Rule-Compliant Trajectory Repairing using Satisfiability Modulo Theories

Yuanfei Lin and Matthias Althoff

***Abstract*— Autonomous vehicles must comply with traffic rules. However, most motion planners do not explicitly consider all relevant traffic rules. Once traffic rule violations of an initially-planned trajectory are detected, there is often not enough time to replan the entire trajectory. To solve this problem, we propose to repair the initial trajectory by investigating the satisfiability modulo theories paradigm. This framework makes it efficient to reason whether and how the trajectory can be repaired and, at the same time, determine the part along the trajectory that can remain unchanged. Moreover, the robustness of traffic rule satisfaction is used to formulate a convex optimization problem for generating rule-compliant trajectories. We compare our approach with trajectory replanning and demonstrate its usefulness with traffic scenarios from the CommonRoad benchmark suite and recorded data. The evaluation result shows that rule-compliant trajectory repairing is computationally efficient and widely applicable.**

I. Introduction

One of the barriers to the development of autonomous driving is the liability issue for traffic accidents. This issue can be addressed, e.g., by unambiguously formalizing traffic rules for autonomous vehicles [1]. If autonomous vehicles always comply with traffic rules, they cannot be held liable for a collision. However, it is computationally nontrivial to ensure the compliance of real-time motion planning with all traffic rule constraints, especially in complex situations.

Compliance with traffic rules can be evaluated with runtime monitors online [2]. If planned trajectories are not rule-compliant or physically infeasible, one can replan them for consecutive planning cycles. Replanning a complete trajectory, however, is often unnecessary and time-consuming. One interesting approach is trajectory repairing, as visualized in Fig. 1 and proposed in our previous work [3], to overcome this challenge. The concept in [3] only considers scenarios with collisions but does not repair trajectories violating traffic rules formalized in temporal logic, which is addressed in this study.

A. Related Work

Subsequently, we categorize related works using rule-based trajectory planning algorithms and satisfiability checking techniques.

a) *Traffic-Rule-Informed Trajectory Planning:* The use of formal methods allows autonomous vehicles to comply with high-level specifications and safely participate in traffic. These rules can be ensured, e.g., by reachability analysis [4], assume-guarantee contract formalisms [5], and partially by

The authors are with the Department of Informatics, Technical University of Munich, 85748 Garching, Germany.
yuanfei.lin@tum.de, althoff@tum.de

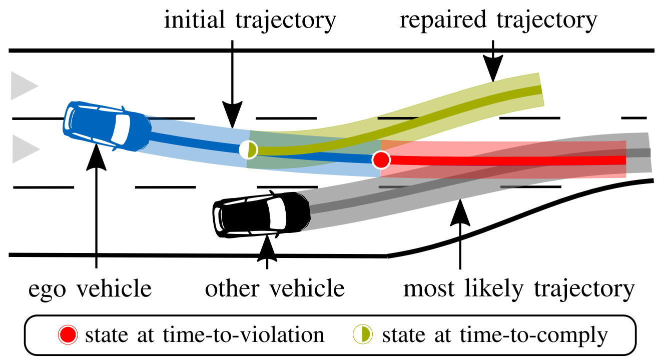


Fig. 1: Sketch of trajectory repairing regarding traffic rule violations. The initially-planned trajectory for the ego vehicle violates the traffic rule since it does not yield to vehicles entering the main carriageway from the access ramp. In our approach, only part of the initial trajectory is repaired to avoid the incompliance.

the responsibility-sensitive safety model [6]. To formalize the traffic rules in a precise and machine-readable manner, temporal logic is often used. Linear temporal logic (LTL) [1], [7], [8] can provide Boolean values for the satisfaction of rules. Metric temporal logic (MTL) [2], [9] extends LTL to support time intervals representing metric constraints. MTL is equipped with quantitative semantics, i.e., the robustness degree [10], [11], which indicates how far a behavior is from satisfying or violating a specification. In [12], the authors introduce the rulebook as a preordered set of rules to select preferred trajectories, which can be used for safety verification [13] and optimal control [14] in autonomous driving.

Trajectory planning with respect to specifications is computationally challenging due to the coupling of dynamical feasibility requirements and high-level specifications [15]. A large group of works uses automata-based approaches [16]–[18] or mixed-integer programming [19], [20] to develop plans that satisfy requirements described by temporal logic. However, common in these works is that they neither are computationally efficient nor take complex specifications and high-dimensional system dynamics into account.

b) *Satisfiability Checking:* Boolean satisfiability (SAT) is the problem of determining whether there exists an evaluation that satisfies a Boolean formula [21]. *Satisfiability modulo theories* (SMT) [22] extend this concept to general formulas by interpreting them within a certain formal theory $\mathcal{T}$ in first-order logic. One of the major approaches for implementing SMT solvers is the lazy approach [23], where a SAT-solving algorithm is integrated with a theory decision procedure ($\mathcal{T}$-solver). This method abstracts the input formula to a propositional one and feeds it to a SAT solver to suggest possible assignments. The $\mathcal{T}$-solver checks the satisfiability of the obtained assignment in a theory $\mathcal{T}$ ($\mathcal{T}$-consistency) to refine the formula and guide the SAT solver.

Satisfiability checking techniques have been successful in tackling system verification and combinatorial search problems. In [24], SMT solvers are used for identifying driving rule violations for autonomous vehicles. Shoukry *et. al.* [25] decompose the robot planning problem into smaller subproblems by leveraging the lazy SMT paradigm, which is extended to address LTL specifications in [26]. However, these works do not quantify the satisfaction of task specifications and cannot refine trajectories in dynamic environments efficiently. Using the lazy SMT framework, we aim at not only generating rule-compliant trajectories but also utilizing the robustness degree of specifications. In this regard, our approach is also inspired by the control synthesis described in [20], which includes the robustness degree of temporal logic specifications as an objective.

B. Contributions

We present the first work to repair trajectories violating traffic rules formalized in temporal logic. The repairability of trajectories can be automatically reasoned using the framework of lazy SMT solvers, i.e., whether and how a trajectory can be repaired to satisfy the traffic rules. Our contributions are as follows:

1) abstracting traffic rules to propositional logic formulas to exploit SAT solvers;
2) utilizing the robustness degree as heuristics in the SAT solver to efficiently find solutions;
3) defining assessment metrics in the $\mathcal{T}$-solver to check the $\mathcal{T}$-consistency of the solution yielded from the SAT solver and to determine the part of the trajectory to be repaired; and
4) applying continuous optimization methods to generate kinematically feasible, comfortable, and rule-compliant repaired trajectories.

The remainder of this paper is structured as follows: In Sec. II, required preliminaries and definitions are introduced. Sec. III provides an overview of our trajectory repairing approach. In Section IV, the lazy SMT-based trajectory repairing framework is described. We demonstrate the benefits of our method by case studies in Sec. V, followed by conclusions in Sec. VI.

