# Computing Specification-Compliant Reachable Sets for Motion Planning of Automated Vehicles

Edmond Irani Liu and Matthias Althoff

***Abstract*— To safely and effectively participate in road traffic, automated vehicles should explicitly consider compliance with traffic rules and high-level specifications. We propose a method that can incorporate traffic and handcrafted rules expressed in time-labeled propositional logic into our reachability analysis, which computes the over-approximative set of states reachable by vehicles. These reachable sets serve as low-level trajectory planning constraints to expedite the search for specification-compliant trajectories. Depending on the adopted specifications, related semantic labels are generated from predicates considering positions, velocities, accelerations, and general traffic situations. We exhibit the applicability of the proposed method with scenarios from the CommonRoad benchmark suite.**

## I. Introduction

Highly automated vehicles (AVs) promise increased road safety compared with human-driven ones. To safely and effectively participate in road traffic, AVs should explicitly consider compliance with traffic and handcrafted rules. Compliance with the former exempts manufacturers from potential liability claims in case an accident happens, whereas the latter contribute to finding motion plans that meet specific requirements.

Determining a drivable trajectory that satisfies a desired discrete specification involves reasoning with both discrete and continuous states of AV, which poses computational challenges originating from (a) vehicle dynamics and collision avoidance, (b) discrete specifications, and (c) interwoven dependencies between continuous trajectories and discrete constraints. Planning on the discrete level may output plans that meet the specifications but do not satisfy dynamic constraints; similarly, motion planning methods may generate collision-free and dynamically feasible trajectories that violate the specifications.

In this study, we address these challenges by extending our previous work [1] to compute specification-compliant reachable sets for a considered ego vehicle. Our over-approximative reachable sets enclose all drivable trajectories of AV and can be used as low-level trajectory planning constraints [2], which expedite the search for specification-compliant trajectories. Depending on the adopted specifications expressed in time-labeled propositional logic, relevant semantic labels are attached to the reachable sets. An exemplary handcrafted rule can be described as "follow vehicle A up to time step $k_1$, then finish overtaking it from the left before time step $k_2$."

All authors are with the Department of Informatics, Technical University of Munich, 85748 Garching, Germany.
{edmond.irani,althoff}@tum.de

The efforts to obtain a specification-compliant trajectory can be roughly categorized into three groups. The first group generates discrete plans to guide the trajectory planning process. For example, Shoukry *et al.* [3] proposed a satisfiability modulo convex programming framework to handle Boolean and convex constraints; Lahijanian *et al.* [4] proposed a multilayered synergistic framework, which is an extension of [5] to cope with newly discovered obstacles. Zhou *et al.* [6] used timed automata to synthesize timed paths that satisfy considered specifications. In these methods, the high-level planners suggest discrete plans based on an abstraction of a system, and the low-level motion planners generate trajectories that comply with the discrete plans. Since the continuous constraints are not explicitly considered by the high-level planners, the drivability of the suggested plans is often not ensured [7], thereby requiring frequent replanning on the discrete level. The second group evaluates the specifications on the planned trajectories with monitors [8], [9]. The monitoring can be efficiently performed; however, in case the trajectory under examination is rejected by the monitor, no alternative candidate trajectory is returned. The third group, to which the present study belongs, considers the specifications before trajectory planning, e.g., in high-level maneuver planners [10]–[13]. Kohlhaas *et al.* [13] generated maneuvers with simple traffic rules by traversing a graph represented in a semantic state space; Esterle *et al.* [10] adopted a similar idea and generated maneuvers that complied with linear temporal logic (LTL) specifications. In a previous work [12], we output so-called driving corridors that satisfied sequences of position relations to other vehicles.

Contrary to these works, we not only consider specifications with position relations to other obstacles but also include predicates considering velocities, accelerations, and general traffic situations based on the recent formalization of German road traffic regulations in temporal logic [14], [15]. In [14], traffic rules related to velocities, overtaking, safe distances, priorities, etc, were formalized using LTL. In comparison, metric temporal logic (MTL), which is an extension of LTL to support time intervals, was used in [15] to formalize a selection of traffic rules related to general and interstate driving situations. The proposed method stands out in that it

1) integrates predicates into reachable set computation for compliance with traffic rules formalized in [15];
2) can incorporate handcrafted specifications with the above-mentioned predicates; and
3) can be combined with any motion planning algorithms.

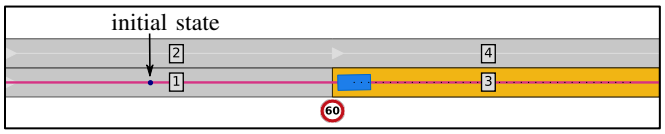



Fig. 1: A scenario with four lanelets (with IDs 1–4 shown with numbered boxes) and a leading vehicle (blue). Lanelet 3 is the goal lanelet (yellow) of an ego vehicle. The straight line (magenta) indicates a reference path leading to the goal lanelet.

The remainder of this article is organized as follows: Sec. II introduces basic definitions and necessary preliminaries. Subsequently, we present the partitioning of the state space and the computation of reachable sets in Sec. III. Afterward, we demonstrate evaluations with exemplary scenarios from the CommonRoad[1] benchmark suite [16] in Sec. IV. Finally, we conclude in Sec. V.

## II. Preliminaries

### A. System Description

In this study, the considered scenarios are described in the CommonRoad format. A typical CommonRoad scenario (see Fig. 1) consists of (a) a road network represented by lanelets [17], whose left and right bounds are modeled with polylines, (b) static and dynamic obstacles, and (c) traffic rule elements, such as traffic signs, traffic lights, and road markings. We use the trajectories provided in the scenarios and consider them as adequate prediction for obstacles over time. Alternatively, one can adopt a set-based prediction [18] for obstacles. Given a planning problem, which includes the initial state of an ego vehicle and a set of goal states, a reference path to the goal lanelet is planned. This path is used to construct the local curvilinear coordinate system of the ego vehicle, as described in [17]. The choice of this coordinate system facilitates the formulation of maneuvers from the ego vehicle's perspective, e.g., lane-following, stopping before an intersection, and preventing driving backward.

