This article has been accepted for publication in IEEE Transactions on Intelligent Vehicles. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIV.2023.3289580

Specification-Compliant Driving Corridors for Motion Planning of Automated Vehicles

Edmond Irani Liu and Matthias Althoff

***Abstract*—It is crucial for automated vehicles to explicitly comply with specifications, including traffic rules, to ensure their safe and effective participation in road traffic. Such compliance is also essential for vehicle manufacturers to avoid liability claims in the event of accidents. We propose a novel approach addressing the problem of specification-compliant motion planning for automated vehicles. Our approach couples set-based reachability analysis with automata-based model checking and outputs specification-compliant driving corridors. These driving corridors serve as motion planning constraints and expedite the generation of trajectories complying with specifications expressed in metric temporal logic. In contrast to existing works, our approach efficiently and exhaustively verifies all driving corridors of an automated vehicle, leveraging mature model checking techniques. We demonstrate the applicability, effectiveness, and efficiency of our approach using various specifications on scenarios from the CommonRoad benchmark suite. Moreover, we benchmark the performance of our prototype against multiple scenarios, indicating that our approach is real-time capable.**

***Index Terms*—automated vehicles, motion planning, traffic rules, temporal logic, reachability analysis, model checking.**

I. Introduction

AUTOMATED vehicles are expected to explicitly comply with traffic rules to safely and effectively participate in mixed road traffic, where both automated and human-driven vehicles coexist. In addition, automated vehicle manufacturers bear the responsibility to certify such compliance and by this avoid liability claims in the event of accidents. Despite the importance of this matter, most previous studies on motion planning of automated vehicles reported in recent surveys [1]–[4] either entirely disregard traffic rules or only consider a limited fraction of them. This is due to the sheer difficulty of formalizing traffic rules in a machine-interpretable way and their integration into motion planners. In this article, the term *specifications* refers to traffic rules and other requirements formalized in temporal logic to which vehicles must adhere. Examples of such formalizations can be found in [5]–[9].

Generating drivable trajectories for vehicles complying with specifications involves reasoning with both their continuous and discrete states. The former typically contains the position, velocity, and orientation of a vehicle; examples of the latter are the operation mode of the vehicle and its logical relation to other traffic participants. Computational challenges arise in generating such trajectories due to factors such as vehicle dynamics, considered specifications (including collision avoidance), and the interdependence of planned trajectories and constraints originating from the specifications [10]. Per recent surveys [1]–[4], no approach exists that plans specification-compliant motions in continuous state space: Classical motion planners generate collision-free and dynamically feasible trajectories but cannot guarantee specification compliance; Also, planning in a discretized state space may output discrete plans that satisfy the specifications but disregard drivability constraints or lead to collisions.

We propose a novel and efficient approach addressing the problem of specification-compliant motion planning for automated vehicles using set-based reachability analysis and automata-based model checking. *Reachability analysis* is a technique for determining the set of states reachable by a system over time (henceforth referred to as *reachable set*), starting from a set of initial states. Computing the reachable sets of a vehicle in an over-approximative fashion enables the exploration of its continuous state space and the identification of all its collision-free *driving corridors* [11]–[13]. A driving corridor represents a timed sequence of position and velocity bounds that can be utilized by motion planners to significantly reduce the planning space, especially in situations with a narrow solution space [12]–[14]. *Model checking* is a formal verification technique that verifies desired behavioral specifications on a suitable model of a given system through systematic inspection of all states of the model. By coupling reachability analysis with model checking, we efficiently identify all driving corridors of automated vehicles that are both collision-free and compliant with enforced specifications. Applying constraints extracted from such driving corridors to motion planners expedites the generation of trajectories complying with enforced specifications.

A. Related Work

We categorize existing works on specification-compliant motion planning based on *when* specifications are considered:

1) Considering Compliance After Motion Planning: Runtime verification, also known as *monitoring*, refers to checking whether an execution of a system meets the expected behaviors. For instance, a monitor for examining the compliance of vehicles with safe distance rules and overtaking rules is presented in [9]. While the monitoring is often efficient, monitors typically only return a *robustness degree* (the extent of satisfaction of specifications) or a verdict (*true* or *false*) on whether the specifications have been satisfied. No alternative trajectory is returned if a trajectory is deemed inferior or

Both authors are with the School of Computation, Information and Technology, Technical University of Munich, 80333 Munich, Germany (e-mail: edmond.irani@tum.de, althoff@tum.de).

rejected by a monitor. This generally leads to replanning and verifying many trajectories for more complex specifications before finding a specification-compliant solution.

Instead of examining individual trajectories, it is also possible to verify infinitely many trajectories at once: The work in [15] describes a method for model checking reachable sets of continuous and hybrid systems against *signal temporal logic* [16] specifications. As with monitoring, it only returns a verdict (possibly with counterexamples in case of violation), which has limited usage for our motion planning application.

2) Considering Compliance During Motion Planning: Existing efforts in this category can be roughly divided into three groups: multilayered approaches, approaches based on *mixed-integer linear programming* (MILP), and approaches based on *rapidly exploring random trees* (RRTs) [17]. Multilayered approaches [18]–[27] commonly handle specification-compliant motion planning problems using a high-level discrete planning layer and a low-level trajectory planning layer. The discrete planning layer relies on discrete abstractions of the system of interest and generates plans satisfying the specifications, which guide the trajectory planning process at a later stage. The discrete plans are generated based on, among others, automata theory [20], [22]–[25], [27], satisfiability modulo theory [18], [21], and monitors [19]. For instance, article [24] adopts timed automata to generate timed paths that satisfy *metric temporal logic* (MTL) [28] specifications for indoor robot navigation; the work in [21] introduces a satisfiability modulo convex programming framework that handles both convex constraints over continuous states and Boolean constraints over discrete states for cyber-physical systems; in [19], the authors obtain high-level driving maneuvers of automated vehicles that respect traffic rules in *linear temporal logic* (LTL) [29] via monitoring. In most cases, the dynamic constraints of the system are not considered in the discrete plans; thus, the drivability of the plans is often not ensured. Consequently, frequent replanning in both the discrete and trajectory planning layers can be expected, especially in complex and highly dynamic environments.

The basic idea of MILP-based approaches is to cast temporal logic specifications as mixed-integer linear constraints. After introducing system dynamic constraints, a solver generates a specification-compliant trajectory while optimizing certain cost functions. MILP problems are NP-hard in nature [30, Ch. 11], and the constraints mentioned above bring about auxiliary decision variables that exponentially increase the complexity and solution time of the optimization problem (e.g., see [31]–[35]). This is often a limiting factor for applications with high real-time requirements such as motion planning of automated vehicles.

RRT-based approaches typically generate specification-compliant trajectories in an incremental manner. The works in [22], [36]–[40] build on the RRT* algorithm [41], which is an asymptotically optimal variant of the well-known RRT algorithm. The growth of the tree is steered or pruned, e.g., using automata [22], [36], [38], [40] or robustness degrees [37], [39] of the specifications. Given enough time and iterations, a trajectory respecting the system dynamics and specifications can be found. While RRT-based methods provide fast solutions to specific problems, they are not well-suited for safety-critical applications due to their inherent characteristic known as *probabilistic completeness* [17], [41]. Moreover, the performance of RRT-based methods typically degrades in situations with a narrow solution space [42].

B. Contributions

Our approach provides the following contributions:

- Extension of [43] by integrating temporal logic specifications (including interstate and intersection traffic rules) into the reachability analysis of automated vehicles.
- Coupling reachability analysis with model checking for identifying collision-free and specification-compliant driving corridors. Such corridors expedite the generation of specification-compliant trajectories for motion planners that accept position and velocity constraints.
- Generation of a *product graph* from which the optimal driving corridor can be determined using arbitrary utility functions in a separate stage.

Our approach has the following properties:

- In contrast to conventional motion planners, our reachable set computation requires less time in more critical scenarios: The computation can be performed the faster, the smaller the solution space is, which is often the case in critical scenarios.
- Efficient and exhaustive verification of all driving corridors of an automated vehicle against considered specifications, owing to mature model checking techniques.
- Detection of conflicting or non-satisfiable specifications before motion planning.
- Applicability in traffic scenarios involving static and dynamic obstacles of arbitrary shapes.
- The total computation time requires only a fraction of the planning horizon.

The remainder of this article is organized as follows: After presenting the preliminaries and problem statement in Sec. II, we describe our methodology in Sec. III. The implementation of our reachability analysis is detailed in Sec. IV, followed by the evaluation of predicates and the rewriting of specifications in Sec. V and Sec. VI, respectively. In Sec. VII, we elaborate on the identification of specification-compliant driving corridors. Our approach is evaluated in Sec. VIII and we finish with conclusions in Sec. IX.

II. Preliminaries and Problem Statement

After introducing the necessary preliminaries, including the general setup, temporal logics to formalize our specifications, set-based reachability analysis, automata-based model checking, and driving corridors, we present the problem statement.

A. General Setup

The vehicle for which trajectories should be planned is referred to as the *ego vehicle*. The road network consists of lanelets [44], each modeled with polylines representing its left and right boundaries. We assume a high-level route planner is available that plans a route through the road network, whose centerline is considered as the reference path $\Gamma : \mathbb{R} \to \mathbb{R}^2$. A local curvilinear coordinate system $F^{\mathtt{L}}$ of the ego vehicle is constructed from the reference path as described in [44], within which (s, d) describes the longitudinal coordinate s along the reference path and the lateral coordinate d orthogonal to $\Gamma(s)$. The adoption of $F^{\mathtt{L}}$ facilitates the formulation of maneuvers from the perspective of the ego vehicle, such as following a lane and stopping before a stop line. We denote by $k \in \mathbb{N}_0$ a step corresponding to time $t_k = k\Delta_t$, with $\Delta_t \in \mathbb{R}_+$ being a predefined time increment. Motions of the ego vehicle are planned up to the planning horizon $k_h \in \mathbb{N}$, whose dynamics is

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k), \tag{1}$$

where $\boldsymbol{x}_k \in \mathcal{X}_k \subset \mathbb{R}^{n_{\boldsymbol{x}}}$ represents the state of the ego vehicle in the state space $\mathcal{X}_k$, $\boldsymbol{u}_k \in \mathcal{U}_k \subset \mathbb{R}^{n_{\boldsymbol{u}}}$ represents an input in the input space $\mathcal{U}_k$. A possible input trajectory over time is denoted by $\boldsymbol{U}$. We also denote by τ_k the valuation of the ego vehicle with state $\boldsymbol{x}_k$ over atomic propositions $\mathcal{AP}$ (see Sec. II-B), each of which indicates a logical relation between the ego vehicle and entities in an environment model such as lanes and obstacles (later detailed in Sec. V).

B. Temporal Logics

Specifications considered in this work are expressed in *MTL with past over finite traces* (MTLp$_f$) [28], [45]. MTLp$_f$ shares the same syntax with MTL and is interpreted over traces of *finite* length. We settle on MTLp$_f$ since (a) it is expressive enough to formulate traffic rules with timing constraints, e.g., see [5]–[7], and (b) traces in our system have finite length.

1) Metric Temporal Logic with Past over Finite Traces: An MTLp$_f$ formula $\varphi^{\mathtt{M}}$ over atomic propositions $\mathcal{AP}$ has the following syntax given in Backus-Naur form [28], [45]:

$$\varphi^{\mathtt{M}} ::= \sigma \,|\, \neg\varphi^{\mathtt{M}} \,|\, \varphi_1^{\mathtt{M}} \wedge \varphi_2^{\mathtt{M}},\, |\, \mathbf{X}_I \varphi^{\mathtt{M}} \,|\, \varphi_1^{\mathtt{M}} \mathbf{U}_I \varphi_2^{\mathtt{M}} \,|\, \mathbf{Y}_I \varphi^{\mathtt{M}} \,|\, \varphi_1^{\mathtt{M}} \mathbf{S}_I \varphi_2^{\mathtt{M}},$$

where $\sigma \in \mathcal{AP}$ is an atomic proposition, $\neg$ (Not) and $\wedge$ (And) are Boolean connectives, **X** (ne**X**t) and **U** (**U**ntil) are future-time connectives, **Y** (**Y**esterday) and **S** (**S**ince) are past-time connectives, and $I = [a, b]$ is a bounded interval. Without loss of generality, we assume $a, b \in \mathbb{N}_0$. We also use the following common abbreviations [28]:

- Contradiction: $\bot \equiv \varphi^{\mathtt{M}} \wedge \neg\varphi^{\mathtt{M}}$,
- Tautology: $\top \equiv \neg\bot$,
- Or: $\varphi_1^{\mathtt{M}} \vee \varphi_2^{\mathtt{M}} \equiv \neg(\neg\varphi_1^{\mathtt{M}} \wedge \neg\varphi_2^{\mathtt{M}})$,
- Implication: $\varphi_1^{\mathtt{M}} \Rightarrow \varphi_2^{\mathtt{M}} \equiv \neg\varphi_1^{\mathtt{M}} \vee \varphi_2^{\mathtt{M}}$,
- **F**uture: $\mathbf{F}_I \varphi^{\mathtt{M}} \equiv \top \mathbf{U}_I \varphi^{\mathtt{M}}$,
- **G**lobally: $\mathbf{G}_I \varphi^{\mathtt{M}} \equiv \neg\mathbf{F}_I \neg\varphi^{\mathtt{M}}$,
- **O**nce: $\mathbf{O}_I \varphi^{\mathtt{M}} \equiv \top \mathbf{S}_I \varphi^{\mathtt{M}}$,
- **H**istorically: $\mathbf{H}_I \varphi^{\mathtt{M}} \equiv \neg\mathbf{O}_I \neg\varphi^{\mathtt{M}}$.

MTLp$_f$ over the *point-wise* semantics [46] is interpreted over timed traces, which can be thought of as sequences of events with timestamps. Given is a trace $\tau := (\tau^0, \ldots, \tau^k, \ldots)$ with length $|\tau|$, where $\tau^k : \mathcal{AP} \to \{\texttt{true}, \texttt{false}\}$ denotes a valuation over $\mathcal{AP}$, i.e., an assignment of `true` or `false` to every atomic proposition $\sigma \in \mathcal{AP}$, at step k. The notation $(\tau, k) \models \varphi^{\mathtt{M}}$ indicates that $\varphi^{\mathtt{M}}$ holds in the k-th valuation of τ, i.e., τ^k. We simplify the semantics of MTLp$_f$ in [28], [45] since valuations τ^k are synchronized with steps k:

- $(\tau, k) \models \sigma$ if and only if (iff) $\tau^k(\sigma) = \texttt{true}$,
- $(\tau, k) \models \neg\varphi^{\mathtt{M}}$ iff $(\tau, k) \not\models \varphi^{\mathtt{M}}$,
- $(\tau, k) \models \varphi_1^{\mathtt{M}} \wedge \varphi_2^{\mathtt{M}}$ iff $(\tau, k) \models \varphi_1^{\mathtt{M}}$ and $(\tau, k) \models \varphi_2^{\mathtt{M}}$,
- $(\tau, k) \models \mathbf{X}_I \varphi^{\mathtt{M}}$ iff $k < |\tau| - 1$, $1 \in I$, and $(\tau, k+1) \models \varphi^{\mathtt{M}}$,
- $(\tau, k) \models \mathbf{Y}_I \varphi^{\mathtt{M}}$ iff $k > 0$, $1 \in I$, and $(\tau, k-1) \models \varphi^{\mathtt{M}}$,
- $(\tau, k) \models \varphi_1^{\mathtt{M}} \mathbf{U}_I \varphi_2^{\mathtt{M}}$ iff $\exists l$, $k \leq l \leq |\tau| - 1$: $(\tau, l) \models \varphi_2^{\mathtt{M}}$, $l - k \in I$, and $\forall m$, $k \leq m < l$: $(\tau, m) \models \varphi_1^{\mathtt{M}}$,
- $(\tau, k) \models \varphi_1^{\mathtt{M}} \mathbf{S} \varphi_2^{\mathtt{M}}$ iff $\exists l$, $0 \leq l \leq k$: $(\tau, l) \models \varphi_2^{\mathtt{M}}$, $k - l \in I$, and $\forall m$, $l < m \leq k$: $(\tau, m) \models \varphi_1^{\mathtt{M}}$.

As examples, formulas $\mathbf{X}_{[2,3]} \varphi^{\mathtt{M}}$ and $\varphi_1^{\mathtt{M}} \mathbf{U}_{[1,4]} \varphi_2^{\mathtt{M}}$ can be respectively read as "next valuation occurs within 2 and 3 steps (from now), in which $\varphi^{\mathtt{M}}$ holds" and "within 1 and 4 steps, a valuation occurs in which $\varphi_2^{\mathtt{M}}$ holds, and $\varphi_1^{\mathtt{M}}$ holds for all valuations before that". The past-time connectives **Y**, **S**, **O**, and **H** mirror their future-time counterparts **X**, **U**, **F**, and **G**, respectively, backward in time.

