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 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 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 (T-solver). This method abstracts the
input formula to a propositional one and feeds it to a SAT solver to suggest possible assignments. The $T$-solver checks the satisfiability of the obtained assignment in a theory $T$ ($T$-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 $T$-solver to check the $T$-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. III required preliminaries and definitions are introduced. Sec. IV provides an overview of our trajectory repairing approach. In Section V the lazy SMT-based trajectory repairing framework is described. We demonstrate the benefits of our method by case studies in Sec. VI followed by conclusions in Sec. VII.

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:

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

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 $x_k \subset \mathbb{R}^n$ and admissible control inputs $u_k \subset \mathbb{R}^m$, 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 $\chi([k, x_0, u([0, k])])$. A complete state trajectory for the time interval $[0, h]$ is abbreviated as $\chi$.

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

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 $x_{\text{cut}}$):**

The cut-off state $x_{\text{cut}}$ [3, Sec. III-A] is the state from which the repaired trajectory begins.

Let $X_{\text{CF}}^k = x_k \setminus O_B(k)$ be the collision-free set of states at time step $k$, $\varphi$ be one or multiple rules since one can combine the rules using conjunction, and $\chi \models \varphi$ denote that a trajectory $\chi$ complies with $\varphi$. The violation-free states are defined as:

**Definition 2 (Violation-Free States $X_{\text{VF}}^k$):**

The set $X_{\text{VF}}^k(\varphi) \subset X_{\text{CF}}^k$ is the set of collision-free states that additionally comply with traffic rules $\varphi$ at time step $k$, i.e., $X_{\text{VF}}^k(\varphi) := \{x_k \in X_{\text{CF}}^k \mid x_k = \chi([k, x_0, u([0, k])]) \land \chi([0, k], x_0, u([0, k])) \models \varphi\}$.

**Definition 3 (Rule-Compliant Set $C$):**

Given traffic rules $\varphi$, the rule-compliant set $C_k(\varphi) \subset X_{\text{VF}}^k(\varphi)$ 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 $C_k(\varphi) := \{x_k \in X_{\text{VF}}^k(\varphi) \mid \exists u([k, h]) : \chi([k, x_k, u([k, h])]) \models \varphi\}$.

This definition is illustrated in Fig. 2. For brevity, we omit the $\varphi$-dependency in the notations for $X_{\text{VF}}^k$ and $C$. With this, we introduce the following measures to find an appropriate $x_{\text{cut}}$ from which the repaired trajectory branches off.
Fig. 3: Violation properties of trajectories. Only trajectory $\chi_1$ continuously complies with traffic rules. Both trajectory $\chi_2$ and $\chi_3$ violate traffic rules at time step $TV$. Trajectory $\chi_3$ leaves the rule-compliant set $C$ 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 $u^{\text{int}}([0, h])$ leaves the set of violation-free states $X^V$:

$$TV := \min \{ k \in \mathbb{N}_0 | \chi(k, x_0, u^{\text{int}}([0, k])) \not\in X^V \}.$$  

If no violation is detected, i.e., all states are within $X^V$, we set $TV = \infty$.

**Definition 5 (Time-To-Comply):**
Assuming that $x_0 \in X^V_0$, the time-to-comply (TC) is the last time step for which a rule-compliant trajectory exists:

$$TC := \max \{ k \in [0, TV] | \chi(k, x_0, u^{\text{int}}([0, k])) \in C_k \}.$$  

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., $x_{\text{cut}} := x_{TC}$. 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 $\varphi$, $\varphi_1$, and $\varphi_2$, the logical $\text{True}$ $\top$, a propositional variable $\sigma$ presenting a Boolean statement, an associated interval $I$ of $\mathbb{N}_0$, the Boolean $\text{negation}$ $\neg$ and $\text{disjunction}$ operator $\lor$, and the temporal $\text{since}$ and $\text{until}$ operator $S_I$ and $U_I$, MTL syntax is defined as [9]:

$$\varphi := \top | \sigma | \neg \varphi | \varphi_1 \lor \varphi_2 | \varphi_1 S_I \varphi_2 | \varphi_1 U_I \varphi_2.$$  

(2)

