Lexicographic Mixed-Integer Motion Planning with STL Constraints

Patrick Halder[1], Fabian Christ[1], and Matthias Althoff[2]

***Abstract*— Autonomous vehicles are subject to various constraints, such as following the rules of the road (ROTR), adhering to schedules, or providing a comfortable driving experience. However, realizing a driving behavior complying with all constraints is challenging since it is not always possible to satisfy them simultaneously, necessitating the formulation of compromises. In this paper, we propose a solution to this challenge by decomposing the specification of an autonomous vehicle into a rulebook utilized by a novel optimization-based minimum-violation motion planner. In particular, our planner uses reachable sets to prevent collisions with other road users, and it minimally violates the ROTR formalized in signal temporal logic (STL). Furthermore, a mixed-integer convex program (MICP) realization of the planner is provided to demonstrate its effectiveness, especially in dynamically changing environments. We evaluate our approach using realistic ROTR on 1780 scenarios from the CommonRoad benchmark suite. Our results show that our planner generates safe and feasible trajectories, indicating its potential for real-world applications.**

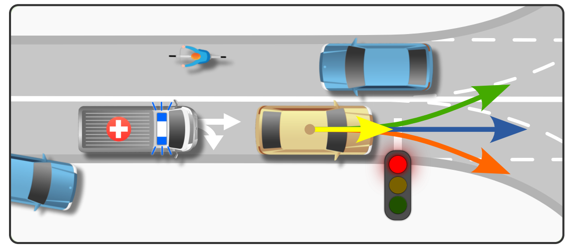


Fig. 1: An example scenario for an inevitable constraint violation: By deliberately crossing the red traffic light, the autonomous vehicle can prevent blocking the emergency vehicle on duty.

I. Introduction

Autonomous vehicles face many constraints, including preventing collisions with other traffic participants, adhering to the rules of the road (ROTR), performing mission tasks, such as stopping at scheduled bus stops, and ensuring high driving comfort for passengers. Nevertheless, the simultaneous fulfillment of these constraints poses a significant challenge, given their number and the fact that their satisfaction is also affected by the behavior of other traffic participants. When constraints contradict each other, this is typically resolved by specifying their respective level of importance [1]. Fig. 1 shows an example where ROTR must be violated. In this paper, we explore a problem formulation, where the contradiction of constraints is resolved by defining a ranking.

A. Related Work

We review related works on the handling of contradicting constraints, reachability analysis, and the formalization of spatio-temporal constraints. Further related work beyond our survey can be found in [2].

Handling of Contradicting Constraints: In motion planning problems, contradicting constraints are typically expressed as a collection of hard and (potentially weighted) soft constraints [3]. To avoid infeasible motion planning problems, one requires a significant effort in designing and tuning the constraints. Hierarchical approaches are often proposed to circumvent these issues, where constraints are prioritized (e.g., see [4]–[7]). Rulebooks [8], [9] gained recent popularity in this regard, as they emphasize the paradigm of *ranking of principles over outcomes* [10]. This implies that the system behavior should be specified by fundamental prioritized principles rather than based on the suitability of specific maneuvers in particular scenarios. Under certain prerequisites, rulebooks induce a lexicographic order on a set of potential system trajectories. Lexicographic preferences provide a clear and comprehensive approach for handling contradicting constraints, ensuring that the most important constraints are satisfied first [1], [11]. Rulebooks can be effectively applied to planning tasks, as demonstrated in [4], [12]. The approach of synthesizing motion plans by violating lower-prioritized constraints to satisfy higher-prioritized constraints is commonly referred to as minimum-violation planning [13]–[15].

Reachability Analysis: A promising technique for ensuring road safety and preventing collisions with road users involves the usage of reachable sets, which can bind all physically feasible and collision-free trajectories of autonomous vehicles [16]. When combined with set-based prediction, reachable sets enable provable safe motion planning [17]. A trajectory planned inside a reachable set is guaranteed to be collision-free (subject to certain assumptions, such as reachset conformance [18]). Methods for planning trajectories in reachable sets include convex optimization [19] and sampling-based techniques [20]. This work utilizes reachable sets within a mixed-integer convex program (MICP) framework.

Formalization of Spatio-Temporal Constraints: Temporal logic is well suited for expressing ROTR due to its spatio-temporal nature (e.g., *stop at the stop line for three seconds*) [21]–[23]. Numerous motion planning algorithms are capable of handling temporal logic constraints. Automata-based methods are typically preferred for linear

[1]Patrick Halder and Fabian Christ are with ZF Friedrichshafen AG, 88046 Friedrichshafen, Germany.

[2]Matthias Althoff is with the School of Computation, Information and Technology, Technical University of Munich, 85748 Garching, Germany.

{patrick.halder, fabian.christ}@zf.com, althoff@tum.de

temporal logic (LTL) constraints, while sampling-based or optimization-based methods are more commonly used for constraints formalized in STL [2]. Although sampling-based motion planning approaches are available for STL (e.g., see [24], [25]), they are often limited to fragments of STL and only exhibit asymptotic optimality. Optimization-based methods, such as mixed-integer approaches [26]–[28] and gradient-based approaches [29]–[31], are commonly used for motion planning with STL constraints. Mixed-integer methods usually lack scalability, while gradient-based methods often require smoothing the STL constraints, altering the underlying problem. Most methods in the literature only use simple STL formulas, revealing little about their real-world applicability (e.g., see [27], [32]). Similarly, to the best of our knowledge, STL is used barely in multi-objective problems (e.g., in [33]), which we aim to address in this work.

B. Contributions

In this paper, we present a novel lexicographic motion planner. In particular, our contributions are:

- decomposing the driving task into a rulebook and presenting the first optimization-based minimum-violation planner, which utilizes reachable sets and is subject to prioritized STL formulas,
- introducing a MICP-based implementation of our planner,
- showing how reachable sets and realistic STL formulas can be encoded as constraints for the ego vehicle, i.e., the vehicle to be controlled, and
- evaluating our approach on 1780 scenarios from the CommonRoad benchmark suite [34].

The remainder of this paper is structured as follows: Sec. II introduces required definitions. In Sec. III, we formulate the problem statement and present our solution concept. Sec. IV shows our MICP realization. Finally, we present experiments in Sec. V and conclude in Sec. VI.