II. Preliminaries and Problem Statement

A. System Description

We introduce discrete-time systems to model the dynamics of the ego vehicle, i.e., the vehicle to be controlled, as:

$$x_{k+1} = f_d(x_k, u_k), \tag{1}$$

where $x_k \in \mathbb{R}^n$ is the n-dimensional state, $u_k \in \mathbb{R}^m$ is the m-dimensional input, and $k \in \mathbb{N}_0$ is the discrete time step corresponding to the time $t_k = k\Delta t$, where $\Delta t \in \mathbb{R}^+$ is the time increment. Without loss of generality, the initial time step is 0 and the final time step is h. The system is subject to the set of admissible states $\mathcal{X}_k \subset \mathbb{R}^n$ and admissible control inputs $\mathcal{U}_k \subset \mathbb{R}^m$, each at time step k. We adhere to the notation $u([0,k])$ to denote input trajectories for the time interval $[0,k]$. The solution of (1) at time step k for an initial state x_0 and an input trajectory $u([0,k])$ is then denoted by the state trajectory $\chi\big(k, x_0, u([0,k])\big)$. A complete state trajectory for the time interval $[0,h]$ is abbreviated as χ.

Let $\square$ be a variable, we use $\square^{\mathtt{int}}$ and $\square^{\mathtt{rep}}$ to denote its initial and repaired values, respectively. The set $\mathcal{B}$ describes rule-relevant obstacles in the scenario. We adhere to the notation $\mathcal{O}_b(k) \subseteq \mathbb{R}^2$ to denote the occupancy of an obstacle $b \in \mathcal{B}$ at time step k. The environment model of the ego vehicle $\Omega := \langle \mathcal{L}, \mathcal{O}_\mathcal{B} \rangle$ consists of a road network $\mathcal{L}$ and the occupancy set $\mathcal{O}_\mathcal{B}$ of other traffic participants, which is the entire sequence of occupancies $\mathcal{O}_\mathcal{B}(k) = \bigcup_{b \in \mathcal{B}} \mathcal{O}_b(k)$.

B. Definitions

As motivated in Sec. I, we select the part of a rule-violating trajectory that remains unchanged until a cut-off state defined as:

Definition 1 (Cut-off State x_{cut}):
The cut-off state x_{cut} [3, Sec. III-A] is the state from which the repaired trajectory begins.

Let $\mathcal{X}_k^{\mathtt{CF}} = \mathcal{X}_k \setminus \mathcal{O}_\mathcal{B}(k)$ be the collision-free set of states at time step k, φ be one or multiple rules since one can combine the rules using conjunction, and $\chi \models \varphi$ denote that a trajectory χ complies with φ. The violation-free states are defined as:

Definition 2 (Violation-Free States $\mathcal{X}^{\mathtt{VF}}$):
The set $\mathcal{X}_k^{\mathtt{VF}}(\varphi) \subseteq \mathcal{X}_k^{\mathtt{CF}}$ is the set of collision-free states that additionally comply with traffic rules φ at time step k, i.e., $\mathcal{X}_k^{\mathtt{VF}}(\varphi) := \{x_k \in \mathcal{X}_k^{\mathtt{CF}} \mid x_k = \chi\big(k, x_0, u([0,k])\big) \wedge \chi\big([0,k], x_0, u([0,k])\big) \models \varphi\}$.

Definition 3 (Rule-Compliant Set $\mathcal{C}$):
Given traffic rules φ, the rule-compliant set $\mathcal{C}_k(\varphi) \subseteq \mathcal{X}_k^{\mathtt{VF}}(\varphi)$ at time step k contains all states from which feasible trajectories exist to remain rule-compliant for a finite time horizon h and is defined as $\mathcal{C}_k(\varphi) := \{x_k \in \mathcal{X}_k^{\mathtt{VF}}(\varphi) \mid \exists u([k,h]) : \chi\big([k,h], x_k, u([k,h])\big) \models \varphi\}$.

This definition is illustrated in Fig. 2. For brevity, we omit the φ-dependency in the notations for $\mathcal{X}^{\mathtt{VF}}$ and $\mathcal{C}$. With this, we introduce the following measures to find an appropriate x_{cut} from which the repaired trajectory branches off:

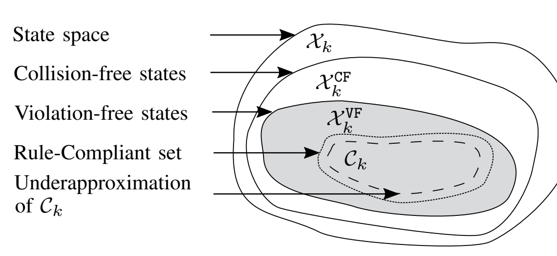


Fig. 2: Relation of state space, collision-free states, violation-free states, and rule-compliant set at time step k.

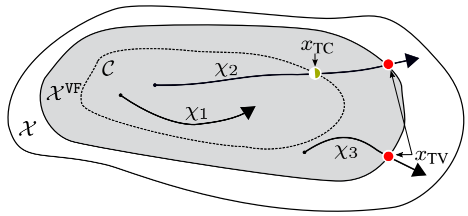


Fig. 3: Violation properties of trajectories. Only trajectory χ_1 continuously complies with traffic rules. Both trajectory χ_2 and χ_3 violate traffic rules at time step TV. Trajectory χ_2 leaves the rule-compliant set $\mathcal{C}$ at TC.

Definition 4 (Time-To-Violation):
The time-to-violation (TV) is the first time step at which the trajectory originating from the initial input $u^{\mathtt{int}}([0,h])$ leaves the set of violation-free states $\mathcal{X}^{\mathtt{VF}}$:

$$TV := \min\{k \in \mathbb{N}_0 \mid \chi\big(k, x_0, u^{\mathtt{int}}([0,k])\big) \notin \mathcal{X}_k^{\mathtt{VF}}\}.$$

If no violation is detected, i.e., all states are within $\mathcal{X}^{\mathtt{VF}}$, we set $TV = \infty$.

Definition 5 (Time-To-Comply):
Assuming that $x_0 \in \mathcal{X}_0^{\mathtt{VF}}$, the time-to-comply (TC) is the last time step for which a rule-compliant trajectory exists:

$$TC := \max\{k \in [0, TV] \mid \chi\big(k, x_0, u^{\mathtt{int}}([0,k])\big) \in \mathcal{C}_k\}.$$

Note that TC is set to $-\infty$ in case there exists no maneuver to avoid the rule violation and $TC = TV = \infty$ if no violation occurs.

As a result, we use the state at TC as the cut-off state, i.e., $x_{\mathrm{cut}} := x_{\mathrm{TC}}$. Fig. 3 shows the violation properties of different trajectories and indicates the states at time steps TV and TC.

