Engineering 33 (2024) 47–62

Contents lists available at ScienceDirect

Engineering

journal homepage: www.elsevier.com/locate/eng

Research
Safety for Intelligent and Connected Vehicles—Article

Semantic Consistency and Correctness Verification of Digital Traffic Rules

Lei Wan [a], Changjun Wang [b,*], Daxin Luo [a], Hang Liu [a], Sha Ma [a], Weichao Hu [b]

[a] Policy, Standard, and Patent Department, Intelligent Automotive Solution BU, Huawei Technologies Co., Ltd., Beijing 100094, China
[b] Research Institute for Road Safety of the Ministry of Public Security, Beijing 100062, China

ARTICLE INFO

Article history:
Received 27 August 2022
Revised 28 November 2022
Accepted 6 April 2023
Available online 14 August 2023

Keywords:
Autonomous driving
Traffic rules
Digitization
Formalization
Verification

ABSTRACT

The consensus of the automotive industry and traffic management authorities is that autonomous vehicles must follow the same traffic laws as human drivers. Using formal or digital methods, natural language traffic rules can be translated into machine language and used by autonomous vehicles. In this paper, a translation flow is designed. Beyond the translation, a deeper examination is required, because the semantics of natural languages are rich and complex, and frequently contain hidden assumptions. The issue of how to ensure that digital rules are accurate and consistent with the original intent of the traffic rules they represent is both significant and unresolved. In response, we propose a method of formal verification that combines equivalence verification with model checking. Reasonable and reassuring digital traffic rules can be obtained by utilizing the proposed traffic rule digitization flow and verification method. In addition, we offer a number of simulation applications that employ digital traffic rules to assess vehicle violations. The experimental findings indicate that our digital rules utilizing metric temporal logic (MTL) can be easily incorporated into simulation platforms and autonomous driving systems (ADS).

1. Introduction

In recent years, there has been increasing interest in autonomous vehicles, particularly in regard to the traffic compliance of autonomous driving systems (ADS). This interest stems from the fact that autonomous vehicles are operated on public roads alongside conventional vehicles. Traffic regulations, which serve as invisible administrators in human society to ensure the safety and efficiency of traffic flow, are absolutely essential. Thus, it is necessary for relevant authorities to mandate that autonomous vehicles adhere to human driver traffic laws. In addition, ADS should be governed by well-established regulations to ensure public safety [1–3]. However, the traffic rules described in natural languages for human drivers are sometimes subjective and nondeterministic, making it difficult for autonomous vehicles to adhere to them, as robots always require objective and concrete digital rules.

Rule digitization is the process of converting regulations written in natural language for human drivers into computer-understandable digital equations. With digital traffic rules, traffic management systems will be able to conduct automated compliance reviews of motor vehicle driving behaviors, vastly improving traffic management efficiency. At the same time, relevant government departments will be able to conduct regulatory tests and evaluations on autonomous vehicles to determine whether the vehicles meet the requirements for driving on the road, and autonomous driving system development departments for car manufacturers will be able to use digital traffic rules.

It is notable that ensuring the consistency and correctness of traffic rules expressed in machine language is an extremely important and difficult topic. In this context, "consistency" indicates that the digital traffic rules and natural language traffic rules are semantically equivalent and describe the same vehicle behavior and road environment, while "correctness" indicates that digital rules can be accurately understood by machines while avoiding the imprecise and difficult quantification of natural language rules due to factors such as speaking skills, human experience, and hidden consensus. The consistency and correctness of digital traffic rules must be ensured before such rules can be used to determine illegal driving behavior and make decisions for ADS.

Nevertheless, natural languages and computer languages are extremely distinct, and it is difficult to translate between them.

* Corresponding author.
E-mail address: wcj121@sina.com (C. Wang).

https://doi.org/10.1016/j.eng.2023.04.016

Natural language processing (NLP) technology has been developed for several decades [4,5]. Moreover, computational linguistics considers the statistical or rule-based modeling of natural language [6]. However, these technologies still fall short of expectations when handling rigorous text such as traffic regulations [7]. Utilizing a formal verification method to validate digital rules during the process of traffic rule digitization is therefore an important and necessary step.