II. Preliminaries

In this section, we provide the required definitions and briefly introduce STL.

A. Definitions

Let $k \in \mathbb{N}_0$ be a discrete time step corresponding to the time $t_k = k\Delta t$, where $\Delta t \in \mathbb{R}^+$ is the time increment. We refer to k_0 and k_f as the initial and final time step, respectively, $\mathbb{K} := \{k_0, k_1, \dots, k_f\}$ is the discrete time domain, and $n_k := k_f - k_0$. Further, let $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ be the set of admissible states, $\mathcal{U} \subseteq \mathbb{R}^{n_u}$ be the set of admissible inputs, and $\mathcal{Y} \subseteq \mathbb{R}^{n_y}$ be the set of admissible outputs. We utilize a curvilinear coordinate system, aligned with a reference path $\Gamma : \mathbb{R} \to \mathbb{R}^2$, mapping a Cartesian position coordinate to the arc length $s \in \mathbb{R}$ and the orthogonal deviation $d \in \mathbb{R}$ [35]. Let $\Theta(s)$ return the orientation at $\Gamma(s)$. Furthermore, let the corridor Λ be the set of points that have a maximum lateral distance of $d_{\text{cor}} \in \mathbb{R}$ to Γ.

Let $x \in \mathcal{X}, u \in \mathcal{U}$, and $y \in \mathcal{Y}$ represent the state, input, and output, respectively. The dynamics of the ego vehicle is given by the discrete-time system:

$$x_{k+1} = f(x_k, u_k), \tag{1}$$
$$y_k = g(x_k, u_k). \tag{2}$$

The solution of (1) for an input trajectory $\boldsymbol{u}(\cdot)$ and an initial state $x_0 \in \mathcal{X}_0$ is expressed by the trajectory $\boldsymbol{x}(\cdot)$, and we denote the corresponding output trajectory by $\boldsymbol{y}(\cdot)$.

Let $o_k^{(j)} := (s, d, v, u, \theta) \in \mathcal{O}$, $j \in \{1, \dots, n_o\}$, be the longitudinal position, lateral position, velocity, acceleration and orientation of the j-th obstacle in a scenario at time step k, respectively, where $\mathcal{O}$ is the set of admissible obstacle states, and n_o is the number of considered obstacles in a scenario. The function $\mathcal{Q}(o_k^{(j)}) \subset \mathbb{R}^2$ provides the occupancy of the obstacle at time step k. The occupancy of the ego vehicle at time step k is denoted by $\mathcal{E}(x_k) \subset \mathbb{R}^2$.

Definition 1 (Projection Operator)**.** *The projection operator* $\text{proj}_{\square} : \Omega \to \mathbb{R}$ *maps a vector* $\omega \in \Omega$ *to the elements specified by* $\square$*.*

Definition 2 (Longitudinal Position Intervals)**.** *We define the longitudinal position interval* $\mathcal{I}_e(x_k) \subset \mathbb{R}$ *of the ego vehicle and the longitudinal position interval* $\mathcal{I}_o(o_k^{(j)}) \subset \mathbb{R}$ *of obstacle* j *at time step* k *as:*

$$\mathcal{I}_e(x_k) := \text{proj}_s(\mathcal{E}(x_k)),\ \mathcal{I}_o(o_k^{(j)}) := \text{proj}_s(\mathcal{Q}(o_k^{(j)})).$$

The function $\text{rear}(\cdot)$ provides the minimum of a longitudinal position interval.

Let $\mathcal{N}_k := \{j \in \{1, \dots, n_o\} \mid \mathcal{Q}(o_k^{(j)}) \cap \Lambda \neq \emptyset\}$ be the indices of obstacles that occupy the corridor Λ at time step k. We then define the set of forbidden states as:

$$\mathcal{F}_k := \left\{ x_k \in \mathcal{X}_k \middle| \left(\mathcal{I}_e(x_k) \cap \bigcup_{j \in \mathcal{N}_k} \mathcal{I}_o(o_k^{(j)}) \right) \neq \emptyset \right\}.$$

Definition 3 (Reachable Set [16])**.** *Let the initial reachable set be* $\mathcal{R}_0 \subseteq \mathcal{X}_0$*. Then, the reachable set* $\mathcal{R}_{k+1}$ *that can be reached from a previous reachable set* $\mathcal{R}_k$ *without intersecting with* $\mathcal{F}_{k+1}$ *is defined as:*

$$\mathcal{R}_{k+1} = \{x_{k+1} \in \mathcal{X}_{k+1} \mid \exists x_k \in \mathcal{R}_k, \exists u_k \in \mathcal{U} : x_{k+1} = f(x_k, u_k) \wedge x_{k+1} \notin \mathcal{F}_{k+1}\}.$$

Since obtaining the exact reachable sets is computationally expensive, we use an over-approximation of $\mathcal{R}_k$, here the union of convex polytopes [16].

Definition 4 (Totally Ordered Rulebook (based on [8, Def. 4]))**.** *Let a totally ordered rulebook be the tuple* $\langle \Phi, < \rangle$*, where* Φ *is a set of rules and* $<$ *is a strict total order.*

We denote the cardinality of Φ by n_Φ. Given a set of trajectories $\mathcal{T}$, a totally ordered rulebook induces a lexicographic order on $\mathcal{T}$ [8].

