Legal Compliance Checking of Autonomous Driving with Formalized Traffic Rule Exceptions

Kumar Manas[1,*], Adrian Paschke[1,2]

[1]Freie Universität Berlin, Department of Computer Science and Mathematics
[2]Fraunhofer FOKUS, Germany

Abstract

Autonomous driving (AD) systems need to obey traffic rules and sometimes execute critical maneuvers that breach existing rules to ensure safe and rule-compliant driving. To endow such legal knowledge to the AD module, we need to formalize rules considering expressiveness, decidability, scalability, and adaptability. This paper critically examines possible formalization methods and demonstrates how we can model traffic rule exceptions for compliance checking of AD models. This ensures that AD systems are safe and can identify situations requiring more complex reasoning, such as exempting ongoing rule processes. We formalize legal traffic rule exceptions hierarchically and modularly in temporal logic and ground them to sensor data for assessing model compliance. Moreover, we introduce a parsed tree structure that supports and aids neural network-based models with formal rules. We evaluate our approach by monitoring vehicle trajectories against formalized traffic rules and handling rule exceptions in various traffic scenarios. Our results show that our approach can effectively represent complex traffic rules and monitor the safety and efficiency of AD systems against legal specifications. This paper contributes to the field of legal reasoning and compliance checking by providing a methodology for formalizing traffic rules from a rule-exception perspective in a machine-readable form based on sensor data limitations.

Keywords

Traffic Rule Formalization, Formal Logic Representation, Autonomous Driving, Rule Monitoring

1. Introduction

Rule compliance is increasingly important in autonomous driving (AD). As we move closer to fully automated driving, more focus is shifting to safety and rule compliance. Traffic rules, regulations, and expert knowledge are vital in human driver decision-making. However, this knowledge is very abstract and needs to be made machine-readable. Current state-of-the-art models are benchmarked using metrics that do not prioritize safety, traffic rule compliance, and emergency situation management. Additionally, traffic rule exceptions in the presence of emergencies and exceptional situations are yet to receive critical attention from autonomous driving researchers. Furthermore, the difference in traffic laws from a legal and technical perspective in various countries poses challenges to existing AD models. Formal methods, such as linear temporal logic (LTL) and metric temporal logic (MTL), can represent traffic rules for

Workshop on Logic Programming and Legal Reasoning in conjunction with 39th International Conference on Logic Programming (ICLP), July 9–15, 2023, London, UK

*Corresponding author.
kumar.manas@fu-berlin.de (K. Manas)

specific countries in a machine-readable and concrete form. However, creating formalized rules for traffic scenarios takes time and requires domain and logic expertise. This formalization is hierarchical, and we need to map the formalized traffic rules to the sensor data information. This way, we can evaluate the system's conformity to the rule using only the available data. This limitation imposed by sensor data limits adapting the wide range of traffic rules in formalized logical form.

Representing formalized rules is challenging in AD. However, rules can be easily added or modified to account for the hierarchical nature of traffic rules, unlike the learning-based model of prediction and planning in AD, where addition and modification are costly and time-consuming due to the training process. This paper works toward the initial approach of formalizing traffic rules from a rule-exception perspective. It uses the formalized rules for model compliance checking and verification of trajectory against the specification. The trajectory is a path the vehicle follows with respect to time. Additionally, we present a parse tree representation of formalized rules for further adaptation and compatibility of formalized rules with the learning-based model. This parse tree representation can be extended for the other temporal logic variant in which the rule can be formalized.

This paper builds on existing work for rule formalization and contributes with the following:

- A systematic analysis of traffic and legal driving rule formalization in the state-of-the-art, focusing on investigating the challenges of applying formal knowledge representation methods to traffic rules.
- We formalize MTL as a parsed tree and a computation graph for extending our rules to the neural network setting.
- An initial approach of formalizing legal driving and traffic rules for rule exception and grounding the rules to the automotive sensor data for trajectory monitoring.

The rest of this paper is organized as follows. Section 2 reviews the literature on the formalization of traffic and driving rules. We also discuss the limitations of sensor data and the challenges of developing predicates for automated driving systems. In section 3, we introduce concepts used across the paper and our methodology behind the selection of formal logic. The methodology for traffic rule modeling for rule exceptions, grounding traffic rules on sensor data, and parse tree representation is discussed in Section 4. In Section 5, we show the results of our experiments and extend our proof-of-concept, then we conclude the paper with a discussion about future research directions.

2. Related Work

In the work of [1], ways of designing self-driving cars that follow the rules of the road were explored. According to [1], a formal representation of traffic rules is essential for reliable reasoning. They argue that traffic rules have a clear structure of conditions and obligations and that the priority relations among the rules determine how to resolve conflicts. Previous works [2], [3] have proposed several approaches to formalize traffic rules for automated driving systems. Based on these works, the logical formalization of traffic rules can be categorized into

two general directions: First-order logic (FOL) and temporal logic (TL). In temporal logic, there are variants in which rules can be formalized, as discussed later in this section.

Uncontrolled intersections and right-of-way traffic rules at intersections using FOL were formalized in [2]. The rules were then implemented using the Clingo ASP (Answer Set Programming) solver. This work introduces time as a parameter in the FOL formula. This means that temporal aspects are not directly integrated, making complete temporal aspect integration in the formula complex compared to temporal logic. FOL is ontologically committed to facts, objects, and relations, not time. For this reason, FOL is not our choice for traffic rule representation.