2) Linear Temporal Logic (with Past over Finite Traces): Since τ^k are synchronized with k, we do not require the full expressiveness of MTLp$_f$ for model checking our system. We interpret MTLp$_f$ formulas as *LTL with past over finite traces* (LTLp$_f$) [47] and further convert them into LTL over *infinite* traces for model checking. This reduces the complexity of model checking from EXPSPACE-complete for MTLp$_f$ [48] to PSPACE-complete for LTL [49]. Moreover, this allows us to employ mature and efficient LTL model checkers such as Spot [50]. MTLp$_f$ is syntactically reduced to LTLp$_f$ by dropping intervals I over the temporal connectives [47]; further dropping past-time connectives results in standard LTL [29]. We respectively denote by $\varphi^{\mathtt{L}}$, φ, and $\mathcal{F}$ an LTLp$_f$ formula, an LTL formula, and the set of formulas converted into LTL.

C. Set-based Reachability Analysis

Next, we define one-step reachable sets and drivable areas of the ego vehicle.

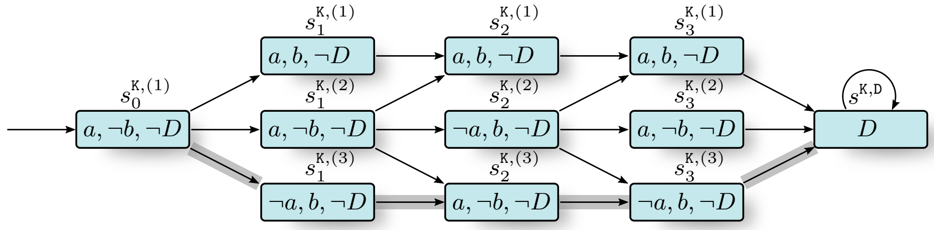
Definition 1 (Occupancy)**.** *The operator* $\operatorname{occ}(\cdot)$ *returns the occupied positions within* $F^{\mathtt{L}}$. *For example,* $\operatorname{occ}(\boldsymbol{x}_k)$ *returns the occupancy of the ego vehicle with state* $\boldsymbol{x}_k$.

Definition 2 (Set of forbidden states)**.** *Let* $\mathcal{O}_k \subset \mathbb{R}^2$ *be the set of positions occupied by all obstacles at step* k *and the space outside the road. The set of forbidden states of the ego vehicle at step* k *is defined as*

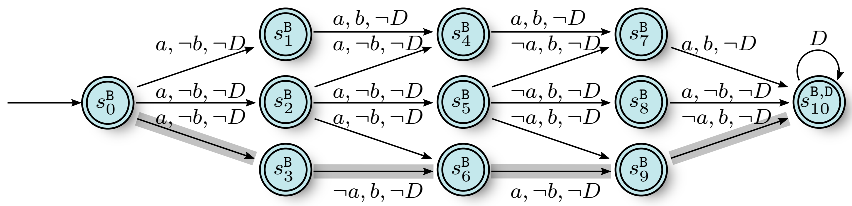
$$\mathcal{X}_k^{\mathtt{F}} := \left\{ \boldsymbol{x}_k \in \mathcal{X}_k \,\middle|\, \operatorname{occ}(\boldsymbol{x}_k) \cap \mathcal{O}_k \neq \varnothing \right\}. \tag{2}$$

Definition 3 (One-Step Reachable Set)**.** *Let* $\mathcal{R}_0^{\mathtt{e}} = \mathcal{X}_0$ *be the exact reachable set of the ego vehicle at the initial step, with* $\mathcal{X}_0$ *being the set of collision-free initial states including measurement uncertainties. The exact reachable set* $\mathcal{R}_{k+1}^{\mathtt{e}}$ *is the set of states reachable from* $\mathcal{R}_k^{\mathtt{e}}$ *without intersecting the set of forbidden states* $\mathcal{X}_{k+1}^{\mathtt{F}}$*, denoted by* $\operatorname{reach}(\mathcal{R}_k^{\mathtt{e}})$*:*

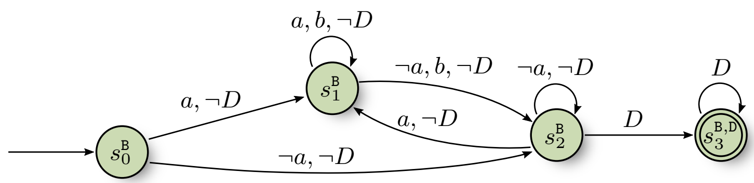
$$\mathcal{R}_{k+1}^{\mathtt{e}} := \underbrace{\Big\{ \boldsymbol{x}_{k+1} \in \mathcal{X}_{k+1} \Big| \exists \boldsymbol{x}_k \in \mathcal{R}_k^{\mathtt{e}}, \exists \boldsymbol{u}_k \in \mathcal{U}_k : \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k), \boldsymbol{x}_{k+1} \notin \mathcal{X}_{k+1}^{\mathtt{F}} \Big\}}_{\operatorname{reach}(\mathcal{R}_k^{\mathtt{e}})}. \tag{3}$$



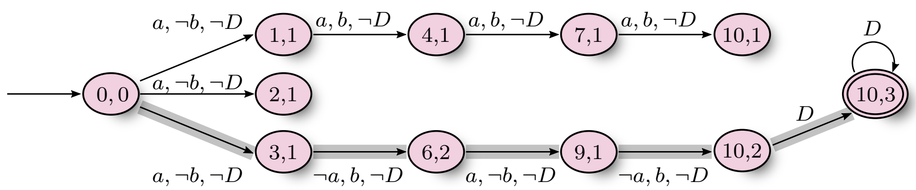
(a)



(b)



(c)



(d)

Fig. 1: Minimal example of automata-based model checking. The colors correspond to components in Fig. 3. (a) A Kripke structure $G^{\mathtt{K}}$ with $\mathcal{AP} = \{a, b, D\}$, $\mathcal{S}^{\mathtt{K}} = \{s_0^{\mathtt{K},(1)}, \ldots, s_3^{\mathtt{K},(3)}, s^{\mathtt{K},\mathtt{D}}\}$, and $\mathcal{S}_0^{\mathtt{K}} = \{s_0^{\mathtt{K},(1)}\}$. For clarity, we show all atomic propositions regardless of their truth values. (b) Automaton $A^{\mathtt{M}}$ converted from graph $G^{\mathtt{K}}$ in (a), with $\mathcal{S}^{\mathtt{B}} = \mathcal{S}^{\mathtt{BA}} = \{s_0^{\mathtt{B}}, \ldots, s_9^{\mathtt{B}}, s_{10}^{\mathtt{B},\mathtt{D}}\}$. (c) Automaton A^{φ} converted from $\varphi := \mathbf{G}(a \Rightarrow \mathbf{X}(b))$: whenever a holds, b should hold in the next valuation. (d) Product automaton $A^{\mathtt{P}} = A^{\mathtt{M}} \otimes A^{\varphi}$. For brevity, we only show the subscripts of the states in $A^{\mathtt{P}}$. For example, the state 9,1 in $A^{\mathtt{P}}$ references $s_9^{\mathtt{B}}$ in $A^{\mathtt{M}}$ and $s_1^{\mathtt{B}}$ in A^{φ}. Transitions along the only accepting run in $A^{\mathtt{P}}$ are colored gray. For comparison, we also show other transitions that do not lead to an accepting state. The accepting run in $A^{\mathtt{P}}$ corresponds to the run $s_0^{\mathtt{B}} \to s_3^{\mathtt{B}} \to s_6^{\mathtt{B}} \to s_9^{\mathtt{B}} \to s_{10}^{\mathtt{B},\mathtt{D}} \to s_{10}^{\mathtt{B},\mathtt{D}} \to \ldots$ in $A^{\mathtt{M}}$ and the execution $s_0^{\mathtt{K},(1)} \to s_1^{\mathtt{K},(3)} \to s_2^{\mathtt{K},(3)} \to s_3^{\mathtt{K},(3)} \to s^{\mathtt{K},\mathtt{D}} \to s^{\mathtt{K},\mathtt{D}} \to \ldots$ in $G^{\mathtt{K}}$ (both colored gray). Please note that an auxiliary atomic proposition D (***D**ead*) and an auxiliary self-looping state $s^{\mathtt{K},\mathtt{D}}$ (respectively $s^{\mathtt{B},\mathtt{D}}$) are required for extending the traces in $G^{\mathtt{K}}$ (respectively $A^{\mathtt{M}}$ and A^{φ}) to infinite length (see Sec. VI-3).

Definition 4 (Projection)**.** *The operator* $\mathrm{proj}_{\Diamond}(\boldsymbol{x})$ *maps the state* $\boldsymbol{x} \in \mathcal{X}$ *to its components* $\Diamond$. *For example,* $\mathrm{proj}_{(s,\dot{s})}(\boldsymbol{x}) = (s, \dot{s})^{\mathsf{T}}$ *for* $\boldsymbol{x} = (s, \dot{s}, \ddot{s})^{\mathsf{T}}$. *A set can be projected using the same operator:* $\mathrm{proj}_{\Diamond}(\mathcal{X}) = \{\mathrm{proj}_{\Diamond}(\boldsymbol{x}) \,|\, \boldsymbol{x} \in \mathcal{X}\}$.

Definition 5 (Drivable Area)**.** *The drivable area* $\mathcal{D}_k^{\mathtt{e}}$ *of the ego vehicle at step* k *is the projection of its reachable set* $\mathcal{R}_k^{\mathtt{e}}$ *onto the position domain:* $\mathcal{D}_k^{\mathtt{e}} := \mathrm{proj}_{(s,d)}(\mathcal{R}_k^{\mathtt{e}})$.

In practice, $\mathcal{X}_k^{\mathtt{F}}$ can be of arbitrary shape and the computation of $\mathcal{R}_k^{\mathtt{e}}$ as well as $\mathcal{D}_k^{\mathtt{e}}$ is generally difficult or even impossible [51]. Therefore, we compute their over-approximations $\mathcal{R}_k$ and $\mathcal{D}_k$, which will be detailed in Sec. IV.

D. Automata-based Model Checking

As motivated in Sec. I, we leverage model checking to efficiently and exhaustively identify all collision-free and specification-compliant driving corridors within the reachable sets of the ego vehicle. Let $A^{\mathtt{M}}$ be a finite state automaton representing a system M. To verify whether all possible executions of M satisfy a given LTL formula φ, denoted by $M \models \varphi$, the basic idea of automata-based model checking is to find a run in $A^{\mathtt{M}}$ that satisfies the negated formula $\neg\varphi$. If such a run does not exist, it can be concluded that $M \models \varphi$. Instead of examining whether $M \models \varphi$, model checking can alternatively be formulated to find the subset of runs in $A^{\mathtt{M}}$ that satisfy φ [52]. We follow the latter formulation since we aim to identify specification-compliant driving corridors rather than verifying whether all driving corridors satisfy the enforced specifications. We introduce two required definitions.

Definition 6 (Nondeterministic Büchi Automaton [53])**.** *A five-tuple* $(\Sigma, \mathcal{S}^{\mathtt{B}}, s_0^{\mathtt{B}}, \mathrm{trans}^{\mathtt{B}}, \mathcal{S}^{\mathtt{BA}})$ *defines a nondeterministic Büchi automaton, where*

- $\Sigma := \mathbb{P}(\mathcal{AP})$[1] *is an alphabet with letters* $\lambda \in \Sigma$,
- $\mathcal{S}^{\mathtt{B}}$ *is a set of states with elements* $s^{\mathtt{B}}$,

[1]The operator $\mathbb{P}(\cdot)$ returns the power set of the input.

- $s^{\mathtt{B}}_0 \in \mathcal{S}^{\mathtt{B}}$ *is the initial state,*
- $\mathrm{trans}^{\mathtt{B}} : \mathcal{S}^{\mathtt{B}} \times \Sigma \rightarrow \mathbb{P}(\mathcal{S}^{\mathtt{B}})$ *is a transition relation,*
- $\mathcal{S}^{\mathtt{BA}} \subseteq \mathcal{S}^{\mathtt{B}}$ *is a set of accepting states.*

A nondeterministic Büchi automaton is an finite state automaton accepting inputs of *infinite* length.

Definition 7 (Product Automaton [52])**.** *Given nondeterministic Büchi automata $A_m = (\Sigma, \mathcal{S}^{\mathtt{B}}_m, s^{\mathtt{B}}_{0,m}, \mathrm{trans}^{\mathtt{B}}_m, \mathcal{S}^{\mathtt{BA}}_m)$, $m \in \{1, 2\}$, their synchronous product is $A = A_1 \otimes A_2 := (\Sigma, \mathcal{S}^{\mathtt{P}}, s^{\mathtt{P}}_0, \mathrm{trans}^{\mathtt{P}}, \mathcal{S}^{\mathtt{PA}})$, where*

- $\mathcal{S}^{\mathtt{P}} = \mathcal{S}^{\mathtt{B}}_1 \times \mathcal{S}^{\mathtt{B}}_2$ *is the set of states with elements $(s^{\mathtt{B}}_1, s^{\mathtt{B}}_2)$,*
- $s^{\mathtt{P}}_0 = (s^{\mathtt{B}}_{0,1}, s^{\mathtt{B}}_{0,2}), s^{\mathtt{P}}_0 \in \mathcal{S}^{\mathtt{P}}$ *is the initial state,*
- $\mathrm{trans}^{\mathtt{P}} : \mathcal{S}^{\mathtt{P}} \times \Sigma \rightarrow \mathbb{P}(\mathcal{S}^{\mathtt{P}})$ *is a transition relation such that $(\tilde{s}^{\mathtt{B}}_1, \tilde{s}^{\mathtt{B}}_2) \in \mathrm{trans}^{\mathtt{P}}((s^{\mathtt{B}}_1, s^{\mathtt{B}}_2), \lambda)$ iff $\tilde{s}^{\mathtt{B}}_1 \in \mathrm{trans}^{\mathtt{B}}_1(s^{\mathtt{B}}_1, \lambda)$ and $\tilde{s}^{\mathtt{B}}_2 \in \mathrm{trans}^{\mathtt{B}}_2(s^{\mathtt{B}}_2, \lambda)$,*
- $\mathcal{S}^{\mathtt{PA}} \subseteq \mathcal{S}^{\mathtt{P}}$ *is a set of accepting states such that $(s^{\mathtt{B}}_1, s^{\mathtt{B}}_2) \in \mathcal{S}^{\mathtt{PA}}$ iff $s^{\mathtt{B}}_1 \in \mathcal{S}^{\mathtt{BA}}_1$ and $s^{\mathtt{B}}_2 \in \mathcal{S}^{\mathtt{BA}}_2$.*

Automaton A is also a nondeterministic Büchi automaton and accepts runs that are accepted by both automata A_1 and A_2. Fig. 1 depicts a minimal example of automata-based model checking, whose steps are presented as follows [52]:

1) Construct Automaton $A^{\mathtt{M}}$: Given a Kripke structure $G^{\mathtt{K}}$ (see Def. 11), it is converted into a nondeterministic Büchi automaton as described in [54]. Fig. 1a–b illustrate an exemplary $G^{\mathtt{K}}$ and the automaton $A^{\mathtt{M}}$ converted from $G^{\mathtt{K}}$.

2) Construct Automaton A^{φ}: An LTL formula φ can be readily translated into a Büchi automaton A^{φ} using, e.g., the tool Spot [50]. The reader is referred to [49], [50], [55] for further details. We use $\mathcal{A}^{\varphi} := \{\dots, A^{\varphi}_m, \dots\}$ to denote the set of automata converted from LTL formulas $\mathcal{F}$.

3) Retrieve Accepting Runs in Product Automaton $A^{\mathtt{P}}$: Let automaton $A^{\mathtt{P}}$ be the product of $A^{\mathtt{M}}$ and $\mathcal{A}^{\varphi}$ (see Sec. VII-A). Based on the Büchi acceptance condition [56], a run in a nondeterministic Büchi automaton is accepting if it visits some accepting states in $\mathcal{S}^{\mathtt{BA}}$ infinitely often. An accepting state is illustrated by a double circle (see Fig. 1b–d).