The system dynamics of the ego vehicle is abstracted by a point-mass model with the center of the vehicle as the reference point. The model is represented with two second-order integrators in longitudinal $s$-direction and lateral $d$-direction of the curvilinear coordinate system. Let $\square$ be a variable, we denote by $\underline{\square}$ and $\overline{\square}$ its minimum and maximum values, respectively. In addition, we attach subscripts $s/d$ to variables to indicate the directions in which they are described. The system dynamics of the ego vehicle is

$$x_{k+1} = \begin{pmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{pmatrix} x_k + \begin{pmatrix} \frac{1}{2}\Delta t^2 & 0 \\ \Delta t & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ 0 & \Delta t \end{pmatrix} u_k, \quad (1)$$

where $k \in \mathbb{N}_0$ is a discrete time step corresponding to a point in time $t_k = k\Delta t$, with $\Delta t \in \mathbb{R}_+$ being a predefined time increment. $x \in \mathcal{X} \subset \mathbb{R}^4$ is a state in the state space $\mathcal{X}$, and $u \in \mathcal{U} \subset \mathbb{R}^2$ is an input. The state of the vehicle is modeled as $x = (p_s, v_s, p_d, v_d)^\mathsf{T}$, with $p$ and $v$ representing the position and velocity, respectively. The system accepts inputs $u = (a_s, a_d)^\mathsf{T}$, where $a$ is the acceleration. The velocities and accelerations in both directions are bounded by the over-approximation of the physically feasible values:

$$\underline{v}_s \le v_{s,k} \le \overline{v}_s, \ \underline{v}_d \le v_{d,k} \le \overline{v}_d, \quad (2a)$$
$$\underline{a}_s \le a_{s,k} \le \overline{a}_s, \ \underline{a}_d \le a_{d,k} \le \overline{a}_d. \quad (2b)$$

The bounds are chosen conservatively to consider the kinematic limitations and effects arising from transforming the system dynamics to the curvilinear coordinate system. Notably, the drivability of the planned trajectories should be examined individually, e.g., using the drivability checker described in [19]. Finally, we denote the length of the ego vehicle by $l$.

### B. Reachable Sets

Given an initial state $x_0$ and an input trajectory $u_{[0,k]}$, we use $\chi_k(x_0, u_{[0,k]})$ to represent the solution to (1) at time step $k$. We assume a set of time-dependent obstacles to be given, the union of whose occupancies at time step $k$ is represented by $\mathcal{O}_k \subset \mathbb{R}^2$. The sets of states whose occupancy (considering the shape of the ego vehicle) are overlapping with $\mathcal{O}_k$ are removed from the reachable sets. Let $\mathcal{X}_k^{\mathtt{CF}} = \mathcal{X} \setminus \mathcal{O}_k$ be the set of collision-free states at time step $k$, the exact collision-free reachable set of the ego vehicle at $k$ starting from the initial set of states $\mathcal{X}_0$ is

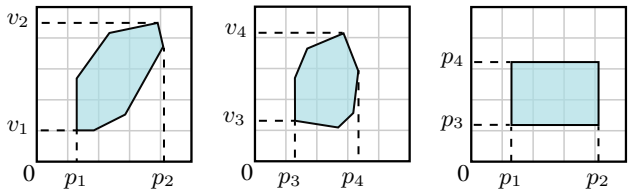
$$\mathcal{R}_k^*(\mathcal{X}_0) := \Big\{ \chi_k(x_0, u_{[0,k]}) \Big| x_0 \in \mathcal{X}_0, \forall \tau \in \{0, \dots, k\} : u_\tau \in \mathcal{U}, \chi_\tau(x_0, u_{[0,\tau]}) \in \mathcal{X}_\tau^{\mathtt{CF}} \Big\}.$$

Subsequently, we omit $\mathcal{X}_0$ for convenience. Obtaining $\mathcal{R}_k^*$ is computationally demanding in general; therefore, we instead compute its over-approximation $\mathcal{R}_k$, which is the union of so-called base sets $\mathcal{R}_k^{(i)}$, $i \in \mathbb{N}$. Each base set $\mathcal{R}_k^{(i)} = \hat{\mathcal{P}}_{s,k}^{(i)} \times \hat{\mathcal{P}}_{d,k}^{(i)}$ is chosen to be a Cartesian product of two convex polytopes $\hat{\mathcal{P}}_{s,k}^{(i)}$ and $\hat{\mathcal{P}}_{d,k}^{(i)}$ which represent the reachable positions and velocities in $(p_s, v_s)$ and $(p_d, v_d)$ planes, respectively (see Fig. 2a–b) [1]. This choice is motivated by the existence of efficient algorithms for required set operations on convex polytopes. To simplify the notation, we also denote the collection of $\mathcal{R}_k^{(i)}$ by $\mathcal{R}_k$, i.e., $\mathcal{R}_k = \{\mathcal{R}_k^{(1)}, \dots, \mathcal{R}_k^{(i)}\}$. The projection of $\mathcal{R}_k^{(i)}$ onto the position domain yields axis-aligned rectangles $\mathcal{D}_k^{(i)}$ (see Fig. 2c), whose union is referred to as the drivable area $\mathcal{D}_k$. Similarly, we use $\mathcal{D}_k$ to denote the collection of $\mathcal{D}_k^{(i)}$.

**Definition 1 (Projection):**
*The operator* $\mathrm{proj}_\Diamond(\cdot)$ *maps the input to its elements* $\Diamond$*. For example,* $\mathrm{proj}_{(p,v)}(x) = (p, v)^\mathsf{T}$ *for* $x = (p, v, a)^\mathsf{T}$*. A set* $\mathcal{X}$ *can be projected using the same notation:* $\mathrm{proj}_\Diamond(\mathcal{X}) = \{\mathrm{proj}_\Diamond(x) | x \in \mathcal{X}\}$.

**Definition 2 (Drivable Area):**
*The drivable area is defined as the projection of the reachable set onto the position domain:* $\mathcal{D}_k := \mathrm{proj}_{(p_s,p_d)}(\mathcal{R}_k)$.

[1] https://commonroad.in.tum.de/




(a) $\hat{\mathcal{P}}^{(i)}_{s,k}$ (b) $\hat{\mathcal{P}}^{(i)}_{d,k}$ (c) $\mathcal{D}^{(i)}_{k}$

Fig. 2: Polytopes and drivable area of a base set $\mathcal{R}^{(i)}_k$.

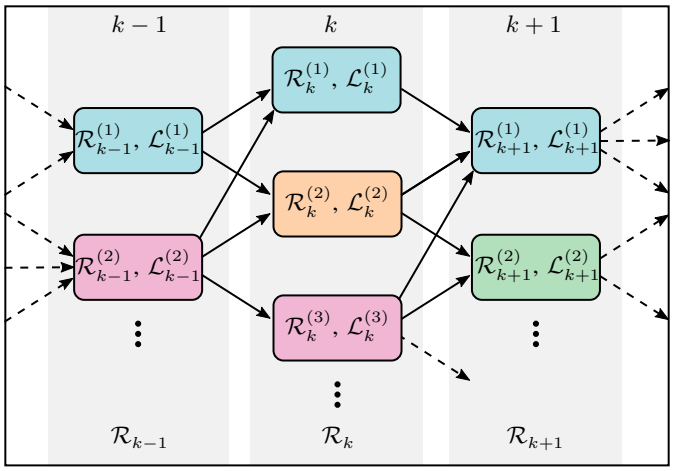



Fig. 3: Reachability graph $\mathcal{G}_{\mathcal{R}}$ holding nodes of different time steps. The nodes with the same labels have the same color.