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

$$\varphi_1 S_I \varphi_2 \iff \exists k' \in (k-I) \cap \mathbb{N}_0 : (\varphi_2 \land \forall k'' \in (k', k) : \varphi_1),$$  

$$\varphi_1 U_I \varphi_2 \iff \exists k' \in (k-I) \cap \mathbb{N}_0 : (\varphi_2 \land \forall k'' \in [k, k') : \varphi_1).$$  

(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:

$$\| \equiv \neg \top$$

$$\varphi_1 \land \varphi_2 := \neg (\neg \varphi_1 \lor \neg \varphi_2)$$

$$\varphi_1 \Rightarrow \varphi_2 := \neg \varphi_1 \lor \varphi_2$$

$$O_I \varphi := \top S_I \varphi$$

$$P_I \varphi := \bot S_I \varphi$$

$$F_I \varphi := \top U_I \varphi$$

$$G_I \varphi := \neg F_I \neg \varphi.$$  

(4)

where $O_I$, $P_I$, $F_I$, and $G_I$ are the $\text{once}$, $\text{previously}$, $\text{finally}$ (aka eventually), $\text{globally}$ (aka always) temporal operators, respectively.

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

$$\rho(\top, \chi, k) := \infty$$

$$\rho(\sigma, \chi, k) := \text{Dist}_E(x_k, X^V_k(\sigma))$$

$$\rho(\neg \varphi, \chi, k) := -\rho(\varphi, \chi, k)$$

$$\rho(\varphi_1 \lor \varphi_2, \chi, k) := \max(\rho(\varphi_1, \chi, k), \rho(\varphi_2, \chi, k))$$

$$\rho(\varphi_1 S_I \varphi_2, \chi, k) := \max_{k' \in (k-I) \cap \mathbb{N}_0} \min_{k'' \in [k, k')} \rho(\varphi_1, \chi, k''))$$

$$\rho(\varphi_1 U_I \varphi_2, \chi, k) := \max_{k' \in (k-I) \cap \mathbb{N}_0} \min_{k'' \in [k, k')} \rho(\varphi_1, \chi, k'')),$$

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

$$\text{Dist}_E(x, X^V_k) := \begin{cases} -\inf\{E(x, x') | x' \in X^V_k \} & \text{if } x \not\in X^V_k \\ \inf\{E(x, x') | x' \in X^V_k \} & \text{if } x \in X^V_k \end{cases}.$$  

(6)

**D. Davis-Putnam-Lovemann-Loveland Algorithm**

The Davis-Putnam-Lovemann-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(T) [29]). To streamline the notation, we write $\varphi^P$ and $\sigma^P$ to denote $\varphi$ and $\sigma$ after propositional abstraction as input for the DPLL algorithm, respectively. The propositional formula also needs to be in conjunctive normal form (CNF) as $\land \land \lor \lor$, i.e., a conjunction of clauses that are disjunctions of literals, where a literal is either a positive or negative atomic proposition $\sigma$. 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 $\phi$.

**E. Problem Statement**

For rule-compliant trajectory repairing, we use traffic rules formalized in MTL. To exploit SAT 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 $G$ and its equivalents, and leave the other rules to future work with a possible integration of specification-compliant reachable sets [4]. For $G \varphi$ to be satisfied, we need $\varphi$ to be satisfied at all time steps. Thus, $G \varphi$ is equivalent to $\exists k \in [0, h] : \varphi_k \land \varphi_{k+1} \land \ldots \land \varphi_h$, where $\varphi_k$ is the valuation of $\varphi$ 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 $\chi^\text{int}$ for the ego vehicle and an MTL monitor $M$ which is constructed by our previous works [2], [11] with the environment model $\Omega$ to evaluate traffic rules $G \varphi$. Our algorithm outputs a repaired trajectory that complies with $G \varphi$.