2. Related works

2.1. Digitization and the implementation of rules

Temporal logic (TL) is an excellent choice for achieving a suitable digital description of rules, due to its safety and expressiveness. TL consists of a collection of logical formalisms that can describe the specific behavior of a system with finite states [8]. Traditional TLs include linear TL (LTL), computational tree logic (CTL), CTL*, metric TL (MTL), and timed propositional TL (TPTL). LTL, which is also known as propositional TL, can encode formulas with temporal modal operators [9]. CTL uses a tree-like structure to represent the time model; it is always employed by certain model checkers [10]. As a superset of CTL and LTL, CTL* is an advanced form of branching-time logic [11]. TPTL is a type of propositional TL that can be used for both natural language specification and formal verification [12]. As an extension of TL, MTL [13,14] is frequently used to describe the behavior and situation of an object [15]. MTL is a flexible formal representation that can be used in real-time systems [16] by utilizing a time-constrained version of temporal operators. Real-time logic contains explicit references to time. It can express behaviors based on quantitative timing requirements [13,16,17], which is useful when describing the behavior of a particular period.

In the higher order logic (HOL) proof assistant Isabelle, Rizaldi and Althoff [18] specified certain highway traffic rules from the Vienna Convention on Road Traffic in order to formalize traffic regulations. Using LTL, Esterle et al. [19] formalized traffic rules for machine interpretability. Based on the German Road Traffic Regulation, Maierhofer et al. [20] formalized interstate traffic regulations. All of these examples are exceptional applications of TL. However, they lack verification information to ensure that their methods are beneficial to natural ADS.

Karimi and Duggirala [21] formalized certain traffic rules in the California Department of Motor Vehicles (DMV) driver's manual and simulated the behavior of autonomous vehicles using the Carla simulation tool. Although this work is an excellent implementation of traffic regulations, it does not check the rule provisions for correctness. Using answer-set programming, Beck et al. [22] presented an implementation of multiple tasks, including consistency testing, diagnosis, and repair. However, this study did not evaluate traffic regulations regarding the movement of the vehicle. Krasowski and Althoff [23] also formalized marine traffic rules from the Convention on the International Regulations for Preventing Collisions at Sea (COLREGS) by employing TL and evaluating the digitized rules with real data. Although this work is an admirable effort, it lacks formal verification.

Esterle et al. [24] used formalized traffic rules represented in LTL to verify the high-level behavior of an autonomous vehicle in a semantic state space but skipped the formal verification phase of the rules. Hannah et al. [25] proposed the use of predicate logic to implement digital traffic rules in the field of industry standards. In order to facilitate the methodical design of propositions and logic, they categorized the descriptors in natural language as behavior, operational design domain (ODD), and assumptions. This method can precisely convert natural language text into predicate logic; however, without systematically describing the scenarios, the implicit meaning of the traffic rules is easily missed. In addition, there is no mention of a verification method for digital traffic rules.

As stated previously, there are a variety of methods and logical languages that can be used to describe traffic regulations, each with distinct expressive capacities and forms of expression. However, the majority of methods lack a design flow that can be used to digitally implement all traffic rules based on the characteristics of traffic scenarios.

2.2. Formalized rules verification

This paper describes rules formalization as a mathematical technique for implementing traffic rule models for ADS. It is absolutely necessary to ensure the correctness of these models. Formal verification is always used to ensure that a design conforms to a precisely stated concept of functional correctness in the fields of software engineering and integrated circuit design [26]. Formal verification is advantageous for demonstrating the correctness of a design, such as a software or hardware system, program, or model. It is primarily accomplished through theorem proving, model checking, and equivalent checking.

Theorem proving—also known as automated theorem proving—is a method for proving mathematical theorems using computer programs [27]. In the formal verification of digital traffic rule models, it is possible to formalize the models as mathematical equations, and then deduce the correctness of those equations using an appropriate definition and theorem. Another method is model checking, which is also known as property checking [28]. Before a system or model is verified, it must be formalized as a finite-state machine (FSM). By defining and programming a few properties, the machine can automatically check the FSM and identify any issues [29,30]. To automatically verify the model and solve the verification problem algorithmically, both the system model and its properties must be expressed in precise mathematical language. The third method, equivalent checking, is primarily used in the field of hardware verification, particularly in the development of integrated circuit design [31]. Literally, an equivalence model with the same function as the system model is created. In theory, these two models should perform identically when both are accurate.

We have not discovered any methods or resources for validating the correctness of a digital traffic rule. Thus, the methodology for verification is the most innovative aspect of this paper. We refer to formal verification techniques used in the design of integrated circuits and propose a new semantic consistency and correctness verification technique that is applicable to digital traffic rules.

