TECHNISCHE UNIVERSITÄT MÜNCHEN

Fakultät für Informatik

Formalizing and Modeling Traffic Rules Within Interactive Behavior Planning

Klemens Esterle

Vollständiger Abdruck der von der Fakultät der Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Prof. Dr. Jan Křetínský
Prüfer der Dissertation:	1.	Prof. Dr.-Ing. habil. Alois Knoll
	2.	Prof. Dr. techn. Daniel Watzenig

Die Dissertation wurde am 29.03.2021 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 18.08.2021 angenommen.

Abstract

This thesis addresses behavior planning for autonomous vehicles in highly dense scenarios, which are challenging because of the necessary interactions with other agents. Here, autonomous vehicles need to flow with traffic but also adhere to all traffic rules – even complex priority rules like the zipper merge. Two problems arise from that: First, traffic rules are fuzzy and not mathematically defined, making them incomprehensible to machines. Without proper formalization, satisfaction cannot be monitored in simulation or testing nor implemented for planning. Second, the planning component needs to integrate those rules, but no interaction-aware planning algorithm exists that provides a mechanism to enforce the compliance to rules, which depend on the behavior of multiple interacting participants.

This work approaches the first problem by introducing a methodology to formalize traffic rules in a formal language. Temporal logic is used as a formal specification language to capture a wide range of traffic rules. The rules are evaluated on public traffic data. The second problem is addressed by designing rule-compliance to traffic rules for two orthogonal interactive planning algorithms. The first approach monitors the rules within a randomized search of a decision tree. It exploits the structure of the decision tree by encoding temporal rule information in the tree node. A simulation-based evaluation toolchain helps to study the effects of modeling a specific rule. This way, contradictions in the rules can be identified and fed back to the rule formalization stage. As randomized solution methods may be challenging for certifying the algorithm, a second approach based on optimal control is presented. The novel model formulation is solved using Mixed-Integer Programming and approximates the rules to be part of the optimization. The effectiveness of the rule approximation is demonstrated in simulation.

Summary: This work first identifies all applicable traffic rules, formalizes them, and provides a simulation toolchain to evaluate them. Then, two orthogonal methods for interactive planning are developed that assert traffic rules adherence during planning.

Zusammenfassung

Die vorliegende Dissertation befasst sich mit der Verhaltensplanung für autonome Fahrzeuge in dichten Szenarien, welche aufgrund der darin notwendigen Interaktionen mit anderen Agenten eine Herausforderung für bestehende Systeme darstellen. In solchen Szenarien müssen autonome Fahrzeuge mit dem Verkehr mitschwimmen, aber auch alle Verkehrsregeln einhalten – selbst komplexe Vorfahrsregeln wie das Reißverschlussverfahren. Daraus ergeben sich zwei Probleme: Erstens sind die Verkehrsregeln unscharf und nicht mathematisch definiert, was eine maschinelle Auswertung verhindert. Ohne Formalisierung kann die Einhaltung weder in Simulationen oder realen Tests überwacht werden, noch innerhalb eines Planungsalgorithmus implementiert werden. Zweitens muss die Planungskomponente diese Regeln integrieren, aber es existiert kein Interaktionen modellierender Planungsalgorithmus, der einen Mechanismus zur Einhaltung von Regeln bietet, die vom Verhalten mehrerer interagierender Teilnehmer abhängen.

Diese Arbeit nähert sich dem ersten Problem durch die Einführung einer Methodik zur Formalisierung von Verkehrsregeln in einer formalen Sprache. Dabei wird temporale Logik als formale Spezifikationssprache verwendet, um eine breite Palette von Verkehrsregeln zu beschreiben. Die Regeln werden mittels öffentlicher Verkehrsdaten ausgewertet. Das zweite Problem wird durch den Entwurf der Regelkonformität für zwei orthogonale interaktive Planungsalgorithmen behandelt. Der erste Ansatz überwacht die Regeln während einer zufallsbasierten Suche innnerhalb eines Entscheidungsbaums, und nutzt dessen Struktur aus, indem er zeitliche Regelinformationen in den Baumknoten kodiert. Ein simulationsbasiertes Evaluierungswerkzeug hilft dabei, die Auswirkungen der Modellierung einer bestimmten Regel zu untersuchen. Auf diese Weise können Widersprüche in den Regeln identifiziert und in die Phase der Regelformalisierung zurückgeführt werden. Da zufallsbasierte Lösungsmethoden eine Herausforderung für die Zertifizierung des Algorithmus darstellen kann, wird ein zweiter Ansatz vorgestellt, der auf einer Optimalsteuerung basiert. Die neuartige Modellformulierung wird mittels gemischt-ganzzahliger Programmierung gelöst und approximiert die Regeln als Teil des Optimierungproblems. Die Wirksamkeit der Regelapproximation wird in der Simulation demonstriert.

Zusammenfassung: Diese Arbeit identifiziert zunächst alle anwendbaren Verkehrsregeln, formalisiert sie und stellt ein Simulations-Werkzeug zur Verfügung, um die Einhaltung der Regeln zu bewerten. Im Anschluss werden zwei orthogonale Methoden für die interaktive Verhaltensplanung entwickelt, welche die Einhaltung der Verkehrsregeln während der Planung durchsetzen.

Contents

Acronyms **3**

List of Symbols **5**

1. Introduction **9**
1.1. Motivation . . . 9
1.2. State of the Art . . . 10
1.2.1. State of the Art on Behavior Planning . . . 10
1.2.2. State of the Art for Ensuring Traffic Rules Within Planning . . . 13
1.2.3. State of the Art for Traffic Rule Formalization . . . 14
1.3. Contributions . . . 15
1.4. Publications . . . 16
1.5. Structure of this Thesis . . . 17

2. Preliminaries **19**
2.1. Benchmarking Behavior in Simulation . . . 19
2.1.1. Overview and Contribution . . . 19
2.1.2. BARK in a Nutshell . . . 20
2.1.3. Behavior Modeling for Realism in Simulation . . . 22
2.1.4. Evaluator Concept for Benchmarking . . . 23
2.2. Monte Carlo Tree Search . . . 24
2.2.1. Monte Carlo Tree Search for Behavior Planning . . . 24
2.2.2. Monte Carlo Tree Search in BARK . . . 24

3. Formalization of Traffic Rules for Machine Interpretability **27**
3.1. Overview and Contribution . . . 27
3.2. Legal Analysis of German Traffic Rules on Dual Carriageways . . . 27
3.2.1. Regulations on Speed . . . 28
3.2.2. Regulations on the Use of Roads and Lanes . . . 30

3.2.3. Regulations on Overtaking . . . 31
3.2.4. Regulations on a Safe Distance . . . 33
3.2.5. Regulations on Being Overtaken . . . 34
3.2.6. Regulations on Priorities . . . 34
3.3. Identifying a Suitable Language for Machine Interpretability . . . 35
3.3.1. Characteristics of Traffic Rules in Dense Highway Scenarios . . . 35
3.3.2. Selection of a Formalization Language . . . 36
3.4. Formalizing Traffic Rules Using Linear Temporal Logic . . . 38
3.4.1. Linear Temporal Logic for Codification . . . 38
3.4.2. Atomic Propositions for Concretization . . . 38
3.4.3. Codified Rules . . . 39
3.5. Conclusion . . . 42

4. Evaluating Traffic Rules in Linear Temporal Logic on Recorded Drives 43
4.1. Overview and Contribution . . . 43
4.2. Linear Temporal Logic on Finite Traces . . . 43
4.3. Automaton-based Verification of LTL_f Formulas . . . 44
4.4. Runtime Monitoring of Traffic Rules . . . 44
4.4.1. Rule Violations . . . 44
4.4.2. Open Rule Monitor . . . 45
4.5. Evaluation . . . 45
4.5.1. Evaluation Methods and Dataset Processing . . . 45
4.5.2. Evaluation of Violation on Recorded Drives . . . 46
4.6. Conclusion . . . 53

5. Monitoring of Traffic Rules Within Interactive Behavior Planning 55
5.1. Overview and Contribution . . . 55
5.2. Related Work . . . 56
5.3. Problem Formulation and Assumptions . . . 56
5.4. Approach . . . 57
5.4.1. Rule Monitoring Within Monte Carlo Tree Search . . . 57
5.4.2. Costs for a Violation Penalty . . . 58
5.4.3. Multi-Objective Reward Function with Priorities . . . 58
5.5. Experiments and Results . . . 59
5.5.1. Experimental Setup . . . 59
5.5.2. Zipper Merge in Merging Scenario . . . 61
5.5.3. Quantitative Evaluation . . . 62
5.6. Conclusion . . . 64

6. Optimal Interactive Behavior Planning Satisfying Traffic Rules 67
6.1. Overview and Contribution . . . 67
6.2. Related Work . . . 68

6.3. Problem Formulation and Assumptions . . . 72
6.4. Region-based Linearization Approach of Nonlinear Constraints . . . 72
6.4.1. Discretized and Disjunctive Modeling of the Orientation . . . 72
6.4.2. Over-Approximating the Collision Shape . . . 74
6.4.3. Modeling the Non-Holonomics . . . 75
6.5. Fitting Method of Linear Polynomials . . . 76
6.5.1. Fitting the Front Axle Position . . . 76
6.5.2. Fitting the Curvature . . . 78
6.6. Formulating the Planning Problem as Linear Dynamic Game . . . 78
6.6.1. Formulating the Vehicle Model as Constraints . . . 79
6.6.2. Modeling the Non-Holonomy as Constraints . . . 80
6.6.3. Approximating the Front Axle Position as Constraints . . . 81
6.6.4. Constraints Limiting the Model to Stay on the Road . . . 81
6.6.5. Formulating Collision Avoidance as Constraints . . . 82
6.6.6. Multi-Agent Collision Constraints . . . 83
6.6.7. Traffic Rules . . . 86
6.6.8. Joint Cost Function for Reference Tracking . . . 90
6.6.9. Optimization Problem . . . 90
6.6.10. Receding Horizon Formulation . . . 90
6.7. Experiments and Results . . . 91
6.7.1. Preserving the Non-Holonomy . . . 91
6.7.2. Staying Within the Road Boundaries . . . 92
6.7.3. Avoiding Dynamic Obstacles . . . 93
6.7.4. Planning for Multiple Agents . . . 96
6.7.5. Evaluation on Traffic Rules . . . 96
6.7.6. Benchmark . . . 99
6.8. Conclusion . . . 103

7. Comparison of Planning Approaches for Traffic Rule Integration 105
7.1. Comparison of Planning Approaches . . . 105
7.1.1. Characteristics of Model . . . 105
7.1.2. Characteristics of Solution Method . . . 106
7.1.3. Possible Extensions of Solution Method . . . 108
7.2. Comparison of Implemented Rules . . . 108
7.3. Conclusion . . . 110

8. Future Work 111
8.1. Traffic Ruleset . . . 111
8.2. Functional Improvements for MCTS with Rules . . . 112
8.3. Performance Improvements for MIQP . . . 113
8.4. Functional Improvements for MIQP with Rules . . . 114

9. Conclusion **115**

A. Appendix **117**
A.1. Parameters for Dataset Evaluation 117
A.2. Parameters for MOBIL-based Behavior Model 118
A.3. Parameters for MCTS-based Planner Evaluation 118
A.4. Parameters for MIQP-based Planner Evaluation 119

List of Figures **121**

List of Tables **123**

List of Algorithms **125**

Bibliography **127**

Acronyms

BARK	Behavior Benchmarking Framework.
BOV	Being Overtaken.
DFA	Deterministic Finite Automaton.
DIST	Distance.
IDM	Intelligent Driver Model.
LO	Lexicographical Ordering.
LS	Lane Selection.
LTL	Linear Temporal Logic.
LTL_f	Linear Temporal Logic on finite traces.
MCTS	Monte Carlo Tree Search.
MDP	Markov Decision Process.
MILP	Mixed-Integer Linear Programming.
MIP	Mixed-Integer Programming.
MIQP	Mixed-Integer Quadratic Programming.
MIQP-SA	Single Agent Variant of the MIQP-based planner.
MOBIL	Minimizing Overall Braking Induced by Lane Changes.
MTL	Metric Temporal Logic.
ODD	Operational Design Domain.
OV	Overtaking.
POMDP	Partially Observable Markov Decision Process.
PRIO	Priority.

QP	Quadratic Programming.
RRT	Rapidly Exploring Random Tree.
RSS	Responsibility Sensitivity Safety Model.
SA	Single Agent.
SA-Lex	Single Agent Variant of MCTS with Lexicographic Ordering.
SD	Safe Distance.
SQP	Sequential Quadratic Programming.
STL	Signal Temporal Logic.
StVO	Strassenverkehrsordnung.
TLO	Thresholded Lexicographical Ordering.
UCT	Upper Confidence Bounds for Trees.
ZIP	Zipper Merge.

List of Symbols

Notation	Description
$(\cdot)(k)$	value $(\cdot)$ at discrete step k
$(\cdot)(t)$	value $(\cdot)$ at continuous time t
$(\cdot)^i$	value $(\cdot)$ of agent i
$(\cdot)^j$	value $(\cdot)$ of agent j
$(\cdot)_{\text{base}}$	base
$(\cdot)_{\text{col}}$	collision
$(\cdot)_{\text{lat}}$	lateral
$(\cdot)_{\text{max}}$	maximum
$(\cdot)_{\text{mean}}$	mean
$(\cdot)_{\text{ref}}$	reference
$(\cdot)_{\text{sd}}$	safe distance
$(\cdot)_{\text{zip}}$	zipper merge
$(\cdot)_{\text{br}}$	brake
$(\cdot)_x$	value in x-direction
$(\cdot)_y$	value in y-direction
$\overline{(\cdot)}$	upper bound
$\underline{(\cdot)}$	lower bound
a	action
$\boldsymbol{a}$	joint action
a	acceleration
A	action space
$\mathcal{A}$	joint action space
$\mathscr{A}$	set of agents
α^{ij}	true iff respective parts of agent i and agent j are not colliding

Notation	Description
B	behavior
$\mathcal{B}$	behavior model
$\mathcal{B}^e$	behavior model of ego agent
$\mathcal{B}^o$	behavior model of other agent
$\overline{\beta}_x^r$	x value of region r upper region borderline
$\underline{\beta}_x^r$	x value of region r lower region borderline
$\overline{\beta}_y^r$	y value of region r upper region borderline
$\underline{\beta}_y^r$	y value of region r lower region borderline
c	conclusion
C_{UCT}	exploration constant
d	distance
D	desired safety distance between two agents
$\mathcal{D}$	slack-softened safety distance between two agents
δ	automata transition function
$\varsigma_p(o)$	true iff vehicle does not collide with obstacle o at the reference point p
$e(\lambda)$	true iff vehicle is not inside the environment sub-polygon λ
ϵ_v	threshold for velocity
F	set of accepting automata states
$\mathcal{F}_{\text{inLane}}$	lane matching parameter
γ	discount factor
Γ	non-convex environment polygon
j	jerk
k	discrete time step
κ	curvature
K	number of time steps
l	line segment
L	wheelbase
λ	convex environment sub-polygon
$\mathcal{L}$	label function
Λ	deterministic finite automaton
m	number of rules

Notation	Description
M	Big-M constant
N_{dense}	number of agents used for dense label calculation
N_{A}	number of agents
n	number of node visits
o	obstacle
Σ	automaton alphabet
p	position in Cartesian space
p	premise
$\boldsymbol{p}$	finite LTL formula
$\mathscr{P}$	polygon
$\mathscr{P}$	linear polynome
Π	set of atomic propositions
π	atomic proposition
Ψ	true iff no region change is allowed
φ	LTL formula
q	automaton state
$\boldsymbol{q}$	automata state vector
Q	state-action-value function
$\mathcal{Q}$	set of automata states
r	region
R_{cc}	radius of collision circle
R_{dense}	radius used for dense label calculation
$\imath$	reward
$\boldsymbol{\imath}$	reward vector
$\rho(r)$	true iff vehicle is in region r
ϱ^r	vehicle is allowed to be in region r
s	state
$\boldsymbol{s}$	environment state
$\mathcal{S}$	environment state space
σ	symbol
t	time
t_{react}	reaction time
t_{stop}	time needed to brake to standstill

Notation	Description
θ	orientation
$\boldsymbol{\tau}$	threshold vector for TLO
τ	threshold for TLO
$\triangle$	cumulative discounted reward
u	input
v	velocity
v_{stop}	velocity theshold for stopping
$\mathbf{V}$	set of vehicles
$\mathcal{V}$	tree node
w	word being a sequence of symbols
w_{lane}	lane width
W	penalty function
ω	weight
w	orientation rate
x	vehicle state
x_{des}	desired state
x_{ex}	executed state
X	vehicle state space
Ξ_{curr}	current lane corridor
Ξ_{left}	left lane corridor
Ξ_{ref}	reference lane corridor
Ξ_{right}	right lane corridor
$\zeta(o)$	true iff vehicle collides with soft obstacle o
ξ^{ij}	slack for vehicle-to-vehicle collision check of vehicles i and j
z	product state

1 Introduction

1.1. Motivation

Autonomous vehicles promise an increase of safety, time and cost efficiency of transport. The development of autonomous vehicles has made great progress in the last decade since the DARPA Urban Challenge [1–3]. However, all vehicles on the street are still prototypes, mostly overseen by test drivers. One of the key challenges for certification is that humans participate in traffic as well. Major problems arise in dense, dynamic situations, where the autonomous vehicle needs to operate in close interaction with vehicles, that are controlled by humans. Since most planning algorithms do not consider the interactions with human drivers, the number of disengagements on freeways and in urban areas, where higher traffic density requires more interactions, is especially large [4].

Traffic rules have been designed to help humans manage the otherwise chaotic traffic environment. If all vehicles were fully autonomous, the current set of rules could be reduced or adapted. In a transition period with mixed traffic however, autonomous vehicles will have to obey the same rules as humans, i.e., the planned behavior is collision-free and lawful [5]. Specifically, the scope of a behavior planning component for autonomous driving is calculating a plan to safely navigate through traffic reaching a desired state within the perceived horizon. Recent efforts to certify autonomous vehicles lead to the question of how an autonomous vehicle shall be programmed to obey traffic rules, and how this can be demonstrated in a structured and methodological manner.

The research line of *interactive behavior planning* addresses planning in dense situations with human drivers but has mostly ignored the aspect to obey traffic rules other than collision prevention and speed compliance. Figure 1.1 shows a possible system architecture if the interactive behavior planner (denoted *performance planner*) were not to consider all traffic rules at planning time. In that case, a *traffic rules monitor* would need to check the generated *behavior plan* regarding rule compliance at runtime. In the case of a detected rule violation, a decision module would need to switch to the safety planner – a planner that must come up with a behavior backup plan that is lawful – possibly by coming to a standstill. If the performance planner does not consider the rules, its behavior plan might yield violations quite often. How often will the decision

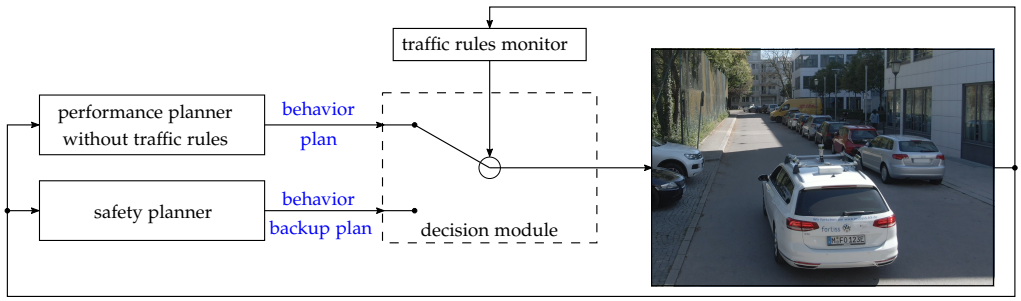


Figure 1.1.: Simplex architecture for a rule-compliant system, if the traffic rules are not considered at planning time.

module need to switch to the safety planner? How acceptable will the backup plan of the safety planner be? Essentially, simplex architectures can be used to guarantee safety for safety-critical systems. However, such architectures are not desirable when monitoring properties that will be violated regularly. A better solution would be to assert rule compliance in the planning component directly.

Previous work proposed combining model checking techniques for traffic rule satisfaction with motion planning in static environments [6]. However, dense scenarios remain difficult for motion planning approaches as they cannot anticipate human reactions correctly. With a focus on interactive behavior planning, this thesis aims to develop methods for rule compliant behavior that explicitly take the other vehicles' reaction into account.

1.2. State of the Art

The following section consists of a review of state-of-the-art planning and traffic rule modeling techniques for autonomous vehicles.

1.2.1. State of the Art on Behavior Planning

In this work, behavior of a controlled agent (from here on denoted *ego agent*) refers to it's desired future trajectory encoding the agent's strategy to reach a short-term goal, e.g., changing lanes. The trajectory may not be executable without deviations due to tracking errors or environmental influences. Schwarting et al. [7] distinguish between three different categories for planning and decision making: Sequential planning, behavior-aware planning, and end-to-end planning, see Figure 1.2.

Non-Interactive Planning Sequential planning is the traditional approach known from robotics, where planning is separated into a behavior layer and a motion or trajectory planner. The behavioral layer passes a high-level decision or maneuver to the motion planner, which computes a feasible trajectory for the next couple of seconds. In the days of DARPA's Urban Challenge, the behavior layer was often realized as a state machine, often summarized under the term *rule-based* [1, 2]. These approaches did well in the DARPA scenario at low speed with only

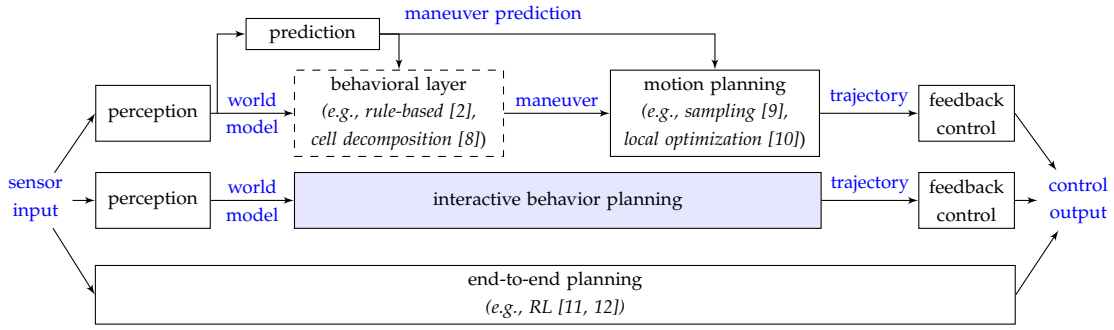


Figure 1.2.: Overview of three types of planning architectures (taken, reproduced and modified from Schwarting et al. [7]). This thesis focuses on interactive behavior-aware planning (highlighted in blue). The behavioral layer (dashed) might be omitted for some search or sampling-based motion planning approaches.

a few other traffic participants, but do not generalize well to new scenarios and unforeseen situations. More recent implementations may employ geometric approaches based on cell decomposition [8, 13]. However, as they do not plan in continuous space, they cannot guarantee the maneuver's feasibility. Sontges et al. [14] propose to employ reachability analysis to ensure the feasibility of the maneuver.

One line of direction for motion planning is *constrained local optimization*, where a vehicle model is forward propagated, and collision avoidance to static and dynamic obstacles is modeled as constraints [15–17]. The optimizer minimizes a cost function to find suitable inputs to the vehicle model along the discretized planning horizon. Instead of employing a vehicle model, Ziegler et al. [10] utilize a geometric approach and impose the vehicle's non-holonomy characteristics as constraints. While the properties of the optimized trajectory in terms of smoothness would be preferable, the problem is usually not convex, which is why a maneuver selection from the behavioral layer is required [16]. Thus, practical applications mostly consider evasive maneuvers, where the combinatorial non-convex problem can be simplified to a convex problem by the help of heuristics [17].

Discrete search- and sampling-based algorithms known from path planning can, in general, be applied to the spatiotemporal state space. They do not require a behavioral layer, although it may be beneficial to limit the search space, as additional knowledge can improve the convergence of the subsequent motion planning methods.

Graph search methods discretize the state or action space yielding a graph, which can be solved using search-based algorithms like [18], or carefully engineered heuristic [19]. Approaches where the graph representation is created only by feasible motion primitives are called *lattice planners* [20, 21]. McNaughton et al. [20] apply dynamic programming to solve the spatiotemporal problem. In structured environments, the input space can be discretized to sample road-aligned primitives [21]. Path-velocity decomposition simplifies the problem, i.e., a path planner obtains a collision-free path around static obstacles and a longitudinal motion planner along that path prevents collisions with dynamic obstacles [18].

Incremental search algorithms do not rely on an initial discretization to obtain a graph. Randomized sampling-based approaches like Rapidly Exploring Random Tree (RRT) or its extension RRT* [9, 22] incrementally construct a search tree by randomly sampling new states. No analytic function exists for a non-holonomic vehicle model to connect two sampled states in the spatiotemporal state space, which Jeon et al. [23] circumvent by using numerical approximations to connect the states. However, the resulting trajectory using RRT* may not be feasible with such an approximation.

All methods above have in common that they use a two-step approach for planning, essentially separating planning and prediction: First, the other vehicles are predicted along a time horizon of a couple of seconds. Second, the reaction of the ego vehicle is planned based on this prediction. However, as long as autonomous vehicles share the road with human drivers, they need to make decisions in an interactive ("What will the other car do if I do this?") and cooperative manner, without any inter-vehicle communication, as otherwise dead-locks or dangerous situations may arise. In a situation like a highway merge, where there is no gap to merge in, but a vehicle may be willing to make space, a prediction model unaware of the ego agent's intention to merge in may predict the car to continue at the same velocity, leaving the planning module with no predicted room to merge.

Interactive Behavior Planning Game-theoretic approaches offer an elegant way to model interactions and dependencies between agents [24–26]. Based on the collaborative assumption, considering every agent's costs, the ego agent can incorporate future interactions with human-driven vehicles into its planning scheme. A decision tree often realizes the game theoretic setting by sampling primitive actions.

Bahram et al. [24] formulate an extensive-form game where the other traffic participants are part of the environment. While the framework is highly flexible, the approach does not ensure convergence to an optimal solution. Schwarting et al. [26] use iterative dynamic programming to solve a multi-agent dynamic, non-zero-sum game. They explicitly model partial observability of the intention of others. The proposed belief-space variant of the iterative Linear Quadratic Gaussian algorithm can be executed in real-time. The exact costs and dynamics of other agents are assumed to be known, and the algorithm converges to a potentially sub-optimal Nash equilibrium. Lenz et al. [25] solve a multi-agent, non-zero-sum dynamic game using Monte Carlo Tree Search (MCTS). A cooperation factor serves as a tuning parameter in the ego agents' cost function. The formulation is highly flexible and can incorporate any transition function for modeling the environment. Kessler et al. [27] use motion primitives to generate a multi-agent motion tree. Applying Mixed-Integer Linear Programming (MILP) to the tree yields an optimal cooperative behavior for a set of agents. Evestedt et al. [28] combine sampling of trajectory candidates from [21] with an iterative prediction of the other traffic participants. Although decision trees are a convenient method to represent game-based formulations, they have one major disadvantage. Due to the search-space discretization, they may fail to obtain valid solutions in convoluted solution spaces, such as dense and complex scenarios.

Optimal control approaches, which promise to make efficient use of the solution space, have employed Mixed-Integer Programming (MIP) for cooperative multi-agent planning [29–31], as it can find an optimal solution to the usually combinatoric problem. Frese et al. [29] formulate a multi-agent optimal control problem and solve it using MILP. Eilbrecht et al. [30] formulate a two-layered approach of iterative conflict resolution using a cooperative cost function, where the underlying trajectory of each agent is optimized for each agent separately. However, both approaches [29, 30] only work for straight driving on straight roads, as they cannot encode the non-holonomy in the optimization problem. Similar problems can be expected from [31], where the abstracted discrete lateral action and state space will complicate applying this approach in reality. To summarize, while the logical properties mirror the nature of – often logical – traffic constraints in a continuous state space in theory [32], there exists no formulation that can be applied to arbitrary roads or driving situations.

Reinforcement Learning can be used to learn a policy. Here, interactions do not need to be modeled explicitly using models or multi-agent settings, but can be learned implicitly. To enable the generalization to unforeseen environments, past works employed prediction or formal methods for safe exploration [33–35], counterfactual reasoning for safe execution [36], learning on abstract semantic representations [37], or fusing RL with search-based methods [38–40]. However, safety, non-interpretability, and efficiency of training data are just some of the problems that remain before such approaches can eventually be applied in safety-critical applications.

End-To-End Planning Both sequential planning and interactive planning operate on an explicit world model, a semantic representation built from the fused perception, consisting of lanes, objects, and other traffic participants. On the other side, end-to-end-planning does not try to construct a world model or explicitly define prediction or interaction models but aims to learn them implicitly. Although there have been some real-world demonstrations [11, 12], verification of such approaches proves to be even more challenging than traditional ones, as it does not allow for any component-wise verification.

1.2.2. State of the Art for Ensuring Traffic Rules Within Planning

This section summarizes what has been proposed in the literature to model rule satisfaction within a planning method.

Behavioral layers that are based on fine state machines allow modeling rules of the road. Practical demonstrations showed rules such as stopping at a stop sign or priorities at intersections [1]. However, their decoupling from motion planning and their lack of interaction limit their usefulness in dense and complex environments.

Search- and sampling-based motion planning techniques can integrate traffic rules by incorporating their penalty in the cost function. Previous work proposed combining model checking techniques for traffic rule satisfaction with motion planning in static environments [6]. However, dense scenarios have proved difficult for such motion planning approaches since they do not consider the interactions with human drivers. This will prevent rules that depend on the other agent's future state to provide an accurate violation penalty in advance. An extension of [6]

to a two-player game, that models both ego agent and the environment (i.e., human drivers) as separate trees, has been proposed in [41]. However, the approach is limited to rules that depend on only one agent, such as "do not cross solid centerlines" or "do not travel in the wrong direction", which only depend on the ego vehicle itself and do not contain a temporal dependency. It cannot incorporate rules such as keeping a safe distance or merging in a zipper fashion. For learning-based methods, the penalty for a rule violation can also be integrated into the reward function during offline training [42]. However, these methods provide no guarantees for rule compliance during online evaluation.

Another direction has been to represent legal aspects such as speed limits or traffic lights as geometric obstacles in space-time [43], often forming spatiotemporal driving corridors that vehicles are allowed to operate in. However, while such an approach works for static rules which can be easily mapped to constraints, it does not scale to more complex behavioral rules with multiple agents.

Karlsson et al. [44] present a sampling-based rule-satisfying algorithm that models interactions, where they model other agents as a Markov Decision Process (MDP). However, their approach relies on a state discretization that will not translate well to dense, complex scenarios. Cai et al. [45] define rules as part of a protocol in a multi-agent environment that all agents follow. However, their approach operates in a discretized action and state space, whereas this work aims for an interactive planning algorithm in continuous state space.

To summarize, current literature only consists of approaches that (1) are based on non-interactive planning and do not translate to interactive behavior planning, (2) can only incorporate rules that rely on the ego agent, e.g., to stay on the right lane, to not enter forbidden areas, and to maintain the speed limit or (3) rely on a discrete state space into cells, that will not work well for real applications. There exists no approach that integrates complex behavioral rules into an interactive behavior planning algorithm.

1.2.3. State of the Art for Traffic Rule Formalization

Traffic rules, which are written in natural language, are often fuzzy and not specified at a level of detail to be comprehensible for machines. Without proper formalization, satisfaction cannot be decided, nor can satisfaction be proved in a safety case for certification.

Works to formalize traffic rules have come from three different communities: First, the planning community, which tries to develop a planner that can follow all applicable rules. Here, the rules are checked on potential predicted outcomes [6, 46, 47]. Second, the safety community, which has tried to establish contracts consisting of a set of rules, which every vehicle should adhere to prove safety [48, 49]. Third, the legal community, which tries to analyze recorded traces to identify liability, which is relevant to insurance companies [50–52].

Vanholme et al. [46] were the first to perform a detailed analysis of the applicable rules for highway driving based on the Vienna Convention on Road Traffic. However, they did not provide a concrete formalization for most of the behavioral rules, such as "overtaking" or "right of way". Decastro et al. [48] formalized the rules regarding lane selection and overtaking. However, they did not consider rules that depend on multiple agents, such as "right of way" or "zipper merge".

Rizaldi et al. [51] provide a formula for a safe distance and use it for the safe overtaking rule [52]. The Responsibility Sensitivity Safety Model (RSS) [49] has formalized the notion of a safe distance and priority. Arechiga [53] employ quantitative semantics about safe distance satisfaction from RSS.

Inspired by the fact that traffic rules shall yield safety and progress to all its participants, the recent work of Pal et al. [54] has tried to design a multi-agent simulation environment for learning such rules emergently instead of implementing them explicitly. However, demonstrating rule compliance within a safety case might be difficult if the learned rules are not really clear even to the developer.

Lanelet2 [55], a map framework for highly automated driving, provides an interface called "regulatory elements" to retrieve traffic signs, traffic light, speed limits, and right of way. For the "right of way", Lanelet2 provides the lanes on which vehicles have the right of way. More elaborate rules, that rely not only on the position of the ego vehicle, such as "overtaking", "distance keeping", or "zipper merge", are not included.

To summarize, most publications may have provided up to a few traffic rules but have not aimed for a comprehensive formalized set of traffic rules, including an operational domain analysis. The only exception is the recent work of Maierhofer et al. [56], which focused on interstate roads.

1.3. Contributions

Existing motion planning methods capable of satisfying traffic rules cannot model interactions, whereas interactive planning methods have not considered complex behavioral traffic rules that depend on the future behavior of multiple agents. No comprehensive machine-interpretable ruleset exists that can be used for planning. This thesis aims to develop a methodology to identify, formalize, model, and evaluate traffic rules within interactive behavior planning. An optimization-based model is developed and solved using MIP. The solver's properties promise to ease the integration of traffic rules. However, to use this solver, the formalized traffic rules must be approximated and cannot be used directly. Thus, a second approach is developed to integrate traffic rules within sampling-based interactive planning. Both approaches are examined in simulation.

This thesis presents the following four major contributions:

1. **Methodology for formalization of traffic rules:** The proposed methodology allows capturing traffic rules from legal texts in a machine-interpretable language. It consists of an informal graphic abstraction and the transfer of the rules to a logical language. A ruleset for a specific operational driving domain is obtained by following the methodology, and a rule monitor framework is provided to check trajectories on rule satisfaction. The framework is embedded in an open-source simulator. The traffic rules are evaluated on human data to check the rules on plausibility and provide insight into how humans follow these rules.

2. **Interactive planning using optimal control:** The proposed optimization program solves a linear differential game for a set of interacting agents using MIP. A disjunctive formulation preserves the vehicle model's non-holonomic motion properties for any orientation and arbitrary formed roads.

3. **Traffic rules satisfaction in interactive planning:** The proposed optimization-based approach cannot incorporate the rule monitors directly. Modeling rule compliance is demonstrated for a rule subset, where the desired property is realized by either calculating a forbidden space before optimization or by encoding it in the reference. In contrast, sampling-based interactive planning methods do not have the same limitations as optimization-based methods in terms of the cost function. As any function can be used to calculate the cost, this work makes direct use of the rule monitor, which creates an automaton for each rule. The state of the rule automaton is encoded within the decision tree to efficiently evaluate the rules.

4. **Simulation-based validation:** The proposed methods to model rule satisfaction are studied in simulation. Within this thesis, an open-source simulator has been developed. It allows evaluating the ramifications of modeling a particular rule on the violation itself, but also collision and progress metrics.

1.4. Publications

This thesis is based on the following publications:

1. K. Esterle, V. Aravantinos, and A. Knoll. "From Specifications to Behavior: Maneuver Verification in a Semantic State Space." In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 2140–2147

2. K. Esterle, T. Kessler, and A. Knoll. "Optimal Behavior Planning for Autonomous Driving: A Generic Mixed-Integer Formulation." In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 1914–1921

3. J. Bernhard, K. Esterle, P. Hart, and T. Kessler. "BARK: Open Behavior Benchmarking in Multi-Agent Environments." In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 6201–6208

4. K. Esterle, L. Gressenbuch, and A. Knoll. "Formalizing Traffic Rules for Machine Interpretability." In: *2020 IEEE 3rd Connected and Automated Vehicles Symposium (CAVS)*. 2020, pp. 1–7

5. K. Esterle, L. Gressenbuch, and A. Knoll. "Modeling and Testing Multi-Agent Traffic Rules within Interactive Behavior Planning." In: *IROS 2020 Workshop "Perception, Learning, and Control for Autonomous Agile Vehicles"*. 2020

6. T. Kessler, K. Esterle, and A. Knoll. "Linear Differential Games for Cooperative Behavior Planning of Autonomous Vehicles Using Mixed-Integer Programming." In: *2020 59th IEEE Conference on Decision and Control (CDC)*. 2020, pp. 4060–4066

1.5. Structure of this Thesis

The relationship of the building blocks of this thesis is shown in Figure 1.3. In Chapter 3, a subset of traffic rules for Germany is formalized. Linear Temporal Logic is used as a formal language to codify behavior. Chapter 4 introduces the basics to evaluate Linear Temporal Logic on finite traces. A runtime monitor is implemented for each formalized rule. The rules are analyzed on a public dataset of interactive scenarios. By this, levels of satisfaction for each respective rule are obtained.

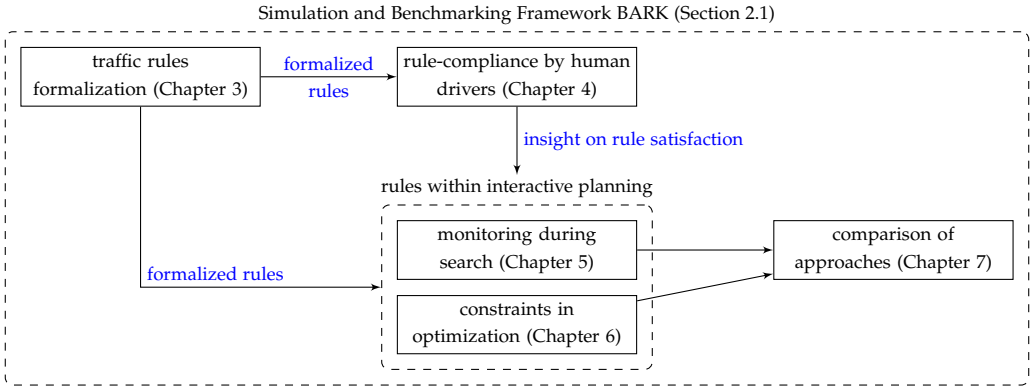


Figure 1.3.: Relationship of chapters of this thesis.

Chapter 5 and Chapter 6 provide two orthogonal ways to model traffic rules as part of an interactive behavior planner. In Chapter 5, a concept is presented to monitor multi-agent and time-dependent traffic rules at runtime within the sampling-based MCTS. Specifically, the proposed method models non-Markovian traffic rules within MCTS. To study the effect of modeling a certain traffic rule within a set of rules, the ruleset is treated to be priority-ordered and incorporated to MCTS by leveraging methods from multi-objective optimization theory. In Chapter 6, the rules are realized as constraints in an optimal control approach. Both approaches are implemented within the Behavior Benchmarking Framework (BARK), which is described in Section 2.1. Chapter 7 compares the approaches from Chapter 5 and Chapter 6 in terms of the rules that can be modeled, the engineering effort for integrating the rules, the computational effort for additional rules, and more. Finally, future directions for research are given.

2 Preliminaries

This chapter introduces the simulation and behavior benchmarking framework BARK. It is a multi-agent environment tailored to develop interactive behavior models and allows to easily exchange and reuse behavior models for planning, prediction and simulation. BARK already offers a range of behavior models, which can be used for predicting and simulating other agents. It provides efficient collision checking and realistic maps. The work of this thesis has been integrated into BARK to conduct experiments. After the introduction of BARK follows the theory and base implementation of MCTS, upon which Chapter 5 will build.

2.1. Benchmarking Behavior in Simulation

2.1.1. Overview and Contribution

Predicting and planning interactive behaviors in complex traffic situations presents a challenging task. Especially in scenarios involving multiple traffic participants that interact densely, autonomous vehicles still struggle to interpret situations, which eventually compromises the autonomous vehicle's mission goal. As driving tests are costly and challenging scenarios are hard to find and reproduce, simulation is widely used to develop, test, and benchmark behavior models. However, most simulations rely on datasets and simplistic behavior models for traffic participants and do not cover the full variety of real-world, interactive human behaviors. This section introduces BARK, an open-source behavior benchmarking environment designed to mitigate the shortcomings stated above. In BARK, behavior models can be used for planning, prediction, and simulation.

This section is based on the author's joint work with Julian Bernhard, Patrick Hart and Tobias Kessler [58].

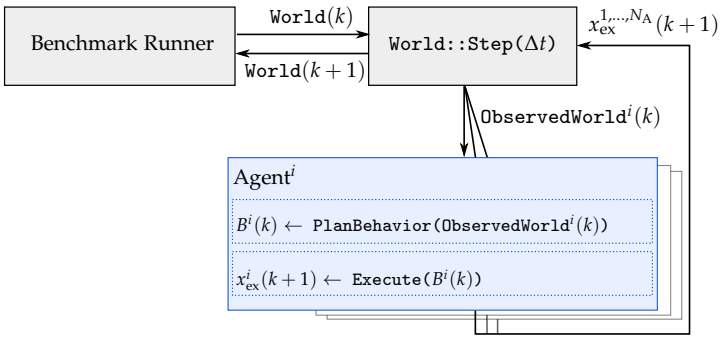


Figure 2.1.: BARK's simulation loop is handled by the benchmark runner holding the current world state at discrete world time k. In each iteration, the benchmark runner calls `World::Step(Δt)`. This function generates an `ObservedWorld` for each agent and passes it to the agent's internal `BehaviorModel` $\mathcal{B}^i$ which generates a behavior $B^i(k)$. The behavior is passed to the agent's `ExecutionModel` calculating the next executed agent state $x^i_{\text{ex}}(k+1)$. The next world state at time $t(k+1) = t(k) + \Delta t$ integrates the updated agent states for all agents of size N_{A} and is returned to the *Benchmark Runner* (modified graphic from [58], ©2020 IEEE).

