$

imply

$$\Delta v(t) = \sqrt{\dot{X}^2(s) + \dot{Y}^2(s)} \geq \dot{X}(s) \geq 0 \quad (\forall s \in [0,1])$$
$$|\dot{Y}_k| \leq \dot{X}_k \tan\eta \quad (\forall k = 0, 1, \ldots, n-1)$$

Recall that speed is invariant to rotation and, by Equation (2), scaled by Δ^{-1}. The curvature of the trajectory is also invariant to rotation and, with the above inequalities, bounded by

$$\begin{aligned}|\kappa(t)| &= \frac{|\dot{X}(s)\ddot{Y}(s) - \ddot{X}(s)\dot{Y}(s)|}{(\dot{X}^2(s) + \dot{Y}^2(s))^{3/2}} \\ &\leq \frac{|\ddot{Y}(s)| + |\ddot{X}(s)\tan\eta|}{\dot{X}^2(s) + \dot{Y}^2(s)} \\ &\leq \frac{|\ddot{Y}(s)| + |\ddot{X}(s)\tan\eta|}{\Delta^2 v^2(s)}\end{aligned}$$

Similar, lateral acceleration $a_{lat}(t) = \kappa(t)v^2(t)$ is bounded by

$$a_{lat}(t) \leq \frac{|\ddot{Y}(s)| + |\ddot{X}(s)\tan\eta|}{\Delta^2} \quad .$$

In the case $\eta = 0$, the bound $|\dot{Y}_k| \leq \dot{X}_k \tan\eta$ (for all $k = 0, 1, \ldots, n-1$) implies $\dot{Y}(s) = 0$ for all $s \in [0,1]$, so $\ddot{Y}(s) = 0$. As a consequence of the above equations, both $a_{lat}(t) = 0$ and $\kappa(t) = 0$.

For $\eta > 0$, the constraint $\Phi_{valid}(I)$ forces $\dot{X}(s) > 0$ and $\ddot{X}(s) = 0$. This simplifies the above approximation for a_{lat} to $a_{lat}(t) \leq |\ddot{Y}(s)|\Delta^{-2}$. Furthermore, it means that $\dot{X}(s)$ is constant. Recall that $\dot{X}$ and $\dot{Y}$ are Bernstein polynomials of degree $n-1$ with $\dot{X}_0 = \dot{X}_1 = \cdots = \dot{X}_{n-1} = \dot{X}(s) > 0$, and control points of $\dot{Y}$ can be bounded by $|\dot{Y}_i| \leq \dot{X}(s)\tan(\eta)$ for $i = 0, 1, \ldots, n-1$. By Bézier curve properties (P1, P3), $\ddot{Y}(s)$ is bounded by

$$|\ddot{Y}(s)| \leq \max_{k=0,\ldots,n-2}(n-1)|\dot{Y}_{k+1} - \dot{Y}_k| \leq 2(n-1)\dot{X}(s)\tan(\eta) \quad .$$

Using $\dot{X}(s)$ as a lower bound for $\Delta \cdot v(t)$, it is

$$|\kappa(s)| \leq \frac{2(n-1)\dot{X}(s)\tan(\eta)}{\dot{X}^2(s)} = \frac{2(n-1)\tan(\eta)}{\dot{X}(s)} \quad .$$

With the bounds $\left|\frac{\ddot{Y}(s)}{\Delta^2}\right| \leq a_{lat,\max}$ and $\dot{X}(s) \geq \frac{2(n-1)L\tan(\eta)}{\tan(\delta_{\max})}$ implied by $\Phi_{valid}(I)$ follows $|a_{lat}(s)| \leq a_{lat,\max}$ and $|\kappa(t)| \leq \frac{\tan(\delta_{\max})}{L}$. □

Proof of Lemma 3

Fix some time point $t \in [t_{i-1}, t_i]$ and let $s := (t - t_{i-1})\Delta^{-1}$.