2.3. Contributions

To ensure the consistency and correctness of digital traffic rules and to fill a technical void in rule verification methodology, we propose a formal verification method for digital traffic rules. In this paper, we develop a closed-loop methodology that includes traffic rule digitization, formal verification, and simulation testing. This methodology can improve the authority, accuracy, and accessibility of digital traffic rules.

First, we propose a method for digitizing traffic rules that uses MTL to easily convert existing traffic rules into digital rules. To allow scenarios and violation conditions to be described hierarchically, we formulate multiple propositions at different levels and define a comprehensive procedure for designing digital rules. This methodology is applicable to numerous types of regulations and reduces the bugs that may result from inadequate consideration.

The second objective of this paper is to determine how to ensure the consistency and correctness of digital traffic rules.

Combining equivalence verification with model checking, we propose a formal verification method to ensure that the digitized MTL traffic rules are reasonable and reassuring. In a final application example, we develop a simulation platform to test and validate the proposed digital traffic rules.

We have developed a formal verification method for digital traffic rules for the first time in the automative industry and have successfully completed the formal implementation, verification, and application of traffic rules in an autonomous driving simulator.

3. Overview of the digitization, verification, and application procedure

Fig. 1 provides a flowchart for the proposed digitization, validation, and implementation of traffic rules. The entire process can be broken down into the following segments: ① digitize and formalize traffic rules to obtain digital rules; ② formally verify the digital rules to ensure their semantic consistency and correctness; and ③ apply digital rules to scenarios such as simulation, field testing, and ADS on vehicles. First, MTL is used to formalize the traffic rules and obtain MTL models, which are also known as digital traffic rules. Digital traffic rules are translations of human drivers' traffic rules into a format that computers can understand. They typically describe behaviors that are expressly permitted or prohibited, regardless of how vehicles or pedestrians perceive or evaluate the current environment. Indeed, vehicle behavior is frequently unpredictable. When a vehicle is on the line of a road, for example, it can be unclear which lane it belongs to. Sometimes this is because the state description is too broad, and at other times it is because the vehicle state must be artificially specified. In practice, we require reasonable definitions in order to obtain specific states. In the field of autonomous driving, formalized digital traffic rules will be used as behavior standards or codes. This MTL model, for example, will be utilized in ADS and will impact the driving decisions and behavior of vehicles. Therefore, it is necessary to ensure the model's correctness, which requires formal validation of the digital rules. Formal verification of the digital rules is thus essential.

We combine model checking with equivalence checking in the formal verification to ensure that the MTL model is consistent and accurate. A state machine provides a powerful method for modeling dynamic driving behavior. For each traffic rule, an FSM is designed to describe its specification in order to ensure that the behaviors are precisely expressed and understood. Because the states of an environment and traffic participants are specified in traffic rules, the transition between states can be expressed precisely using a state machine. The state machine is used in two ways: First, the state machine and the MTL model are placed in an equivalence checker to validate their behavioral consistency; and, second, some properties that the MTL model should possess are extracted from the state machine and then used for model checking. The equivalence checker and model checker collaborate to ensure the correctness of the MTL model. If any issues with a digital rule are discovered during formal verification, the MTL model must be modified. This procedure will continue until the digital traffic rule is proven to be accurate.

The digital traffic rules are then implemented in various scenarios where traffic rules must be checked, including but not limited to ① automatic driving simulation, which is used to determine whether the driving trajectory of the tested vehicle or the planning algorithm will result in illegal behavior; ② self-driving vehicle testing, which uses instruments or algorithms at the test site to determine whether the vehicle being tested violates the rules; and ③ autonomous vehicle operation, during which the planning algorithm can use digital rules to determine the legality of its planned routes, thereby preventing illegal driving.

Fig. 2 depicts a method for applying traffic rule digitization. To determine whether a vehicle violates traffic laws, it is necessary to implement not only digital traffic rules but also checker software to parse the MTL model and perform calculations. A checker's inputs consist of environmental data (e.g., weather, road structure, etc.), vehicle trajectory data, and vehicle status data (e.g., speed and lighting data). The checker parses digital traffic rules in MTL, determines whether the test subject violates the digital traffic rules based on the input data, and outputs the traffic rule compliance result.