TL is an appropriate choice for domains such as AD and robotics that require planning to achieve their objectives. Planning involves imposing constraints on the timing, sequencing, and duration of events to achieve the desired outcome, and these parameters are critical for AD. Temporal logic allows operators to encode timing constraints directly. For example, in [4], the rules for safe distance and overtaking on highway driving were formalized as LTL, and initial attempts were made to analyze the legal aspects of these rules. Whereas, [5] used LTL to formalize traffic rules and employed LTL rules as a model checker to ensure that the model-generating trajectory satisfied the specifications. MTL is another variant of temporal logic, an extension of LTL that introduces constraints over a time interval, as explained further in Sec. 3.1. Computationally, MTL is more complex than LTL, but due to the timing constraints, it is more suitable for trajectory prediction and planning in the AD domain. Many traffic rules have constraints based on the time interval during which they will be enforced; therefore, MTL is the preferred choice for our use case. Some research has used MTL for the AD domain, as demonstrated by [3, 6]. Specifically, [3] focused on formalizing traffic rules for interstate driving, while [6] addressed intersection driving rules for different types of intersections, such as signalized and unregulated intersections.

Various application domains can benefit from traffic rules information. In their work, [7] utilized a pre-ordered set of traffic regulations to aid decision-making in autonomous vehicles through rule violation metrics. Formalized traffic rules were employed in [3], [8], and [9] for rule compliance monitoring of trajectories. Furthermore, formalized rules can serve as a control strategy in trajectory planners [10]. In [11], the authors integrated rules alongside reinforcement learning-based planners and utilized rule-based planners as a backup in the case of law-violation forecasting.

Another way to leverage formalized rules from a neuro-symbolic perspective is to use them as STL in neural network-based trajectory prediction models [12]. Unlike the discrete signals used in LTL and MTL, STL formulas operate on continuous real-valued signals. They are suitable for our use case as they are close to MTL in terms of constraint over the time interval of the rule but are more complex to process. STLs are a better choice for integration with neural network-based systems due to their ability to handle continuous signals. However, they are only preferred for trajectory monitoring and planning with a neural network-based approach. Thus, in this paper, we chose MTL for our work.

In their work, [13] defined the terms scenario, situation, and scenes for testing and simulation in autonomous driving. Lanelet [14] and OpenDrive [15] provided a methodology to describe traffic situations and scenarios with the road network. This paper introduces a concept for creating formal logic independent of road network format and reusable across various data formats and logic choices.

Most previous research in traffic legal rule formalization focused on rule formalization without considering rule exceptions. We define legal rule exceptions as cases where a rule does not apply or is overridden by another rule, and legal rule violations as cases where a rule is broken or disobeyed. Additionally, described previous works concentrated on manually crafting rules in a logical language, with comparatively little emphasis on their expandability to facilitate the use of rules with faster decision-making models (neural networks). In contrast, our work demonstrates and evaluates an approach for representing formalized rules for rule exceptions and representing them as a parsed tree structure, enhancing their expandability and ease of modification.

Apart from FOL and TL, in [16] and [17], deontic logic modalities, namely obligation, prohibition, and permission, were used to model the Queensland overtaking traffic rules. However, only explicit traffic norms are considered. *Explicit traffic norms* are the written rules and guidelines that specify and regulate the behavior of road users through laws, signs, and markings in contrast to *implicit*, which are picked up from experience and local driving traditions and can not be determined explicitly. Our concept extends to both implicit and explicit traffic rules. Other methods of formalizing traffic rules are using an ontology [18] and propositional logic [19]. However, in these works notion of time was not directly integrated, or they used a workaround to integrate time.

3. Preliminaries

In this section, we will explain concepts and definitions to understand the rest of the paper and some choices we made in our work for traffic rule formalization and evaluation.

3.1. Formal Logic Representation

Our primary motivation behind traffic rule formalization is compliance checking of the trajectory prediction model of AD. To achieve this, we transformed this problem into a monitoring problem, where the trajectory generated by the model is monitored against the formalized traffic rule. Such monitoring is an integral part of a safe AD experience and reduces the legal liability of the manufacturers. Furthermore, autonomous driving systems are cyber-physical systems, and their model's properties can be verified with temporal logic, as shown in the domain of robotics [20] [9] where temporal logic helps us to verify or monitor systems against the specification over time.

STL, LTL, and MTL are the options explored when formulating temporal logic rules. LTL deals with propositions and relations which change over time on a linear sequence of states. Where *state* is the current condition of vehicles and their surrounding environment and provides the necessary information for decision-making, state information of the vehicle can include position, velocity, acceleration, and orientation. At the same time, MTL is an extension of LTL by adding time constraints to the modal operators. For example, wait until *3 seconds*; here, the temporal operator *until* has the additional constraint of 3 seconds. Formalized traffic rules based on temporal logic are more flexible than models based on neural networks. Formal rules can be easily extended with new rules and regulations and modified to handle situations where the current rules are overridden by new circumstances on the road. Neural networks, on the

other hand, rely on learning from data and require lengthy and costly training to adapt to new scenarios, which limits their adoption of new rules to be integrated.