We first prove $\underline{v}(s) \leq v(t) \leq \overline{v}(s)$. Define $v_X(s) := \Delta^{-1}\dot{X}(s)$ and $v_Y(s) = \Delta^{-1}\dot{Y}(s)$ and recall that $v(t) = \sqrt{v_X(s)^2 + v_Y(s)^2}$ and $v_X(s) \geq 0$. So, $\underline{v}(t) \leq v_X(s) \leq v(t)$.

In case $\eta = 0$, the constraint $|\dot{Y}(s)| \leq \dot{X}(s)\tan\eta$ implied by $\Phi_H(I)$ enforces $v_Y(s) = 0$. Hence, $v(t) = v_X(s)$ and by definition $\overline{v}(s) = v_X(s) = v(t)$.

The remaining case is $\eta > 0$. Recall (from the above proofs) that then $v_X(s) > 0$. Because $v(t) = \sqrt{v_X(s)^2 + v_Y(s)^2} > 0$ and $0 < |v_Y(s)| \leq v_X(s)\tan\eta$ we can find some (unique) $\eta' \in [-\eta, +\eta]$ such that $v_X(s) = v(t)\cos\eta'$ and $v_Y(s) = v(t)\sin\eta'$. It is

$$\begin{aligned}v(t) &= v(t)\cos(\eta') + v(t)(1 - \cos(\eta')) \\ &= v(t)\cos(\eta') + \frac{1-\cos(\eta')}{\sin(\eta')}v(t)\sin(\eta') \\ &= v_X(s) + \frac{1-\cos(\eta')}{\sin(\eta')}v_Y(s) \\ &= v_X(s) + \tan\left(\frac{\eta'}{2}\right)v_Y(s)\end{aligned}$$

and by monotony of tan on the interval $(-\pi, +\pi)$

$$\begin{aligned}&\leq v_X(s) + \tan\left(\frac{\eta}{2}\right)|v_Y(s)| \\ &= \Delta^{-1}\dot{X}(s) + \Delta^{-1}\tan\left(\frac{\eta}{2}\right)|\dot{Y}(s)| \leq \overline{v}(s)\end{aligned}$$

by definition of $\Phi_v(I)$. So, $\underline{v}(s) \leq v(t) \leq \overline{v}(s)$.

Now, we prove $\underline{acc}(s) \leq v(t) \leq \overline{acc}(s)$. The acceleration $acc(t) = \dot{v}(t)$ is invariant to rotation of the spline segment. Therefore

$$acc(t) = \frac{\ddot{X}(s)\dot{X}(s) + \ddot{Y}(s)\dot{Y}(s)}{\Delta^2\sqrt{\dot{X}(s)^2 + \dot{Y}(s)^2}} \quad .$$

If $\eta = 0$, then by constraint $\Phi_H(I)$, it is $\dot{Y}(s) = \ddot{Y}(s) = 0$ and therefore $acc(t) = \Delta^2\ddot{X}(s) = \underline{acc}(t) = \overline{acc}(t)$. If $\eta > 0$, then $\Phi_{valid}(I)$ enforces $\ddot{X}(s) = 0$. Together with $\sqrt{\dot{X}(s)^2 + \dot{Y}(s)^2} \geq |\dot{Y}(s)|$ follows $|acc(t)| \leq \Delta^{-2}|\ddot{Y}(s)|$. Then, $\underline{acc}(s) \leq v(t) \leq \overline{acc}(s)$ follows with the definition of $\Phi_{acc}(I)$.