C. Metric Temporal Logic

Given formulas φ, φ_1, and φ_2, the logical *True* $\top$, a propositional variable σ presenting a Boolean statement, an associated interval I of $\mathbb{N}_0$, the Boolean *negation* and *disjunction* operator $\neg$ and $\vee$, and the temporal *since* and *until* operator $\mathbf{S}_I$ and $\mathbf{U}_I$, MTL syntax is defined as [9]:

$$\varphi := \top \mid \sigma \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \mathbf{S}_I \varphi_2 \mid \varphi_1 \mathbf{U}_I \varphi_2. \tag{2}$$

The semantics of *since* and *until* operators is equivalent to the following first-order logic expressions [27, Sec. 1]:

$$\begin{aligned}\varphi_1 \mathbf{S}_I \varphi_2 &\Leftrightarrow \exists k' \in (k-I) \cap \mathbb{N}_0 : (\varphi_2 \wedge \forall k'' \in (k', k] : \varphi_1),\\ \varphi_1 \mathbf{U}_I \varphi_2 &\Leftrightarrow \exists k' \in (k+I) \cap \mathbb{N}_0 : (\varphi_2 \wedge \forall k'' \in [k, k') : \varphi_1).\end{aligned} \tag{3}$$

For ease of notation, I is dropped from the grammar when considering the total signal domain. According to [27, Sec. 2.1], we use the following symbols and operators for our convenience:

$$\begin{aligned}\bot &:= \neg\top \\ \varphi_1 \wedge \varphi_2 &:= \neg(\neg\varphi_1 \vee \neg\varphi_2) \\ \varphi_1 \Rightarrow \varphi_2 &:= \neg\varphi_1 \vee \varphi_2\end{aligned} \qquad \begin{aligned}\mathbf{O}_I\varphi &:= \top \mathbf{S}_I \varphi \\ \mathbf{P}\varphi &:= \bot \mathbf{S} \varphi \\ \mathbf{F}_I\varphi &:= \top \mathbf{U}_I \varphi \\ \mathbf{G}_I\varphi &:= \neg\mathbf{F}_I \neg\varphi,\end{aligned} \tag{4}$$

where $\mathbf{O}_I$, $\mathbf{P}$, $\mathbf{F}_I$, and $\mathbf{G}_I$ are the *once*, *previously*, *finally* (aka *eventually*), *globally* (aka *always*) temporal operators, respectively.

We denote the robustness degree of φ with respect to a trajectory χ at time step k as $\rho(\varphi, \chi, k)$, which is defined as [10, Def. 22]:

$$\begin{aligned}\rho(\top, \chi, k) &:= \infty\\ \rho(\sigma, \chi, k) &:= \mathbf{Dist}_{\mathcal{E}}(x_k, \mathcal{X}_k^{\mathtt{VF}}(\sigma))\\ \rho(\neg\varphi, \chi, k) &:= -\rho(\varphi, \chi, k)\\ \rho(\varphi_1 \vee \varphi_2, \chi, k) &:= \max(\rho(\varphi_1, \chi, k), \rho(\varphi_2, \chi, k))\\ \rho(\varphi_1 \mathbf{S}_I \varphi_2, \chi, k) &:= \max_{k' \in (k-I) \cap \mathbb{N}_0} \big(\min\big(\rho(\varphi_2, \chi, k'),\\ &\qquad \min_{k'' \in (k', k]} \rho(\varphi_1, \chi, k'')\big)\big)\\ \rho(\varphi_1 \mathbf{U}_I \varphi_2, \chi, k) &:= \max_{k' \in (k+I) \cap \mathbb{N}_0} \big(\min\big(\rho(\varphi_2, \chi, k'),\\ &\qquad \min_{k'' \in [k, k')} \rho(\varphi_1, \chi, k'')\big)\big),\end{aligned} \tag{5}$$

where the signed distance $\mathbf{Dist}_{\mathcal{E}}$ is defined based on a metric $\mathcal{E}$ (typically the Euclidean distance) as:

$$\mathbf{Dist}_{\mathcal{E}}(x, \mathcal{X}^{\mathtt{VF}}) := \begin{cases} -\inf\{\mathcal{E}(x, x') | x' \in \mathcal{X}^{\mathtt{VF}}\} & \text{if } x \notin \mathcal{X}^{\mathtt{VF}} \\ \inf\{\mathcal{E}(x, x') | x' \in \mathcal{X} \setminus \mathcal{X}^{\mathtt{VF}}\} & \text{if } x \in \mathcal{X}^{\mathtt{VF}}. \end{cases} \tag{6}$$

D. Davis-Putnam-Logemann-Loveland Algorithm

The Davis-Putnam-Logemann-Loveland (DPLL) algorithm [28] is often used in SAT solvers to check the satisfiability of abstracted Boolean propositional formulas in the SMT framework (aka DPLL($\mathcal{T}$) [29]). To streamline the notation, we write φ^p and σ^p to denote φ and σ after propositional abstraction as input for the DPLL algorithm, respectively. The propositional formula also needs to be in conjunctive normal form (CNF) as $\bigwedge_i \bigvee_j (\neg)\sigma_{i,j}$, i.e, a conjunction of clauses that are disjunctions of literals, where a literal is either a positive or negative atomic proposition σ. If all individual clauses are satisfied (`SAT`) by partial variable assignments, i.e., only the values of some literals are fixed, the entire formula is solved as `SAT` and the DPLL algorithm constructs a partial satisfying solution ϕ.

E. Problem Statement

For rule-compliant trajectory repairing, we use traffic rules formalized in MTL. To exploit SMT solvers, we need to eliminate temporal operators in the rule to obtain first-order logic formulas [23, Sec. 2.1], which can be achieved by instantiating it for each time step according to (3) since quantifiers with finite intervals are equivalent to logical conjunction or disjunction of instances. As the number of satisfiable instances for existential quantification grows exponentially with the time horizon, it is generally computationally expensive to enumerate all satisfying possibilities [30, Sec. 9.5]. We only address the rules starting with temporal operators that can be expressed by universal quantifiers in this work, i.e., the operator $\mathbf{G}$ and its equivalents, and leave the other rules to future work with a possible integration of specification-compliant reachable sets [4]. For $\mathbf{G}\varphi$ to be satisfied, we need φ to be satisfied at all time steps. Thus, $\mathbf{G}\varphi$ is equivalent to $[\![\varphi]\!]_{[0,h]} := [\![\varphi]\!]_0 \wedge \ldots \wedge [\![\varphi]\!]_k \wedge \ldots \wedge [\![\varphi]\!]_h = \top$, where $[\![\varphi]\!]_k$ is the valuation of φ at time step k. Although we restrict ourselves to some temporal operators, all the interstate rules for autonomous vehicles driving on German highways in [2] can be considered by our approach.

III. Overall Algorithm

Alg. 1 summarizes our approach for rule-compliant trajectory repair. We assume that our method receives as input an initially-planned trajectory $\chi^{\mathtt{int}}$ for the ego vehicle and an MTL monitor $\mathfrak{M}$ which is constructed by our previous works [2], [11] with the environment model Ω to evaluate traffic rules $\mathbf{G}\varphi$. Our algorithm outputs a repaired trajectory that complies with $\mathbf{G}\varphi$.