2.1.2. BARK in a Nutshell

BARK focuses on providing a software framework for the systematic evaluation and improvement of behavior models. The same implementations can be used to either plan the motion of a vehicle, predict other vehicles' motions, and forward simulate a driving scenario. For example, a traffic model, such as the Intelligent Driver Model (IDM) [62], can, on the one hand, be used to populate a simulation with agents but also as a generative model to predict other agents' motion from the viewpoint of the ego vehicle. To ensure this, BARK models the world as a multi-agent system with agents performing *simultaneous movements* in the simulated world. Figure 2.1 visualizes the core concept of BARK's simulation. A benchmark runner calls the simulated world's step-function at fixed, discrete world time-steps. Each agent then plans its next action using its behavior model, which only has access to the agent's observation of the world, called observed world. This observed world may not necessarily be equal to the simulator's correct world. The concept of simultaneous movement ensures that a behavior model can plan based on reproducible input information. It avoids timing artifacts that may occur in message-passing, middleware-based simulation architectures.

BARK uses behavior models not only for behavior planning but also for predicting other agents in the world. As the observed world of each planner derives from the actual world, it can be used to obtain predictions by calling `ObservedWorld::Step(Δt)` multiple times. All agents in this observed world behave according to their prediction configuration. Figure 2.2 visualizes BARK's observed world model. By deriving an observed world with various prediction configurations from the actual world definition, the behavior planner's potential errors caused by inaccurately predicting other traffic participants can be systematically examined.

BARK provides several state-of-the-art behavior models ranging from conventional planning to machine learning approaches. BARK's current behavior model implementation will be discussed in Section 2.1.3. The following section will take a more detailed look at other BARK components.

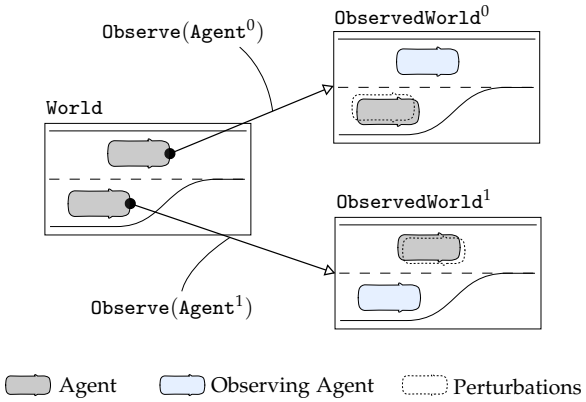


Figure 2.2.: Each agent in BARK employs an observed world for planning its behavior. Each observed world holds an observing agent (depicted in blue) from whose perspective the observation is being made. Perturbations can be injected by, e.g., exchanging the other agents' behavior models and model parameters in the observed world (modified graphic from [58], ©2020 IEEE).

World and Observed World Model The `World` model contains the map, all objects, and agents. Object lists represent static and dynamic objects. This work will refer to dynamic objects as agents in the following. The `ObservedWorld` model, on the other hand, reflects the world that is perceived by an agent i. BARK's `ObservedWorld` model accounts for the fact that the observing agent has no access to other agents' actual (world) behavior model. BARK can model different degrees of observability by either entirely restricting access to the world behavior model or only by perturbing the world behavior model's parameters. Additionally, occlusions and sensor noise can be introduced in the `ObservedWorld` model.

Agent Models Figure 2.1 shows the two main interfaces of an agent:

- $B^i(k) \leftarrow$`Agent::PlanBehavior(ObservedWorld`$^i(k)$`)`: calls the agent's behavior model $\mathcal{B}^i$, which generates a behavior trajectory $B^i(k) = \left(x^i_{\text{des}}(k), x^i_{\text{des}}(k+1), \ldots, x^i_{\text{des}}(k+K)\right)$, being a sequence of desired future physical agent states $x^i_{\text{des}}(k) = (t, p_x, p_y, \theta, v)$ between current simulation world time $t(k)$ and at least the end time of the simultaneous movement $t(N) \geq t(k) + \Delta t$. The time discretization of the behavior trajectory can be arbitrary.
- $x^i_{\text{ex}}(k+1) \leftarrow$ `Agent::Execute(`$B^i(k)$`)`: calls the agent-specific execution model determining the agent's next state $x^i_{\text{ex}}(k+1)$ in the world based on the generated behavior trajectory $B^i(k)$. The interface could allow examining the robustness of behavior planners against execution errors systematically. As of now, an interpolation-based is employed as an execution model.

To implement these two primary interfaces, an agent holds the following additional agent-specific information:

- `GoalDefinition`: BARK considers agents to be goal-driven. It provides an abstract goal specification with several inherited types of agent goals, e.g., geometric goal regions or lane-based goals. Each agent holds a single goal definition instance.

- `RoadCorridor`: During initialization, an agent computes the roads and corresponding lanes necessary to reach its goal. The topology information on how roads are connected is extracted from the map. Also, the geometric data of the map, such as lane boundaries, are being discretized. This precomputation avoids computational overhead during the simulation.

- `Polygon`: A two-dimensional polygon defines the shape of the agent.

Scenario A BARK scenario contains a list of agents with their initial states, behavior models, execution models, and a goal definition for each agent. Further, it contains a map file in the OpenDrive format. To support the benchmark of a behavior model, each scenario specifies which agent is considered the "controlled" agent during the simulation. A BARK scenario does not explicitly specify how agents will behave over time, e.g., using predefined maneuvers or trajectories.

BARK provides a scenario generation module for configuring sets of scenarios. This modularized concept allows to model a variety of scenario types and enables an easy extension with novel scenario types.

Software Design BARK has a monolithic, single-threaded core, written in C++. Most of the expensive calculations (e.g., geometry functions) are part of the core. The core of BARK is wrapped in Python using pybind [63] including pickling support for all C++ types. Scenario generation and benchmarking, as well as service methods for parameter handling and visualization, are implemented in Python. A simulator step is entirely deterministic, which enables the simulation and experiments to be reproducible. Bazel [64] is being used as the build system. Its sandboxed build environment simplifies the reproducibility of experiments since dependency versions can be tracked easily over multiple repositories.

2.1.3. Behavior Modeling for Realism in Simulation

BARK contains a set of reference implementations for behavior models from the literature. All behavior models inherit from the abstract BehaviorModel class:

```
class BehaviorModel {
  public:
    ...
    virtual Trajectory Plan(float delta_time, const ObservedWorld& world)) = 0;
    ...
};
```

Behavior models overload the `Plan(·)` function and return a time-dependent state trajectory. Since all behavior models share the same interface, this makes them easily exchangeable. By accessing the ObservedWorld, a behavior model can query egocentric semantic information, e.g., `GetLeftLane()→GetCenterLine()`, as well as semantic relations to other agents, e.g., `GetAgentInFront()`. Together with the `Evaluator` concept and implemented standard

behavior models, this allows for rapid prototyping and development of new behavior models in BARK. The relevant behavior models for this work are highlighted in the following.

Intelligent Driver Model The popular IDM [62] is an easy-to-implement vehicle-following model often used in microscopic traffic simulation. To model maintaining a gap from the vehicle in front, it leverages the spacing between the agents, speed asymmetries, and realistic braking profiles. The model cannot handle lane changes, intersections, and unstructured scenarios. To support larger world step times in BARK, the IDM implementation assumes that the preceding vehicle maintains a constant velocity throughout a world simulation step.

MOBIL Model For multi-lane scenarios, the Minimizing Overall Braking Induced by Lane Changes (MOBIL) model was implemented in BARK, a decision-making entity to trigger lane changes to improve traffic flow [65]. It maintains a politeness factor, which captures how much other agents would be slowed down by a potential lane change. In case of a lane change decision, the centerline of the respective left or right driving corridor is started to be tracked [66]. The longitudinal motion is controlled using the IDM.

2.1.4. Evaluator Concept for Benchmarking

For the systematic development of behavior models, BARK supports large scale evaluation of behavior models over a collection of scenario sets. The benchmark runner shown in Figure 2.1 allows to evaluate specific behavior models with different parameter configurations over the entire benchmarking database. The evaluation is based on an abstract evaluator interface calculating a Boolean, integer or real-valued metric based on the current simulation world state.

The existing evaluators in BARK are:

- `StepCount`: returns the step count the scenario is at.
- `GoalReached`: checks if a controlled agent's `GoalDefinition` is satisfied.
- `DrivableArea`: checks whether the agent is inside its `RoadCorridor`.
- `Collision`: checks whether any agent or only the currently controlled agent collides.
- `GoalDistance`: calculates an euclidean distance to the `GoalDefinition`.

Evaluators can be used for benchmarking and internally by the behavior models, e.g., for the reward calculation in search- or reinforcement learning-based planners. The `BenchmarkRunner` runs each scenario of the database evaluating world states of the simulation. It terminates the scenario run based on criteria defined with respect to the evaluators.

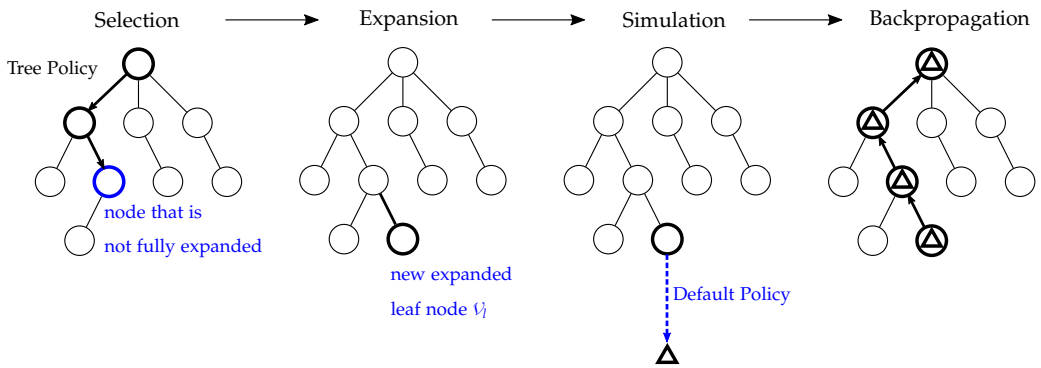


Figure 2.3.: Four steps performed for each iteration of the MCTS algorithm (taken, reproduced and modified from [67]).

2.2. Monte Carlo Tree Search

2.2.1. Monte Carlo Tree Search for Behavior Planning

Monte Carlo Tree Search (MCTS) is an online tree search method to approximate the solution of sequential decision problems via sampling [67]. Each node v of the search tree maintains four information: The state s, the incoming action a, the state-action-value Q, and the number of visits n. Throughout the search, the estimate of $Q(s,a)$, which maps the expected cumulative reward of performing action a in state s, is updated iteratively. Each iteration consists of *selection*, *expansion*, *rollout* (also called *simulation*), and *backup*, which are illustrated in Figure 2.3.

For *selecting* a new node, the tree is traversed following the tree policy until a leaf node, i.e., a node with untried actions, is reached. During *expansion*, an untried action is applied, and the resulting child node is added to the tree. A value estimate is obtained from the newly expanded node using a heuristic *rollout*: Based on a default policy, actions are applied until a terminal state is reached. The observed value is *backed up* to the root node and used to update $Q(s,a)$. The search is repeated until some predefined computation budget (time or iterations) is reached. Finally, the best performing root action is returned.

MCTS has been adapted to interactive driving by using information sets assuming simultaneous movements of traffic participants [25, 38]. Lenz et al. [25] apply it to the context of cooperative planning, meaning that they introduce a joint cost function, which minimizes the costs for all agents. The size of the search tree grows exponentially with the number of agents. Thus, only a subset of all agents is included in the joint action, while the actions of the remaining agents are based on a predefined model. A special case of this is called *Single-Agent MCTS*, where all other agents are predicted using this model.

2.2.2. Monte Carlo Tree Search in BARK

A single-agent variant of MCTS based on a template-based open-source library [68] has been implemented in BARK and was presented in [58]. Algorithm 1 shows the implementation in pseudocode. With a slight abuse of notation, $s(v)$ yields state s of node v. Effectively, $s(v)$ is an

instance of `ObservedWorld`. During expansion and simulation, a new child node $\mathcal{V}'$ is obtained from the function `BarkStep`$(\mathcal{V}, a)$, where action a is applied for the controlled agent, and the others behave according to their behavior model. The other agents can be configured to follow any available behavior model in BARK. These are set via the prediction setup of the observed world before the MCTS algorithm is started. As a selection strategy, Upper Confidence Bounds for Trees (UCT) is used, which balances exploitation and exploration:

$$\texttt{SelectUct}(s) := \underset{a \in A}{\arg\max} \left(\underbrace{Q(s,a)}_{\text{Exploitation}} + \underbrace{C_{\text{UCT}} \sqrt{\frac{2 \ln n(s)}{n(s,a)}}}_{\text{Exploration}} \right), \tag{2.1}$$

where $n(s,a)$ is the number of times action a has been selected from state s, $n(s) = \sum_a n(s,a)$ denotes the total node visit, and C_{UCT} denotes a constant to tune the exploration. During the simulation phase, random actions are selected for the ego agent.

A reward $r(s,s',a)$ is given for each transition from s to s' when applying a. The cumulative discounted reward follows as

$$\triangle = \sum_{0}^{K} \gamma^k r(k), \tag{2.2}$$

where K denotes the number of steps and γ denotes a discount factor.

Algorithm 1 MCTS algorithm with UCT selection strategy (adopted from [67]).

function MctsSearch(s_0)
  create root node v_0 from state s_0
  while within computational budget **do**
    $v_l \leftarrow$ TreePolicy(v_0) ▷ Selection and Expansion
    $\triangle \leftarrow$ DefaultPolicy(v_l) ▷ Simulation
    BackUp($v_l, \triangle$) ▷ Backpropagation
  return $\arg\max_{a \in A} \left(Q(s_0, a)\right)$
function TreePolicy(v)
  while $s(v)$ is non-terminal **do**
    if v has untried actions **then** ▷ Expansion
      randomly choose action a from untried actions
      add $v' \leftarrow$ BarkStep(v, a) as new child node to v
    else ▷ Selection
      select best child node v' from best action $a \leftarrow$ SelectUct$(s(v))$
      $v \leftarrow v'$
  return v
function DefaultPolicy(v)
  $\triangle \leftarrow 0$
  while $s(v)$ is non-terminal **do**
    randomly choose action a
    $v' \leftarrow$ BarkStep(v, a)
    $s \leftarrow s(v)$ and $s' \leftarrow s(v')$
    $\triangle \leftarrow \triangle + \gamma r(s, s', a)$
    $v \leftarrow v'$
  return $\triangle$
function BackUp($v', \triangle$)
  $(v, a) \leftarrow$ parent of v' and action leading from v to v'
  while $v \neq \emptyset$ **do**
    $s \leftarrow s(v)$ and $s' \leftarrow s(v')$
    $n(s, a) \leftarrow n(s, a) + 1$
    $Q(s, a) \leftarrow Q(s, a) + \frac{\triangle - Q(s,a)}{n(s,a)}$
    $\triangle \leftarrow r(s, s', a) + \gamma\triangle$
    $v' \leftarrow v$
    $(v, a) \leftarrow$ parent of v' and action leading from v to v'

(Update node statistics)

3 Formalization of Traffic Rules for Machine Interpretability

3.1. Overview and Contribution

Traffic regulations such as the Strassenverkehrsordnung (StVO) [69], which is the German concretization of the Vienna Convention on Road Traffic [70], define rules all drivers should obey. These traffic rules are often fuzzy and subject to interpretation, encouraging the need for a formalized machine-interpretable definition of traffic rules. No work has yet provided a comprehensive and consistent set of traffic rules for a specific operational domain. Likewise, there is no methodology to derive such rules.

Based on the operational design domain of dual carriageways in Germany, this chapter provides an analysis of the applicable traffic rules. Characteristics of these rules are then aggregated and used to identify a suitable machine-interpretable formalism into which the natural language shall be translated. The contributions of this chapter are

- a methodology to formalize traffic rules from legal texts to a formal language and
- a formalized set of traffic rules for dual carriageways.

The chapter is based on the author's previously published work [59].

3.2. Legal Analysis of German Traffic Rules on Dual Carriageways

At first, the Operational Design Domain (ODD) needs to be defined. This thesis analyzes traffic rules for a passenger vehicle. The vehicle does not exceed 3.5 tons, is legally allowed to drive on motorways, and has no trailer. The formalization will be based on the German Road Traffic Regulations StVO[1], while also including references to the Vienna Convention on Road Traffic [70]. Specifically, rules applicable to dual carriageways will be analyzed, such as highways. Rules that

[1] An English translation of the StVO is available at `https://germanlawarchive.iuscomp.org/`

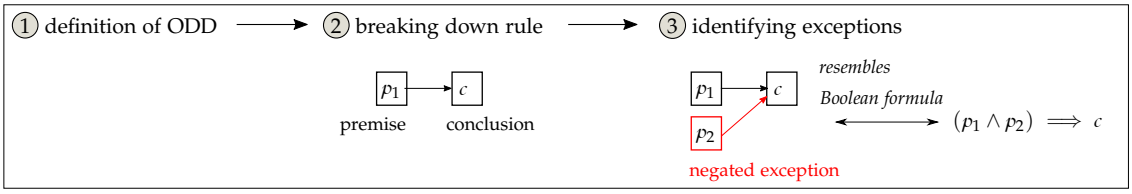


Figure 3.1.: Legal analysis to identify abstract Boolean formulas.

include pedestrians or cyclists will not be considered. Without limiting the generality, this work assumes right-handed traffic and the carriageways to be structurally separated.

This work focused on behavioral rules for road users, especially for multiple road users involved. Thus, the following special cases will not be considered:

- parking, breakdowns, and towing,
- necessary post-accident actions including clearing,
- signaling such as indicator signals or lighting,
- regulatory signs, including lane markings, informatory signs, traffic installations, and
- limited visibility.

To identify all relevant rules and remove ambiguity in them, the following methodology will be used:

1. Identify a rule and separate it into an initial premise and a conclusion. If there is no premise, start with "always".
2. Identify all exceptions to the premise. Use negated exceptions to update the premise.

Using Boolean algebra, the initial premise and exceptions can be combined by conjunction. A graphical representation is used for identification and aggregation. Implications are illustrated by an arrow. Exceptions are marked in red. Figure 3.1 summarizes this process.

3.2.1. Regulations on Speed

This section analyzes speed regulations.

Rule 3.2.1.1: Keep Control
"A person operating a vehicle may only travel at a speed that allows them to be in constant control of their vehicle" [§3(1) StVO].

Control is lost if vehicle tires cannot exert the required forces on the road. This happens when the lateral or longitudinal accelerations exceed the limits of the friction circle. This is not only a rule but also a safety requirement for any autonomous vehicle. Hence, this work assumes control algorithms to limit the requested accelerations accordingly.

Rule 3.2.1.2: Above Minimum Speed

"No motor vehicle must, without good reason, travel so slowly as to impede the flow of traffic" [§3(2) StVO; similar to VC 13.4].

Decastro et al. [48] define "impeding the traffic flow" as $|v^i - v_{\text{mean}}| > \epsilon_v$, where v_{mean} denotes the average speed of the surrounding vehicles and ϵ_v denotes a threshold.

Rule 3.2.1.3: Below Speed Limit

Adhere to the "maximum permissible speed" [§3(3) StVO].

This work assumes the maximum speed limit to be available from a map.

Rule 3.2.1.4: No Stopping

On motorways and motor roads, "stopping is prohibited, including on verges" [§18(8) StVO].

Motorways are roads that are only allowed for motor vehicles and have specific entry and exit terminals [70]. They usually consist of separate carriageways for two-way traffic. Motor road is an old-fashioned name for motorway [71]. The road type is assumed to be available. Figure 3.2 shows the codified speed limit rules.

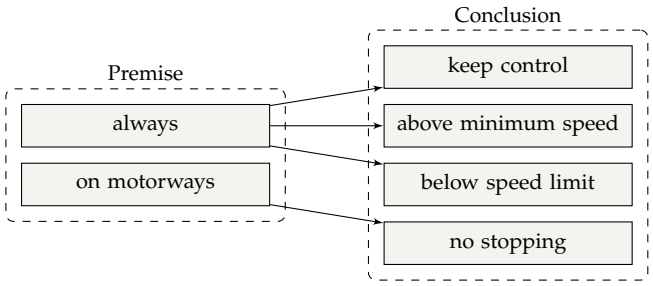


Figure 3.2.: Speed limit rules separated into premise and conclusion (graphic from [59], ©2020 IEEE).

3.2.2. Regulations on the Use of Roads and Lanes

This section analyzes rules that specify which lanes should be used by motorists.

Rule 3.2.2.1: Keep Right
"Keep as far to the right as possible" [§2(2) StVO; similar to VC 10.3].

This is often referred to as "staying on the right side". However, in dense traffic, drivers might ignore the keep-right directive:

Rule Exception 3.2.2.1: Dense Traffic with Multiple Lanes
"This might be ignored on carriageways with several lanes for one direction, [...] if traffic density justifies" [§7(1) StVO].

Wuthishuwong et al. [72] define traffic density as the number of vehicles per lane that measure to the 1 km length of the observed street. However, there are currently no threshold values for dense traffic that autonomous vehicles could use. Another exception is made within built-up areas[2] on roads that are no motorways:

Rule Exception 3.2.2.2: Built-up Area
"On carriageways with several marked lanes for one direction of traffic [...] within built-up areas – with the exception of motorways – [...], vehicles [...] are free to choose their lane, even at no dense traffic" [§7(3) StVO].

An exception to the keep-right directive exists for outside built-up areas on roads with more than two lanes for one-way traffic.

Rule Exception 3.2.2.3: Outside Built-up Area With Three or More Lanes
"Outside built-up areas [with] three lanes for one direction of traffic, vehicles may, in derogation from the rule that they must keep as far to the right as possible, [stay in] the middle lane in places where – even if only now and then – a vehicle is stationary or moving in the nearside lane. On carriageways with more than three lanes marked in this way for one direction of traffic, the same applies to the second lane from the right" [§7(3c) StVO].

However, this exception brings in a new rule. By using this exception as a non-negated premise, a new rule can be defined as "keep outside the left-most lane". Figure 3.3 shows the codified lane selection rules.

[2] "Inside a built-up area" is a synonym for inner-city.

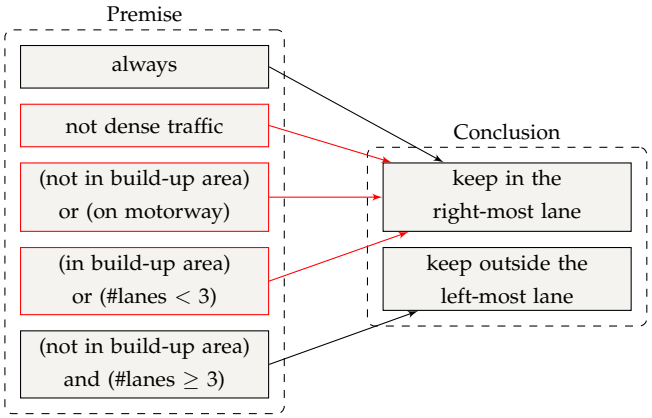


Figure 3.3.: Lane selection rules separated into premise and conclusion (graphic from [59], ©2020 IEEE).

3.2.3. Regulations on Overtaking

An explicit definition for overtaking is missing in the StVO and in the Vienna Convention on Road Traffic. Whereas Rizaldi et al. [52] define overtaking as the process of changing lanes, passing a vehicle, and returning to the initial lane, court rulings have clarified that passing a vehicle is already considered overtaking [73]. This work follows this definition and replaces overtaking with passing whenever necessary.

Rule 3.2.3.1: Speed Advantage During Overtaking

Only overtake if the ego vehicle travels "at a speed substantially higher than that of the vehicle to be overtaken" [§5(2) StVO].

As the term "substantially higher speed" is vague, court rulings have since clarified the minimal speed advantage for trucks to be 10 km/h [74]. Missing other concrete values, this work will use this value for passenger vehicles as well.

Rule 3.2.3.2: Overtaking Maneuver

"Make sure that traffic approaching from behind is not endangered. [Keep] a sufficient lateral distance [...] from other road users [...]. Move back to the right-hand side of the road as soon as possible" [§5(4) StVO; similar to VC 11.2a, VC 11.4].

This overtaking rule can be divided into three parts. First, when changing to the outer lane, traffic shall not be jeopardized. Rizaldi et al. [52] define this as keeping a safe distance to the new follower. Second, sufficient lateral space will be inherently satisfied by a motion planner. Thus, it will not be considered explicitly in the following. Third, a vehicle shall move back as soon as possible. Following Rizaldi et al. [52], the phrase "as soon" means when a safe distance to the new follower can be established. However, as stated before in Section 3.2.2, there are multiple exceptions to this rule. This rule will be relaxed in this work and interpreted as "do not return to the initial lane before a safe distance can be established". This means that when performing a lane-change, a safe distance to the rear vehicle should be ensured.

Rule 3.2.3.3: No Passing on the Right Side

Only "overtake [...] on the left side" [§5(1) StVO; similar to VC 11.1].

This rule will be interpreted to apply to the passing of a vehicle. The rule then also implies that vehicles on the right lane should not travel faster than those on the left. However, there are several exceptions for special lane types:

Rule Exception 3.2.3.1: Diverging Lane

"Where lanes diverge from the main carriageway [...] vehicles turning off may [...] travel faster than traffic on the main carriageway" [§7a(1) StVO].

Rule Exception 3.2.3.2: Acceleration Lane

"On motorways and other roads outside built-up areas, vehicles may travel faster in acceleration lanes than traffic on the main carriageway" [§7a(2) StVO].

Rule Exception 3.2.3.3: Deceleration Lane

"If traffic on the main carriageway is moving slowly or is stationary, vehicles in a deceleration lane may overtake at a moderate speed" [§7a(3) StVO].

There is another exception for built-up areas:

Rule Exception 3.2.3.4: Build-up Area

"On carriageways with several marked lanes for one direction of traffic [...] within built-up areas – with the exception of motorways – [...], traffic on the right may move faster than traffic on the left" [§7(3) StVO].

The condition for several lanes can be skipped, as overtaking would not be possible anyway (for dual carriageways). Also, there are exceptions for dense traffic:

Rule Exception 3.2.3.5: Dense Traffic

In dense traffic, "traffic on the right (nearside lane, middle lane) may move faster than traffic on the left" [§7(2) StVO].

Rule Exception 3.2.3.6: Dense Traffic

Vehicles queues at low speed or standstill may be overtaken "on the right" [§7(2a) StVO; similar to VC 11.6].

This work argues that the above two exceptions essentially have the same meaning if overtaking is interpreted as passing: In dense traffic, passing vehicles on the right side is allowed. Figure 3.4 shows the codified overtaking rules.

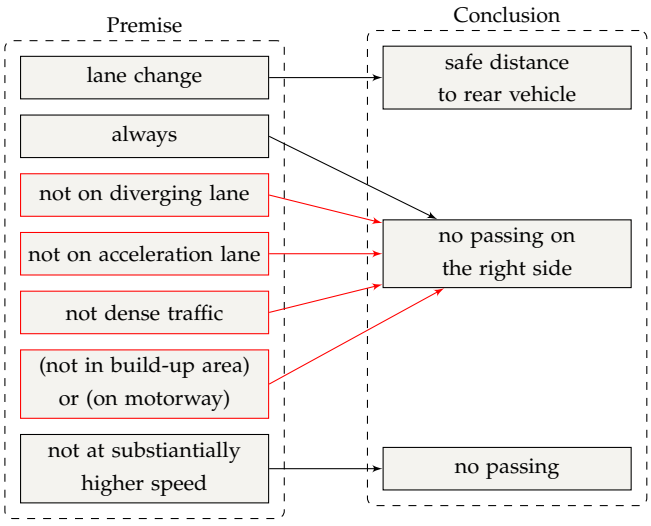


Figure 3.4.: Overtaking rules separated into premise and conclusion (modified graphic from [59], ©2020 IEEE).

3.2.4. Regulations on a Safe Distance

A driver shall always keep a safe distance to a preceding vehicle:

Rule 3.2.4.1: Safe Distance

"A person operating a vehicle moving behind another vehicle must, as a rule, keep a sufficient distance from that other vehicle so as to be able to pull up safely even if it suddenly slows down or stops" [§4(1) StVO; similar to VC 13.5].

This rule has been treated frequently in literature [6, 46, 49, 51], although implementations vary in the order of state derivatives used to calculate the distance. Estimating the maximum possible braking deceleration is also not clearly defined and may change depending on the road surface. Figure 3.5 shows the codified distance rule.

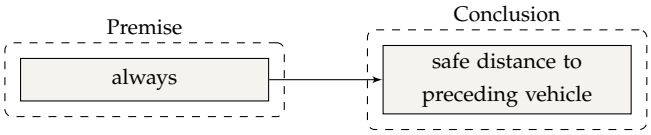


Figure 3.5.: Safe distance rule separated into premise and conclusion (graphic from [59], ©2020 IEEE).

3.2.5. Regulations on Being Overtaken

A vehicle being overtaken shall obey to the following:

Rule 3.2.5.1: Being Overtaken

A vehicle "being overtaken must not increase the vehicle's speed" [§5(6) StVO].

Being overtaken will be defined to be close to a vehicle on the left lane, which is similar to the definition used in [48]. This is sufficient, as overtaking on the right is prohibited. Figure 3.6 shows the codified rule when being overtaken.

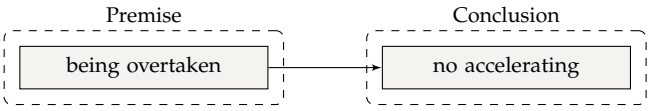


Figure 3.6.: Being overtaken rule separated into premise and conclusion (graphic from [59], ©2020 IEEE).

3.2.6. Regulations on Priorities

This section deals with priority rules. Giving way means that a driver "continues or resumes his advance or maneuver if by so doing he might compel the drivers of other vehicles to change the direction or speed of their vehicle abruptly" [70]. If a vehicle has the right of way, the other drivers shall be giving way.

Rule 3.2.6.1: Right of Way

On "motorways and motor roads [...] traffic on the main carriageway has the right of way" [§18(3) StVO; similar to VC 25.2].

Rule 3.2.6.2: Zipper Merge

"If, on roads with several lanes for one direction, uninterrupted travel in one of the lanes is not possible, or if a lane comes to an end, vehicles traveling in the adjacent lane must allow vehicles in the other lane to change lanes immediately before the road narrows, in such a way as to let them join their line of traffic in turn after each vehicle traveling in the uninterrupted lane" [§7(4) StVO].

The rule demands that vehicles should merge at the end of the lane in an alternating zip fashion from both lanes. Following the zipper merge has proved to reduce congestion while ensuring the safety of motorists, as the complexity in changing lanes is removed. Some US states have begun adopting this concept [75]. If a driver in a continuing lane does not obey the zipper merge, a driver wishing to merge is not allowed to enforce it [76]. Thus, if an accident occurs during the zipper merge, the blame is often shared [76, 77]. The zipper merge does not apply on accelerating lanes of motorways [78], where right of way has been identified to be valid. Figure 3.7 shows the codified priority rules.

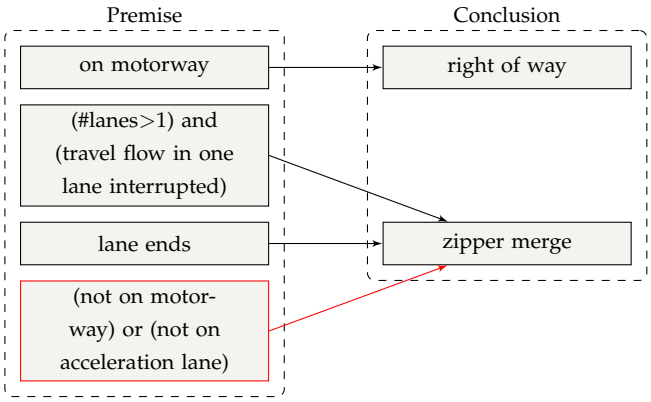


Figure 3.7.: Priorities rules separated into premise and conclusion (modified graphic from [59], ©2020 IEEE).

3.3. Identifying a Suitable Language for Machine Interpretability

Logical languages are a formal way to represent rules. A logical language needs (1) to be expressive enough to codify natural language and (2) to have a mechanism for model-checking the formulas.

3.3.1. Characteristics of Traffic Rules in Dense Highway Scenarios

Some rules are only based on the ego agent, such as staying below the speed limit or to keep in the rightmost lane. Other rules however need to be evaluated with respect to the other vehicles within the sensor range. For example, keeping a safe distance or not passing on the right side require to be evaluated with respect to each other agent. Even more, some rules depend on two or more other agents, such as the zipper merge. Table 3.1 shows the number of other agents for each rule.

Some rules only depend on the current time instance, which means that they can be evaluated without any past information, such as the safe distance rule, staying below the speed limit or to keep in the rightmost lane. Other rules however state a temporal evolution of a scene. For example, an overtaking maneuver cannot be detected based on only one time instance. Likewise, rules such as the right of way or the zipper merge cannot be evaluated based on only one instance of the scene. This can be viewed as non-Markovian property. Creating a Markovian state from this by also including the last two or three time instants is not sufficient here, as this would depend on the rule and the specific scene. Thus, the markovian property should be achieved by encoding historical information on a more abstract level.

This thesis refers to these rules as *multi-agent, time-dependent traffic rules*. Selecting a common formalization concept to cover the formalization of all these rules will be the subject of the following section.

Table 3.1.: Number of other agents the respective rules depend on. The rule classes are Lane Selection (LS), Overtaking (OV), Distance (DIST), Being Overtaken (BOV) and Priority (PRIO).

Class	Rule	Number of other agents
VEL	Below speed limit	0
VEL	No stopping	0
LS	keep in rightmost lane	0
LS	keep outside leftmost lane	0
OV	no passing on the right side	1
OV	safe lane change	1
OV	speed advantage during overtaking	1
DIST	safe distance	1
BOV	being overtaken	1
PRIO	right of way	1
PRIO	zipper merge	2

3.3.2. Selection of a Formalization Language

Different techniques on how to model the rules have been employed in the past. To identify the limitations of a specific formalism, Table 3.2 maps the most relevant formalization works in the literature to the identified rules from the legal analysis.

Vanholme et al. [46] use inequality comparisons of real numbers to express the rules. However, they do not provide a concrete formalization for most behavioral rules, such as overtaking, right of way or the zipper merge. Decastro et al. [48] also use inequality comparisons of real numbers to formalize lane selection rules, overtaking rules, and others. Their overtaking rules rely on velocity instead of a sequence of positions and thus circumvent the formalism's lack of a temporal operator. However, Decastro et al. [48] do not consider rules such as right of way on four-way intersections or the zipper merge, where this approach would not work anymore. RSS also uses inequality comparisons of real numbers but only consists of the notion of a safe distance and priority [49]. They also formulate right of way for priority lanes.

Castro et al. [6] use Linear Temporal Logic (LTL) to express traffic rules such as "do not cross solid centerlines" or "do not travel in the wrong direction", which only depend on the ego vehicle itself and do not contain any temporal aspect. Rizaldi et al. [51] define a safe distance and prove its correctness using a theorem prover. In [52], they employ this safe distance function for the safe overtaking rule, which they formulate in LTL.

The recent work of Maierhofer et al. [56] provides one of the most complete rule sets so far. It has been published at the same time as the work [59] of this thesis' author. Maierhofer et al. use Metric Temporal Logic (MTL) [79], which is an extension of LTL. Compared to LTL, time-constrained versions replace the temporal operators. It thus allows specifying properties to be true within a time interval instead. Other works rely on Signal Temporal Logic (STL) [80] to obtain quantitative semantics about rule satisfaction [53, 81]. Such quantitative semantics might be beneficial for relaxing the requirements to satisfy a rule.

Table 3.2.: Rule formulations in the literature. The works of Vanholme et al. [46], Decastro et al. [48], Shalev-Shwartz et al. [49], Rizaldi et al. [52], [56] and Koschi et al. [82] are categorized along the technology being used, namely inequality comparison of real numbers ir, LTL ltl, MTL mtl, and set theory set. If the rule is mentioned but a clear description of the implementation is missing, it is denoted by X. If a rule has not been discussed in the publication, this is denoted by –.

Class	Rule	[46]	[48]	[49]	[52]	[56]	[82]
VEL	below speed limit	ir	ir	–	–	mtl	set
VEL	no stopping	–	ir	–	–	mtl	set
LS	keep in rightmost lane	X	ir	–	–	–	–
LS	keep outside leftmost lane	X	ir	–	–	–	–
OV	safe lane change	X	ir	–	ltl	–	–
OV	no passing on the right side	X	ir	–	–	mtl	–
OV	speed advantage during overtaking	–	–	–	–	–	–
DIST	safe distance	ir	ir	ir	ltl	mtl	set
BOV	being overtaken	–	ir	–	–	–	
PRIO	right of way	X	–	ir	–	–	set
PRIO	zipper merge	–	–	–	–	–	–

Koschi et al. [82] formulate illegal sets, which are occupancies that the traffic rule forbids to enter. However, they only express non-temporal rules such as speed regulations or right of way for vehicles in priority lanes.

To summarize, previous works have adopted inequality constraints or set theory to express rules that can be evaluated based on only a single time instance [46, 48]. However, some rules need to include past time instances, which is why other works identified Temporal Logics as a suitable formal language to specify traffic rules [6, 47, 50, 56]. During this work's legal analysis, conjunction, disjunction, negation, and implication proved to be powerful and useful operators for formalizing rules, which calls for formal logic usage. LTL is a discrete formal logic to reason not just about an absolute truth but about truths that might hold only at some points in time. It can thus be used to represent temporal (non-Markovian) properties. As LTL is the most mature temporal logic in tool support, this work decided to use LTL. However, this work's formalized rules can easily be transferred to other formal logic such as MTL.

3.4. Formalizing Traffic Rules Using Linear Temporal Logic

After the legal analysis of the rules, they will now be formalized in a formal language. Similar to [52], this thesis will distinguish the formalization between codification (representing natural language specifications as logical entities) and concretization (concretely interpreting the symbols).

3.4.1. Linear Temporal Logic for Codification

Let Π be a set of atomic propositions. The powerset of Π, i.e., the set of all subsets of Π, is denoted by 2^{Π}. Formally, the language φ of LTL formulas is defined as

$$\varphi ::= \pi \,|\, \neg\varphi \,|\, \varphi_1 \wedge \varphi_2 \,|\varphi_1 \vee \varphi_2|\varphi_1 \implies \varphi_2 \,|\, \circ\varphi \,|\, \varphi_1 \mathsf{U} \varphi_2 \,|\, \Box\varphi \,|\, \Diamond\varphi, \tag{3.1}$$

where $\pi \in \Pi$ denotes an *atomic* proposition, $\neg$ (resp. $\wedge$, $\vee$, $\implies$) denote the Boolean operators "not", "and", "or" and "implies", and $\circ$ (resp. U, $\Box$, $\Diamond$) denote the temporal operators "next", "until", "globally" (or "always"), "finally" (or "eventually"). See [83] for a definition of the semantics.

3.4.2. Atomic Propositions for Concretization

First, suitable atomic propositions need to be defined. Then, a function to calculate them needs to be provided. While this is trivial for some labels, such as collision or speed limit violation, it is certainly not for others, such as the notion of a safe distance. To prove correctness of the safe distance function, Rizaldi et al. [51] used a theorem prover.

This work defines a labeling function

$$\mathcal{L} : \mathcal{S} \to 2^{\Pi}, \tag{3.2}$$

that maps state $s \in \mathcal{S}$ to a subset of propositions in Π. Table 3.3 shows the atomic propositions used in this thesis. They are evaluated based on the observed scene (from simulation or dataset replay) before passing them to the rule monitor, which will check the rule formula at each time step. The labeling function to calculate them has been added to the BARK framework. The relational position labels are calculated according to Figure 3.8, i.e., in a partially overlapping manner. Maps are currently an important part of automated driving and can provide location information (built-up vs. non-built-up) and road types (road, highway) [55]. Special lane types such as diverging or accelerating lanes are thus assumed to be available from a map.

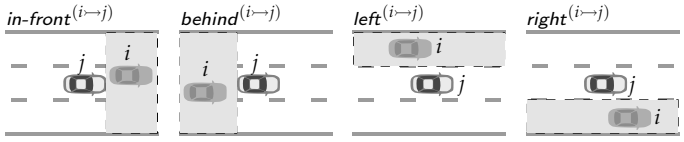


Figure 3.8.: Relational position labels, which describe that vehicle i located in the gray area is in *in-front* / *behind* / *left* / *right* to vehicle j (graphic from [59], ©2020 IEEE).

Table 3.3.: Atomic propositions and their respective interpretations from the perspective of vehicle i in relation to another vehicle j.

$\pi \in \Pi$	Interpretation
$\mathit{dense}^{(i)}$	i is closer than R_{dense} to N_{dense} or more vehicles
$\mathit{succ}^{(i \rightarrowtail j)}$	i is the successor of j
$\mathit{right}^{(i \rightarrowtail j)}$	i is to the right of j
$\mathit{left}^{(i \rightarrowtail j)}$	i is to the left of j
$\mathit{in\text{-}front}^{(i \rightarrowtail j)}$	i is in the front of j
$\mathit{behind}^{(i \rightarrowtail j)}$	i is behind of j
$\mathit{merged}^{(i)}$	i has passed a static merging point, from which on a merge is not possible anymore
$\mathit{sd\text{-}front}^{(i)}$	i has a safe distance to the preceding vehicle
$\mathit{sd\text{-}rear}^{(i)}$	i has a safe distance to the following vehicle
$\mathit{collide}^{(i)}$	i is colliding with road boundaries or any other vehicle or obstacle
$\mathit{lane\text{-}change}^{(i)}$	i is crossing a lane boundary
$\mathit{near}^{(i \rightarrowtail j)}$	i is closer than d_{near} to j
$\mathit{near\text{-}lane\text{-}end}^{(i)}$	i has less than s_{rem} remaining to the end of the lane
$\mathit{acc}^{(i)}$	i accelerates with $a > a_{\text{lim}}$
$\mathit{below\text{-}speed}(v_{\text{lim}})^{(i)}$	i drives at a velocity below v_{lim}
$\mathit{speed\text{-}adv}^{(i \rightarrowtail j)}$	i is faster than j and some threshold v_{diff}
$\mathit{built\text{-}up}^{(i)}$	i is within a built-up area
$\mathit{motorway}^{(i)}$	i is on a road type: motorway
$\mathit{div\text{-}lane}^{(i)}$	i is on a lane type: diverging lane
$\mathit{acc\text{-}lane}^{(i)}$	i is on a lane type: acceleration lane
$\mathit{rightmost\text{-}lane}^{(i)}$	i is in the rightmost lane
$\mathit{leftmost\text{-}lane}^{(i)}$	i is in the leftmost lane
$\mathit{on\text{-}road}^{(i)}$	i is on the road
$\mathit{num\text{-}lanes\text{-}ge}(n)^{(i)}$	i is on a road with n or more parallel lanes in the same direction

3.4.3. Codified Rules

The rules will be formalized as close as possible to the legal analysis. Humans might have relaxations of those rules in mind, i.e., under which circumstances they would not follow a specific rule. However, this should not be included in the codified rules as this would yield coding the desired behavior for a particular scenario into a formalized rule. Table 3.4 shows the formalized rules, which will be discussed in the following. The codification of right of way (see Section 3.2.6) is left to future work. The rules are written from the viewpoint of a vehicle i.