E. Driving Corridor

The reachable sets $\mathcal{R}_k$ of the ego vehicle enclose the collision-free solution space for motion planning; however, they may (a) be disconnected in the position domain due to the presence of obstacles and (b) contain states $\boldsymbol{x}_k$ having different valuations τ_k. This renders the direct usage of the reachable sets unsuitable for obtaining constraints for generating specification-compliant trajectories. To address this problem, we identify collision-free and specification-compliant *driving corridors* that are subsets of the reachable sets, which can be utilized as constraints over the states $\boldsymbol{x}_k$ in the motion planning problem. We present the necessary definitions.

Definition 8 (Connected Component)**.** *A connected component $\mathcal{C}_k \subseteq \mathcal{R}_k$ with valuation τ_k over $\mathcal{AP}$ is a set such that*

(a) $\mathcal{C}_k$ is a connected set [57] and collision-free in the position domain, i.e., $\mathcal{C}_k \cap \mathcal{X}^{\mathtt{F}}_k = \varnothing$,

(b) the states $\boldsymbol{x}_k$ in $\mathcal{C}_k$ have the same valuation τ_k.

Definition 9 (Driving Corridor)**.** *A driving corridor DC is a sequence of connected components $\mathcal{C}_k$ over steps 0 to k_h.*

Definition 10 (Specification-Compliant Driving Corridor)**.** *A driving corridor complying with specifications $\mathcal{F}$ is one such that $\forall \varphi \in \mathcal{F} : (\tau_0, \dots, \tau_{k_h}) \models \varphi$.*

F. Problem Statement

The problem we aim to solve is formally defined as follows:

Problem 1 (Optimal Specification-Compliant Driving Corridor Identification)**.** *The optimal specification-compliant driving corridor $DC^{\mathtt{O}}$ of the ego vehicle is one with the maximum utility over steps k:*

$$\max \sum_{k=0}^{k_h} u_k \tag{4a}$$

$$\text{subject to} \quad \boldsymbol{x}_0 \in \mathcal{C}_0, \tag{4b}$$

$$\forall k \in \{0, \dots, k_h - 1\} : \mathcal{C}_{k+1} \cap \mathrm{reach}(\mathcal{C}_k) \neq \varnothing, \tag{4c}$$

$$\forall \varphi \in \mathcal{F} : (\tau_0, \dots, \tau_{k_h}) \models \varphi, \tag{4d}$$

where u_k is the utility of $\mathcal{C}_k$ (see Sec. VII-C).

Constraints (4c) and (4d) respectively encode the reachability of successive connected components of $DC^{\mathtt{O}}$ and its compliance with the enforced specifications. Collision-freeness of $DC^{\mathtt{O}}$ follows directly from Def. 8. We aim to obtain $DC^{\mathtt{O}}$ and extract constraints over $\boldsymbol{x}_k$ for motion planning: Given a driving corridor, the motion planning problem can be formulated such that the trajectory of the ego vehicle is contained within the driving corridor.

Problem 2 (Motion Planning with Driving Corridor)**.** *Given a driving corridor, the motion planning problem is to minimize the cost function $J : \mathbb{R}^{n_{\boldsymbol{x}}} \times \mathbb{R}^{n_{\boldsymbol{u}}} \rightarrow \mathbb{R}$ over steps k:*

$$\min_{\boldsymbol{U}} \sum_{k=0}^{k_h} J(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{5a}$$

$$\text{subject to} \tag{5b}$$

$$\forall k \in \{0, \dots, k_h\} : \boldsymbol{x}_k \in \mathcal{C}_k, \tag{5c}$$

$$\forall k \in \{0, \dots, k_h - 1\} : \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k). \tag{5d}$$

III. Methodology

The input to our approach is the current environment model, including the road network, a curvilinear coordinate system $F^{\mathtt{L}}$, the set $\mathcal{F}$ of considered specifications, and all relevant obstacles, e.g., those perceived within a certain field of view of the ego vehicle. Without loss of generality, we assume the obstacles to be vehicles, each denoted by V_n. Furthermore, we assume that the predicted trajectories of all vehicles, e.g., their most likely trajectories, are given as input. For demonstration purposes, we consider interstate and intersection traffic rules formalized in [6], [7] as specifications. Nevertheless, our approach can be easily extended to handle other specifications expressible in MTLp_f, e.g., traffic rules described in [5], [8].

Let us introduce our approach for identifying collision-free and specification-compliant driving corridors, whose solution concept and relevant components are respectively illustrated in Fig. 2 and Fig. 3. As a first step, we apply reachability

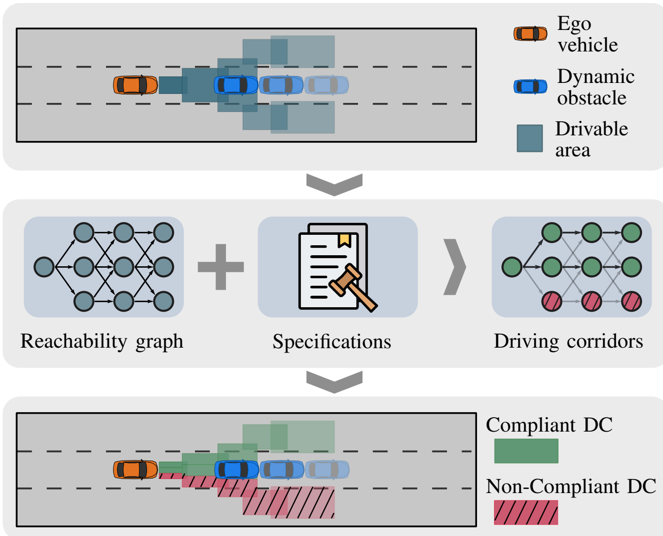


Fig. 2: Solution concept for identifying collision-free and specification-compliant driving corridors (DCs). Such driving corridors can be determined by model checking an automaton constructed from the reachability graph against automata translated from the enforced specifications. In this example, we assume overtaking from the right side is forbidden; thus, driving corridors corresponding to this maneuver are dismissed.

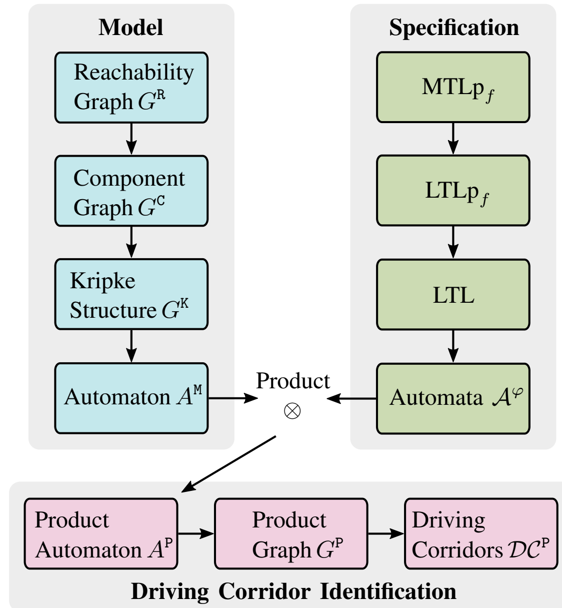


Fig. 3: Relationships of different components.

analysis on the ego vehicle to obtain its reachable set. Due to the presence of obstacles, the reachable set is represented as the union of multiple partial reachable sets, whose reachability and time relationships are stored in a *reachability graph* (see Sec. IV-A). Next, we process the reachability graph and the considered specifications for model checking:

1) As addressed in Sec. II-E, partial reachable sets may be disconnected or may contain states $\boldsymbol{x}_k$ having different valuations. To utilize their bounds as constraints for motion planning, they are grouped into *connected components*. These connected components form a *component graph*, based on which driving corridors are identified (see Sec. IV-B). A finite state automaton is constructed from the component graph and represents discrete state transitions of the ego vehicle over time (see Sec. II-D1).
2) The valuations of the partial reachable sets required for checking the compliance with specifications $\mathcal{F}$ are determined by evaluating a set of predicates (see Sec. V). Also, as motivated in Sec. II-B2, we rewrite MTLp_f formulas as LTL formulas, whose details are presented in Sec. VI. The LTL formulas are translated into multiple finite state automata determining the accepting sequences of discrete state transitions (see Sec. II-D2).
3) As the number of driving corridors within a component graph grows exponentially with the planning horizon k_h, we rely on model checking to efficiently and exhaustively identify driving corridors complying with the enforced specifications $\mathcal{F}$. By computing the synchronous product of all automata, we obtain a product automaton based on which specification-compliant sequences of discrete state transitions and their corresponding driving corridors are identified (see Sec. II-D3).

Since numerous candidate driving corridors may exist, we generate a *product graph* from the product automaton, from which the optimal driving corridor is identified based on user-defined utilities (see Sec. VII). If the solution to (5) cannot be found within a driving corridor, we select the next optimal driving corridor. As long as time permits, trajectories can be planned for each available driving corridor; thus, it is possible to obtain multiple trajectory options.

The state and input of our vehicle model for reachability analysis only capture the position, velocity, and acceleration components of the ego vehicle (see Sec. IV); therefore, specifications concerning other components such as orientation and jerk cannot be handled using our approach. We resort to a trajectory repairer [58] to repair the planned trajectories so that the unconsidered specifications are also satisfied (whenever possible). If this also does not work, we execute a fail-safe trajectory as described in [59].

IV. Reachability Analysis

We describe the computation of reachability graphs based on [43] as well as its component graphs and driving corridors.

A. Reachability Graph

As motivated in Sec. II-C, we aim to compute the over-approximations of the exact reachable set $\mathcal{R}^{\mathtt{e}}_k$ and drivable area $\mathcal{D}^{\mathtt{e}}_k$ of the ego vehicle. For computational efficiency, the dynamics of the ego vehicle is abstracted by two double integrators within the coordinate system $F^{\mathtt{L}}$, with the geometric center of the ego vehicle set as the reference point. The states and inputs in our model are $\boldsymbol{x}_k = (s_k, \dot{s}_k, d_k, \dot{d}_k)^\mathsf{T}$ and $\boldsymbol{u}_k = (\ddot{s}_k, \ddot{d}_k)^\mathsf{T}$, respectively:

$$\boldsymbol{x}_{k+1} = \underbrace{\begin{pmatrix} 1 & \Delta_t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta_t \\ 0 & 0 & 0 & 1 \end{pmatrix} \boldsymbol{x}_k + \begin{pmatrix} \frac{1}{2}\Delta_t^2 & 0 \\ \Delta_t & 0 \\ 0 & \frac{1}{2}\Delta_t^2 \\ 0 & \Delta_t \end{pmatrix} \boldsymbol{u}_k}_{f(\boldsymbol{x}_k, \boldsymbol{u}_k)}. \tag{6}$$

This abstraction ensures that the reachable sets of the adopted model always subsume those of high-fidelity vehicle models; alternative abstractions can be found in [60], [61]. Let $\square$ be a variable with its minimum and maximum values respectively denoted by $\underline{\square}$ and $\overline{\square}$. The velocities and accelerations at (s_k, d_k) are bounded by

$$\underline{\dot{s}}(\Gamma, s_k) \leq \dot{s}_k \leq \overline{\dot{s}}(\Gamma, s_k),\ \underline{\dot{d}}(\Gamma, s_k) \leq \dot{d}_k \leq \overline{\dot{d}}(\Gamma, s_k), \tag{7a}$$

$$\underline{\ddot{s}}(\Gamma, s_k) \leq \ddot{s}_k \leq \overline{\ddot{s}}(\Gamma, s_k),\ \underline{\ddot{d}}(\Gamma, s_k) \leq \ddot{d}_k \leq \overline{\ddot{d}}(\Gamma, s_k). \tag{7b}$$

These bounds are chosen conservatively to consider the kinematic limitations within a curvilinear coordinate system, see, e.g., [62]. As a final check, the drivability of the planned trajectories should be examined separately, e.g., using the drivability checker described in [63].

Following [11], we under-approximate $\mathrm{occ}(\boldsymbol{x}_k)$ by its inscribed circle and $\mathcal{O}_k$ by axis-aligned rectangles accounting for its arbitrary shape, yielding an under-approximative set of forbidden states in (2). Therefore, the over-approximative reachable sets $\mathcal{R}_k \supseteq \mathcal{R}_k^{\mathsf{e}}$ enclose all drivable trajectories of the ego vehicle. To reduce computational complexity, we adopt the union of so-called *base sets* $\mathcal{R}_k^{(i)}$, $i \in \mathbb{N}$, as the set representation for $\mathcal{R}_k$, i.e., $\mathcal{R}_k := \cup_i \mathcal{R}_k^{(i)}$. Every base set $\mathcal{R}_k^{(i)}$ is a Cartesian product of two convex polytopes that enclose the reachable positions and velocities of the ego vehicle in the $(s, \dot{s})$ and $(d, \dot{d})$ planes, respectively. To simplify the notation, we also denote the collection[2] of $\mathcal{R}_k^{(i)}$ with $\mathcal{R}_k$, i.e., $\mathcal{R}_k := \{\ldots, \mathcal{R}_k^{(i)}, \ldots\}$. The unified valuation of the states $\boldsymbol{x}_k$ within $\mathcal{R}_k^{(i)}$ over atomic propositions $\mathcal{AP}$ is denoted by $\tau_k^{(i)}$. A directed and acyclic *reachability graph* G^{R} is computed as described in [43] to store the relationships of $\mathcal{R}_k^{(i)}$ in terms of reachability, see Fig. 4a. An edge $(\mathcal{R}_k^{(i)}, \mathcal{R}_{k+1}^{(j)})$ in graph G^{R} indicates that set $\mathcal{R}_{k+1}^{(j)}$ is reachable from set $\mathcal{R}_k^{(i)}$ after one step. Similar to Def. 5, the projections of $\mathcal{R}_k$ and $\mathcal{R}_k^{(i)}$ onto the position domain are respectively denoted by $\mathcal{D}_k$ and $\mathcal{D}_k^{(i)}$.

B. Component Graph and Driving Corridors

To facilitate the identification of driving corridors, we group the base sets $\mathcal{R}_k^{(i)}$ in a graph G^{R} into connected components $\mathcal{C}_k^{(j)}$, whose collection is denoted by $\mathcal{C}_k^{\mathsf{C}}$. Based on Def. 8, every connected component $\mathcal{C}_k^{(j)}$ with valuation $\tau_k^{(j)}$ over $\mathcal{AP}$ is a collection of base sets $\mathcal{R}_k^{(i)}$ such that (a) sets $\mathcal{R}_k^{(i)}$ form a connected set [57] and their drivable areas $\mathcal{D}_k^{(i)}$ are collision-free and (b) sets $\mathcal{R}_k^{(i)}$ and $\mathcal{C}_k^{(j)}$ have the same valuation, i.e., $\tau_k^{(j)} = \tau_k^{(i)}$. Without loss of generality, we assume that

[2]Throughout this article, a set of sets is referred to as a *collection*.

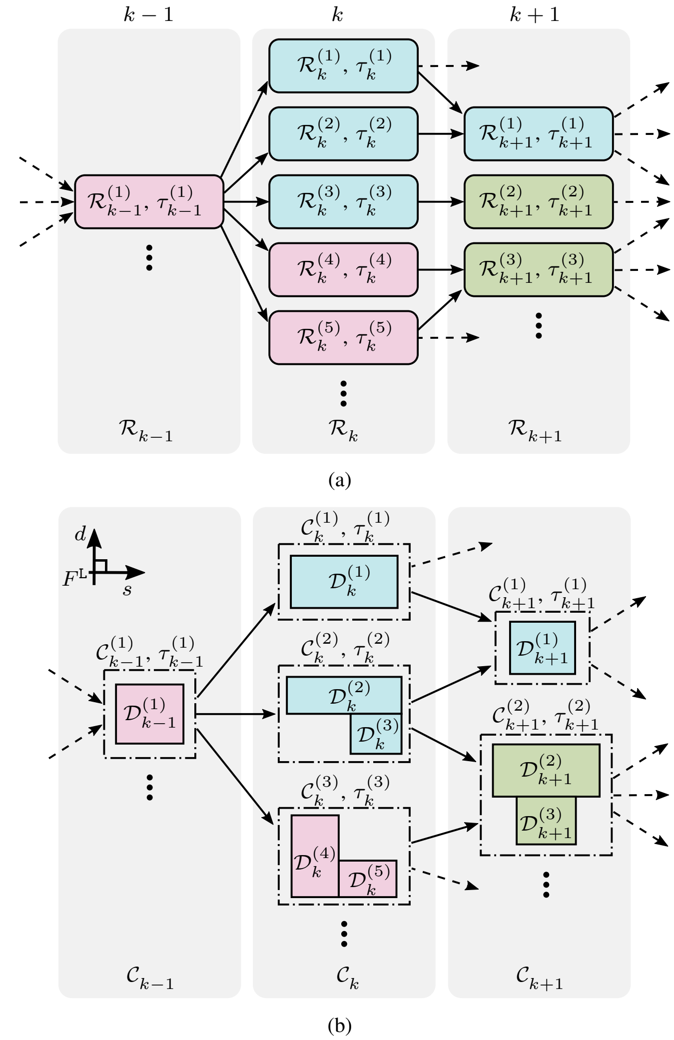


Fig. 4: A reachability graph G^{R} and its component graph G^{C}. Nodes of the same color have the same set of atomic propositions. (a) Graph G^{R} connecting nodes of different steps. (b) Graph G^{C} resulted from grouping the base sets $\mathcal{R}_k^{(i)}$ in G^{R} into connected components $\mathcal{C}_k^{(j)}$.

the set of initial states $\mathcal{X}_0$ of the ego vehicle is enclosed in the connected component $\mathcal{C}_0^{(1)}$. Connected components $\mathcal{C}_k^{(j)}$, together with edges connecting them, form a *component graph* G^{C}, see Fig. 4b. An edge $(\mathcal{C}_k^{(j)}, \mathcal{C}_{k+1}^{(l)})$ in G^{C} indicates that at least one base set in $\mathcal{C}_k^{(j)}$ reaches a base set in $\mathcal{C}_{k+1}^{(l)}$ within one step. We also define the Kripke structure [64] from a graph G^{C}, which is required for model checking, see Sec. II-D1. Fig. 1a shows an example of a Kripke structure G^{K}.