The checker's input information, including the environment, trajectories, and status, can come from a variety of sources, including perception algorithms, sensors, and simulation platforms, depending on the application. In the simulation test, if the only concern is whether the vehicle violates traffic rules, the input data can be obtained directly from the simulation software, as opposed to the output of the perception result of a vehicle under testing, as we do in this paper. When a checker is implemented in an autonomous driving system, the majority of input data can only be gathered from sensors and perception algorithms. In such cases, the precision of the perception algorithms could diminish the precision of the compliance result. This paper focuses primarily on the methodology of digitizing traffic rules—that is, the process of designing digital traffic rules and ensuring semantic consistency with laws and regulations expressed in natural language.

It is notable that traffic behaviors occur in continuous space; consequently, the calculation of traffic compliance should also be

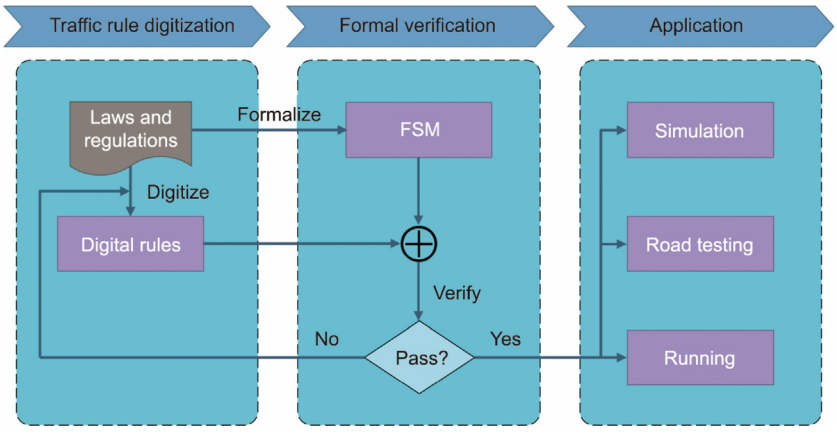


Fig. 1. The flow of the proposed traffic rule digitization.

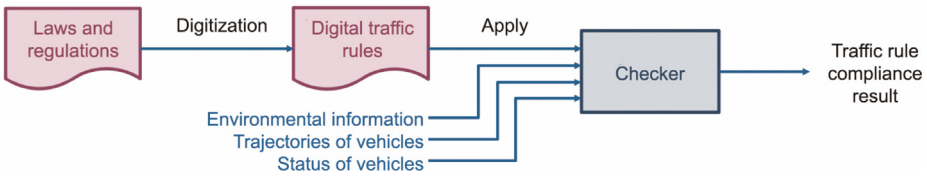


Fig. 2. The proposed digitalization of traffic regulations.

continuous. In practice, however, we adopt a discretization method to facilitate calculation: We sample data at specific time intervals and calculate the MTL digital rules at each sampling point. This method significantly improves performance and simplifies software development. Nevertheless, an excessively large sampling interval may result in an error. For example, if the speeds at two adjacent sampling points are normal, a potential over-speeding behavior between the two sampling points would not be identified. The interval's value is a compromise between precision and performance.

4. The methodology for digitizing traffic rule

A digital traffic rule consists of various propositions and operators. It can be viewed as a mathematical formula or model that restricts the behavior of a vehicle in a particular environment. Computers can parse and calculate the formula to determine whether the tested vehicle's behavior conforms to applicable regulations. The propositions are the fundamental building blocks of digital rules. In this paper, we propose a method for proposition design that can accurately and completely describe the road environment and the behavior of traffic participants.

TL in the metric system is used here to describe digital traffic rules. The advantage of MTL is that it extends TL by employing time-constrained versions of temporal operators, such as "until" and "next" operators [32]. As a prominent specification formalism for real-time systems [33], MTL is ideally suited for formalizing a vehicle's behavior-describing set of trajectory points. The principal MTL operators are listed in Table 1.

The traffic rule specifications used in this paper come from the Road Traffic Safety Law of the People's Republic of China [34], Regulations on the Implementation of the Road Traffic Safety Law of the People's Republic of China [35], and Operating Specifications for Safe and Civilized Motor Vehicle Drivers [36]. This paper's proposed digitization and verification methods can also be applied to the traffic regulations of other nations and to a variety of other types of regulations.

Table 1
The MTL operators.

Type	Operator	Description
Logical operators	¬	Not, negation
	∨	Or, disjunction
	∧	And, conjunction
	→	Material implication, if…then…
Temporal modal operators	U	Until, where $U_{[T0,T1]}$ means "from T0 until T1"
	X	Next
	G	Always, where $G_{[T0,T1]}$ means "always happens in [T0, T1]"
	F	Future, where $F_{[T0,T1]}$ means "will happen in [T0, T1]"

T0: the start time of the operator's scope; T1: the end time of the operator's scope.