Additional considerations were placed on expressiveness and decidability based on previous research in the logic community. We will focus more on the form of the temporal logic, MTL, which extends the propositional logic with the notion of time. [21] showed that MTL could be as expressive as FOL, and in terms of decidability, MTL is decidable when we have finite timed trace length [22]. For our use case, trajectory information is sampled at a fixed rate, and we always assume it to be a finite trace, so MTL provides a good combination of expressiveness and model compliance checking ability for safety and real-time system. Additionally, MTL is sufficiently expressive due to support for past and future with precise timing constraints over the temporal operator. One possible problem with FOL is that three variable fragments of FOL is not decidable [23]. If we integrate time in the FOL formula, which is needed for many traffic situation resolutions, then formalization usually spans to three or more fragments as in [2].

In the following, we informally introduce the operators used in MTL based on [24] and our assumptions. MTL specifications considered in this paper can be written with MTL grammar:

$$\varphi ::= p \mid \neg p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2$$

Where $p \in P$ and P is a set of possible atomic propositions; φ is the task specification, φ_1 and φ_2 are MTL formula. We also have the following temporal operators that utilize time intervals.

$$\varphi ::= \mathrm{G}_t(\varphi) \mid \varphi_1 \mathrm{U}_t \varphi_2 \mid \mathrm{X}_t(\varphi) \mid \mathrm{F}_t(\varphi) \mid \mathrm{P}_t(\varphi)$$

where *G*, *U*, *X*, *F*, and *P* are temporal operators, and the subscript t represents an interval $[\mathrm{t}_1, \mathrm{t}_2]$ expressing time constraints when these operators are active.In addition, the logical connectives *negation* ($\neg$), *and* ($\wedge$), *or* ($\vee$) and *implication* ($\Rightarrow$) are used to write formulas. The future globally operator *G* specifies that φ holds within a time interval for all future states. The until operator *U* specifies that φ_1 holds until φ_2 becomes true within the time interval specified by *t*. The next state operator, *X*, specifies that φ holds for the next state within the time interval specified by *t*. The future operator, *F*, specifies that φ holds within a time interval for some future state, and the past operator, *P*, specifies that φ holds within a time interval for some past or previous state. LTL formalization is similar to MTL but without the time subscript t in temporal operators. MTL operators are defined recursively, meaning temporal operators can be composed using each other. MTL is a generalization of LTL and can satisfy more safety properties for cyber-physical systems.

Assumption 1: The time steps are uniformly spaced and discrete.

Assumption 2: MTL is interpreted over finite traces of predicate, and atomic propositions represent a Boolean statement.

Assumption 3: We restrict subscript t a time interval to be either $[t_1, t_2]$, where $0 \leq t_1 \leq t_2$, or omitted, in which case the operator applies until the end of the trace. This does not affect the generality of the results.

Predicate, alongside parameters and arguments, is an important component of MTL specifications for trajectory monitoring. The predicates in the logical formula specify traffic conditions, action, and static and dynamic features of the traffic scene and are used to make decisions about how to drive safely in our AD use case. *Action* are maneuvers performed by ego or agent, such

as overtaking or crossing. Conditions are obtained from static or dynamic features, such as *at_intersection*, *front_of*. Some dynamic predicates are event-based. Their boolean true or false evaluation happens at a specific time or time interval. For dynamic predicate, the temporal value is specified as a temporal operator in temporal logic or as a parameter in FOL. In our MTL formalization, we can have meta predicates, which combine already defined predicates and temporal operators to give the more compact meaning of complex traffic rules.

3.2. Traffic Scene Representation

In the context of traffic rules, evaluating the represented knowledge is a critical step that requires identifying the scene and situation of the traffic. Formalized rules can only be applied and evaluated when specific conditions hold in a traffic scene. Traffic scenes can be obtained from state-of-the-art autonomous driving motion prediction datasets, such as those available in [25], [26], and [27], or can be generated using scenario designer tools such as CommonRoad scenario designer [28]. These tools and datasets provide semantic information, including map data, automated vehicle trajectories, and other surrounding vehicles, known as agents. In this discussion, the term "ego" vehicle denotes autonomous vehicles whose motion is of interest to us. A "Traffic Scene" comprises stationary and moving elements, such as road segments and traffic participants, including vehicles, pedestrians, and traffic control infrastructure [13].

4. Methodology

This section explains the methodology of formal logic modeling of traffic and driving rules exception as an MTL formula. Then we ground the MTL formulas on sensor data via predicates and functions so that we can evaluate the formula. This methodology can also be extended to other forms of logic, such as FOL, STL, and LTL. The behavior of a vehicle is primarily influenced by three factors: traffic rules, static road and traffic infrastructure (e.g., lane types, traffic signs), and dynamic components (e.g., other agents' behavior, changes in traffic lights, crossing STOP line). Among these three factors, traffic rules and regulations comprise static, dynamic, and temporal features depending on the rule's requirements. Thus, we begin by exploring the traffic rule book's relevant rules for rule exceptions and explaining the rule's semantics. We then illustrate how to break down the textual traffic rules from the German traffic rulebook *Straßenverkehrsordnung* (StVO)[1] into formal logic that can be evaluated later. Furthermore, we define the semantics of the predicate.