Definition 11 (Kripke Structure of Component Graph)**.** *The Kripke structure G^{K} of a component graph G^{C} is a four-tuple $(\mathcal{S}^{\mathsf{K}}, \mathcal{S}_0^{\mathsf{K}}, \mathrm{trans}^{\mathsf{K}}, \mathrm{label}^{\mathsf{K}})$:*

- $\mathcal{S}^{\mathsf{K}} = \{\ldots, s_k^{\mathsf{K},(j)}, \ldots\} \cup \{s^{\mathsf{K,D}}\}$ *is a set of states, where a state $s_k^{\mathsf{K},(j)}$ maps to a connected component $\mathcal{C}_k^{(j)}$ in G^{C}; $s^{\mathsf{K,D}}$ is an auxiliary self-looping state required for extending a trace in G^{K} to infinite length.*
- $\mathcal{S}_0^{\mathsf{K}} = \{s_0^{\mathsf{K},(1)}\}$ *is a set of initial states.*
- $\mathrm{trans}^{\mathsf{K}} : \mathcal{S}^{\mathsf{K}} \to \mathbb{P}(\mathcal{S}^{\mathsf{K}})$ *is a transition relation and is defined as: $s_{k+1}^{\mathsf{K},(l)} \in \mathrm{trans}^{\mathsf{K}}(s_k^{\mathsf{K},(j)})$ if the edge $(\mathcal{C}_k^{(j)}, \mathcal{C}_{k+1}^{(l)})$ exists in G^{C}; $s^{\mathsf{K,D}} \in \mathrm{trans}^{\mathsf{K}}(s_{k_h}^{\mathsf{K},(j)})$; $s^{\mathsf{K,D}} \in \mathrm{trans}^{\mathsf{K}}(s^{\mathsf{K,D}})$.*

TABLE I: SELECTION OF CONSIDERED PREDICATES.

Category	**Type**	**Predicate**	**Rule** (see [6], [7])
Position	VI	in_lanelet, on_main_carriageway, behind_stop_line, at_traffic_sign, ...	R-I5, R-IN1
	VD	in_front_of, behind, beside, left_of, right_of, in_same_lane, ...	R-G1, R-I2
Velocity	VI	keeps_lane_speed_limit, perserves_flow, in_standstill, ...	R-G3, R-G4, R-I1, R-IN2
	VD	keeps_safe_velocity_prec, drives_faster, ...	R-G1, R-I2
Acceleration	VI	admissible_braking	R-G2
Priority	VD	has_priority_over, same_priority_as, ...	R-IN3, R-IN4, R-IN5
Traffic Situation	VI	changes_lanelet, passing_stop_line, turning_left, turning_right, ...	R-IN1, R-IN3
	VD	slow_leading_vehicle, in_congestion, cut_in, ...	R-G1, R-G4, R-I1

- $\mathrm{label}^{\mathtt{K}} : \mathcal{S}^{\mathtt{K}} \to \mathbb{P}(\mathcal{AP})$ *is a labeling function that labels each state with a set of atomic propositions, which is defined as* $\sigma \in \mathrm{label}(s_k^{\mathtt{K},(j)})$ *if* $\tau_k^{(j)}(\sigma) = \mathtt{true}$.

Each path in a graph $G^{\mathtt{C}}$ corresponds to a collision-free driving corridor based on Def. 9. For example, let $k = 1$ and $k_h = 2$ in Fig. 4b. Sequences $DC_1 := (\mathcal{C}_0^{(1)}, \mathcal{C}_1^{(1)}, \mathcal{C}_2^{(1)})$ and $DC_2 := (\mathcal{C}_0^{(1)}, \mathcal{C}_1^{(3)}, \mathcal{C}_2^{(2)})$ correspond to two driving corridors of the ego vehicle. We utilize the position and velocity bounds of the connected component $\mathcal{C}_k^{(j)}$ within a driving corridor as constraints over the state $\boldsymbol{x}_k$ to restrict the planning space:

$$[\underline{\boldsymbol{x}}_k, \overline{\boldsymbol{x}}_k] = \mathrm{hull}(\mathcal{C}_k^{(j)}), \tag{8}$$

where $\mathrm{hull}(\cdot)$ returns the interval hull of a set.

V. PREDICATE EVALUATION

The valuations over atomic propositions required for determining the satisfaction of specifications are often generated by evaluating a set of *predicates* formulated in higher-order logic. Our predicates have the general form of $\mathrm{predicate}(\boldsymbol{x}_k; \cdot)$ and accept appropriate arguments. Tab. I lists selected predicates pertinent to rules formalized in [6], [7], which are divided into different categories and types. The evaluation of a *vehicle-dependent* (VD) predicate relies on other vehicles, whereas that of a *vehicle-independent* (VI) predicate does not.

Let us define some sets and functions to assist the evaluation of predicates. We denote by $\mathcal{L}$ the lanelets along the reference path Γ and their adjacent lanelets; the set $\mathcal{L}^{\mathtt{dir}} \subset \mathcal{L}$ refers to lanelets having the same driving direction as the ego vehicle. The lanelets occupied by the ego vehicle with state $\boldsymbol{x}_k$ are obtained as follows:

$$\mathrm{lanelets}(\boldsymbol{x}_k) := \left\{ L \in \mathcal{L} \,\middle|\, \mathrm{occ}(L) \cap \mathrm{occ}(\boldsymbol{x}_k) \neq \varnothing \right\},$$
$$\mathrm{lanelets\_dir}(\boldsymbol{x}_k) := \mathrm{lanelets}(\boldsymbol{x}_k) \cap \mathcal{L}^{\mathtt{dir}}.$$

The functions $\mathrm{type}(L)$ and $\mathrm{traffic\_sign}(L)$ return the type of a lanelet L (main carriageway, access ramp, etc.) and the set of traffic signs referenced by L, respectively. The functions $\mathrm{front}(\cdot)$ and $\mathrm{rear}(\cdot)$ return the s coordinate of the front and rear bumper of the input within $F^{\mathtt{L}}$, respectively. Variable $\boldsymbol{x}_{n,k}^{\mathtt{oth}}$ denotes the state of vehicle V_n at step k. We only describe a few exemplary predicates from each category for a concise presentation. The reader is referred to [6], [7] for detailed definitions of other predicates.

1) Position Predicates: Vehicle-independent position predicates relate to lanelets and traffic rule elements in the scenario. We provide three examples:

$$\mathrm{in\_lanelet}(\boldsymbol{x}_k; L) \Leftrightarrow L \in \mathrm{lanelets}(\boldsymbol{x}_k),$$
$$\mathrm{on\_main\_carriageway}(\boldsymbol{x}_k) \Leftrightarrow$$
$$\mathrm{main\_carriage\_way} \in \{\mathrm{type}(L) | L \in \mathrm{lanelets}(\boldsymbol{x}_k)\},$$
$$\mathrm{at\_traffic\_sign}(\boldsymbol{x}_k; TS) \Leftrightarrow$$
$$\exists L \in \mathrm{lanelets\_dir}(\boldsymbol{x}_k) : TS \in \mathrm{traffic\_sign}(L),$$

where TS stands for a traffic sign. Vehicle-dependent position predicates reflect positional relations between the ego vehicle and other vehicles. For example, the mutually exclusive predicates $\mathrm{in\_front\_of}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\mathtt{oth}})$, $\mathrm{behind}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\mathtt{oth}})$, and $\mathrm{beside}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\mathtt{oth}})$ along the s direction can be evaluated with respect to V_n as follows:

$$\mathrm{in\_front\_of}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\mathtt{oth}}) \Leftrightarrow \mathrm{rear}(\boldsymbol{x}_k) > \mathrm{front}(\boldsymbol{x}_{n,k}^{\mathtt{oth}}),$$
$$\mathrm{behind}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\mathtt{oth}}) \Leftrightarrow \mathrm{front}(\boldsymbol{x}_k) < \mathrm{rear}(\boldsymbol{x}_{n,k}^{\mathtt{oth}}),$$
$$\mathrm{beside}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\mathtt{oth}}) \Leftrightarrow \big(\mathrm{left\_of}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\mathtt{oth}}) \vee \mathrm{right\_of}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\mathtt{oth}})\big)$$
$$\wedge \neg\, \mathrm{in\_front\_of}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\mathtt{oth}}) \wedge \neg\, \mathrm{behind}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\mathtt{oth}}),$$

where the mutually exclusive predicates $\mathrm{left\_of}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\mathtt{oth}})$, $\mathrm{right\_of}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\mathtt{oth}})$, and $\mathrm{aligned\_with}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\mathtt{oth}})$ are analogously defined along the d direction.

2) Velocity Predicates: Vehicle-independent velocity predicates typically describe minimum or maximum velocity requirements. Rules R-G3 and R-G4 [7] specify different velocity limits that vehicles should respect, including limits introduced by the restricted field of view of a vehicle, the type of lane(let) in which the vehicle is driving, and the type of the vehicle. For instance, given the maximum velocity limit of the lane occupied with state $\boldsymbol{x}_k$, denoted by $\dot{s}^{\mathtt{lane}}$, we have:

$$\mathrm{keeps\_lane\_speed\_limit}(\boldsymbol{x}_k; \dot{s}^{\mathtt{lane}}) \Leftrightarrow \mathrm{proj}_{\dot{s}}(\boldsymbol{x}_k) \leq \dot{s}^{\mathtt{lane}}.$$

Examples of vehicle-dependent velocity predicates indicate whether the ego vehicle is driving at a safe velocity with respect to a leading vehicle or driving faster than a vehicle. The latter predicate can be evaluated as follows:

$$\mathrm{drives\_faster}(\boldsymbol{x}_k; \boldsymbol{x}_{n,k}^{\mathtt{oth}}) \Leftrightarrow \mathrm{proj}_{\dot{s}}(\boldsymbol{x}_k) \geq \mathrm{proj}_{\dot{s}}(\boldsymbol{x}_{n,k}^{\mathtt{oth}}).$$

3) Acceleration Predicates: These predicates relate to the acceleration component of the ego vehicle. As an example, rule R-G2 [7] specifies situations in which a vehicle is allowed to brake harder than a predefined threshold. If the input $\boldsymbol{u}_k$ of the state $\boldsymbol{x}_k$ is within the range of admissible acceleration, the predicate $\mathrm{admissible\_braking}(\boldsymbol{x}_k)$ evaluates to `true`.

4) Priority Predicates: Vehicles should respect driving priorities specified by traffic regulations, which can be inferred from the road structure or indicated by traffic signs. The predicates $\mathrm{has\_priority\_over}(\boldsymbol{x}_k; \boldsymbol{x}^{\mathtt{oth}}_{n,k})$ and $\mathrm{same\_priority\_as}(\boldsymbol{x}_k; \boldsymbol{x}^{\mathtt{oth}}_{n,k})$ reflect whether the ego vehicle has priority over V_n or has the same priority as V_n, respectively. They are evaluated by comparing the driving priorities determined based on the current traffic scenario and road priorities listed in [6, Tab. II].

5) Traffic Situation Predicates: The truth of these predicates depends on given traffic situations. For example, vehicle-independent predicates may indicate whether the ego vehicle is passing a stop line:

$$\begin{aligned}&\mathrm{passing\_stop\_line}(\boldsymbol{x}_k) \Leftrightarrow \\ &\qquad \mathrm{behind\_stop\_line}(\boldsymbol{x}_k) \wedge \mathbf{X}(\neg\, \mathrm{behind\_stop\_line}(\boldsymbol{x}_k)),\end{aligned}$$

where $\mathrm{behind\_stop\_line}(\boldsymbol{x}_k)$ evaluates to `true` if the ego vehicle is behind a stop line. Vehicle-dependent predicates may indicate whether a slow leading vehicle exists and whether a vehicle is stuck in traffic congestion.

VI. Specification Rewriting

As motivated in Sec. II-B2, we rewrite and interpret an MTLp$_f$ formula $\varphi^{\mathtt{M}}$ as an LTL formula φ on our system:

1) Eliminate Intervals over Temporal Connectives: Since valuations τ_k of our traces are synchronized with steps k, we rewrite an MTLp$_f$ connective as a combination of $\mathbf{X}$ and $\mathbf{Y}$ connectives in LTLp$_f$. We use the notation $\mathbf{X}_{[\tilde{k}]}$ as a shorthand for $\tilde{k}$ consecutive $\mathbf{X}$ connectives:

$$\mathbf{X}_{[\tilde{k}]}\varphi^{\mathtt{L}} := \underbrace{\mathbf{X}\mathbf{X}\ldots\mathbf{X}}_{\tilde{k}\ \text{times}\ \mathbf{X}}\varphi^{\mathtt{L}}. \tag{9}$$

It follows from the semantics of $\mathbf{X}_I$, $\mathbf{U}_I$ (see Sec. II-B1) that the time interval $I = [a, b]$ over future-time connectives can be eliminated:

$$\mathbf{X}_{[a,b]}\varphi^{\mathtt{L}} = \begin{cases}\mathbf{X}\varphi^{\mathtt{L}}, & \text{if } 1 \in [a, b].\\ \bot, & \text{otherwise.}\end{cases} \tag{10}$$

$$\varphi_1^{\mathtt{L}}\mathbf{U}_{[a,b]}\varphi_2^{\mathtt{L}} = \bigvee_{a\le\tilde{k}\le b}\big(\mathbf{G}_{[0,\tilde{k}-1]}\varphi_1^{\mathtt{L}} \wedge \mathbf{X}_{[\tilde{k}]}\varphi_2^{\mathtt{L}}\big), \tag{11}$$

$$\mathbf{G}_{[a,b]}\varphi^{\mathtt{L}} = \bigwedge_{a\le\tilde{k}\le b}\mathbf{X}_{[\tilde{k}]}\varphi^{\mathtt{L}}, \tag{12}$$

$$\mathbf{F}_{[a,b]}\varphi^{\mathtt{L}} = \bigvee_{a\le\tilde{k}\le b}\mathbf{X}_{[\tilde{k}]}\varphi^{\mathtt{L}}. \tag{13}$$

That is, $\mathbf{X}_{[a,b]}\varphi^{\mathtt{L}}$ is only satisfiable by τ if the unit step jump is within $[a, b]$ and $\varphi^{\mathtt{L}}$ holds in the next valuation; $\varphi_1^{\mathtt{L}}\mathbf{U}_{[a,b]}\varphi_2^{\mathtt{L}}$ is satisfied if within a and b steps, a valuation occurs in which $\varphi_2^{\mathtt{L}}$ holds, and $\varphi_1^{\mathtt{L}}$ continuously holds for valuations before that; $\mathbf{G}_{[a,b]}\varphi^{\mathtt{L}}$ ($\mathbf{F}_{[a,b]}\varphi^{\mathtt{L}}$) is satisfied if $\varphi^{\mathtt{L}}$ holds in all (any) valuations occurring within a and b steps. Intervals over the past-time connectives $\mathbf{Y}_I$, $\mathbf{O}_I$, $\mathbf{H}_I$, and $\mathbf{S}_I$ can be analogously eliminated by rewriting using the $\mathbf{Y}$ connective.