4.1. Grading and classification of propositions

A proposition represents a behavior that is declarative. Each proposition p, where $p \in P$ and P is a set of propositions, represents a Boolean statement. In other words, "a proposition is the non-linguistic carrier of truth or falsity, rendering any sentence expressing it either true or false" [37]. We formulate various propositions based on the requirements for describing traffic scenarios. Using the six-layer scenario model of Pegasus [38], we classify propositions into the following six categories: road model, infrastructure, temporary modification, status description, behavior description, and environmental condition. Thus, we can create propositions that describe the testing environment and scenario in great detail. As shown in Table 2, we also creatively rank and classify the propositions to make the digital approach more hierarchical.

The nesting and calling of hierarchical or graded propositions are facilitated by their hierarchical or graded nature. Low-level propositions are used to compose high-level propositions, while programming languages implement the lowest-level propositions. Table 3 provides a comparison example describing the same rule using two distinct methods. Using hierarchical propositions can obviously make the rule simpler, more direct, and more easily understood. Low-level propositions can be simultaneously reused in multiple different high-level propositions, resulting in efficient implementation.

4.2. The digitization of a traffic rule

Given the set P of propositions defined above, any traffic rule φ can be formulated using MTL. Each digital traffic rule comprises propositions and operators. The result of digitizing traffic

Table 2
Proposition hierarchy and examples.

Category	Level	Proposition example	Description
Environmental condition	0	envWeather	Environmental and weather information
Status description	0	stProperDist	Moving targets and traffic participant status
		stProperSpeed	—
		stPosition	—
		stLampStatus	—
Behavior description	0	actTurn	Moving targets and traffic participant behavior
	1	actCross	—
		actSurpass	—
	2	actOvertake	—
Temporary modification	0	tmpRoadMaintenance	Facilities and events over time
Infrastructure	0	underSign	Road facilities and traffic signs
Road model	0	onRoadType	Road geometry information

Table 3
Comparison of various digitization techniques.

Rule	No speeding when overtaking
No hierarchical propositions	$(\neg \text{stObjPosition}(\text{Ego}, \text{Obj}, \text{Ahead})$
	$\wedge \text{X}(\text{stObjPosition}(\text{Ego}, \text{Obj}, \text{Ahead})))$
	$\wedge \text{F}(\text{stObjPosition}(\text{Ego}, \text{Obj}, \text{SameLane}))$
	$\wedge \text{F}_{[-T,0]}(\text{stObjPosition}(\text{Ego}, \text{Obj}, \text{SameLane}))$
	$\rightarrow \text{stProperSpeed}$
With hierarchical propositions	$\text{actOvertake} \rightarrow \text{stProperSpeed}$
	$\text{actOvertake} \triangleq \text{actSurpass}(\text{Ego}, \text{Obj})$
	$\wedge \text{F}(\text{stObjPosition}(\text{Ego}, \text{Obj}, \text{SameLane}))$
	$\wedge \text{F}_{[-T,0]}(\text{stObjPosition}(\text{Ego}, \text{Obj}, \text{SameLane}))$
	$\text{actSurpass}(\text{Ego}, \text{Obj}) \triangleq \neg \text{stObjPosition}(\text{Ego}, \text{Obj}, \text{Ahead})$
	$\wedge \text{X}(\text{stObjPosition}(\text{Ego}, \text{Obj}, \text{Ahead}))$

T: the length of time; $[-T, 0]$: a period of time of length T that has just passed.

rules—which we refer to as digital traffic rules—is a formula (called the MTL model) that describes permissible or prohibited behavior. Fig. 3 depicts a step-by-step process for digitizing a traffic rule. This methodology can simplify and expedite the digitization process while preventing the omission of conditions that lead to errors in digital rules.

First, as in step 1 in Fig. 3, normalize the traffic rule described in natural language to obtain a quantitative description that is easy for computers to parse. Second, as in steps 2 and 3, our method adheres strictly to the six-layer scenario model when describing the traffic scenario. Using static and dynamic propositions to describe the scenario, it is possible to obtain the environment status and behavior status of each traffic participant. Third, as in steps 4 and 5, use propositions to describe the judgment conditions corresponding to the traffic rules, thereby obtaining the final digital traffic rules. Following step 5, the output is the digital traffic rule, which is an MTL statement that can also be viewed as a formula or program code.