Next, we prove $\exists k \in \mathbb{Z}.\, \underline{\mathsf{head}} \leq \theta(t) + 2k\pi \leq \overline{\mathsf{head}}$. In the case $\eta = 0$ recall from the proof of Lemma 1 that $\sin\theta(t) = \sin\hat{\theta}$, $\cos\theta(t) = \cos\hat{\theta}$, and $\hat{\theta} = \underline{\mathsf{head}} = \overline{\mathsf{head}}$. Otherwise, it follows from $\Phi_H(I)$ that $|\dot{Y}(s)| \leq \dot{X}(s)\tan(\eta)$, $X(s) > 0$ and therefore $v(t) > 0$. Recall that $0 \leq \eta < \pi/2$ and by construction $X(s) = x(t)\cos\hat{\theta} + y(t)\sin\hat{\theta} = v(t)\cos\theta(t)\cos\hat{\theta} + v(t)\sin\theta(t)\sin\hat{\theta} = v(t)\cos(\theta(t) - \hat{\theta})$ and $Y(s) = -x(t)\sin\hat{\theta} + y(t)\cos\hat{\theta} = -v(t)\sin\theta(t)\cos\hat{\theta} + v(t)\cos\theta(t)\sin\hat{\theta} = v(t)\sin(\theta(t) - \hat{\theta})$. So the constraint $\Phi_H(I)$ implies $|\sin(\theta(t) - \hat{\theta})| \leq \cos(\theta(t) - \hat{\theta})\tan\eta$. We can rewrite this as $|\tan(\theta(t) - \hat{\theta})| \leq \tan\eta$, so there exists some $k \in \mathbb{Z}$ such that $-\eta \leq \theta(t) - \hat{\theta} + 2k\pi \leq \eta$. Because $\underline{\mathsf{head}} \leq \theta - \eta$ and $\overline{\mathsf{head}} \geq \hat{\theta} + \eta$ we have $\underline{\mathsf{head}} \leq \theta(t) + 2k\pi \leq \overline{\mathsf{head}}$.

Recall that the bounding box extends $bound(\theta)$ (for $bound \in BB$) also have a period of 2π. The constraint

$\Phi_{BB}(I)$ implies $\underline{\mathsf{bound}} \leq bound(\theta') \leq \overline{\mathsf{bound}}$ for all $\theta' \in [\hat{\theta} - \eta, \hat{\theta} + \eta]$. With $\hat{\theta} - \eta \leq \theta(t) \leq \hat{\theta} + \eta$ shown above, it is $\underline{\mathsf{bound}} \leq bound(\theta(t)) \leq \overline{\mathsf{bound}}$. □

Proof of Lemma 4

We show this by structural induction over the considered spatial and non-spatial relations. The induction starts linear inequalities of the form $\sum_j a_j A_j \lhd b$ as in Construction 2 and defines

$$\tilde{A}_j(s) := \sum_{k=0}^{n} \mathcal{B}_k^n(s) \tilde{A}_{j,k}$$

with $\tilde{A}_{j,k}$ as in the construction. In the following, $A_j(t)$ denotes the value that $\sigma(t)$ assigns to A_j. Assume that $\bigwedge_{k=0}^{n} \sum_j a_j \tilde{A}_{j,k} \lhd_k b$ holds, so $\sum_j a_j \tilde{A}_{j,0} \lhd b$ and $\sum_j a_j \tilde{A}_{j,k} \leq b$ for $k > 0$. For Bernstein basis polynomials $\mathcal{B}_k^n(s)$ hold $\sum_{k=0}^{n} \mathcal{B}_k^n(s) = 1$ and $\mathcal{B}_0^n(s) > 0$ for all $0 \leq s < 1$. So, the following are equivalent for all $s \in [0, 1)$:

$$\sum_j a_j \tilde{A}_{j,0} \lhd b$$

$$\sum_j a_j \tilde{A}_{j,0} \mathcal{B}_0^n(s) \lhd b \mathcal{B}_0^n(s)$$

$$\sum_j a_j \tilde{A}_{j,0} \mathcal{B}_0^n(s) + \sum_{k=1}^{n} \sum_j a_j \tilde{A}_{j,k} \mathcal{B}_k^n(s) \lhd b \mathcal{B}_0^n(s) + \sum_{k=1}^{n} b \mathcal{B}_k^n(s)$$

$$\sum_{k=0}^{n} \sum_j a_j \tilde{A}_{j,k} \mathcal{B}_k^n(s) \lhd b$$

By Lemma 3, $a_j A_j(t) \leq a_j \tilde{A}_j(s)$ for all A_j, $t \in [t_{i-1}, t_i]$, and $s = \frac{t - t_{i-1}}{t_i - t_{i-1}}$ (recall that $\tilde{A}_j(s)$ is an upper bound for $A_j(s)$ if $a_j \geq 0$, and a lower bound if $a_j < 0$). Therefore,

$$\sigma(t) \models \sum_j a_j A_j \lhd b$$

for all $t \in [t_{i-1}, t_i)$.