As a first step, we run the monitor to obtain TV and the robustness degree of $\chi^{\mathtt{int}}$ at all time steps which is denoted by $\rho_{\varphi}^{\mathtt{int}}$ (see line 1). If $\chi^{\mathtt{int}}$ violates the traffic rule $\mathbf{G}\varphi$ and TV $\neq 0$, φ is abstracted to φ^p in CNF (see line 3), which is later explained in Sec. IV-A. Our approach is iterative using the DPLL($\mathcal{T}$) algorithm with a lazy SMT propagation. At each iteration, we start by checking the Boolean satisfiability of φ^p for the time interval $[\mathrm{TV}, h]$ using DPLL-based SAT solvers (see line 4). In case the result is `SAT`, we simultaneously obtain a satisfying solution ϕ for φ^p (see line 5). Then the $\mathcal{T}$-solver reasons about the repairability $\mathfrak{r} \in \mathbb{B}$ of ϕ (see line 6). If $\mathfrak{r}$ is equal to $\top$, we return the corresponding repaired trajectory $\chi^{\mathtt{rep}}$ (see line 8). Otherwise, we update φ^p by treating ϕ as a conflicting clause $\neg\phi$ in the future runs (see line 10). We repeat this process (lines 4-11) until a feasible trajectory is found or the SAT solver returns `UNSAT`. In the latter case, we can execute a minimum-violation trajectory [17] or a fail-safe trajectory [31], which is however not the focus of this work.

Algorithm 1 ruleCompliantTrajectoryRepairing

Input: initial trajectory $\chi^{\mathtt{int}}$, traffic rule monitor $\mathfrak{M}$, rule $\mathbf{G}\varphi$
Output: repaired trajectory $\chi^{\mathtt{rep}}$

1: TV, $\rho_{\varphi}^{\mathtt{int}} \leftarrow \mathfrak{M}$.evaluate($\chi^{\mathtt{int}}$, $\mathbf{G}\varphi$)
2: **if** TV $\notin \{0, \infty\}$ **then**
3:     $\varphi^p \leftarrow$ abstractTrafficRule($\mathbf{G}\varphi$) ▷ Sec. IV-A
4:     **while** SAT.check(φ^p, $\rho_{\varphi}^{\mathtt{int}}$, TV) == `SAT` **do** ▷ Sec. IV-B
5:         $\phi \leftarrow$ SAT.solution( )
6:         $\mathfrak{r}$, $\chi^{\mathtt{rep}} \leftarrow \mathcal{T}$-solver.check($\phi$, $\mathfrak{M}$, $\rho_{\varphi}^{\mathtt{int}}$) ▷ Sec. IV-C
7:         **if** $\mathfrak{r} == \top$ **then**
8:             **return** $\chi^{\mathtt{rep}}$
9:         **else**
10:             $\varphi^p \leftarrow \varphi^p \wedge \neg\phi$
11:         **end if**
12:     **end while**
13: **end if**
14: **return** $\emptyset$

IV. SMT-based Trajectory Repairing

In this section, we apply the lazy DPLL($\mathcal{T}$)-based SMT paradigm. We start by abstracting the traffic rules formalized in MTL to propositional logic formulas. Afterward, the SAT solver and the trajectory-repairing framework in the $\mathcal{T}$-solver are introduced.

A. Propositional Logic Formula Abstraction

The formalization of traffic rules and the robustness degree definition of predicates are based on [2], [11]. To provide a more intuitive understanding of the rules, they can be rewritten as request-response requirements [32] containing at least one implication operator. Thus, when a rule is violated, a true antecedent implies a false consequent ($\top \Rightarrow \bot$).

Running example: Consider the general traffic rule R_G1 from [2], i.e., keeping a safe distance to the preceding vehicle, which is reformulated as:

$$\begin{aligned}\mathbf{G}\varphi = \mathbf{G}\big(&\mathrm{in\_same\_lane}(x_{\mathrm{ego}}, x_b) \wedge \mathrm{in\_front\_of}(x_{\mathrm{ego}}, x_b) \wedge \\ &\neg\mathbf{O}_{[0,t_c]}\big(\mathrm{cut\_in}(x_b, x_{\mathrm{ego}}) \wedge \mathbf{P}\big(\neg\mathrm{cut\_in}(x_b, x_{\mathrm{ego}})\big)\big) \\ &\Rightarrow \mathrm{keeps\_safe\_distance\_prec}(x_{\mathrm{ego}}, x_b)\big),\end{aligned} \tag{7}$$

where x_b is the state of the rule-relevant vehicle $b \in \mathcal{B}$, t_c is the recovery time for the safe distance violation caused by a cut-in maneuver of b, and $\mathrm{in\_same\_lane}$ (whether two vehicles are in the same lane), $\mathrm{in\_front\_of}$ (whether b is in front of the ego vehicle), $\mathrm{cut\_in}$ (whether b enters the lane of ego vehicle), and $\mathrm{keeps\_safe\_distance\_prec}$ (whether the ego vehicle keeps a safe distance to b) are the predicates.

When investigating SMT solvers, we first need to eliminate the temporal operator $\mathbf{G}$ by instantiating $\mathbf{G}\varphi$ to $[\![\varphi]\!]_{[0,h]} = \top$ and abstract φ to a propositional logic formula φ^p. The latter can be achieved by replacing the predicates and the elements starting with temporal operators within φ with propositional variables σ^p. With this, φ^p is then deduced to an equivalent formula in CNF, which is trivial and can be, e.g., performed by the *Tseitin* transformation [33].

Running example: If (7) is violated with respect to the traffic participant b', φ is abstracted to a propositional formula φ^p in CNF as:

$$\begin{aligned}\varphi^p := &\underbrace{\mathrm{in\_same\_lane}(x_{\mathrm{ego}}, x_{b'})}_{\sigma_1^p} \wedge \underbrace{\mathrm{in\_front\_of}(x_{b'}, x_{\mathrm{ego}})}_{\sigma_2^p} \wedge \\ &\neg\underbrace{\mathbf{O}_{[0,t_c]}\big(\mathrm{cut\_in}(x_{b'}, x_{\mathrm{ego}}) \wedge \mathbf{P}\big(\neg\mathrm{cut\_in}(x_{b'}, x_{\mathrm{ego}})\big)\big)}_{\sigma_3^p} \\ &\Rightarrow \underbrace{\mathrm{keeps\_safe\_distance\_prec}(x_{\mathrm{ego}}, x_{b'})}_{\sigma_4^p} \\ \equiv &\neg\sigma_1^p \vee \neg\sigma_2^p \vee \sigma_3^p \vee \sigma_4^p,\end{aligned} \tag{8}$$

of which the violating assignments at time step k can only be $[\![\sigma_1^p]\!]_k = \top$, $[\![\sigma_2^p]\!]_k = \top$, $[\![\sigma_3^p]\!]_k = \bot$, and $[\![\sigma_4^p]\!]_k = \bot$.