As a first step, we run the monitor to obtain $TV$ and the robustness degree of $\chi^\text{int}$ at all time steps which is denoted by $\rho^\varphi_{\text{int}}$ (see line 1). If $\chi^\text{int}$ violates the traffic rule $G \varphi$ and $TV \neq 0$, $\varphi$ is abstracted to $\varphi^p$ in CNF (see line 3), which is later explained in Sec. IV-A. Our approach is iterative using the DPLL($T$) algorithm with a lazy SMT propagation. At each iteration, we start by checking the Boolean satisfiability of $\varphi^p$ for the time interval $[TV, h]$ using DPLL-based SAT solvers (see line 4). In case the result is SAT, we simultaneously obtain a satisfying solution $\phi$ for $\varphi^p$ (see line 5). Then the $T$-solver reasons about the repairability $r \in \mathbb{B}$ of $\phi$ (see line 6). If $r$ is equal to $T$, we return the corresponding repaired trajectory $\chi^{\text{rep}}$ (see line 7). Otherwise, we update $\varphi^p$ by treating $\phi$ as a conflicting clause $\neg \phi$ in the future runs (see line 8). 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 $\chi^\text{int}$, traffic rule monitor $M$, rule $G \varphi$
Output: repaired trajectory $\chi^{\text{rep}}$
1: $TV, \rho^\varphi_{\text{int}} \leftarrow \text{MONITOR}(\chi^\text{int}, G \varphi)$
2: if $TV \notin (0, \infty)$ then
3: $\varphi^p \leftarrow \text{ABSTRACTTRAFFICRULE}(G \varphi)$ \(\triangleright \text{Sec. IV-A}\)
4: while SAT.CHECK$(\varphi^p, \rho^\varphi_{\text{int}}, TV) \equiv \text{SAT} \lor \triangleright \text{Sec. IV-B}$
5: $\phi \leftarrow \text{SAT.SOLUTION()}$ \(\triangleright \text{Sec. IV-C}\)
6: $r, \chi^{\text{rep}} \leftarrow \text{T-SOLVER.CHECK}(\phi, M, \rho^\varphi_{\text{int}})$ \(\triangleright \text{Sec. IV-C}\)
7: if $r = \top$ then
8: return $\chi^{\text{rep}}$
9: else
10: $\varphi^p \leftarrow \varphi^p \land \neg \phi$
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($T$)-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 $T$-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:

$$G \varphi = G(\text{in\_same\_lane}(x_{ego}, x_b) \land \text{in\_front\_of}(x_{ego}, x_b) \land \neg O[0, t_c](\text{cut\_in}(x_b, x_{ego}) \land P(\neg \text{cut\_in}(x_b, x_{ego})))) \quad (7)$$

where $x_b$ is the state of the rule-relevant vehicle $b \in B$, $t_c$ is the recovery time for the safe distance violation caused by a cut-in maneuver of $b$, and in_same_lane (whether two vehicles are in the same lane), in_front_of (whether $b$ is in front of the ego vehicle), cut_in (whether $b$ enters the lane of ego vehicle), and 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 $G$ by instantiating $G \varphi$ to $[\varphi][0,h] = \top$ and abstract $\varphi$ to a propositional logic formula $\varphi^p$. The latter can be achieved by replacing the predicates and the elements starting with temporal operators within $\varphi$ with propositional variables $\sigma^p$. With this, $\varphi^p$ 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 $b'$, $\varphi$ is abstracted to a propositional formula $\varphi^p$ in CNF as:

$$\varphi^p := \text{in\_same\_lane}(x_{ego}, x_{b'}) \land \text{in\_front\_of}(x_{b'}, x_{ego}) \land \neg O[0, t_c](\text{cut\_in}(x_{b'}, x_{ego}) \land P(\neg \text{cut\_in}(x_{b'}, x_{ego})))$$

$$\Rightarrow \text{keeps\_safe\_distance\_prec}(x_{ego}, x_{b'})$$

$$\equiv \neg \sigma^p_1 \lor \neg \sigma^p_2 \lor \sigma^p_3$$

of which the violating assignments at time step $k$ can only be $[\sigma^p_1]_k = \top$, $[\sigma^p_2]_k = \top$, $[\sigma^p_3]_k = \bot$, and $[\sigma^p_4]_k = \bot$.

B. SAT Solver

After obtaining the abstracted formula, we can use the DPLL algorithm to solve the Boolean satisfiability of $\varphi^p$ for the time interval $[TV, h]$. To reduce computational load, we assume that the assignment of selected propositions keeps unchanged within $[TV, h]$. 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.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Time-to-brake (TTB)</td>
<td>Full braking with maximum deceleration.</td>
<td>Longitudinal position, velocity</td>
<td>in_front_of, keeps_safe_distance_prec, keeps_lane_speed_limit, keeps_fov_speed_limit, keeps_type_speed_limit, ...</td>
</tr>
<tr>
<td>Time-to-kick-down (TTK)</td>
<td>Full accelerating until reaching the maximum velocity and then maintaining the velocity.</td>
<td>Lateral position in_same_lane, left_of, drives_rightmost, ...</td>
<td></td>
</tr>
<tr>
<td>Time-to-steer (TTS)</td>
<td>Full steering to reach a certain lateral offset.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time-to-maintain-velocity (TTMV)</td>
<td>Maintaining a steady velocity.</td>
<td>Acceleration brakes_abruptly, ...</td>
<td></td>
</tr>
</tbody>
</table>