For conjunctions and disjunctions of inequalities, this holds because of

$$(\forall t \in [t_{i-1}, t_i).\, \sigma(t) \models \varphi_1) \wedge (\forall t \in [t_{i-1}, t_i).\, \sigma(t) \models \varphi_2)$$

implies

$$\forall t \in [t_{i-1}, t_i).\, \sigma(t) \models (\varphi_1 \wedge \varphi_2)$$

and

$$(\forall t \in [t_{i-1}, t_i).\, \sigma(t) \models \varphi_1) \vee (\forall t \in [t_{i-1}, t_i).\, \sigma(t) \models \varphi_2)$$

implies

$$\forall t \in [t_{i-1}, t_i).\, \sigma(t) \models (\varphi_1 \vee \varphi_2) \ ,$$

which closes the induction proof. □

Proof of Lemma 5

Assume some chart BC and a family $(\mathsf{b}_i)_i$ of N variables and let $\Omega, (\phi_i)_i = \mathrm{BC2SAT}(\mathsf{BC}, (\mathsf{b}_i)_i)$. We show by induction over the structure of basic charts that, for all integers $p, q \leq N$, if

- Ω and ϕ^N hold,
- $\mathsf{s}_{sv}^i = \mathit{true}$ implies $\forall t \in [i\Delta, (i+1)\Delta).\, \sigma(t) \models \varphi_{sv}$ for all spatial views sv and $i = 0, \ldots, N-1$,
- $\mathsf{b}^i = \mathit{false}$ for all $i < p$, and
- $\mathsf{b}^i = \mathit{true}$ for all $i \geq p$,

then $\sigma, [p\Delta, q\Delta) \models \mathsf{BC}$.

The induction start is invariant nodes. We show by induction over $i = p, p+1, \ldots, N$ that $\mathsf{s}_{\mathsf{BC}}^i = \mathit{true}$ implies $\forall t \in [p\Delta, i\Delta) : \sigma(t) \models \varphi_{sv}$. This is trivially true for $i = p$. Assume we have shown this for some arbitrary but fixed $i \geq p$. Recall that Ω contains a constraint $(\mathsf{s}_{\mathsf{BC}}^{i+1} \wedge \mathsf{b}^i \rightarrow \mathsf{s}_{\mathsf{BC}}^i$ and that we assume $\mathsf{b}^i = \mathit{true}$ for $i \geq p$. Hence, $\mathsf{s}_{\mathsf{BC}}^{i+1} = \mathit{true}$ implies $\mathsf{s}_{\mathsf{BC}}^i = \mathit{true}$, so, by induction hypothesis, $\forall t \in [p\Delta, i\Delta).\, \sigma(t) \models \varphi_{sv}$. Furthermore, by assumption, it implies $\forall t \in [i\Delta, (i+1)\Delta).\, \sigma(t) \models \varphi_{sv}$. As a consequence, $\forall t \in [p\Delta, (i+1)\Delta).\, \sigma(t) \models \varphi_{sv}$. Assume that ϕ^q evaluates to *true*. By construction, $\phi^0 \equiv \mathit{false}$ so $q > 0$ and in that case $\phi^q \equiv \mathsf{s}^{q-1} \wedge \mathsf{b}^{q-1}$. By the above, this implies $\sigma(t) \models \varphi_{sv}$ for all $t \in [p\Delta, q\Delta)$ and $q > p$. Hence, $\sigma, [p\Delta, q\Delta) \models \mathsf{BC}$.