B. SAT Solver

After obtaining the abstracted formula, we can use the DPLL algorithm to solve the Boolean satisfiability of φ^p for the time interval $[\mathrm{TV}, h]$. To reduce computational load, we assume that the assignment of selected propositions keeps unchanged within $[\mathrm{TV}, h]$. If this assumption does not hold in a rare case, our approach fails to find a feasible solution and we will execute a fail-safe maneuver as presented in, e.g., [31]. As the robustness degree captures how close a trajectory comes to reaching the violation or satisfaction of the rule, we can utilize it to determine the sequential order of branching atomic propositions in the DPLL algorithm. For better comparability of different robustness values, we normalize them to the interval $[-1, 1]$ as described in [11, Sec. IV-A] and only compare their values at TV. Afterward, to select the least robust proposition, the atomic propositions are sorted in an ascending order based on their absolute robustness degree obtained from the initial trajectory.

TABLE I: Description of the assessment metric for compliant maneuvers.

Metric	Description	Category [4]	Predicate [2]
Time-to-brake (TTB)	Full braking with maximum deceleration.	**Longitudinal position, Velocity**	in_front_of, keeps_safe_distance_prec, keeps_lane_speed_limit, keeps_fov_speed_limit, keeps_type_speed_limit, ...
Time-to-kick-down (TTK)	Full accelerating until reaching the maximum velocity and then maintaining the velocity.		
Time-to-steer (TTS)	Full steering to reach a certain lateral offset.	**Lateral position**	in_same_lane, left_of, drives_rightmost, ...
Time-to-maintain-velocity (TTMV)	Maintaining a steady velocity.	**Acceleration**	brakes_abruptly, ...

Running example: Assuming the sequence of the atomic propositions in (8) is $(\sigma_1^p, \sigma_3^p, \sigma_2^p, \sigma_4^p)$ based on the robustness degree evaluation, the first obtained solution from the DPLL algorithm is determined as $[\![\sigma_1^p]\!]_{[\mathrm{TV},h]} = \bot$.

C. $\mathcal{T}$-Solver

In the $\mathcal{T}$-solver, the $\mathcal{T}$-consistency of ϕ needs to be determined regarding the environment model Ω and the system dynamics of the ego vehicle, which is summarized in Alg. 2. To address this, we first obtain propositions ϕ_r to be repaired by comparing the valuations of atomic propositions in ϕ with the violating assignments for $[\![\varphi^p]\!]_{[\mathrm{TV},h]} = \top$, i.e., only the atomic propositions with a different value in ϕ are considered (see line 1). After determining the TC of ϕ_r (cf. Sec. IV-C.1), we formulate a continuous optimization problem to check whether feasible repaired trajectories can be obtained (cf. Sec. IV-C.2).

Running example: If the obtained solution ϕ from the SAT solver is $[\![\sigma_1^p \wedge \sigma_4^p]\!]_{[\mathrm{TV},h]} = \top$, we can obtain the proposition to be repaired as $\phi_r = \sigma_4^p$ since $[\![\sigma_1^p]\!]_{[\mathrm{TV},h]}$ is already equal to $\top$ (cf. (8)).

1) Time-To-Comply Search: Calculating the TC for establishing the cut-off state is challenging since all possible maneuvers must be evaluated. Similar to the calculation of time-to-react in [3], we underapproximate the TC (cf. Fig. 2) by using a point-mass vehicle model [34] and focusing only on the predicates contained in ϕ_r. To achieve this, we introduce assessment metrics based on the category of predicates from [4], which are listed in Tab. I and illustrated in Fig. 4.

The initial values of TC and $\mathfrak{r}$ are set to $-\infty$ and $\bot$, respectively (see line 2 in Alg. 2). Next, we automatically obtain the compliant maneuvers according to Tab. I based on the category of predicates in ϕ_r (see line 3). Afterward, we use binary search to detect the maximum remaining time for executing rule-compliant maneuvers in the function

Algorithm 2 $\mathcal{T}$-SOLVER.CHECK

Input: solution from the SAT solver ϕ, $\mathfrak{M}$, $\rho_{\varphi}^{\mathtt{int}}$
Output: repairability $\mathfrak{r}$, repaired trajectory $\chi^{\mathtt{rep}}$

1: $\phi_r \leftarrow$ OBTAINREPAIREDPROPOSITIONS(ϕ, $\rho_{\varphi}^{\mathtt{int}}$)
2: TC $\leftarrow -\infty$, $\mathfrak{r} \leftarrow \bot$
3: $\mathcal{M} \leftarrow$ SETCOMPLIANTMANEUVERS(ϕ_r)
4: TC $\leftarrow$ SEARCHTC($\mathcal{M}$, $\mathfrak{M}$)
5: **if** TC $\neq -\infty$ **then**
6: $\quad\chi^{\mathtt{rep}} \leftarrow$ OPTIMIZATIONBASEDREPAIR($\rho_{\varphi}^{\mathtt{int}}$, TC, ϕ, ϕ_r)
7: $\quad\mathfrak{r} \leftarrow$ CHECKRULECOMPLIANCE($\mathfrak{M}$, $\chi^{\mathtt{rep}}$)
8: **end if**
9: **return** $\mathfrak{r}$, $\chi^{\mathtt{rep}}$

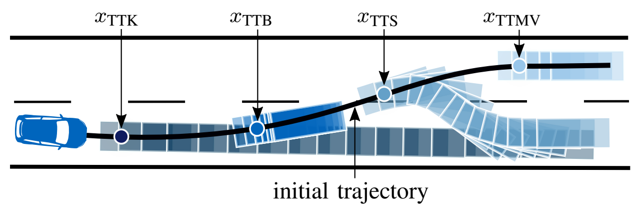


Fig. 4: Illustration of different compliant maneuvers. The points indicate the start of the corresponding maneuvers.

SEARCHTC($\cdot$), which is adapted from [35, Alg. 2] by setting violation-free as search conditions (see line 4).

Running example: The assessment metrics for repairing the assignment of σ_4^p consist of TTB and TTK according to Tab. I since it contains the predicate keeps_safe_distance_prec, which belongs to the longitudinal position category.