Velocity The rule φ_{vel_1} to stay below a maximum speed limit is trivial: The always operator $\square$ requires the value of $\mathit{below\text{-}speed}(v_{\max})^{(i)}$ to hold true at all times. The rule to not stop requires to not stay below a speed limit v_{stop}. However, this shall only hold if the predecessor has not stopped.

Lane Selection The formalized lane selection rules φ_{ls_1} and φ_{ls_2} follow directly from the legal analysis in Figure 3.3 and the usage of De Morgan's laws. If the respective premises hold true, the vehicle must be in the rightmost lane (φ_{ls_1}) and not in the leftmost lane (φ_{ls_2}). The always operator $\square$ ensures that the rules must hold true at all times.

Table 3.4.: Formalized rules in LTL.

Class	Rule	Formula φ
VEL	below speed limit	$\varphi_{\text{vel}_1} = \square\ \textit{below-speed}(v_{\max})^{(i)}$
VEL	no stopping	$\varphi_{\text{vel}_2} = \square\ \Big(\neg\big(\textit{succ}^{(i\rightarrowtail j)} \wedge \textit{below-speed}(v_{\text{stop}})^{(j)}\big) \implies \neg\textit{below-speed}(v_{\text{stop}})^{(i)}\Big)$
LS	keep in right-most lane	$\varphi_{\text{ls}_1} = \square\Big(\neg\textit{dense}^{(i)} \wedge (\neg\textit{built-up}^{(i)} \vee \textit{motorway}^{(i)}) \wedge (\textit{built-up}^{(i)} \vee \neg\textit{num-lanes-ge}(3)) \implies \textit{rightmost-lane}^{(i)}\Big)$
LS	keep outside left-most lane	$\varphi_{\text{ls}_2} = \square\Big(\neg\textit{built-up}^{(i)} \wedge \textit{num-lanes-ge}(3) \implies \neg\textit{leftmost-lane}^{(i)}\Big)$
OV	no passing on the right side	$\varphi_{\text{ov}_1} = \square\Big(\neg\textit{div-lane}^{(i)} \wedge \neg\textit{acc-lane}^{(i)} \wedge \neg\textit{dense}^{(i)} \wedge (\neg\textit{built-up}^{(i)} \vee \textit{motorway}^{(i)}) \implies \neg(\textit{behind}^{(i\rightarrowtail j)} \wedge \circ(\textit{behind}^{(i\rightarrowtail j)} \mathsf{U} \textit{right}^{(i\rightarrowtail j)} \mathsf{U} \textit{in-front}^{(i\rightarrowtail j)}))\Big)$
OV	safe lane change	$\varphi_{\text{ov}_2} = \square\Big(\textit{lane-change}^{(i)} \implies \textit{sd-rear}^{(i)}\Big)$
OV	speed advantage for overtaking	$\varphi_{\text{ov}_3} = \square\Big(\textit{behind}^{(i\rightarrowtail j)} \wedge \circ(\textit{behind}^{(i\rightarrowtail j)} \mathsf{U} \textit{left}^{(i\rightarrowtail j)} \mathsf{U} \textit{in-front}^{(i\rightarrowtail j)}) \implies (\textit{near}^{(i\rightarrowtail j)} \implies \textit{speed-adv}^{(i\rightarrowtail j)})\Big)$
DIST	safe distance	$\varphi_{\text{sd}} = \square\ \textit{sd-front}^{(i)}$
BOV	being overtaken	$\varphi_{\text{bov}} = \square\Big(\textit{right}^{(i\rightarrowtail j)} \wedge \textit{near}^{(i\rightarrowtail j)} \implies \neg\textit{acc}^{(i)}\Big)$
PRIO	zipper merge	$\varphi_{\text{zip}} = \Big((\neg\textit{rightmost-lane}^{(i)} \wedge \neg\textit{rightmost-lane}^{(j)})\mathsf{U}(\textit{left}^{(i\rightarrowtail k)} \wedge \neg\textit{in-front}^{(i\rightarrowtail k)} \wedge \textit{near}^{(i\rightarrowtail k)} \wedge \textit{near-lane-end}^{(k)} \wedge \textit{succ}^{(i\rightarrowtail j)} \wedge \neg\textit{merged}^{(i)})\Big) \implies \square\Big(\textit{merged}^{(i)} \wedge \textit{on-road}^{(k)} \implies \neg\textit{succ}^{(i\rightarrowtail j)} \vee \textit{behind}^{(i\rightarrowtail k)}\Big)$

Overtaking Based on the previous legal analysis, three rules are formalized for overtaking (see Table 3.4). To forbid passing on the right side, φ_{ov_1} uses the temporal sequence of relational position behind – right – front:

$$\varphi_{\text{ov}_1} = \square\Big(\overbrace{\neg\textit{div-lane}^{(i)} \wedge \neg\textit{acc-lane}^{(i)} \wedge \neg\textit{dense}^{(i)} \wedge (\neg\textit{built-up}^{(i)} \vee \textit{motorway}^{(i)})}^{\text{premise where rule applies}} \implies \underbrace{\neg\big(\textit{behind}^{(i\rightarrowtail j)} \wedge \circ\big(\textit{behind}^{(i\rightarrowtail j)} \mathsf{U} \textit{right}^{(i\rightarrowtail j)} \mathsf{U} \textit{in-front}^{(i\rightarrowtail j)}\big)\big)}_{\text{forbidden overtaking sequence}}\Big)$$

By formulating $\textit{behind}^{(i\rightarrowtail j)} \wedge \circ(\cdot)$, only complete overtakes are allowed, and partial ones where the vehicles already start side by side are removed. In contrast to the author's previous work [47], the relational labels are defined as partially overlapping, which changes the meaning of the rule from overtaking to passing. The always operator $\square$ ensures that the premise will be checked each time and not stay at false, if, e.g., the traffic is dense for some time and then dissolves.

The safe lane change rule requires a safe distance to the rear vehicle when performing a lane change. Thus, φ_{ov_2} covers the rules in [52], where overtaking is defined as the process of changing lanes and passing, and for which a safe rear distance at the beginning and finish of the overtake has to hold.

The third formula φ_{ov_3} ensures that a vehicle shall only overtake if it has a significant speed advantage over the other vehicle:

$$\varphi_{\text{ov}_3} = \square\Big(\overbrace{\textit{behind}^{(i\rightarrowtail j)} \wedge \bigcirc\big(\textit{behind}^{(i\rightarrowtail j)}\,\mathsf{U}\,\textit{left}^{(i\rightarrowtail j)}\,\mathsf{U}\,\textit{in-front}^{(i\rightarrowtail j)}\big)}^{\text{overtaking sequence}} \Longrightarrow \underbrace{\big(\textit{near}^{(i\rightarrowtail j)} \Longrightarrow \textit{speed-adv}^{(i\rightarrowtail j)}\big)}_{\text{speed advantage check if close}}\Big)$$

If there is no significant speed advantage while they are close, there will be a penalty once the overtaking vehicle has passed the other. Note that for the rule to be violated, the speed advantage only has to be false once during the overtake, and not necessarily at the end of the overtake. The regulations do not provide any exceptions to φ_{ov_3}. It will thus be considered to be valid in built-up areas as well.

Safe Distance The rule to ensure a safe distance of vehicle i to any preceding vehicle in front (denoted $(\cdot)^f$) is formalized using the safe distance definition in [52], which is defined as follows:

$$d^i_{\text{safe}_0} = v^i \cdot t_{\text{react}} - \frac{(v^i)^2}{2 \cdot a^i_{\text{br,max}}} \tag{3.3a}$$

$$d^i_{\text{safe}_1} = v^i \cdot t_{\text{react}} - \frac{(v^i)^2}{2 \cdot a^i_{\text{br,max}}} + \frac{(v^f)^2}{2 \cdot a^f_{\text{br,max}}} \tag{3.3b}$$

$$d^i_{\text{safe}_2} = \frac{\left(v^f + a^f_{\text{br,max}} \cdot t_{\text{react}} - v^i\right)^2}{2 \cdot (a^f_{\text{br,max}} - a^i_{\text{br,max}})} - v^f \cdot t_{\text{react}} - \frac{1}{2} a^f_{\text{br,max}} \cdot t^2_{\text{react}} + v^i \cdot t_{\text{react}} \tag{3.3c}$$

$$d^i_{\text{safe}_3} = v^i \cdot t_{\text{react}} - \frac{(v^i)^2}{2 \cdot a^i_{\text{br,max}}} - v^f \cdot t_{\text{react}} - \frac{1}{2} a^f_{\text{br,max}} \cdot t^2_{\text{react}}, \tag{3.3d}$$

where $a_{\text{br,max}} < 0$ and

$$v^{f*} = \begin{cases} v^f + a^f_{\text{br,max}} \cdot t_{\text{react}} & \text{if } t_{\text{react}} \leq t^i_{\text{stop}} \\ 0 & \text{otherwise.} \end{cases} \tag{3.4}$$

The time durations to brake to standstill are defined as:

$$t^i_{\text{stop}} = -v^i / a^i_{\text{br,max}} \tag{3.5a}$$

$$t^f_{\text{stop}} = -v^f / a^f_{\text{br,max}} \tag{3.5b}$$

$$t^{f*}_{\text{stop}} = -v^{f*} / a^f_{\text{br,max}} \tag{3.5c}$$

The safe distance label then follows as

$$\textit{sd-front}^{(i)} := \begin{cases} \top & \text{if } (d^{(i\rightarrowtail f)} > d^i_{\text{safe}_0}) \vee \big((t_{\text{react}} \leq t^f_{\text{stop}}) \wedge (d > d^i_{\text{safe}_3})\big) \\ d > d^i_{\text{safe}_2} & \text{else if } (t_{\text{react}} \leq t^f_{\text{stop}}) \wedge (a^i_{\text{br,max}} > a^f_{\text{br,max}}) \wedge (v^{f*} < v^i) \wedge (t^i_{\text{stop}} < t^{f*}_{\text{stop}}) \\ d > d^i_{\text{safe}_1} & \text{otherwise,} \end{cases} \tag{3.6}$$

where $d^{(i\rightarrowtail f)}$ denotes the longitudinal distance between the rear end of the front vehicle and the front end of the ego vehicle. The final rule formula φ_{sd} is simple: a safe distance has to hold true at all times.

Being Overtaken When being overtaken, acceleration shall be prohibited. Being overtaken could be defined as a temporal sequence of relational positions. However, with this rule, the proximity check already serves this work's needs, as it is computationally cheaper with less labels needed to be calculated.

Zipper Merge For the zipper merge, the driver on the non-ending lane is supposed to let the driver on the ending lane in. This work defines a zipping situation as a situation where a vehicle i is close and to the left but not in front of a vehicle k, which is in an ending or blocked lane. In addition, vehicle i follows another vehicle j and has not passed the final merging point yet. Figure 3.9 shows the naming conventions for this rule. The inclusion of the label $\mathit{near\text{-}lane\text{-}end}^{(k)}$ ensures that vehicles that merge early, meaning they merge far ahead of the lane ending, will not count into the rule. The rule can be formalized as

$$\varphi_{\text{zip}} = \Big(\varphi_{\text{zip,prem}} \mathsf{U} \big(\overbrace{\mathit{left}^{(i \rightarrowtail k)} \wedge \neg \mathit{in\text{-}front}^{(i \rightarrowtail k)} \wedge \mathit{near}^{(i \rightarrowtail k)} \wedge \mathit{near\text{-}lane\text{-}end}^{(k)} \wedge \mathit{succ}^{(i \rightarrowtail j)} \wedge \neg \mathit{merged}^{(i)}}^{\text{zipping situation}}\big)\Big) \\ \implies \square\Big(\mathit{merged}^{(i)} \wedge \mathit{on\text{-}road}^{(k)} \implies \neg \mathit{succ}^{(i \rightarrowtail j)} \vee \mathit{behind}^{(i \rightarrowtail k)}\Big)$$

with $\varphi_{\text{zip,prem}} = \big(\neg \mathit{rightmost\text{-}lane}^{(i)} \wedge \neg \mathit{rightmost\text{-}lane}^{(j)}\big)$. If a zipping situation has been detected and vehicles i and j have not been in the rightmost lane before, the rule shall guarantee the following: Once vehicle i has merged and vehicle $\mathit{on\text{-}road}^{(k)}$ has not left the road, vehicle i must

- either not follow vehicle j directly (as vehicle k from the other lane has merged in-between and has become the new predecessor of vehicle i, or as vehicle j has left the lane to the left)
- or vehicle i must be behind of vehicle k (as vehicle k has merged before j).

3.5. Conclusion

This chapter formalized traffic rules for dual carriageways according to German traffic regulations. For this, a methodology for a legal analysis was presented, which was used to codify these rules. The rule formalization and especially the description of labels using mathematical functions inevitable yields ambiguities, which was approached by studying and including court rulings. This work shall help to start a discussion to remove these ambiguities. Going forward, the traffic law should be tightened by agreeing to the same formalized rules.

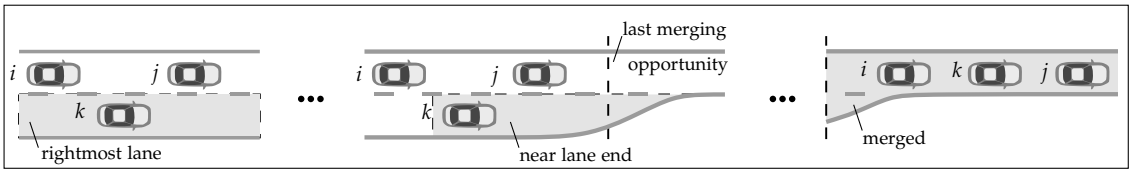


Figure 3.9.: Naming convention of vehicles i, j, k during a zipper merge and the temporal evaluation of a sequence that satisfies $\varphi_{\text{zip,prem}}$.

4 Evaluating Traffic Rules in Linear Temporal Logic on Recorded Drives

4.1. Overview and Contribution

In this chapter, the previously formalized rules are evaluated on recorded traces of human drivers, which can help to identify errors in the formalized rules. Eventually, once the formalization process has been completed, it provides valuable insight into the extent to which humans follow these rules. For this, the theory on Linear Temporal Logic on finite traces (LTL_f) is introduced, followed by the theory on automata-based verification of such formulas. A concept to evaluate a traffic rule in LTL_f using a rule monitor is then presented. The main contributions in this chapter are

- a generic framework for monitoring a set of traffic rules which has been embedded in BARK and
- insights on the extent to which humans follow these rules.

The methodology and parts of the results are based on the author's previously published works in [47] and [59]. The experiments have been extended to include all formalized rules of this work.

4.2. Linear Temporal Logic on Finite Traces

In the context of continuously re-planning over a receding horizon, LTL_f is used since it provides a formalism to reason over bounded periods of time. It uses the same syntax as LTL, which was introduced in Section 3.4.1. The definition of the semantics of LTL_f can be found in [84].

Following the categorization of Manna et al. [85], this work considers formulas with obligation properties, which include safety and guarantee properties. Safety formulas can be represented as $\square \boldsymbol{p}$, whereas guarantees are generally captured by $\lozenge \boldsymbol{p}$, where $\boldsymbol{p}$ is a finite LTL formula consisting of only atomic propositions, the operators $\neg, \vee, \wedge, \implies, \bigcirc$, and past-LTL operators. Note that past-LTL operators are not used in this work.

4.3. Automaton-based Verification of LTL_f Formulas

Automata-based model checking will be used to verify if a trace satisfies the LTL_f formula. Given a formula in LTL_f, a Deterministic Finite Automaton (DFA) can be constructed that recognizes words satisfying the formula. For the LTL_f formula to be translated into a DFA, this work follows the concept of [84]. An additional atomic proposition *alive* is introduced to the formula before translation to the automaton. This symbol will be set to true initially and set to false once the end of the trace has been reached. The automaton is defined over the alphabet $\Sigma = 2^\Pi$, i.e., the powerset of atomic propositions of the LTL_f formula.

Definition 1 (DFA). *A DFA is a tuple $\Lambda = (\mathcal{Q}, q_0, \Sigma, \delta, F)$, where*

- *$\mathcal{Q}$ is a set of states*
- *$q_0 \in \mathcal{Q}$ is the initial state*
- *Σ is a finite alphabet*
- *$\delta : \mathcal{Q} \times \Sigma \to \mathcal{Q}$ is a transition function*
- *$F \subseteq \mathcal{Q}$ denotes a set of accepting (or final) states.*

A label function $\mathcal{L}$ yields a symbol $\sigma_k \in \Sigma$ at a given time instance k. For a word w, given as a sequence of symbols $\sigma_k\ \sigma_{k+1} \ldots$, and a DFA Λ, the automaton state is sequentially updated from the initial automaton state with respect to δ and w. Therefore, if the automaton halts in an accepting state, w satisfies the formula.

4.4. Runtime Monitoring of Traffic Rules

4.4.1. Rule Violations

Previous work [6, 86] introduced a weighted transition for rule violation to the automaton. During planning, each rule automaton is evaluated over the word w, and the weight equals the penalty for the respective rule. Specifically, a self-looping automaton is used in [6], i.e., a weighted transition to the same state is added for a violating transition. However, this formulation does not allow the modeling of violations that cause a penalty once vs. violations that accumulate penalties over time. To account for this, Schluter et al. [86] extended [6] by defining an explicit violation symbol.

Although violations of safety properties can be detected on partial traces, a safety property $\square \boldsymbol{p}$ will never be satisfied again after being violated. Consequently, violating a safety property leads to a non-accepting state in the automaton that only has a self-loop. Thus, the automaton cannot be brought back to an accepting state once it has registered a violation. To obtain penalties for consecutive rule violations, such as multiple times overtaking on the right [47], the automaton

is reset to its initial state after a violation occurred. Formally, $\Lambda = (\mathcal{Q}, q_0, \Sigma, \delta, F)$ is modified to $\overline{\Lambda} = (\mathcal{Q}, q_0, \Sigma, \overline{\delta}, F)$, where

$$\overline{\delta}(q,\sigma) := \begin{cases} q_0 & \text{if } \forall \sigma' \in \Sigma : \delta(\delta(q,\sigma),\sigma') = \delta(q,\sigma) \wedge \delta(q,\sigma) \notin F \\ q' \in F & \text{if } q = q_0 \wedge \mathit{alive} \notin \sigma \\ \delta(q,\sigma) & \text{otherwise.} \end{cases} \tag{4.1}$$

4.4.2. Open Rule Monitor

Implementation-wise, Spot [87], a C++ library for model checking, is used to translate the formalized LTL formula to a deterministic finite automaton, and to manipulate the automaton. Each formula is captured in a `RuleMonitor`, which has been published as open-source [88] as part of this work. To monitor rule compliance throughout the simulation, the `RuleMonitor` is encapsulated in an `EvaluatorLTL` object, which implements BARK's abstract evaluator interface (see Section 2.1.4). The violation count is incremented if the modified automaton $\overline{\Lambda}_\varphi$ gets reset to its initial state (violation of a safety property $\square \boldsymbol{p}$) or the automaton does not halt in an accepting state (violation of a guarantee property $\Diamond \boldsymbol{p}$).

4.5. Evaluation

Evaluating the formalized rules on recorded drives of humans will help to validate the formalized rules and also provide valuable insight into the extent to which humans follow the rules. The INTERACTION dataset [89], which focuses on dense interactions, is used to analyze the compliance of each vehicle to the traffic rules. To the best of the author's knowledge, [51, 56] form the only works that evaluated their respective formalized traffic rules on recorded drivers.

4.5.1. Evaluation Methods and Dataset Processing

The approach is studied in the benchmarking framework BARK, which was introduced in Section 2.1. The rule monitors from Section 4.4.2 are used to monitor rule compliance throughout the simulation, effectively replaying the dataset.

Behavior Model for Dataset Tracking To get real-world data into the simulation, a BARK behavior model was implemented that tracks recorded trajectories. The simulation can be started at an arbitrary timestamp and include all or only a subset of recorded vehicles.

Recorded Drives The two-lane German merging scenario DR_DEU_Merging_MT and the three-lane road lower part of the Chinese highway merging scenario DR_CHN_Merging_ZS will be analyzed. The two scenarios are summarized in Table 4.1. The formalized rules from the German traffic code are evaluated on DR_CHN_Merging_ZS, although differences between the two countries' local rulesets can be expected. To understand the dataset first, Figure 4.1 shows the distributions of the mean velocity, the vehicle's mean gap distance to the preceding

Table 4.1.: Analyzed scenarios from the INTERACTION dataset.

Scenario Name	DR_DEU_Merging_MT	DR_CHN_Merging_ZS (lower)
Country	Germany	China
Location	Built-up	Built-up
Carriageway	Dual	Dual
Road Type	Road	Motorway
Max. Speed Limit	$50\,\mathrm{km/h} \approx 14\,\mathrm{m/s}$	$80\,\mathrm{km/h} \approx 22\,\mathrm{m/s}$
Number of vehicles	483	3178
Lanes	$2 \rightarrow 1$	$3 \rightarrow 2$

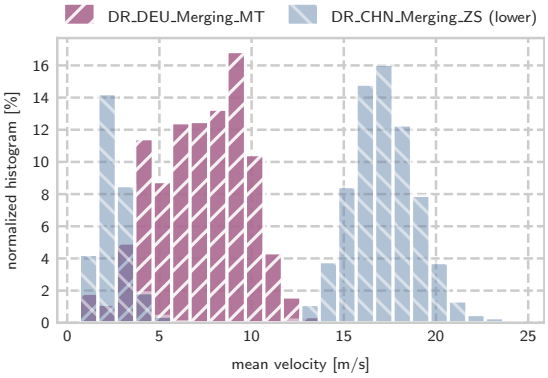
Table 4.2.: Validity of the rules in the analyzed scenarios from Table 4.1.

Class	Rule	DR_DEU_Merging_MT	DR_CHN_Merging_ZS (lower)
VEL	below speed limit	yes	yes
VEL	no stopping	yes	yes
LS	keep in right-most lane	no (built-up and not motorway)	yes
LS	keep outside left-most lane	no (built-up)	no (built-up)
OV	no passing on the right side	no (built-up and not motorway)	yes
OV	safe lane change	yes	yes
OV	speed advantage for overtaking	yes	yes
DIST	safe distance	yes	yes
BOV	being overtaken	yes	yes
PRIO	zipper merge	yes	yes

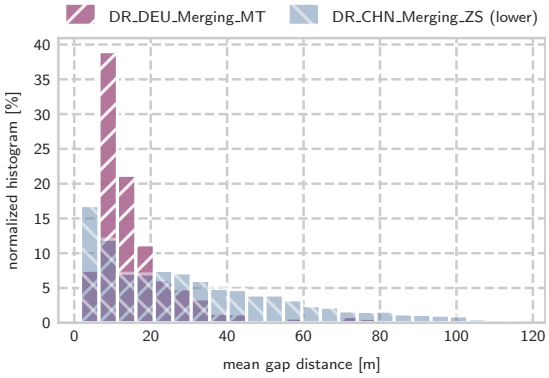
vehicle, and the number of vehicles present at the same time. These distributions are valuable when interpreting the rule violation data and selecting reasonable model parameters when re-simulating the scenarios in closed loop. The scenario DR_DEU_Merging_MT shows one homogeneous driving class that represents urban traffic. Up to 20 cars drive in the scenario at the same time at $4\,\mathrm{m/s}$ to $12\,\mathrm{m/s}$. The gap between vehicles, which can be observed most from the data, is about 10 m. The scenario DR_CHN_Merging_ZS (lower), on the other hand, consists of two major driving classes: A slow and dense part, which is presumably a traffic jam, and a higher speed part where most vehicles are driving at $15\,\mathrm{m/s}$ to $20\,\mathrm{m/s}$. The recorded road section is larger for DR_CHN_Merging_ZS (lower), which together with more lanes explains the higher number of vehicles being present simultaneously in contrast to DR_DEU_Merging_MT.

4.5.2. Evaluation of Violation on Recorded Drives

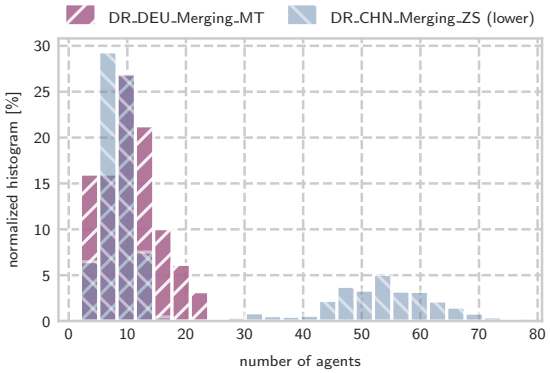
For the evaluation, the rule evaluator's parameters are set according to Table A.1. Different values of the threshold d_{near} for the "zipper merge" rule and the "being overtaken" rule are being used for the two locations. The threshold s_{rem} is location-dependent, as it is tightly coupled with the lane geometry. In comparison to the Chinese map, the ending lane in the German map gets narrow much earlier. The "safe distance" label is parametrized with the values from [51]. Only passenger vehicles with a maximum length of 5 m are included, which yields small differences in comparison to the results published in [59].



(a) Normalized histogram of the mean velocity.



(b) Normalized histogram of the mean gap distance to the preceding vehicle.



(c) Normalized histogram of the number of vehicles.

Figure 4.1.: Distributions of mean velocity, mean gap distance and the number of vehicles from the dataset.

To illustrate one of the more complex rules, Figure 4.2 and 4.3 show two examples for rule violations of the zipper merge rule. For simplicity, only the truth values of the labels for the relevant vehicles are shown. A crossed hatch indicates a violation. In Figure 4.2, vehicles 7 and 8 are initially side by side. However, vehicle 8 drives faster and passes vehicle 7 instead of leaving the gap to let vehicle 7 merge in a zipper fashion. Eventually, once vehicle 8 has merged, the rule evaluator returns false, as its predecessor is still vehicle 6. In Figure 4.3, vehicle 268 should let vehicle 271 in but does not leave a gap. After nudging in the other lane first, vehicle 271 ultimately slows down and lets vehicle 268 pass. Once vehicle 268 has merged, the rule evaluator returns false, as its predecessor is still vehicle 263.

These two examples already show the value of testing the rules on real data: There invariably exist edge cases, such as vehicle 267 leaving the drivable area by crossing solid lane markings and overtaking on the right, that one did not think of when formalizing the rules. Still, the rules need to be formulated soundly despite such irrational behavior. Another challenge of writing sound temporal rules is that they need to work despite the continuous emerging of new vehicles on the map, which is of course similar to vehicles emerging within the field of view of an autonomous vehicle.

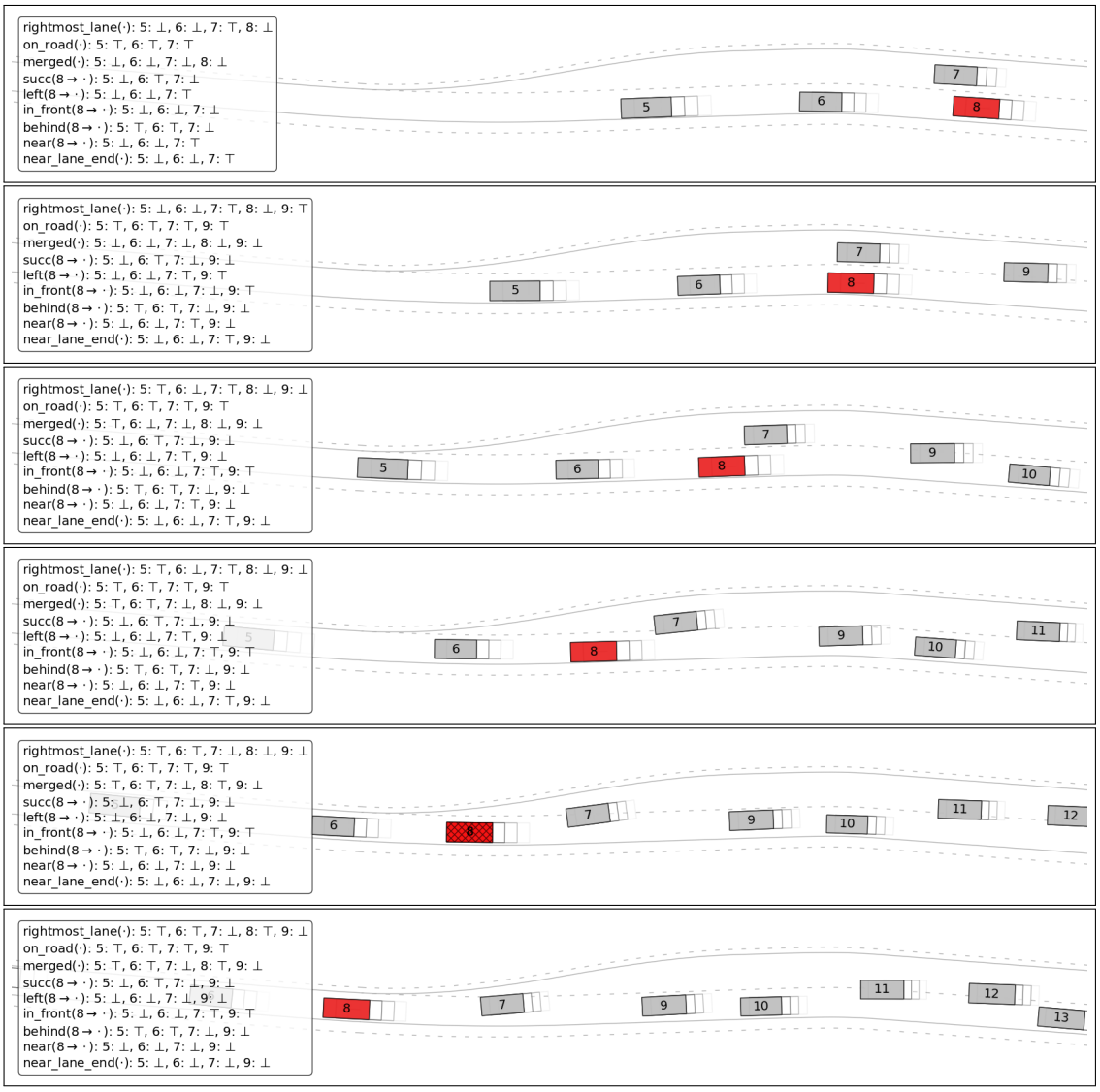


Figure 4.2.: Zipper merge violation in DR_DEU_Merging_MT by vehicle 8 of trackfile 9. The snapshots are depicted at a 1 s time increment. The valuations of the atomic propositions are shown on the left. While vehicles 7 and 8 are initially side by side, vehicle 8 drives faster and passes vehicle 7 instead of leaving the gap to let vehicle 7 merge in a zipper fashion. Eventually, once vehicle 8 has merged, the rule evaluator returns false, as its predecessor is still vehicle 6. The violation is illustrated by a crossed hatch.

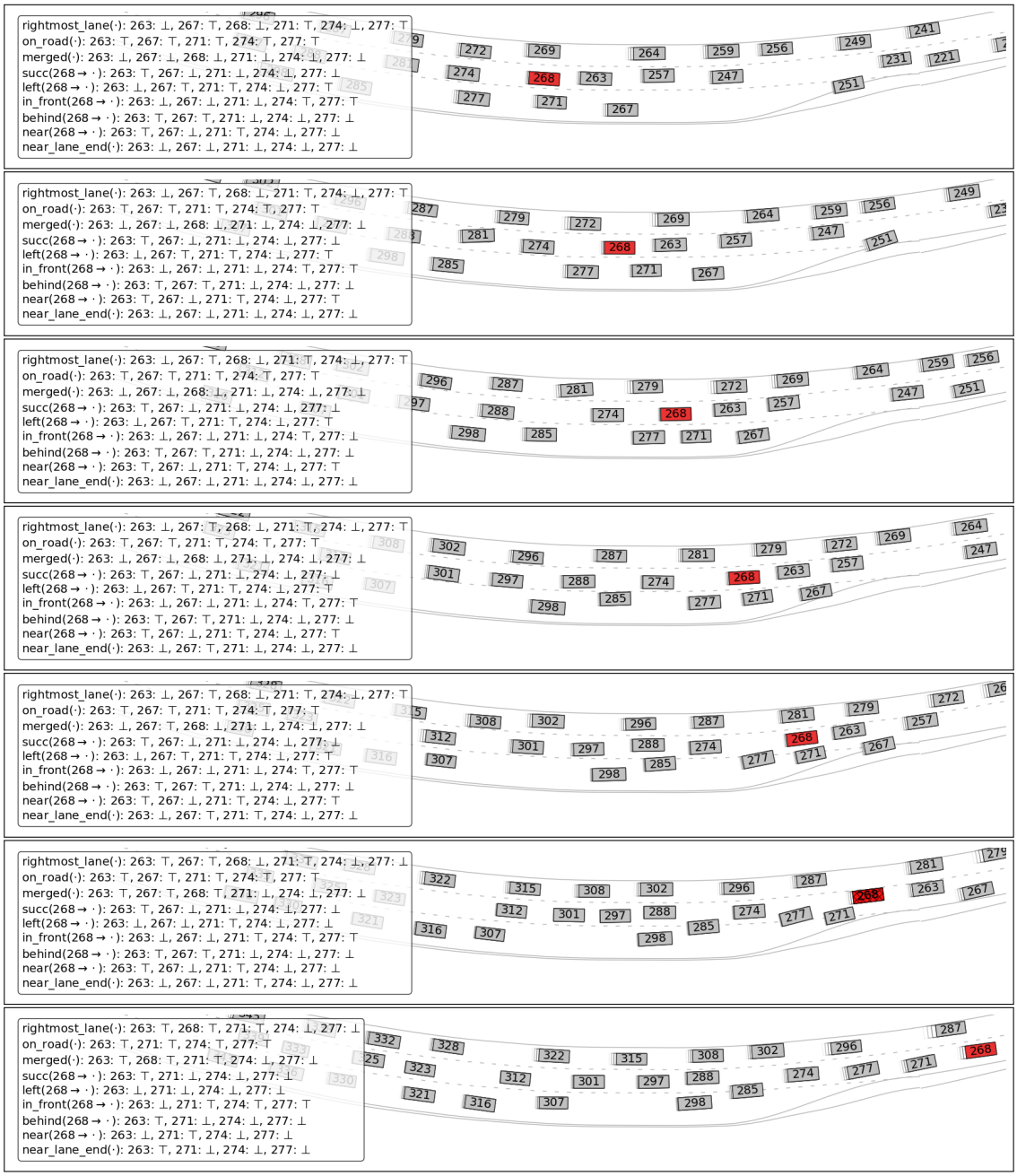


Figure 4.3.: Zipper merge violation in DR_CHN_Merging_ZS (lower) by vehicle 268 of trackfile 8. The snapshots are depicted at a 5 s time increment. The valuations of the atomic propositions are shown on the left. Vehicle 268 should let vehicle 271 in, but does not leave a gap. After nudging into the other lane first, vehicle 271 eventually slows down and lets vehicle 268 pass. Once vehicle 268 has merged, the rule evaluator returns false, as its predecessor is still vehicle 263. The violation is illustrated by a crossed hatch.

For the quantitative analysis, the percentage of rule violation per vehicle is calculated as

$$\text{rule violation per vehicle} := \frac{\sum_{i \in \mathbf{V}} \begin{cases} 1, & \text{if } \varphi = \bot \text{ at least once} \\ 0, & \text{otherwise} \end{cases}}{|\mathbf{V}|}, \tag{4.2}$$

where $\mathbf{V}$ denotes the set of all vehicles in the dataset. Table 4.2 displays the validity of the rules within the two scenarios based on the map-based features from Table 4.1.

Figure 4.4 shows the results of the quantitative evaluation. Violations of close to 100% would probably indicate an error in the formalization. The evaluation is performed for step sizes of 0.1 s, 0.3 s, and 0.5 s. The results do not show any significant discrepancies for various step sizes, which enables their use in planning modules, that might have a planning step size above 0.1 s.

In the Chinese scenario, about 85% of the vehicles do not always keep in the rightmost lane. If the number of violations is that high, three reasons are possible: First, China's local law might be different in that it does not include this rule in this scenario. Second, the local interpretation of dense traffic might be different or, third, the rule's priority to human drivers might be rather low.

About 10% of the vehicles in the German scenario and 6% of the cars in the Chinese scenario do not perform a safe lane change. This shows that human drivers either underestimate the safe distance when changing lanes or accept some level of risk to drive fluently within the bulk. Similar things can be said for the safe distance, where 27 to 32% of the vehicles violate the safe distance at least once.

The rule to have a "speed advantage for overtaking" is being violated in the Chinese scenario by more than 13% of the vehicles, whereas no violations can be observed in the German scenario. This could either stem from the distinct regulations, different local interpretation of what significant speed advantage means, or is due to the differences of the traffic situations (the Chinese scenario's road section is more extensive than the German scenario's, making a completed overtaking more likely). About 1% of the vehicles in the German scenario violate the zipper merge rule. In the Chinese scenario, less than 1% do not merge according to this rule. Such a small number indicates that most human drivers follow this rule very well.

The analysis reveals that human drivers violate the formalized rules. However, violation of a single rule should not be used directly to assign blame. Assuming that rule violations happen out of the necessity of the situation and not on purpose, one would be tempted to include all those situational exceptions into the formalized rule. Unfortunately, this would effectively mean codifying the vehicle's desired behavior for each scenario, which is not desirable due to the sheer number of possible scenarios. Instead, if the ruleset is complete, these exceptions will arrive implicitly by introducing weights or priorities between the rules and evaluating the full ruleset instead of a single rule. It might be required to re-simulate or re-plan a short section before a violation to study whether other alternatives were available.

Correlations This paragraph aims to identify correlations between the rule violations and the traffic situation. The correlation study can help gain insight into the rules and identify definitions gaps if, e.g., correlations are identified that seem implausible. Specifically, the distributions used

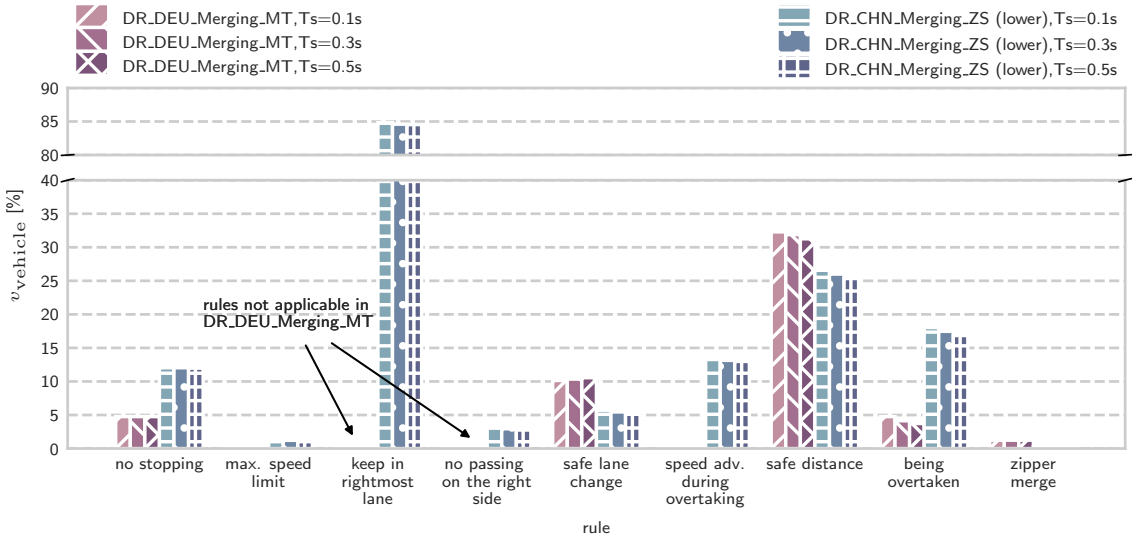


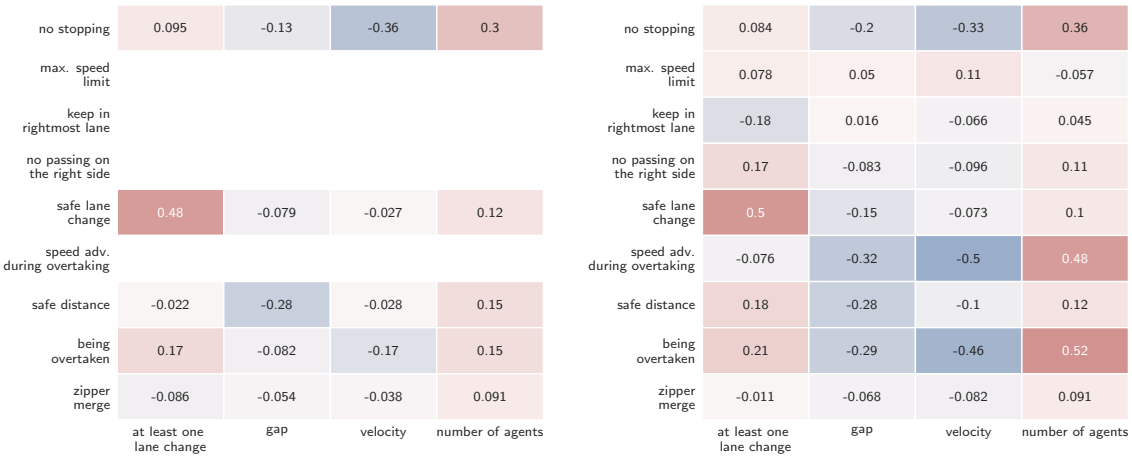
Figure 4.4.: Rule violations in the dataset for two different locations and three different evaluation time step sizes.

to characterize the traffic situation are the gap distance to the preceding vehicle, the vehicle's average velocity, whether the vehicle changed lanes, and the number of vehicles being present simultaneously. The Pearson correlation coefficient is employed, where a correlation factor of $+1$ indicates a strong positive correlation, a correlation factor of 0 indicates no correlation, and a correlation factor of -1 indicates a strong negative correlation. Figure 4.5 shows the results. As the Chinese dataset is more diverse in terms of mean velocity and the number of vehicles present simultaneously, the results from DR_CHN_Merging_ZS (lower) might yield more convincing correlations. For those rules that were not applicable or where not enough violations were detected, no correlation coefficient can be computed, and the field is left blank.