In this study, each base set $\mathcal{R}^{(i)}_k$ additionally carries a set $\mathcal{L}^{(i)}_k$ of semantic labels, whose collection is denoted by $\mathcal{L}_k$. The generation of these labels will be explained in Sec. III-D.3. Let us introduce the reachability graph $\mathcal{G}_{\mathcal{R}}$, which is a directed graph connecting base sets $\mathcal{R}^{(i)}_k$ so that their temporal and spatial relationships can be queried (see Fig. 3). In $\mathcal{G}_{\mathcal{R}}$, each node represents exactly one base set $\mathcal{R}^{(i)}_k$. An edge connecting $\mathcal{R}^{(i)}_k$ and $\mathcal{R}^{(j)}_{k+1}$ indicates that $\mathcal{R}^{(j)}_{k+1}$ is reachable from $\mathcal{R}^{(i)}_k$. A base set $\mathcal{R}^{(i)}_k$ may reach several base sets $\mathcal{R}^{(j)}_{k+1}$ in the next time step.

## III. Methodology

To obtain specification-compliant reachable sets, we should (a) semantically label reachable sets with relevant predicates and (b) constrain reachable sets to subsets satisfying the desired specifications. We partition the state space based on position predicates to expedite the labeling process. Similar strategies have been employed in [10], [13], [20]. We do not consider velocity predicates at this stage since it requires the computationally demanding splitting of the state space with (non)linear curves (see Fig. 5c–d). Instead, we directly evaluate them on individual reachable sets (detailed in Sec. III-D.2). The selection of considered predicates is listed in Tab. I: Evaluating a *dynamic* predicate is obstacle-dependent, whereas that of a *static* predicate is not.

### A. Formulation of Partitions

Computing the partitions of the state space involves operations such as set intersection and difference. To avoid gross approximations while maintaining the computational complexity at an acceptable level, they are modeled with a set of hyperrectangles $\texttt{rect}_q$. Notably, such a choice is not mandatory, any other representation that captures the partitions suffices. Each $\texttt{rect}_q$ is defined as a Cartesian product of intervals over the position and velocity domains:

$$\begin{aligned}\texttt{rect}_q := &\big([\underline{p}_{q,s}, \overline{p}_{q,s}] \times [\underline{v}_{q,s}, \overline{v}_{q,s}]\big) \times \\ &\big([\underline{p}_{q,d}, \overline{p}_{q,d}] \times [\underline{v}_{q,d}, \overline{v}_{q,d}]\big),\end{aligned} \tag{3}$$

where $p_{q,s}$ and $v_{q,s}$ denote the position and velocity of the $q$-th hyperrectangle in the $s$-direction, respectively. The same applies to $p_{q,d}$ and $v_{q,d}$ in the $d$-direction. A regular grid of pairwise-disjoint axis-aligned cells is formed along the reference path. Let $\mathcal{C}^{(q)} = [\underline{p}_{q,s}, \overline{p}_{q,s}] \times [\underline{p}_{q,d}, \overline{p}_{q,d}] \subset \mathbb{R}^2$ be the $q$-th cell in the grid. By computing the Cartesian product of $\mathcal{C}^{(q)}$ and velocity intervals $[\underline{v}_{q,s}, \overline{v}_{q,s}]$ and $[\underline{v}_{q,d}, \overline{v}_{q,d}]$, a hyperrectangle $\texttt{rect}_q$ can be created. By default, the velocity intervals $[\underline{v}_s, \overline{v}_s]$ and $[\underline{v}_d, \overline{v}_d]$ (see (2a)) are used.

Let $\mathcal{P} = \{\texttt{pred}_1, \texttt{pred}_2, \dots\}$ be the set of considered position predicates, with its power set denoted by $2^{\mathcal{P}}$. We denote by $\mathrm{part}(k; \mathcal{Z}_j)$ the set of hyperrectangles for which the predicates in $\mathcal{Z}_j \in 2^{\mathcal{P}}$ evaluate to $\texttt{True}$ at time step $k \in \mathbb{N}_0$. $\mathcal{Z}_j$ is realizable if $\exists k \in \{0, \dots, k_h\} : \mathrm{part}(k; \mathcal{Z}_j) \neq \varnothing$, with $k_h$ being a predefined planning horizon. Fig. 5b illustrates exemplary partitions projected onto the $(p_s, v_s)$ plane. We aim to obtain the collection $\mathcal{Z} \subseteq 2^{\mathcal{P}}$ of all realizable $\mathcal{Z}_j$ considering relevant lanelets and obstacles:

$$\mathcal{Z} = \Big\{ \mathcal{Z}_j \in 2^{\mathcal{P}} \Big| \exists k \in \{0, \dots, k_h\} : \mathrm{part}(k; \mathcal{Z}_j\}) \neq \varnothing \Big\}.$$

$\mathcal{Z}$ is used for splitting of reachable sets (see Sec. III-D.2).

### B. Evaluation of Position Predicates

*1) Static Position Predicates:* These predicates do not depend on obstacles. We formulate two examples as follows:

- **InLanelet**$(k; \texttt{rect}_q, L_{\texttt{id}})$: $\texttt{True}$ if $\texttt{rect}_q$ is within the lanelet with ID $\texttt{id}$, denoted by $L_{\texttt{id}}$, at time step $k$.
- **DrivesRightmost**$(k; \texttt{rect}_q, \texttt{area})$: $\texttt{True}$ if $\texttt{rect}_q$ intersects with the rightmost area of lanelets, denoted by $\texttt{area}$. The distance between any point within this area to the right bound of the lanelet does not exceed a predefined distance.

We use $\mathcal{Z}^{\mathrm{L}} \subseteq 2^{\mathcal{P}}$ to denote the power set of considered static position predicates. For the sake of brevity, we only keep the lanelets and obstacles (explained later) in the arguments of predicates in the rest of this work.

*2) Dynamic Position Predicates:* These predicates reflect position relationships between the ego vehicle and obstacles. In this study, we use vehicles as examples of obstacles. Let $\mathcal{V} = \{V_1, \dots, V_N\}$ be the set of other vehicles with IDs $\mathcal{N} = \{1, \dots, N\}$. In addition, let $\mathrm{occ}(k; V_n)$ return the occupancy of $V_n$ at time step $k$, with bounds in the $s$-direction denoted by $\underline{p}_{n,s,k}$ and $\overline{p}_{n,s,k}$, respectively. Along the $s$-direction, the mutually exclusive predicates $\mathcal{P}_{n,s} = \big\{\{\textbf{InFrontOf}(V_n)\}, \{\textbf{Behind}(V_n)\}, \{\textbf{Beside}(V_n)\}\big\}$ can be evaluated on $\texttt{rect}_q$ with respect to $V_n$ at time step $k$ as follows:

TABLE I: The selection of considered predicates inspired by [15].