B. Signal Temporal Logic

STL is a formal language to specify spatio-temporal properties of dynamical systems over real-valued trajectories [36]. We define the STL syntax over output signals $\boldsymbol{y}(\cdot)$ by the following grammar [37, Sec. 2.1]:

$$\varphi := \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathbf{U}_I \varphi_2 \mid \varphi_1 \mathbf{S}_I \varphi_2,$$

where μ is a predicate of the form $\mu := p(\boldsymbol{y}, k) \geq 0$, with $p : \mathbb{R}^{n_y \times n_k} \times \mathbb{K} \to \mathbb{R}$. Furthermore, φ, φ_1, and φ_2 are STL formulas, and $\neg$ and $\wedge$ are negation and conjunction, respectively. The until operator $\mathbf{U}_I$ specifies that φ_1 holds *until* φ_2, and the since operator $\mathbf{S}_I$ specifies that φ_1 holds *since* φ_2. The operators are defined over the time intervals $I \subseteq \mathbb{N}_0$. We can derive further operators given the above ones specifying that a property has to hold *globally* ($\mathbf{G}_I$), *once* ($\mathbf{O}_I$), or *eventually* ($\mathbf{F}_I$). We denote the satisfaction of an STL formula φ by a trajectory $\boldsymbol{y}$ at time step k with $(\boldsymbol{y}, k) \models \varphi$. The robustness of a finite trajectory $\boldsymbol{y}$ regarding an STL formula φ is denoted by $\rho^{\varphi}(\boldsymbol{y}, k) \in \mathbb{R}$. Intuitively, robustness provides a quantitative assessment of the degree of compliance or violation of a trajectory for a given formula. The robustness $\rho^{\varphi}(\boldsymbol{y}, k)$ is positive iff $\boldsymbol{y} \models \varphi$. For the definition of further temporal operators, short-hand notations, and the qualitative and quantitative semantics of STL, we refer the reader to [37].

III. Problem Statement and Solution Concept

We now present our problem statement and our proposed solution concept.

A. Problem Statement

Fig. 2 shows one possible decomposition of the specification of an autonomous vehicle into a rulebook, where collision avoidance has the highest priority, followed by the ROTR, mission, and comfort specifications.

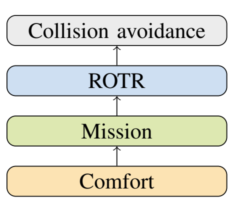


Fig. 2: A possible rulebook of the specification of an autonomous vehicle. The arrows indicate higher prioritized specifications.

As defined in Def. 3, reachable sets contain all collision-free configurations of the ego vehicle over time. Thus, collision avoidance can be enforced by stating:

$$\forall k \in \mathbb{K} : x_k \in \mathcal{R}_k. \tag{3}$$

When the satisfaction of this constraint is infeasible, we assume that a fail-safe maneuver is triggered (e.g., computed as in [17]).

Let the ROTR and the mission be the STL formulas $\varphi_i \in \Phi$, with $i \in \{1, \ldots, n_\Phi\}$, where lower indices i indicate a higher priority. Our objective is that rules are violated as little as possible, which we formalize using the robustness:

$$\varrho(\boldsymbol{y}, \varphi) := \min(0, \rho^{\varphi}(\boldsymbol{y}, 0)). \tag{4}$$

By bounding the robustness by zero, only the rule violation generates costs in the optimization problem.

The driving comfort is represented as a quadratic cost term which, e.g., makes it possible to punish high acceleration values:

$$q(\boldsymbol{y}) := \sum_{k \in \mathbb{K}} y_k^T Q y_k, \tag{5}$$

where Q is a weight matrix of appropriate size.

We now transform our rulebook into a lexicographic optimization problem and gather the ROTR, the mission, and the comfort requirements in the vector-valued objective function $\boldsymbol{h} : \mathbb{R}^{n_y \times n_k} \to \mathbb{R}^{n_\Phi + 1}$:

$$\boldsymbol{h}(\boldsymbol{y}) := [\varrho(\boldsymbol{y}, \varphi_1), \varrho(\boldsymbol{y}, \varphi_2), \ldots, \varrho(\boldsymbol{y}, \varphi_{n_\Phi}), -q(\boldsymbol{y})].$$

We can now define our lexicographic optimization problem as follows:

$$\begin{aligned}
\operatorname*{lex\,max}_{x \in \mathcal{X}, u \in \mathcal{U}} \ & \boldsymbol{h}(\boldsymbol{y}) && \textit{(ROTR, mission, comf.)} \\
\text{s.t. } & x_0 \in \mathcal{X}_0, && \textit{(initial state)} \\
& \forall k \in \mathbb{K}: && \\
& x_{k+1} = f(x_k, u_k), && \textit{(dynamics)} \\
& y_k = g(x_k, u_k), && \textit{(output)} \\
& x_k \in \mathcal{R}_k, && \textit{(coll. avoidance)}
\end{aligned} \tag{6}$$

where the $\operatorname{lex\,max}$ operator defines a lexicographic optimization of $\boldsymbol{h}(\boldsymbol{y})$ (cf. [1, Sec. 5.1]).

B. Solution Concept

We propose Alg. 1 to solve problem (6). It is based on the *preemptive* solution procedure for lexicographic optimization problems (cf. [38]), where several optimization problems are solved successively. We utilize this procedure to obtain intermediate solutions for problem (6), providing the advantage of an anytime-like behavior of the algorithm, as will be discussed in the next section. Additionally, Alg. 1 performs a reachability analysis and considers the robustness of STL formulas.

The inputs to Alg. 1 are the initial state x_0, the rulebook $\langle \Phi, < \rangle$, the quadratic cost function q, and the intermediate solution trigger ε. We start by performing a reachability analysis based on the initial state x_0 over the planning time horizon $\mathbb{K}$ (see line 1), returning the reachable sets $\mathcal{R}$ for all time steps. If the reachable sets vanish (see line 3), i.e., the reachable set $\mathcal{R}_{k_f}$ of the final time step is empty, then we return $\emptyset$ and trigger a fail-safe maneuver (e.g., as in [17]). Afterward, a solver is instantiated, and the hard constraints originating from the reachable set are added (see lines 5 and 6). Subsequently, we loop over the n_Φ formulas of our rulebook and solve one optimization problem per loop (see lines 7 to 16). The objective is to maximize the robustness of the current formula φ_i, while the optimal robustness values $\hat{\rho}^{\varphi_{i-1}}$ of previous loops constrain their satisfaction, respectively a violation. Note that we directly maximize the robustness (see line 8) instead of (4) and bound the robustness by zero in the constraint of the higher prioritized rules (see line 10). The benefit is that after each loop, we can output an intermediate solution of the problem which maximizes the satisfaction of the current formula and minimally violates the more important formulas (see line 14). Hence, this anytime-like behavior allows guaranteeing to satisfy (or minimally violate) the unimportant formulas, while the satisfaction of the more important formulas is assured. We see this to be especially beneficial when the runtime is limited or the planning must be terminated for some reason. This differentiates our approach from other hierarchical methods (e.g., [4]). The external condition to return an intermediate solution is denoted by ε. As a last step, we minimize the quadratic cost function q (see lines 17 to 19). The output of Alg. 1 is the lexicographically optimal output trajectory $\hat{\boldsymbol{y}}^q$.