The "no stopping" rule is sometimes violated when the traffic jam has started to resolve, such that the vehicle in front has started to drive again, but the rear vehicle is slow to start as well. Thus not surprisingly, a correlation of this rule's violations can be observed with parameters describing a traffic jam (slow velocity, many vehicles). The "safe lane change" rule correlates with vehicles doing at least one lane change for both scenarios DR_DEU_Merging_MT and DR_CHN_Merging_ZS (lower), which is an obvious link. The gap distance, which is one indicator for traffic density, negatively correlates with the safe distance. Subsequently, the smaller the gap distance, the more often the safe distance is violated. However, the correlation is only slight, as the velocity also plays into the safe distance formula.

The slower the vehicles are driving, the more often they accelerate when "being overtaken". This might stem from the fact that slow vehicles are being overtaken more often. Also, the slower the cars are driving, the more often they do not have a significant "speed advantage during overtaking". These two rules also correlate with the number of simultaneous vehicles.

For other rules such as "zipper merge", "maximum speed limit", or "lane selection", no significant correlations can be observed between the violations and the dataset's properties. Essentially, the evaluation did not prompt any implausibilities for the formalized rules.



(a) Correlations for DR_DEU_Merging_MT dataset.

(b) Correlations for DR_CHN_Merging_ZS (lower) dataset.

Figure 4.5.: Correlations between observed rule violations and traffic characteristics such as the gap distance to the preceding vehicle, the average velocity, whether or not the vehicle changed lanes, and the number of vehicles.

Lane Matching The evaluation of real data in lane-based functions is difficult, as the matching between the data and the map might be inaccurate. Thus the calculations that depend on an accurate matching to lanes might introduce errors to the results. The labels for the safe distance *sd-front*, *sd-rear* and for the succeeding relation *succ* rely on an accurate lane matching

$$\texttt{inLane(lane, agent)} = \begin{cases} \top, & \text{if } \mathcal{P}_{\text{lane}} \cap \mathcal{P}_{\text{agent}} \neq \emptyset \wedge |d_\perp| < w_{\text{lane}} \cdot \mathcal{F}_{\text{inLane}} \\ \bot, & \text{otherwise,} \end{cases} \tag{4.3}$$

where $\mathcal{P}_{\text{lane}}$, $\mathcal{P}_{\text{agent}}$ denote the respective polygons, $d_\perp$ the perpendicular distance of the lane's centerline to the ego agent's rear axle point, w_{lane} the width of the lane, and $\mathcal{F}_{\text{inLane}}$ a parameter to control the accuracy of the lane matching. Figure 4.6 illustrates the lane matching for two other vehicles. At $\mathcal{F}_{\text{inLane}} = 0.0$, no lateral offset of the vehicle to the centerline is allowed for it to be viewed as being in the lane. Contrary at $\mathcal{F}_{\text{inLane}} = 2.0$, all vehicles whose shape intersects with the lane polygon will be matched to that lane, as the maximum allowed lateral offset is twice the lane width. Note that the lane matching definition is not exclusive, which means that a vehicle might be inside two lanes.

Figure 4.7 displays the evaluation of the "safe lane change" rule, the "safe distance" rule, and the "zipper merge" rule for different lane matching parameters $\mathcal{F}_{\text{inLane}}$. The evaluation reveals

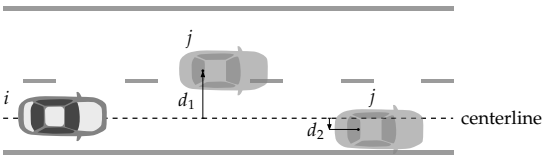


Figure 4.6.: Lane matching calculation for the labels *sd-front*$^{(i)}$, *sd-rear*$^{(i)}$, *succ*$^{(i \rightarrowtail j)}$ based on the lateral distance to the centerline of the ego vehicle's lane.

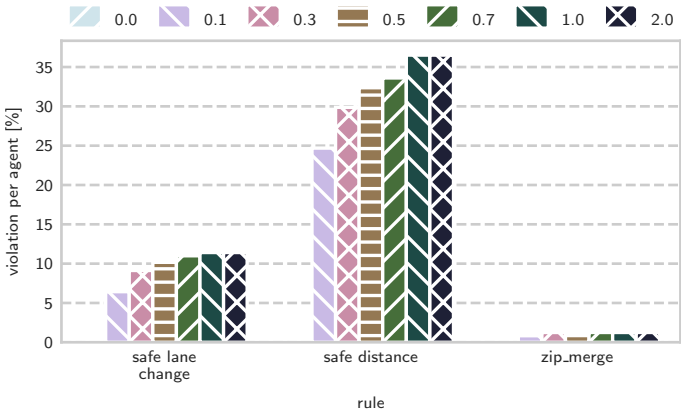


Figure 4.7.: Violations per time of the safe lane change and the "safe distance" rule when varying the lane matching parameter of the preceding vehicle calculation on DR_DEU_Merging_MT.

how sensitive the labels and subsequent rule evaluations are to inaccurate lane matching and that the quantitative results inherently hold some uncertainty. $\mathcal{F}_{\text{inLane}} = 0.5$ was used for the experiments above, which means that the vehicle must be with at least one half within the lane. Note that correct lane matching for rule evaluations is even more critical when evaluating the rules in a car online from perceived and noisy object lists.

4.6. Conclusion

The evaluation of real data helped to concretize the calculation of the labels iteratively. However, some of the parameters should be defined in a collaborative initiative of rule makers and engineers. The evaluation showed that humans violate formal traffic rules to varying extents. Still, the violation of a single rule does not necessarily imply a driver's fault. The driver may have violated the rule in order to satisfy another. Codifying such situational exceptions into the rules would be cumbersome and error-prone due to the sheer number of possible scenarios. Instead, these exceptions will arrive implicitly with a complete ruleset by introducing weights or priorities between the rules and evaluating the entire ruleset instead of a single rule. As a next step, false negatives and false positives need to be identified by extending the evaluation to other scenarios. To automate this step, the datasets' rule violations should be labeled and made public. This would ease the collaboration with legal experts to verify the legal analysis and extend it to cover all regulations. Also, the rules should be evaluated and tested on locally recorded data from a car, which is different in terms of the perception horizon than the statically recorded traffic data from the INTERACTION dataset that was used in this chapter.

This work lays the foundation for integrating traffic rules into a planning component and leveraging formalization benefits to evaluate the rules' compliance.

5 Monitoring of Traffic Rules Within Interactive Behavior Planning

5.1. Overview and Contribution

Traffic rules often depend on the actions of other agents or the past. Merging scenarios have proven challenging due to the dense interaction with others, combined with the eventual lane ending, prompting the need for the driver to make a decision. Game-theoretic approaches offer an elegant way to model such interactions. This chapter proposes a game-based planning approach that monitors traffic rules depending on multiple agents and past information at planning time. To solve this formulation, Monte Carlo Tree Search (MCTS) is used. Evaluating time-dependent traffic rules violates the Markov property, i.e., it does not rely only on the current state but also on previous states. This raises the question how to integrate those rules in MCTS, which relies on the Markov property to be computationally efficient. The main contributions of this chapter are

- a method to monitor multi-agent and time-dependent traffic rules within MCTS,
- a framework to evaluate the effect of modeling a specific traffic rule within a set of rules by implementing a thresholded lexicographical ordering within MCTS, and by using the exact same rule monitor to plan and evaluate a scenario run and
- a study of the ramifications on collision, progress, and the violation of rules in the simulated scenarios when modeling the applicable traffic rules within MCTS.

This chapter is based on the author's previously published work [60]. The results have been updated to more realistic model parameters for the other traffic participants, including a comparative study to real traffic data.

5.2. Related Work

Multiple goals, such as collision and safety metrics, various traffic rules, and comfort metrics, should be considered to drive safely. Some of these goals are favored over others. Expressing the preference using a weighted sum scalarization is tedious, sometimes even impossible. Therefore, Castro et al. [6] realize the preference relation through a Lexicographical Ordering (LO). Compared to a weighted sum scalarization, their method does not require adapting the weights of all the cost terms when adding or modifying goals.

Previous work extended RRT* to follow traffic rules [6]. However, the approach does not consider interactions. The problem has been extended to the coordination of multiple vehicles [90], but assumes explicit communication. Chaudhari et al. [41] define a two-player non-zero-sum non-cooperative game. Both the ego agent and the environment (the other agents) try not to violate any traffic rules expressed in LTL. Each agent builds up a tree as in [6]. The separated planning trees prohibit the approach from incorporating rules that depend on other agents than the ego agent, such as the safe distance or zipper merge rule.

To account for the road rules in interactive planning, Karlsson et al. [44] employ a decoupled approach of interaction-aware joint prediction with sampling-based motion planning. However, the discrete joint prediction does not consider the road rules, which will create discrepancies in dense, complex scenarios between the predicted ego-motion (not rule-compliant) and the planned motion of the rules-aware motion planner.

LTL-based runtime verification on full traces of MCTS has been proposed [91], i.e., the evaluation of the LTL formula always starts at the initial node. The authors employ rules that only depend on the ego agent and demonstrate the approach for an intersection. However, their approach does not allow for incorporating multi-agent rules, nor does it make efficient use of the search tree to evaluate the rules.

To summarize, no work exists that allows studying multi-agent traffic rules in dense scenarios by modeling it as part of the interactive planning problem.

5.3. Problem Formulation and Assumptions

In a lane-merging scenario populated with multiple agents, this work aims to plan the behavior of a single agent $B^i(k) = (x^i_{\text{des}}(k), x^i_{\text{des}}(k+1), \ldots, x^i_{\text{des}}(k+K))$, i.e. the generation of a sequence of desired future dynamic states, while obeying the traffic rules. This work assumes perfect observation of the dynamic state of the other agents and the map, as well as perfect localization within that. Thus, the 'des' subscript is dropped in the following. This setting of interactive behavior planning is treated as a dynamic game, that formally consists of:

- A set of agents $\mathscr{A}$, each agent i having a dynamic state $x_i \in X^i$,
- An environment state $\boldsymbol{s} \in \mathcal{S} = \times X^i$ with state space $\mathcal{S}$,
- Agent 1 (referred to as "ego agent") having an action space A^1 with discrete actions,

- Agents 2...N_A (referred to as "other agents") following a behavior model $\mathcal{B}^o$, that determines the agent's action $a_i^t \sim \mathcal{B}^o(\boldsymbol{s}^t)$.

Based on the joint action of all agents, the environment state $\boldsymbol{s}$ transitions to $\boldsymbol{s}'$. The ego agent gets a reward $r(\boldsymbol{s}, \boldsymbol{s}', \boldsymbol{a})$ after joint action $\boldsymbol{a} \in \mathcal{A} = \times A^i$ is applied. The goal is to find an optimal action a for the ego vehicle that maximizes its cumulative reward in Equation (2.2) along a planning horizon of K steps, where the reward incorporates penalties for discomfort, traffic rule violation, and collision. MCTS is used to obtain an approximation of the optimal solution using sampling since obtaining an optimal solution of the decision problem is infeasible due to the size of the state space.

The formalized traffic rules from Chapter 3 shall be employed, which are formulated in $\mathrm{LTL_f}$. Some of these traffic rules such as the zipper merge or overtaking do not rely only on the current state $\boldsymbol{s}$ but also on previous information. To be used in an interactive planning framework as described above, this work aims to find a Markovian formulation for evaluating the history-dependent rules for each agent.

5.4. Approach

This section presents this work's method for monitoring traffic rules within interactive behavior planning based on MCTS. It is described how to obtain a reward for the rule violation, and how to extend MCTS to rewards in a lexicographical ordering.

5.4.1. Rule Monitoring Within Monte Carlo Tree Search

To allow for a valuation of the traffic rules given in $\mathrm{LTL_f}$, the environment state $\boldsymbol{s}$ is labeled to the atomic propositions described in Section 3.4. To monitor temporal rules within MCTS, the straightforward way is to evaluate the run from the root node for each expansion and simulation step. However, this will significantly limit the performance of the MCTS. Instead, this work exploits the structure of the decision tree by augmenting the tree nodes of the MCTS with the automaton state. Thus, non-Markovian properties captured in LTL can be efficiently verified during the MCTS by encoding necessary historical information into the automaton's state.

At first, each φ_i formula is translated into their corresponding DFA representation $\overline{\Lambda}_i$, which was introduced in Section 4.4. Instead of combining them to a single product automaton, they are considered as independent automata, which is advantageous for the time complexity of finding a valid transition. The states of m automata form the automata state vector $\boldsymbol{q}$

$$\boldsymbol{q} = \begin{pmatrix} q_1 & \dots & q_m \end{pmatrix}^\mathsf{T}. \tag{5.1}$$

With the joint state $\boldsymbol{s}$ and the automata state vector $\boldsymbol{q}$, the combined state z

$$z(k) = \begin{pmatrix} \boldsymbol{s}(k) & \boldsymbol{q}(k) \end{pmatrix}^\mathsf{T} \tag{5.2}$$

can be defined.

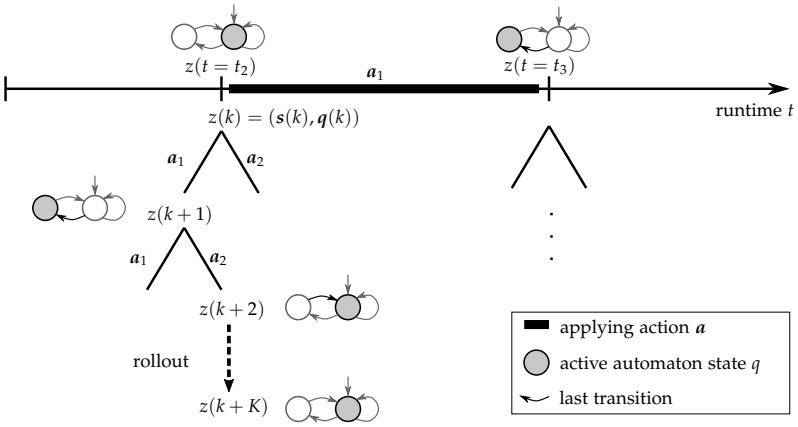


Figure 5.1.: The evolution of the product state z for a single rule monitor. An MCTS planning step is depicted exemplary at t_2. The state of the rule automaton q is tracked along both planning time and runtime (graphic from [60], ©2020 IEEE).

The successor state is then defined by

$$z(k+1) = \begin{pmatrix} \boldsymbol{s}(k+1) \\ \overline{\delta}_1\big(q_1(k), \mathcal{L}(\boldsymbol{s}(k+1))\big) \\ \vdots \\ \overline{\delta}_m\big(q_m(k), \mathcal{L}(\boldsymbol{s}(k+1))\big) \end{pmatrix}, \tag{5.3}$$

where $\boldsymbol{s}(k+1)$ is defined by the joint action $\boldsymbol{a}$. Figure 5.1 illustrates the evolution of the product state z during MCTS and over time for a single rule monitor.

5.4.2. Costs for a Violation Penalty

Let a given rule formalized in LTL φ be represented by automaton $\overline{\Lambda}_\varphi$. The weighting function $W : \mathcal{Q} \times \Sigma \rightarrow$ defines the penalty for violating that rule, assigning each transition a scalar weight:

$$W(q,\sigma) := \begin{cases} \omega_\varphi & \text{if } \forall \sigma' \in \Sigma : \overline{\delta}(\overline{\delta}(q,\sigma),\sigma') = \overline{\delta}(q,\sigma) \wedge \overline{\delta}(q,\sigma) \notin F \\ \omega_\varphi & \text{if } \textit{alive} \notin \Sigma \wedge q \notin F \\ 0 & \text{otherwise.} \end{cases} \tag{5.4}$$

A penalty ω_φ is returned, if the modified automaton $\overline{\Lambda}_\varphi$ gets reset to its initial state (violation of a safety property $\square \boldsymbol{p}$) or the automaton does not halt in an accepting state (violation of a guarantee property $\Diamond \boldsymbol{p}$). Finally, the reward for violating φ is then defined as

$$r(\boldsymbol{s}(k), \boldsymbol{s}(k+1)) = W(q, \mathcal{L}(\boldsymbol{s}(k+1))). \tag{5.5}$$

5.4.3. Multi-Objective Reward Function with Priorities

To model the multi-objective reward (e.g., safety, legal, comfort) and to prevent weight-tuning for a scalar reward, the reward elements is treated as a vector:

$$\boldsymbol{r} = \begin{pmatrix} r_1 & \dots & r_n \end{pmatrix}^\mathsf{T} \tag{5.6}$$

If goals are meant to be traded off among each other, a weighted sum scalarization is performed for these. While Wang et al. [92] try to find elements on the Pareto-optimal front, the multi-objective formulation simplifies if the objective vector is ordered according to the priority levels. An entry at index i in the reward vector denotes a higher priority than at index $i+1$.

UCT for Vectorized Rewards To incorporate a vectorized reward to the MCTS, the MCTS selection strategy is modified by computing the UCT value per reward vector element.

Relaxed Lexicographical Order If employing strict LO, the selection would favor tree branches that are suboptimal in terms of all other criteria over tree branches, where one single outcome was a collision (if that is the top-level priority). To account for the inaccuracy of the sampling-based approximation of the optimal solution, a relaxation of LO called Thresholded Lexicographical Ordering (TLO) is implemented [93]. Comparing two reward vectors $\boldsymbol{r}, \boldsymbol{r}'$ according to TLO is defined as

$$\boldsymbol{r} \preceq_{\text{TLO}} \boldsymbol{r}' \iff \exists i : r_i \leq r'_i \wedge \forall j < i : (r_j > \tau_j \wedge r'_j > \tau_j) \vee r_j = r'_j, \tag{5.7}$$

where $\preceq_{\text{TLO}}$ denotes the thresholded lexicographic comparison. TLO uses a threshold vector $\boldsymbol{\tau}$ to determine if a goal is sufficiently met. If a goal of the same priority in two compared vectors is sufficiently met, i.e., it is above the threshold in both vectors, the next lower priority objective is considered for comparison. Otherwise, strict LO is applied.

Figure 5.2 shows comparison using TLO as an example. At first, the values at index 0 (collision penalty) are compared. The *FastDec* and *KeepGap* actions' cumulative reward estimates are above the threshold and are thus next compared at index 1 (rule violation penalty). At index 1, both remaining actions are below the threshold, and thus action *KeepGap* with the highest Q-value element is selected.

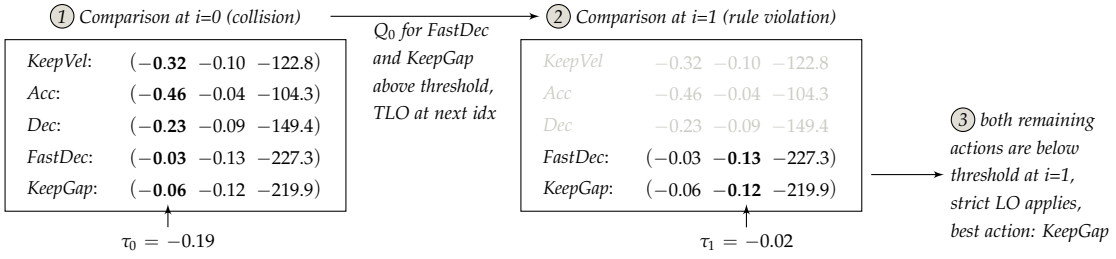


Figure 5.2.: Cumulative reward comparison for a reward vector with the three elements collision, rule violation, and comfort.

5.5. Experiments and Results

5.5.1. Experimental Setup

The approach will be studied using the open-source benchmarking framework BARK proposed in [58] and introduced in Section 2.1. The code of the proposed planning method has been

published as open-source [94]. The `RuleMonitor` from Section 4.4.2 is used to translate the formalized LTL formula to a DFA, and to manipulate the automaton. The rule monitoring and the multi-objective reward function are implemented on top of a template-based MCTS library [68]. Each scenario run is initialized with the same random seed.

Variants To study the adherence to the Safe Distance (SD) and Zipper Merge (ZIP) rules, the following variants will be compared:

- ***SA*** as a scalar single agent variant of MCTS with penalties for collision and comfort,
- ***SA-Lex*** as a lexicographic baseline implementing the same rewards as *SA*, but as a vectorized reward,
- ***SA-Lex (ZIP)*** extending *SA-Lex* by including a penalty for the ZIP rule,
- ***SA-Lex (SD)*** extending *SA-Lex* by including a penalty for the SD rule,
- ***SA-Lex (ZIP>SD)*** and ***SA-Lex (SD>ZIP)*** extending *SA-Lex* by including a penalty for the ZIP and the lower-prioritized SD rule, and vice versa.

A base reward

$$\tau_{\text{base}} = \tau_a + \tau_{\text{lat}} + \tau_{\Delta v} + \tau_\phi \tag{5.8}$$

is defined which consists of penalties for

- longitudinal acceleration $\tau_a = -\omega_a a^2 \Delta t$,
- lateral acceleration $\tau_{\text{lat}} = -\omega_{\text{lat}} |w| v^2 \Delta t$ and
- difference to desired velocity $\tau_{\Delta v} = -\omega_v |v - v_{\text{ref}}| \Delta t$

with respective weights $\omega_\square$, velocity v, reference velocity v_{ref}, acceleration a, orientation rate w and time increment Δt. The convergence of MCTS can be accelerated by incorporating additional domain knowledge. Following Kurzer et al. [95], a potential function

$$\phi(k) = -\omega_\phi |v - v_{\text{ref}}| \Delta t \tag{5.9}$$

and the potential-based shaping function

$$\tau_\phi = \gamma \phi(k+1) - \phi(k) \tag{5.10}$$

are defined. τ_{col} denotes the penalty for not colliding. τ_{sd} and τ_{zip} denote the penalties for violating the SD and ZIP rules, respectively. The ZIP rule requires vehicles on a continuing lane to let vehicles on an ending lane merge in a zipper fashion. The SD rule requires to leave a safe distance to the vehicle in front. Both rules have been defined in Chapter 3. Table 5.1 shows the reward vectors of these variants.

Action Space The discrete action space for the ego agent is modeled as lane keeping at constant acceleration for $a \in \{0\text{m/s}^2, 1\text{m/s}^2, -2\text{m/s}^2, -8\text{m/s}^2\}$, lane changing at constant velocity, and gap keeping *KeepGap* based on IDM [62]. The parameters for the *KeepGap* primitive are shown in Table A.3.

Table 5.1.: Variants of $\mathcal{B}^e$ showing the reward vector $\boldsymbol{r}$ and its size.

$\mathcal{B}^e$	Reward vector $\boldsymbol{r}$	size($\boldsymbol{r}$)
SA	$\boldsymbol{r} = (r_{\text{col}} + r_{\text{base}})^\mathsf{T}$	1
SA-Lex	$\boldsymbol{r} = (r_{\text{col}} \quad r_{\text{base}})^\mathsf{T}$	2
SA-Lex (ZIP)	$\boldsymbol{r} = (r_{\text{col}} \quad r_{\text{zip}} \quad r_{\text{base}})^\mathsf{T}$	3
SA-Lex (SD)	$\boldsymbol{r} = (r_{\text{col}} \quad r_{\text{sd}} \quad r_{\text{base}})^\mathsf{T}$	3
SA-Lex (ZIP>SD)	$\boldsymbol{r} = (r_{\text{col}} \quad r_{\text{zip}} \quad r_{\text{sd}} \quad r_{\text{base}})^\mathsf{T}$	4
SA-Lex (SD>ZIP)	$\boldsymbol{r} = (r_{\text{col}} \quad r_{\text{sd}} \quad r_{\text{zip}} \quad r_{\text{base}})^\mathsf{T}$	4

Behavior Model for Others The actions of the other vehicles are calculated using the behavior model $\mathcal{B}^o$, for which BARK's rule-based model `BehaviorMobilRuleBased` is used, with IDM as a longitudinal and MOBIL [65] as a lateral model, and a lane filtering mechanism on top of MOBIL. The model has been introduced in Section 2.1.3. To cause challenging situations for the ego vehicle, the other vehicles will travel at a desired speed of 10 m/s, which is slower than the ego vehicle at 14 m/s. Table A.2 shows all parameters of the model.

Rule Evaluators BARK provides an abstract evaluator class that calculates a given metric such as collision or step count based on the simulated world state. Section 4.4.2 extended this evaluator concept to evaluate arbitrary LTL formulas on finite traces. Each rule is captured in a `RuleMonitor`, which can be used to monitor compliance throughout the simulation. Both the behavior model (for planning) and the evaluator (for simulation) employ the same `RuleMonitor`. This allows to investigate whether the model to plan the behavior of the ego agent $\mathcal{B}^e$ truly satisfies the modeled rules.

5.5.2. Zipper Merge in Merging Scenario

The following experiment examines the variants of $\mathcal{B}^e$ in a merging scenario. Figure 5.3 shows the unfolding of the scenario over time, as well as the velocity, the traveled distance, and the zipper merge violation of the ego agent.

In Figure 5.3a, $\mathcal{B}^e$ is not modeled to follow any rules. The ego vehicle closes the gap to its preceding vehicle, leaving no space for the vehicle in the right lane to merge. In Figure 5.3b, *SA-Lex (ZIP)* models the ZIP rule, which lets it slow down to let the vehicle from the right merge in. As this variant of the ego agent must not keep a safe distance, it slows down just as much to prevent a collision but does not retain any additional space. *SA-Lex (SD)* on the other hand slows down to maintain a safe distance to the vehicle in front, but this is not enough to make enough room to let the vehicle from the right merge in. It violates the ZIP rule five steps later than the variants *SA* and *SA-Lex* did, which can be observed from Figure 5.3e. Only *SA-Lex (SD>ZIP)* in Figure 5.3d combines both desired attributes. It slows down to satisfy the zipper merge and also keeps a safe distance to the new preceding vehicle that just merged.

The experiment illustrates the behavior when different rules are modeled. If the rules are not in conflict, modeling them together yields the combined behavior of what was observed

separately for each rule before. To confirm this observation, the variants will be studied in the following in a quantitative evaluation.

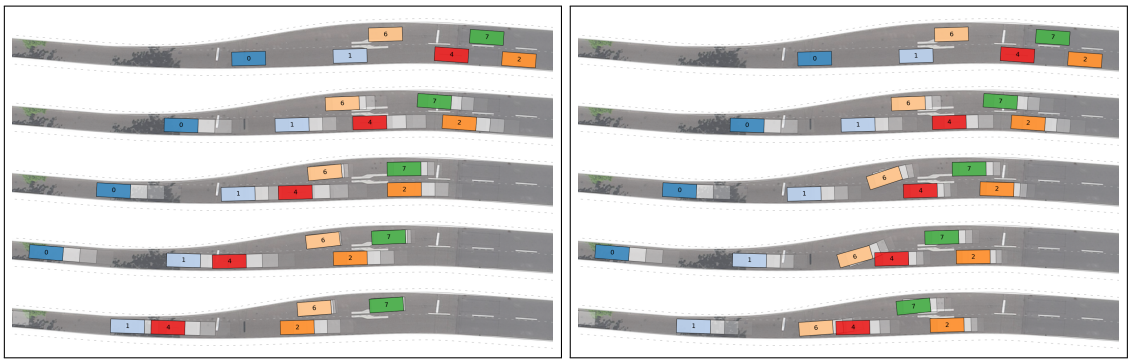
5.5.3. Quantitative Evaluation

The approach is studied using real-world data from the INTERACTION dataset [89]. Specifically, the German scenario DR_DEU_Merging_MT is used. However, it is not sufficient to replace the behavior model of the agent under test and just replay the other agents since the other agents would not react to the model under test anymore. Thus, the vehicles' initial configuration will be preserved from the dataset. Their behavior is then simulated according to $\mathcal{B}^o$. Of course, this closed-loop simulation does not precisely mimic the traffic from the dataset. To gain more insight, Figure 5.4 compares the model-based simulation and the dataset replay for the distributions of mean velocity and gap distance between the vehicles. While the traffic travels at similar speeds, the observed gaps in the dataset indicate that the parameters controlling the gap distance of IDM could be decreased even more. Despite this, the carefully tuned model parameters for $\mathcal{B}^o$ in Table A.2 yield closed-loop scenarios that mimic real traffic adequately enough for the studies of this work.

Each vehicle is simulated as the ego agent in one scenario. The ego agent is controlled by the behavior model $\mathcal{B}^e$. The scenario ends if (1) the ego agent collides with another agent, (2) the maximum number of steps are reached or (3) the ego agent reaches the goal region. The goal region is created from the last pose of the agent in the dataset. Only in the latter case (goal reached), the scenario is passed successfully. The simulation step size is 0.25 s. Figure 5.5 shows the resulting evaluation framework. $\mathcal{B}^e$ is evaluated for 200 and 500 search iterations as the termination criteria of an MCTS planning step. Figure 5.6 shows the share of controlled agents to collide, successfully reach the goal region (within a maximum scenario duration of 30 s), and violate the ZIP or SD rule.

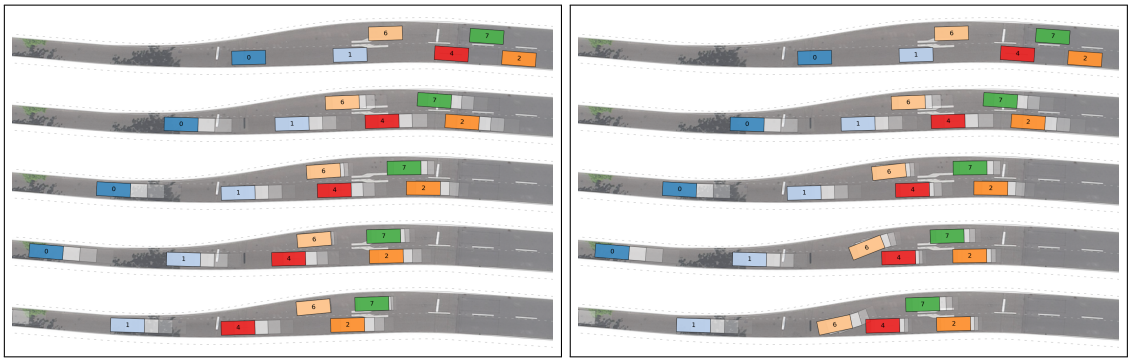
Nearly no collisions can be observed. This indicates that MCTS with high-level actions can approximate the solution well if the predicted and simulated behavior match. *SA* and *SA-Lex* do violate the ZIP rule at about 16 to 19%, which shows that this rule will not be satisfied implicitly and motivates its explicit modeling. *SA-Lex (SD)* violates it to some less extent, as keeping a safe distance all the time sometimes leaves enough space for the merging vehicle to fit in. The variants not implementing the SD rule usually do not leave much space to the front vehicle, as the correct prediction model yields an accurate estimate of what will happen given a specific action, as long as the tree is explored enough. However, to obey the SD rule and be prone to human drivers' unexpected actions, it must be modeled within the planner. The remaining safe distance violations stem from unsafe initial states at the beginning of the scenario. The variants *SA-Lex (ZIP>SD)* and *SA-Lex (SD>ZIP)* do not cause any violations of the zipping rule and also keep a safe distance for most scenarios. The results for both are similar, indicating that there exists no tradeoff in the generated scenarios between the SD and ZIP rule.

When varying the number of search iterations, the results remain mostly unchanged for the variants *SA*, *SA-Lex* and *SA-Lex (SD)*. With more search iterations, $\mathcal{B}^e$ becomes more certain about which actions will prevent rule violations. This then often yields more conservative



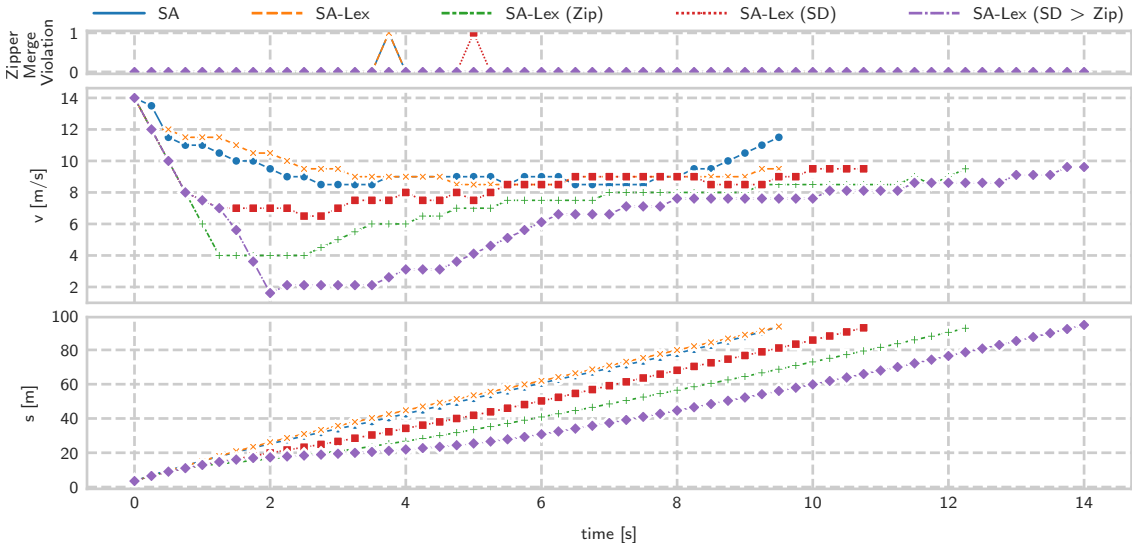
(a) $\mathcal{B}^e$ = *SA-Lex*.

(b) $\mathcal{B}^e$ = *SA-Lex (ZIP)*.



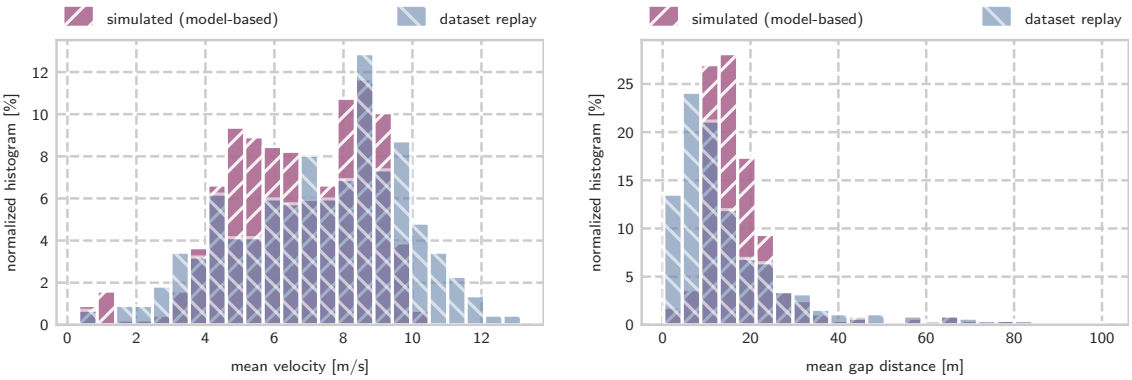
(c) $\mathcal{B}^e$ = *SA-Lex (SD)*.

(d) $\mathcal{B}^e$ = *SA-Lex (SD>ZIP)*.



(e) Violation of the Zipper Merge, velocity and traveled distance are shown for the ego agent over time.

Figure 5.3.: Merging scenario with the ego vehicle (red) on the left lane. The ego agent needs to let the vehicle in to satisfy the zipper merge rule. The experiment is performed with four variants of $\mathcal{B}^e$. Past agent states are shown with increasing transparency.



(a) Normalized histogram of mean velocity.

(b) Normalized histogram of mean gap distance to the preceding vehicle.

Figure 5.4.: Distributions of mean velocity, mean gap distance in simulated scenarios using $\mathcal{B}^o$ as behavior model for the other agents. The distributions are compared to the dataset from Table 4.1.

decisions. The slight increase of collisions for *SA-Lex (SD)* is because of some scenarios, where $\mathcal{B}^e$ does not merge early (prohibited due to the safe distance) and eventually is being hit by $\mathcal{B}^o$ from behind. In these dense lane-ending scenarios, the shortcomings of rule-based simulation models become apparent. Going forward, more elaborate models will be needed to validate driving functions in simulation. In addition, a notion of fault could help to distinguish between errors in $\mathcal{B}^o$ and $\mathcal{B}^e$ for collision cases.

5.6. Conclusion

This chapter investigated how interactive behavior planning for an autonomous vehicle can be modeled to obey the traffic rules and how this can be evaluated and tested. The interaction was modeled as a dynamic game. MCTS was used to solve the problem and fused with ideas from model checking and multi-objective optimization.

The proposed method was evaluated systematically in simulation for a merging scenario at dense traffic. Two rules were selected for the evaluation: keeping a safe distance and merging in

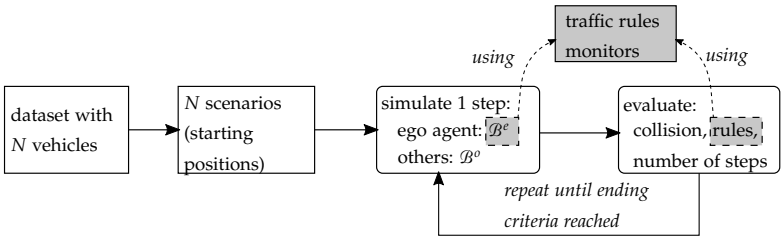


Figure 5.5.: Framework to evaluate the MCTS-based behavior of the ego agent $\mathcal{B}^e$ in a closed-loop simulation. The initial starting positions for the vehicles are taken from the dataset. The other vehicles are simulated using a behavior model $\mathcal{B}^o$. The traffic rule monitors are used within $\mathcal{B}^e$ and to evaluate the simulation (graphic from [60], ©2020 IEEE).

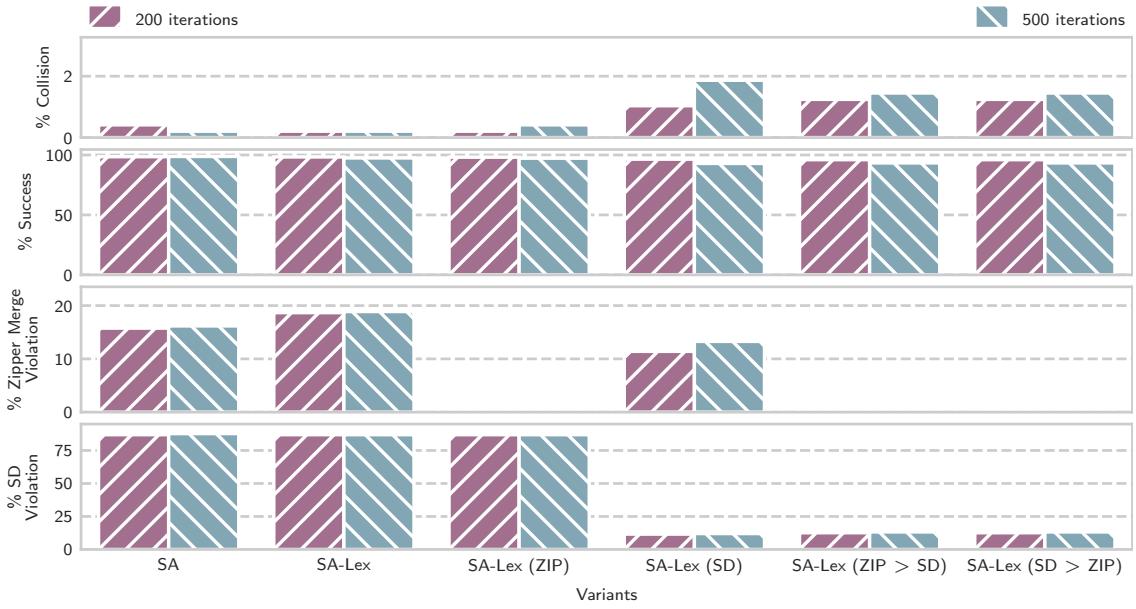


Figure 5.6.: Benchmark of the variants for $\mathcal{B}^e$. The comparison of *SA-Lex (ZIP>SD)* and *SA-Lex (SD>ZIP)* with reversed priorities shows that both modeled rules can be satisfied at the same time and thus no tradeoff between those rules is necessary in these scenario.

a zipper fashion. None of these rules were satisfied by the planning implicitly. When modeled within planning, the rules were not violated anymore in most scenarios. Some residual violations stem from inevitable situations, where for example, SD was violated at the start of the simulation already. Interestingly, by prioritizing SD higher than ZIP and vice versa, the simulation revealed that they are not contradicting in the studied scenarios. Such evaluations can help to identify contradictions in the rules.

To summarize, the proposed method allows modeling the satisfaction of multi-agent rules within planning. Such rules apply to more than one agent, whose future motion is modeled interactively. With the proposed method and evaluation toolchain, multi-agent time-dependent traffic rules can be studied on whether they have been correctly formalized and on the rules' ramifications on safety and progress.

6 Optimal Interactive Behavior Planning Satisfying Traffic Rules

6.1. Overview and Contribution

The interactive behavior planning problem is usually solved using iterative or sampling-based algorithms, such as the MCTS-based approach from Chapter 5. However, even if converging to a Nash equilibrium, the result will often be only sub-optimal and only work well for a limited set of scenarios due to discretizations of the action or state space. Furthermore, these algorithms often rely on a random sampling of the solution space or lack guarantees of convergence and thus pose open questions towards such systems' certification.