2) Optimization-based Trajectory Repairing: If the TC is finite, we specify an optimization problem for generating repaired trajectories (see line 6). To ensure a fast convergence to the optimal solution, we use convex linear-quadratic programs [36, Sec. 4.4], similarly to [31]. The motion (cf. (1)) starting from the cut-off state is separated into longitudinal and lateral components $(x_{\mathrm{lon}}, x_{\mathrm{lat}})^T$ in a curvilinear coordinate system [37] aligned with a predefined reference path Γ, which is typically obtained from a high-level route planner. The longitudinal state $x_{\mathrm{lon}} = (s, v, a, j)^T$ consists of position s, velocity v, acceleration a, and jerk j. The lateral motion is described by $x_{\mathrm{lat}} = (d, \theta, \kappa, \dot{\kappa})^T$, where d is the lateral distance to Γ, θ is the orientation, κ is the curvature, and $\dot{\kappa}$ is the change of curvature.

a) Cost Function: The optimization problem for both longitudinal and lateral motions is to minimize a quadratic cost function J for all $k \in \{\mathrm{TC}, \dots, h\}$, which comprises a performance term J_p and a robustness term J_r. J_p focuses on the trajectory quality concerning control and smoothing cost, using the definitions in [31, (12) and (18)] to achieve comfortable motions. In contrast, J_r is chosen to increase the robustness of rule compliance. However, the robustness degree of the entire traffic rule formula is nonconvex and nondifferentiable in general (cf. (5) and [2]), which makes online optimization a challenge [38]. To remedy this, we relax the problem inspired by the nature of min and max functions and optimize only the robustness degree of ϕ_r. In order to preserve convexity for any predicate, a positive quadratic robustness term is chosen with weight $\omega_r \in \mathbb{R}^+$ as:

$$J_r(x, \hat{x}) = \omega_r \sum_{k=\mathrm{TC}}^{h} (x_k - \hat{x}_k)^2, \tag{9}$$

which is to keep the result close to the reference state $\hat{x}$ with a sufficiently large robustness degree greater than a predefined threshold $\epsilon_r \in \mathbb{R}^+$, i.e., $|\rho(\phi_r, \hat{x}, k)| \geq \epsilon_r$.

b) Constraints: The repaired trajectory must simultaneously adhere to a set of system and rule constraints. Assuming the latter can be formulated as linear or at least linearizable constraints according to [4, Sec. III], they can be addressed by adding lower and upper bounds of states and inputs based on the definition and assignment of propositions. Otherwise, we relax the requirement and check the rule compliance after the optimization process (see line 7). All atomic propositions have the same assignments from TC to TV as the initial trajectory. In contrast, the assignment of ϕ is used for constructing rule constraints in the remaining time steps since φ^p is checked as SAT by ϕ in the SAT solver (cf. Sec. IV-B).

Running example: If ϕ is $[\![\sigma_4^p]\!]_{[\mathrm{TV},h]} = \top$, rule constraints for all $k \in [\mathrm{TV}, h]$ can be formed as:

$$s_k \in (-\infty, \mathrm{rear}(x_{b',k}) - \Delta_{\mathrm{safe}}(v_k, x_{b',k})], \tag{10}$$

where Δ_{safe} is the safe distance defined in [31, (4)] and $\mathrm{rear}(\cdot)$ is the position of the rear bumper of a vehicle.

V. Case Studies

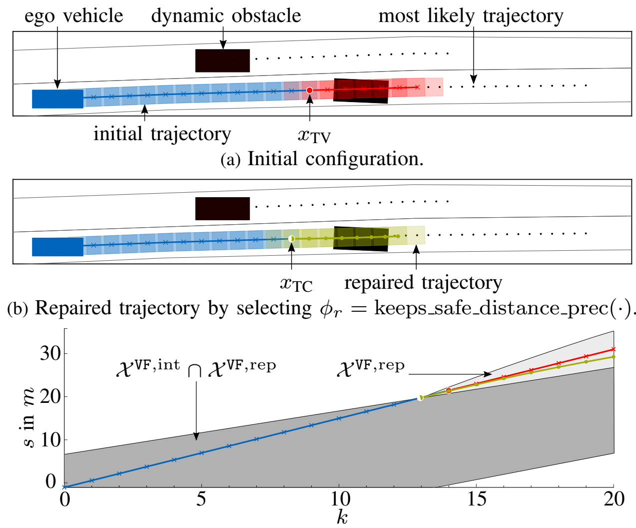
This section shows the applicability and efficacy of our rule-compliant trajectory repairing approach to traffic scenarios from the CommonRoad benchmark suite[1] [34] and the highD dataset [39]. In our implementation, we use the convex programming package CVXPY [40] and the solver OSQP [41] to model the trajectory optimization problem. All approaches are implemented in *Python* on a computer with an Intel Core i7-1165G7 CPU and 16 GB of memory. The parameters for the traffic rule evaluation are obtained from [2], [11]. We set the planning horizon h to $2s$ with a time increment $\Delta t = 0.1s$. The animation of the evaluation can be found at https://mediatum.ub.tum.de/1641743.

A. Keeping a Safe Distance to the Preceding Vehicle (R_G1)

We first evaluate a rural scenario[2] where the initial trajectory of the ego vehicle violates the safe distance rule (cf. rule R_G1 in (7)) starting from TV = 14. Fig. 5a shows the initial configuration of the scenario and the initial robustness degrees of the atomic propositions within the time interval $[\mathrm{TV}, h]$ are listed in Tab. II. The occupancy of the ego vehicle that violates the rule is marked in red. The first obtained solution from the SAT solver is $[\![\sigma_4^p]\!]_{[\mathrm{TV},h]} = \top$. In the $\mathcal{T}$-solver, we check the $\mathcal{T}$-consistency of σ_4^p and obtain TC = 13 using TTB (cf. Tab. I). Afterward, we obtain the repaired trajectory with convex linear-quadratic programs. As illustrated in Fig. 5b, the ego occupancy along the repaired trajectory is marked in green where a braking maneuver is executed after the cut-off state to enlarge the violation-free area. We visualize $\mathcal{X}^{\mathrm{VF}}$ in the k-s plane together with the initial and repaired trajectory in Fig. 5c.

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

[2] CommonRoad ID: DEU_Gar-1_1_T-1



(a) Initial configuration.

(b) Repaired trajectory by selecting $\phi_r = \mathrm{keeps\_safe\_distance\_prec}(\cdot)$.

(c) Comparison of $\mathcal{X}^{\mathrm{VF}}$ projected onto the k-s plane.

Fig. 5: Rural scenario in which the ego vehicle violates the rule R_G1.