| Category | Type | Predicate | Source/Inspiration |
|---|---|---|---|
| **Position** | Static | **InLanelet**, **DrivesRightmost**, **OnMainCarriageWay**, **OnAccessRamp**, ... | R_I2, R_I4 |
| | Dynamic | **Behind**, **Beside**, **InFrontOf**, **LeftOf**, **AlignedWith**, **RightOf**, ... | R_G1, R_I2 |
| **Velocity** | Static | **BelowFOVVLimit**, **BelowTypeVLimit**, **AboveMinimumVLimit**, ... | R_G3, R_I1 |
| | Dynamic | **SafeFollowingVelocity(To)**, **SafeLeadingVelocity(To)**, **DrivesFaster**, ... | R_G1, R_I2 |
| **Acceleration** | Static | **AdmissibleBraking**, ... | R_G2 |
| **General** | Static | **ChangeLanelet**, **PreservesTrafficFlow**, **StandingStill**, ... | R_G4, R_I1 |
| | Dynamic | **InCongestion**, **SlowLeadingVehicle**, ... | R_G1, R_G4, R_I1 |

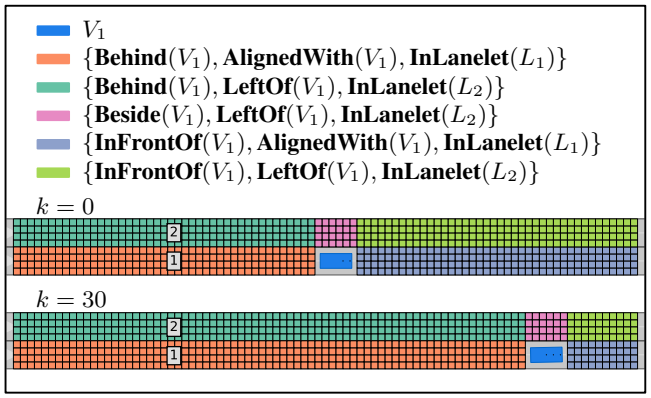



Fig. 4: Projection of the partitions of $\mathcal{Z}_j \in \mathcal{Z}$ onto the position domain. Lanelet IDs are shown with numbered boxes.

- **InFrontOf**($V_n$): `True` if $\underline{p}_{q,s} - l/2 > \overline{p}_{n,s,k}$.
- **Behind**($V_n$): `True` if $\overline{p}_{q,s} + l/2 < \underline{p}_{n,s,k}$.
- **Beside**($V_n$): `True` if $\neg$**InFrontOf**($V_n$) $\wedge \neg$**Behind**($V_n$) $\wedge \operatorname{proj}_{(p_s,p_d)}(\texttt{rect}_q) \cap \operatorname{occ}(k; V_n) = \varnothing$.

Analogously, along the $d$-direction, the mutually exclusive set of predicates $\mathcal{P}_{n,d} = \big\{\{$**LeftOf**$(V_n)\}, \{$**RightOf**$(V_n)\}, \{$**AlignedWith**$(V_n)\}\big\}$ can be evaluated on $\texttt{rect}_q$.

### C. Realizable Sets of Position Predicates

The operator $\operatorname{product}(\cdot)$ over $\tilde{n}$ collections $\mathcal{A}_1, \dots, \mathcal{A}_{\tilde{n}}$ is defined as

$$\operatorname{product}(\mathcal{A}_1, \dots, \mathcal{A}_{\tilde{n}}) = \big\{\mathcal{A}'_1 \cup \dots \cup \mathcal{A}'_{\tilde{n}} \big| \; \mathcal{A}'_i \in \mathcal{A}_i, i \in \{1, \dots, \tilde{n}\}\big\}.$$

As an example, in the presence of a vehicle $V_1$, $\operatorname{product}(\mathcal{P}_{1,s}, \mathcal{P}_{1,d}) = \big\{\{$**InFrontOf**$(V_1), $**LeftOf**$(V_1)\}, \dots, \{$**Beside**$(V_1), $**AlignedWith**$(V_1)\}\big\}$ (see Sec. III-B.2). We denote by $\mathcal{Z}_n^{\mathtt{V}}$ the collection of realizable sets of position predicates created with respect to $V_n$ that can be projected onto the curvilinear coordinate system of the ego vehicle. The formulation of $\mathcal{Z}_n^{\mathtt{V}}$ and $\mathcal{Z}$ (see Sec. III-A) are presented in Alg. 1: we iteratively examine all possible combinations of predicates and keep the ones that have a nonempty partition for at least one time step within the planning horizon $k_h$. Fig. 4 illustrates the projection of the partitions of $\mathcal{Z}_j \in \mathcal{Z}$ created for an exemplary scenario onto the position domain at two time steps. Owing to our formulation of the predicates, the mentioned projection is collision-free with respect to the obstacles and are pairwise-disjoint at any specific time step.

**Algorithm 1** Realizable Sets of Position Predicates

**Inputs:** Set $\mathcal{N}$ of IDs of vehicles, planning horizon $k_h$, Collection $\mathcal{Z}^{\mathtt{L}}$
**Output:** Collection $\mathcal{Z}$

1: **for** $n$ in $\mathcal{N}$ **do** ▷ Formulation of $\mathcal{Z}_n^{\mathtt{V}}$
2: $\quad\mathcal{Z}_n^{\mathtt{V}} \leftarrow \{\}$
3: $\quad\mathcal{P}_{n,s}, \mathcal{P}_{n,d} \leftarrow$ ObtainPredicates($n$) ▷ see Sec. III-B.2
4: $\quad$**for** $\mathcal{Z}_i$ in product($\mathcal{P}_{n,s}, \mathcal{P}_{n,d}$) **do**
5: $\quad\quad$**if** HasNonEmptyPartition($\mathcal{Z}_i$) **then**
6: $\quad\quad\quad\mathcal{Z}_n^{\mathtt{V}}$.Add($\mathcal{Z}_i$)
7: $\quad\quad$**end if**
8: $\quad$**end for**
9: **end for**
10:
11: $\mathcal{Z} \leftarrow \mathcal{Z}^{\mathtt{L}}$ ▷ Formulation of $\mathcal{Z}$
12: **for** $n$ in $\mathcal{N}$ **do**
13: $\quad\mathcal{Z}' \leftarrow \{\}$
14: $\quad$**for** $\mathcal{Z}_i$ in product($\mathcal{Z}, \mathcal{Z}_n^{\mathtt{V}}$) **do**
15: $\quad\quad$**if** HasNonEmptyPartition($\mathcal{Z}_i$) **then**
16: $\quad\quad\quad\mathcal{Z}'$.Add($\mathcal{Z}_i$)
17: $\quad\quad$**end if**
18: $\quad$**end for**
19: $\quad\mathcal{Z} \leftarrow \mathcal{Z}'$
20: **end for**
21: **return** $\mathcal{Z}$
22:
23: **function** HasNonEmptyPartition($\mathcal{Z}_i$)
24: $\quad$**for** $k = 0$ to $k_h$ **do**
25: $\quad\quad$**if** part($k; \mathcal{Z}_i$) $\neq \varnothing$ **then**
26: $\quad\quad\quad$**return** True
27: $\quad\quad$**end if**
28: $\quad$**end for**
29: $\quad$**return** False
30: **end function**

### D. Computation of Reachable Sets

In addition to our previous works [1], [12], we semantically label the reachable sets and constrain them to states satisfying the desired specifications (Alg. 2 lines 4–6). As the reachable sets are computed iteratively over time, it suffices to give a detailed explanation for one step of the computation.