Alternatively, optimal control theory provides deterministic solution algorithms converging to an optimum. Optimization-based methods incorporate a model of the kinematics, which is forward propagated for a planning time horizon, and usually formulate constraints to account for feasibility and safety while constructing a cost function to account for comfort and other desired aspects. Although continuous local optimization such as Sequential Quadratic Programming (SQP) has been applied for trajectory planning [10], local optimality of the solution limits their usage for interactive behavior planning.

Instead, MIP offers multiple benefits that are favorable for optimal behavior planning. First, MIP will yield a globally optimal solution, whereas SQP only guarantees a local optimum. Secondly, it allows logical and integer constraints to be incorporated, whereas they usually lead to numerical issues with continuous solvers. Logical constraints are certainly favorable when formulating rules as part of the optimization problem. However, MIP cannot use a non-linear vehicle model, such as the bicycle model. Past studies used second- or third-order integrators to mitigate that [29, 32]. However, these proposed methods can only be used on a small subset of scenarios, namely straight roads, and become invalid in any other environment (roundabouts, intersections), or even during obstacle avoidance at low speeds, as the valid scope of the model formulation is limited. Outside of the valid model scope, safety is at risk, and eventually, a collision can occur.

This chapter's aim is to formulate traffic rule satisfaction for an optimization-based method. Due to the shortcomings of the existing methods to build upon, a novel optimization program is presented as well. Specifically, a linear differential game is defined for a set of interacting agents. The desired properties of collision-freeness and rule-satisfaction are realized as hard and soft constraints. The problem is solved to optimality using Mixed-Integer Quadratic Programming (MIQP). The model applies to the full-fledged range of orientations. A disjunctive formulation of the orientation enables to formulate linear constraints to prevent agent-to-agent collision while preserving the vehicle model's non-holonomic motion properties. Soft constraints account for prediction errors. The contributions of this chapter are

- linear over-approximating collision constraints,
- linear non-holonomy constraints of the vehicle,
- the methodology to compute all model parameters by linear least-square fits,
- a set of linear constraints as inter-agent collision check,
- a methodology to handle inaccurate models of other agents using soft constraints,
- a methodology to model hard or soft traffic rule satisfaction as part of the optimization program,
- a study of the feasibility of the model regarding non-holonomy and
- an analysis evaluating the consequences of the linearizations of the traffic rule formulations.

The work on behavior planning for single agents is based on the author's joint work with Tobias Kessler [57]. The extension of the formulation to multiple agents by formulating collision-constraints between multiple agents and a joint cost function is based on the author's joint work with Tobias Kessler [61]. This chapter extends these publications by encoding traffic rules within the planning formulation and a detailed evaluation studying the capabilities and limitations of the model.

6.2. Related Work

Ziegler et al. [10] proposed an optimal control formulation for single-agent trajectory planning, where they use the cost functionals and constraints shown in Table 6.1. They employ a triple integrator as a vehicle model while bounding the curvature to account for non-holonomy. The Bertha-Benz drive showed the applicability of the triple integrator model if correctly constrained. Their approach yields a nonlinear optimization problem, which is solved using SQP but only finds a local optimum. Motivated by that, approaches for maneuver planning [8, 13] have focused on finding the correct maneuver in a preliminary step. However, these approaches usually rely on a geometric partitioning of the workspace and thus scale poorly, and cannot be extended to account for interactive or cooperative planning.

Table 6.1.: Comparison of continuous optimal control formulations. Linear functionals are expressed using ①, quadratic using ②, and all other nonlinear functionals using ⓝ. The counterparts ①, ②, and ⓝ express functionals that are not (and cannot be) implemented. For the respective validity scope of the model, the models are compared along the possibility to formulate cost functionals that this work assumes to be desirable, mostly following [10]. $j_\square$ denote the cost terms, κ the curvature, v the velocity, and a the acceleration.

Source	Ziegler et al. [10]	Gutjahr et al. [15]	Nilsson et al. [96]
Problem formulation	SQP	Quadratic Programming (QP) for long, lat each	QP for x, y each
Model	triple integrator	Frenet bicycle model	double integrator
Reference frame	Cartesian, fixed	Frenet, streetwise	Cartesian, fixed
Multi-agent	no	no	no
Validity of the formulation	any road or orientation	any road or orientation	straight roads, vehicle orientation must be aligned with road
Non-holonomy constraint	κ, ⓝ	κ, ①	① by coupling v_x, v_y
Acceleration constraint $\lvert a \rvert$	②	①	ⓝ
Collision shape	disks	disks	road-aligned rectangle
Other obstacles/ agents as $(\cdot)$	disks	disks	road-aligned rectangle
"Ego-to-others" constraint	ⓝ	①	①
Road can be $(\cdot)$	everything	everything	rectangle
"Ego-to-road" constraint	ⓝ	①	①
j_{velocity} as in [10]	ⓝ	②	② in x-direction
$j_{\text{acceleration}}$ as in [10]	②	②	② in x- and y-direction
j_{jerk} as in [10]	②	②	② (if triple integrator)
j_{yawrate} as in [10]	ⓝ	② using $\dot{\kappa}$ [1]	ⓝ
$j_{\text{reference}}$ as in [10]	ⓝ	②	ⓝ

Nilsson et al. [96] introduce two QP formulations, supposedly for longitudinal and lateral control of a single vehicle based on a linear double integrator model. To account for non-holonomy, they use a linear inequality constraint that couples lateral and longitudinal velocity. However, this is only valid for small yaw angles and will yield non-drivable trajectories at intersections and roundabouts for example, since the road curvature is not taken into account. Thus, the model formulation must be seen in Cartesian coordinates and not in streetwise coordinates. The separation into consecutive optimization calls yields sub-optimal solutions and limits the solution space, as it prohibits constraints or costs on states in x- and y-direction at the same time. For example, velocity costs can only be formulated in the x-direction, which is sufficient for a straight road but does not resemble the longitudinal velocity in any other setting. The approach thus does not translate to any real scenario.

Gutjahr et al. [15] introduce two longitudinal and lateral QP in the Frenet frame based on the bicycle model. The model yields good results for driving in static environments. Similar to Ziegler et al. [10], this approach relies on a maneuver selection, as shown by Esterle et al. [16]. However, the transformation of the static environment and dynamic obstacles to local coordinates is computationally costly. With an increasing number of obstacles, the transformation outweighs

Table 6.2.: Comparison of discrete optimal control formulations. Linear functionals are expressed using ①, quadratic using ②, and all other nonlinear functionals using ⓝ. The counterparts ①, ②, and ⓝ express functionals that are not (and cannot be) implemented. For the respective validity scope of the model, the models are compared along the possibility to formulate cost functionals that this work assumes to be desirable, mostly following [10]. $j_\square$ denote the cost terms, κ the curvature, v the velocity, and a the acceleration.

Source	Qian et al. [32]	Frese et al. [29]	Fabiani et al. [31]
Problem formulation	MIQP	MILP	MIQP
Model	double integrator	double integrator	double integrator (lon.), discrete (lat.)
Reference frame	Cartesian, fixed	Cartesian, fixed	Frenet, streetwise
Multi-agent	yes, see [100]	yes	yes
Validity of the formulation	only valid on straight road, vehicle orientation must be aligned with road		
Non-holonomy constraint	① by coupling v_x, v_y	a_y, ①	no
Acceleration constraint $\lvert a \rvert$	ⓝ	approximated as ①	①
Collision shape	road-aligned rectangle	road-aligned rectangle	road-aligned rectangle
Other obstacles/ agents as $(\cdot)$	road-aligned convex polygon	road-aligned rectangle	rectangle (if in same lane) or points (if in other lane)
"Ego-to-others" constraint	①	①	①
Road can be $(\cdot)$	non-convex	non-convex	rectangle
"Ego-to-road" constraint	①	①	①
j_{velocity} as in [10]	② in x-direction	ⓝ	② in lon. direction
$j_{\text{acceleration}}$ as in [10]	② in x- and y-direction	②, approximated as ①	② in lon. direction
j_{jerk} as in [10]	② (if triple integrator)	②	② in lon. direction
j_{yawrate} as in [10]	ⓝ	ⓝ	ⓝ
$j_{\text{reference}}$ as in [10]	ⓝ	ⓝ	② (reference lane)

the benefits of the fast QP. As with [96], the separation in consecutive optimization calls yields sub-optimal solutions and limits the solution space.

Table 6.1 compares the continuous optimization works [10, 15, 96] in regards to their applicability for optimal multi-agent planning. Essentially, none of these formulations can be used. Extending [10, 15, 96] to multiple agents would yield a nonlinear problem, which prohibits the use of local solvers to find a global optimum. Likewise, the desirable nonlinear cost functionals and nonlinear constraints of [10] already prohibit the use of a mixed-integer solver, whereas the formulation of [96] has limited validity.

Miller et al. [97] base their work on [15] and formulate two consecutive longitudinal and lateral programs using MIQP similar to [96], but in local streetwise coordinates instead. However, the approach cannot account for any model-based prediction or multi-agent planning, as the agents would need to rely on separate local coordinate systems. Translating these into each other would yield nonlinear equations and prohibit using a QP or MIQP solver.

Frese et al. [29] propose a double integrator based on MILP for multiple agents, leveraging the solver's capability of yielding a global solution. That comes with the restrictions that both only linear constraints and objective functions can be formulated. Thus, the discrete optimization formulations are analyzed in Table 6.2 in terms of linear, quadratic, and nonlinear functionals. The vehicle's non-holonomy is assured by limiting an approximation of the lateral acceleration.

However, that is only accurate for small yaw angles and yields a lot of invalid solutions [98]. The collision checks with an arbitrary road polygon are modeled using a disjunctive collision check with convex polygons. They also propose to check for collision using rectangle-based vehicle-approximation of consecutive states, with each variant introducing a lot of invalid trajectories [98].

Qian et al. [32] apply the double integrator to MIQP. Similar to Nilsson et al. [96], they use a Cartesian reference frame and model the non-holonomy by bounding the velocity v_y. That leaves them to use longitudinal and lateral and x and y synonymously, which is only valid for straight roads and if the vehicle is road-oriented. Even when evading an obstacle, the yaw angle may exceed 20°, which can yield bends in the optimized trajectories that cannot be executed with a real vehicle. Thus, the work is limited to straight roads and straight driving, whereas turning at intersections is impossible. With the quadratic cost function of an MIQP, differences to longitudinal or lateral values (such as acceleration) are only possible if the reference signal is zero (such as for acceleration). Thus, deviations from the absolute velocity cannot be expressed. Eiras et al. [99] also employ a double integrator optimization within MIQP. While they perform a coordinate transformation of the reference from Cartesian to streetwise coordinates, lane boundaries, and vehicle states, they also do not model the non-holonomies, possibly leading to infeasible bends in the optimized trajectory. Burger et al. [100] extend the work of Qian et al. [32] to the cooperative multi-agent setting by extending the state space to multiple agents. However, the shortcomings of the formulation are inherited as well. Burger et al. [101] extends this optimal control approach by an intention estimation. Essentially, they solve the optimization problem for both intentions (cooperative vs. non-cooperative) while ensuring both optimal solutions match at the front section so that switching between both solutions is possible if the intention estimation was inaccurate.

Fabiani et al. [31] propose using a mixed state and action space, with a double integrator in the longitudinal direction and a discrete lane change action in the lateral direction. Their approach cannot ensure the feasibility of the lateral action regarding kinematics and collision-freeness, which they try to overcome by using heuristic rules. However, these only work for the simple use case of straight roads and will not translate to arbitrary scenarios.

Wolff et al. [102] synthesize the trajectory by formulating an MILP. They propose a method to transform linear temporal logic constraints to mixed-integer constraints. In this way, temporal constraints can be integrated into the optimization program without creating an abstraction to verify the LTL formula. However, their rules are limited to reach-avoid tasks, and the vehicle model is a linearized bicycle model not transferable to any real-world environment.

Table 6.2 summarizes the findings of identifying optimal multi-agent planning formulations. By limiting the validity of the formulations to straight roads with the vehicle orientation aligned to the road, [31, 32, 100, 101] certainly leverage the advantages of MIP (global solution, logical constraints). However, no method currently exists that is valid in generic environments.

6.3. Problem Formulation and Assumptions

This chapter aims to find a control strategy in an environment with multiple agents, subject to the following set of assumptions. Firstly, only one agent is controlled, and perfect observation of the dynamic state of other agents and their goals is assumed. Secondly, a perfect perception of the map is assumed and the ego agent's localization within that. Thirdly, other agents do not behave destructively, e.g., not aim for collisions. The problem then consists of

- multiple agents with continuous actions,
- a scene consisting of all individual states, road geometry, and obstacle information,
- each agent tries to achieve its own goals, without adversarial behavior, and
- perfect observation of states at t_0.

This is formulated as a multi-agent, non-zero-sum, differential game [103] with

- a set of agents $\mathscr{A}$ with continuous actions, and
- a joint, non-zero-sum cost function.

The strategy is executed in a receding horizon fashion. The time horizon of one planning iteration is discretized into N steps with a time interval Δt. The discrete time step is denoted by k and the range of N steps by $\mathscr{K}$. Thus, the goal is to find a sequence of actions for the ego vehicle that minimizes its costs while avoiding collisions with the environment.

6.4. Region-based Linearization Approach of Nonlinear Constraints

Following Qian et al. [32], the vehicle is modeled as a third-order point-mass system with positions $p_x(t)$, $p_y(t)$, velocities $v_y(t)$, $v_y(t)$, and accelerations $a_x(t)$, $a_y(t)$ as states. Jerk in both directions $j_x(t)$ and $j_y(t)$ are inputs of the model. To express the vehicle model as linear constraints, all nonlinearities have to be eliminated. The following section introduces how the proposed model guarantees validity for all orientations and how it performs collision checks.

6.4.1. Discretized and Disjunctive Modeling of the Orientation

Although the state space does not contain the vehicle's orientation θ, the orientation is required for an adequate collision check in Cartesian coordinates within the optimization problem. Perfect traction is assumed and therefore vehicle and tire slip are neglected. The orientation can be calculated via $\theta = \text{atan}(v_y/v_x)$. However, this equation is nonlinear, as are the trigonometric operations

$$\sin(\theta) = \sin(\text{atan}(v_y/v_x)) \tag{6.1a}$$

$$\cos(\theta) = \cos(\text{atan}(v_y/v_x)), \tag{6.1b}$$

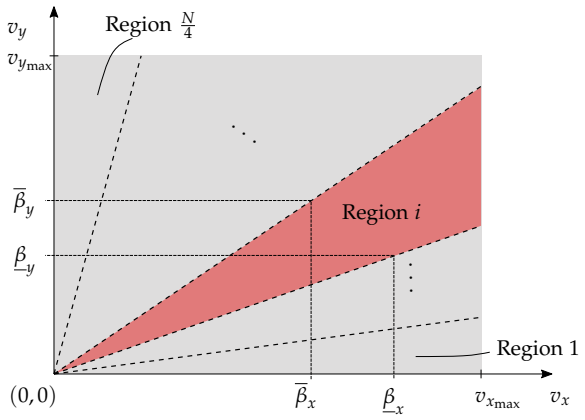


Figure 6.1.: Construction of region i described through two lines $(0,0) - (\underline{\beta}_x, \underline{\beta}_y)$ and $(0,0) - (\overline{\beta}_x, \overline{\beta}_y)$ following Equation (6.2) (graphic from [57], ©2020 IEEE).

which are necessary to calculate the front axle position. To formulate collision constraints, Equation (6.1) needs to be linearized. For the model to be valid for orientations $\theta \in [0, 2\pi]$, the orientation is discretized by introducing *regions* in the (v_x, v_y) plane, see Figure 6.1. The regions are defined by the area between two line segments starting at the origin. Consequently, for every (v_x, v_y) point within a region i, the following inequalities hold with region-dependent line parameters:

$$\underline{\beta}_x v_y \geq \underline{\beta}_y v_x \tag{6.2a}$$

$$\overline{\beta}_x v_y \leq \overline{\beta}_y v_x \tag{6.2b}$$

Subsequently, model equations will be formulated that are valid in each region, i.e., approximating the front axle position and the curvature. Figure 6.2 summarizes the proposed method of piecewise linear approximation.

Most motion planners limit the maximum longitudinal and lateral acceleration, deceleration, and jerk for driving comfort reasons. From these desired values in the vehicle's driving direction, region-specific bounds are computed in global x, y coordinates by rotating the original longitudinal and lateral limits along with the vehicle orientation. As the orientation angle, the

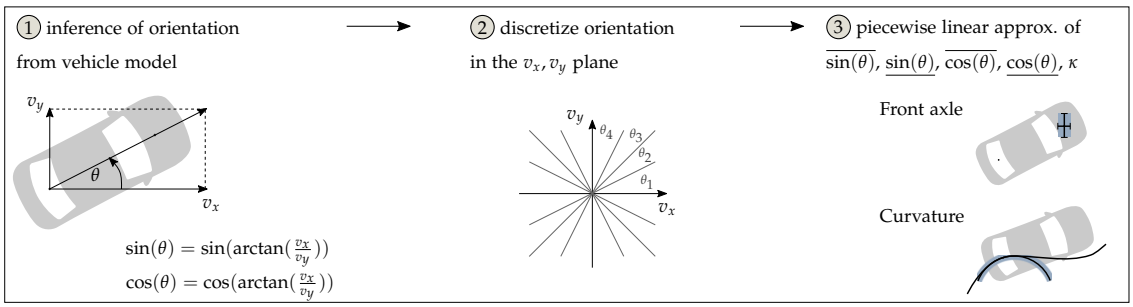


Figure 6.2.: Methodology overview of piecewise linear approximation of sine, cosine and curvature based on a discretization of the possible orientations of the vehicle. The blue shading in the last step indicates the approximation of the front axle position and the curvature.

mean angle of the respective region is chosen. By this, compliance with the original bounds in driving direction is ensured in terms of absolute values and directions.

6.4.2. Over-Approximating the Collision Shape

If the orientation were to be known, a popular approach to approximate the vehicle shape would be to use three circles with radius R_{cc} for the rear axle, middle position, and front axle similar to [10], as this facilitates efficient collision checking to arbitrary polygons. When aiming for a linear vehicle model, the true (highly nonlinear) orientation is unknown. With only an approximated orientation at hand, but not wanting to underestimate any collisions, upper and lower bound of the sine and cosine of the orientation will be calculated. Figure 6.3 illustrates this concept. With that and the vehicle's wheelbase L, upper and lower bounds for the x and y position of the front axle are calculated:

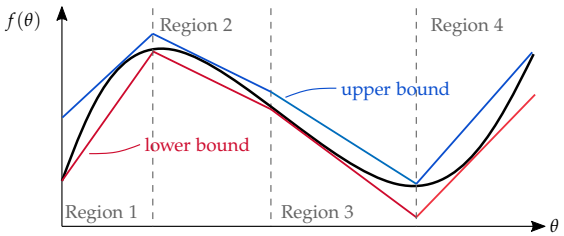


Figure 6.3.: Exemplary nonlinear function and the respective piecewise linear upper and lower bounds (graphic from [57], ©2020 IEEE).

$$\overline{f_x} := p_x + L\,\overline{\cos(\theta)} \tag{6.3a}$$

$$\underline{f_x} := p_x + L\,\underline{\cos(\theta)} \tag{6.3b}$$

$$\overline{f_y} := p_y + L\,\overline{\sin(\theta)} \tag{6.3c}$$

$$\underline{f_y} := p_y + L\,\underline{\sin(\theta)} \tag{6.3d}$$

Permuting $\overline{f_x}, \underline{f_x}$ with $\overline{f_y}, \underline{f_y}$ yields four circles for the front axle, which represent an over-approximation of the true front axle circle, as shown in Figure 6.4. For now, the middle point of the vehicle is not modeled, as this would increase the complexity of the model. However, a similar approach can be applied to the mid axle. Two methods for obtaining bounds for the sine and cosine are presented in the following.

Constant Approximation A constant approximation of the sine using the maximum and minimum orientation for each region is proposed:

$$\overline{\sin(\theta)} \doteq \max[\sin(\operatorname{atan}(\overline{\beta}_y^r, \overline{\beta}_x^r)), \sin(\operatorname{atan}(\underline{\beta}_y^r, \underline{\beta}_x^r))] \tag{6.4a}$$

$$\underline{\sin(\theta)} \doteq \min[\sin(\operatorname{atan}(\overline{\beta}_y^r, \overline{\beta}_x^r)), \sin(\operatorname{atan}(\underline{\beta}_y^r, \underline{\beta}_x^r))] \tag{6.4b}$$

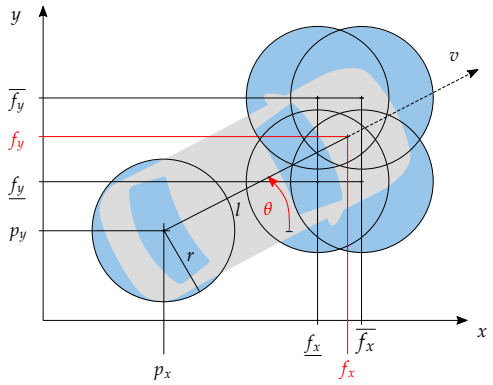


Figure 6.4.: Vehicle model with wheelbase l and disk-based collision-shape of radius R_{cc}. The variables in red are unavailable in the MIQP model formulation. The orientation θ is defined clockwise (graphic from [57], ©2020 IEEE).

The cosine can be calculated accordingly. With a higher number of regions, the error for this type of approximation will decrease. Note that this is only valid as long as a region is not defined over multiple quadrants since sine and cosine are only monotonic functions within a quadrant.

Velocity-Dependent Approximation As MIP only allows for linear constraints, only a linear combination of the state variables can be used. The $\sin(\theta)$ term is upper bounded by a first order polynomial

$$\overline{\sin(\theta)} \doteq p_{00} + p_{10} v_x + p_{01} v_y := \overline{\mathscr{P}^r_{\sin}}(v_x, v_y) \tag{6.5}$$

depending on region r with three parameters $p_\square$. Linear polynomials for the lower bound of the sinus $\underline{\mathscr{P}^r_{\sin}}$ function and the bounds of the cosine $\underline{\mathscr{P}^r_{\cos}}$ and $\overline{\mathscr{P}^r_{\cos}}$ are fitted analogously. The methodology on how to compute these parameters $p_\square$ is introduced in Section 6.5.1. This approximation will lead to a front axle position that depends on the respective velocity terms. However, that is not the case if the orientation is calculated analytically and thus leads to high errors for low velocities.

6.4.3. Modeling the Non-Holonomics

Previous MIQP formulations [30, 32, 100] have approximated the non-holonomics by bounding acceleration in x- and y-direction and by coupling the velocities via

$$v_y \in [v_x \tan(\theta_{\min}), v_x \tan(\theta_{\max})], \tag{6.6}$$

with $\theta_{\min}$ and $\theta_{\max}$ being the valid orientation range of that model. However, decoupled acceleration bounds cannot yield a non-holonomic behavior. Ziegler et al. [10] calculate the curvature using

$$\kappa = \frac{v_x a_y - v_y a_x}{\sqrt[3]{v_x^2 + v_y^2}} \tag{6.7}$$

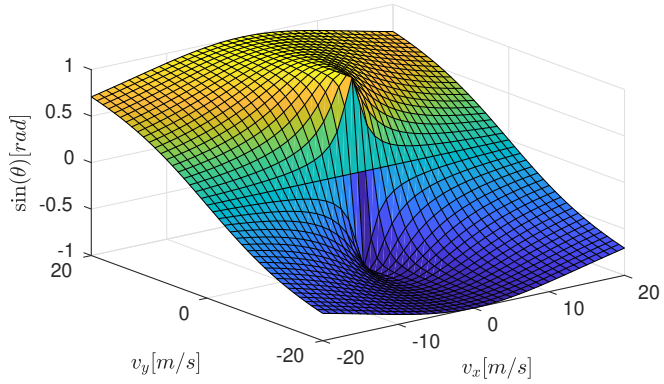


Figure 6.5.: Plot of $\sin(\theta) = \sin(\text{atan}(v_y/v_x))$ with respect to v_x and v_y. The function is obviously highly nonlinear but can be approximated in a piecewise linear fashion (graphic from [57], ©2020 IEEE).

and formulate the bound constraints $\kappa \in [\kappa_{\min}, \kappa_{\max}]$. However, Equation (6.7) is highly nonlinear and thus curvature constraints cannot be expressed as a linear constraint for MIQP. To obtain constraints dependent on a_x, a_y, the curvature limits $\kappa_{\max}, \kappa_{\min}$ are transformed using Equation (6.7) to

$$\frac{\kappa_{\max}}{v_x} \sqrt[3]{v_x^2 + v_y^2} + \frac{v_y}{v_x} a_x \geqslant a_y \tag{6.8a}$$

$$\frac{\kappa_{\min}}{v_x} \sqrt[3]{v_x^2 + v_y^2} + \frac{v_y}{v_x} a_x \leqslant a_y. \tag{6.8b}$$

The concept of region-wise linearization described in 6.4.1 can then be applied to obtain linear constraints. Two linear polynomials for upper $\overline{\mathscr{P}_\kappa^r}$ and lower $\underline{\mathscr{P}_\kappa^r}$ are fitted for bounding the curvature as shown in Section 6.5.2.

6.5. Fitting Method of Linear Polynomials

This section describes the fitting of the parameters in 6.4.2 and 6.4.3. The use of MIP only allows linear constraints. The non-linear functions are thus approximated per region. This yields polynomials of the form of Equation (6.5).

6.5.1. Fitting the Front Axle Position

Section 6.4.2 motivated the need to linearize the trigonometric functions Equation (6.1a) and Equation (6.1b), which depend on v_x and v_y. Both sine and cosine are highly nonlinear, as can be seen in Figure 6.5 for the sine function. The problem is formulated to find a piecewise upper bound to a two-dimensional nonlinear function of v_x and v_y as a linear least-squares problem with linear constraints. Figure 6.6 shows the obtained errors from the fitting. For the upper

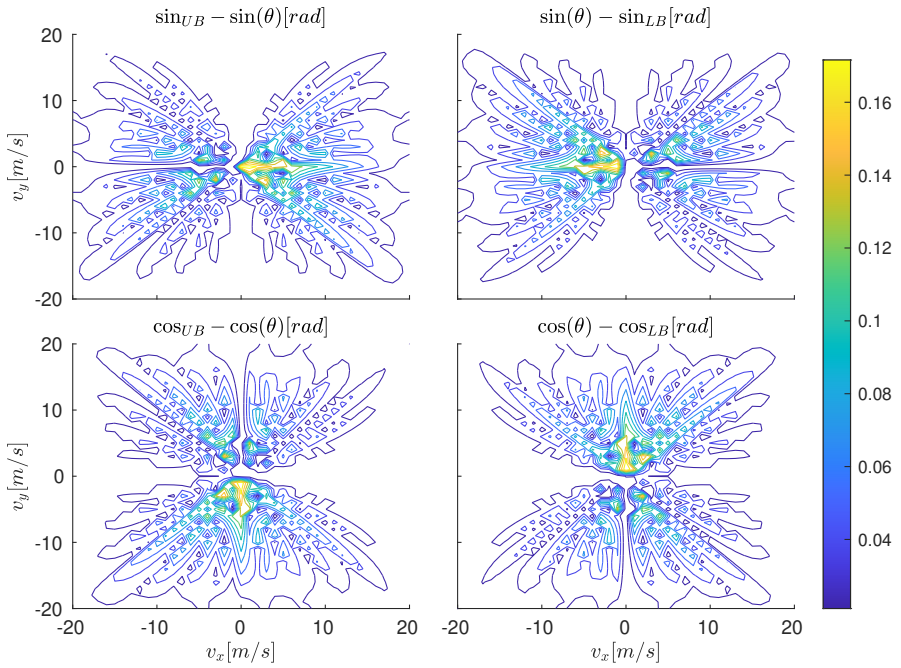


Figure 6.6.: Errors of the piecewise linear fitting using 32 regions of upper $\square_{UB}$ and lower bounds $\square_{LB}$ trigonometric functions (graphic from [57], ©2020 IEEE).

and lower bounds of the sine function $\overline{\mathscr{P}^r_{\sin}}$, $\underline{\mathscr{P}^r_{\sin}}$ and cosine function $\overline{\mathscr{P}^r_{\cos}}$, $\underline{\mathscr{P}^r_{\cos}}$, the orientation error is always below 0.16 rad. Due to

$$\sin(v_x = 0, v_y \to \pm 0) = \pm\infty \tag{6.9a}$$

$$\cos(v_x \to \pm 0, v_y = 0) = \pm\infty, \tag{6.9b}$$

the largest error occur close to the origin. For higher velocities, the errors are significantly smaller due to a better approximation of the non-linear function. Table 6.3 shows the positional error of the front axle for a variety of orientations in the first quadrant compared to the constant approximation (denoted by const.). It can be observed that the upper and lower bound always have different signs, which means that the actual axle position is always within these bounds. This shows the validity of the over-approximation of the front axle. As expected, the error becomes smaller with an increasing number of regions, as smaller linear sections are used to approximate the non-linear functions. In general, the error of the velocity-dependent approximation is less than for the constant approximation. For velocities $\lessapprox 0.1\,\mathrm{m/s}$, a constant approximation yields smaller errors, as motivated by Equation (6.9). Implementing an optimal transition strategy between the two approximations is the subject of future work.

Table 6.3.: Absolute positional errors $(f_x - [\underline{f_x}, \overline{f_x}])$ and $(f_y - [\underline{f_y}, \overline{f_y}])$ for the approximation of the front and rear axle in meters using 32 regions.

θ	v		16 regions	32 regions	64 regions	128 regions
0°	const.	x	0.21, 0.00	0.05, 0.00	0.01, 0.00	0.00, 0.00
		y	0.00, -1.07	0.00, -0.55	0.00, -0.27	0.00, -0.14
	$0.1\frac{m}{s}$	x	0.20, -0.01	0.07, -0.01	0.01, -0.01	0.01, -0.01
		y	0.01, -0.97	0.01, -0.52	0.01, -0.09	0.01, -0.01
	$20\frac{m}{s}$	x	0.01, -0.08	0.01, -0.03	0.01, -0.01	0.01, -0.01
		y	0.01, -0.01	0.01, -0.01	0.01, -0.01	0.01, -0.01
45°	const.	x	0.00, -0.61	0.00, -0.35	0.00, -0.18	0.00, -0.09
		y	0.91, 0.00	0.42, 0.00	0.20, 0.00	0.10, 0.00
	$0.1\frac{m}{s}$	x	0.01, -0.61	0.01, -0.36	0.01, -0.17	0.01, -0.03
		y	0.89, -0.01	0.43, -0.01	0.18, -0.01	0.03, -0.01
	$20\frac{m}{s}$	x	0.04, -0.19	0.02, -0.12	0.02, -0.06	0.01, -0.02
		y	0.27, -0.11	0.14, -0.05	0.06, -0.02	0.02, -0.01
90°	const.	x	0.00, -1.07	0.00, -0.55	0.00, -0.27	0.00, -0.14
		y	0.21, 0.00	0.05, 0.00	0.01, 0.00	0.00, 0.00
	$0.1\frac{m}{s}$	x	0.01, -0.98	0.01, -0.53	0.01, -0.10	0.01, -0.01
		y	0.20, -0.01	0.07, -0.01	0.01, -0.01	0.01, -0.01
	$20\frac{m}{s}$	x	0.01, -0.01	0.01, -0.01	0.01, -0.01	0.01, -0.01
		y	0.01, -0.08	0.01, -0.03	0.01, -0.01	0.01, -0.01

6.5.2. Fitting the Curvature

As discussed in Section 6.4.3, the curvature constraints are approximated by Equation (6.8). The polynomials $\mathscr{P}_\kappa^r$ are fitted on

$$\frac{\kappa_{\max}}{v_x}\sqrt[3]{v_x^2 + v_y^2} \equiv \overline{\mathscr{P}_\kappa^r} \geq a_y - \frac{v_y}{v_x}a_x \tag{6.10a}$$

$$\frac{\kappa_{\min}}{v_x}\sqrt[3]{v_x^2 + v_y^2} \equiv \underline{\mathscr{P}_\kappa^r} \leq a_y - \frac{v_y}{v_x}a_x \tag{6.10b}$$

and used in inequality constraints bounding $a_\star$ as Equation (6.10) indicates. The term v_y/v_x in the MIQP formulation is approximated by the mean orientation within the respective region using the region boundaries, see Equation (6.2). The larger the regions (the fewer number of regions), the higher the error will be. Then, two unconstrained linear least-square problems minimize the error to Equation (6.10a), yielding the linear polynomial $\overline{\mathscr{P}_\kappa^r}$, and Equation (6.10b), yielding the linear polynomial $\underline{\mathscr{P}_\kappa^r}$.

6.6. Formulating the Planning Problem as Linear Dynamic Game

In this section, a differential game is formulated without restricting the model's validity scope and then solved using MIQP. The linear constraint formulation yields a linear differential game.

The vehicle motion model is expressed as discrete linear constraints. Using binary variables, collision-freeness and a correct non-holonomic motion of each vehicle is ensured. A joint objective function keeps the solution close to the reference.

Table 6.4.: Decision variables used throughout this chapter, with discrete time dependency k, region dependency r, environment dependency λ, and obstacle dependency o.

Variable	Range
$p_x(k)$, $p_y(k)$	free
$v_x(k)$, $v_y(k)$	$[\underline{v}, \overline{v}]$
$a_x(k)$, $a_y(k)$	$[\underline{a}, \overline{a}]$
$u_x(k)$, $u_y(k)$	$[\underline{u}, \overline{u}]$
$\overline{f_x}(k)$, $\underline{f_x}(k)$, $\overline{f_y}(k)$, $\underline{f_y}(k)$	free
$\xi_x^{ij}(k)$, $\xi_y^{ij}(k)$	$[0,\ D(k)]$
$\zeta(k,o)$	binary
$\rho(k,r)$	binary
$\Psi(k)$	binary
$e(k,\lambda)$	binary
$\varsigma_p(k,o)$	binary
$\alpha_1^{ij}(k)$, $\alpha_2^{ij}(k)$, $\alpha_3^{ij}(k)$, $\alpha_4^{ij}(k)$	binary

Subsequently, decision variables used in the following are shown in Table 6.4. The subscript $\square_{\text{ref}}$ denotes the respective reference. A discrete time range from k_2 to k_N is optimized with Δt increment. $\mathscr{K}$ denotes the discrete time horizon $[k_1, \ldots, k_N]$. All decision variables are initialized with the current state of the vehicle at k_1. The speed is bounded by the minimal and maximal values $\underline{v}$ and $\overline{v}$ from the fitting, as the approximations are only valid there. A slack is used for the agent-to-agent collision check and bounded by the desired threshold D. The subscript $\square_{\star}$ denotes the respective term for both x- and y-direction. In the following, this notation will be used for the equations' compactness wherever it does not lead to ambiguity.

6.6.1. Formulating the Vehicle Model as Constraints

The vehicle dynamics are defined by:

$$\begin{bmatrix} p_{\star}(k_{i+1}) \\ v_{\star}(k_{i+1}) \\ a_{\star}(k_{i+1}) \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & \Delta t^2/2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{\star}(k_i) \\ v_{\star}(k_i) \\ a_{\star}(k_i) \end{bmatrix} + \begin{bmatrix} \Delta t^3/6 \\ \Delta t^2/2 \\ \Delta t \end{bmatrix} u_{\star}(k_i) \quad \forall k_i \in [k_1, \ldots, k_{N-1}] \tag{6.11}$$

With this linear model, correct non-holonomic motion and correct acceleration and steering angle limits are only valid around a small reference orientation. By introducing validity regions, the proposed approach overcomes this limitation, see Section 6.4.1. The set of regions covering the entire orientation range of 360° degrees is denoted by $\mathscr{R}$. Region-dependent parameters are denoted in the following by the superscript $\square^r$.

The binary decision variable $\rho(k,r)$ shall define in which region r the vehicle is in at time k. It is set according to the two lines defining the region, cf. Equation (6.2). The solution is forced to lie in exactly one region:

$$\sum_{r \in \mathscr{R}} \rho(k,r) = 1 \quad \forall k \in \mathscr{K} \tag{6.12}$$

Since with an increasing number of regions, the evaluation time of the model increases as well, the optimization is restricted a-priori to only use a set of *allowed* regions. A parameter ϱ^r is pre-computed accordingly. Non-allowed regions are those that cannot physically be reached anyway within the planning horizon. To implement logical constraints, the well-known technique of introducing a big constant M to switch inequalities is used. The intuitive explanation of $M(1-\rho(k,r))$ is "this equation is active if and only if region r is active at time step k". The following set of constraints sets the active region:

$$\underline{\beta}_x^r v_y(k) \geq \underline{\beta}_y^r v_x(k) - M(1-\rho(k,r)) \tag{6.13a}$$

$$\overline{\beta}_x^r v_y(k) \leq \overline{\beta}_y^r v_x(k) + M(1-\rho(k,r)) \tag{6.13b}$$

$$\forall k \in \mathscr{K},\ \forall r \in \mathscr{R},\ \text{if } \varrho^r(r) = 1$$

Regions marked as impossible by ϱ^r may not be selected. With this implementation, the model formulation is generic for all scenarios.

$$\rho(k,r) = 0 \quad \forall k \in \mathscr{K},\ \forall r \in \mathscr{R},\ \text{if } \varrho^r = 0 \tag{6.14}$$

Region-dependent limits on acceleration and jerk are imposed to make sure to always meet the correct absolute possible acceleration and jerk. As acceleration $a_{\star}$ and jerk $u_{\star}$ are defined in a fixed Cartesian system, the limits are rotated with the regions. Equation (6.15) states the constraints for the acceleration, the formulation for jerk is alike. Note that the speeds are naturally bounded correctly by Equation (6.13).

$$a_{\star}(k) \leq \overline{a_{\star}^r} + M(1-\rho(k,r)) \tag{6.15a}$$

$$a_{\star}(k) \geq \underline{a_{\star}^r} - M(1-\rho(k,r)) \tag{6.15b}$$

$$\forall k \in \mathscr{K},\ \forall r \in \mathscr{R},\ \text{if } \varrho^r = 1$$

6.6.2. Modeling the Non-Holonomy as Constraints

To ensure a correct non-holonomic movement, the lateral acceleration is limited using the maximally available curvature per region as motivated in Section 6.4.3. Lower and upper linear approximation polynomials $\overline{\mathscr{P}_\kappa^r}$ and $\underline{\mathscr{P}_\kappa^r}$ are fitted, that depend on v_x, v_y, and the region r. The following constraints model the inequalities bounding the curvature κ, as stated in Equation (6.8):

$$a_y(k) - \frac{\underline{\beta}_y^r + \overline{\beta}_y^r}{\underline{\beta}_x^r + \overline{\beta}_x^r} a_x(k) \leq \overline{\mathscr{P}_\kappa^r}(v_x(k), v_y(k)) + M(1-\rho(k,r)) + M\Psi(k) \tag{6.16a}$$

$$a_y(k) - \frac{\underline{\beta}_y^r + \overline{\beta}_y^r}{\underline{\beta}_x^r + \overline{\beta}_x^r} a_x(k) \geq \underline{\mathscr{P}_\kappa^r}(v_x(k), v_y(k)) + M(1-\rho(k,r)) + M\Psi(k) \tag{6.16b}$$

$$\forall k \in \mathscr{K},\ \forall r \in \mathscr{R},\ \text{if } \varrho^r = 1$$

At very low vehicle speeds, too tight curvature constraints limit the acceleration so that accelerations other than zero are not possible. In contrast, too loose curvature constraints will violate the non-holonomy. Therefore, a minimum speed limit is added. If either $|v_x|$ or $|v_y|$ is below that limit, region changes are not allowed, which is indicated by setting the binary helper variable Ψ to true.

$$\rho(k_i, r) - \rho(k_{i-1}, r) \leq 1 - \Psi(k_i) \tag{6.17a}$$
$$\rho(k_i, r) - \rho(k_{i-1}, r) \geq -1 + \Psi(k_i) \tag{6.17b}$$
$$\forall k_i \in [k_2, \ldots, k_N],\ \forall r \in \mathscr{R}$$

6.6.3. Approximating the Front Axle Position as Constraints

In Section 6.4.2, it was introduced how to approximate lower and upper bounds for the position of the front axle of the vehicle. Using this idea, this section formulates constraints performing an (over-)approximative collision check for the front axle instead of computing the intersection or distance of the actual vehicle shape with obstacles or the environment. The lower and upper bounds $\underline{f_{\star}}$, $\overline{f_{\star}}$ of the actual, unknown, front position (f_x, f_y) are calculated for the current region r where l denotes the wheelbase of the vehicle. Only the equations for the upper bounds are stated here. The lower bound constraints are formulated alike:

$$M(\rho(k,r)-1) \leq \overline{f_x}(k) - p_x(k) - l\overline{\mathscr{P}^r_{\cos}}(v_x(k), v_y(k)) \tag{6.18a}$$
$$M(1-\rho(k,r)) \geq \overline{f_x}(k) - p_x(k) - l\overline{\mathscr{P}^r_{\cos}}(v_x(k), v_y(k)) \tag{6.18b}$$
$$M(\rho(k,r)-1) \leq \overline{f_y}(k) - p_y(k) - l\overline{\mathscr{P}^r_{\sin}}(v_x(k), v_y(k)) \tag{6.18c}$$
$$M(1-\rho(k,r)) \geq \overline{f_y}(k) - p_y(k) - l\overline{\mathscr{P}^r_{\sin}}(v_x(k), v_y(k)) \tag{6.18d}$$
$$\forall k_i \in \mathscr{K},\ \forall r \in \mathscr{R},\ \text{if } \varrho^r(r) = 1$$

6.6.4. Constraints Limiting the Model to Stay on the Road

This section introduces how the vehicle is enforced to stay within an environment modeled as an arbitrary, potentially non-convex closed polygon Γ [29]. The environment is deflated with the radius of the collision circles, see Figure 6.7.