For sequences, parallel composition, and choice charts BC, assume we have shown the lemma for the subcharts A and B of BC, and for all p, q. For sequences, let $\Omega_A, (\phi_A^i)_i = \mathrm{BC2SAT}(A, (\mathsf{b}^i)_i)$ and $\Omega_B, (\phi_B^i)_i = \mathrm{BC2SAT}(B, (\mathsf{m}_{\mathsf{BC}}^i)_i)$. Find the index r with $m^r = \mathit{true}$ and $\mathsf{m}_i = \mathit{false}$ for all $i < r$ (we can show by induction that ϕ_B^q evaluates to *false* if no such r exists, which contradicts our assumptions). By construction, Ω contains constraints $\mathsf{m}_{\mathsf{BC}}^i \rightarrow \mathsf{m}_{\mathsf{BC}}^{i+1}$ and $\mathsf{m}_{\mathsf{BC}}^{i+1} \rightarrow (\mathsf{m}_{\mathsf{BC}}^i \vee \phi_A^{i+1})$ for each $i = 0, \ldots, N-1$, respectively $m^0 \rightarrow \phi_B^0$. Since $\mathsf{m}_{\mathsf{BC}}^{r-1} = \mathit{false}$ (if $r > 0$), this implies $\beta \models \phi_B^r$. Furthermore, this implies inductively that $\mathsf{m}_{\mathsf{BC}}^i = \mathit{true}$ for all $i \geq r$. So, by induction hypothesis holds $\sigma, [p\Delta, r\Delta) \models A$ and $\sigma, [r\Delta, q\Delta) \models B$.

For parallel composition and choice, let $\Omega_A, (\phi_A^i)_i = \mathrm{BC2SAT}(A, (\mathsf{b}^i)_i)$ and $\Omega_B, (\phi_B^i)_i = \mathrm{BC2SAT}(B, (\mathsf{b}^i)_i)$. For parallel composition, $\phi^q \equiv \phi_A^q \wedge \Phi_B^q$. So, by induction hypothesis, both $\sigma, [p\Delta, q\Delta) \models A$ and $\sigma, [p\Delta, q\Delta) \models B$ hold. Analogously, at least one of them holds if BC is a choice chart (here, $\phi^q \equiv \phi_A^q \wedge \Phi_B^q$). In all the cases, this implies $\sigma, [p\Delta, q\Delta) \models \mathsf{BC}$. □