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

C. $\mathcal{T}$-Solver

In the $\mathcal{T}$-solver, the $\mathcal{T}$-consistency of $\phi$ needs to be determined regarding the environment model $\Omega$ and the system dynamics of the ego vehicle, which is summarized in Alg. 2. To address this, we first obtain propositions $\phi_r$ to be repaired by comparing the valuations of atomic propositions in $\phi$ with the violating assignments for $[\phi^p]_{TV,h} = \top$, i.e., only the atomic propositions with a different value in $\phi$ are considered (see line [1]). After determining the TC of $\phi_r$ (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 $\phi$ from the SAT solver is $[\sigma_1 \land \sigma_2]_{TV,h} = \top$, we can obtain the proposition to be repaired as $\phi_r = \sigma_2$ since $[\sigma_1]_{TV,h}$ 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 undersample the TC (cf. Fig. [3]) by using a point-mass vehicle model [34] and focusing only on the predicates contained in $\phi_r$. To achieve this, we introduce assessment metrics based on the category of predicates from [4], which are listed in Tab. [1] and illustrated in Fig. [4].

The initial values of TC and $\tau$ are set to $-\infty$ and $\bot$, respectively (see line [2] in Alg. [2]). Next, we automatically obtain the compliant maneuvers according to Tab. [1] based on the category of predicates in $\phi_r$ (see line [3]). Afterward, we use binary search to detect the maximum remaining time for executing rule-compliant maneuvers in the function $\text{search}TC()$, 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 $\sigma_4$ consist of TTB and TTK according to Tab. [4] 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 [5]). 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 $(x_{lon}, x_{lat})^T$ in a curvilinear coordinate system [37] aligned with a predefined reference path $\Gamma$, which is typically obtained from a high-level route planner. The longitudinal state $x_{lon} = (s, v, a, j)^T$ consists of position $s$, velocity $v$, acceleration $a$, and jerk $j$. The lateral motion is described by $x_{lat} = (d, \theta, \kappa, \dot{\kappa})^T$, where $d$ is the lateral distance to $\Gamma$, $\theta$ is the orientation, $\kappa$ 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 $k \in \{T, \ldots, h\}$, 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. [3] 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 $\phi_r$. In order to preserve convexity for any predicate, a positive quadratic robustness term is chosen with weight $\omega_r \in \mathbb{R}^+$ as:

$$J_r(x, \dot{x}) = \omega_r \sum_{k=\text{TC}}^h (x_k - \dot{x}_k)^2,$$

Fig. 4: Illustration of different compliant maneuvers. The points indicate the start of the corresponding maneuvers.
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 2).

All atomic propositions have the same assignments from TC that violate the rule is marked in red. The first interval $[T-1, h]$ starting from TV shows the initial configuration of the scenario and the initial area. We visualize $\chi_1$ executed after the cut-off state to enlarge the violation-free trajectory is marked in green where a braking maneuver is illustrated in Fig. 5b, the ego occupancy along the repaired trajectory with convex linear-quadratic programs. As $TC = 13$ the trajectory of the ego vehicle violates the safe distance rule for all $\phi$ time steps since $\phi$ is used for constructing rule constraints in the remaining positions. Otherwise, we relax the requirement and check the $\phi$ 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 2), the $\phi$ robustness degrees of the atomic propositions within the time steps since $\phi$ is checked as SAT by $\phi$ in the SAT solver (cf. Sec. IV-B).

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

$$s_k \in (-\infty, \text{rear}(x_{v', k}) - \Delta_{\text{safe}}(v_k, x_{v', k})),$$

where $\Delta_{\text{safe}}$ is the safe distance defined in [31, (4)] and $\text{rear}(\cdot)$ 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 [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](https://mediatum.ub.tum.de/1641743)

A. Keeping a Safe Distance to the Preceding Vehicle (R,G1)

We first evaluate a rural scenario 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 $[TV, h]$ 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 $[\sigma_4^{\text{r}}]_{[TV, h]} = \top$. In the $T$-solver, we check the $T$-consistency of $\sigma_4^p$ and obtain TC = 13 using TTB (cf. Tab. II). 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 $\chi_{\text{VF}}$ in the k-s plane together with the initial and repaired trajectory in Fig. 5c.