Non-convex environment polygons are split into several convex sub-polygons λ. The vehicle is enforced to be in at least one of these convex sub-polygons. These sub-polygons are represented by a set of line segments, where a line segment between two points a^l and b^l is denoted by l. With this strategy, the polygon-to-polygon collision check narrows down to a point-to-polygon check. The rear axle position $p_{\star}$ and the lower and upper bounds of the front axle position, namely the four points $(\underline{f_x}, \underline{f_y})$, $(\underline{f_x}, \overline{f_y})$, $(\overline{f_x}, \underline{f_y})$, and $(\overline{f_x}, \overline{f_y})$, are enforced to be within the environment polygon. Each set of constraints is formulated in a similar manner, Equation (6.19) states the equations for the point (p_x, p_y). The decision variable e models that all five points do not collide

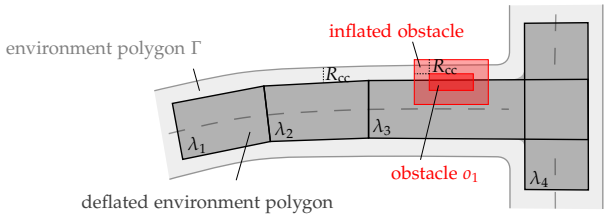


Figure 6.7.: Schematic sketch showing how the environment polygon Γ is shrinked by the collision circle radius R_{cc} and split into several convex polygons $\lambda_\square$ and how obstacles $o_\square$ are inflated with R_{cc}. (modified graphic from [57], ©2020 IEEE)

with the environment sub-polygon λ at time k. By $l_\star$, this work denotes $b^l_\star - a^l_\star$ for x and y respectively.

$$l_x(p_y(k) - a^l_y) - l_y(p_x(k) - a^l_x) \leq -Me(k,\lambda) \tag{6.19a}$$
$$l_x(\underline{f_y}(k) - a^l_y) - l_y(\underline{f_x}(k) - a^l_x) \leq -Me(k,\lambda) \tag{6.19b}$$
$$l_x(\underline{f_y}(k) - a^l_y) - l_y(\overline{f_x}(k) - a^l_x) \leq -Me(k,\lambda) \tag{6.19c}$$
$$l_x(\overline{f_y}(k) - a^l_y) - l_y(\underline{f_x}(k) - a^l_x) \leq -Me(k,\lambda) \tag{6.19d}$$
$$l_x(\overline{f_y}(k) - a^l_y) - l_y(\overline{f_x}(k) - a^l_x) \leq -Me(k,\lambda) \tag{6.19e}$$
$$\forall k \in \mathcal{K},\ \forall l \in \lambda,\ \forall \lambda \in \Gamma$$

The following constraint ensures that all vehicle points are at least within one of the environment polygons:

$$\sum_{\lambda \in \Gamma} e(k,\lambda) \leq |\Gamma| - 1 \quad \forall k \in \mathcal{K}, \tag{6.20}$$

where $|\Gamma|$ denotes the number of convex sub-environments.

6.6.5. Formulating Collision Avoidance as Constraints

The vehicle must not collide with an arbitrary number of static or dynamic convex obstacle polygons. Obstacles are thus inflated with the radius of the collision circles R_{cc}, cf. Figure 6.7.

Let l again denote one line segment of one obstacle o within the set of obstacles O with startpoint a^l and endpoint b^l. The vehicle rear axle position $p_\star$ and the four permutations of the front axle bounds are enforced to be collision-free. In contrast to the environment, which is assumed as constant over time, the obstacle polygons may vary their position and shape over time but preserve the polygon topology. The decision variables $\varsigma_\star$ indicate whether none of the five points collide with obstacle o. Equation (6.21) states the inequalities for point (p_x, p_y) constraining ς_p. The four points representing the collision shape approximation of the front axle $f_\star$ are each taken into account by four more sets of similar decision variables $\varsigma_\star$ and sets of inequalities:

$$l_x(p_y(k) - a^l_y) - l_y(p_x(k) - a^l_x) \leq M\varsigma_p(k,l) \quad \forall k \in \mathcal{K},\ \forall l \in o,\ \forall o \in O \tag{6.21}$$

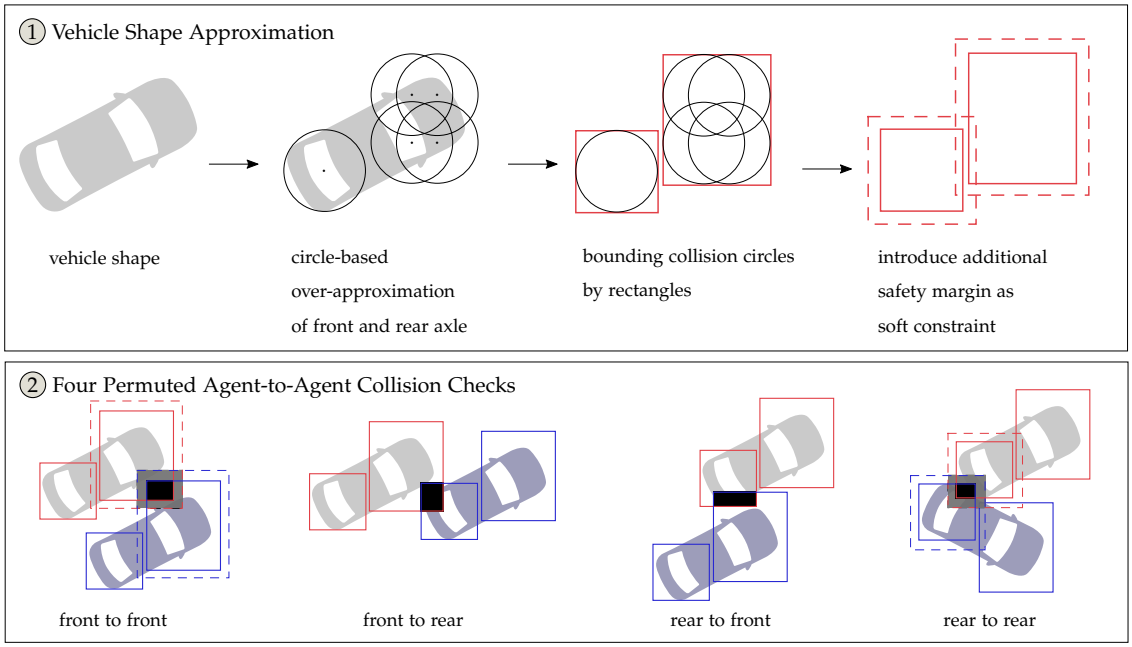


Figure 6.8.: Approximation of the vehicle shape to formulate the agent-to-agent collision check based on the rectangles of the respective agents. The black area indicates a collision. The gray area indicates colliding safety margins.

Denoting the number of line segments in a sub-polygon by $|o|$, each of the five points are enforced to not lie within an obstacle:

$$\sum_{l \in o} \varsigma_p(k,l) \leq |o| - 1 \quad \forall k \in \mathscr{K},\ \forall o \in O \tag{6.22}$$

6.6.6. Multi-Agent Collision Constraints

Each vehicle's shape is approximated for the agent-to-agent collision constraint. In the proposed linear model, the actual vehicle's orientation is unavailable. However, with the region-based formulation, a lower- and upper-bounding rectangle can be computed for the front axle's center.

In the following, the superscript $\square^i$ refers to the respective variable of the agent i. For collision avoidance, the vehicle shape is approximated by circles with radius R^i_{cc}, one around the rear axle center, and four for the front axle approximation. To avoid agent-to-agent collisions, the proposed approach over-approximates these circles with axis-aligned squares again, as sketched in Figure 6.8. A better approximation of the circles, such as two rectangles, yields more constraints and binary variables, increasing the runtime without providing a huge benefit. Thus, four sets of constraints are formulated: The first prevents collisions between the rear parts of two agents. The second prevents collisions between the rear part of the first and the front part of the second agent, the third vice versa. The fourth prevents collisions between the front parts of both agents for each pair of agents.

Rear-To-Rear Collision The rear-part-to-rear-part collision constraint of two agents A^i and A^j is based on the following logical formula. The sum of both radii is defined as $R^{i+j} := R^i + R^j$. A collision occurs at one time step k if and only if

$$\begin{aligned} p_x^i(k) &\geq p_x^j(k) - R^{i+j} \wedge \\ p_x^i(k) &\leq p_x^j(k) + R^{i+j} \wedge \\ p_y^i(k) &\geq p_y^j(k) - R^{i+j} \wedge \\ p_y^i(k) &\leq p_y^j(k) + R^{i+j}. \end{aligned} \tag{6.23}$$

Intuitively, Equation (6.23) states that a collision occurs if both the absolute distance in x-direction $|p_x^i(k) - p_x^j(k)|$ and y-direction $|p_y^i(k) - p_y^j(k)|$ is smaller than R^{i+j}. Logical negation yields that two agents do not collide at time step k if and only if

$$\begin{aligned} p_x^i(k) &\leq p_x^j(k) - R^{i+j} \vee \\ p_x^i(k) &\geq p_x^j(k) + R^{i+j} \vee \\ p_y^i(k) &\leq p_y^j(k) - R^{i+j} \vee \\ p_y^i(k) &\geq p_y^j(k) + R^{i+j}. \end{aligned} \tag{6.24}$$

This is transformed to linear constraints using a set of four decision variables $\alpha_{\square}^{ij}$, one for each inequality and an appropriately chosen big constant M:

$$p_x^i(k) \leq p_x^j(k) - R^{i+j} + M\alpha_1^{ij}(k) \tag{6.25a}$$

$$p_x^i(k) \geq p_x^j(k) + R^{i+j} - M\alpha_2^{ij}(k) \tag{6.25b}$$

$$p_y^i(k) \leq p_y^j(k) - R^{i+j} + M\alpha_3^{ij}(k) \tag{6.25c}$$

$$p_y^i(k) \geq p_y^j(k) + R^{i+j} - M\alpha_4^{ij}(k) \tag{6.25d}$$

$$3 \geq \sum_{a=1}^{4} \alpha_a^{ij}(k) \; \forall \; k \in \mathscr{K} \tag{6.25e}$$

Equation (6.25e) represents the logical formulas in Equation (6.24) by coupling the four constraints in Equation (6.25a) - Equation (6.25d) and makes sure no more than three are active, and hence no rear part-to-rear part collision occurs.

To cope with agents that are not controlled directly (e.g., human-driven vehicles), the computed motion will not exactly match the reality. This yields prediction errors for the uncontrolled agents, possibly leading to infeasible optimization problems or imminent collisions. To consider these prediction errors, an additional safety distance for the ego agent to all uncontrolled agents is introduced as a *soft constraint*. Inspired by the formulation in [104], *slack variables* ξ are introduced to the agent-to-agent collision constraints. With the desired safety distance of both agents $D(k)$ and a set of slack variables $\xi_x(k), \xi_y(k) \in [0,\ D(k)]$, the slack-based safety distance for the rear-to-rear collision check can be defined as

$$\mathscr{D}_{\bigstar,RR}^{ij}(k) := D(k) - \xi_{\bigstar}^{ij,RR}(k). \tag{6.26}$$

Using this, Equation (6.25) can now be modified to

$$p_x^i(k) \leq p_x^j(k) - R^{i+j} - \mathcal{D}_{x,RR}^{ij}(k) + M\alpha_1^{ij}(k) \tag{6.27a}$$
$$p_x^i(k) \geq p_x^j(k) + R^{i+j} + \mathcal{D}_{x,RR}^{ij}(k) - M\alpha_2^{ij}(k) \tag{6.27b}$$
$$p_y^i(k) \leq p_y^j(k) - R^{i+j} - \mathcal{D}_{y,RR}^{ij}(k) + M\alpha_3^{ij}(k) \tag{6.27c}$$
$$p_y^i(k) \geq p_y^j(k) + R^{i+j} + \mathcal{D}_{y,RR}^{ij}(k) - M\alpha_4^{ij}(k) \tag{6.27d}$$
$$3 \geq \sum_{a=1}^{4} \alpha_a^{ij}(k) \; \forall \; k \in \mathscr{K}. \tag{6.27e}$$

The slack variables will be included in the cost function (see Section 6.6.8). The optimizer will then seek to keep the slack variables as small as possible. Consequently, in Equation (6.27), the additional slack-based safety distance $\mathcal{D}$ will be as high as possible. With this concept, fatal prediction errors (immanent collisions) are mitigated. Note that adding a hard safety margin only leads to more conservative behavior and does not prevent the optimization problem from becoming infeasible.

Rear-To-Front Collision To prevent collisions between the rear part of agent A^i and the front part of agent A^j, logical constraints similar to 6.23 are formulated: A collision occurs at k if and only if

$$\begin{aligned} p_x^i(k) &\geq \underline{f_x}^j(k) - R^{i+j} \wedge \\ p_x^i(k) &\leq \overline{f_x}^j(k) + R^{i+j} \wedge \\ p_y^i(k) &\geq \underline{f_y}^j(k) - R^{i+j} \wedge \\ p_y^i(k) &\leq \overline{f_y}^j(k) + R^{i+j}. \end{aligned} \tag{6.28}$$

Here, the point $(p_x^i,\ p_y^i)$ of agent A^i is forced to be outside the front axle approximation rectangle of agent A^j, which gets enlarged by the sum of the collision circle radii. The set of constraints is formulated as described for the rear-part-to-rear-part collision case. This can again be formulated as a set of five constraints with appropriate decision variables:

$$p_x^i(k) \leq \underline{f_x}^j(k) - R^{i+j} + M\alpha_5^{ij}(k) \tag{6.29a}$$
$$p_x^i(k) \geq \overline{f_x}^j(k) + R^{i+j} - M\alpha_6^{ij}(k) \tag{6.29b}$$
$$p_y^i(k) \leq \underline{f_y}^j(k) - R^{i+j} + M\alpha_7^{ij}(k) \tag{6.29c}$$
$$p_y^i(k) \geq \overline{f_y}^j(k) + R^{i+j} - M\alpha_8^{ij}(k) \tag{6.29d}$$
$$3 \geq \sum_{a=5}^{8} \alpha_a^{ij} \; \forall \; k \in \mathscr{K} \tag{6.29e}$$

Another analog set of constraints prevents collisions between the rear part of A^j and the front part of A^i. This can be derived from 6.29 by interchanging the agent's indices:

$$p_x^j(k) \leq \underline{f_x}^i(k) - R^{i+j} + M\alpha_9^{ij}(k) \tag{6.30a}$$
$$p_x^j(k) \geq \overline{f_x}^i(k) + R^{i+j} - M\alpha_{10}^{ij}(k) \tag{6.30b}$$
$$p_y^j(k) \leq \underline{f_y}^i(k) - R^{i+j} + M\alpha_{11}^{ij}(k) \tag{6.30c}$$
$$p_y^j(k) \geq \overline{f_y}^i(k) + R^{i+j} - M\alpha_{12}^{ij}(k) \tag{6.30d}$$
$$3 \geq \sum_{a=9}^{12} \alpha_a^{ij} \; \forall \; k \in \mathscr{K} \tag{6.30e}$$

After weighing the potential benefit with the computational restrictions, the soft constraints for the rear-to-front and front-to-rear collision constraints have not been implemented, as the soft constraints front-to-front and rear-to-rear should be sufficient to push the agents away from each other.

Front-to-front collision Collisions between the fronts of agents A^i and A^j are avoided utilizing the same strategy but enforcing the center point of the front axle approximation rectangle of A^j to preserve sufficient distance to the front axle approximation rectangle of A^i. Concretely, the sufficient distance is defined as R^{i+j} plus the size of the approximation rectangle of agent A^j. Hence, no front-part-to-front-part collision occurs at time step k if and only if

$$\begin{aligned}
\frac{1}{2}\left(\overline{f_x}^j(k) + \underline{f_x}^j(k)\right) &\leq \underline{f_x}^i(k) - R^{i+j} - \frac{1}{2}\left(\overline{f_x}^j(k) - \underline{f_x}^j(k)\right) \vee \\
\frac{1}{2}\left(\overline{f_x}^j(k) + \underline{f_x}^j(k)\right) &\geq \overline{f_x}^i(k) + R^{i+j} + \frac{1}{2}\left(\overline{f_x}^j(k) - \underline{f_x}^j(k)\right) \vee \\
\frac{1}{2}\left(\overline{f_y}^j(k) + \underline{f_y}^j(k)\right) &\leq \underline{f_y}^i(k) - R^{i+j} - \frac{1}{2}\left(\overline{f_y}^j(k) - \underline{f_y}^j(k)\right) \vee \\
\frac{1}{2}\left(\overline{f_y}^j(k) + \underline{f_y}^j(k)\right) &\geq \overline{f_y}^i(k) + R^{i+j} + \frac{1}{2}\left(\overline{f_y}^j(k) - \underline{f_y}^j(k)\right).
\end{aligned} \tag{6.31}$$

Equation (6.31) is used again to derive a set of constraints as in the rear-to-rear collision case, that satisfy the slack-controlled safety distance.

$$0 \leq \underline{f_x}^i(k) - \overline{f_x}^j(k) - R^{i+j} - \mathscr{D}_{x,FF}^{ij}(k) + M\alpha_{13}^{ij}(k) \tag{6.32a}$$
$$0 \geq \overline{f_x}^i(k) - \underline{f_x}^j(k) + R^{i+j} + \mathscr{D}_{x,FF}^{ij}(k) - M\alpha_{14}^{ij}(k) \tag{6.32b}$$
$$0 \leq \underline{f_y}^i(k) - \overline{f_y}^j(k) - R^{i+j} - \mathscr{D}_{y,FF}^{ij}(k) + M\alpha_{15}^{ij}(k) \tag{6.32c}$$
$$0 \geq \overline{f_y}^i(k) - \underline{f_y}^j(k) + R^{i+j} + \mathscr{D}_{y,FF}^{ij}(k) - M\alpha_{16}^{ij}(k) \tag{6.32d}$$
$$3 \geq \sum_{a=13}^{16} \alpha_a^{ij}(k) \; \forall \; k \in \mathscr{K}. \tag{6.32e}$$

6.6.7. Traffic Rules

In this section, a set of traffic rules is selected to be implemented as part of the optimization to show the feasibility of the modeling approach. The optimization program cannot incorporate

the penalty for violating a rule from the `RuleMonitor`, as MIQP does not allow nonlinear terms in the cost function. Thus, the rules need to be re-modeled in the optimization program.

Lane Selection – Keep in Right-Most Lane The reference trajectory, which is included in the cost function, is generated based on the centerline of the reference lane corridor Ξ_{ref}. Thus, the rule to keep in the right-most lane can be realized by selecting Ξ_{ref}. Algorithm 2 shows the selection algorithm, where a function to check the validity and existence of a lane corridor `isValid`$(\cdot)$ and a function to return the remaining length of a lane corridor `lengthUntilEnd`$(\cdot)$ are assumed to be available. A Boolean flag keepRight is used to enable the lane selection rule. Otherwise, the reference lane corridor is only changed if the current lane corridor is coming to an end, preferably right than left.

Algorithm 2 Selection of a reference lane corridor

Input ego position p, current, left and right lane corridors $\Xi_{\text{curr}}, \Xi_{\text{left}}, \Xi_{\text{right}}$, flag to activate "keep in right-most lane" rule `keepRight`
Output target lane corridor Ξ_{ref}
$d_{\text{ref}} \leftarrow v_{\text{ref}} \cdot \Delta t \cdot N$
$d_{\text{end,curr}} \leftarrow$ `lengthUntilEnd`(Ξ_{curr}, p)
if `keepRight` $\wedge$ `isValid`(Ξ_{right}) **then**
  $\Xi_{\text{ref}} \leftarrow \Xi_{\text{right}}$
else
  if `isValid`$(\Xi_{\text{right}}) \wedge (d_{\text{end,curr}} < d_{\text{ref}}) \wedge$(`lengthUntilEnd`$(\Xi_{\text{right}}, p) > d_{\text{end,curr}}$) **then**
    $\Xi_{\text{ref}} \leftarrow \Xi_{\text{right}}$
  else if `isValid`$(\Xi_{\text{left}}) \wedge (d_{\text{end,curr}} < d_{\text{ref}}) \wedge$ (`lengthUntilEnd`$(\Xi_{\text{left}}, p) > d_{\text{end,curr}}$) **then**
    $\Xi_{\text{ref}} \leftarrow \Xi_{\text{left}}$
  else $\Xi_{\text{ref}} \leftarrow \Xi_{\text{curr}}$

This rule can be seen as a soft rule, as it is included as the reference deviation in the cost function and not modeled as an optimization constraint. The vehicle will presumably obey this rule consistently if there are no slower vehicles in front. Otherwise, the costs for not following the desired speed part of the reference trajectory may exceed the costs for deviating from Ξ_{ref}.

Safe Distance The "safe distance" rule enforces to keep a safe distance to the preceding vehicle. In this work, the required safe distance is calculated in a pre-processing step before the optimization. A joint prediction is used, which is obtained by calling `ObservedWorld::Step(`Δt`)` multiple times. Here, the other agents are predicted based on their respective behavior model. For the ego agent's behavior model, a constant velocity lane following model is employed, as this resembles the velocity encoded in the reference trajectory. If the prediction model was different, for example by using an IDM, which would introduce gap keeping, it would invalidate the soft safe distance constraint in the optimization model.

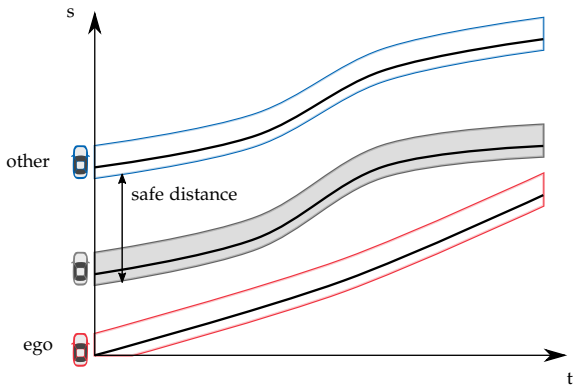


Figure 6.9.: The path-time diagram shows how the safe distance is established between the ego vehicle (red) and the preceding vehicle (blue). A ghost vehicle is introduced (gray), which serves as a convex obstacle constraint for the ego vehicle.

The safe distance of an agent i to an agent f in front is then calculated following Equation (6.33), which is an overapproximation of the checker in Section 3.4.3, if assuming equal maximum braking distances $a^f_{\text{br,max}} = a^i_{\text{br,max}}$:

$$d_{\text{safe}_1} = v^i \cdot t_{\text{react}} - \frac{(v^i)^2}{2 \cdot a^i_{\text{br,max}}} + \frac{(v^f)^2}{2 \cdot a^f_{\text{br,max}}} \tag{6.33}$$

This process is repeated along the planning horizon. The method is displayed in Figure 6.9. Based on the safe distance d_{safe_1}, an occupancy of the shape of a vehicle is calculated, which the ego vehicle is not allowed to enter. This occupancy can be interpreted as a ghost vehicle, with which the vehicle must not collide. They are modeled as dynamic convex obstacle polygons within the optimization.

The described implementation so far resembles a hard constraint. If a safe distance cannot be established, this results in a failing optimization. This work thus modifies the obstacle constraint implementation to allow for soft constraints as well. Specifically, Equation (6.22) is changed, where a slack variable ζ is used to lift the constraint of the five points (representing the vehicle shape) not to lie within the obstacle. As hard constraints are still desirable for collision avoidance with dynamic obstacles, the optimization program is formulated to differentiate between soft and hard obstacles. The constraint follows as:

$$\sum_{l \in o} \varsigma_p(k,l) - \zeta(k,l) \leq |o| - 1 \quad \text{if } o \text{ is soft} \tag{6.34a}$$

$$\sum_{l \in o} \varsigma_p(k,l) \leq |o| - 1 \quad \text{otherwise} \tag{6.34b}$$

$$\forall k \in \mathcal{K},\ \forall o \in O$$

Overtaking – No Passing on the Right Side To prevent right-side passing, an area the ego vehicle must not enter is calculated in a pre-processing step before the optimization. First, a

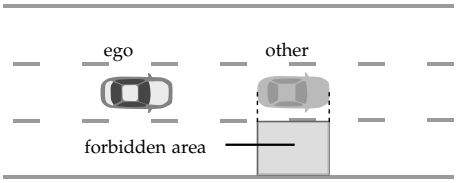


Figure 6.10.: Exemplary scene showing how the forbidden area for the "no-right-passing" rule is calculated. The ego vehicle is not allowed to enter this area.

joint prediction via `ObservedWorld::Step(`Δt`)` is applied for one time step. The IDM is used as a behavior model for the ego vehicle. Of course, it would also be possible to predict the ego vehicle based on the optimizer's solution from the previous time step. However, the correct interpretation of this rule in the optimization is not as sensitive to the longitudinal prediction model as with the "safe distance" rule. If one of the other agents is not in the right-most lane, the projected rectangle area of the agent's vehicle shape into the right lane is blocked. This concept is illustrated in Figure 6.10. The polygon is added to the optimization as a convex obstacle polygon. Repeating this along the time horizon yields a set of dynamic convex obstacle polygons, which resemble the time-space-constraints to prevent right-side passing.

6.6.8. Joint Cost Function for Reference Tracking

When formulating an MIQP problem, the objective function has to be a sum of squared or linear terms. Therefore, the absolute velocity (or acceleration) cannot be used in the objective, since with $|v| = \sqrt{(v_x^2 + v_y^2)}$, the term $(|v| - |v_{\text{ref}}|)^2$ is not quadratic. Likewise, costs on the angular velocity would result in non-quadratic terms. Moreover, as no distance to the objects is calculated in the model, such distance terms cannot be used in the cost function. Thus, the proposed formulation makes use of the position distances to the vehicle's reference to track the reference. Note that this yields a tracking of a reference trajectory and not a reference path. The trajectory reference is a sequence of x and y coordinates along the discrete time k, calculated from a reference path and reference speed in a preprocessing step.

The optimization problem's objective is to minimize the sum of individual cost functions per agent. The individual cost function term J^i of agent A^i are formulated as

$$J^i = \sum_{k \in \mathcal{K}} \Big(\omega_{\text{p}} \big(p_x^i(k) - p_{x,\text{ref}}^i(k)\big)^2 + \omega_{\text{p}} \big(p_y^i(k) - p_{y,\text{ref}}^i(k)\big)^2 + \ \omega_{\text{u}} u_x^i(k)^2 + \omega_{\text{u}} u_y^i(k)^2 \\ + \omega_\zeta \sum_{o \in O} \sum_{l \in o} \zeta^i(k,l)^2 \Big) \tag{6.35}$$

to track the reference trajectory and penalize the jerk and the violation ζ of the soft obstacle constraints. Suitable cost terms $q_\square$ balance the solution. The overall cost function is then defined as

$$J = \sum_{i \in \mathcal{A}} J^i + \omega_\xi \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{A},\ j \in \mathcal{A} \setminus i} \big(\xi_x^{ij}(k) + \xi_y^{ij}(k)\big)^2 \tag{6.36}$$

with the second term penalizing high values of the slack variables using the weighting factor ω_ξ.

6.6.9. Optimization Problem

The joint cost function Equation (6.36) for all controlled agents shall be minimized. Collecting all constraints from above, the final optimization problem can be written as

$$\begin{aligned} &\text{minimize } (6.36) \\ &\text{subject to } (6.11), (6.12), (6.13), (6.14), (6.15), (6.16), (6.17), (6.18), \\ &\qquad (6.19), (6.20), (6.21), (6.34), (6.27)(6.29)(6.30)(6.32) \end{aligned}$$

Note that Equation (6.34) is used instead of Equation (6.22), which is different than in the author's publications [57, 61] due to the implementation of the "safe distance" rule in this work.

The formulation is a standard MIQP model that can be solved with an off-the-shelf solver. Acceleration $a_\star$ and jerk $u_\star$ are bound with constant values $\underline{a}$, $\overline{a}$ and $\underline{u}$, $\overline{u}$, which are calculated from the minima and maxima of the region-dependent limits $\underline{a_\star^r}$, $\overline{a_\star^r}$ and $\underline{u_\star^r}$, $\overline{u_\star^r}$, respectively.

6.6.10. Receding Horizon Formulation

At each time step, an instance of the MIQP model is solved. The algorithm is executed in a receding horizon fashion and, therefore, only the first step of the optimized trajectory will be

executed (as the step sizes of simulation and planning are equal). The states of each agent and the current region of each agent are set as initial conditions. Adding the latter helps to avoid infeasible problems on the region boundaries. In such cases, where the initial region is possibly ambiguous due to numerical inaccuracies, the optimization is started for both possible regions.

6.7. Experiments and Results

The proposed optimization model will now be evaluated. For this, the proposed optimization program has been implemented as a behavior model within BARK. Section 6.7.1 proves, that the result of the optimization is a drivable trajectory for a non-holonomic vehicle model. This evaluation has been published in [57]. Then, the model is shown to yield a trajectory that stays within the road boundaries (Section 6.7.2) and can avoid dynamic obstacles, i.e., maneuver predictions of other agents (Section 6.7.3). Then, rule compliance of the "safe distance" rule and the "no passing on the right side" rule are shown for two selected scenarios (Section 6.7.5). Finally, Section 6.7.6 provides a benchmark of the developed model, where variants with the "safe distance" rule as a soft and hard rule are compared to a variant without any rule.

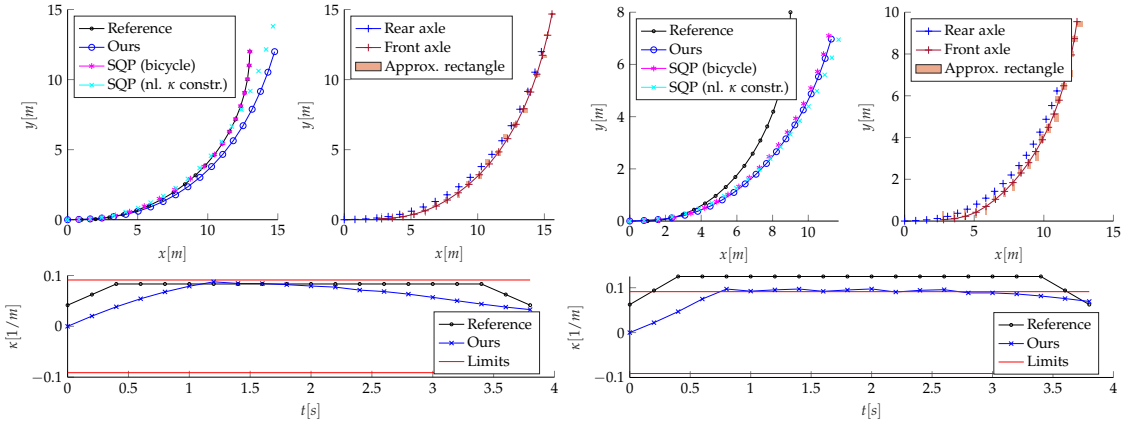
A model with 32 regions and velocity-dependent front axle approximation is used. Table A.4 summarizes the parameters used within the following experiments. CPLEX 12.10 as a commercial off-the-shelf MIQP solver is used [105]. The experiments run on a laptop with Intel Core i7-9850H (6 physical cores, 12 threads @ 2.6 GHz) and 32 GB RAM. The code of the proposed planning method has been published as open-source [106].

6.7.1. Preserving the Non-Holonomy

To show the effectiveness of the proposed constraints guaranteeing non-holonomic motions, two different reference trajectories will be optimized, both forming a circle to perform a 90-degree turn. While it is preferable to generate reference trajectories with valid curvatures only, this experiment is used to confirm that the proposed model preserves the non-holonomy. The same property is also necessary for obstacle avoidance. As baseline algorithms for comparison, two SQP-based optimization models are implemented, one using a standard bicycle model and another using the same triple integrator as this work's MIQP model but with a nonlinear curvature constraint, following Equation (6.7).

Figure 6.11a shows the results for a constant turning radius that the proposed vehicle model can follow. The optimization yields a trajectory that closely follows the reference. The solution is close to the solutions obtained from the SQP optimizer. The slight difference towards the SQP solution using the integrator model arises from the different ways to model the curvature constraint. The difference towards the SQP solution using the bicycle model arises from possibly non-equivalent weights.

The second reference in Figure 6.11b models a turning radius that is too small (the curvature of the reference exceeding the limits). As desired, the proposed model does not follow the reference and yields a trajectory that stays within the curvature bounds. As the curvature approximation polynomials are fitted for each region's mean orientation (see Section 6.5.2), the solution can



(a) Results for a kinematically feasible references.

(b) Results for a kinematically infeasible references.

Figure 6.11.: If the reference is feasible, the proposed model stays within the curvature bounds (lower) and shows as good trajectory tracking behavior as the reference implementations using an SQP optimizer does (upper left). The front axle position is always within the lower/upper bound approximation rectangles of the front axle (upper right). If the reference is infeasible, all three optimization solutions cannot track the reference despite operating at maximum curvature (graphics from [57], ©2020 IEEE)

possibly exceed the curvature bound at the region boundaries slightly. However, this could easily be mitigated using a safety margin on the curvature constraints for the fitting.

The experiment shows that the model can indeed respect the curvature constraint, even when operating on a non-feasible reference. Thus, the proposed model is the first to yield kinematically valid solutions using an MIQP-based solver that generalizes to various road geometries.

6.7.2. Staying Within the Road Boundaries

This example considers a lane ending scenario, where the ego vehicle drives at the speed of 10 m/s and needs to change lanes. The reference trajectory represents the centerline of the right lane traveling at the reference speed. Figure 6.12 shows that the optimizer finds trajectories that stay within the road boundaries. This work implemented a simple logic to select the reference lane corridor: If the vehicle is close to a lane end, the reference corridor changes, cf. Algorithm 2. At the beginning of the scenario, the right lane corridor gets selected. As the lane corridor is getting narrow, following the centerline would result in leaving the road. The optimizer can handle this successfully and yields a trajectory that stays within the road boundaries. Despite the reference trajectory not being continuous, the solution safely remains within the steering angle limits. Both the rear axle and the over-approximated front axle position stay within the shrunk road area. Note that the space in the single lane is limited. By increasing the number of regions, the approximation rectangles of the front axle would get smaller.

The experiment shows that the model can respect road boundaries and find a solution that stays within.

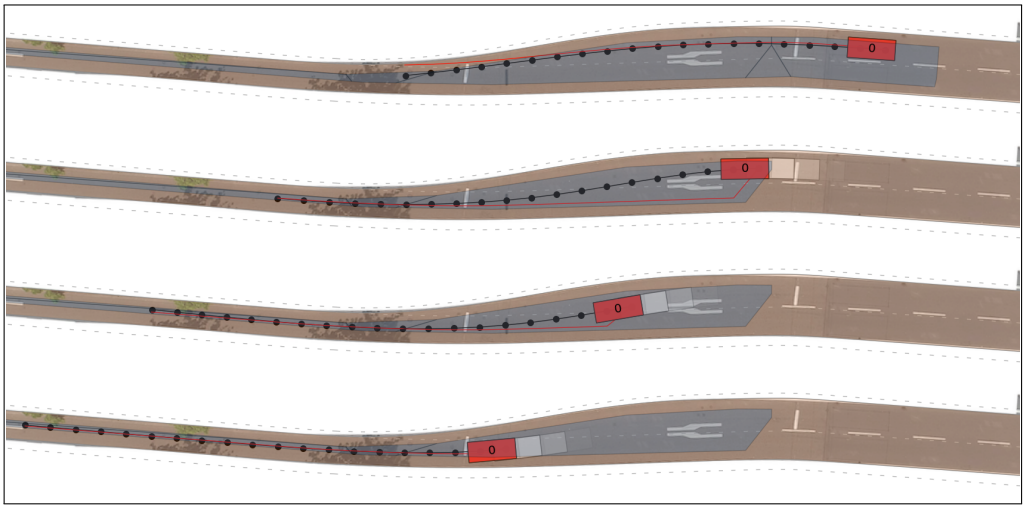


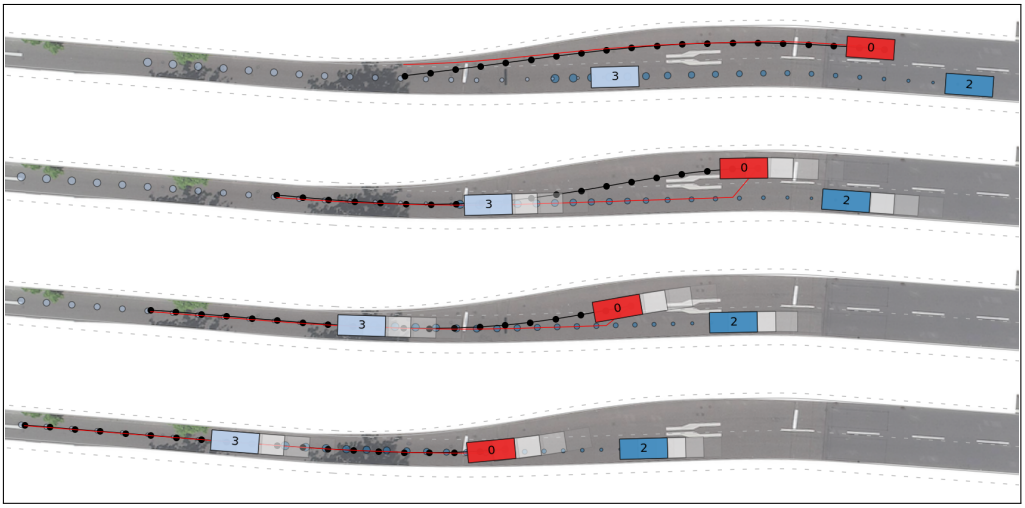
Figure 6.12.: Merging scenario with MIQP-based planner staying within the road boundaries. World configuration are shown at $t = 0s$, $t = 1.25s$, $t = 2.5s$ and $t = 3.75s$. The environment polygon is shrunk and split into several convex polygons. Both the rear axle and the over-approximated front axle position stay within the shrunk road area. The reference trajectory is shown in red. The planned trajectory is shown as , with the states displayed as dots. Past agent states are shown with increasing white transparency.

6.7.3. Avoiding Dynamic Obstacles

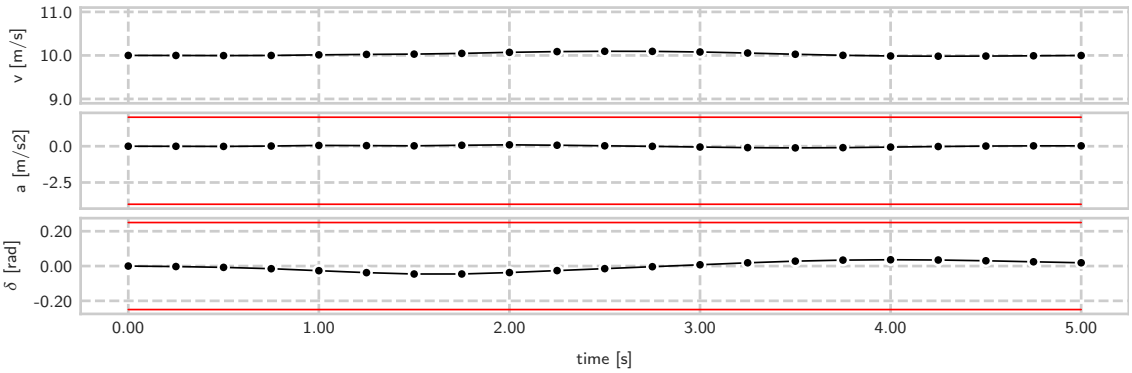
This example considers the lane ending scenario again, where the ego vehicle has a speed of 10 m/s and needs to change lanes. In Figure 6.13, two other vehicles in the left lane are traveling at 10 m/s. Deterministic predictions of those two traffic participants are obtained and included as dynamic obstacles in the optimization. The optimizer finds solutions that allow the ego vehicle to change lanes and fit in-between the two other cars. Figure 6.13b shows that the acceleration and the steering angle stays within limits. The velocity remains close to the reference speed of 10 m/s. However, as the reference speed is imposed indirectly through the non-smooth reference trajectory, the real motion differs slightly.

In Figure 6.14, the two other vehicles are traveling at 15 m/s, while the ego vehicle is still driving at 10 m/s. By predicting the two traffic participants and including them as dynamic obstacles, the optimizer realizes that following the trajectory as in the previous example would yield a crash. The ego vehicle decides to slow down and merge after the rear car has passed. The velocity slightly drops to 9.5 m/s. Still, the ego vehicle adheres to the acceleration and steering bounds.

The experiment shows that the success of the optimization program to find non-colliding solutions that respect the actor limits of the car does not depend on the reference (if given enough time to the optimizer). This is certainly desirable and different from local optimization techniques, as discussed in Section 6.2.

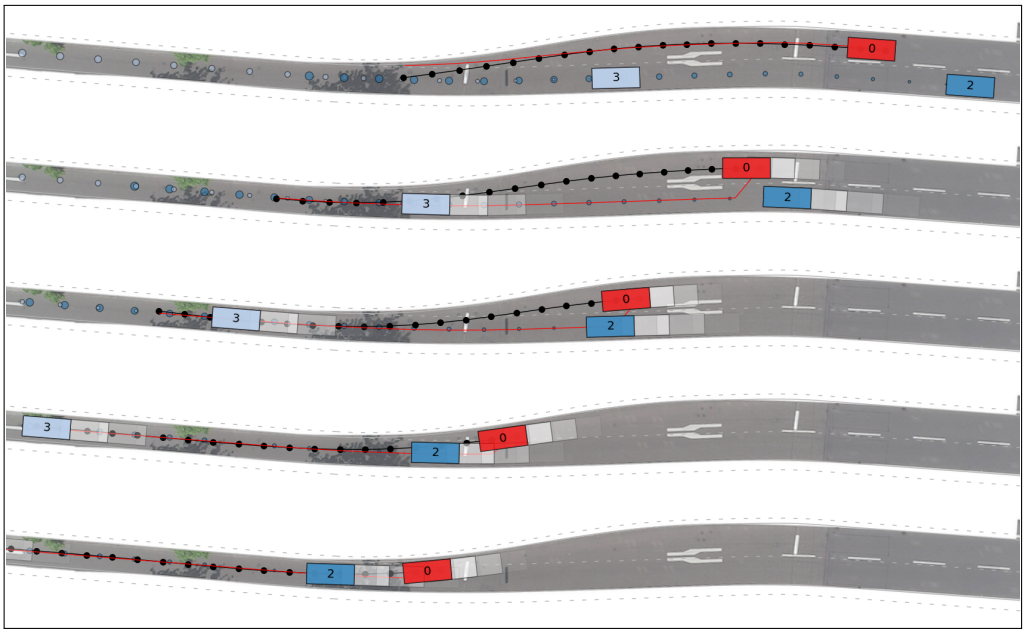


(a) World configuration at $t = 0s$, $t = 1.25s$, $t = 2.5s$ and $t = 3.75s$. The ego vehicle and its reference trajectory are shown in red. The planned trajectory is shown as ●—●—●, with the states displayed as dots. The predicted states of the other vehicles are shown as ∘ ● ●, where the increasing size indicates the predicted time progress. Past agent states are shown with increasing white transparency.