Funding information Open Access funding enabled and organized by Projekt DEAL.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Babisch, S., Neurohr, C., Westhofen, L., Schoenawa, S., Liers, H.: Leveraging the GIDAS database for the criticality analysis of automated driving systems. J. Adv. Transp. (2023). https://doi.org/10.1155/2023/1349269
2. Bach, J., Otten, S., Sax, E.: Model based scenario specification for development and test of automated driving functions. In: 2016 IEEE Intelligent Vehicles Symposium (IV), pp. 1149–1155 (2016). https://doi.org/10.1109/IVS.2016.7535534
3. Becker, J.S.: Partial consistency for requirement engineering with traffic sequence charts. In: Software Engineering (Workshops), vol. 54 (2020)
4. Becker, J.S.: Safe linear encoding of vehicle dynamics for the instantiation of abstract scenarios. In: Formal Methods for Industrial Critical Systems. LNCS, vol. 14952. Springer, Berlin (2024)
5. Becker, J., Koopmann, T., Neurohr, B., Neurohr, C., Westhofen, L., Wirtz, B., Böde, E., Damm, W.: Simulation of Abstract Scenarios: Towards Automated Tooling in Criticality Analysis, pp. 42–51 (2022). https://doi.org/10.5281/zenodo.5907154
6. Bjørner, N., de Moura, L., Nachmanson, L., Wintersteiger, C.M.: Programming Z3, pp. 148–201. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17601-3_4
7. Böde, E., Büker, M., Eberle, U., Fränzle, M., Gerwinn, S., Kramer, B.: Efficient splitting of test and simulation cases for the verification of highly automated driving functions. In: Gallina, B., Skavhaug, A., Bitsch, F. (eds.) Computer Safety, Reliability, and Security, pp. 139–153. Springer, Cham (2018)
8. Böde, E., Büker, M., Damm, W., Fränzle, M., Kramer, B., Neurohr, C., Vander Maelen, S.: Identifikation und Quantifizierung von Automationsrisiken für hochautomatisierte Fahrfunktionen. Tech. Rep., Insitute for Information Technology (OFFIS) e.V. (2019) https://doi.org/10.5281/zenodo.14141022
9. Borchers, P., Koopmann, T., Westhofen, L., Becker, J.S., Putze, L., Grundt, D., de Graaff, T., Kalwa, V., Neurohr, C.: TSC2CARLA: an abstract scenario-based verification toolchain for automated driving systems. Sci. Comput. Program. **242**, 103256 (2025). https://doi.org/10.1016/j.scico.2024.103256
10. Brade, T., Kramer, B., Neurohr, C.: Paradigms in scenario-based testing for automated driving. In: 2021 International Symposium on Electrical, Electronics and Information Engineering, pp. 108–114 (2021). https://doi.org/10.1145/3459104.3459208
11. Butz, M., Heinzemann, C., Herrmann, M., Oehlerking, J., Rittel, M., Schalm, N., Ziegenbein, D.: SOCA: domain analysis for highly automated driving systems. In: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), pp. 1–6 (2020). https://doi.org/10.1109/ITSC45102.2020.9294438
12. Choi, J.W., Curry, R., Elkaim, G.: Path planning based on Bézier curve for autonomous ground vehicles. In: Advances in Electrical and Electronics Engineering-IAENG Special Edition of the World Congress on Engineering and Computer Science 2008, pp. 158–166. IEEE (2008)
13. Damm, W., Kemper, S., Möhlmann, E., Peikenkamp, T., Rakow, A.: Using traffic sequence charts for the development of HAVs. In: ERTS 2018, 9th European Congress on Embedded Real Time Software and Systems (ERTS 2018), Toulouse, France (2018)