B. Avoiding Unnecessary Braking (R_G2)

Let us consider the unnecessary braking rule (R_G2) from [2], i.e., braking abruptly ($a < a_{\mathrm{abrupt}}$) is not allowed without justification (violation of safe distance or its preceding vehicle brakes abruptly), where $a_{\mathrm{abrupt}} \in \mathbb{R}^-$ denotes the predefined acceleration threshold. The rule can be reformulated based on the modification in [11, Tab. I] as:

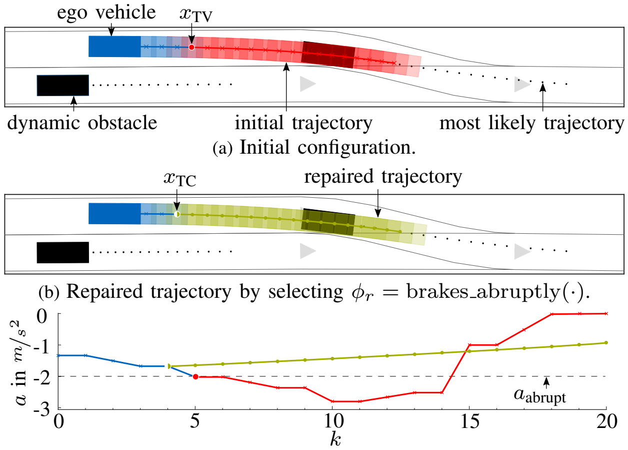
$$\mathbf{G}\varphi = \mathbf{G}\big(\underbrace{\mathrm{brakes\_abruptly}(x_{\mathrm{ego}})}_{\sigma_1^p} \Rightarrow \underbrace{\mathrm{braking\_justification}(x_{\mathrm{ego}}, \Omega)}_{\sigma_2^p}\big), \tag{11}$$

where the predicate $\mathrm{brakes\_abruptly}$ specifies whether a vehicle brakes abruptly and $\mathrm{braking\_justification}$ is a general traffic situation predicate that indicates whether abrupt braking is allowed in the current environment. Fig. 6a depicts a lane-merging scenario[3], in which the ego vehicle brakes abruptly. However, this is unnecessary since $\mathrm{braking\_justification}(x_{\mathrm{ego}})$ is $\perp$ detected by the traffic rule monitor. Thus, the initially-planned trajectory violates R_G2 starting from time step TV = 5 and needs repair. After abstracting the rule to $\varphi^p = \neg\sigma_1^p \vee \sigma_2^p$ and running the SAT

TABLE II: Robustness degree of the atomic propositions obtained from $\chi^{\mathtt{int}}$ at time step TV, the corresponding TC, and average computation times of evaluated scenarios. A dash denotes that the value is not needed.

	Scenario I		**Scenario II**		**Scenario III**	
Proposition	**Rob.**	**TC**	**Rob.**	**TC**	**Rob.**	**TC**
σ_1	0.1132	—	**0.0017**	4	**−0.0026**	14
σ_2	0.0695	—	−0.0017	—	1.0000	—
σ_3	−0.0249	—			0.5175	—
σ_4	**−0.0027**	13			0.4167	—
Computation Time in ms						
Repairing	223		168		147	
Replanning	887		399		398	

[3] CommonRoad ID: ZAM_Zip-1_56_T-1



(a) Initial configuration.

(b) Repaired trajectory by selecting ϕ_r = brakes_abruptly$(\cdot)$.

(c) Initial and repaired acceleration profile.

Fig. 6: Lane-merging scenario where the ego vehicle violates the rule R_G2.

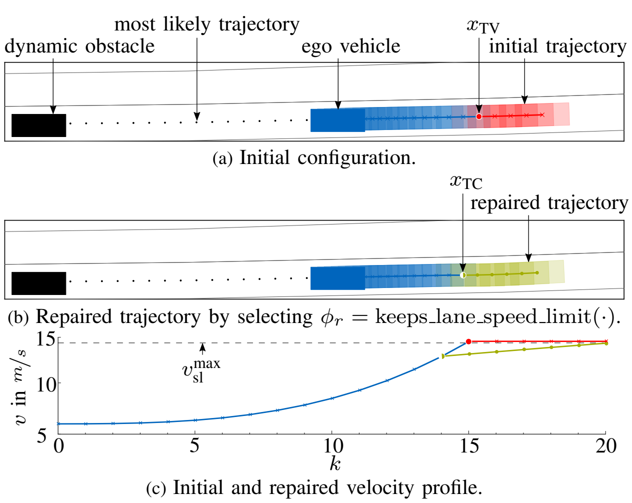
solver, we obtain a satisfying solution for $[\![\varphi^p]\!]_{[\text{TV},h]} = \top$ as $[\![\sigma_1^p]\!]_{[\text{TV},h]} = \bot$. In the $\mathcal{T}$-solver, TC is computed as 4 using TTMV (cf. Tab. I). As illustrated in Fig. 6b, we obtain a rule-compliant trajectory with which the ego vehicle keeps a more reasonable deceleration than the initial trajectory (see Fig. 6c).

C. Adhering to the Speed Limit (R_G3)

Next, we present the rule for limiting maximum driving velocity (R_G3) in [2] as:

$$
\begin{aligned}
\mathbf{G}\varphi = \mathbf{G}\big(&\text{keeps\_lane\_speed\_limit}(x_{\text{ego}})\wedge \\
&\text{keeps\_type\_speed\_limit}(x_{\text{ego}})\wedge \\
&\text{keeps\_fov\_speed\_limit}(x_{\text{ego}})\wedge \qquad (12)\\
&\text{keeps\_braking\_speed\_limit}(x_{\text{ego}})\big) \\
\equiv \mathbf{G}(\top \Rightarrow &\sigma_1^p \wedge \sigma_2^p \wedge \sigma_3^p \wedge \sigma_4^p) \equiv \mathbf{G}(\sigma_1^p \wedge \sigma_2^p \wedge \sigma_3^p \wedge \sigma_4^p),
\end{aligned}
$$

with which the ego vehicle is not allowed to exceed 1) the speed limit of driving lanes $v_{\text{sl}}^{\max} \in \mathbb{R}_0$ (keeps_lane_speed_limit), 2) the maximum velocity allowed for the vehicle type $v_{\text{type}} \in \mathbb{R}_0$ (keeps_type_speed_limit), 3) the speed limit to ensure enough field of view $v_{\text{fov}} \in \mathbb{R}_0$ (keeps_fov_speed_limit), and 4) the speed for comfortable braking $v_{\text{br}} \in \mathbb{R}_0$ (keeps_braking_speed_limit). We demonstrate the repairing process as for R_G3 with an urban scenario[4] as shown in Fig. 7. After using our approach, smooth, comfortable, and rule-compliant trajectories are generated to avoid the ego vehicle exceeding the speed limit.



(a) Initial configuration.

(b) Repaired trajectory by selecting ϕ_r = keeps_lane_speed_limit$(\cdot)$.

(c) Initial and repaired velocity profile.

Fig. 7: Urban scenario in which the ego vehicle violates the rule R_G3.

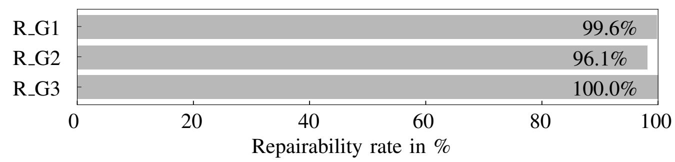


Fig. 8: Repairability of the evaluated trajectories.