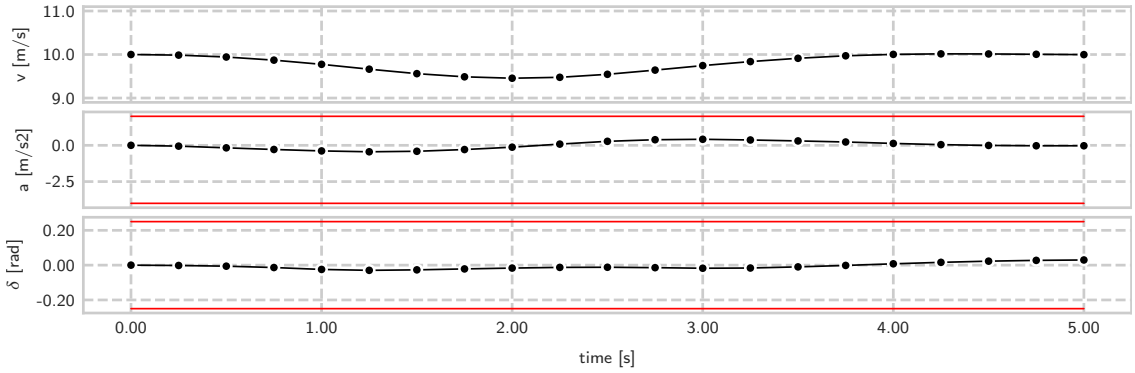


(b) The executed actions for the bicycle model and the resulting velocity are shown in blue. The respective limits are shown in red.

Figure 6.13.: Merging scenario with two dynamic obstacles on the left lane. The ego agent controlled by the MIQP-based behavior model changes lanes and fits in-between the two vehicles on the left lane.



(a) World configuration at $t = 0s$, $t = 1.25s$, $t = 2.5s$, $t = 3.75s$ and $t = 4.5s$. The ego vehicle and its reference trajectory are shown in red. The planned trajectory is shown as ●—●—●, with the states displayed as dots. The predicted states of the other vehicles are shown as ∘ ○ ●, where the increasing size indicates the predicted time progress. Past agent states are shown with increasing white transparency.



(b) The executed actions for the bicycle model and the resulting velocity are shown in blue. The respective limits are shown in red.

Figure 6.14.: Merging scenario with two dynamic obstacles on the left lane. Both other agents travel faster than the ego agent. The ego agent controlled by the MIQP-based behavior model slows down and changes lanes after both vehicles have passed.

6.7.4. Planning for Multiple Agents

In the example presented in Figure 6.15b, two vehicles are driving close by. If the ego agent would not plan cooperatively or anticipate the other agent's reactions, it could just keep its velocity. However, both agents are configured to plan using the multi-agent variant of the MIQP-based planner, i.e., $\mathcal{B}^e = \mathcal{B}^o$. The optimization aims to find a joint solution that stays within the road, respects each agent's model's constraints, and prevents collision between the agents.

Figure 6.15b displays the velocity of both agent's over time. The optimal cooperative solution's velocity is symmetric, i.e., agent 2 accelerates up to a velocity of 11 m/s, whereas agent 0 slows down to a velocity below of 9 m/s. This way, agent 0 can change lanes and merge before agent 2 instead of slowing down and possibly stopping to let agent 2 pass.

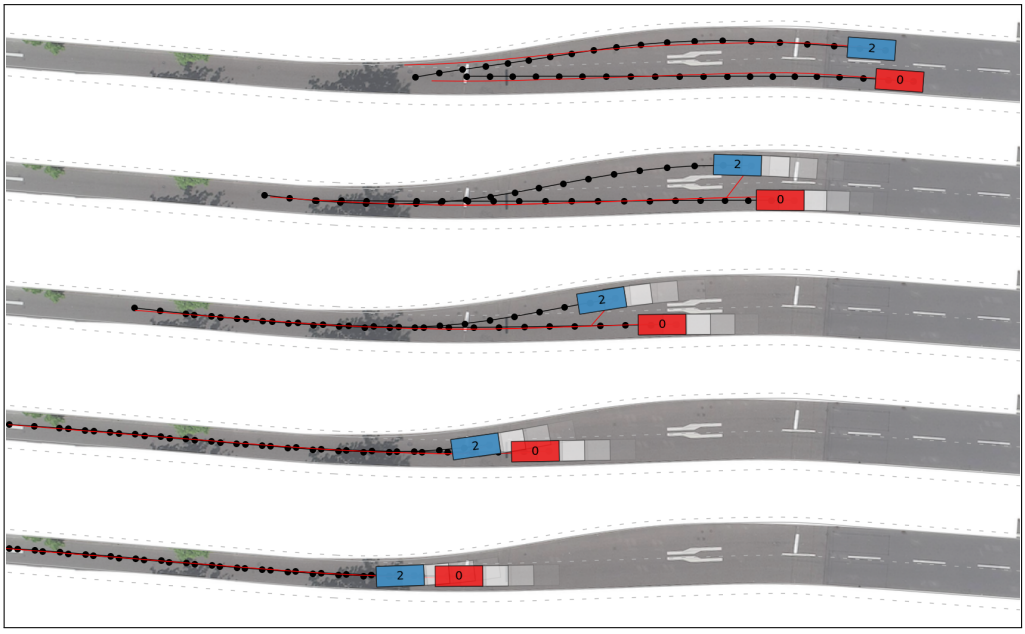
The experiment shows that the optimization program can successfully find optimal solutions for cooperative driving that satisfy the respective constraints of each agent. Going forward, the multi-agent cost function needs to be tuned to mimic the actual behavior of others, which is not necessarily fully cooperative.

6.7.5. Evaluation on Traffic Rules

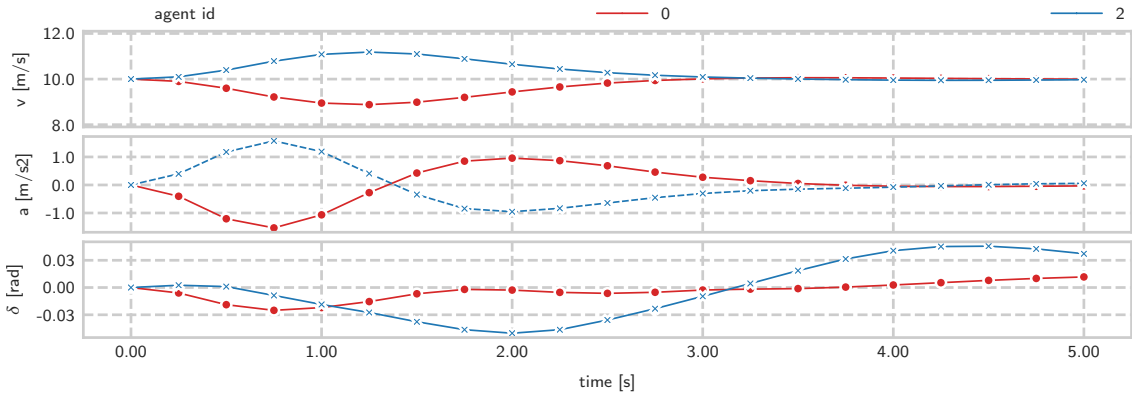
No Passing on the Right Side The following experiment demonstrates the effectiveness of the "no passing on the right side" rule. Figure 6.16 shows the experiment, which is conducted on a straight road with two lanes. The ego vehicle travels at 15 m/s and approaches a slower vehicle at 10 m/s from behind. In Figure 6.16a, the planner is modeled without the rule. Despite the reference (red) staying in the left lane, the optimizer can find a solution that avoids the blue car on the right. As only obstacle avoidance has been modeled – and no distance that would push the solution away from the obstacle – the solution comes very close to the other car, as it aims to minimize its deviation to the faster reference. Once the ego vehicle reaches the right lane, the reference switches to that lane. Eventually, the ego vehicle passes the blue car, which would violate the rule.

Figure 6.16b shows the variant with the rule. The blue crosses in the right lane indicate the forbidden occupancy due to the rule. As the ego vehicle is not allowed to enter those, there is not much speed benefit that can outweigh the steering and displacement costs from the reference. Thus, the ego vehicle stays in the left lane, closing up as close as possible to the other car.

The experiment shows that the optimization program yields solutions that satisfy the "no passing on the right side" rule. As the rule is modeled as a hard constraint, its satisfaction does not depend on the reference or cost function weights. More specifically, despite the reference favoring a continuation at 15 m/s, the vehicle slows down and respects the rule.

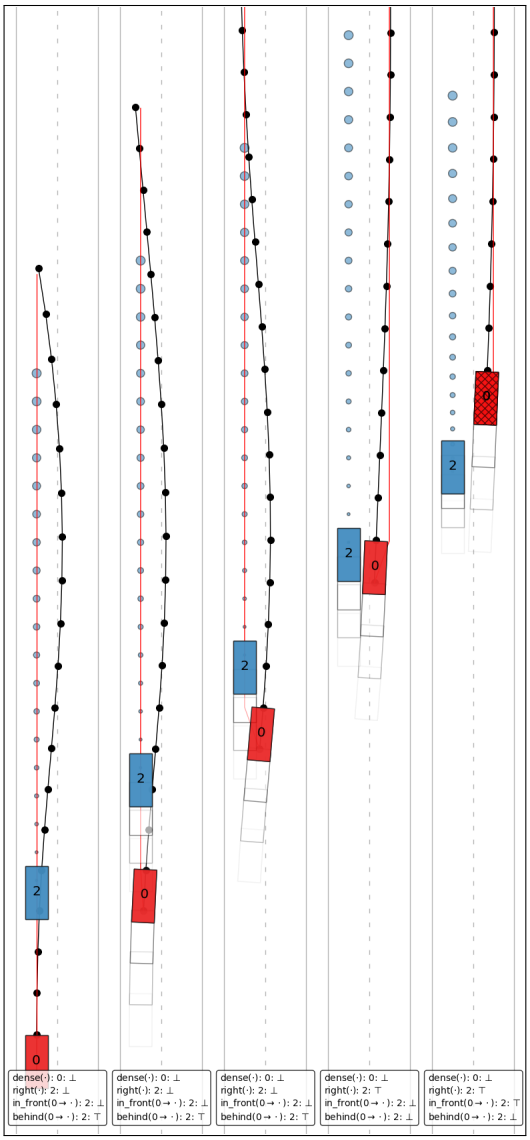


(a) World configuration at $t = 0s$, $t = 1.25s$, $t = 2.5s$, $t = 3.75s$ and $t = 4.5s$. The reference trajectories for both agents are shown in red. The planned trajectories are shown as ●—●—●, with the states displayed as dots.

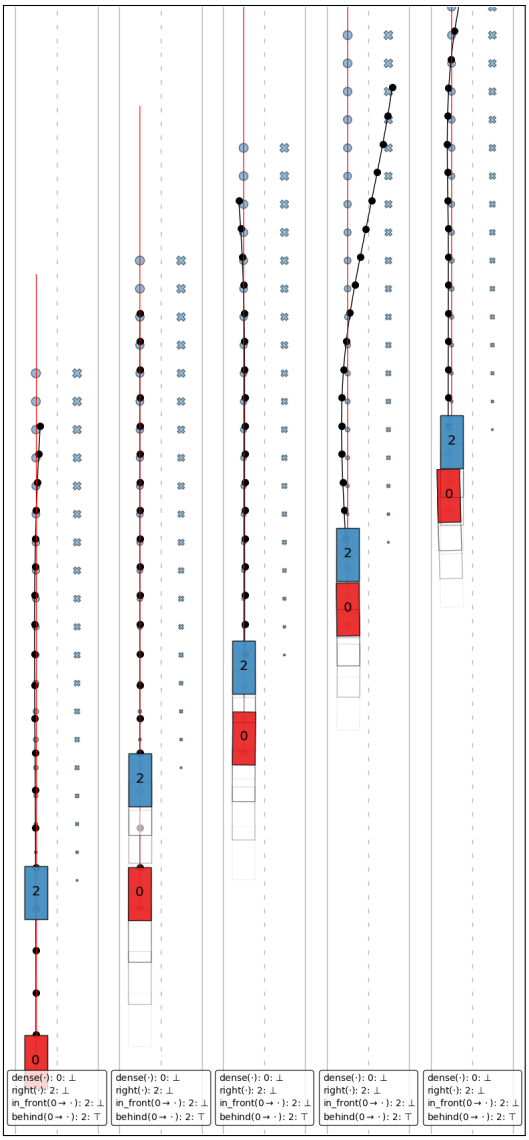


(b) The executed actions for the bicycle model and the resulting velocity are shown for both agents.

Figure 6.15.: Merging scenario with two agents close by. Both agents are controlled by the MIQP-based behavior model, which plans for both agents simultaneously.



(a) World configuration at 0 s, 1 s, 2 s, 3 s, 4 s without the rule "no passing on the right side".



(b) World configuration at 0 s, 1 s, 2 s, 3 s, 4 s with the rule "no passing on the right side". The blue crosses ✱ ✖ ✖ on the right lane indicate the projected position of the blue vehicle to the right lane, which are forbidden for the ego vehicle to collide with.

Figure 6.16.: Two-lane road scenario with a slower preceding vehicle at 10 m/s in front of the ego vehicle (red), which is traveling at 15 m/s. The ego agent is controlled by the MIQP-based behavior model. If the rule "no passing on the right side" is not modeled, the ego agent changes lanes to proceed at a faster speed. If the rule is modeled, the space right next to the other agent is forbidden, leaving the ego agent to stay behind the other vehicle.

Safe Distance To showcase the effectiveness of the safe distance modeled as a soft constraint, two vehicles shall drive on a straight road at the same speed. In Figure 6.17a, the cars are placed too close to each other so that the safe distance is initially violated. If the safe distance were modeled as a hard constraint, the optimization would fail, as the optimizer would not be able to find a solution. The scenario is simulated with and without the rule being violated. Figure 6.17b shows the calculated safe distance (calculated according to Equation (6.33)) between the vehicles and the actual distance between the vehicles. Essentially, if the distance is larger than the safe distance, no penalty will result from the soft rule constraint in the cost function. The optimizer thus seeks to find solutions that eventually lead to the rule not being violated any more. At the first three time instances, the safe distance is below the actual distance, which leads to the rule evaluator showing a violation until $k = 3$.

If the rule is not modeled, the ego vehicle keeps traveling at the same speed, merely following the reference, as the only other constraint is to stay within the road boundaries and not collide with the front vehicle. If the rule is included, the ego vehicle slows down immediately to establish a safe distance. Once this is accomplished, it continues to drive at the reference speed again. The additional constraint for the safe distance, however, increases the complexity of the optimization. Based on repeating the simulation 10 times, the solution time's mean and standard deviation (the time it takes for the optimizer to calculate the solution) are calculated and displayed in Figure 6.17b. It can be seen that the solution time is generally higher than without the rule being considered.

The experiment shows that the soft safe distance rule implementation can prevent the vehicle from driving too close and re-establish a safe distance. This is unquestionably necessary when considering other vehicles to potentially cut-in before the ego vehicle.

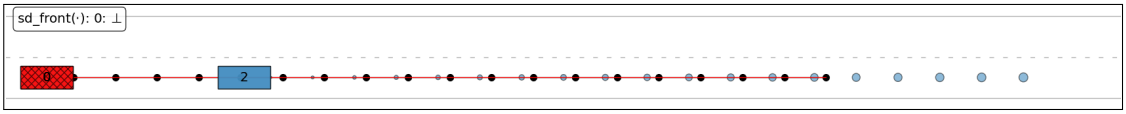
6.7.6. Benchmark

For benchmarking the Single Agent Variant of the MIQP-based planner (MIQP-SA), a comparable experimental setup as in Section 5.5 is used. The vehicles' initial starting positions are extracted from the INTERACTION dataset and simulated using the behavior model $\mathcal{B}^o$, for which the parameters are shown in Table A.2. The German merging scenario is adopted, which is the same map used throughout most of the chapter's evaluations. The same rule evaluators as in Section 5.5 are being utilized to evaluate the simulation run, but not as part of the planner. Figure 6.18 presents the resulting evaluation framework.

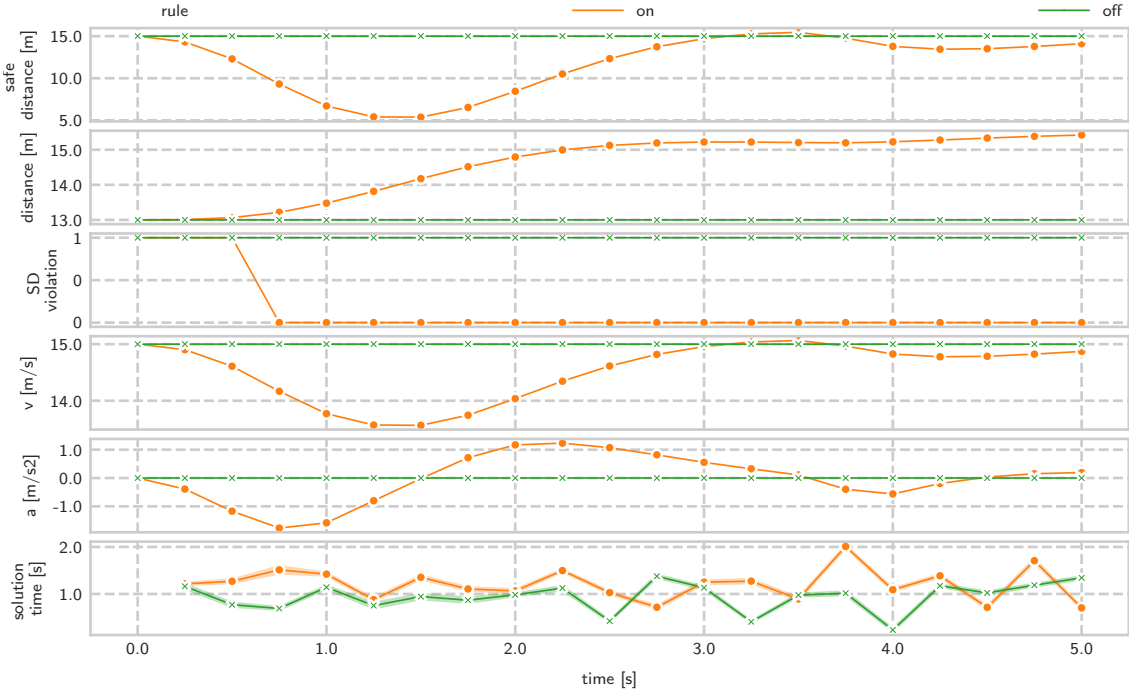
Three variants of the optimization-based planner are examined in the following:

- ***MIQP-SA*** as a single agent variant of MIQP without any rules,
- ***MIQP-SA (SD-soft)*** with the SD rule as a soft constraint and
- ***MIQP-SA (SD-hard)*** with the SD rule as a hard constraint.

The maximum solution time available to the optimizer is varied from 2 s to 100 s. If the optimizer cannot find a solution within this time, the optimizer stops, the scenario is terminated



(a) Initial world configuration at 0 s.



(b) Time series for the safe distance, the distance between the rear end of the front vehicle and the front end of the ego vehicle, the velocity, acceleration time, and the solution time of the optimizer.

Figure 6.17.: Two-lane scenario with a preceding vehicle in front of the ego vehicle (red). Both vehicles are traveling at 15 m/s. The MIQP-based behavior model controls the ego agent. Two variants are compared: The first with the rule "safe distance" modeled as a soft rule, the second without the rule. Initially, the ego vehicle is too close to the other vehicle, effectively violating the "safe distance" rule. With the rule as a soft constraint, the vehicle slows down and establishes a safe distance again.

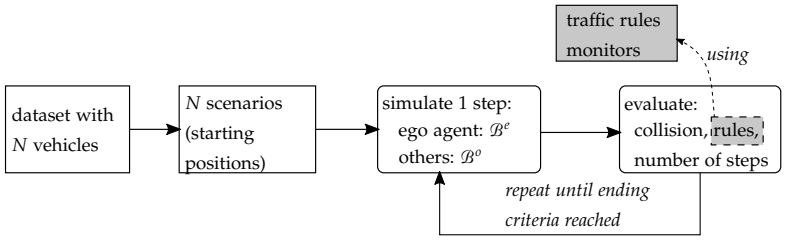


Figure 6.18.: Framework to evaluate the MIQP-based behavior of the ego agent $\mathcal{B}^e$ in a closed-loop simulation. The initial starting positions for the vehicles are taken from the dataset. The other vehicles are simulated using a behavior model $\mathcal{B}^o$. The traffic rule monitors are used to evaluate the simulation.

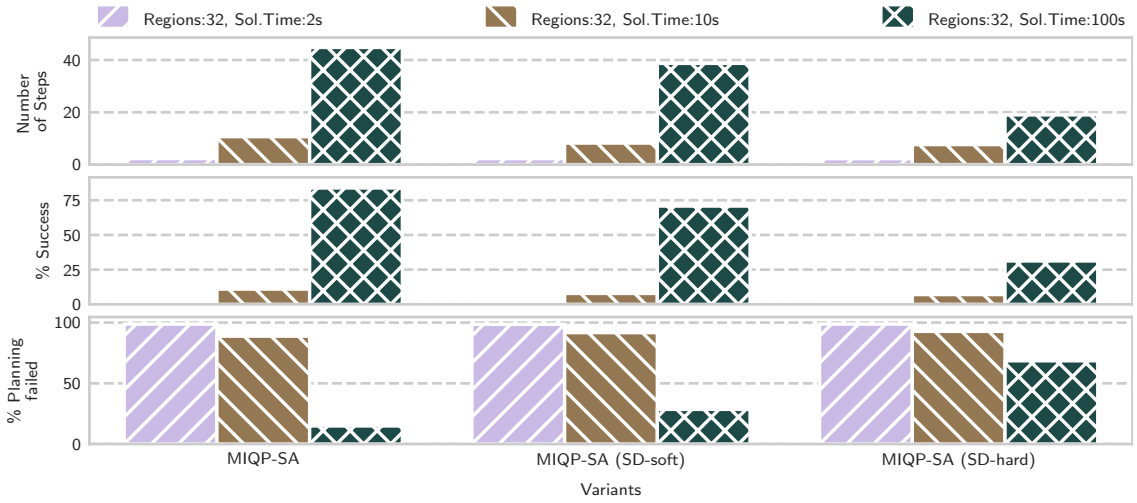


Figure 6.19.: Benchmark of the MIQP-based planner of the variants implementing no rule, the safe distance as a soft rule, and hard rule. The maximum solution time available to the optimizer is varied from 2 s to 100 s.

and labeled as *Planning failed*. If the ego agent reaches the goal region, placed at the end of the merging map, within the maximum number of steps, the scenario is marked as *Success*.

Figure 6.19 shows the obtained results. With 100 s to find a solution, *MIQP-SA* succeeds in 84 % of the scenarios. For the remaining scenarios, the planner could not find a solution within the time limit. The success rate drops to 11 % for a maximum solution time of 10 s and is at 1 % for 2 s. This can be explained by the complexity of the scenario, with sometimes up to 8 cars, which are all being processed as dynamic obstacles in the optimization. A better scheme to select only relevant cars would certainly reduce the overall planning time. Considering the rule as a soft constraint in *MIQP-SA (SD-soft)* further reduces the success rate from 84 % to 71 %, as the time limit is met more often. The *MIQP-SA (SD-hard)* variant is hardly useful at all, as it results in a lot of infeasible scenarios. The number of simulation steps increases with more successfully passed scenarios, as failing scenarios are terminating early.

Figure 6.20 shows the percentage of the safe distance violation of the MIQP-based planner during the benchmark. Only the subset of the scenarios, where each variant's scenarios terminate with success, is used for a fair comparison. *MIQP-SA* yields numerous safe distance violations. If compared to *MIQP-SA (SD-soft)*, the violations improve from 62 % to 52 %. When traveling at 10 m/s and assuming the front vehicle travels at the same velocity, this already results in a safe distance of 10 m. Such an initial distance is not available in some of the extracted scenarios. Another reason for the residual violations of *MIQP-SA (SD-soft)* is the initial determination of the preceding vehicle before each optimization start, which will cause errors when leaving the lane. Also, the assumption of constant velocity for the ego vehicle introduces errors. The former could be improved by adding the soft safe distance constraint for the vehicle on the neighboring lane at the cost of increased runtime. In contrast, the latter might be improved by reusing the planners' previous solution for the joint prediction to determine the safe distance. As expected, *MIQP-SA (SD-hard)* does not yield any rule violations when considering the subset where even

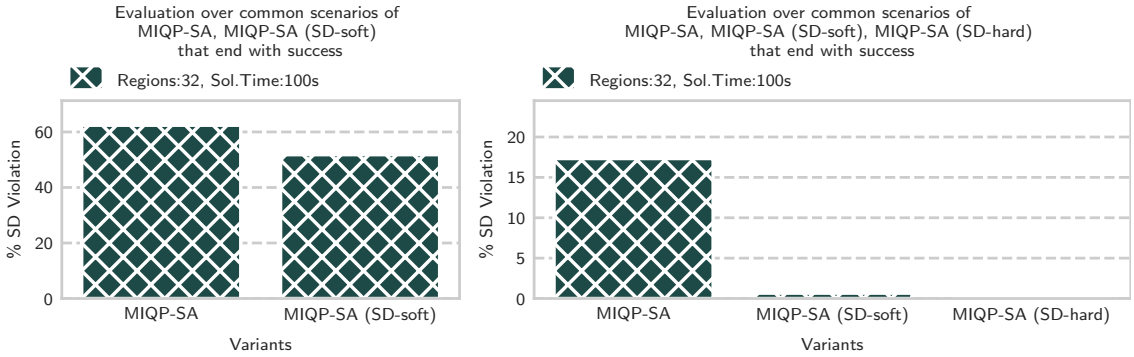


Figure 6.20.: Percentage of safe distance being violated at once per scenario from the benchmark of the MIQP-based planner of the variants implementing no rule, the safe distance as a soft rule and as a hard rule.

the hard safe distance rule can find a solution. Presumably, the scenarios which even the hard rule variant can solve are characterized by less dense traffic with enough space between the vehicles. Positively, *MIQP-SA (SD-soft)* is also able to solve nearly all of these scenarios without a violation. Less than 1 % violates the safe distance – an indicator that the penalty for softening the rule might have to be set even higher. *MIQP-SA* still has about 17 % rule violations.

Figure 6.21 shows the average planning time of the MIQP-based planner during the benchmark. Again, the subset of scenarios is used, where each variant's scenarios terminate with success. Otherwise, in complex scenarios where one variant failed early and the other was eventually successful, these time-consuming scenarios would presumably worsen the variant's average planning time. The average planning time increases from 6 s for *MIQP-SA* to 12 s for *MIQP-SA (SD-soft)* for a maximum solution time of 100 s. This shows that the additional constraints significantly slow down the optimization program.

The subset where all three variants end successfully is much smaller. In those scenarios, the average planning time only increases from 4 s (*MIQP-SA*) to 5.5 s (*MIQP-SA (SD-soft)*). Presumably, those scenarios are easier to solve, i.e., there exists enough space to the preceding

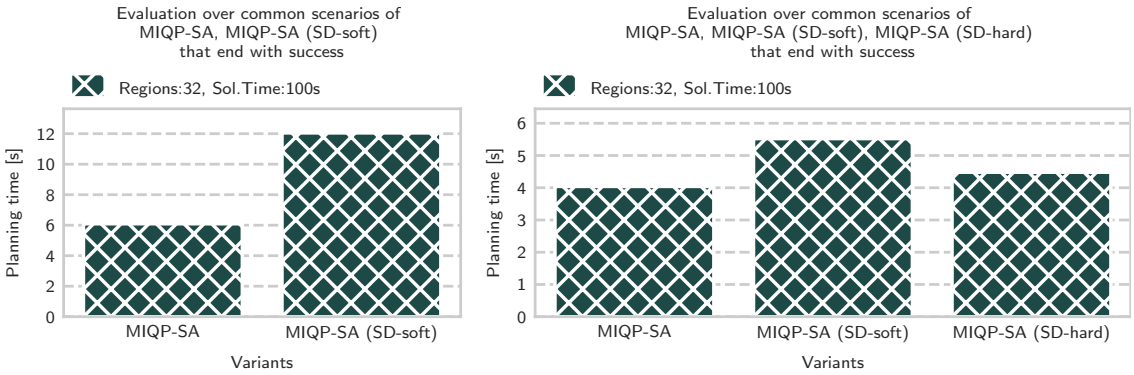


Figure 6.21.: Average planning times from the benchmark of the MIQP-based planner of the variants implementing no rule, the safe distance as a soft rule and as a hard rule.

vehicle or no preceding vehicle at all. In such scenarios, the safe distance constraint does not limit the solution very much. The runtime drops from 5.5 s for *MIQP-SA (SD-soft)* to 4.5 s for *MIQP-SA (SD-hard)*. The soft rule's decision variables are part of both constraint and cost function, which makes the problem more difficult to solve.

Still, the average planning time is well below the maximum planning time. This shows that only a fraction of the scenarios require such a long planning time and opens the door for runtime improvements by limiting the number of agents to be used as obstacle constraints.

The experiment shows that the MIQP-based planner can solve a wide range of perturbations of the merging scenario with different numbers of vehicles and placements within the map. It also shows that the modeling of the rules is working. However, it also identifies the runtime as the most significant remaining drawback. There exist various ways to improve the runtime. Firstly, the MIQP solver could be warm-started using the solution from the previous receding horizon instance [107], either by reusing and extrapolating the decision variables or by a more advanced strategy, such as warm-starting the cuts or the branch-and-bound tree. Secondly, the optimization problem could be simplified, e.g., by selecting only relevant agents as obstacles or by employing a better reference trajectory. Strategies for runtime improvements will be discussed in more detail in Section 8.3.

6.8. Conclusion

This chapter introduced a novel optimization program for the multi-agent behavior planning problem and realized multiple traffic rules as part of the formulation. The optimization program is solved using MIQP. The formulation generates correct non-holonomic motion for arbitrary road curvatures and all possible orientations of the vehicle. Using linear overapproximations of the collision shape, the formulation prevents collisions with static obstacles, dynamic obstacles, and collisions between the controlled vehicles. The introduction of slack variables to the collision constraints makes the method robust against inaccuracies of the modeled future motion of other agents. The traffic rules are expressed as linear constraints within the optimization problem or encoded in the reference. The effectiveness of this formulation of the traffic rules was evaluated on a wide set of merging scenarios. Here, the effectiveness and computational costs of modeling the safe distance rule as a soft and hard rule were compared.

The novel method obtains globally optimal solutions without any randomness of the solution. This is certainly a favorable feature for validation and certification. The solutions are kinematically valid and respect the traffic rules. Future work should study what an inaccurate prediction means for the approximated rules and the feasibility of the optimization in general.

7 Comparison of Planning Approaches for Traffic Rule Integration

The two approaches presented in Chapter 5 and Chapter 6 to model traffic rule satisfaction are dual to each other. This chapter compares those two approaches and the formalized rules that were implemented for each planning approach.

7.1. Comparison of Planning Approaches

7.1.1. Characteristics of Model

Vehicle Model and Action Space The selection of a suitable vehicle model dictates the action space. Besides that, the action space can be continuous or discrete. Continuous action space is preferred, as any discretization yields a reduced solution space, which becomes problematic in dense scenarios. The optimal control approach using MIQP relies on a triple integrator vehicle model that is being forward propagated over the planning horizon. Thus, no action space discretization is required. The ability to employ the vehicle model for MIQP comes at the downside of some approximations: The front axle is over-approximated, which reduces the solution space in dense scenarios. There, the approximation accuracy can be traded for an increase in the complexity of the problem (number of regions), which then becomes problematic for real-time usage. MCTS, on the other hand, allows the sampling of arbitrary actions and even motion primitives, such as gap keeping and lane changing. Still, the action space is discrete, and the size of a balanced search tree grows exponentially with the number of actions. Thus, similar to MIQP, the MCTS-based approach holds a trade-off between the size of the solution space and the complexity of the problem (number of actions). Still, the continuous action space of MIQP is certainly more favorable than the discretized actions of MCTS.

Interaction Modeling Interactive behavior for the MIQP-based approach can be achieved by following two different directions. Either, other agents are included in the planning (multi-agent), which works well if the costs and the reference are correct. Still, the optimization time

Table 7.1.: Comparison of the planning approaches of this work.

	Criteria	MCTS (Chapter 5)	MIQP (Chapter 6)
Characteristics of Model	Vehicle Model	arbitrary	integrator in Cartesian frame
	Vehicle Model Approx.	none required	front axle for collision checking, curvature constraint
	Action Space	discrete	continuous
	Interaction Modeling	decision tree	multi-agent planning or two-step of joint prediction and planning
	Planning horizon	$20 \times 0.5\,\mathrm{s} = 10\,\mathrm{s}$	$20 \times 0.25\,\mathrm{s} = 5\,\mathrm{s}$
	Rule Enforcement	arbitrary penalty for non-linear cost function	linear constraints or cost term for quadratic cost function
	Rule Priorities	vectorized reward with TLO	not implemented
Characteristics of Solution Method	Optimality	asymptotically optimal w.r.t. action space	optimal w.r.t. vehicle model
	Randomness	yes	no
	Completeness	complete w.r.t. action space	complete w.r.t. vehicle model
	Anytime	yes	no
	Real-time (<1 Hz)	no	no
Possible Extensions of Solution Method	Parallelization	yes	yes
	Guidance of Solution	yes (default policy, tree policy)	yes (branches, cuts)
	Warmstart	no, but history heuristics	yes (initializing decision variables)
Other	Commercical Solver	no	yes

quickly becomes intractable for real-time use cases for more than two agents. Another direction is to employ a joint prediction of the ego agent and the other agents and solve the resulting single-agent problem. The MCTS-based approach is more flexible, as the decision tree structure allows to employ any generative model within the joint planning.

Rule Enforcement and Priorities The flexible MCTS allows to evaluate the `RuleMonitor` inside the search tree and include the violation penalty in the reward function. The rules' priorities are realized by defining the reward as a vector and applying TLO to it. The optimization program solved by MIQP cannot include the rule violation penalty from a non-linear function such as the `RuleMonitor`. Thus, the rules are either encoded in the reference (lane selection) or calculated as spatiotemporal occupancies based on a joint prediction. This work chose a planning horizon of 5 s for the MIQP-based approach, which was sufficient for the rules covered, but might be increased to the 10 s horizon used for MCTS to realize rules that rely on multiple agents. Although this work's MIQP-based approach does not incorporate rule priorities, it would certainly be possible to realize a multi-objective optimization using MIQP and even apply LO to the cost function [105]. Thus, a practical implementation of the MIQP-based approach would not need to pass up this desirable rule satisfaction feature. The comparison is summarized in Table 7.1.

7.1.2. Characteristics of Solution Method

Optimality Optimality is a well-known criterion for planning algorithms [103], which describes the algorithm's ability to find the global optimum for some quality criterion subject to the given constraints, and not only a local optimum from the non-convex solution space. Optimality is

desirable not only because of being the best solution but also because of the solution's consistency in a replanning scheme. The optimal control approach combined with MIQP as a solution method can solve even non-convex problems and enable the MIQP-based planning approach to yield optimal solutions. Optimality here means optimal with respect to the employed vehicle model and the made approximations, which is not necessarily the optimal solution of the problem itself. For search-based methods such as MCTS, obtaining an optimal solution usually requires an exhaustive search over the state space. The UCT selection strategy of MCTS creates an asymmetric search tree that asymptotically converges to the optimal solution [67]. However, practically, the termination criteria for real-time applications will prohibit this. Even if UCT were to find the optimum, the optimal solution would only be optimal with respect to the selected action set. Increasing the action set improves this but increases the tree branching factor, which will reduce the ability to find the optimal solution in real-time even more.

Randomness in Solution Method The MIQP-based approach does not have any randomness in the solution method. When terminating early, however (as was done in the benchmark in Section 6.7.6), the result depends on the available computational resources during runtime. On the other hand, the MCTS variant used within this work uses random action selection and random action sampling for the default policy. The same random seed was used for all evaluations to initialize the pseudorandom number generator in this work. However, this might still yield different results across various hardware platforms.

Completeness Completeness describes the algorithm's ability to find a solution if one exists in a finite amount of time. The MIQP-based planner is capable of doing so and can thus be classified as complete. However, as the vehicle model's approximations reduce the solution space, it is only complete with respect to the vehicle model. Such completeness cannot be achieved with search-based or sampling-based methods [103]. The MCTS operates on a finite action space, meaning that even an exhaustive search would only be resolution complete.

Anytime Anytime describes the algorithm's ability to find a feasible solution quickly, which will usually be suboptimal. The solution is continually improved given more time [108]. MCTS allows to do so: The search can be stopped at any time to provide the best solution so far and be continued after that. An MIQP solver does not offer this out of the box. However, the optimization can be terminated early if a valid but sub-optimal solution was found already and restarted again afterward (by initializing all decision variables). Practically, one would prefer to start the optimization with an updated perception instead of improving the solution based on the previous perception information.

Real-Time This work focuses not on real-time execution but on fusing rule satisfaction with state-of-the-art planning methods. Thus, the code has not been optimized in terms of runtime. Consequently, none of the two presented methods runs at a frequency below 1 Hz. The following section will discuss possible extensions that the respective solution methods allow. Chapter 8 will then discuss performance improvements.

7.1.3. Possible Extensions of Solution Method

Parallelization IBM's CPLEX solver, which this work used to solve the MIQP problem, distributes the problem solving on multiple cores [105]. The experiments of this work have been executed on a laptop with 6 physical cores. The MCTS implementation used in this work only operates on one core, but MCTS is well-suited to be parallelized thanks to the independent nature of the simulation runs. This then requires incorporating and synchronizing the results from the simulations into the search tree, which has led to different parallelization techniques of MCTS. An overview is given in [67].

Guidance of Solution and Warmstart The two solution methods of this work have not been optimized towards performance. The popularity of MCTS lies in its highly flexible nature. Various possible extensions exist for MCTS to guide the solution, which usually means incorporating domain knowledge or even situational knowledge or data: For example, more realistic traffic models for the rollout phase can be learned from human data as in [109]. Another direction is to inform the selection process of the tree policy by sampling the actions from a distribution of human actions [109]. Directly warmstarting MCTS (i.e., reusing the tree from the last planning step) is difficult, as the tree's statistics will not be valid anymore. Past work has studied history heuristics, which correlate with the concepts presented to guide the solution: Using historic information to improve action selection and using historic information to improve the rollout [67].

The MIQP problem is solved using CPLEX's commercial implementation of a *branch & cut search* method [105], which is a combination of a *cutting plane method* and a *branch & bound algorithm*. It maintains a search tree of nodes, which are subproblems of the original problem. The root node is the continuous relaxation of the problem. Suppose the solution has at least one fractional variable. In that case, the solver will find cuts, i.e., new constraints to the problem that are supposed to remove areas of the feasible region containing fractional solutions. A branch yields two child nodes from a parent node, usually by altering the bounds on a variable in a dual manner for the two child nodes. Essentially, the branch & cut search method consists of performing branches and applying cuts to the nodes of the tree. Tuning the branching strategy and cuts of the solver to the specific problem can certainly reduce the solution time. Practical information on performance tuning can be found in CPLEX's documentation [105]. Warmstarting the optimization is possible, as the initial states of all decision variables can be set. For example, the previous step's solution could be forward integrated and used to initialize the problem.

To summarize, there exist various directions to accelerate the planning algorithms. Even a combination of the approaches could be explored, i.e., guiding the CPLEX solver using MCTS as in [110]. If meaningful planning time accelerations were achieved, such a combination of approaches would undoubtedly be desirable to leverage both approaches' respective strengths.

7.2. Comparison of Implemented Rules

This section compares the realized traffic rules within the two planning approaches. They are summarized in Table 7.2 and will now be discussed in detail.

Table 7.2.: Comparison of rules within planning approaches of this work. Possible implementations of not-implemented (abbreviated as n.i.) rules are depicted in blue.

Class	Rule	MCTS (Chapter 5)	MIQP (Chapter 6)
VEL	below speed limit	reward penalty	n.i. – soft constraint based on joint prediction
LS	keep in right-most lane	reward penalty	reference for cost function
	keep outside left-most lane	reward penalty	n.i. – substract lane from road polygon
OV	no passing on the right side	reward penalty	hard constraint based on joint prediction
	safe lane change	reward penalty	n.i. – soft constraint based on joint prediction
	speed adv. for overtaking	reward penalty	n.i. – reference for cost function
DIST	safe distance	reward penalty	soft constraint based on joint prediction
BOV	being overtaken	reward penalty	n.i. – soft constraint based on joint prediction
PRIO	zipper merge	reward penalty	n.i. – soft constraint based on joint prediction combined with rule state calculation in 'plan' function

The flexible nature of MCTS allows to include all rules that were formalized and studied in Chapter 3 and Chapter 4. This is not the case for MIQP, which has restrictions for the cost function and constraints. These restrictions require to approximate the rules instead of reusing the rule monitors. A subset of all identified rules was realized in Chapter 6 to demonstrate the approach. For completeness purposes, Table 7.2 shows the possible concepts for implementing the remaining rules. By performing a joint prediction first, most rules can be viewed as additional occupancies and modeled as soft or hard constraints, corresponding to soft rules (e.g., the "safe distance" rule) and hard rules (e.g., "no passing on the right side"). Essentially, choosing whether to model the rule as a soft or hard constraint comes down to identifying whether inaccurate predictions can lead to the problem becoming infeasible (such as with the "safe distance" in Section 6.6.7), which of course is not desirable and requires the use of a soft rule.

Staying below the speed limit could theoretically be realized by penalizing the exceedance in the cost function, but an accurate quadratic formulation of this is not possible in this work's MIQP program, and approximations might bear undesirable side effects. While the reference trajectory inherently encodes a reference velocity, the selected weights do not impose a speed limit at the moment, and increasing those weights to follow the trajectory too much would yield suboptimal behavior in terms of acceleration and jerk at other times. Thus, a viable implementation should realize the speed limit as an additional soft obstacle over time.