Fig. 3 depicts a method for easily and quickly digitizing a traffic rule and ensuring that the digitized rule is logically clear and semantically complete. Table 4 [34,36] contains two examples of results from digitization. The contents of Table 4 correspond to each step's output in Fig. 3, and the output of step 5 is the final digital traffic rule. These two digital traffic rules will also be utilized in the formal verification and simulation tests that will follow.

Table 4 displays normalized rules, which are quantitative representations of rules in natural language. This allows us to utilize convenient temporal and procedural expressions when describing traffic regulations. Example 1 describes a traffic rule regarding maintaining a safe distance between vehicles. Due to the difficulty of describing subjective things in programming languages, we transform them into objective representations, which necessitates the introduction of distance measures. In this instance, we define safe distance using time headway (THW) [39,40]. The traffic rule in example 2 restricts lane-changing behavior. Similar to example 1, we evaluate the rationality of continuous lane-changing behavior using time intervals. When evaluating digital traffic rules in MTL format, the results of the propositions must first be calculated using the trajectory information of each participant in the scenario; only then can the MTL result be obtained.

5. Formal verification of traffic rules

As stated previously, digital traffic rules must be correct and semantically consistent with human driver traffic rules. As a result, we propose the formal verification method shown in Fig. 4. This paper improves the traditional methods of theorem proving and model checking for traffic rule verification. Theorem derivation and verification are more suited to theorem proofs. Rizaldi et al. [41] used Isabelle/HOL to formalize and verify digital traffic rules. However, this method requires numerous definitions of fundamental concepts, and it is only useful for certain types of traffic

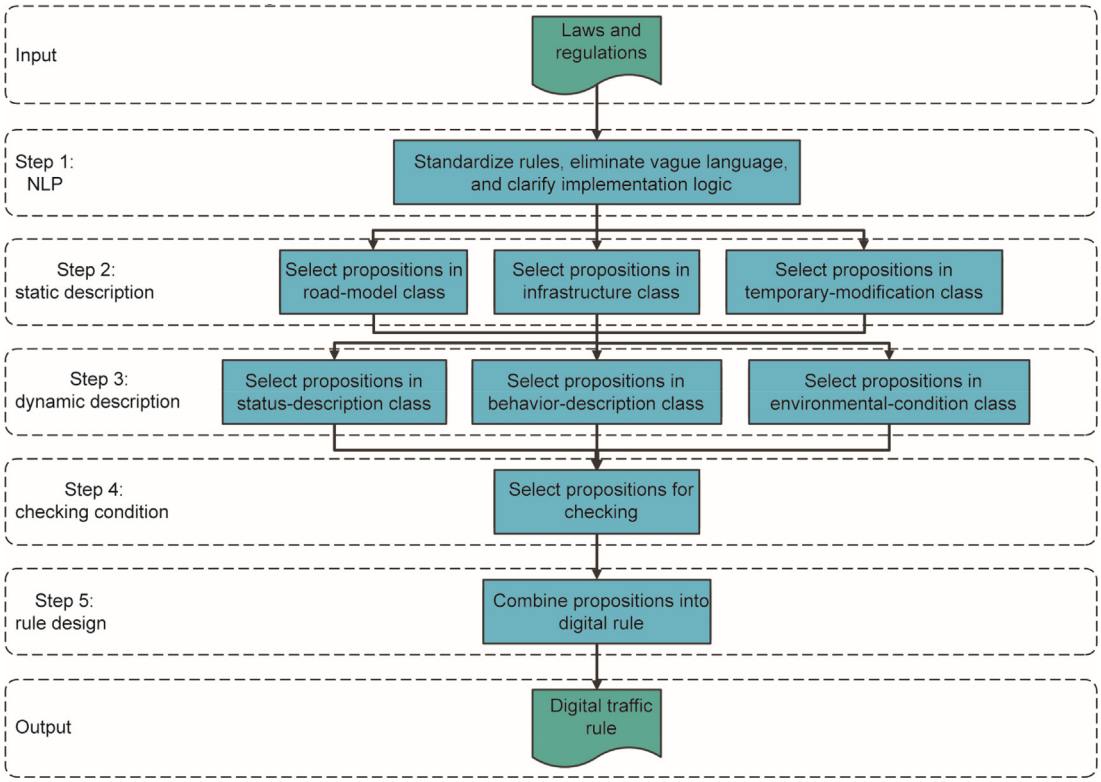


Fig. 3. The traffic rule digitization flowchart.