Algorithm 1 LEXICOGRAPHIC STL PLANNER

Input: Initial state x_0, rulebook $\langle \Phi, < \rangle$, quadratic cost function q, intermediate solution trigger ε
Output: Optimal solution $\hat{\boldsymbol{y}}^q$, intermediate solution $\hat{\boldsymbol{y}}^{\varphi_i}$, or $\emptyset$

1: $\mathcal{R} \leftarrow$ PERFORMREACHABILITYANALYSIS$(x_0, \mathbb{K})$
2: **if** $\mathcal{R}_{k_f} = \emptyset$ **then**
3:     **return** $\emptyset$ ▷ trigger fail-safe maneuver
4: **end if**
5: $sol \leftarrow$ SOLVER( )
6: sol.ADDHARDCONSTR$(\mathcal{R})$
7: **for** $i \in \{1, \dots, n_\Phi\}$ **do**
8:     sol.SETMAXIMIZATIONOBJECTIVE$(\rho^{\varphi_i}(\boldsymbol{y}, 0))$
9:     **if** $i > 1$ **then**
10:         sol.ADDHARDCONSTR$(\rho^{\varphi_{i-1}} \geq \min(0, \hat{\rho}^{\varphi_{i-1}}))$
11:     **end if**
12:     $\langle \hat{\boldsymbol{y}}^{\varphi_i}, \hat{\rho}^{\varphi_i} \rangle \leftarrow sol$.SOLVE( )
13:     **if** ε is satisfied **then**
14:         **return** $\hat{\boldsymbol{y}}^{\varphi_i}$
15:     **end if**
16: **end for**
17: sol.SETMAXIMIZATIONOBJECTIVE$(-q(\boldsymbol{y}))$
18: sol.ADDHARDCONSTR$(\rho^{\varphi_{n_\Phi}} \geq \min(0, \hat{\rho}^{\varphi_{n_\Phi}}))$
19: $\langle \hat{\boldsymbol{y}}^q, \hat{q} \rangle \leftarrow sol$.SOLVE( )
20: **return** $\hat{\boldsymbol{y}}^q$

IV. MICP REALIZATION

Alg. 1 applies to arbitrary motion planning applications. In this section, we provide a possible realization of Alg. 1 for longitudinal motion planning using a MICP formulation. We choose this formulation since it allows us to find a global optimum [26] if it exists and to solve problem (6) without adaptions to the STL semantics, e.g., as it would be required for gradient-based formulations (e.g., see [32]). Also, we use the mixed-integer encoding for STL formulas of [39] to consider them as constraints in our optimization problem.

A. System

For our MICP realization of Alg. 1, we use the following discrete-time linear system:

$$x_{k+1} = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} x_k + \begin{pmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{pmatrix} u_k, \tag{7}$$

where $x = (s, v)^T$ describes the position s of the ego vehicle along Γ as well as its velocity v, u is the acceleration, and $y = (s, v, u)^T$ is the system output. We assume that the ego vehicle exactly follows the reference path Γ and never leaves the corridor Λ. To also consider lateral motion, we use the method from [40] and constrain the velocity based on the curvature of the reference and the maximum lateral acceleration of the ego vehicle.

B. STL Formulas

We use the general driving rules $\varphi_{\mathrm{G1}}, \varphi_{\mathrm{G2}}, \varphi_{\mathrm{G3}}$ and φ_{G4} from [22]. These rules, originally proposed for monitoring purposes, are adapted to suit our motion planning problem and are evaluated for all obstacles and speed limits in a scenario. Additionally, we add a custom mission rule φ_{M1}.

The STL formulas are shown in Tab. I. They express the following: φ_{G1} ensures a safe distance to preceding vehicles and to restore it after $k_c \in \mathbb{N}_0$ time steps in case of a cut-in maneuver of other vehicles; φ_{G2} punishes unnecessary breaking without reason; φ_{G3} ensures maximum speed limits; φ_{G4} enforces not to impede the traffic flow (which is equivalent to moving faster than a required speed); and φ_{M1} ensures reaching the goal area in the time interval I_{g}.

TABLE I: The used STL rules.

Rule	Definition
φ_{G1}	$\mathbf{G}(\text{in\_same\_lane} \wedge \text{in\_front\_of} \wedge \neg\mathbf{O}_{[0,k_c]}(\varphi_{\text{cut\_in}} \wedge \mathbf{O}_{[1,1]}(\neg\varphi_{\text{cut\_in}})) \implies \text{keeps\_safe\_distance\_prec})$ $\varphi_{\text{cut\_in}} := \neg\text{single\_lane} \wedge ((\text{is\_left} \wedge \neg\text{orientation\_is\_positive}) \vee (\neg\text{is\_left} \wedge \text{orientation\_is\_positive})) \wedge \text{in\_same\_lane}$
φ_{G2}	$\mathbf{G}\ (\text{is\_braking} \implies \neg\varphi_{\text{reason\_1}} \wedge \neg\varphi_{\text{reason\_2}})$ $\varphi_{\text{reason\_1}} := \text{brakes\_abruptly} \wedge (\neg\text{in\_same\_lane} \vee \neg\text{in\_front\_of})$ $\varphi_{\text{reason\_2}} := (\text{in\_same\_lane} \wedge \text{in\_front\_of} \wedge \text{keeps\_safe\_distance\_prec} \wedge \text{brakes\_abruptly\_relative})$
φ_{G3}	$\mathbf{G}\ (\text{is\_after\_limit\_start} \wedge \text{is\_before\_limit\_end}) \implies \text{is\_below\_speed\_limit})$
φ_{G4}	$\mathbf{G}\ (\neg\varphi_{\text{slow\_leading\_vehicle}} \implies \varphi_{\text{preserves\_flow}})$ $\varphi_{\text{slow\_leading\_vehicle}} := \text{in\_same\_lane} \wedge \text{in\_front\_of} \wedge \text{is\_slow}$ $\varphi_{\text{preserves\_flow}} := (\text{is\_after\_limit\_start} \wedge \text{is\_before\_limit\_end}) \implies \text{is\_above\_required\_speed}$
φ_{M1}	$\mathbf{F}_{I_{\mathrm{g}}}\ (\text{is\_after\_goal\_start} \wedge \text{is\_before\_goal\_end})$