*1) Propagation of Base Sets (Alg. 2, line 3):* Each base set $\mathcal{R}_{k-1}^{(i)} \in \mathcal{R}_{k-1}$ of the previous time step is propagated according to the system model (1), resulting in the propagated sets $\mathcal{R}_k^{\mathtt{P},(i)} \in \mathcal{R}_k^{\mathtt{P}}$ (see Fig. 5a). The propagation is performed similarly to the method described in [1], except that one can additionally impose acceleration constraints. As an example, rule R_G2 [15] describes situations in which braking abruptly is allowed, i.e., braking harder than a predefined value $a_{\texttt{def}} >$

**Algorithm 2** Computation of Reachable Sets

**Inputs:** Specification $\phi$, base sets $\mathcal{R}_0^{(i)} \in \mathcal{R}_0$ with their semantic labels $\mathcal{L}_0^{(i)} \in \mathcal{L}_0$, planning horizon $k_h$, realizable sets of position predicates $\mathcal{Z}$.

**Output:** Reachability graph $\mathcal{G}_\mathcal{R}$.

1: $\mathcal{G}_\mathcal{R}$.ADD($\mathcal{R}_0, \mathcal{L}_0$)
2: **for** $k = 1$ to $k_h$ **do**
3: $\quad\mathcal{R}_k^{\mathtt{P}} \leftarrow$ PROPAGATE($\mathcal{R}_{k-1}$) ▷ Sec. III-D.1
4: $\quad\mathcal{R}_k^{\mathtt{S}} \leftarrow$ SPLITTING($\mathcal{R}_k^{\mathtt{P}}, \mathcal{Z}$) ▷ Sec. III-D.2
5: $\quad\mathcal{R}_k^{\mathtt{L}}, \mathcal{L}_k \leftarrow$ LABELING($\mathcal{R}_k^{\mathtt{S}}, \phi$) ▷ Sec. III-D.3
6: $\quad\mathcal{R}_k^{\mathtt{C}} \leftarrow$ CHECKCOMPLIANCE($\mathcal{R}_k^{\mathtt{L}}, \mathcal{L}_k, \phi$) ▷ Sec. III-D.4
7: $\quad\mathcal{R}_k \leftarrow$ CREATENEWBASESETS($\mathcal{R}_k^{\mathtt{C}}$) ▷ Sec. III-D.5
8: $\quad\mathcal{G}_\mathcal{R}$.ADD($\mathcal{R}_k, \mathcal{L}_k$)
9: **end for**
10: **return** $\mathcal{G}_\mathcal{R}$

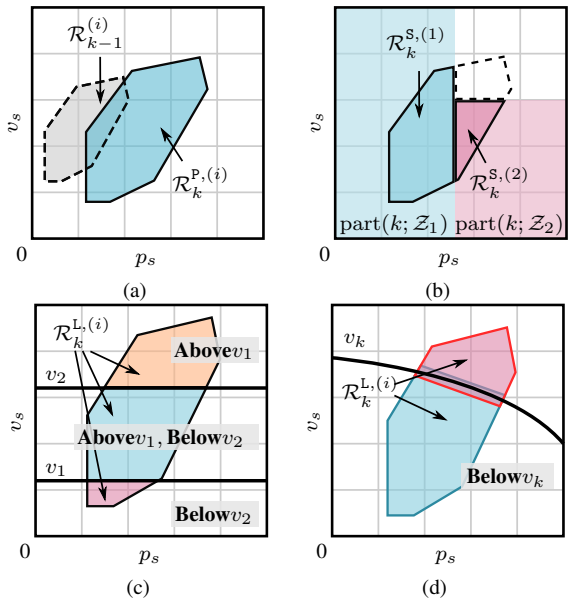



Fig. 5: Propagation, splitting, and labeling of base sets. For clarity, we only show the operations in the $s$-direction. Labels of polytopes are shown in gray boxes. (a) Propagation. (b) Splitting regarding partitions. (c) Splitting regarding static velocity predicates: three polytopes split with predicates **Above**$v_1$ and **Below**$v_2$. (d) Splitting regarding dynamic velocity predicates: two polytopes split with a predicate **Below**$v_k$. Notably, the two newly split polytopes are slightly over-approximated and convexified due to the nonlinearity introduced by the predicate.

$\underline{a}_s$. If this rule is considered, we only propagate the base sets with the acceleration interval $[\underline{a}_s, \overline{a}_s]$ under the specified situations, otherwise $[a_{\mathtt{def}}, \overline{a}_s]$. Sets $\mathcal{L}_k^{(i)}$ of propagated sets $\mathcal{R}_k^{\mathtt{P},(i)}$ are initialized with empty sets.

*2) Splitting (Alg. 2, line 4):* The propagated sets $\mathcal{R}_k^{\mathtt{P},(i)}$ are split into new sets $\mathcal{R}_k^{\mathtt{S},(i)} \in \mathcal{R}_k^{\mathtt{S}}$ regarding position and velocity predicates. We first introduce the velocity predicates.

*a) Static Velocity Predicates:* These predicates typically relate to the constant extremum requirements on velocities. Rule R_G3 [15] requires that maximum velocity limits originating from various sources to be respected. These include limits introduced due to the limited field of view of the ego vehicle and vehicle type-specific limits.

*b) Dynamic Velocity Predicates:* These predicates depend on other dynamic obstacles present in the scenario. Examples are predicates indicating whether the ego vehicle is driving at a safe velocity regarding a leading or a following vehicle $V_n$ [15, cf. Sec. IV.C].

The splitting of $\mathcal{R}_k^{\mathtt{P},(i)}$ is performed as follows:

1) $\mathcal{R}_k^{\mathtt{P},(i)}$ are split such that the newly split sets intersect only with a single partition of $\mathcal{Z}$ (see Fig. 5b).
2) The split sets are further split, over-approximated and convexified (if needed) regarding velocity predicates (see Fig. 5c–d).

*3) Semantic Labeling (Alg. 2, line 5):* Next, we evaluate the general traffic situation predicates introduced by the specification (see details in Sec. III-D.4) on $\mathcal{R}_k^{\mathtt{S},(i)}$, and update their semantic labels $\mathcal{L}_k^{(i)}$. These predicates may reveal whether the ego vehicle has conducted a lanelet-change maneuver (see Sec. IV-A) and if it is stuck in a traffic congestion, etc. The labels $\mathcal{L}_k^{(i)}$ are updated as follows:

1) Sets $\mathcal{R}_k^{\mathtt{S},(i)}$ propagated with acceleration-specific rules include corresponding predicates in their sets of semantic labels $\mathcal{L}_k^{(i)}$.
2) Sets $\mathcal{R}_k^{\mathtt{S},(i)}$ include the position predicates associated with the partition with which it intersects, velocity predicates, and general traffic situation predicates that evaluate to `True` in their sets of semantic labels $\mathcal{L}_k^{(i)}$.