4.1. Rule Formulation and Predicate Grounding

The formalization of traffic rules offers multiple options, as explained in Sec. 3.1. This paper aims to create robust predicates that accurately convey the meaning of traffic rules. Building upon the works of [3] and [6], we also used top-down approach to formalize traffic rules. This approach involves defining rules at a higher level of abstraction using corresponding predicates (or meta-predicate), which are then grounded to the sensor data using additional functions. In

[1] https://www.gesetze-im-internet.de/stvo_2013/

this way, additional functions maps the predicate to sensor data information, so that rules can be directly evaluated based on limited set of sensor data. For example, to determine whether a vehicle is overtaking another vehicle, we need to evaluate the formula using the predicate "overtake". However, we cannot directly obtain sensor data indicating the Boolean evaluation of the predicate "overtake". We need to ground this predicate on sensor data using functions. This grounding can be achieved using sensor data such as vehicle occupancy, ego, agent orientation, and speed comparisons along the testing time horizon.

Defining and formalizing terms and phrases from rule books like StVO is critical to traffic rule formalization. These terms can be informal and assume basic world knowledge of users. For instance, a rule like "Do not cross the solid line" is easily understandable for humans. However, for an autonomous system, terms such as "crossing" must be precisely defined and formalized in a machine-readable format.

We formalize traffic rules using MTL formulas comprising predicates, temporal operators, and logical connectives. These formulas also use parameters and arguments. Some predicates usually consist of facts and events directly extracted from sensor data so that they can be evaluated directly. On the other hand, most predicates need to be expanded using additional functions to make them evaluable. Formalized traffic rules need to be grounded in the traffic scene to evaluate and apply these rules. We explain this concept and rule exception use for the trajectory monitoring from German legal law, StVO, and Vienna Convention on Road Traffic (VCoRT)[2] and later based on this we show our MTL rule evaluation result in Sec. 5. Based on StVO § 37(1), § 37(2)(1), and traffic sign 294, which provides guidelines for stopping at the intersection in the presence of red traffic light and these legal guidelines can be summarised as: *"if the vehicle is at an intersection and the traffic light is red, then the vehicle must stop before the stop line"*. In MTL, this legal rule can be formalized as in Eq. 1:

$$G\Big(\big(at\_intersection(ego) \wedge at\_traffic\_light(ego, red)\big) \Rightarrow \big(in\_front(stopLine, ego) \wedge stop(ego)\big)\Big) \quad (1)$$

Eq. 1 shows that traffic rules usually apply to the driving scenario. However, in some driving scenarios, rules need to be violated or suspended due to new situations arising during driving. One such permitted driving rule violation is giving way to priority vehicles such as ambulances and other emergency services vehicles operating with indicating light or sound (VCoRT Article.34, StVO §35(5)(a)). These situations are underrepresented or absent in the datasets because they involve rule exceptions or rule suspensions for the time being in favor of another traffic rule. To check the compliance of the AD model in these situations, our approach focusing on the formalization of rule exceptions can be helpful in determining the fail-safe and secure behavior of the AD model. Now we can modify the Eq. 1 based on rule exception as follows:

$$G\Big(\big(at\_intersection(ego) \wedge at\_traffic\_light(ego, red)\big) \Rightarrow \big(in\_front(stopLine, ego) \wedge stop(ego)\big) \vee \big(in\_behind(priorityVehicle, ego) \wedge cross\_stopLine(ego)\big)\Big) \quad (2)$$

Eq. 2 shows that by introducing logical OR in the implication part, even if the trajectory generated by the model is violating the initial rule but due to the presence of rule exception, the

[2] https://unece.org/DAM/trans/conventn/crt1968e.pdf

trajectory will satisfy the specification for the compliance checking. To summarize, we assessed the rule compliance of the AD trajectory model by using predicate evaluation. We did not need to rank the rules hierarchically because we only used them to monitor the trajectory generation module, not to generate trajectories directly. Also, it can be seen that violation or crossing *stopLine* will only take place when the priority vehicle is behind the ego. For brevity, we skipped additional details depending on the evaluation setup, where we might need to add a predicate to evaluate if the blue light or siren is on, as this indicates an emergency operation. More details about the evaluation of the predicate are in Sec. 4.2. Similarly, we can formulate formal rule exceptions on top of the existing rule, which allows us to maintain the hierarchical nature of decision-making in the formalized rule. Similarly, we can define rules for other exceptions, such as not crossing crosswalks in the presence of pedestrians and green traffic lights, but we skipped them for simplicity purposes.

4.2. Predicates and Functions

We introduce the predicates and functions needed to complete the formalization of legal traffic rules from the exception point of view. For readers, we reference the MTL rule as shown in Eq. 2. We acknowledge that different ways of formalizing the rules and predicates may have advantages and disadvantages. Creating a reusable predicate with a modular structure can facilitate the formalization process and enhance the real-time performance of the trajectory monitoring system. One predicate evaluation can be reused for multiple rules to determine their satisfaction.

at_intersection and *at_traffic_light* predicate can be directly evaluated based on the current lane occupancy of vehicle and sensor data information. Other predicates can be defined as follows:

$in\_front(x, y) \iff \Big(distance(x, y) \leq d_{th} \wedge \big(lane\_id(x) = lane\_id(y)\big)\Big)$

d_{th} is the user-defined parameter, and *1.0* meter in our case, and distance refers to the Euclidean distance based on the chosen coordinate system. Predicate *in_behind* can be defined similarly.

$stop(x) \iff -v_{th} \leq velocity(x) \leq v_{th}$

Here, v_{th} is the user-defined threshold velocity parameter, and it is 0.1 m/s.