![Diagram](https://commonroad.in.tum.de/

2CommonRoad ID: DEU_Gar-1_1_T-1

3CommonRoad ID: ZAM_Zip-1_56_T-1

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Rob.</th>
<th>TC</th>
<th>Rob.</th>
<th>TC</th>
<th>Rob.</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_1$</td>
<td>0.1132</td>
<td>0.0017</td>
<td>0.0004</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\sigma_2$</td>
<td>0.0695</td>
<td>-0.0017</td>
<td>1.0000</td>
<td>0.0004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\sigma_3$</td>
<td>-0.0249</td>
<td></td>
<td>0.5175</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\sigma_4$</td>
<td>-0.0027</td>
<td>13</td>
<td>0.4176</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Computation Time in ms**

<table>
<thead>
<tr>
<th>Type</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repairing</td>
<td>223</td>
</tr>
<tr>
<td>Replanning</td>
<td>887</td>
</tr>
</tbody>
</table>

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 ($\alpha < \alpha_{\text{abrupt}}$) is not allowed without justification (violation of safe distance or its preceding vehicle brakes abruptly), where $\alpha_{\text{abrupt}} \in \mathbb{R}^-$ denotes the predefined acceleration threshold. The rule can be reformulated based on the modification in [11, Tab. I] as:

$$G\phi = G\left(\text{brakes_abruptly}(x_{ego}) \Rightarrow \sigma_4^p\right)$$

where the predicate brakes_abruptly specifies whether a vehicle brakes abruptly and 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 in which the ego vehicle brakes abruptly. However, this is unnecessary since braking_justification($x_{ego}$, $\Omega$) 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 $\phi = \neg\sigma_1^p \lor \sigma_2^p$ and running the SAT algorithm, 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 $\chi_{\text{VF}}$ in the k-s plane together with the initial and repaired trajectory in Fig. 5c.

<table>
<thead>
<tr>
<th>Proposition</th>
<th>Scenario I</th>
<th>Scenario II</th>
<th>Scenario III</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_1$</td>
<td>0.1132</td>
<td>0.0017</td>
<td>-0.0026</td>
</tr>
<tr>
<td>$\sigma_2$</td>
<td>0.0695</td>
<td>-0.0017</td>
<td>1.0000</td>
</tr>
<tr>
<td>$\sigma_3$</td>
<td>-0.0249</td>
<td></td>
<td>0.5175</td>
</tr>
<tr>
<td>$\sigma_4$</td>
<td>-0.0027</td>
<td>13</td>
<td>0.4176</td>
</tr>
</tbody>
</table>

**Computation Time in ms**

<table>
<thead>
<tr>
<th>Type</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repairing</td>
<td>223</td>
</tr>
<tr>
<td>Replanning</td>
<td>887</td>
</tr>
</tbody>
</table>
solver, we obtain a satisfying solution for \([\varphi^*]_{TV,g} = T\) as \([\sigma^*]_{TV,g} = \bot\). In the \(T\)-solver, TC is computed as 4 using TTMV (cf. Tab. 1). 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:

\[
G \varphi = G \text{(keeps_lane_speed_limit}(x_{ego}) \wedge \text{keeps_type_speed_limit}(x_{ego}) \wedge \text{keeps_fov_speed_limit}(x_{ego}) \wedge \text{keeps_braking_speed_limit}(x_{ego}))
\]

\[\equiv G(\tau \Rightarrow \sigma^0 \wedge \sigma^1 \wedge \sigma^2 \wedge \sigma^3) \equiv G(\sigma^0 \wedge \sigma^1 \wedge \sigma^2 \wedge \sigma^3),\]

with which the ego vehicle is not allowed to exceed 1) the speed limit of driving lanes \(v_{d}^{\text{max}} \in \mathbb{R}_0\) (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 \(v_{br} \in \mathbb{R}_0\) (keeps_braking_speed_limit). We demonstrate the repairing process as for R\(_{G3}\) with an urban scenario 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.

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 \(2\mathfrak{M}\) 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 \(2\mathfrak{M}\), 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.