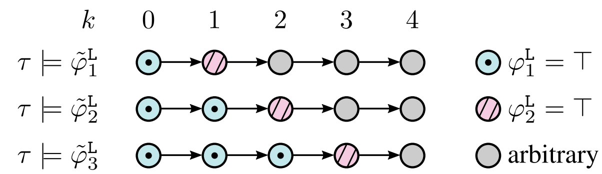


Fig. 5: Example traces satisfying $\tilde{\varphi}_1^{\mathtt{L}}$, $\tilde{\varphi}_2^{\mathtt{L}}$, and $\tilde{\varphi}_3^{\mathtt{L}}$, respectively. $\tilde{\varphi}_1^{\mathtt{L}} := \mathbf{G}_{[0,0]}\varphi_1^{\mathtt{L}} \wedge \mathbf{X}_{[1]}\varphi_2^{\mathtt{L}}$, $\tilde{\varphi}_2^{\mathtt{L}} := \mathbf{G}_{[0,1]}\varphi_1^{\mathtt{L}} \wedge \mathbf{X}_{[2]}\varphi_2^{\mathtt{L}}$, $\tilde{\varphi}_3^{\mathtt{L}} := \mathbf{G}_{[0,2]}\varphi_1^{\mathtt{L}} \wedge \mathbf{X}_{[3]}\varphi_2^{\mathtt{L}}$. A circle represents a valuation in which the atomic proposition corresponding to the color is assigned `true`. A sequence of circles represents a trace.

Running example:

$$\begin{aligned}&\varphi_1^{\mathtt{L}}\mathbf{U}_{[1,3]}\varphi_2^{\mathtt{L}} \overset{(11)}{=} \tilde{\varphi}_1^{\mathtt{L}} \vee \tilde{\varphi}_2^{\mathtt{L}} \vee \tilde{\varphi}_3^{\mathtt{L}}, \text{where}\\ &\qquad \tilde{\varphi}_1^{\mathtt{L}} := \mathbf{G}_{[0,0]}\varphi_1^{\mathtt{L}} \wedge \mathbf{X}_{[1]}\varphi_2^{\mathtt{L}} \overset{(12)}{=} \varphi_1^{\mathtt{L}} \wedge \mathbf{X}\varphi_2^{\mathtt{L}},\\ &\qquad \tilde{\varphi}_2^{\mathtt{L}} := \mathbf{G}_{[0,1]}\varphi_1^{\mathtt{L}} \wedge \mathbf{X}_{[2]}\varphi_2^{\mathtt{L}} \overset{(12)}{=} \varphi_1^{\mathtt{L}} \wedge \mathbf{X}\varphi_1^{\mathtt{L}} \wedge \mathbf{X}_{[2]}\varphi_2^{\mathtt{L}},\\ &\qquad \tilde{\varphi}_3^{\mathtt{L}} := \mathbf{G}_{[0,2]}\varphi_1^{\mathtt{L}} \wedge \mathbf{X}_{[3]}\varphi_2^{\mathtt{L}}\\ &\qquad\quad \overset{(12)}{=} \varphi_1^{\mathtt{L}} \wedge \mathbf{X}\varphi_1^{\mathtt{L}} \wedge \mathbf{X}_{[2]}\varphi_1^{\mathtt{L}} \wedge \mathbf{X}_{[3]}\varphi_2^{\mathtt{L}}.\end{aligned}$$

Fig. 5 shows traces satisfying $\tilde{\varphi}_1^{\mathtt{L}}$, $\tilde{\varphi}_2^{\mathtt{L}}$, and $\tilde{\varphi}_3^{\mathtt{L}}$.

2) Eliminate Past-Time Connectives: A syntactic procedure for separating past-time and future-time connectives in LTL is presented in [65], which has been further applied to LTLp$_f$ in [47]. Although the procedure offers straightforward rules for rewriting, it leads to so-called non-elementary blow-up in formula size [66], [67]. As an alternative, one can explicitly reformulate $\varphi^{\mathtt{L}}$ using only future-time connectives with an exponential growth in the formula size [68]. For practical reasons, we adopt a less strict rewriting procedure to avoid the mentioned unfavorable complexities. To this end, we temporarily switch from the *strong* semantics defined in Sec. II-B1 to the *repeat* semantics as described in [69] for the $\mathbf{X}$ and $\mathbf{Y}$ connectives, which allows one to cancel out pairs of $\mathbf{X}$ and $\mathbf{Y}$: Intuitively, we expect that the *previous* step of the *next* step along a trace τ is the *current* step, and vice versa:

$$\mathbf{X}\mathbf{Y}\varphi^{\mathtt{L}} \Rightarrow \varphi^{\mathtt{L}}, \tag{14}$$

$$\mathbf{Y}\mathbf{X}\varphi^{\mathtt{L}} \Rightarrow \varphi^{\mathtt{L}}. \tag{15}$$

After canceling out all pairs of $\mathbf{X}$ and $\mathbf{Y}$, we restore the *strong* semantics for interpreting the remaining $\mathbf{Y}\varphi^{\mathtt{L}}$, i.e., they all evaluate to $\bot$: $\mathbf{Y}\varphi^{\mathtt{L}}$ asserts that there exists a valuation prior to τ^0 and $\varphi^{\mathtt{L}}$ is true therein, which does not hold since our traces start with τ^0 at step $k = 0$. As for the $\mathbf{S}$ connective, we apply the axiom [70, A12]

$$\varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}} = \varphi_2^{\mathtt{L}} \vee (\varphi_1^{\mathtt{L}} \wedge \mathbf{Y}(\varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}})) \tag{16}$$

and examine the expanded formula.

Running example:

$$\begin{aligned}&\mathbf{F}_{[0,2]}(\varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}}) \overset{(13)}{=} \tilde{\varphi}_1^{\mathtt{L}} \vee \tilde{\varphi}_2^{\mathtt{L}} \vee \tilde{\varphi}_3^{\mathtt{L}}, \text{where}\\ &\qquad \tilde{\varphi}_1^{\mathtt{L}} := \varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}}\\ &\qquad\quad \overset{(16)}{=} \varphi_2^{\mathtt{L}} \vee (\varphi_1^{\mathtt{L}} \wedge \mathbf{Y}(\varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}})) = \varphi_2^{\mathtt{L}},\\ &\qquad \tilde{\varphi}_2^{\mathtt{L}} := \mathbf{X}_{[1]}(\varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}})\\ &\qquad\quad \overset{(16)}{=} \mathbf{X}(\varphi_2^{\mathtt{L}} \vee (\varphi_1^{\mathtt{L}} \wedge \mathbf{Y}(\varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}})))\end{aligned}$$

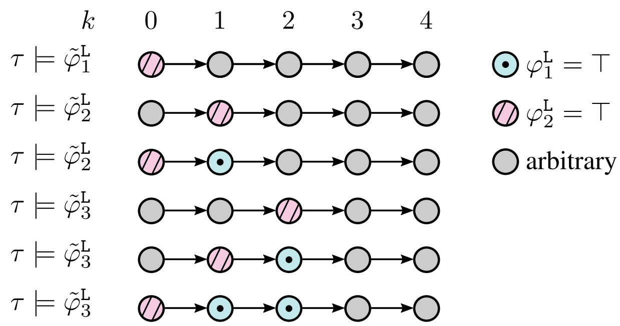


Fig. 6: Example traces satisfying $\tilde{\varphi}_1^{\mathtt{L}}$, $\tilde{\varphi}_2^{\mathtt{L}}$, and $\tilde{\varphi}_3^{\mathtt{L}}$, respectively. $\tilde{\varphi}_1^{\mathtt{L}} := \varphi_2^{\mathtt{L}}$, $\tilde{\varphi}_2^{\mathtt{L}} := \mathbf{X}\varphi_2^{\mathtt{L}} \vee (\mathbf{X}\varphi_1^{\mathtt{L}} \wedge \varphi_2^{\mathtt{L}})$, $\tilde{\varphi}_3^{\mathtt{L}} := \mathbf{X}_{[2]}\varphi_2^{\mathtt{L}} \vee (\mathbf{X}_{[2]}\varphi_1^{\mathtt{L}} \wedge (\mathbf{X}\varphi_2^{\mathtt{L}} \vee (\mathbf{X}\varphi_1^{\mathtt{L}} \wedge \varphi_2^{\mathtt{L}})))$.

$$
\begin{aligned}
&= \mathbf{X}\varphi_2^{\mathtt{L}} \vee (\mathbf{X}\varphi_1^{\mathtt{L}} \wedge (\underline{\varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}}}))\\
&\overset{\tilde{\varphi}_1^{\mathtt{L}}}{=} \mathbf{X}\varphi_2^{\mathtt{L}} \vee (\mathbf{X}\varphi_1^{\mathtt{L}} \wedge \underline{\varphi_2^{\mathtt{L}}}),\\
\tilde{\varphi}_3^{\mathtt{L}} &:= \mathbf{X}_{[2]}(\varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}})\\
&\overset{(16)}{=} \mathbf{X}_{[2]}(\varphi_2^{\mathtt{L}} \vee (\varphi_1^{\mathtt{L}} \wedge \mathbf{Y}(\varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}})))\\
&= \mathbf{X}_{[2]}\varphi_2^{\mathtt{L}} \vee (\mathbf{X}_{[2]}\varphi_1^{\mathtt{L}} \wedge \underline{\mathbf{X}(\varphi_1^{\mathtt{L}}\mathbf{S}\varphi_2^{\mathtt{L}})})\\
&\overset{\tilde{\varphi}_2^{\mathtt{L}}}{=} \mathbf{X}_{[2]}\varphi_2^{\mathtt{L}} \vee\\
&\quad (\mathbf{X}_{[2]}\varphi_1^{\mathtt{L}} \wedge (\underline{\mathbf{X}\varphi_2^{\mathtt{L}} \vee (\mathbf{X}\varphi_1^{\mathtt{L}} \wedge \varphi_2^{\mathtt{L}})})).
\end{aligned}
$$

Fig. 6 shows traces satisfying $\tilde{\varphi}_1^{\mathtt{L}}$, $\tilde{\varphi}_2^{\mathtt{L}}$, and $\tilde{\varphi}_3^{\mathtt{L}}$.

3) Conversion to LTL: An LTLp$_f$ formula without past-time connectives is converted to an LTL formula φ based on [71], [72]. This conversion introduces an auxiliary atomic proposition D (***Dead***) in φ and an auxiliary self-looping state $s^{\mathtt{B,D}}$ in the automaton A^{φ} translated from φ (see Fig. 1c).

VII. Driving Corridor Identification

This section describes the computation of automaton $A^{\mathtt{P}}$ and the generation of its product graph, based on which we identify (optimal) specification-compliant driving corridors.

A. Product Automaton Computation

Given an automaton $A^{\mathtt{M}}$ and a set of automata $\mathcal{A}^{\varphi}$ (see Sec. II-D2), their product $A^{\mathtt{P}}$ can be computed considering factors such as flexibility and priorities in case the specifications are conflicting, i.e., cannot be satisfied by any trace. Rulebooks [73] specify qualitative relations between specifications as a pre-order and assigns the same priority to specifications in a group. Following this concept, automaton $A^{\mathtt{P}}$ can be computed to expedite compliance with the groups of specifications of higher priorities:

$$
A_0^{\mathtt{P}} = A^{\mathtt{M}},\ A_{\tilde{m}}^{\mathtt{P}} = A_{\tilde{m}-1}^{\mathtt{P}} \otimes \underbrace{(\cdots \otimes A_m^{\varphi} \otimes \dots)}_{\tilde{A}_{\tilde{m}}^{\varphi}}, \tag{17}
$$

where $\tilde{A}_{\tilde{m}}^{\varphi}$ denotes the product of the automata in a group with priority $\tilde{m}$ (a smaller $\tilde{m}$ indicates a higher priority). Automaton $A_{\tilde{m}-1}^{\mathtt{P}}$ is assigned to $A^{\mathtt{P}}$ if an accepting run exists in $A_{\tilde{m}-1}^{\mathtt{P}}$ but not in $A_{\tilde{m}}^{\mathtt{P}}$. Two special instances with drawbacks exist that should ideally be avoided:

Algorithm 1 Remove Unreachable Base Sets

Inputs: Collections $\mathcal{C}_k^{\mathtt{C}}$ of connected components $\mathcal{C}_k^{(j)}$.
Output: Updated connected components $\mathcal{C}_k^{(j)}$.

```
R_0^keep ← C_0^(1).BASESETS()                         ▷ Initialization
for k = 1 to k_h do
    R_k^keep ← ∅                 ▷ Collection of base sets to keep at k
    for C_k^(j) ∈ C_k^C do
        R^C ← C_k^(j).BASESETS()        ▷ Base sets to keep in C_k^(j)
        for R_k^(i) ∈ C_k^(j).BASESETS() do
            if R_k^(i).PARENTBASESETS() ∩ R_{k-1}^keep = ∅ then
                R^C ← R^C \ {R_k^(i)}                  ▷ Remove R_k^(i)
            else
                R_k^keep ← R_k^keep ∪ {R_k^(i)}        ▷ Keep R_k^(i) at k
            end if
        end for
        C_k^(j).BASESETS() ← R^C          ▷ Update base sets in C_k^(j)
    end for
end for
```

1) Assigning the Same Priority to All Automata A_m^{φ}: This instance yields an exponential growth in the number of states in $\tilde{A}_{\tilde{m}}^{\varphi}$ with respect to $|\mathcal{A}^{\varphi}|$ (see Def. 7). Moreover, it does not allow one to flexibly adjust enforced specifications per the current traffic situation or their orders based on user-defined measures such as importance or criticality. The latter property is unfavorable when not all prescribed specifications can be satisfied: possible reasons are conflicts in the specifications, misbehavior of other vehicles, etc.

2) Assigning a Unique Priority to Each Automaton A_m^{φ}: This instance allows one to explicitly prioritize the specifications and expedite compliance with those of higher priorities; however, meticulously ordering specifications becomes non-trivial as $|\mathcal{A}^{\varphi}|$ increases.

B. Product Graph Generation

Given an automaton $A^{\mathtt{P}}$ with at least an accepting run, we convert it into a directed, acyclic, and weighted graph $G^{\mathtt{P}}$, which is referred to as a *product graph*. Graph $G^{\mathtt{P}}$ retains the general structure of $A^{\mathtt{P}}$ and consists of nodes referencing corresponding connected components $\mathcal{C}_k^{(j)}$. States in $A^{\mathtt{P}}$ with outgoing edges for which the auxiliary atomic proposition D is assigned `true` are dismissed in $G^{\mathtt{P}}$ since they are irrelevant to the identification of driving corridors, see Fig. 7. Every edge $(\mathcal{C}_{k-1}^{(l)}, \mathcal{C}_k^{(j)})$ in $G^{\mathtt{P}}$ is weighted by the utility of $\mathcal{C}_k^{(j)}$, denoted by $u_k^{(j)}$ (detailed in Sec. VII-C). The paths in $G^{\mathtt{P}}$ from $\mathcal{C}_0^{(1)}$ to $\mathcal{C}_{k_h}^{(j)}$ correspond to specification-compliant driving corridors (see Def. 9 and Def. 10) and are stored in a collection $\mathcal{DC}^{\mathtt{P}}$.