$cross\_stopLine(x) \iff position(x) > position(stopLine)$

Position can be directly extracted from querying sensor data of ego and stopLine. Here, we want to show how the predicate formulation can be optimized. Instead of creating predicate *cross_stopLine*, we can have a more modular and reusable predicate such as *crossing*, which takes two parameters instead of one in *cross_stopLine*, and this predicate can be used to define any road or traffic infrastructure crossing. By Boolean predicate evaluation, we can evaluate the legal compliance of the AD model. We have seen how to manually create and map a formalized rule to the sensor data information. It can be time-consuming, requiring selecting a predicate and creating functions, parameters, and arguments. Next, we explain the parse tree representation of MTL formulas and their creation.

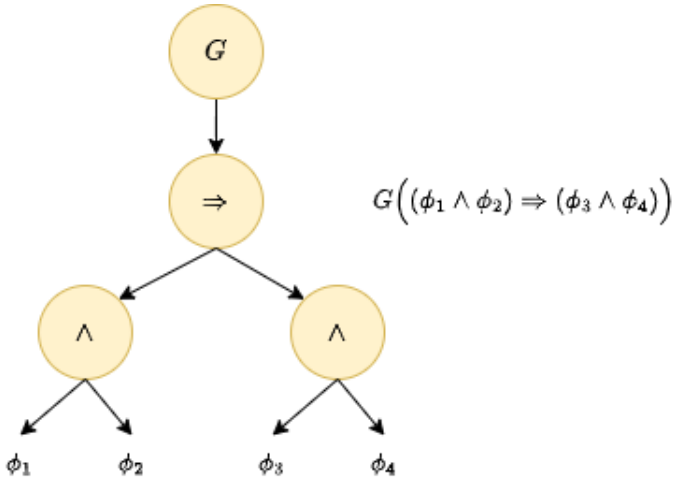


Figure 1: Parse tree representation of Stopping at intersection rule (Eq. 1). MTL rule of Eq. 1 is simplified by replacing predicates and their entities with $\phi_1 \dots \phi_4$.

4.3. Parse Tree Representation and Rules

We use a parse tree representation of the MTL formula. Many traffic rules are hierarchical and context-dependent, and their application over the traffic scene is based on evaluating lower-hierarchy context evaluation. Context is also crucial in formalizing traffic rules, as it serves as background information for evaluating vehicle behavior. As in our earlier introduced rule exception in Sec. 4.1, in the presence of context such as priority vehicle in the current driving lane. We may need to modify the rule to account for this context introduction in the traffic rule.

Furthermore, a parsed tree representation allows us to introduce new concepts, context, or rules in the existing rule representation. Such representation is suitable for MTL, as MTL formulas are defined recursively. The node in the parsed tree represents the operation, and the leaves indicate the predicate. For trajectory monitoring purposes, the parsed tree is less critical apart from visualization and easier modification rules. However, it is better leveraged when we want our MTL or corresponding STL version of the formula to be used with neural network-based models.

Definition 1: Let PT be the parse tree for MTL formula ϕ and operation over this parse tree be denoted as Define $\mathcal{O}_\phi = \{\varphi_1, \varphi_2, \dots \varphi_t\}$ as the post-order traversal of PT.

The more complex and hierarchical the rule, the deeper the tree structure. Higher abstractions of rules can be represented by combinations of predicates (meta-predicates) and functions. Fig.1 illustrates a parse tree with post-order traversal for an MTL formula introduced in Eq. 1. The order of operation of this PT is $\mathcal{O}_\phi = \{\varphi_1, \varphi_2, \wedge, \varphi_3, \varphi_4, \wedge, \Rightarrow, G\}$. Representing rules as a parse tree provides us following advantages:

- **Integration with Neural Network:** We can transform a parsed tree network into a computation graph [29], which is suitable for backpropagation because it is differentiable. Backpropagation enables us to combine formalized rules with a neural network model, enhancing the neural network's fast decision-making with rule-based reasoning, such as

handling rule exceptions. Formal rules in this form can improve the quality and safety of trajectory planning. However, more research is needed to integrate formal rules into sub-symbolic networks, creating a neuro-symbolic network. We can covert the MTL into STL formalization and use the formal logic in a non-Boolean way to determine the robustness of the degree of satisfaction of rules instead of complete satisfaction as needed in our MTL formalization.

- **Hierarchy and Recursive**: Parse tree enables us to represent traffic rules in a hierarchical tree structure, reflecting traffic rules' hierarchical nature. As a result, we can manage and prioritize rules more efficiently for complex driving scenarios and recursively combine them to create more complex rules. This also allows us to create predicate creations efficiently by making them reusable.

Py-metric-temporal-logic[3] and LTLf2DFA[4] provides a tool for parsing temporal logic formulas based on the MTL grammar introduced in Sec. 3.1. This parsing helps us create a tree structure by providing predicates, temporal operators, logical connectives, and parameters from the MTL formula. These parsed pieces of information constitute our tree structure's nodes (including the leaf nodes). Once we have all the values, we can connect the nodes based on the relationships extracted from the formula, which is then used as discussed above.