Having a significant speed advantage during overtaking could theoretically be implemented as a soft occupancy constraint. This occupancy would be behind the vehicle, imposing a certain speed. An alternative would be to represent this rule through the reference trajectory. If there is enough space in the front during overtaking, the vehicle will follow the reference's speed. Keeping outside the left-most lane could be realized by subtracting the left lane polygon from the road polygon. Thus, the vehicle would not be allowed to enter the left lane at all. Similar to the "safe distance" rule, the "safe lane change" rule and the "being overtaken" rule could be implemented as soft constraints based on a joint prediction.

The zipper merge is probably the most delicate one, as it will be the most difficult to model in the MIQP-based planner. While soft constraints can realize forbidden areas over time, this rule's temporal characteristic requires the rule state to be calculated outside of the optimization problem based on the past states. It has to be seen how much inevitable inaccuracies in the joint prediction will affect the rule.

To summarize, MIQP's ability to incorporate logical constraints and to solve non-convex problems is certainly favorable. However, modeling the rules as part of the multi-agent problem is often impossible without undesirable approximations or simplifications. Thus, this work decided to model the rules in MIQP for each agent separately based on a joint prediction. This two-step approach might work well for rules that only rely on the ego agent but will eventually depend on a very accurate joint prediction for multi-agent rules. In contrast, the MCTS-based approach presented in this work is well-equipped to model rules between multiple agents and offers various probabilistic extensions towards uncertain intentions, agent models, or transition models.

7.3. Conclusion

The MIQP-based planner has advantages over MCTS in terms of action space, optimality, completeness, and the lack of randomness. However, the flexible nature of MCTS promises to mitigate some of these shortcomings: For example, accelerating the convergence through learned heuristics will improve the runtime and should allow increasing the action space. More actions, in turn, would absorb some of the disadvantages of the discretized action space.

MCTS is certainly more favorable to incorporate additional penalties for adhering to traffic rules. There is no restriction to the function that calculates this penalty, whereas the MIQP-based approach will either require carefully designed approximations and linearizations or rely on an additional prediction.

Eventually, the choice of a suitable algorithm should depend on the ODD and the safety argument used for certification. For extensive ODDs for SAE Level 4 or Level 5 vehicles with a large set of multi-agent road rules, the MCTS-based will be preferable. However, following the characteristics discussed above, certifying an MCTS-based algorithm with its possibly learned extensions to make it real-time capable might be more complicated than certifying the MIQP-based approach. Instead, the MIQP-based approach could be deployed in limited ODDs such as retirement communities or autonomous parking garages [111]. Reduced speeds, a lower number of agents, and a compact ruleset would not expose the MIQP-based approach's current disadvantages (runtime, dependence on accurate joint prediction) and would allow leveraging its advantages (optimality, lack of randomness) sooner than later.

8 Future Work

This chapter outlines future research questions that evolve from this thesis. Section 8.1 describes further steps for the formalization of traffic rules. Functional improvements of the MCTS-based approach are given in Section 8.2. A detailed overview over runtime improvements for MCTS can be found in [109]. Then, Section 8.3 describes possible improvements for the runtime of the MIQP-based approach. Section 8.4 on the other hand describes functional extensions and the validation of the MIQP approach in the real world.

8.1. Traffic Ruleset

Extend Operational Driving Domain This work excluded particular objects from the ODD, like road signs, road markers, pedestrians and cyclists. The priority rule was not formalized, as the used dataset of this work did not offer scenarios to test and validate this rule. Thus, the formalized ruleset is not complete yet, and neither is the ODD. Future work should formalize rules for those objects and extend the ODD to more scenarios.

Establish Priorities for Ruleset This thesis motivated priorities between rules and demonstrated how to implement them in interactive planning. Future work will need to define those priorities for a complete ruleset, cf. [112]. The priorities could, for example, be learned from data, synthesized from simulation [54], or defined by regulators.

Traffic Rule Data Recorder A possible use case of a formalized ruleset aside from the planning algorithm is a rule violation data recorder similar to a flight data recorder that can be studied after a crash occurred. For both planning and recording, the rules would need to operate on perceived data with a limited horizon. So far, the rules have only been evaluated on statically recorded data. Do the formulations still work with a new and updating receding horizon, or do they have to be tweaked? Is it possible to assign blame after a collision by strictly evaluating the ruleset and checking which driver eventually violated a rule? Or will it be necessary to simulate the last seconds before a collision to identify viable legal options that would have prevented the crash?

Extension of Rules To Irrational and Forbidden Behavior of Others In this work, a methodology was presented to formalize traffic rules from legal texts, with an emphasis on not adding any subjective interpretation or exceptions that are not clearly stated. However, as the legal texts usually do not describe any scenarios specifically, they do not include exceptions for other traffic participants' misbehavior, where a rule may subsequently be dropped. Essentially, the traffic rule formulations must be made robust against such irrational or forbidden behavior of others. Future work should study this by defining the nominal behavior of other agents. The first steps following this idea have been taken in the context of set-based prediction [82]. The premises of this work's rules could be extended to exclude the irrational behavior of others.

Probabilistic Evaluation of Rules The perceived data in a car will always be noisy. This work did a study on lane matching accuracy and its implication on rule satisfaction. Slight deviations of the location might make the difference between violation or not. Further work should study whether rules could be defined in a belief state and use uncertainty information from perception frameworks. A viable direction for this could be STL, which is an extension of LTL, that replaces boolean predicates by real-valued predicates, and discrete time instances by continuous constrained time. This would increase the robustness of the formalized rules against noisy input data.

8.2. Functional Improvements for MCTS with Rules

Uncertain Intentions and Prediction Models This work's MCTS formulation did not address model uncertainties. The formalism of an MDP allows addressing any model uncertainties. For example, the traffic participant's transition from the current state to the next state can be modeled probabilistically. An extension to the MDP is the Partially Observable Markov Decision Process (POMDP). While MDPs model the state to be known, POMDPs estimate the state from an observation. MCTS can be used to approximate the solutions of such problems. For example, the intention of other traffic participants [109] or their level of cooperation [113] can be modeled to be non-observable. Bernhard et al. [114] models parameters of the other agent's model as a discrete distribution to become robust against potentially inaccurate model parameters. As a next step, these approaches need to be fused with this work's ruleset compliance within MCTS.

Other Violation Signals The LTL-based rule evaluation only distinguishes between 0 (no violation) and 1 (violation). This essentially yields sparse rewards, similar to collision or goal achievement. Reward shaping can be used to tailor the reward to the specific problem, such as distance to goal or the potential-based reward shaping in Section 5.5. Robustness semantics over STL can yield quantitative semantics about rule satisfaction between 0 and 1 [53, 115]. Applying those robustness semantics to MCTS could improve the convergence of the search.

8.3. Performance Improvements for MIQP

Section 7.1 already discussed the performance limitations of this work's MIQP-based approach. Nevertheless, the following topics should be approached to lift this burden.

Reference The optimization's cost function depends on the deviation towards the reference trajectory. In this work, this reference was very basic: The reference speed was used to interpolate future points on the reference line, disregarding the agent's current state, model constraints, or other agents. Improving the reference should make it easier for the solver to converge to a feasible solution in less time. The reference could for example come from a feasible solution from the previous time step or be obtained from a search-based method such as the MCTS-based method from this work.

Road Polygon The decomposition of a usually non-convex road polygon to multiple convex polygons is another potential bottleneck. Essentially, fewer environment polygons mean fewer constraints in the optimization and less time required to find the solution. This work employed Boost's Voronoi diagram implementation to obtain triangles and then merged the triangles to convex polygons. More sophisticated decomposition schemes or the acceptance of more imprecise approximations of the original road shape scheme could significantly reduce the solution time.

Solver Tuning Advances in solver theory have led the commercial off-the-shelf CPLEX solver to achieve a speedup of factor 90 in 14 years from 1998 to 2012 [116]. In addition to the anticipation of more advanced future solvers, the solver could be tuned to the problem at hand. Section 7.1 already proposed possible ways to improve the convergence of a generic off-the-shelf solver, such as branching and cutting. Past studies have aimed to learn this [117], which could be a viable way to introduce domain knowledge to the solver.

Fitting As discussed before, the number of regions of the polynomial fits is a tuning parameter of the proposed model. More regions yield a better approximation of the nonlinear functions. For the front axle approximation, it essentially means trading accuracy for complexity. However, there are other ways to increase the approximation accuracy: Reducing the maximum velocity limits the operational range of the fit, thus increasing accuracy for the remaining operational points. Consequently, for low-speed vehicles like people movers or cargo vehicles, the fit's inaccuracy could be decreased, enabling the usage of less than the 32 regions in the evaluations of this work, which in return would speed up the optimization.

Simplifications The proposed optimization program solves the insufficiencies of past optimization models from the literature. This work demonstrated what is possible using MIQP optimization. To apply this model in reality, the model's complexity could be simplified by reducing the number of steps while keeping the planning horizon constant. Contrary to the MCTS, this should not significantly reduce the solution space thanks to the continuous action

space. Similar to the two-stage optimization approach from Eiras et al. [99], MIQP could then serve as the coarse high-level optimization, and a fast local optimization could further improve and smooth the solution.

8.4. Functional Improvements for MIQP with Rules

The MIQP-based planning approach has only been used in simulation so far. The following steps should be taken to transfer the approach from simulation to reality.

Integration in Autonomous Driving Stack Kessler et al. [118] proposed the opportunity for research institutions to validate their planning algorithms in real-world scenarios by integrating the algorithms to the open-stack Apollo platform. The proposed MIQP-based planner should be integrated into the framework. This way, state-of-the-art components from Apollo like perception, routing or control could be re-used.

Multi-Agent Cost Function Human drivers will presumably not match the cooperative cost function of the autonomous vehicle. To address this, Kessler et al. [61] introduced a cooperation factor to the multi-agent cost function to model the other's behavior between altruistic and egoistic. Future work should focus on estimating such a cooperation factor and the cost terms online or learn them offline from situational data.

Planning Under Model Inaccuracies The real-world application will yield inaccuracies in the assumed model and inaccuracies in the perceived world state. The extra soft distance to the front-to-front collision check is one way to address such inaccuracies that could be transferred to other aspects of the optimization model. Uncertainty measures from the perception pipeline could be used to parametrize the soft constraints online. Another direction could be to integrate the MIQP-based planning approach as a nominal planning method to safe planning layers based on reachable sets [119] or RSS [49].

Rule-Compliant Prediction Most of the rules implemented in this work's MIQP-based approach rely on an initial joint prediction of all vehicles. This prediction does not consider traffic rules yet. Incorporating them into the prediction could make those predictions more accurate and allow to drop rule compliance in the subsequent planning step if another agent behaves irrationally, e.g., violates a rule.

9 Conclusion

This thesis studied how to model traffic rules within interactive behavior planning. Chapter 3 proposed a methodology to formalize traffic rules in a machine-interpretable way. This methodology involves an abstraction to premise and conclusion, which eases the iterative identification of exceptions to the rules. The rules were then formalized in temporal logic. In Chapter 4, the rules were evaluated on recorded human drives to validate the rules and study human drivers' rule compliance. Then, rule adherence was realized for two orthogonal algorithms for interactive planning. First, a rule-compliant algorithm based on MCTS was developed in Chapter 5. The integration of the rule violation monitoring into the tree allows evaluating the rules within the search efficiently. To respect the priorities in the approximate solution algorithm, TLO is used. The resulting algorithm can realize compliance with the full formalized ruleset. The evaluation studied the rule satisfaction and the ramifications of modeling the respective rules in a merging scenario. An optimal control approach has some significant advantages over tree search methods. As no generic global optimization method exists, Chapter 6 proposed a formulation based on MIQP. To facilitate this, approximations for collision-checking and non-holonomy were introduced. Based on the proposed approach, a subset of traffic rules was formalized as constrains in the MIQP-based approach.

In summary, this dissertation contains the following theoretical contributions:

T1 The introduction of a methodology to formalize traffic rules in a machine-interpretable way from legal texts to a formal language.

T2 The combination of a game-theoretic solution approximation algorithm based on sampling with an automata-based evaluation of LTL formulas. The proposed method can realize adherence to a priority-based ruleset.

T3 The introduction of a novel optimization program that plans cooperatively for multiple agents and can avoid dynamic predictions. Traffic rule adherence was implemented and demonstrated for three rules: No overtaking on the right side, keeping a safe distance, and keeping on the right.

Moreover, the practical contributions of this work are:

P1 The presentation of a consistent ruleset with nine rules for a well-defined ODD. The traffic rules can be applied and checked in simulation or replay of real traffic data.

P2 The presentation of a study to which extent human drivers follow the rules based on real traffic data.

P3 The development of the simulation and benchmarking framework BARK, which was developed as a joint effort. BARK shall serve other researchers as a platform for developing behavior planning algorithms and benchmarking them against each other to obtain a fair comparison.

P4 The extension of BARK's benchmarking to evaluate traffic rule satisfaction for arbitrary LTL formulas.

P5 The implementation of both planning algorithms in C++ and their integration into BARK. Various simulations of synthetic and recorded traffic scenarios were conducted to demonstrate the rule adherence of both algorithms.

These two planning algorithms of this work represent two promising but distinct classes of algorithms. By studying rule-compliance for both, the advantages and shortcomings were identified, compared and discussed in Chapter 7. However, more research is necessary for further functional and runtime improvements of the underlying planning algorithms before running them on a real autonomous vehicle. Likewise, the formalization focused on a subset of the full road rules and should be tested with noisy and receding horizon data on a real car.

A Appendix

A.1. Parameters for Dataset Evaluation

Table A.1 shows the rule evaluator's parameters for the INTERACTION dataset analysis on rule violations in Section 4.5.2.

Table A.1.: Parameters of the rule evaluators used for the dataset evaluation.

Label	Description	Parameter with value
near-lane-end	threshold indicating lane ending (DR_DEU_Merging_MT)	$s_{\text{rem}} = 55\,\text{m}$
near-lane-end	threshold indicating lane ending (DR_CHN_Merging_ZS (lower))	$s_{\text{rem}} = 20\,\text{m}$
near	threshold indicating proximity used for "zipper merge" rule	$d_{\text{near}} = 5\,\text{m}$
near	threshold indicating proximity used for "being overtaken" and "speed advantage" rule	$d_{\text{near}} = 3\,\text{m}$
acc	threshold indicating an acceleration	$a_{\text{lim}} = 0.5\,\text{m/s}^2$
speed-adv	threshold indicating speed advantage	$v_{\text{diff}} = 10\,\text{km/h}$
dense	threshold indicating dense traffic	$N_{\text{dense}} = 8$
dense	radius in which vehicles are counted	$R_{\text{dense}} = 20\,\text{m}$
sd-front, sd-rear	reaction time	$t_{\text{react}} = 1\,\text{s}$
sd-front, sd-rear	maximum deceleration	$a_{\text{br,max}} = -7.84\,\text{m/s}^2$
below-speed(v_{stop})	velocity threshold for "no stopping" rule	$v_{\text{stop}} = 1\,\text{m/s}$

A.2. Parameters for MOBIL-based Behavior Model

Table A.2 shows the parameters of the `BehaviorMobilRuleBased` model, which was used in the evaluations in Section 5.5 and Section 6.7.

Table A.2.: Parameters of `BehaviorMobilRuleBased` for $\mathcal{B}^o$. The lane change rules serve as an additional filter of the lanes to which MOBIL can change.

	Parameter	Unit	Value
IDM	Desired velocity	[m/s]	10
	Maximum acceleration	[m/s^2]	1.7
	Desired time headway	[s]	1.5
	Comfortable deceleration	[m/s^2]	2
	Minimum distance	[m]	2
Lane change rules	Min. rear distance	[m]	0.5
	Min. front distance	[m]	1
	Time Gap	[s]	0.5
MOBIL	Politeness Factor	[1]	0
	Safe deceleration	[m/s^2]	12
	Acceleration threshold	[m/s^2]	0.2

A.3. Parameters for MCTS-based Planner Evaluation

Table A.3 shows the parameters of the *KeepGap* directive, which was used as a discrete action for the MCTS-based planner in the evaluations in Section 5.5.

Table A.3.: Parameters of *KeepGap* directive.

	Parameter	Unit	Value
IDM	Desired velocity	[m/s]	14
	Maximum acceleration	[m/s^2]	1.7
	Desired time headway	[s]	2.5
	Comfortable deceleration	[m/s^2]	2
	Minimum distance	[m]	2

A.4. Parameters for MIQP-based Planner Evaluation

Table A.4 shows the parameters of the MIQP-based planner in the evaluations in Section 6.7.

Table A.4.: Parameters of MIQP-based behavior planner.

Description	Parameter with value
Desired Velocity	$10\,\mathrm{m/s}$
Number of Regions	32
Number of Steps	$N = 20$
Number of Neighbouring Possible Regions	3
Time step size	$0.25\,\mathrm{s}$
Maximum Solution Time	$100\,\mathrm{s}$
Constant Agent Safety Distance	$D = 3\,\mathrm{m}$
Minimum Region Change Speed	$2.0\,\mathrm{m/s}$
Wheel Base	$l = 2.7\,\mathrm{m}$
Radius of disks for vehicle shape	$R_{\mathrm{cc}} = 1\,\mathrm{m}$
Agent to Agent Slack Weight	$\omega_{\tilde{\zeta}} = 30.0$
Soft Obstacle Slack Weight	$\omega_{\zeta} = 2 \times 10^3$
Jerk Weight	$\omega_{\mathrm{u}} = 0.5$
Position Weight	$\omega_{\mathrm{p}} = 2.0$
Maximum Longitudinal Acceleration	$2\,\mathrm{m/s^2}$
Maximum Braking Acceleration	$4\,\mathrm{m/s^2}$
Maximum steering angle	$0.25\,\mathrm{rad}$

List of Figures

1.1. System architecture for a performance planner without traffic rules 10
1.2. Planning architectures overview 11
1.3. Relationship of chapters of this thesis 17

2.1. BARK's simulation loop 20
2.2. BARK's concept of an observed world 21
2.3. One iteration of the MCTS algorithm 24

3.1. Legal analysis of traffic rules 28
3.2. Premise and conclusion for speed limit rules 29
3.3. Premise and conclusion for lane selection rules 31
3.4. Premise and conclusion for overtaking rules 33
3.5. Premise and conclusion for safe distance rules 33
3.6. Premise and conclusion for being overtaken rules 34
3.7. Premise and conclusion for priority rules 35
3.8. Relational position labels. 38
3.9. Naming convention during a zipper merge 42

4.1. Distributions of traffic characteristics from the dataset 47
4.2. Zipper merge violation in DR_DEU_Merging_MT 48
4.3. Zipper merge violation in DR_CHN_Merging_ZS (lower) 49
4.4. Rule violations in the dataset 51
4.5. Rule correlations with traffic characteristics 52
4.6. Lane matching calculation 52
4.7. Lane matching study for rule violations 53

5.1. Evolution of product state for single rule monitor within MCTS planning 58
5.2. Cumulative reward comparison 59
5.3. MCTS-based planner satisfying the zipper merge rule 63

5.4. Distributions of other agents in closed-loop scenarios 64
5.5. Framework to evaluate rule satisfaction of MCTS in closed-loop simulation 64
5.6. Benchmark of the MCTS variants implementing various rule sets 65

6.1. Construction of regions for linear approximation 73
6.2. Methodology overview of piecewise linear approximation 73
6.3. Piecewise linear approximation of a nonlinear function 74
6.4. Linear vehicle model with unavailable variables 75
6.5. Sine function from velocity-based orientation estimation 76
6.6. Errors of the piecewise linear fitting 77
6.7. Deflation and inflation of environment and obstacles for collision checking 82
6.8. Approximation of the vehicle shape for the agent-to-agent collision check 83
6.9. Safe distance constraint as a ghost vehicle 88
6.10. No right passing constraint as forbidden area 89
6.11. Satisfaction of the non-holonomy constraint 92
6.12. MIQP-based planner staying within the road boundaries 93
6.13. MIQP-based planner merging in between the other vehicles 94
6.14. MIQP-based planner merging after the other vehicles 95
6.15. MIQP-based planner planning for multiple agents 97
6.16. MIQP-based planner with the “no passing on the right side” rule 98
6.17. MIQP-based planner with the “safe distance” rule 100
6.18. Framework to evaluate rule satisfaction of MIQP in closed-loop simulation 100
6.19. Benchmark of the MIQP-based planner 101
6.20. Safe distance violations from the benchmark of the MIQP-based planner 102
6.21. Average planning times from the benchmark of the MIQP-based planner 102

List of Tables

3.1. Number of other agents relevant for traffic rules 36
3.2. Rule formulations in the literature 37
3.3. Atomic propositions 39
3.4. Formalized rules 40

4.1. Analyzed scenarios from the INTERACTION dataset 46
4.2. Validity of the rules in the analyzed scenarios 46

5.1. Variants of MCTS with different rule sets 61

6.1. Comparison of continuous optimal control formulations 69
6.2. Comparison of discrete optimal control formulations 70
6.3. Absolute positional errors for the approximation 78
6.4. Decision variables 79

7.1. Comparison of planning approaches of this work 106
7.2. Comparison of rules within planning approaches of this work 109

A.1. Parameters of the rule evaluators 117
A.2. Behavior model parameters for other traffic participants 118
A.3. Parameters of gap keeping directive 118
A.4. Parameters of MIQP-based behavior planner for experiments 119

List of Algorithms

1. MCTS algorithm with UCT selection strategy 26

2. Selection of a reference lane corridor 87

Bibliography

[1] M. Montemerlo et al. “Junior: The Stanford entry in the Urban Challenge.” In: *Journal of Field Robotics* 25.9 (2008), pp. 569–597.

[2] C. Urmson et al. “Autonomous driving in urban environments: Boss and the Urban Challenge.” In: *Journal of Field Robotics* 25.8 (2008), pp. 425–466.

[3] S. Kammel et al. “Team AnnieWAY’s autonomous system for the 2007 DARPA Urban Challenge.” In: *Journal of Field Robotics* 25.9 (2008), pp. 615–639.

[4] F. Favarò, S. Eurich, and N. Nader. “Autonomous vehicles’ disengagements: Trends, triggers, and regulatory limitations.” In: *Accident Analysis and Prevention* 110 (2018), pp. 136–148.

[5] M. Wood et al. “Safety first for automated driving.” In: *Aptiv, Audi, BMW, Baidu, Continental Teves, Daimler, FCA, HERE, Infineon Technologies, Intel, Volkswagen* (2019).

[6] L. I. R. Castro et al. “Incremental sampling-based algorithm for minimum-violation motion planning.” In: *52nd IEEE Conference on Decision and Control*. 2013, pp. 3217–3224.

[7] W. Schwarting, J. Alonso-Mora, and D. Rus. “Planning and Decision-Making for Autonomous Vehicles.” In: *Annual Review of Control, Robotics, and Autonomous Systems* 1.1 (2018).

[8] P. Bender, Ö. Ş. Taş, J. Ziegler, and C. Stiller. “The combinatorial aspect of motion planning: Maneuver variants in structured environments.” In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. 2015, pp. 1386–1392.

[9] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller. “Anytime motion planning using the RRT*.” In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 1478–1483.

[10] J. Ziegler, P. Bender, T. Dang, and C. Stiller. “Trajectory planning for Bertha—A local, continuous method.” In: *2014 IEEE intelligent vehicles symposium proceedings*. 2014, pp. 450–457.

[11] M. Bojarski et al. “End to End Learning for Self-Driving Cars.” In: *pre-print* (2016). arXiv: 1604.07316.

[12] A. Kendall et al. “Learning to Drive in a Day.” In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 8248–8254.

[13] F. Altché and A. De La Fortelle. "Partitioning of the free space-time for on-road navigation of autonomous ground vehicles." In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. 2018, pp. 2126–2133.

[14] S. Sontges and M. Althoff. "Computing possible driving corridors for automated vehicles." In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 160–166.

[15] B. Gutjahr, C. Pek, L. Gröll, and M. Werling. "Recheneffiziente Trajektorienoptimierung für Fahrzeuge mittels quadratischem Programm." In: *At-Automatisierungstechnik* 64.10 (2016), pp. 786–794.

[16] K. Esterle, P. Hart, J. Bernhard, and A. Knoll. "Spatiotemporal Motion Planning with Combinatorial Reasoning for Autonomous Driving." In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 1053–1060.

[17] C. Pek and M. Althoff. "Fail-Safe Motion Planning for Online Verification of Autonomous Vehicles Using Convex Optimization." In: *IEEE Transactions on Robotics* 37.3 (2020), pp. 798–814.

[18] C. Hubmann, M. Aeberhard, and C. Stiller. "A generic driving strategy for urban environments." In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. 2016, pp. 1010–1016.

[19] C. Chen, M. Rickert, and A. Knoll. "Kinodynamic motion planning with Space-Time Exploration Guided Heuristic Search for car-like robots in dynamic environments." In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 2666–2671.

[20] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee. "Motion planning for autonomous driving with a conformal spatiotemporal lattice." In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 4889–4895.

[21] M. Werling, S. Kammel, J. Ziegler, and L. Groll. "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds." In: *The International Journal of Robotics Research* 31.3 (2012), pp. 346–359.

[22] S. Karaman and E. Frazzoli. "Incremental Sampling-based Algorithms for Optimal Motion Planning." In: *Robotics Science and Systems VI* 104.2 (2010).

[23] J. H. Jeon, S. Karaman, and E. Frazzoli. "Anytime computation of time-optimal off-road vehicle maneuvers using the RRT." In: *2011 50th IEEE Conference on Decision and Control and European Control Conference*. 2011, pp. 3276–3282.

[24] M. Bahram, A. Lawitzky, J. Friedrichs, M. Aeberhard, and D. Wollherr. "A Game-Theoretic Approach to Replanning-Aware Interactive Scene Prediction and Planning." In: *IEEE Transactions on Vehicular Technology* 65.6 (2016).

[25] D. Lenz, T. Kessler, and A. Knoll. "Tactical Cooperative Planning for Autonomous Vehicles using Monte-Carlo Tree Search." In: *2016 IEEE Intelligent Vehicles Symposium (IV)*. 2016, pp. 447–453.

[26] W. Schwarting, A. Pierson, S. Karaman, and D. Rus. "Stochastic Dynamic Games in Belief Space." In: *IEEE Transactions on Robotics* (2021).

[27] T. Kessler and A. Knoll. "Cooperative Multi-Vehicle Behavior Coordination for Autonomous Driving." In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 1953–1960.

[28] N. Evestedt, E. Ward, J. Folkesson, and D. Axehill. "Interaction aware trajectory planning for merge scenarios in congested traffic situations." In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. 2016, pp. 465–472.

[29] C. Frese and J. Beyerer. "A comparison of motion planning algorithms for cooperative collision avoidance of multiple cognitive automobiles." In: *2011 IEEE intelligent vehicles symposium (IV)*. 2011, pp. 1156–1162.

[30] J. Eilbrecht and O. Stursberg. "Cooperative driving using a hierarchy of mixed-integer programming and tracking control." In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 673–678.

[31] F. Fabiani and S. Grammatico. "Multi-vehicle automated driving as a generalized mixed-integer potential game." In: *IEEE Transactions on Intelligent Transportation Systems* 21.3 (2020), pp. 1064–1073.

[32] X. Qian, F. Altché, P. Bender, C. Stiller, and A. de La Fortelle. "Optimal trajectory planning for autonomous driving integrating logical constraints: An MIQP perspective." In: *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)*. 2016, pp. 205–210.

[33] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer. "Safe Reinforcement Learning with Scene Decomposition for Navigating Complex Urban Environments." In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 1469–1476.

[34] D. Isele, A. Nakhaei, and K. Fujimura. "Safe Reinforcement Learning on Autonomous Vehicles." In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1–6.

[35] H. Krasowski, X. Wang, and M. Althoff. "Safe Reinforcement Learning for Autonomous Lane Changing Using Set-Based Prediction." In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 2020, pp. 1–7.

[36] P. Hart and A. Knoll. "Counterfactual Policy Evaluation for Decision-Making in Autonomous Driving." In: *IROS 2020 Workshop Perception, Learning, and Control for Autonomous Agile Vehicles*. 2020.

[37] P. Hart and A. Knoll. "Graph Neural Networks and Reinforcement Learning for Behavior Generation in Semantic Environments." In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 1589–1594.

[38] C. Paxton, V. Raman, G. D. Hager, and M. Kobilarov. "Combining neural networks and tree search for task and motion planning in challenging environments." In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 6059–6066.

[39] P. Weingertner, M. Ho, A. Timofeev, S. Aubert, and G. Pita-Gil. "Monte Carlo Tree Search With Reinforcement Learning for Motion Planning." In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 2020, pp. 1–7.

[40] J. Bernhard, R. Gieselmann, K. Esterle, and A. Knoll. "Experience-Based Heuristic Search: Robust Motion Planning with Deep Q-Learning." In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 3175–3182.

[41] P. Chaudhari, T. Wongpiromsarny, and E. Frazzoli. "Incremental minimum-violation control synthesis for robots interacting with external agents." In: *2014 American Control Conference*. 2014, pp. 1761–1768.

[42] J. Talamini, A. Bartoli, A. D. Lorenzo, and E. Medvet. "On the impact of the rules on autonomous drive learning." In: *Applied Sciences (Switzerland)* 10.7 (2020).

[43] Z. Ajanovic, B. Lacevic, B. Shyrokau, M. Stolz, and M. Horn. "Search-Based Optimal Motion Planning for Automated Driving." In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 4523–4530.

[44] J. Karlsson and J. Tumova. "Intention-aware motion planning with road rules." In: *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. 2020, pp. 526–532.

[45] K. X. Cai, T. Phan-Minh, S.-J. Chung, and R. M. Murray. "Rules of the Road: Towards Safety and Liveness Guarantees for Autonomous Vehicles." In: *pre-print* (2020). arXiv: 2011.14148.

[46] B. Vanholme, D. Gruyer, B. Lusetti, S. Glaser, and S. Mammar. "Highly Automated Driving on Highways Based on Legal Safety." In: *IEEE Transactions on Intelligent Transportation Systems* 14.1 (2012), pp. 333–347.

[47] K. Esterle, V. Aravantinos, and A. Knoll. "From Specifications to Behavior: Maneuver Verification in a Semantic State Space." In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 2140–2147.

[48] J. Decastro et al. "Counterexample-Guided Safety Contracts for Autonomous Driving." In: *Workshop on the Algorithmic Foundations of Robotics (WAFR)*. 2018.

[49] S. Shalev-Shwartz, S. Shammah, and A. Shashua. "On a Formal Model of Safe and Scalable Self-driving Cars." In: *pre-print* (2018). arXiv: 1708.06374.

[50] A. Rizaldi and M. Althoff. "Formalising Traffic Rules for Accountability of Autonomous Vehicles." In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. 2015, pp. 1658–1665.

[51] A. Rizaldi, F. Immler, and M. Althoff. "A formally verified checker of the safe distance traffic rules for autonomous vehicles." In: *NASA Formal Methods Symposium*. 2016, pp. 175–190.

[52] A. Rizaldi et al. "Formalising and Monitoring Traffic Rules for Autonomous Vehicles in Isabelle/HOL." In: *International conference on integrated formal methods*. 2017, pp. 50–66.

[53] N. Arechiga. "Specifying Safety of Autonomous Vehicles in Signal Temporal Logic." In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 58–63.

[54] A. Pal, J. Philion, Y.-H. Liao, and S. Fidler. "Emergent Road Rules In Multi-Agent Driving Environments." In: *pre-print* (2021). arXiv: 2011.10753.

[55] F. Poggenhans et al. "Lanelet2: A high-definition map framework for the future of automated driving." In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 1672–1679.

[56] S. Maierhofer, A.-K. Rettinger, E. C. Mayer, and M. Althoff. "Formalization of Interstate Traffic Rules in Temporal Logic." In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 752–759.

[57] K. Esterle, T. Kessler, and A. Knoll. "Optimal Behavior Planning for Autonomous Driving: A Generic Mixed-Integer Formulation." In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 1914–1921.

[58] J. Bernhard, K. Esterle, P. Hart, and T. Kessler. "BARK: Open Behavior Benchmarking in Multi-Agent Environments." In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 6201–6208.

[59] K. Esterle, L. Gressenbuch, and A. Knoll. "Formalizing Traffic Rules for Machine Interpretability." In: *2020 IEEE 3rd Connected and Automated Vehicles Symposium (CAVS)*. 2020, pp. 1–7.

[60] K. Esterle, L. Gressenbuch, and A. Knoll. "Modeling and Testing Multi-Agent Traffic Rules within Interactive Behavior Planning." In: *IROS 2020 Workshop "Perception, Learning, and Control for Autonomous Agile Vehicles"*. 2020.

[61] T. Kessler, K. Esterle, and A. Knoll. "Linear Differential Games for Cooperative Behavior Planning of Autonomous Vehicles Using Mixed-Integer Programming." In: *2020 59th IEEE Conference on Decision and Control (CDC)*. 2020, pp. 4060–4066.

[62] M. Treiber, A. Hennecke, and D. Helbing. "Congested traffic states in empirical observations and microscopic simulations." In: *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics* 62.2 (2000), pp. 1805–1824.

[63] Jakob, Wenzel. *pybind*. Version 11. URL: https://github.com/pybind/pybind11/.

[64] Google. *Bazel*. URL: https://bazel.build/.

[65] A. Kesting, M. Treiber, and D. Helbing. "General lane-changing model MOBIL for car-following models." In: *Transportation Research Record* 1999.1 (2007), pp. 86–94.

[66] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun. "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing." In: *2007 American Control Conference*. 2007, pp. 2296–2301.

[67] C. Browne et al. "A survey of monte carlo tree search methods." In: *IEEE Transactions on Computational Intelligence and AI in games* 4.1 (2012), pp. 1–43.

[68] J. Bernhard. *MA-MCTS: A configurable library for Multi-Agent Monte Carlo Tree Search in C++*. URL: https://github.com/juloberno/mamcts/.

[69] Bundesministeriums der Justiz und für Verbraucherschutz. *Straßenverkehrs-Ordnung (StVO)*. 2013.

[70] United Nations Economic Commission for Europe. *Convention on road traffic*. United Nations Conference on Road Traffic. 1968.

[71] Collins. "Motor Road." In: *Collins English Dictionary*. URL: https://www.collinsdictionary.com/dictionary/english/motor-road/ (visited on 01/03/2021).

[72] C. Wuthishuwong and A. Traechtler. "Coordination of multiple autonomous intersections by using local neighborhood information." In: *2013 International Conference on Connected Vehicles and Expo (ICCVE)*. 2013, pp. 48–53.

[73] BGH. *Beschluss vom 03.05.1968 – 4 StR 242/67*. 1968.

[74] OLG Zweibrücken. *Beschluss vom 16.11.2009 - 1 SsRs 45/09*. 2009.

[75] A. Marshall. *Nice Minnesotans Don't Get the Cruelly Efficient Zipper Merge*. Visited on 2020–05-29. 2016. URL: https://www.wired.com/2016/06/nice-minnesotans-dont-get-cruelly-efficient-zipper-merge/.

[76] AG Düsseldorf. *Urteil vom 17.11.2006 – 41 C 3418/06*. 2006.

[77] LG München I. *Beschluss vom 07.06.2002 – 17 S 22537/01*. 2002.

[78] P. Zollner. *Reissverschlussverfahren*. Visited on 2021–02-01. 2020. URL: https://www.adac.de/verkehr/recht/verkehrsvorschriften-deutschland/reissverschlussverfahren/.

[79] R. Alur and T. A. Henzinger. "Real-time logics: complexity and expressiveness." In: *[1990] Proceedings. Fifth Annual IEEE Symposium on Logic in Computer Science*. 1990, pp. 390–401.

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

[81] K. Cho, T. Ha, G. Lee, and S. Oh. "Deep Predictive Autonomous Driving Using Multi-Agent Joint Trajectory Prediction and Traffic Rules." In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 2076–2081.

[82] M. Koschi and M. Althoff. "Set-based prediction of traffic participants considering occlusions and traffic rules." In: *IEEE Transactions on Intelligent Vehicles* 6.2 (2020), pp. 249–265.

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

[84] G. De Giacomo and M. Y. Vardi. "Linear Temporal Logic and Linear Dynamic Logic on Finite Traces." In: *Twenty-Third International Joint Conference on Artificial Intelligence*. 2013.

[85] Z. Manna and A. Pnueli. "A hierarchy of temporal properties." In: *Proceedings of the ninth annual ACM symposium on Principles of distributed computing*. 1990, pp. 377–410.

[86] H. Schluter, P. Schillinger, and M. Burger. "On the Design of Penalty Structures for Minimum-Violation LTL Motion Planning." In: *2018 IEEE Conference on Decision and Control (CDC)*. 2018, pp. 4153–4158.

[87] A. Duret-Lutz et al. "Spot 2.0 — a framework for LTL and ω-automata manipulation." In: *International Symposium on Automated Technology for Verification and Analysis*. 2016, pp. 122–129.

[88] Esterle, Klemens and Gressenbuch, Luis. *Rule Monitor*. URL: `https://github.com/bark-simulator/rule-monitoring/`.

[89] W. Zhan et al. "INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps." In: *pre-print* (2019). arXiv: `1910.03088`.

[90] J. Karlsson, C.-I. Vasile, J. Tumova, S. Karaman, and D. Rus. "Multi-vehicle motion planning for social optimal mobility-on-demand." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 7298–7305.

[91] J. Lee, A. Balakrishnan, A. Gaurav, K. Czarnecki, and S. Sedwards. "WiseMove: A Framework to Investigate Safe Deep Reinforcement Learning for Autonomous Driving." In: *International Conference on Quantitative Evaluation of Systems*. 2019, pp. 350–354.

[92] W. Wang and M. Sebag. "Multi-objective monte-carlo tree search." In: *Asian conference on machine learning*. 2012, pp. 507–522.

[93] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker. "Empirical evaluation methods for multiobjective reinforcement learning algorithms." In: *Machine Learning* 84.1 (2011), pp. 51–80.

[94] Esterle, Klemens and Gressenbuch, Luis. *Planner Rules Mcts*. URL: `https://github.com/bark-simulator/planner-rules-mcts/`.

[95] K. Kurzer, C. Zhou, and J. Marius Zollner. "Decentralized cooperative planning for automated vehicles with hierarchical Monte Carlo tree search." In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018, pp. 529–536.

[96] J. Nilsson, M. Brannstrom, J. Fredriksson, and E. Coelingh. "Longitudinal and Lateral Control for Automated Yielding Maneuvers." In: *IEEE Transactions on Intelligent Transportation Systems* 17.5 (2016), pp. 1404–1414.

[97] C. Miller, C. Pek, and M. Althoff. "Efficient Mixed-Integer Programming for Longitudinal and Lateral Motion Planning of Autonomous Vehicles." In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018, pp. 1954–1961.

[98] C. Frese. "Planung kooperativer Fahrmanöver für kognitive Automobile." PhD thesis. Karlsruhe Institute of Technology, 2012.

[99] F. Eiras, M. Hawasly, S. V. Albrecht, and S. Ramamoorthy. "A Two-Stage Optimization Approach to Safe-by-Design Planning for Autonomous Driving." In: *pre-print* (2020). arXiv: `2002.02215`.

[100] C. Burger and M. Lauer. "Cooperative Multiple Vehicle Trajectory Planning using MIQP." In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 602–607.

[101] C. Burger, T. Schneider, and M. Lauer. "Interaction aware cooperative trajectory planning for lane change maneuvers in dense traffic." In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 2020, pp. 1–8.

[102] E. M. Wolff, U. Topcu, and R. M. Murray. "Optimization-based trajectory generation with linear temporal logic specifications." In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 5319–5325.

[103] S. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[104] B. Gutjahr, L. Gröll, and M. Werling. "Lateral Vehicle Trajectory Optimization Using Constrained Linear Time-Varying MPC." In: *IEEE Transactions on Intelligent Transportation Systems* 18.6 (2016), pp. 1586–1595.

[105] IBM. *IBM ILOG CPLEX Optimization Studio CPLEX User's Manual*. 2017.

[106] Esterle, Klemens and Kessler, Tobias. *Mixed-Integer Behavior and Motion Planning*. URL: `https://github.com/bark-simulator/planner-miqp/`.

[107] T. Marcucci and R. Tedrake. "Warm Start of Mixed-Integer Programs for Model Predictive Control of Hybrid Systems." In: *IEEE Transactions on Automatic Control* (2020).

[108] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli. "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles." In: *IEEE Transactions on Intelligent Vehicles* 1.1 (2016), pp. 33–55.

[109] D. Lenz. "Motion Planning for Highly-Automated Vehicles under Uncertainties and Interactions with Human Drivers." PhD thesis. TU Munich, 2018.

[110] A. Sabharwal, H. Samulowitz, and C. Reddy. "Guiding combinatorial optimization with UCT." In: *International conference on integration of artificial intelligence (AI) and operations research (OR) techniques in constraint programming*. 2012, pp. 356–361.

[111] T. Kessler et al. "Roadgraph Generation and Free-Space Estimation in Unknown Structured Environments for Autonomous Vehicle Motion Planning." In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018.

[112] A. Censi et al. "Liability, Ethics, and Culture-Aware Behavior Specification using Rulebooks." In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 8536–8542.

[113] C. Hubmann, J. Schulz, G. Xu, D. Althoff, and C. Stiller. "A Belief State Planner for Interactive Merge Maneuvers in Congested Traffic." In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 1617–1624.

[114] J. Bernhard and A. Knoll. "Robust Stochastic Bayesian Games for Behavior Space Coverage." In: *Robotics: Science and Systems (RSS), Workshop on Interaction and Decision-Making in Autonomous-Driving*. 2020.

[115] L. Gressenbuch and M. Althoff. "Predictive Monitoring of Traffic Rules." In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021.

[116] O. Günlük. *Lecture notes ORF 523: Cutting Planes for Mixed-Integer Programming: Theory and Practice*. Visited on 2021–02-01. 2018. URL: http://www.princeton.edu/~aaa/Public/Teaching/ORF523/ORF523_Lec17_guest.pdf.

[117] M.-F. Balcan, T. Dick, T. Sandholm, and E. Vitercik. "Learning to Branch." In: *International conference on machine learning*. 2017, pp. 344–353.

[118] T. Kessler et al. "Bridging the Gap between Open Source Software and Vehicle Hardware for Autonomous Driving." In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 1612–1619.

[119] C. Pek, S. Manzinger, M. Koschi, and M. Althoff. "Using online verification to prevent autonomous vehicles from causing accidents." In: *Nature Machine Intelligence* 2.9 (2020), pp. 518–528.