Let $\mathcal{C}^{\mathtt{P}}$ and $\mathcal{C}^{\mathtt{C}}$ represent the collection of all connected components in graphs $G^{\mathtt{P}}$ and $G^{\mathtt{C}}$, respectively. Since $\mathcal{C}^{\mathtt{P}} \subseteq \mathcal{C}^{\mathtt{C}}$, we update the reachability relationship between the base sets $\mathcal{R}_k^{(i)}$ in the connected components and by this remove $\mathcal{R}_k^{(i)}$ that no longer have a valid parent. For example, suppose $DC_1 := (\mathcal{C}_0^{(1)}, \mathcal{C}_1^{(2)}, \mathcal{C}_2^{(2)})$ in Fig. 4b is the only path in $G^{\mathtt{P}}$, set $\mathcal{R}_2^{(3)}$ is no longer reachable along DC_1 as per Fig. 4a. Alg. 1 removes unreachable base sets from connected components: For every step k, we maintain a collection $\mathcal{R}_k^{\mathtt{keep}}$ of base sets to be kept, with $\mathcal{R}_0^{\mathtt{keep}}$ initialized with the base sets in $\mathcal{C}_0^{(1)}$

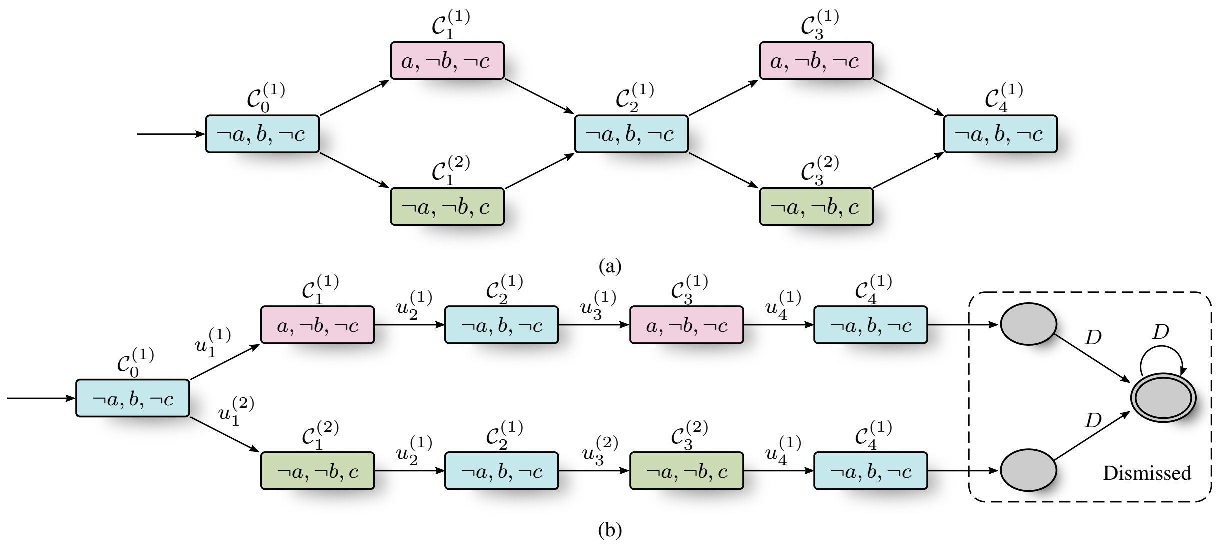


Fig. 7: Example of a component graph $G^{\mathtt{C}}$ and a product graph $G^{\mathtt{P}}$. Nodes of the same color have the same set of true atomic propositions. (a) Graph $G^{\mathtt{C}}$ with $k_h = 4$. (b) Graph $G^{\mathtt{P}}$ converted from a product automaton $A^{\mathtt{P}}$, which is the output of model checking $G^{\mathtt{C}}$ against specification $\varphi := \mathbf{G}(\neg a) \vee \mathbf{G}(\neg c)$: either a never holds, or c never holds. Dismissed states in $A^{\mathtt{P}}$ are shown in gray (cf. Fig. 1d).

(Alg. 1, line 1). For steps 1 to k_h, we iterate through $\mathcal{C}_k^{(j)} \in \mathcal{C}_k^{\mathtt{C}}$ and examine each of its base sets $\mathcal{R}_k^{(i)}$. If none of the parent base sets of $\mathcal{R}_k^{(i)}$ is present in $\mathcal{R}_{k-1}^{\mathtt{keep}}$, $\mathcal{R}_k^{(i)}$ is removed from $\mathcal{C}_k^{(j)}$; otherwise it is added to $\mathcal{R}_k^{\mathtt{keep}}$ (Alg. 1, lines 7–11).

C. Utility Computation

Identifying the optimal specification-compliant driving corridor, i.e., the solution to Prob. 1, requires computing the utility $u_k^{(j)}$ of connected components $\mathcal{C}_k^{(j)}$. Since multiple base sets $\mathcal{R}_k^{(i)}$ may exist in a connected component $\mathcal{C}_k^{(j)}$, we define a function $\mathrm{w\_mean}(\mathcal{C}_k^{(j)}, \Diamond)$ that returns the weighted mean of component $\Diamond$ in $\mathcal{C}_k^{(j)}$:

$$\mathrm{w\_mean}(\mathcal{C}_k^{(j)}, \Diamond) := \sum_{\mathcal{R}_k^{(i)} \in \mathcal{C}_k^{(j)}} w_k^{(i)} \, \mathrm{mean}\big(\mathrm{proj}_{(\Diamond)}(\mathcal{R}_k^{(i)})\big), \tag{18}$$

$$w_k^{(i)} := \frac{\mathrm{area}(\mathcal{R}_k^{(i)})}{\sum_{\mathcal{R}_k^{(l)} \in \mathcal{C}_k^{(j)}} \mathrm{area}(\mathcal{R}_k^{(l)})}, \tag{19}$$

where $w_k^{(i)}$ is the weight of $\mathcal{R}_k^{(i)}$ within $\mathcal{C}_k^{(j)}$ and $\mathrm{area}(\cdot)$ returns the area of the input in the position domain. The utility $u_k^{(j)}$ of $\mathcal{C}_k^{(j)}$ is defined as the weighted sum of partial utilities:

$$u_k^{(j)} := \boldsymbol{w}^{\mathsf{T}} \boldsymbol{u}_k^{(j)}, \tag{20}$$

where $\boldsymbol{w}$ is a weighting vector and $\boldsymbol{u}_k^{(j)}$ is a vector of user-defined partial utilities. We consider the following partial utilities, which are all normalized to $[0, 1]$:

1) Area: We reward $\mathcal{C}_k^{(j)}$ of a larger area in the position domain since this generally yields more flexible position constraints for subsequent trajectory planning:

$$u^{\mathtt{area}}(\mathcal{C}_k^{(j)}) := \frac{\mathrm{area}(\mathcal{C}_k^{(j)})}{\max_{\mathcal{C}_k^{(l)} \in \mathcal{C}^{\mathtt{P}}} \mathrm{area}\, \mathcal{C}_k^{(l)}}. \tag{21}$$

2) Velocity: We reward $\mathcal{C}_k^{(j)}$ of higher weighted longitudinal velocity to increase the traffic flow:

$$u^{\mathtt{vel}}(\mathcal{C}_k^{(j)}) := \frac{\mathrm{w\_mean}(\mathcal{C}_k^{(j)}, \dot{s}) - \dot{s}_0}{\overline{\ddot{s}}(\Gamma, s_k)\, \Delta_t\, k}. \tag{22}$$

3) Position: We encourage $\mathcal{C}_k^{(j)}$ of longer weighted traveled distance in the longitudinal direction of the reference path:

$$u^{\mathtt{pos}}(\mathcal{C}_k^{(j)}) := \frac{\mathrm{w\_mean}(\mathcal{C}_k^{(j)}, s) - s_0}{0.5\, \overline{\ddot{s}}(\Gamma, s_k)\, (\Delta_t\, k)^2 + \dot{s}_0\, \Delta_t\, k}, \tag{23}$$

4) Reference Path: We penalize $\mathcal{C}_k^{(j)}$ of larger weighted lateral deviation from the reference path:

$$u^{\mathtt{ref}}(\mathcal{C}_k^{(j)}) := \exp\big(-w^{\mathtt{ref}}\ \mathrm{w\_mean}(\mathcal{C}_k^{(j)}, d)\big), \tag{24}$$

where $w^{\mathtt{ref}} \in \mathbb{R}_+$ is a factor dictating how fast $u^{\mathtt{ref}}(\mathcal{C}_k^{(j)})$ approaches zero as the lateral deviation increases. Alternative utilities such as comfort, criticality measures, and robustness degrees of specifications can be taken into consideration, whose computation is out of the scope of this article.

D. Optimal Driving Corridor

Given a graph $G^{\mathtt{P}}$, graph-search and sampling-based techniques can be employed to extract optimal paths in $G^{\mathtt{P}}$ with respect to $u_k^{(j)}$. For instance, the longest paths from the root node $\mathcal{C}_0^{(1)}$ to nodes $\mathcal{C}_{k_h}^{(j)}$ can be efficiently obtained using a single-source shortest path algorithm on graph $-G^{\mathtt{P}}$ in which the weights are negated [74]. These paths correspond to collision-free and specification-compliant driving corridors with the maximum cumulative weights and are stored in the collection $\mathcal{DC}^{\mathtt{O}}$. Every candidate in $\mathcal{DC}^{\mathtt{O}}$ is processed again using Alg. 1 to remove unreachable base sets. We identify the optimal driving corridor $DC^{\mathtt{O}}$ with the highest cumulative weight, within which trajectories are planned.

TABLE II: SELECTED PARAMETERS USED IN THE EXPERIMENTS

Parameter	k_h	Δ_t	$\underline{\dot{s}}$	$\overline{\dot{s}}$	$\underline{\dot{d}}$	$\overline{\dot{d}}$	$\underline{\ddot{s}}$	$\overline{\ddot{s}}$	$\underline{\ddot{d}}$	$\overline{\ddot{d}}$
Value	15	0.2	0.0	20.0	−4.0	4.0	−6.0	6.0	−2.0	2.0

VIII. EVALUATION

This section evaluates our approach and demonstrates its applicability, effectiveness, and efficiency. To this end, we integrate identified driving corridors into two sampling-based motion planners and compare the planning results under different traffic scenarios and specifications. In addition, we evaluate the performance of our approach under increasingly critical scenarios and compare computation times. Furthermore, we benchmark of computation time of our prototype against multiple scenarios. Lastly, we compare our approach with that described in [19].

A. Implementation Details

For evaluation, we adopt scenarios from the CommonRoad benchmark suite[3] [75], whose typical components are a road network consisting of lanelets, static and dynamic obstacles, traffic rule elements such as traffic signs and traffic lights, the initial state of the ego vehicle, and a goal region. Every scenario has a unique benchmark ID and can be unambiguously reproduced. The prototype of our approach extends [76] and is partially implemented in Python and C++. We ran the experiments on a laptop with an Intel Core i7-7700HQ 2.8GHz processor. Tab. II lists selected parameters. The weights in $\boldsymbol{w}$ for driving corridor identification are all empirically set to 1.0. We briefly introduce the two adopted motion planners:

1) Reactive Planner: The popular motion planner described in [77], which we refer to as the reactive planner, generates a finite set of candidate trajectories connecting the initial state of the ego vehicle to different goal states. These goal states are generated based on samples of longitudinal velocity, lateral position, and the terminal time of the lateral maneuver. The candidate trajectories are checked for (a) feasibility (including drivability and collisions) using the drivability checker in [63] and (b) compliance with specifications using Spot [50].

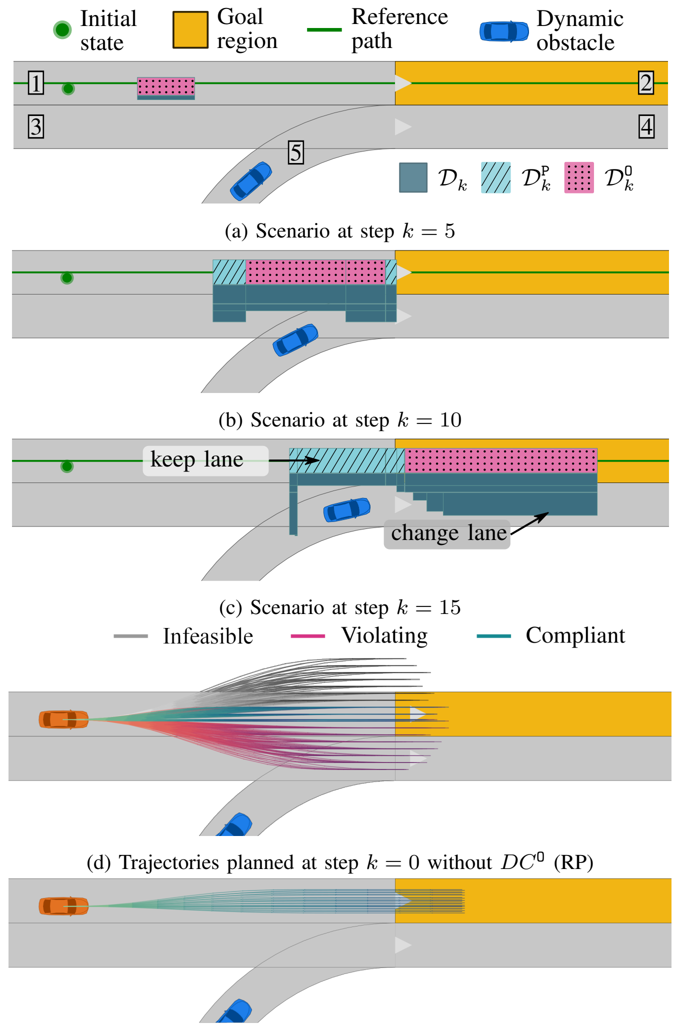
2) RRT:* To showcase the possibility of integrating our approach with RRT-based planners that we reviewed in Sec. I-A, we also consider the RRT* planner [41]. In our implementation, a tree is incrementally constructed from sampled nodes, between which a trajectory is generated using Dubins car model [78]. As with the reactive planner, we check the feasibility and compliance with the specifications of the trajectories and terminate once a solution is found.

Besides these planners, it has been shown in [12], [13] that optimization-based planners also substantially benefit from the integration of driving corridors.

B. Scenario I: Merging via On-Ramp

Fig. 8 depicts a baseline scenario where the ego vehicle is driving on a two-lane main carriageway and another vehicle

[3] http://commonroad.in.tum.de/



(a) Scenario at step $k = 5$

(b) Scenario at step $k = 10$

(c) Scenario at step $k = 15$

(d) Trajectories planned at step $k = 0$ without DC^0 (RP)

(e) Trajectories planned at step $k = 0$ with DC^0 (RP+DC/RP+DC*)

Fig. 8: Drivable areas and planned trajectories for scenario I (benchmark ID: `ZAM_TIV-1_1_T-1`).

is approaching via an on-ramp. While the ego vehicle is dynamically able to proceed in its current lane or to change to the lane on the right, rule R-I5 [7] prohibits the latter maneuver as the ego vehicle has to respect entering vehicles:

$$
\begin{aligned}
&\mathbf{G}\Big(\big(\mathrm{on\_main\_carriageway}(\boldsymbol{x}_k) \wedge \mathrm{behind}(\boldsymbol{x}_k; \boldsymbol{x}^{\mathtt{oth}}_{n,k}) \wedge \\
&\quad \mathrm{on\_access\_ramp}(\boldsymbol{x}^{\mathtt{oth}}_{n,k}) \wedge \\
&\quad \mathbf{F}\big(\mathrm{on\_main\_carriageway}(\boldsymbol{x}^{\mathtt{oth}}_{n,k})\big)\big) \\
&\quad \Rightarrow \\
&\quad \big(\mathrm{on\_main\_carriageway\_right\_lane}(\boldsymbol{x}_k) \vee \\
&\quad \mathbf{G}\big(\neg\, \mathrm{on\_main\_carriageway\_right\_lane}(\boldsymbol{x}_k)\big)\big)\Big).
\end{aligned}
$$

In addition to the baseline scenario, we create two alternative scenarios with increased difficulty by adding secondary specifications: in variant 1, we require that the ego vehicle reaches lanelet 2 before the end of the planning horizon; in variant 2, lanelet 2 should be reached between steps 5 and 12, which is a stricter requirement with a smaller solution space.

Fig. 8a–c visualize the computed drivable areas at different steps. Because the connected components in $G^{\mathtt{P}}$ reference a subset of base sets in G^{R}, their drivable areas at step k, denoted by $\mathcal{D}_k^{\mathsf{P}}$, is a subset of $\mathcal{D}_k$. Furthermore, the drivable areas of the optimal driving corridor DC^{O} at step k, represented by $\mathcal{D}_k^{\mathsf{O}}$, is a subset of $\mathcal{D}_k^{\mathsf{P}}$ since DC^{O} corresponds to a path in G^{P}. That is, $\mathcal{D}_k^{\mathsf{O}} \subseteq \mathcal{D}_k^{\mathsf{P}} \subseteq \mathcal{D}_k$. The non-empty drivable area $\mathcal{D}_k^{\mathsf{O}}$ implies that one may find a specification-compliant trajectory within the position and velocity bounds extracted from DC^{O}. We generate three sets of trajectories using the reactive planner under three settings:

- RP: the basic implementation of the reactive planner with fixed sampling intervals [77].
- RP+DC: enhances RP by drawing time, position, and velocity samples within DC^{O} as described in [14].
- RP+DC*: in addition to [14], enforces constraint (5c) and discards a trajectory if any of its states is outside DC^{O}.

Fig. 8d–e illustrate the sampled trajectories under different settings. While the trajectories sampled with RP covers both lanes and even off-road region, those planned with RP+DC/RP+DC* lie within the current lane of the ego vehicle.