5. Experimental Evaluation

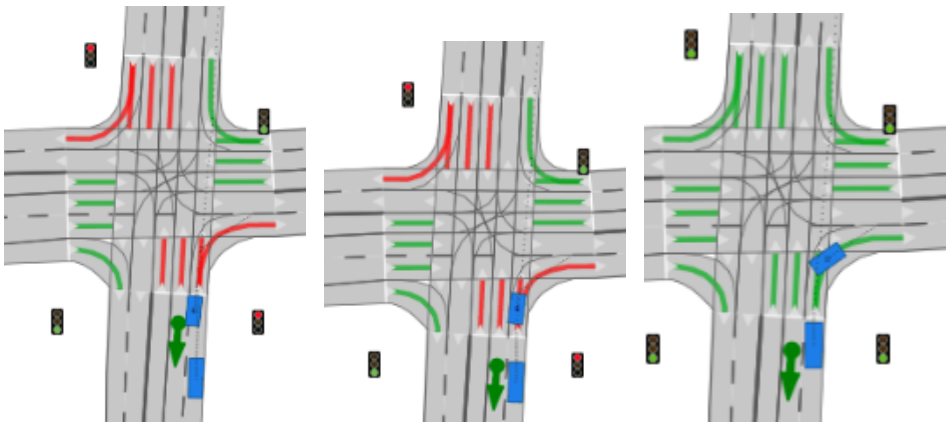
To evaluate our rule exception and trajectory monitoring, we extracted and modified the traffic scenario from commonroad [28]. Finding scenarios where rules are violated is difficult in most publicly available datasets. However, we modified some handcrafted scenarios available in the commonroad and performed trajectory monitoring against specifications written in MTL. For our manual modification, we changed the position coordinate of the vehicle with respect to the time in the XML file having the trajectory information. Each scenario consists of the start and end goal points of the vehicles, the time limit to reach the goal, and the semantic information of the environment. The scenarios are stored in a hierarchical XML file that simplifies the data processing compared to working with raw automotive sensor data.

The trajectory monitoring and model compliance checking pipeline begins by extracting trajectory information such as position, velocity, orientation, and acceleration, along with map information that provides semantic details such as lane markings, traffic light information, and lane speed limits. Once we have this information, we can evaluate the conformity of the vehicle trajectory against our MTL traffic rules using monitors for each rule which we are evaluating. In this way, if the trajectory provided to us conforms to the MTL formalization, it implies the model complies with legal traffic rules and vice-versa. In this work, we assume trajectory is provided to us by the underlying model or provided directly as a recorded scenario at each pre-defined time step.

Fig. 5 shows one of the evaluated traffic scenes. As seen in Fig. 5, here, traffic rule exception regarding crossing *stopLine* (Eq. 2) is successfully performed, as ego gives way to the emergency

[3] https://github.com/mvcisback/py-metric-temporal-logic
[4] https://github.com/whitemech/LTLf2DFA

vehicle. To create a negative sample, we modified the scenario file, such as shown in Fig. 5, so that the ego vehicle stays behind the stop line even in the presence of the priority vehicle, and we kept the trajectory of the priority vehicle as it is. Then in such a scenario, collision will happen as ego did not cross the stop line, which signifies the model's failure against the specification. Such modification will be termed a rule violation scene as ego cars need to give way to priority vehicles in the same lane.



(a) (b) (c)

Figure 2: Example of traffic scene evaluated for trajectory monitoring. Traffic scene for three different time steps in increasing order. Traffic lights and road networks are shown (better in color version and zoom). Two vehicles are in the scene. a) Ego car is behind the stop line, the priority vehicle (bigger in size) is behind the ego, and the traffic light is red. d) Ego crosses the stop line due to the priority vehicle, and the traffic light is still red. c) Ego crossed, and the traffic light is green now. Adapted from [28].

We tested the rule complaince of the generated trajectory with extracted and handcrafted scenarios (with modifications to introduce exception situations). We used two scenario files for every situation, one involving a trajectory accommodating the rule exception situation (directly extracted from commonroad) and another following the trajectory without accounting for the rule exception. For the scenario not following the rule exception, we modified the scenarios obtained from the commonroad by changing the trajectory of the ego vehicle and keeping other parameters intact as before. Since in the recorded dataset, we don't have accident or violation scenarios. We tested two exception situations: crossing the solid line as defined in Eq. 1 and Eq. 2 and stopping at the pedestrian crossing even if the traffic light is green but a pedestrian is present. Our method provided a trace in the form of Boolean evaluation against specification at every one-second interval for each predicate in the MTL rule. This is achieved using monitors for each rule which monitor the trajectory at specified time intervals. This time interval can be changed to the user's requirement. Based on predicate trace, we can infer the rule over the trajectory at the discrete time and which gives model rule compliance ability and failure time steps based on which model can be improved further based on diagnosis information provided by trace and failure.

Our current approach relies on recorded data, where the detection of exception situations

is assumed to be correct, which may not always be true. Therefore, we need to evaluate our approach on a driving simulator further. We also measured the time required for each rule evaluation in the monitoring process and found that it takes approx 2.7 seconds on average. This may be too long for real-time online monitoring, which involves perception, prediction, and planning components apart from rule evaluation, so we suggest invoking the rule exception monitoring only when certain conditions are met, such as the presence of priority vehicles or pedestrians, which can reduce the computation time for our use case.