The sets with updated $\mathcal{L}_k^{(i)}$ are denoted by $\mathcal{R}_k^{\mathtt{L},(i)} \in \mathcal{R}_k^{\mathtt{L}}$.

*4) Checking Specification Compliance (Alg. 2, line 6):* In this step, we iterate through $\mathcal{R}_k^{\mathtt{L},(i)}$ and examine the compliance of their labels $\mathcal{L}_k^{(i)}$ with the given specification. Let $\sigma$ be an atomic proposition, a time-labeled propositional formula $\phi$ is defined in Backus-Naur form as:

$$\phi ::= \sigma \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid \mathrm{G}_I(\phi),$$

where the operator $\mathrm{G}_I$ dictates a time interval $I$ for which $\phi$ should hold. If $I$ is unspecified, we assume it to be the entire planning horizon $[0, k_h]$. Let $\mathcal{L}_k^{(i)}$ be the set of labels to be examined, its compliance with $\sigma$ is defined as: $\mathcal{L}_k^{(i)} \models \sigma$ iff $\sigma \in \mathcal{L}_k^{(i)}$. As an example, the following specification enforces the ego vehicle to follow $V_1$ between time steps 0 and 10, and never to be on its right for the entire planning horizon:

$$\mathrm{G}_{[0,10]}(\textbf{Behind}(V_1) \wedge \textbf{AlignedWith}(V_1)) \wedge \mathrm{G}(\neg\textbf{RightOf}(V_1)).$$

We discard $\mathcal{R}_k^{\mathtt{L},(i)}$ whose set of semantic labels $\mathcal{L}_k^{(i)}$ do not comply with $\phi$, and refer to the remaining sets as $\mathcal{R}_k^{\mathtt{C},(i)} \in \mathcal{R}_k^{\mathtt{C}}$. Recall that the reachable sets enclose all drivable trajectories of the vehicle, thus an empty set $\mathcal{R}_k^{\mathtt{C}}$ implies that $\phi$ is unsatisfiable and cannot be complied with by any possible trajectory of the ego vehicle. In such a case, one can either recompute the reachable sets with a different specification, or trigger previously computed fail-safe trajectories [21]. Integrating MTL specifications into our reachable sets using model checkers will be a future study.

*5) Creation of New Base Sets (Alg. 2, line 7):* Finally, the new base sets $\mathcal{R}_k^{(i)} \in \mathcal{R}_k$ are created from the nonempty sets $\mathcal{R}_k^{\mathtt{C},(i)}$. The substeps include obtaining the drivable areas $\mathcal{D}_k^{\mathtt{C},(i)}$ of $\mathcal{R}_k^{\mathtt{C},(i)}$, merging and repartitioning $\mathcal{D}_k^{\mathtt{C},(i)}$, and ultimately producing the sets $\mathcal{R}_k^{(i)}$. These are performed similarly to the description in [1, Alg. 1] with one difference: to preserve the set of position predicates at the merging step, only $\mathcal{D}_k^{\mathtt{C},(i)}$ projected from sets $\mathcal{R}_k^{\mathtt{C},(i)}$ with the same partition are merged. The reachability graph $\mathcal{G}_{\mathcal{R}}$ is updated in the end by inserting sets $\mathcal{R}_k^{(i)}$ along with their labels $\mathcal{L}_k^{(i)}$ as new nodes.

## IV. Evaluation

In this section, we exhibit the applicability of our method using varied specifications on three scenarios. Selected parameters and computation results are listed in Tab. II. The animation of the evaluation can be found at https://mediatum.ub.tum.de/1595757. Before presenting the evaluation results, we introduce relevant labels and explain how selected traffic rule elements can be incorporated.

### A. Relevant Labels

- **AdmissibleBraking**: This indicates that the rule R_G2 is considered (see Sec. III-D.1).
- **SafeFollowingVelocity**, **SafeLeadingVelocity**: These indicate that the ego vehicle has respected safe following/leading velocities to other dynamic obstacles, respectively (see Sec. III-D.2).
- **ChangeLanelet**$(L_1, L_2)$: This indicates that the ego vehicle has performed a lanelet-change maneuver from $L_i$ to $L_j$.

### B. Incorporating Traffic Rule Elements

*1) Prohibiting Change of Lanelet:* Assuming a case where the ego vehicle is not allowed to change from $L_1$ to $L_2$, which may be imposed by different traffic rule elements, such as road markings, no-overtaking signs, and traffic lights. We model this with $\mathrm{G}_I(\neg$**ChangeLanelet**$(L_1, L_2))$.

*2) Lanelet-specific Velocity Limits:* Lanelet-specific velocity limits can neither be modeled as static nor dynamic predicates (as described in Sec. III-D.2). Recalling that the partitions of sets of position predicates are modeled with hyperrectangles $\mathtt{rect}_q$, we adjust the velocity intervals $[\underline{v}_{q,s}, \overline{v}_{q,s}]$ of $\mathtt{rect}_q$ in the lanelets to incorporate these velocity limits.

### C. Scenario I: Precise Overtaking

The first scenario illustrates a situation where the ego vehicle should overtake a leading vehicle $V_1$ in the presence of another vehicle $V_2$. The following specification is issued, for example, by a high-level maneuver planner, which should be precisely followed by the ego vehicle:

$$\mathrm{G}_{[0,15]}(\textbf{Behind}(V_1) \wedge \textbf{AlignedWith}(V_1)) \wedge$$
$$\mathrm{G}_{[16,38]}(\textbf{InLanelet}(L_2) \vee \textbf{InLanelet}(L_4)) \wedge$$
$$\mathrm{G}_{[39,45]}(\textbf{InFrontOf}(V_1) \wedge \textbf{Behind}(V_2) \wedge \textbf{InLanelet}(L_3)) \wedge$$
$$\mathrm{G}(\textbf{AdmissibleBraking}).$$