Table 4
Examples of traffic rule digitization.

Example	Step	Content or result
Example 1: maintain a safe distance	Input	For any two motor vehicles operating in the same driveway, the vehicle in the rear must maintain a safe distance sufficient for emergency braking (source: Refs. [34], No. 43)
	NLP	When two motor vehicles are operating in the same driveway, the vehicle in the rear must maintain a distance of at least [T] seconds from the vehicle in front
	Static description	onRoadType(Highway)
	Dynamic description	stObjPosition(Obj, Ego, Ahead) stObjPosition(Ego, Obj, SameLane)
	Checking condition	stProperDistance
	Rule design	onRoadType(Highway) $\wedge$ stObjPosition(Obj, Ego, Ahead) $\wedge$ stObjPosition(Ego, Obj, SameLane) $\rightarrow$ stProperDistance
Example 2: no continuous lane change	Input	The vehicle shall not change two or more lanes consecutively when changing lanes (source: Ref. [36] No. GA/T 1773.2–2021 8.3.2)
	NLP	The vehicle cannot perform the same lane change action within [T] seconds twice or more
	Static description	—
	Dynamic description	actCross(Left) actCross(Right)
	Checking condition	$G_{(-T,0)}(\neg$actCross(Left) $\vee \neg X(\neg$actCross(Right) $\cup$ actCross(Left))) $G_{(-T,0)}(\neg$actCross(Right) $\vee \neg X(\neg$actCross(Left) $\cup$ actCross(Right)))
	Rule design	actCross(Left)$\rightarrow G_{(-T,0)}(\neg$actCross(Left) $\vee \neg X(\neg$actCross(Right) $\cup$ actCross(Left))) actCross(Right)$\rightarrow G_{(-T,0)}(\neg$actCross(Right) $\vee \neg X(\neg$actCross(Left) $\cup$ actCross(Right)))

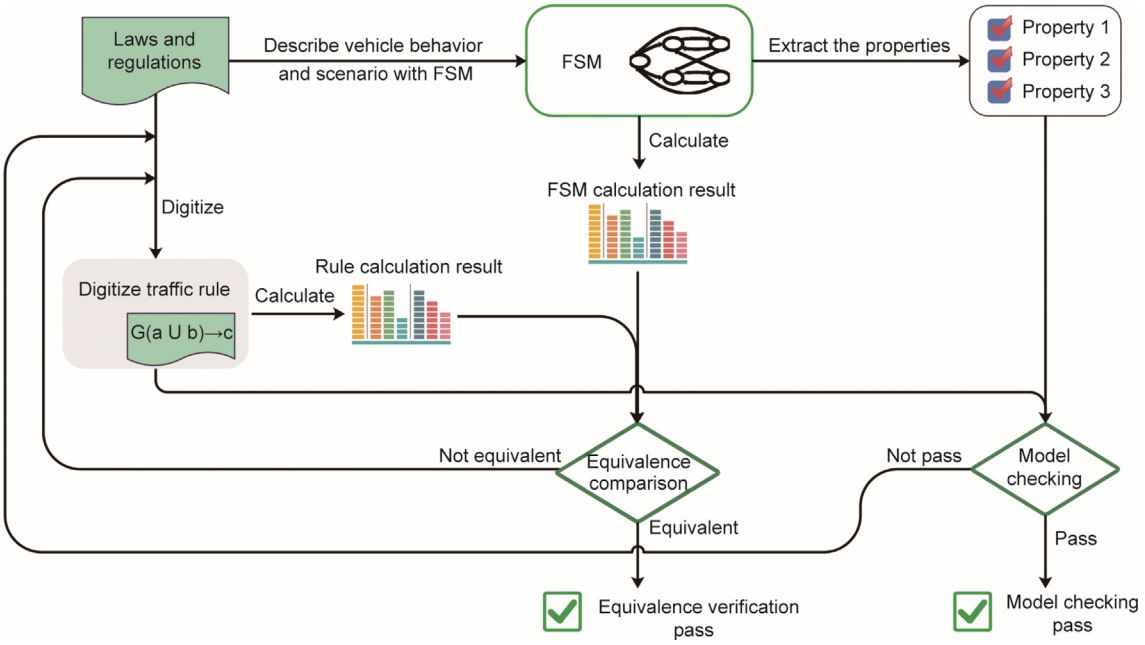


Fig. 4. Flowchart of the proposed formal verification method.

regulations. Using the method of theorem proving for complex traffic behaviors would make this task exponentially more difficult. In addition, a traditional model checker uses an FSM to simply express a system [42]. This is not an acceptable practice for digital traffic rules, as MTL formulas are intuitive and not excessively complex. Instead of their FSM, the digital traffic rules themselves should be used to ensure correctness.