Tab. III reports the computation results, among which we focus on the feasible trajectories and their compliance rate. In the baseline scenario, while around 40% of the trajectories planned with RP violate rule R-IN5 by entering the lane on the right, all trajectories considering DC^{O} comply with the rule. In both alternative scenarios, the compliance rate of the trajectories sampled with RP significantly decreases, with that drastically reduced to around 0.5% in variant 2. Although the compliance rate of RP+DC exhibits a milder drop than that of RP, it is less than ideal because not all states of the planned trajectories are entirely contained in DC^{O}. In contrast, RP+DC* performed consistently well in the given scenarios. Further enforcing a conflicting or non-satisfiable specification (e.g., $\mathbf{F}_{[0,5]}\big(\text{in\_lanelet}(L_2)\big)$, see Fig. 8a) would yield an empty product graph G^{P}; thus, no DC^{O} would be output and we can reject the specification before trying to plan a trajectory satisfying the specification.

We also compare the time required to obtain the first specification-compliant trajectory under different settings. Since the trajectory sampling and the feasibility check are shared among all three settings, we focus on generating a compliant trajectory from the feasible candidates. The computation is repeated for 50 times and the candidate trajectories are shuffled in each iteration. For RP+DC and RP+DC*, we also include the computation time of reachable sets. Fig. 9 depicts the computation results: RP required less (median) computation time than the other two settings in the baseline scenario and variant 1. This is justified by the fact that the enforced specifications in these two scenarios are relatively easy to be satisfied by the ego vehicle. With increased difficulty in variant 2, the computation time of RP grows remarkably (almost two orders of magnitude) since it struggles to find the few compliant trajectories among a large number of candidates. In contrast, the computation times of the settings adopting our reachable sets are consistent across the scenarios regardless of the considered specifications. In variant 2, the overhead of our reachable set computation is compensated by restricting the sampling space and, with RP+DC*, avoiding excessive compliance checks for sampled trajectories. Although RP+DC and RP+DC* performed similarly in these scenarios, adopting the latter allows us to explicitly constrain the sampled trajectories to the optimal driving corridor concerning user-defined utilities presented in Sec. VII-C.

TABLE III: Number of trajectories and compliance rate under different settings in scenario I: `ZAM_TIV-1_1_T-1`.

Method	#Sampled	#Feasible	#In Corridor	#Compliant	%Compliant
Baseline: R-IN5					
RP	13464	3839	-	2232	58.14 %
RP+DC	11288	4031	-	4031	100.00 %
RP+DC*	11288	4031	64	64	100.00 %
Variant 1: R-IN5 + $\mathbf{F}_{[0,k_h]}\big(\text{in\_lanelet}(L_2)\big)$					
RP	13464	3839	-	968	25.21 %
RP+DC	11288	4031	-	3418	84.79 %
RP+DC*	11288	4031	62	62	100.00 %
Variant 2: R-IN5 + $\mathbf{F}_{[5,12]}\big(\text{in\_lanelet}(L_2)\big)$					
RP	13464	3839	-	18	0.47 %
RP+DC	11288	4031	-	1021	25.33 %
RP+DC*	11288	4031	62	62	100.00 %

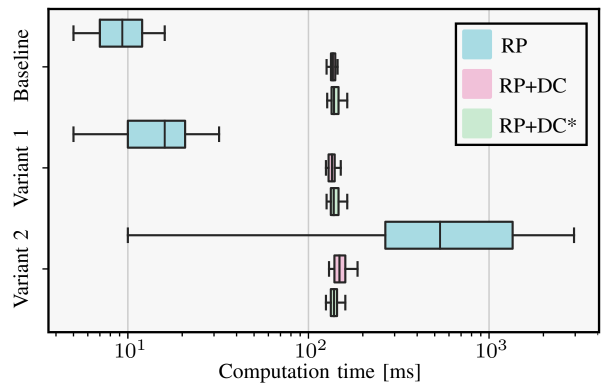
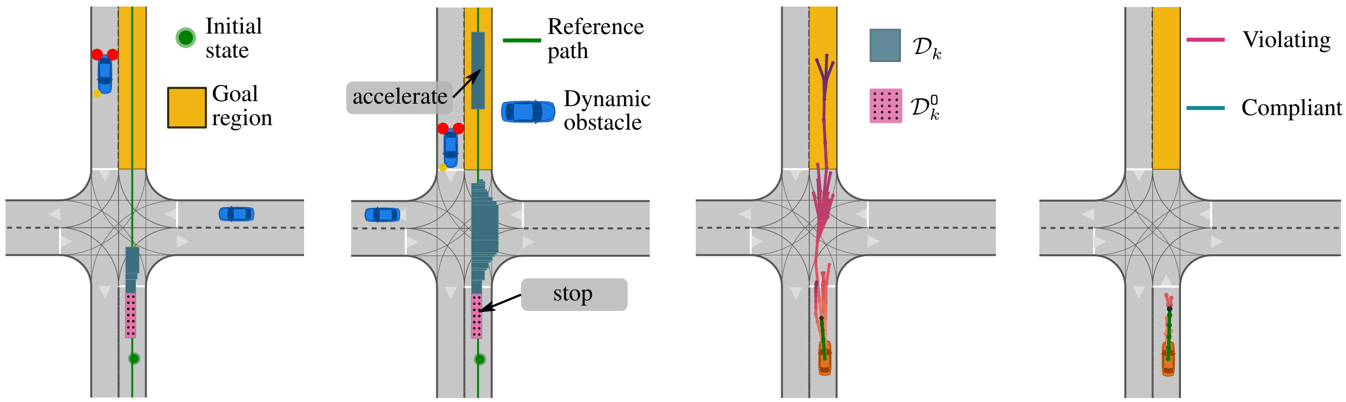


Fig. 9: Computation times in scenario I: `ZAM_TIV-1_1_T-1`. For better visibility, outliers of the box plot are not shown.

C. Scenario II: Four-way Intersection

Next, we consider a scenario in which the ego vehicle must come to a full stop before an intersection to respect passing priorities. Rule R-IN3 [6] dictates that the ego vehicle should not endanger another entering vehicle at an intersection if it is left of the other vehicle. Due to space limitations, we refer the reader to [6] for the MTL formulation of this rule. We also create an alternative scenario to increase the difficulty of the planning problem. Specifically, we alter the initial velocity of the ego vehicle from $7.0\,\mathrm{m/s}$ to $9.0\,\mathrm{m/s}$, which reduces the compliant drivable areas and state space.

Fig. 10a–b illustrate the drivable areas of the ego vehicle in the baseline scenario. The ego vehicle can, among other maneuvers, accelerate and pass through the intersection before the vehicle entering from the right or respect the passing priority and stop before the intersection. The optimal driving corridor DC^{O} is the only path in the product graph G^{P}, thus $\mathcal{D}_k^{\mathsf{P}} = \mathcal{D}_k^{\mathsf{O}}$. We demonstrate the benefits of our approach for RRT-based planners by comparing the following settings:



(a) Scenario at step $k = 7$ (b) Scenario at step $k = 15$ (c) Trajectories planned at step $k = 0$ without DC^0 (RRT*) (d) Trajectories planned at step $k = 0$ with DC^0 (RRT*+DC*)

Fig. 10: Drivable areas and planned trajectories for scenario II (benchmark ID: `ZAM_TIV-2_1_T-1`).

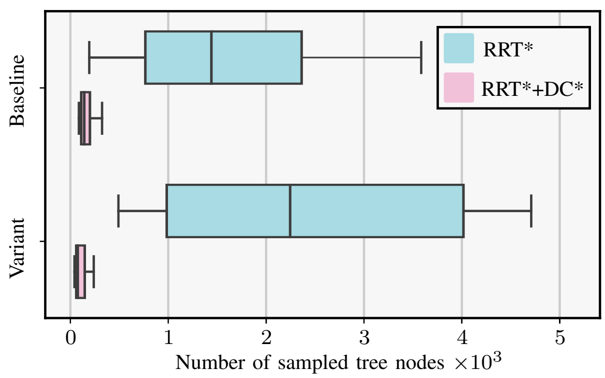


Fig. 11: Number of sampled tree nodes before finding a compliant trajectory in scenario II: `ZAM_TIV-2_1_T-1`. For better visibility, outliers of the box plot are not shown.

- RRT*: the basic implementation of RRT*. For a fairer comparison, we restrict the state and sample spaces to lanelets on the route leading to the goal region.
- RRT*+DC*: based on RRT*, we enforce constraint (5c) by restricting the state and sample spaces to DC^0.

Fig. 10c–d show exemplary explored trees under different settings. While the tree explored by RRT* spans to incoming and outgoing lanelets of the intersection, the tree explored by RRT*+DC* is, as expected, contained within the incoming lanelet before the intersection.

We compare the number of tree nodes required to generate a collision-free and specification-compliant trajectory lasting 3.0 seconds. To account for the stochastic nature of RRT*, we ran the planners for 50 times and present the results in Fig. 11. For both the baseline and variant scenarios, the median numbers of sampled tree nodes of RRT*+DC* are substantially lower than those of RRT*. This can be explained by the fact that only a fraction of the state and sample spaces are relevant for planning a trajectory satisfying rule R-IN3. Reducing the specification-compliant drivable areas noticeably increases the effort for planning a compliant trajectory by RRT*, which is not the case for RRT*+DC*. These observations are in line with our findings in Sec. VIII-B.

D. Scenarios with Decreasing Solution Spaces

It has been demonstrated in [12, Sec. VII-E] that the computation times of reachable sets are proportionally reduced with a decreasing solution space. We verify the validity of this finding on our reachable set computation by considering a cluttered scenario populated with vehicles and cyclists, see Fig. 12. To decrease the solution space, we gradually raise the initial velocity of the ego vehicle by $30\,\%$ at a time until a collision is unavoidable. This process increases the criticality of the scenario based on measures such as Time-to-Collision and Time-to-React, which can be evaluated using the CriMe toolbox [79]. We repeat the computations for 50 times and list the results in Tab. IV. Increasing the initial velocity of the ego vehicle leads to reduced numbers of base sets in the reachability graph and required set operations, resulting in lower mean computation times and smaller overall sizes of the drivable area cumulated over steps k. We observe that with the initial velocity raised to $310\,\%$, which yields the most critical scenario with inevitable collision using parameters in Tab. II, the computation time is exceptionally low, at approximately $4\,\mathrm{ms}$. The results thus confirm the favorable property of our reachable set that less computation time is required in more critical scenarios with smaller solution spaces.

E. Computation Time

The performance of our prototype is benchmarked by computing the reachable sets for over 50 randomly chosen scenarios from the CommonRoad benchmark suite. We only focus on position predicates concerning lanelets and vehicles as well as traffic situation predicates. The former causes frequent splitting of reachable sets and the latter requires relatively more effort in the annotation operation [43]. Fig. 13 illustrates the computation times of required operations in our reachable set computation as described in [43, Sec. III-D]. Our current implementation, with 75% of the computations executed within $250\,\mathrm{ms}$, requires only a fraction of the planning horizon, specifically $3.0\,\mathrm{s}$, thereby demonstrating its real-time capability. To further improve the performance of our prototype, adequate optimization and parallelized computation techniques can be employed. For instance, our observations

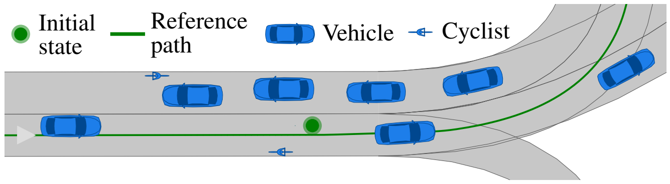


Fig. 12: A cluttered scenario with vehicles and cyclists at step $k = 0$ (benchmark ID: `ESP_Monzon-2_2_T-1`).

TABLE IV: Mean computation time, number of base sets, and size of the drivable area (in relative percentage) cumulated over steps k, when increasing the initial velocity in scenario `ESP_Monzon-2_2_T-1`.

Init. Vel.	%Vel.	Comput. Time	#Base Sets	%Drivable Area
4.69 m/s	100 %	139.51 ms	165	100.00 %
6.10 m/s	130 %	126.87 ms	160	99.33 %
7.51 m/s	160 %	124.47 ms	155	97.44 %
8.92 m/s	190 %	113.07 ms	148	96.06 %
10.32 m/s	220 %	98.10 ms	134	86.30 %
11.73 m/s	250 %	84.60 ms	118	71.67 %
13.14 m/s	280 %	58.30 ms	83	52.24 %
14.55 m/s	310 %	3.88 ms	5	0.13 %

from [76] suggest that translating the annotation operation from Python to C++ is expected to accelerate its computation by a factor of 20. For both scenarios I and II presented in the previous subsections, computing the product automaton $A^{\mathtt{P}}$ and determining the optimal driving corridor $DC^{\mathtt{O}}$ required only about $100\,\mu$s and $1\,$ms, respectively.

F. Comparison

While most of the works that we reviewed in Sec. I-A focus on reach-avoid problems with temporal requirements for robot navigation, to the best of our knowledge, article [19] is the only work that aims to achieve a goal similar to ours. Specifically, the article (a) focuses on constraint extraction for motion planning of automated vehicles, (b) considers compliance with specifications in temporal logic, and (c) handles dynamic obstacles. For this reason, we compare our approach to [19].

Let us recapitulate the approach presented in [19]: The authors first partition the collision-free state space and construct a so-called *navigation graph* $G^{\mathtt{N}}$, in which a node denotes a segment of a lanelet with a unique position relation concerning other vehicles. Connecting such nodes forms a path representing a timed *envelope*, i.e., position constraints, enclosing a set of homotopic trajectories. Next, to examine the compliance of these envelopes with traffic rules expressed in LTL, each path in $G^{\mathtt{N}}$ is individually verified using runtime verification. Finally, the authors assign heuristic costs to specification-compliant envelopes, from which the best solutions are output as constraints for trajectory planning.

Our approach outweighs [19] in the following two aspects:

1) Model Accuracy: Article [19] does not incorporate a vehicle model accounting for the dynamics of the ego vehicle and only constructs $G^{\mathtt{N}}$ at the sub-lanelet level. In contrast, our approach adopts a double-integrator point mass model (6), effectively capturing the ego vehicle's position, velocity, and acceleration components. While both approaches extract position constraints for the ego vehicle that comply with enforced specifications, our driving corridors additionally offer velocity

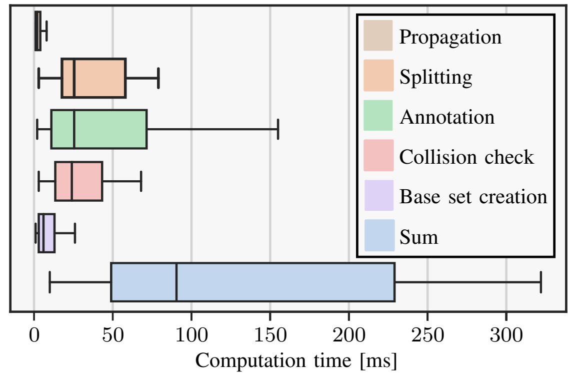


Fig. 13: Benchmarked computation times of our reachable set computation. For better visibility, outliers of the box plot are not shown.

constraints. This allows us to integrate specifications pertinent to the velocity of the ego vehicle (see Tab. I). Also, specifications on the accelerations can be handled directly during our computation of reachable sets through the modification of input bounds (7b).

In addition, our approach provides a less over-approximative abstraction of the ego vehicle. This can be substantiated by comparing the sizes of the discrete system models in the two approaches. For comparison, we consider a scenario (benchmark ID: `ZAM_TIV-3_1_T-1`) featuring three parallel lanelets, each containing two other vehicles. Due to the limitations of [19], only the position predicates relative to other vehicles are considered in the comparison. Using the setting described in Sec. VIII-A, our approach generates a component graph $G^{\mathtt{C}}$ comprised of nearly 180 nodes, which is significantly less than about 520 nodes in graph $G^{\mathtt{N}}$.

2) Verification Efficiency: Since the number of possible paths in graphs $G^{\mathtt{N}}$ and $G^{\mathtt{C}}$ grows exponentially in relation to the planning horizon k_h, even for the relatively simple scenario described in Sec. VIII-F1, graph $G^{\mathtt{N}}$ already contains about 250 billion paths to be monitored. This task is computationally demanding, if not intractable, for motion planning of automated vehicles with strict real-time requirements. Moreover, the task is incomplete unless all paths are examined. In stark contrast, our employment of automata-based model checking ensures that all paths in graph $G^{\mathtt{C}}$ are efficiently verified. At the same time, the computational complexity only increases linearly with the number of nodes in graph $G^{\mathtt{C}}$ [49], thereby demonstrating a far superior efficiency compared to runtime verification adopted by [19].