TABLE II: Selected Parameters and Computation Results

| Description | Unit | I | | II | | III | |
|---|---|---|---|---|---|---|---|
| **Parameter** | | | | | | | |
| $k_h$ | step | 45 | 45 | 40 | 40 | 40 | 40 |
| $\Delta t$ | s | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| $\overline{v}_s$ | $\mathrm{m/s}$ | 16.6 | 16.6 | 16.6 | 16.6 | 16.6 | 16.6 |
| $\underline{v}_s$ | $\mathrm{m/s}$ | 0 | 0 | 0 | 0 | 0 | 0 |
| $v_{s,0}$ | $\mathrm{m/s}$ | 12.0 | 12.0 | 14.5 | 14.5 | 14.0 | 14.0 |
| $\overline{v}_d$ | $\mathrm{m/s}$ | 4.0 | 4.0 | 4.0 | 4.0 | 4.0 | 4.0 |
| $\underline{v}_d$ | $\mathrm{m/s}$ | -4.0 | -4.0 | -4.0 | -4.0 | -4.0 | -4.0 |
| $\overline{a}_s$ | $\mathrm{m/s^2}$ | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 |
| $\underline{a}_s$ | $\mathrm{m/s^2}$ | -6.0 | -6.0 | -6.0 | -6.0 | -6.0 | -6.0 |
| $a_{\mathtt{def}}$ | $\mathrm{m/s^2}$ | -2.0 | -2.0 | -2.0 | -2.0 | -2.0 | -2.0 |
| Grid cell size | $\mathrm{m}^2$ | 1×1 | 2×2 | 1×1 | 2×2 | 1×1 | 2×2 |
| **State Space Partition** | | | | | | | |
| $\lvert\mathcal{Z}\rvert$ | - | 14 | 14 | 14 | 14 | 52 | 52 |
| # Hyperrectangles | - | 1800 | 450 | 1800 | 450 | 1700 | 430 |
| **Computation Time** | | | | | | | |
| Propagation | ms | 360 | 131 | 434 | 213 | 288 | 174 |
| Splitting | ms | 439 | 138 | 715 | 296 | 1360 | 532 |
| Labeling | ms | 25 | 9 | 95 | 20 | 85 | 37 |
| Compliance check | ms | 92 | 31 | 123 | 56 | 268 | 112 |
| Creation of new base sets | ms | 16 | 8 | 30 | 17 | 43 | 22 |
| Sum | ms | 932 | 317 | 1397 | 602 | 2044 | 877 |

We compute the reachable sets over time as explained in the previous section. The resulting reachable sets are nonempty, thereby implying that one may find a trajectory that satisfies thet specification. Fig. 6 visualizes the drivable areas of the ego vehicle at different time steps, as well as an exemplary trajectory planned within the reachable sets [2]. Changing **Behind**$(V_2)$ to **InFrontOf**$(V_2)$ for $k \in [39, 45]$ in the specification yields an empty reachable set; thus, we can reject it before trying to plan a trajectory that satisfies the specification.

### D. Scenario II: Respecting Safe Velocities

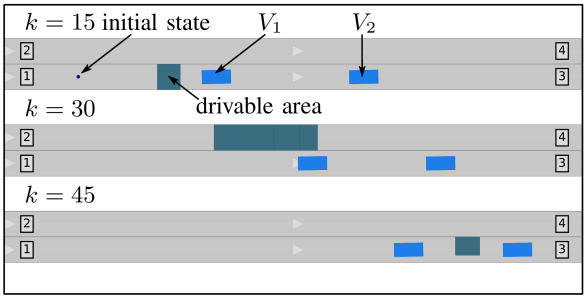
In this scenario, the ego vehicle is driving on the left side of two other vehicles $V_1$ and $V_2$, and wishes to change to lanelets on the right (1 and 3). The unrestricted reachable sets (see Fig. 7a) show that the ego vehicle can reach lanelet 1 at time step $k = 20$ and deliberately cut in between $V_1$ and $V_2$ for $k \in [30, 40]$. By contrast, after considering the specification with safe velocity rules

$$\mathrm{G}(\textbf{SafeFollowingVelocity} \wedge$$
$$\textbf{ChangeLanelet}(\cdot,\cdot) \rightarrow \textbf{SafeLeadingVelocity}),$$

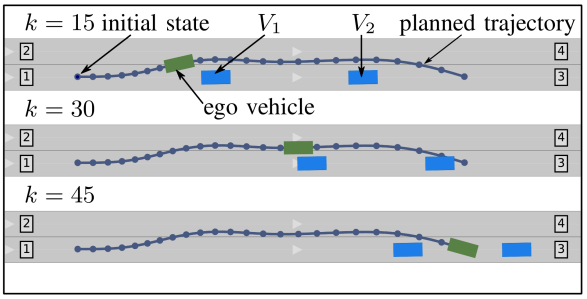
being in lanelet 1 is no longer legal at time step $k = 20$, so is the case with being between $V_1$ and $V_2$ for $k \in [30, 40]$ (see Fig. 7b). Following $V_1$ in lanelets 1 and 3 is still a legal maneuver at later time steps, provided the ego vehicle has sufficiently slowed down to respect the safe following velocity rule.

### E. Scenario III: Overtaking from the Left

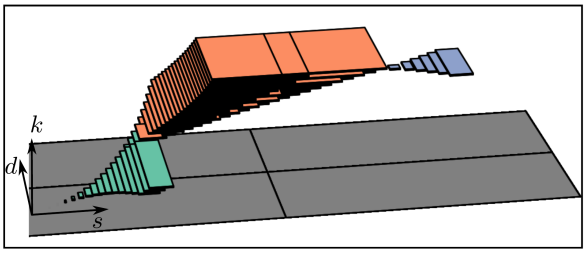
Next, we present a scenario in which the ego vehicle intends to overtake its preceding vehicle $V_1$. The unrestricted reachable sets propagate to all lanelets in the scenario and enclose various maneuvers, including overtaking $V_1$ from the right (see Fig. 8a). By imposing the constraint that the ego




(a)




(b)




(c)

Fig. 6: Scenario I: Precise overtaking. (a) Drivable area at different time steps. (b) Exemplary trajectory planned within the reachable set. (c) Drivable area over time. Each color corresponds to a clause in the specification whose time interval is specified.

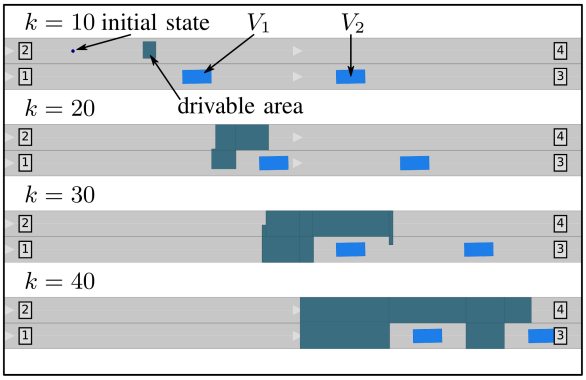
vehicle should overtake $V_1$ at some future time steps, and not being on its right for the entire planning horizon, we explicitly demand that the ego vehicle can only overtake $V_1$ from its left. In addition, we add a constraint to forbid entering lanelet 4 to further restrict the reachable sets:

$$\mathrm{G}_{[35,45]}(\textbf{InFrontOf}(V_1) \wedge \textbf{AlignedWith}(V_1)) \wedge$$
$$\mathrm{G}(\neg\textbf{RightOf}(V_1) \wedge \neg\textbf{ChangeLanelet}(L_3, L_4)).$$

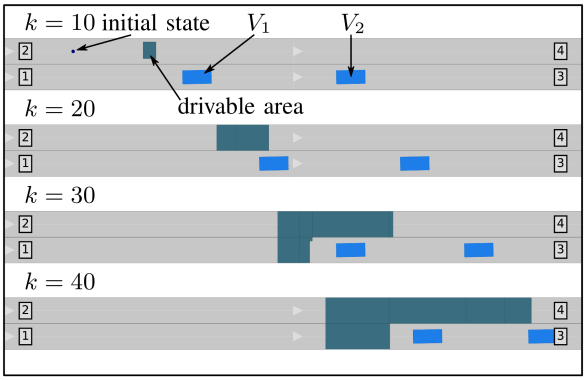
As can be seen from Fig. 8b, the reachable sets flow to lanelet 3 from the left of $V_1$ at $k = 20$, return back to lanelet 2 at $k = 30$, and finally end in front of $V_1$ at $k = 40$, which is exactly required by the specification.