C. Predicates

All predicates in Tab. I (except keeps_safe_distance_prec) can be stated in the following linear form:

$$p(\boldsymbol{y}, k) := \frac{1}{\nu} \left(\alpha(k)^T y_k - \beta(k) \right), \tag{8}$$

where $\alpha : \mathbb{K} \to \mathbb{R}^{n_y}$, $\beta : \mathbb{K} \to \mathbb{R}$, and $\nu \in \mathbb{R}$ is a predicate-specific scaling factor. This linear form allows one to encode the predicates with the approach presented in [39].

The definitions of $\alpha(k)$ and $\beta(k)$ follow from [22] and are presented in Tab. II. Since we only consider longitudinal motion along the reference path Γ, some predicates become independent of the ego state, realized by $\alpha(k) = \mathbf{0} := (0, 0, 0)$. Further, $\mathbf{e}_i$ is a zero vector, where the i-th entry is one. The functions $\beta_1(\cdot)$ and $\beta_2(\cdot)$ are derived from [41, (1)] and [41, (2)], respectively. A speed limit σ along the reference path Γ is defined by a minimum position $\underline{s}_\sigma$, a maximum position $\overline{s}_\sigma$, and a maximum velocity $\overline{v}_\sigma$. We denote the goal region of the scenario (e.g., a lanelet) by γ, and $\underline{s}_\gamma$ and $\overline{s}_\gamma$ are the minimum and maximum position of γ projected onto the reference path Γ, respectively. For the definition of $v_{\text{max\_2}}(\cdot)$, please see [22, Sec. IV.B]. Further, $\overline{u}_{\text{abr}}$ and Δv_{fl} are adjustable parameters, and the length of the ego vehicle is denoted by $\ell \in \mathbb{R}^+$.

TABLE II: Parameters of the used predicates.

Predicate	$\alpha(k)$	$\beta(k)$
in_same_lane	$\mathbf{0}$	$-\beta_1(o_k^{(j)})$
in_front_of	$-\mathbf{e}_1$	$\ell/2 - \text{rear}(\mathcal{I}_o(o_k^{(j)}))$
single_lane	$\mathbf{0}$	$-\beta_2(o_k^{(j)})$
is_left	$\mathbf{0}$	$\text{proj}_d(o_k^{(j)})$
orientation_is_positive	$\mathbf{0}$	$\text{proj}_\theta(o_k^{(j)}) - \Theta(\text{proj}_s(o_k^{(j)}))$
is_braking	$-\mathbf{e}_3$	0
brakes_abruptly	$-\mathbf{e}_3$	$-\overline{u}_{\text{abr}}$
brakes_abruptly_relative	$-\mathbf{e}_3$	$-\text{proj}_u(o_k^{(j)}) - \overline{u}_{\text{abr}}$
is_after_limit_start	$\mathbf{e}_1$	$\underline{s}_\sigma$
is_before_limit_end	$-\mathbf{e}_1$	$-\overline{s}_\sigma$
is_above_required_speed	$\mathbf{e}_2$	$\overline{v}_\sigma - \Delta_{\text{fl}}$
is_below_speed_limit	$-\mathbf{e}_2$	$-\overline{v}_\sigma$
is_slow	$\mathbf{0}$	$\text{proj}_v(o_k^{(j)}) + \Delta v_{\text{fl}} - v_{\text{max\_2}}(o_k^{(j)})$
is_after_goal_start	$\mathbf{e}_1$	$\underline{s}_\gamma$
is_before_goal_end	$-\mathbf{e}_1$	$-\overline{s}_\gamma$

The predicate keeps_safe_distance_prec is the only non-linear predicate since it depends quadratically on the ego velocity. It is defined as [22, Sec. IV.C]:

$$p(\boldsymbol{y}, k) := \frac{1}{\nu}\big(\text{rear}(\mathcal{I}_o(o_k^{(j)})) - (\text{proj}_s(y_k) + \ell/2) \\ - \Delta_{\text{safe}}(\text{proj}_v(y_k))\big), \tag{9}$$

where $\Delta_{\text{safe}}(\cdot)$ is the required safe distance to a preceding obstacle [42, (4.7)]. We approximate (9) with n_h hyperplanes $h := a_h^T y_k - b_h = 0$, where $a_h \in \mathbb{R}^{n_y}$ and $b_h \in \mathbb{R}$. Therefore, we utilize the piecewise-linear approximation of the safe distance from [42, (4.12)]. The resulting approximation of (9) is under-approximative and, thus, assures safety.

Fig. 3 visualizes the linearization of (9) with two hyperplanes. The hyperplanes h are encoded in our optimization problem using the standard big-M constraints from [39].

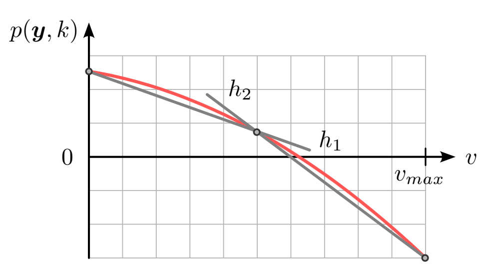


Fig. 3: The function $p(\boldsymbol{y}, k)$ of the predicate keeps_safe_distance_prec and two linear approximations h_1 and h_2, visualized for $\text{proj}_s(y_k) = 0$.

D. Encoding of Reachable Sets

For each convex polytope $\mathcal{P}_k^{(l)} \subset \mathbb{R}^2$ at time step k, with $l \in \mathbb{N}_0$, we define a corresponding binary variable $z_{k,l} \in \{0, 1\}$, and state:

$$z_{k,l} = \begin{cases} 1, & \text{if } x_k \in \mathcal{P}_k^{(l)}, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

The set membership $x_k \in \mathcal{P}_k^{(l)}$ can be encoded using linear functions (cf. [16]). Let us introduce the number of polytopes at time step k as n_p^k. We can then encode the collision avoidance constraint (3) in a MICP problem by:

$$\forall k \in \mathbb{K}: \; 1 \leq \sum_{l \in \{1,\dots,n_p^k\}} z_{k,l}, \tag{11}$$

because x_k must be in at least one polytope $\mathcal{P}_k^{(l)}$ per time step.