14. Damm, W., Kemper, S., Möhlmann, E., Peikenkamp, T., Rakow, A.: Traffic sequence charts: a visual language for capturing traffic scenarios. In: Embedded Real Time Software and Systems, vol. ERTS2018 (2018)
15. Damm, W., Möhlmann, E., Peikenkamp, T., Rakow, A.: A Formal Semantics for Traffic Sequence Charts pp. 182–205. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-95246-8_11
16. de Moura, L., Bjørner, N.: Z3: an efficient SMT solver. In: Tools and Algorithms for the Construction and Analysis of Systems: 14th International Conference, TACAS 2008. Proceedings, pp. 337–340. Springer, Berlin (2008). https://doi.org/10.1007/978-3-540-78800-3_24
17. Eggers, A., Stasch, M., Teige, T., Bienmüller, T., Brockmeyer, U.: 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 (2018)
18. Foretellix Ltd.: Measurable Scenario Description Language Reference Tech. Rep., Foretellix Ltd. (2020) https://www.foretellix.com/wp-content/uploads/2020/07/M-SDL_LRM_OS.pdf. Accessed on 2024-04-19
19. Frehse, G.: PHAVer: algorithmic verification of hybrid systems past HyTech. In: International Workshop on Hybrid Systems: Computation and Control, pp. 258–273. Springer, Berlin (2005)
20. Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O.: SpaceEx: scalable verification of hybrid systems. In: International Conference on Computer Aided Verification, pp. 379–395. Springer, Berlin (2011)
21. Fremont, D.J., Kim, E., Dreossi, T., Ghosh, S., Yue, X., Sangiovanni-Vincentelli, A.L., Seshia, S.A.: Scenic: a language for scenario specification and data generation. Mach. Learn. **112**(10), 3805–3849 (2023). https://doi.org/10.1007/s10994-021-06120-5
22. Goyal, S., Griggio, A., Kimblad, J., Tonetta, S.: Automatic Generation of Scenarios for System-level Simulation-based Verification of Autonomous Driving Systems (2023). arXiv preprint arXiv:2311.09784
23. Grundt, D., Köhne, A., Saxena, I., Stemmer, R., Westphal, B., Möhlmann, E.: Towards runtime monitoring of complex system requirements for autonomous driving functions. In: Fourth International Workshop on Formal Methods for Autonomous Systems (FMAS) (2022). https://doi.org/10.48550/arXiv.2209.14032
24. Han, L., Yashiro, H., Nejad, H.T.N., Do, Q.H., Mita, S.: Bezier curve based path planning for autonomous vehicle in urban environment. In: 2010 IEEE Intelligent Vehicles Symposium, pp. 1036–1042. IEEE (2010)
25. Hartjen, L., Philipp, R., Schuldt, F., Friedrich, B., Howar, F.: Classification of driving maneuvers in urban traffic for parametrization of test scenarios. In: 9. Tagung Automatisiertes Fahren (2019)
26. International Organization for Standardization: ISO 21448: Road vehicles – Safety of the intended functionality (2022)
27. ISO 26262: Road vehicles – functional safety (2018)
28. Klischat, M., Althoff, M.: Synthesizing traffic scenarios from formal specifications for testing automated vehicles. In: 2020 IEEE Intelligent Vehicles Symposium (IV), pp. 2065–2072. IEEE (2020)
29. Kong, J., Pfeiffer, M., Schildbach, G., Borrelli, F.: Kinematic and dynamic vehicle models for autonomous driving control design pp. 1094–1099 (2015). https://doi.org/10.1109/IVS.2015.7225830
30. Koopmann, T., Putze, L., Westhofen, L., Gansch, R., Adee, A., Neurohr, C.: Grasping causality for the explanation of criticality for automated driving. IEEE Access **13**, 54739–54756 (2025). https://doi.org/10.1109/ACCESS.2025.3555177