IX. Conclusions

Our novel approach offers a promising solution to the problem of specification-compliant motion planning for automated vehicles, paving the way to safer and more efficient road traffic. By coupling set-based reachability analysis with automata-based model checking, we identify collision-free and specification-compliant driving corridors of the ego vehicle. The driving corridors can be integrated into arbitrary motion planners accepting position and velocity constraints to expedite the generation of specification-compliant trajectories. In contrast to existing works, our approach realizes exhaustive verification of all possible driving corridors of the ego vehicle while accounting for its system dynamics and not sacrificing real-time capability. Moreover, the generation of a product graph enables detecting conflicting or non-satisfiable specifications before actually planning a trajectory. The experiments show that our approach can be easily integrated into motion planners to efficiently obtain trajectories complying with temporal specifications, especially when the solution space is increasingly small. Although our computation of reachable sets requires only a fraction of time of the planning horizon, as demonstrated with benchmarking over 50 CommonRoad scenarios, we will further improve the implementation so that it can achieve even better run time.

Acknowledgment

This work was funded by the German Research Foundation (DFG) under grant No. AL 1185/20-1, the German Federal Ministry for Education and Research (BMBF) under grant No. 03ZU1105KA (MCube), and Huawei Technologies under grant No. YBN2020035151. The authors also appreciate the fruitful collaboration with the project partners.

References

[1] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 740–759, 2020.

[2] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser, "A review of motion planning for highway autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 5, pp. 1826–1848, 2019.

[3] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, 2016.

[4] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1135–1145, 2015.

[5] Y. Sun, C. M. Poskitt, J. Sun, Y. Chen, and Z. Yang, "LawBreaker: An approach for specifying traffic laws and fuzzing autonomous vehicles," in *Profc. of the IEEE/ACM Int. Conf. Autom. Software Eng.*, 2022, pp. 1–12.

[6] S. Maierhofer, P. Moosbrugger, and M. Althoff, "Formalization of intersection traffic rules in temporal logic," in *Proc. of the IEEE Intell. Veh. Symp.*, 2022, pp. 1135–1144.

[7] S. Maierhofer, A.-K. Rettinger, E. C. Mayer, and M. Althoff, "Formalization of interstate traffic rules in temporal logic," in *Proc. of the IEEE Intell. Veh. Symp.*, 2020, pp. 752–759.

[8] K. Esterle, L. Gressenbuch, and A. Knoll, "Formalizing traffic rules for machine interpretability," in *Proc. of the IEEE Connect. Autom. Veh. Symp.*, 2020, pp. 1–7.

[9] A. Rizaldi, J. Keinholz, M. Huber, J. Feldle, F. Immler, M. Althoff, E. Hilgendorf, and T. Nipkow, "Formalising and monitoring traffic rules for autonomous vehicles in Isabelle/HOL," in *Int. Conf. Integr. Formal Methods*, 2017, pp. 50–66.

[10] E. Plaku and S. Karaman, "Motion planning with temporal-logic specifications: Progress and challenges," *AI Commun.*, vol. 29, no. 1, pp. 151–162, 2016.

[11] S. Söntges and M. Althoff, "Computing the drivable area of autonomous road vehicles in dynamic road scenes," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 6, pp. 1855–1866, 2018.

[12] S. Manzinger, C. Pek, and M. Althoff, "Using reachable sets for trajectory planning of automated vehicles," *IEEE Trans. Intell. Veh.*, vol. 6, no. 2, pp. 232–248, 2020.

[13] L. Schäfer, S. Manzinger, and M. Althoff, "Computation of solution spaces for optimization-based trajectory planning," *IEEE Trans. Intell. Veh.*, vol. 8, no. 1, pp. 216–231, 2021.

[14] G. Würsching and M. Althoff, "Sampling-based optimal trajectory generation for autonomous vehicles using reachable sets," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2021, pp. 828–835.

[15] H. Roehm, J. Oehlerking, T. Heinz, and M. Althoff, "STL model checking of continuous and hybrid systems," in *Int. Symp. Autom. Technol. Verif. Anal.*, 2016, pp. 412–427.

[16] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Tech., Model., Anal. Timed Fault-Tolerant Syst.*, 2004, pp. 152–166.

[17] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *Int. J. Rob. Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[18] R. R. Da Silva, V. Kurtz, and H. Lin, "Automatic trajectory synthesis for real-time temporal logic," *IEEE Trans. Autom. Control*, vol. 67, no. 2, pp. 780–794, 2021.

[19] K. Esterle, V. Aravantinos, and A. Knoll, "From specifications to behavior: Maneuver verification in a semantic state space," in *Proc. of the IEEE Intell. Veh. Symp.*, 2019, pp. 2140–2147.

[20] C. K. Verginis, C. Vrohidis, C. P. Bechlioulis, K. J. Kyriakopoulos, and D. V. Dimarogonas, "Reconfigurable motion planning and control in obstacle cluttered environments under timed temporal tasks," in *Proc. of the IEEE Int. Conf. Robot. Autom.*, 2019, pp. 951–957.

[21] Y. Shoukry, P. Nuzzo, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, "SMC: Satisfiability modulo convex programming," *Proc. of the IEEE*, vol. 106, no. 9, pp. 1655–1679, 2018.

[22] K. Cho, J. Suh, C. J. Tomlin, and S. Oh, "Cost-aware path planning under co-safe temporal logic specifications," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2308–2315, 2017.

[23] M. Lahijanian, M. R. Maly, D. Fried, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal planning in uncertain environments with partial satisfaction guarantees," *IEEE Trans. Rob.*, vol. 32, no. 3, pp. 583–599, 2016.

[24] Y. Zhou, D. Maity, and J. S. Baras, "Timed automata approach for motion planning using metric interval temporal logic," in *Proc. of the Eur. Control Conf.*, 2016, pp. 690–695.

[25] D. Maity and J. S. Baras, "Motion planning in dynamic environments with bounded time temporal logic specifications," in *Mediterr. Conf. Control Autom.*, 2015, pp. 940–946.

[26] R. Kohlhaas, T. Bittner, T. Schamm, and J. M. Zöllner, "Semantic state space for high-level maneuver planning in structured traffic scenes," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2014, pp. 1060–1065.

[27] E. M. Wolff, U. Topcu, and R. M. Murray, "Automaton-guided controller synthesis for nonlinear systems with temporal logic," in *Proc. of the IEEE Int. Conf. Intell. Robot. Syst.*, 2013, pp. 4332–4339.

[28] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-time Syst.*, vol. 2, no. 4, pp. 255–299, 1990.

[29] A. Pnueli, "The temporal logic of programs," in *Annu. Symp. Found. of Comput. Sci.*, 1977, pp. 46–57.

[30] M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, *50 years of integer programming 1958-2008: From the early years to the state-of-the-art*. Springer Science & Business Media, 2009.

[31] D. Sun, J. Chen, S. Mitra, and C. Fan, "Multi-agent motion planning from signal temporal logic specifications," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 3451–3458, 2022.

[32] Z. Lin and J. S. Baras, "Optimization-based motion planning and runtime monitoring for robotic agent with space and time tolerances," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1874–1879, 2020.

[33] U. A. Fiaz and J. S. Baras, "Fast, composable rescue mission planning for UAVs using metric temporal logic," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 15 404–15 411, 2020.

[34] S. Saha and A. A. Julius, "An MILP approach for real-time optimal controller synthesis with metric temporal logic specifications," in *Proc. of the Am. Control Conf.*, 2016, pp. 1105–1110.

[35] E. M. Wolff and R. M. Murray, "Optimal control of nonlinear systems with temporal logic specifications," in *Rob. Res.*, 2016, pp. 21–37.

[36] C. I. Vasile, X. Li, and C. Belta, "Reactive sampling-based path planning with temporal logic specifications," *Int. J. Rob. Res.*, vol. 39, no. 8, pp. 1002–1028, 2020.

[37] J. Karlsson, F. S. Barbosa, and J. Tumova, "Sampling-based motion planning with temporal logic missions and spatial preferences," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 15 537–15 543, 2020.

[38] F. S. Barbosa, L. Lindemann, D. V. Dimarogonas, and J. Tumova, "Integrated motion planning and control under metric interval temporal logic specifications," in *Eur. Control Conf.*, 2019, pp. 2042–2049.

[39] C.-I. Vasile, V. Raman, and S. Karaman, "Sampling-based synthesis of maximally-satisfying controllers for temporal logic specifications," in *Proc. of the IEEE Int. Conf. Intell. Robot. Syst.*, 2017, pp. 3840–3847.

[40] L. I. R. Castro, P. Chaudhari, J. Tůmová, S. Karaman, E. Frazzoli, and D. Rus, "Incremental sampling-based algorithm for minimum-violation

motion planning," in *Proc. of the IEEE Conf. Decis. Control*. IEEE, 2013, pp. 3217–3224.

[41] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Rob. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[42] L. Zhang and D. Manocha, "An efficient retraction-based RRT planner," in *Proc. of the IEEE Int. Conf. Robot. Autom.* IEEE, 2008, pp. 3743–3750.

[43] E. Irani Liu and M. Althoff, "Computing specification-compliant reachable sets for motion planning of automated vehicles," in *Proc. of the IEEE Intell. Veh. Symp.*, 2021, pp. 1037–1044.

[44] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Proc. of the IEEE Intell. Veh. Symp.*, 2014, pp. 420–425.

[45] G. De Giacomo, A. Murano, F. Patrizi, and G. Perelli, "Timed trace alignment with metric temporal logic over finite traces," in *Proc. of the Int. Conf. Principles Knowl. Represent. and Reasoning*, vol. 18, no. 1, 2021, pp. 227–236.

[46] D. DSouza and P. Prabhakar, "On the expressiveness of MTL in the pointwise and continuous semantics," *Int. J. Software Tools for Technol. Transfer*, vol. 9, no. 1, pp. 1–4, 2007.

[47] A. Cecconi, C. D. Ciccio, G. D. Giacomo, and J. Mendling, "Interestingness of traces in declarative process mining: The Janus LTLp$_f$ approach," in *Int. Conf. Bus. Process Manage.* Springer, 2018, pp. 121–138.

[48] J. Ouaknine and J. Worrell, "Some recent results in metric temporal logic," in *Int. Conf. Formal Model. and Anal. Timed Syst.* Springer, 2008, pp. 1–13.

[49] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[50] A. Duret-Lutz, A. Lewkowicz, A. Fauchille, T. Michaud, E. Renault, and L. Xu, "Spot 2.0 – A framework for LTL and ω-automata manipulation," in *Int. Symp. Autom. Technol. Verif. Anal.*, 2016, pp. 122–129.

[51] G. Lafferriere, G. J. Pappas, and S. Yovine, "Symbolic reachability computation for families of linear vector fields," *J. Symb. Comput.*, vol. 32, no. 3, pp. 231–253, 2001.

[52] G. Holzmann, "Explicit-state model checking," in *Handbook of model checking*, 2018, pp. 153–170.

[53] O. Kupferman, "Automata theory and model checking," in *Handbook of model checking*, 2018, pp. 107–151.

[54] H. Tauriainen and K. Heljanko, "Testing LTL formula translation into Büchi automata," *Int. J. Software Tools for Technol. Transfer*, vol. 4, no. 1, pp. 57–70, 2002.

[55] P. Gastin and D. Oddoux, "Fast LTL to Büchi automata translation," in *Int. Conf. Comput. Aided Verif.* Springer, 2001, pp. 53–65.

[56] J. R. Büchi, "On a decision method in restricted second order arithmetic," in *Proc. of the Int. Congr. Logic, Method. and Philos. Sci.*, 1962, pp. 1–11.

[57] H. T. Croft, K. Falconer, and R. K. Guy, *Unsolved problems in geometry: Unsolved problems in intuitive mathematics*. Springer Science & Business Media, 2012, vol. 2.

[58] Y. Lin and M. Althoff, "Rule-compliant trajectory repairing using satisfiability modulo theories," in *Proc. of the IEEE Intell. Veh. Symp.*, 2022, pp. 449–456.

[59] C. Pek and M. Althoff, "Fail-safe motion planning for online verification of autonomous vehicles using convex optimization," *IEEE Trans. Rob.*, vol. 37, no. 3, pp. 798–814, 2020.

[60] B. Schürmann, D. Heß, J. Eilbrecht, O. Stursberg, F. Köster, and M. Althoff, "Ensuring drivability of planned motions using formal methods," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2017, pp. 1–8.

[61] M. Althoff and J. M. Dolan, "Reachability computation of low-order models for the safety verification of high-order road vehicle models," in *Proc. of the Am. Control Conf.*, 2012, pp. 3559–3566.

[62] J. Eilbrecht and O. Stursberg, "Challenges of trajectory planning with integrator models on curved roads," in *Proc. of the IFAC World Congr.*, 2020, pp. 15 588–15 595.

[63] C. Pek, V. Rusinov, S. Manzinger, M. C. Üste, and M. Althoff, "CommonRoad Drivability Checker: Simplifying the development and validation of motion planning algorithms," in *Proc. of the IEEE Intell. Veh. Symp.*, 2020, pp. 1013–1020.

[64] S. A. Kripke, "A completeness theorem in modal logic," *J. Symb. Log.*, vol. 24, no. 1, pp. 1–14, 1959.

[65] D. Gabbay, "The declarative past and imperative future," in *Temporal Log. Specification*, 1989, pp. 409–448.

[66] G. P. Maretić, M. T. Dashti, and D. Basin, "Anchored LTL separation," in *Proc. of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2014, pp. 1–9.

[67] I. M. Hodkinson and M. Reynolds, "Separation – past, present, and future." in *We Will Show Them! (2)*, 2005, pp. 117–142.

[68] N. Markey, "Temporal logic with past is exponentially more succinct," *Eur. Assoc. Theor. Comput. Sci.*, vol. 79, pp. 122–128, 2003.

[69] G. Roşu and K. Havelund, "Rewriting-based techniques for runtime verification," *Autom. Software Eng.*, vol. 12, no. 2, pp. 151–197, 2005.

[70] M. Reynolds, "More past glories," in *Proc. of the IEEE Symp. Logic. Comput. Sci.*, 2000, pp. 229–240.

[71] S. Dutta and M. Y. Vardi, "Assertion-based flow monitoring of SystemC models," in *Proc. of the ACM/IEEE Int. Conf. Formal Methods and Models Co-Des.*, 2014, pp. 145–154.

[72] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *Proc. of the Int. Joint Conf. Artif. Intell.* Association for Computing Machinery, 2013, pp. 854–860.

[73] A. Censi, K. Slutsky, T. Wongpiromsarn, D. Yershov, S. Pendleton, J. Fu, and E. Frazzoli, "Liability, ethics, and culture-aware behavior specification using rulebooks," in *Proc. of the IEEE Int. Conf. Robot. Autom.*, 2019, pp. 8536–8542.

[74] E. L. Lawler, *Combinatorial optimization: Networks and matroids*. Courier Corporation, 2001.

[75] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intell. Veh. Symp.*, 2017, pp. 719–726.

[76] E. Irani Liu, G. Würsching, M. Klischat, and M. Althoff, "CommonRoad-Reach: A toolbox for reachability analysis of automated vehicles," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2022, pp. 2313–2320.

[77] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenet frame," in *Proc. of the IEEE Int. Conf. Robot. Autom.*, 2010, pp. 987–993.

[78] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *Am. J. Math.*, vol. 79, no. 3, pp. 497–516, 1957.

[79] Y. Lin and M. Althoff, "CommonRoad-CriMe: A toolbox for criticality measures of autonomous vehicles," in *Proc. of the IEEE Intell. Veh. Symp.*, 2023, pp. 1–8.

Edmond Irani Liu is currently a Ph.D. candidate and joined the Cyber-Physical Systems Group at the Technical University of Munich under Prof. Dr.-Ing. Matthias Althoff in 2019. He received his B.Sc. degree in automation in 2015 and his M.Sc. degree in control science and engineering in 2018, both from Shanghai Jiao Tong University, China. His research interests include specification-compliant reachability analysis and motion planning of automated vehicles.

Matthias Althoff received the diploma engineering degree in mechanical engineering and the Ph.D. degree in electrical engineering from the Technical University of Munich, Germany, in 2005 and 2010, respectively. He is currently an associate professor in computer science with the Technical University of Munich. From 2010 to 2012 he was a postdoctoral researcher with Carnegie Mellon University, Pittsburgh, USA, and from 2012 to 2013 an assistant professor at Ilmenau Technical University, Germany. His research interests include formal verification of continuous and hybrid systems, reachability analysis, planning algorithms, nonlinear control, automated vehicles, and power systems.