V. Numerical Experiments

We implemented the presented MICP realization in Python based on [28], and the reachability analysis is performed with the CommonRoad-Reach toolbox [43]. We use Gurobi[1] as the solver for the MICP problems. The experiments are executed on an Intel Core$^{\text{TM}}$ i5-10310U CPU. Our code can be accessed at gitlab.lrz.de/tum-cps/mvp. For all experiments, the STL rules have the order $\varphi_{\text{G1}} > \varphi_{\text{G2}} > \varphi_{\text{G3}} > \varphi_{\text{G4}} > \varphi_{\text{M1}}$, and the planning time increment is $\Delta t = 0.2\,\text{s}$.

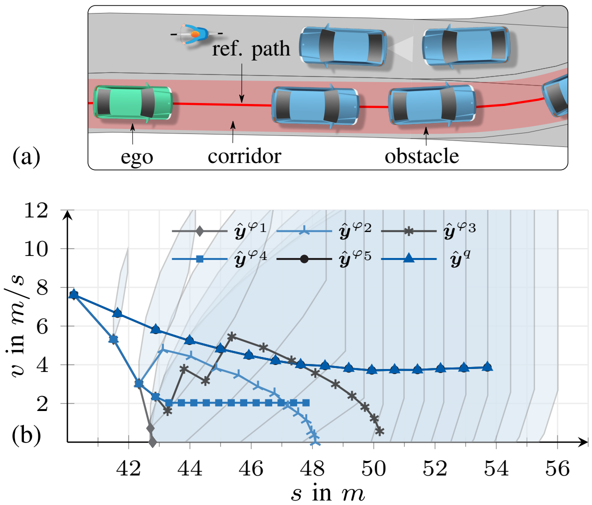
Let us first present *scenario I*[2], where the violation of rules is inevitable. Fig. 4 shows the initial configuration of this scenario and the intermediate trajectories resulting from Alg. 1, line 14. All trajectories remain in the reachable sets (see the blue polytopes in (b)). With each intermediate solution, we converge more to the lexicographic optimal solution. The first two intermediate solutions minimize the violation for rule φ_{G1} and rule φ_{G2}, and the minimum violation of these rules is assured for all subsequent intermediate solutions (indicated by the blue background in (c)). For rule φ_{G3} and rule φ_{G4}, no violation is necessary, and the robustness for the subsequent solutions can vary (indicated by the yellow background in (c)). For rule φ_{M1}, a violation is again inevitable, and the final solution $\hat{\boldsymbol{y}}^q$ is a smooth trajectory (resulting from the quadratic cost function), that minimally violates the rules.

Next, we evaluate our algorithm based on 1780 CommonRoad scenarios. We compare our lexicographic formulation (*lex*) with three alternative formulations:

- Single hard constraint planner (*shc*): $\max_{x \in \mathcal{X}, u \in \mathcal{U}} \left(-q(\boldsymbol{y})\right)$, s.t. $\rho^{\varphi_\Phi}(\boldsymbol{y}, 0) \geq 0$;
- Single soft constraint planner (*ssc*): $\max_{x \in \mathcal{X}, u \in \mathcal{U}}(-q(\boldsymbol{y}) + w_s \varrho(\boldsymbol{y}, \varphi_\Phi))$;
- Multi soft constraint planner (*msc*): $\max_{x \in \mathcal{X}, u \in \mathcal{U}}(-q(\boldsymbol{y}) + \sum_{i \in \{1,\dots,n_\Phi\}} w_i \varrho(\boldsymbol{y}, \varphi_i))$;

where $\varphi_\Phi := \bigwedge_{i=1}^{n_\Phi} \varphi_i$ is the conjunction of all formulas Φ in the rulebook and $w_s, w_i \in \mathbb{R}$ are weights. We tuned these weights based on the *scenario I* using a grid search to gain a close approximation of the solution from the *lex* planner. Further, we define the metric m to compare

[1] https://www.gurobi.com
[2] CommonRoad ID: DEU_Guetersloh-6_1_T-1



Sol.	$\rho^{\varphi_{G1}}$	$\rho^{\varphi_{G2}}$	$\rho^{\varphi_{G3}}$	$\rho^{\varphi_{G4}}$	$\rho^{\varphi_{M1}}$	q
$\hat{\boldsymbol{y}}^{\varphi_1}$	**-0.04**	-0.94	0.07	0.67	-10.20	54.18
$\hat{\boldsymbol{y}}^{\varphi_2}$	-0.04	**-0.89**	0.07	0.67	-9.67	53.21
$\hat{\boldsymbol{y}}^{\varphi_3}$	-0.04	-0.89	**0.07**	0.63	-9.46	75.82
$\hat{\boldsymbol{y}}^{\varphi_4}$	-0.04	-0.89	0.07	**0.87**	-9.70	41.03
$\hat{\boldsymbol{y}}^{\varphi_5}$	-0.04	-0.89	0.07	0.38	**-9.11**	19.25
$\hat{\boldsymbol{y}}^{q}$	-0.04	-0.89	0.07	0.38	-9.11	**6.03**

(c)

Fig. 4: (a) Initial configuration of *scenario I*. (b) Intermediate solutions and reachable sets from Alg. 1. (c) Robustness and cost values for the intermediate solutions.

the solution $\boldsymbol{y}_{lex}$ of our lexicographic formulation with the solutions of the alternative formulations, denoted by $\boldsymbol{y}_{\triangle}$, with $\triangle \in \{shc, ssc, msc\}$:

$$m(\boldsymbol{y}_{lex}, \boldsymbol{y}_{\triangle}) := \sum_{i \in \{1,\dots,n_\Phi\}} \tilde{m}, \text{ with}$$

$$\tilde{m} := \begin{cases} 1, & \text{if } \rho^{\varphi_i}(\boldsymbol{y}_{\triangle}, 0) < \rho^{\varphi_i}(\boldsymbol{y}_{lex}, 0) < 0, \\ 0, & \text{otherwise.} \end{cases}$$