31. Kramer, B., Neurohr, C., Büker, M., Böde, E., Fränzle, M., Damm, W.: Identification and quantification of hazardous scenarios for automated driving. In: Zeller, M., Höfig, K. (eds.) Model-Based Safety and Assessment, pp. 163–178. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58920-2_11
32. Ma, J., Che, X., Li, Y., Lai, E.M.K.: Traffic scenarios for automated vehicle testing: a review of description languages and systems. Machines **9**(12), 342 (2021)
33. Menzel, T., Bagschik, G., Maurer, M.: Scenarios for development, test and validation of automated vehicles. In: 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 1821–1827. IEEE (2018). https://doi.org/10.1109/IVS.2018.8500406
34. Menzel, T., Bagschik, G., Isensee, L., Schomburg, A., Maurer, M.: From functional to logical scenarios: detailing a keyword-based scenario description for execution in a simulation environment. In: 2019 IEEE Intelligent Vehicles Symposium (IV), pp. 2383–2390 (2019). https://doi.org/10.1109/IVS.2019.8814099
35. Neurohr, C., Westhofen, L., Henning, T., de Graaff, T., Mohlmann, E., Bode, E.: Fundamental considerations around scenario-based testing for automated driving. In: 2020 IEEE Intelligent Vehicles Symposium (IV), pp. 121–127. IEEE (2020). https://doi.org/10.1109/IV47402.2020.9304823
36. Neurohr, C., Westhofen, L., Butz, M., Bollmann, M.H., Eberle, U., Galbas, R.: Criticality analysis for the verification and validation of automated vehicles. IEEE Access **9**, 18016–18041 (2021). https://doi.org/10.1109/ACCESS.2021.3053159
37. Neurohr, C., Westhofen, L., Butz, M., Bollmann, M.H., Putze, L., Koopmann, T., Gansch, R., Knoop, M., Rasch, A., Cojocaru, B., Daube, J.: Advances on the Criticality Analysis for Automated Driving Systems (2024). https://doi.org/10.5281/zenodo.10815308
38. Neurohr, C., Westhofen, L., Duong Quang, T., Koopmann, T., Gansch, R., Schoenawa, S.: Requirements on Simulation Systems and Models Posed by a Criticality Analysis for Automated Driving Systems (2024). https://doi.org/10.5281/zenodo.10815530
39. Phan, Q.S., Malacaria, P.: All-solution satisfiability modulo theories: applications, algorithms and benchmarks. In: 2015 10th International Conference on Availability, Reliability and Security, pp. 100–109 (2015). https://doi.org/10.1109/ARES.2015.14
40. Plaku, E., Kavraki, L.E., Vardi, M.Y.: Hybrid systems: from verification to falsification. In: International Conference on Computer Aided Verification, pp. 463–476. Springer, Berlin (2007)
41. Plaku, E., Kavraki, L.E., Vardi, M.Y.: Falsification of LTL safety properties in hybrid systems. Int. J. Softw. Tools Technol. Transf. **15**(4), 305–320 (2013)
42. Prautzsch, H., Boehm, W., Paluszny, M.: Bézier and B-Spline Techniques. Springer, Berlin (2013)
43. Qian, X., Navarro, I., de La Fortelle, A., Moutarde, F.: Motion planning for urban autonomous driving using Bézier curves and MPC. In: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), pp. 826–833. IEEE (2016)
44. Rauschert, A., Amid, G.: ASAM OpenSCENARIO DSL 2.1.0: Release presentation (2024). Presentation https://www.asam.net/standards/detail/openscenario-dsl/. Accessed on 2024-05-08
45. Rauschert, A., Amid, G.: ASAM OpenSCENARIO XML 1.3.0: Release presentation (2024). Presentation https://www.asam.net/standards/detail/openscenario-xml/. Accessed on 2024-05-08
46. Reisgys, F., Plaum, J., Schwarzhaupt, A., Sax, E.: Scenario-based X-in-the-loop test for development of driving automation. In: 14. Workshop Fahrerassistenzsysteme und Automatisiertes Fahren. Uni-DAS e.V, pp. 207–216 (2022)
47. SAE International: J3016: Taxonomy and Definitions for Terms Related to Driving Automation Systems for on-Road Motor Vehicles (2021)
48. Scheibler, K., Eggers, A., Teige, T., Walz, M., Bienmüller, T., Brockmeyer, U.: Solving constraint systems from traffic scenarios for the validation of autonomous driving. In: Proceedings of the 4th SC-Square Workshop, pp. 2–12 (2019)
49. Schramm, D., Hiller, M., Bardini, R.: Single track models. In: Vehicle Dynamics, pp. 223–253. Springer, Berlin (2014)
50. Schuldt, F., Reschka, A., Maurer, M.: A Method for an Efficient, Systematic Test Case Generation for Advanced Driver Assistance Systems in Virtual Environments pp. 147–175. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-61607-0_7
51. Schwammberger, M.: An abstract model for proving safety of autonomous urban traffic. Theor. Comput. Sci. **744**, 143–169 (2018)
52. Ulbrich, S., Menzel, T., Reschka, A., Schuldt, F., Maurer, M.: Defining and substantiating the terms scene, situation, and scenario for automated driving. In: 2015 IEEE 18th International Conference on Intelligent Transportation Systems, pp. 982–988. IEEE (2015)
53. Wachenfeld, W., Winner, H.: The Release of Autonomous Vehicles pp. 425–449. Springer, Berlin (2016). https://doi.org/10.1007/978-3-662-48847-8_21
54. Weissensteiner, P., Stettinger, G.: Explainable Safety Argumentation for the Deployment of Automated Vehicles. Electronics **13**(23) (2024). https://doi.org/10.3390/electronics13234606
55. Westhofen, L., Neurohr, C., Butz, M., Scholtes, M., Schuldes, M.: Using ontologies for the formalization and recognition of criticality for automated driving. IEEE Open J. Intell. Transp. Syst. **3**, 519–538 (2022). https://doi.org/10.1109/OJITS.2022.3187247
56. Westhofen, L., Neurohr, C., Koopmann, T., Butz, M., Schütt, B., Utesch, F., Neurohr, B., Gutenkunst, C., Böde, E.: Criticality metrics for automated driving: a review and suitability analysis of the state of the art. Arch. Comput. Methods Eng. **30**(1), 1–35 (2023)
57. Zhang, X., Khastgir, S., Jennings, P.: Scenario description language for automated driving systems: a two level abstraction approach. In: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 973–980. IEEE (2020)

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.