D. Performance Evaluation

We compare our approach to trajectory replanning using a sampling-based trajectory planner [42], which computes trajectories as jerk-optimal quintic polynomials. The MTL monitor $\mathfrak{M}$ evaluates each sampled polynomial and the violation-free one with the minimum cost is selected as the optimal trajectory. According to the computation times in Tab. II, replanning is more computationally expensive than our approach since each sampled trajectory needs evaluation, and rule-compliant trajectories often have low priority due to high input costs.

Furthermore, we use the highD dataset [39] to test our approach on over 1,000 rule-violating trajectories obtained using $\mathfrak{M}$, where each lasts several seconds and is rule-compliant for the initial time step. The evaluation result for rules R_G1-R_G3 is shown in Fig. 8. Since highD scenarios are non-interactive, i.e., other traffic participants do not react to the ego vehicle, we do not count the rear-end collisions caused by other vehicles as a rule violation for the ego vehicle. After implementing our algorithm, over 95% of the trajectories can be repaired to comply with the traffic rules. The irreparability of the rest is either caused by the initial state being already outside the rule-compliant set or by our method relaxing the constraints (cf. Sec. IV-C.2.b).

VI. Conclusions

This paper proposes a novel concept for generating rule-compliant trajectories for autonomous vehicles based on a trajectory-repairing framework. Unlike most existing studies on motion planning with specifications, our approach can not only bridge temporal logic formulas with satisfiability checking technologies, but it can also reuse the rule-violating planned result to generate repaired trajectories efficiently. In addition, the MTL robustness degree is utilized as a heuristic for the SMT paradigm and an optimization objective for traffic rule satisfaction. We demonstrated the benefits of our rule-compliant trajectory repairing algorithm with German interstate rules in real traffic scenarios and compare the results by replanning the entire trajectory. Future work will focus on improving the accuracy of the TC using reachability analysis, learning the robustness degree from data, and extending the current framework to cooperative driving scenarios.

[4]CommonRoad ID: DEU_Muc-4_2_T-1

Acknowledgments

The authors gratefully acknowledge partial financial support by the German Federal Ministry for Digital and Transport (BMDV) within the project *Cooperative Autonomous Driving with Safety Guarantees* (KoSi).

References

[1] A. Rizaldi, F. Immler, B. Schürmann, and M. Althoff, "A formally verified motion planner for autonomous vehicles," in *Proc. of the Int. Symposium on Automated Technology for Verification and Analysis*, 2018, pp. 75–90.

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

[3] Y. Lin, S. Maierhofer, and M. Althoff, "Sampling-based trajectory repairing for autonomous vehicles," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2021, pp. 572–579.

[4] E. Irani Liu and M. Althoff, "Computing specification-compliant reachable sets for motion planning of automated vehicles," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2021, pp. 1–8.

[5] T. Phan-Minh, K. X. Cai, and R. M. Murray, "Towards assume-guarantee profiles for autonomous vehicles," in *Proc. of the IEEE Conf. on Decision and Control*, 2019, pp. 2788–2795.

[6] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2018.

[7] T. Wongpiromsarn, S. Karaman, and E. Frazzoli, "Synthesis of provably correct controllers for autonomous vehicles in urban environments," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2011, pp. 1168–1173.

[8] K. Esterle, L. Gressenbuch, and A. Knoll, "Formalizing traffic rules for machine interpretability," in *Proc. of the IEEE Connected and Automated Vehicles Symposium*, 2020, pp. 1–7.

[9] R. Alur and T. A. Henzinger, "Real-time logics: complexity and expressiveness," *Information and Computation*, vol. 104, no. 1, pp. 35–77, 1993.

[10] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.

[11] L. Gressenbuch and M. Althoff, "Predictive monitoring of traffic rules," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2021, pp. 915–922.

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

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

[14] W. Xiao, N. Mehdipour, A. Collin, A. Y. 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 Systems*, 2021, pp. 143–154.

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

[16] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning for dynamical systems," in *Proc. of the IEEE Conf. on Decision and Control held jointly with Chinese Control Conf.*, 2009, pp. 5997–6004.

[17] L. I. R. Castro, P. Chaudhari, J. Tumová, 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.

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

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

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

[21] A. Biere and D. Kröning, "SAT-based model checking," in *Handbook of Model Checking*. Springer, 2018, pp. 277–303.

[22] C. Barrett and C. Tinelli, "Satisfiability modulo theories," in *Handbook of Model Checking*. Springer, 2018, pp. 305–343.

[23] R. Sebastiani, "Lazy satisfiability modulo theories," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 3, no. 3-4, pp. 141–224, 2007.

[24] Q. Zhang, D. K. Hong, Z. Zhang, Q. A. Chen, S. Mahlke, and Z. M. Mao, "A systematic framework to identify violations of scenario-dependent driving rules in autonomous vehicle software," *Proc. of the ACM on Measurement and Analysis of Computing Systems*, vol. 5, no. 2, pp. 1–25, 2021.

[25] Y. Shoukry, P. Nuzzo, I. Saha, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, "Scalable motion planning using lazy SMT-based solving," in *Proc. of the IEEE Conf. on Decision and Control*, 2016, pp. 6683–6688.

[26] Y. Shoukry, P. Nuzzo, A. Balkan, I. Saha, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, "Linear temporal logic motion planning for teams of underactuated robots using satisfiability modulo convex programming," in *Proc. of the IEEE Conf. on Decision and Control*, 2017, pp. 1132–1137.

[27] 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," in *Lectures on Runtime Verification*. Springer, 2018, pp. 135–175.

[28] M. Davis, G. Logemann, and D. Loveland, "A machine program for theorem-proving," *Communications of the ACM*, vol. 5, no. 7, pp. 394–397, 1962.

[29] R. Nieuwenhuis, A. Oliveras, and C. Tinelli, "Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL (T)," *Journal of the ACM*, vol. 53, no. 6, pp. 937–977, 2006.

[30] D. Kroening and O. Strichman, *Quantified Formulas*. Springer, 2016, pp. 199–227.

[31] C. Pek and M. Althoff, "Fail-safe motion planning for online verification of autonomous vehicles using convex optimization," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 798–814, 2021.

[32] A. Dokhanchi, S. Yaghoubi, B. Hoxha, and G. Fainekos, "Vacuity aware falsification for MTL request-response specifications," in *Proc. of the IEEE Conf. on Automation Science and Engineering*, 2017, pp. 1332–1337.

[33] G. S. Tseitin, "On the complexity of derivation in propositional calculus," in *Automation of Reasoning*. Springer, 1983, pp. 466–483.

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

[35] A. Tamke, T. Dang, and G. Breuel, "A flexible method for criticality assessment in driver assistance systems," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2011, pp. 697–702.

[36] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge: Cambridge University Press, 2004.

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

[38] 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.

[39] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highD dataset: A drone dataset of naturalistic vehicle trajectories on German highways for validation of highly automated driving systems," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2018, pp. 2118–2125.

[40] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2909–2913, 2016.

[41] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[42] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frénet frame," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2010, pp. 987–993.