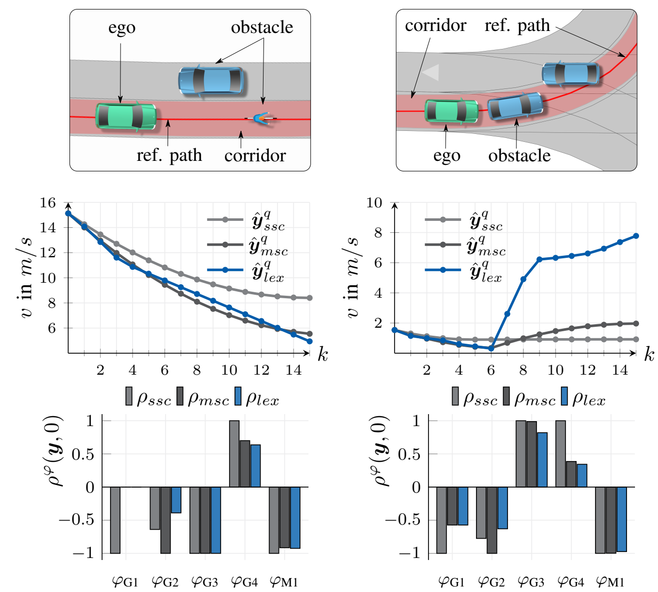
The metric m describes the number of rules for which there is a larger violation than lexicographically necessary.

Tab. III shows the results of the evaluations on the scenarios. The optimization problems contain each around 19300 optimization variables and 21900 constraints.

TABLE III: Evaluation results on 1780 CommonRoad scenarios.

Planner	Converged scenarios	Scenarios with $m > 0$	m avg.	m max	Avg. solver time in sec.
shc	99	0	-	-	0.02
ssc	1780	1526	0.87	3	0.04
msc	1780	832	0.48	2	0.05
lex	1780	0	0	0	0.33

Tab. III indicates that the hard constraint planner *shc* does not converge in many cases, while the other planners always provide a solution, indicating the necessity to violate rules. Although we conscientiously tuned the weights of the *ssc* and *msc* planners for *scenario I*, these planners provide many non-minimum-violation trajectories, as indicated by the m-values. This shows the drawback of weighted cost functions since a cumbersome scenario-dependent weight tuning is required. Fig. 5 visualizes this issue for two example scenarios. We can observe from the m-values of the *ssc* and *msc* planner that the more weights we can tune, the closer we can approximate the solution of the *lex* planner. However, a direct consequence is the increased effort in weight tuning, which is not required by our lexicographic approach. Finally, the calculation time for the *lex* planner is higher than the others since it solves six consecutive optimization problems, while the other planners only solve one optimization problem. We do not consider this to be a drawback since Alg. 1 provides a minimum-violation intermediate solution in case the runtime is limited and believe that this is more beneficial than either not providing a solution at all or providing a solution with incorrect preferences.



(a) *Scenario II*[3]. (b) *Scenario III*[4].

Fig. 5: Two example scenarios, where the *ssc* and *msc* planner violate the rules more than necessary. The initial configuration, the resulting trajectories, and the respective robustness values are shown for the *lex*, *ssc*, and *msc* planner, respectively. The robustness values are scaled to the range $[-1, 1]$ per rule for better visualization.

VI. Conclusions

In this paper, we present the benefits of decomposing the specification of an autonomous vehicle as a rulebook to handle contradicting constraints effectively. We demonstrate how this problem can be transformed into a lexicographic optimization problem and propose a novel optimization-based minimum-violation planner based on a preemptive lexicographic optimization procedure. Unlike existing work, our planer assures collision avoidance using reachable sets and minimally violates the ROTR, formalized in STL. Furthermore, we present a MICP-based realization of our algorithm considering realistic ROTR. The experimental results demonstrate that our lexicographic method outperforms classical methods of constraint prioritization. Future work

[3]CommonRoad ID: DEU_Flensburg-98_1_T-1
[4]CommonRoad ID: PRI_Barceloneta-2_1_T-1

involves investigating non-preemptive lexicographic problem formulations and incorporating more ROTR in the planning. Finally, we want to embed our approach into a safe motion planning framework.

REFERENCES

[1] M. Ehrgott, *Multicriteria optimization*. Springer Science & Business Media, 2005, vol. 2.

[2] N. Mehdipour, M. Althoff, R. D. Tebbens, and C. Belta, "Formal methods to comply with rules of the road in autonomous driving: State of the art and grand challenges," *Automatica*, vol. 152, 2023.

[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. on Intell. Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[4] W. Xiao, N. Mehdipour, A. Collin, A. Bin-Nun, E. Frazzoli, R. D. Tebbens, and C. Belta, "Rule-based optimal control for autonomous driving," in *Proc. of the ACM/IEEE Int. Conf. on Cyber-Physical Sys.*, 2021, pp. 143–154.

[5] M. Althoff, S. Maierhofer, and C. Pek, "Provably-correct and comfortable adaptive cruise control," *IEEE Trans. on Intell. Vehicles*, vol. 6, no. 1, pp. 159–174, 2021.

[6] E. Aasi, C.-I. Vasile, and C. Belta, "A control architecture for provably-correct autonomous driving," in *Proc. of the American Control Conf.*, 2021, pp. 2913–2918.

[7] S. Veer, K. Leung, R. K. Cosner, Y. Chen, P. Karkus, and M. Pavone, "Receding horizon planning with rule hierarchies for autonomous vehicles," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2023, pp. 1507–1513.

[8] 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 Int. Conf. on Robotics and Automation*, 2019, pp. 8536–8542.

[9] A. Collin, A. Bilka, S. Pendleton, and R. D. Tebbens, "Safety of the intended driving behavior using rulebooks," in *Proc. of the IEEE Intell. Vehicles Symposium*, 2020, pp. 136–143.

[10] J. De Freitas, A. Censi, B. Walker Smith, L. Di Lillo, S. E. Anthony, and E. Frazzoli, "From driverless dilemmas to more practical commonsense tests for automated vehicles," *Proc. of the national academy of sciences*, vol. 118, no. 11, 2021.