6. Conclusion and Future Work

This paper analyzes the state-of-the-art methods for formalizing traffic rules in formal logic and identifies the challenges and limitations of existing approaches. Based on the trade-off of expressivity and temporal-time constraints, we explain traffic rules' metric temporal logic formulation. We propose a methodology for formalizing traffic rules hierarchically and modularly using predicates grounded on limited sensor data extracted from automated driving sensors. We explain how to break down the textual traffic rules into formal logic that can be evaluated using parse tree structures. In the future, we want to research further how to automate a human-in-loop system so that a meta predicate can be relatively grounded to the simpler predicate and use rules in a neuro-symbolic for decision making. Furthermore, Semantic role labeling [30] can extract predicates and arguments from natural language traffic rules by using the context information in the text. Usually, this is achieved manually, requiring expertise in logic and domain expertise. Our proposed methodology is promising for formalizing traffic rules in formal logic. It addresses the challenges and limitations of existing approaches and is flexible enough to represent complex rules. Our approach will be helpful in the development of safe and efficient autonomous vehicles. By understanding the situation and the beliefs of automated vehicles and other traffic participants, we can gain insight into how to choose scenarios and rules for different traffic scenes. For such understanding, epistemic reasoning over the driving situation might be needed.

Acknowledgments

This work has been partially funded by the German Federal Ministry for Economic Affairs and Climate Action within the project "KI Wissen". We sincerely thank Dr. Stefan Zwicklbauer for his constant support and mentorship throughout the work.

References

[1] H. Prakken, On the problem of making autonomous vehicles conform to traffic law, Artificial Intelligence and Law 25 (2017) 341–363. URL: https://doi.org/10.1007/s10506-017-9210-0. doi:10.1007/s10506-017-9210-0.

[2] A. Karimi, P. S. Duggirala, Formalizing traffic rules for uncontrolled intersections, in: 2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS), IEEE,

Sydney, Australia, 2020, pp. 41–50. URL: https://ieeexplore.ieee.org/document/9096003/. doi:10.1109/ICCPS48487.2020.00012.

[3] S. Maierhofer, A.-K. Rettinger, E. C. Mayer, M. Althoff, Formalization of Interstate Traffic Rules in Temporal Logic, in: 2020 IEEE Intelligent Vehicles Symposium (IV), 2020, pp. 752–759. doi:10.1109/IV47402.2020.9304549, iSSN: 2642-7214.

[4] A. Rizaldi, J. Keinholz, M. Huber, J. Feldle, F. Immler, M. Althoff, E. Hilgendorf, T. Nipkow, Formalising and Monitoring Traffic Rules for Autonomous Vehicles in Isabelle/HOL, in: N. Polikarpova, S. Schneider (Eds.), Integrated Formal Methods, volume 10510, Springer International Publishing, Cham, 2017, pp. 50–66. URL: http://link.springer.com/10.1007/978-3-319-66845-1_4. doi:10.1007/978-3-319-66845-1_4, series Title: Lecture Notes in Computer Science.

[5] K. Esterle, V. Aravantinos, A. Knoll, From Specifications to Behavior: Maneuver Verification in a Semantic State Space, in: 2019 IEEE Intelligent Vehicles Symposium (IV), 2019, pp. 2140–2147. doi:10.1109/IVS.2019.8814241, iSSN: 2642-7214.

[6] S. Maierhofer, P. Moosbrugger, M. Althoff, Formalization of Intersection Traffic Rules in Temporal Logic, 2022. doi:10.1109/IV51971.2022.9827153, pages: 1144.

[7] A. Censi, K. Slutsky, T. Wongpiromsarn, D. Yershov, S. Pendleton, J. Fu, E. Frazzoli, Liability, Ethics, and Culture-Aware Behavior Specification using Rulebooks, in: 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 8536–8542. doi:10.1109/ICRA.2019.8794364, iSSN: 2577-087X.

[8] M. Hekmatnejad, S. Yaghoubi, A. Dokhanchi, H. B. Amor, A. Shrivastava, L. Karam, G. Fainekos, Encoding and monitoring responsibility sensitive safety rules for automated vehicles in signal temporal logic, in: Proceedings of the 17th ACM-IEEE International Conference on Formal Methods and Models for System Design, MEMOCODE '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 1–11. URL: https://dl.acm.org/doi/10.1145/3359986.3361203. doi:10.1145/3359986.3361203.

[9] N. Aréchiga, Specifying Safety of Autonomous Vehicles in Signal Temporal Logic, in: 2019 IEEE Intelligent Vehicles Symposium (IV), 2019, pp. 58–63. doi:10.1109/IVS.2019.8813875, iSSN: 2642-7214.

[10] J. Karlsson, 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. doi:10.1109/CASE48305.2020.9217037, iSSN: 2161-8089.

[11] J. Lin, W. Zhou, H. Wang, Z. Cao, W. Yu, C. Zhao, D. Zhao, D. Yang, J. Li, Road Traffic Law Adaptive Decision-making for Self-Driving Vehicles, in: 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), 2022, pp. 2034–2041. doi:10.1109/ITSC55140.2022.9922208.

[12] X. Li, G. Rosman, I. Gilitschenski, J. DeCastro, C.-I. Vasile, S. Karaman, D. Rus, Differentiable Logic Layer for Rule Guided Trajectory Prediction, in: Proceedings of the 2020 Conference on Robot Learning, PMLR, 2021, pp. 2178–2194. URL: https://proceedings.mlr.press/v155/li21b.html, iSSN: 2640-3498.

[13] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, M. Maurer, Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving, in: 2015 IEEE 18th International Conference on Intelligent Transportation Systems, 2015, pp. 982–988. doi:10.1109/ITSC.2015.164, iSSN: 2153-0017.