### *F. Analysis of Computation Results*

The computation times listed in Tab. II are obtained through a prototype implemented using Python and C++ on a 2.8 GHz laptop. The computation times for operations, including splitting, labeling, and compliance checking, are linear to the number of nodes in the reachability graph, which is, in turn, proportional to the number of hyperrectangles: For all three scenarios, the evaluations were performed using




(a)



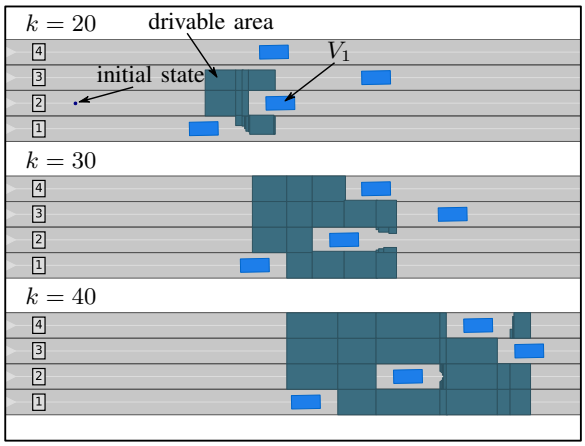

(b)

Fig. 7: Scenario II: Respecting safe velocities to other vehicles. (a) Not considering specifications. (b) Considering specifications.

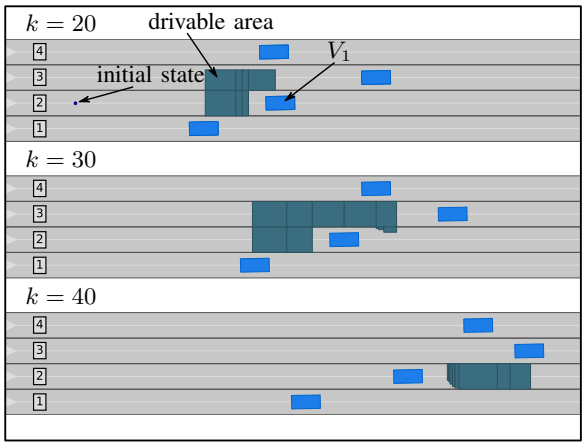
two grid cell sizes. By doubling the cell sizes and thereby reducing the number of hyperrectangles, we observed a drastic decrease (approximately 60%) in the computation time, which could be further improved by representing the partitions with more sophisticated polytopes. We refer to [1, Appendix B] for a detailed explanation of the computation complexity of propagation and creation of base sets. The method for accelerating the reachable set computation in [22] did not consider dynamic velocity and acceleration constraints; how this method can be coupled with our reachability analysis should be investigated in the future.

## V. Conclusions

In this study, we proposed a method to obtain specification-compliant reachable sets for a considered ego vehicle, which is used to guide motion planners to find specification-compliant trajectories. Compared with existing methods, the proposed method can not only consider traffic and handcrafted rules considering position predicates but also velocity, acceleration, and general traffic situation predicates. The evaluations showed that our method could easily incorporate desired specifications as well as identify and reject the unsatisfiable ones. In the future, we will investigate how to fully incorporate traffic rules formulated in MTL formulas into our reachable sets. MTL formulas are much more expressive than propositional logic and LTL by having temporal operators over time, such as $\textbf{Once}_I$ and




(a)




(b)

Fig. 8: Scenario III: Overtaking from the left. (a) Not considering specifications. (b) Considering specifications.

**Future**$_I$. In addition, it is also worth investigating that how specifications should be manipulated when they cannot be fully complied with, e.g., when a collision cannot be avoided without making a prohibited lane change.

## Acknowledgments

Special thanks to Stefanie Manzinger for her support to this work. This work was funded by the Deutsche Forschungsgemeinschaft (German Research Foundation) within the Priority Program SPP 1835 *Cooperative Interacting Automobiles* under grant No. AL 1185/4-2, and within the Huawei-TUM collaboration project *Research on Key Technologies of Safety Assurance for Autonomous Vehicles*. The authors appreciate the fruitful collaboration with the project partners.

## References

[1] 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.

[2] S. Manzinger, C. Pek, and M. Althoff, "Using reachable sets for trajectory planning of automated vehicles," *IEEE Trans. Intell. Veh.*, 2020 (to appear).

[3] 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.

[4] 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. Robot.*, vol. 32, no. 3, pp. 583–599, 2016.

[5] A. Bhatia, M. R. Maly, L. E. Kavraki, and M. Y. Vardi, "Motion planning with complex goals," *IEEE Robot. Autom. Mag.*, vol. 18, no. 3, pp. 55–64, 2011.

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

[7] E. Plaku and S. Karaman, "Motion planning with temporal-logic specifications: progress and challenges," *AI Commun.*, vol. 29, no. 1, pp. 151–162, 2015.

[8] M. Hekmatnejad, S. Yaghoubi, A. Dokhanchi, H. B. Amor, A. Shrivastava, L. Karam, and G. Fainekos, "Encoding and monitoring responsibility sensitive safety rules for automated vehicles in signal temporal logic," in *Proc. of the ACM/IEEE Int. Conf. Formal Method. Model. Syst. Des.*, 2019, pp. 1–11.

[9] A. Rizaldi, F. Immler, and M. Althoff, "A formally verified checker of the safe distance traffic rules for autonomous vehicles," in *NASA Formal Method. Symp.*, 2016, pp. 175–190.

[10] 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.

[11] A. Best, S. Narang, D. Barber, and D. Manocha, "Autonovi: autonomous vehicle planning with dynamic maneuvers and traffic constraints," in *Proc. of the IEEE Int. Conf. Intell. Robot. Syst.*, 2017, pp. 2629–2636.

[12] S. Söntges and M. Althoff, "Computing possible driving corridors for automated vehicles," in *Proc. of the IEEE Intell. Veh. Symp.*, 2017, pp. 160–166.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] M. Koschi and M. Althoff, "Set-based prediction of traffic participants considering occlusions and traffic rules," *IEEE Trans. Intell. Veh.*, 2020 (to appear).

[19] 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.

[20] F. Altché and A. De La Fortelle, "Partitioning of the free space-time for on-road navigation of autonomous ground vehicles," in *Proc. of the IEEE Conf. Decis. Control*, 2017, pp. 2126–2133.

[21] C. Pek and M. Althoff, "Fail-safe motion planning for online verification of autonomous vehicles using convex optimization," *IEEE Trans. Robot.*, 2020 (to appear).

[22] M. Klischat and M. Althoff, "A multi-step approach to accelerate the computation of reachable sets for road vehicles," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2020, pp. 1–7.