[11] P. Halder and M. Althoff, "Minimum-violation velocity planning with temporal logic constraints," in *Proc. of the IEEE Int. Conf. on Intell. Transp. Sys.*, 2022, pp. 2520–2527.

[12] B. Helou, A. Dusi, A. Collin, N. Mehdipour, Z. Chen, C. Lizarazo, C. Belta, T. Wongpiromsarn, R. D. Tebbens, and O. Beijbom, "The reasonable crowd: Towards evidence-based and interpretable models of driving behavior," in *Proc. of the IEEE/RSJ Int. Conf. on Intell. Robots and Sys.*, 2021, pp. 6708–6715.

[13] L. I. Reyes 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. on Decision and Control*, 2013, pp. 3217–3224.

[14] C.-I. Vasile, J. Tůmová, S. Karaman, C. Belta, and D. Rus, "Minimum-violation scLTL motion planning for mobility-on-demand," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2017, pp. 1481–1488.

[15] T. Wongpiromsarn, K. Slutsky, E. Frazzoli, and U. Topcu, "Minimum-violation planning for autonomous systems: Theoretical and practical considerations," in *Proc. of the American Control Conf.*, 2021, pp. 4866–4872.

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

[17] C. Pek, S. Manzinger, M. Koschi, and M. Althoff, "Using online verification to prevent autonomous vehicles from causing accidents," *Nature Machine Intelligence*, vol. 2, no. 9, pp. 518–528, 2020.

[18] H. Roehm, J. Oehlerking, M. Woehrle, and M. Althoff, "Model conformance for cyber-physical systems: A survey," *ACM Trans. on Cyber-Physical Sys.*, vol. 3, no. 3, pp. 1–26, 2019.

[19] S. Manzinger, C. Pek, and M. Althoff, "Using reachable sets for trajectory planning of automated vehicles," *IEEE Trans. on Intell. Vehicles*, vol. 6, no. 2, pp. 232–248, 2021.

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

[21] K. Esterle, L. Gressenbuch, and A. Knoll, "Modeling and testing multi-agent traffic rules within interactive behavior planning," in *IROS Workshop on Perception, Learning, and Control for Autonomous Agile Vehicles*, 2020.

[22] 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. Vehicles Symposium*, 2020, pp. 752–759.

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

[24] J. Karlsson and J. Tůmová, "Intention-aware motion planning with road rules," in *Proc. of the IEEE Int. Conf. on Automation Science and Engineering*, 2020, pp. 526–532.

[25] J. Karlsson, S. van Waveren, C. Pek, I. Torre, I. Leite, and J. Tůmová, "Encoding human driving styles in motion planning for autonomous vehicles," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2021, pp. 1050–1056.

[26] C. Belta and S. Sadraddini, "Formal methods for control synthesis: An optimization perspective," *Annual Review of Control, Robotics, and Autonomous Sys.*, vol. 2, no. 1, pp. 115–140, 2019.

[27] Y. E. Sahin, R. Quirynen, and S. D. Cairano, "Autonomous vehicle decision-making and monitoring based on signal temporal logic and mixed-integer programming," in *Proc. of the American Control Conf.*, 2020, pp. 454–459.

[28] V. Kurtz and H. Lin, "Mixed-integer programming for signal temporal logic with fewer binary variables," *IEEE Control Sys. Letters*, vol. 6, pp. 2635–2640, 2022.

[29] Y. V. Pant, H. Abbas, and R. Mangharam, "Smooth operator: Control using the smooth robustness of temporal logic," in *Proc. of the IEEE Conf. on Control Technology and Applications*, 2017, pp. 1235–1240.

[30] N. Mehdipour, C.-I. Vasile, and C. Belta, "Arithmetic-geometric mean robustness for control from signal temporal logic specifications," in *Proc. of the American Control Conf.*, 2019, pp. 1690–1695.

[31] Y. Mao, B. Acikmese, P.-L. Garoche, and A. Chapoutot, "Successive convexification for optimal control with signal temporal logic specifications," in *Proc. of the ACM Int. Conf. on Hybrid Systems: Computation and Control*, no. 9, 2022.

[32] Y. Gilpin, V. Kurtz, and H. Lin, "A smooth robustness measure of signal temporal logic for symbolic control," *IEEE Control Sys. Letters*, vol. 5, no. 1, pp. 241–246, 2021.

[33] N. Mehdipour, C.-I. Vasile, and C. Belta, "Specifying user preferences using weighted signal temporal logic," *IEEE Control Sys. Letters*, vol. 5, no. 6, pp. 2006–2011, 2021.

[34] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intell. Vehicles Symposium*, 2017, p. 719–726.

[35] E. Héry, S. Masi, P. Xu, and P. Bonnifait, "Map-based curvilinear coordinates for autonomous vehicles," in *Proc. of the IEEE Int. Conf. on Intell. Transp. Sys.*, 2017, pp. 1–7.

[36] O. Maler and D. Ničković, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.

[37] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Ničković, and S. Sankaranarayanan, *Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications*. Springer, 2018, pp. 135–175.

[38] L. Lai, L. Fiaschi, M. Cococcioni, and K. Deb, "Pure and mixed lexicographic-paretian many-objective optimization: State of the art," *Natural Computing*, vol. 22, no. 2, pp. 227–242, 2022.

[39] S. Sadraddini and C. Belta, "Formal synthesis of control strategies for positive monotone systems," *IEEE Trans. on Automatic Control*, vol. 64, no. 2, pp. 480–495, 2019.

[40] J. Eilbrecht and O. Stursberg, "Challenges of trajectory planning with integrator models on curved roads," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 15 588–15 595, 2020.

[41] L. Gressenbuch and M. Althoff, "Predictive monitoring of traffic rules," in *Proc. of the IEEE Int. Conf. on Intell. Transp. Sys.*, 2021, pp. 915–922.

[42] C. F. Pek, "Provably safe motion planning for autonomous vehicles through online verification," Ph.D. dissertation, Technische Universität München, 2020.

[43] E. I. 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. on Intell. Transp. Sys.*, 2022, pp. 2313–2320.