[14] P. Bender, J. Ziegler, C. Stiller, Lanelets: Efficient map representation for autonomous driving, in: 2014 IEEE Intelligent Vehicles Symposium Proceedings, 2014, pp. 420–425. doi:10.1109/IVS.2014.6856487, iSSN: 1931-0587.

[15] ASAM OpenDRIVE, 2019. URL: https://www.asam.net/standards/detail/opendrive/.

[16] H. Bhuiyan, G. Governatori, A. Bond, S. Demmel, M. Badiul Islam, A. Rakotonirainy, Traffic Rules Encoding Using Defeasible Deontic Logic, in: S. Villata, J. Harašta, P. Křemen (Eds.), Frontiers in Artificial Intelligence and Applications, IOS Press, 2020. URL: http://ebooks.iospress.nl/doi/10.3233/FAIA200844. doi:10.3233/FAIA200844.

[17] H. Bhuiyan, G. Governatori, A. Rakotonirainy, M. W. Wong, A. Mahajan, Traffic rule formalization for autonomous vehicle (2022). URL: https://ink.library.smu.edu.sg/cgi/viewcontent.cgi?article=6124&context=sol_research.

[18] M. Buechel, G. Hinz, F. Ruehl, H. Schroth, C. Gyoeri, A. Knoll, Ontology-based traffic scene modeling, traffic regulations dependent situational awareness and decision-making for automated vehicles, in: 2017 IEEE Intelligent Vehicles Symposium (IV), 2017, pp. 1471–1476. doi:10.1109/IVS.2017.7995917.

[19] Q. Zhang, D. K. Hong, Z. Zhang, Q. A. Chen, S. Mahlke, Z. M. Mao, A Systematic Framework to Identify Violations of Scenario-dependent Driving Rules in Autonomous Vehicle Software, Proceedings of the ACM on Measurement and Analysis of Computing Systems 5 (2021) 15:1–15:25. URL: https://dl.acm.org/doi/10.1145/3460082. doi:10.1145/3460082.

[20] M. Luckcuck, M. Farrell, L. A. Dennis, C. Dixon, M. Fisher, Formal Specification and Verification of Autonomous Robotic Systems: A Survey, ACM Computing Surveys 52 (2019) 100:1–100:41. URL: https://dl.acm.org/doi/10.1145/3342355. doi:10.1145/3342355.

[21] P. Hunter, J. Ouaknine, J. Worrell, When is Metric Temporal Logic Expressively Complete?, 2013. URL: http://arxiv.org/abs/1209.0516, arXiv:1209.0516 [cs].

[22] R. Alur, T. A. Henzinger, Logics and models of real time: A survey, in: J. W. de Bakker, C. Huizing, W. P. de Roever, G. Rozenberg (Eds.), Real-Time: Theory in Practice, Springer Berlin Heidelberg, Berlin, Heidelberg, 1992, pp. 74–106.

[23] Y. Gurevich, ON THE CLASSICAL DECISION PROBLEM, in: Current Trends in Theoretical Computer Science, WORLD SCIENTIFIC, 1993, pp. 254–265. URL: http://www.worldscientific.com/doi/abs/10.1142/9789812794499_0020. doi:10.1142/9789812794499_0020.

[24] P. Thati, G. Roşu, Monitoring Algorithms for Metric Temporal Logic Specifications, Electronic Notes in Theoretical Computer Science 113 (2005) 145–162. URL: https://www.sciencedirect.com/science/article/pii/S1571066104052570. doi:10.1016/j.entcs.2004.01.029.

[25] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. Qi, Y. Zhou, Z. Yang, A. Chouard, P. Sun, J. Ngiam, V. Vasudevan, A. McCauley, J. Shlens, D. Anguelov, Large Scale Interactive Motion Forecasting for Autonomous Driving : The Waymo Open Motion Dataset, 2021. URL: http://arxiv.org/abs/2104.10133. doi:10.48550/arXiv.2104.10133, arXiv:2104.10133 [cs].

[26] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kummerle, H. Konigshof, C. Stiller, A. de La Fortelle, M. Tomizuka, INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Seman-

tic Maps, 2019. URL: http://arxiv.org/abs/1910.03088. doi:10.48550/arXiv.1910.03088, arXiv:1910.03088 [cs, eess].

[27] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, O. Beijbom, nuScenes: A multimodal dataset for autonomous driving, 2020. URL: http://arxiv.org/abs/1903.11027. doi:10.48550/arXiv.1903.11027, arXiv:1903.11027 [cs, stat].

[28] S. Maierhofer, M. Klischat, M. Althoff, CommonRoad Scenario Designer: An Open-Source Toolbox for Map Conversion and Scenario Creation for Autonomous Vehicles, in: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), 2021, pp. 3176–3182. doi:10.1109/ITSC48978.2021.9564885.

[29] K. Leung, N. Arechiga, M. Pavone, Backpropagation for Parametric STL, in: 2019 IEEE Intelligent Vehicles Symposium (IV), IEEE, Paris, France, 2019, pp. 185–192. URL: https://ieeexplore.ieee.org/document/8814167/. doi:10.1109/IVS.2019.8814167.

[30] S. Oliveira, D. Loureiro, A. Jorge, Improving Portuguese Semantic Role Labeling with Transformers and Transfer Learning, in: 2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA), 2021, pp. 1–9. URL: http://arxiv.org/abs/2101.01213. doi:10.1109/DSAA53316.2021.9564238, arXiv:2101.01213 [cs].