As shown in the right half of Fig. 4, the proposed method primarily consists of two branches—namely, equivalence verification and model checking—to perform formal verification from multiple perspectives. Initially, the digital rule can be obtained in the form of a formula or a piece of MTL code using the digitization method described in the previous section. Using software tools and scenario-specific data, it is possible to calculate the outcome of this digital rule.

The scenario related to the traffic rule in question and the behavior of traffic participants are described using a state machine. By comparing the output of the state machine and the rule calculation result for equivalence verification, it is possible to determine whether the digital traffic rule is correct or, if it is not correct, where the error lies. When conducting equivalence comparisons, the formal verification tool traverses all possible states and state transitions, thereby covering the entire state space. Even though equivalence verification is computationally intensive and frequently takes hours to run, it is necessary in order to ensure the correctness of the digital traffic rules. Due to the adaptability and thoroughness of the state machine, it is simple to fix the bugs in a digital traffic rule by comparing it with the FSM. In addition, with this state machine, it is possible to abstract and design a series of traffic rule properties. Taking advantage of these properties, model checking can be used to verify digital traffic rules.

The state machine has excellent abstracting and describing capabilities. It can precisely and completely describe traffic situations and vehicle conditions. Although its accuracy cannot be absolutely ensured, it is very simple to use the state diagram to determine whether it adheres to the original traffic regulations. Implementing a complex state machine requires hundreds of lines of code, which makes maintenance and debugging extremely difficult; therefore, an FSM is unsuitable as a digital traffic rule and is better suited as a verification instrument. MTL and state machines are two distinct methods for expressing traffic rules. By comparing MTL with a state machine, it is feasible and simple to identify problems.

The following is a formal verification example for example 2 in Table 4.

5.1. State machine design

The traffic regulation described in Table 4 example 2 is used as an instance in this section. The propositions are the model's inputs. The [T] parameter quantifies the time intervals between two lane change actions in this example. By modifying this parameter, the author of the traffic rule can adjust its severity. In experiments, time T = 10 s is typically employed. Without this criterion, the traffic rule is subjective and indeterminable.

As illustrated in Fig. 5, $\triangle T$ indicates the interval between two lane changes, when $\triangle T \geq T$, the vehicle violates this traffic regulation by making two left-lane changes in succession. Typically, when a vehicle makes a second left-lane change, a violation will be issued.

Using the traffic rule description from Table 4, we design the FSM depicted in Fig. 6.

Here, STn (n is a natural number) is used to represent the various states in this FSM. The parameter t represents the length of time the vehicle stays in the current lane. A threshold (TH) is manually set to indicate how long the vehicle must remain in the lane before it is considered to have returned. When t exceeds the TH, this lane-change behavior is deemed to be unrelated to the previous lane-change behavior, indicating that it is not continuous lane changing.

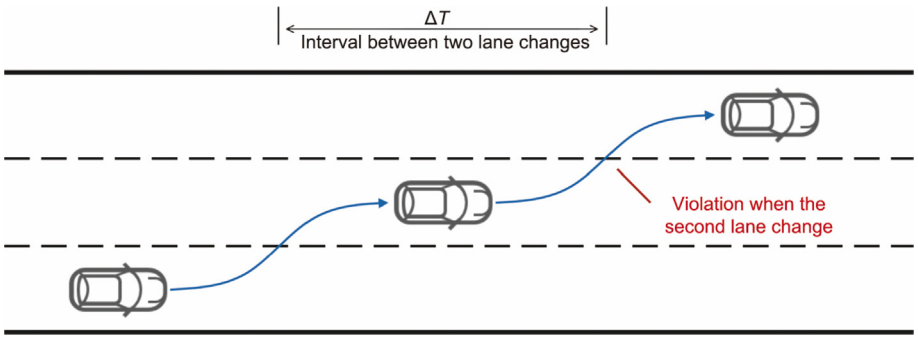


Fig. 5. Example of a traffic regulation.

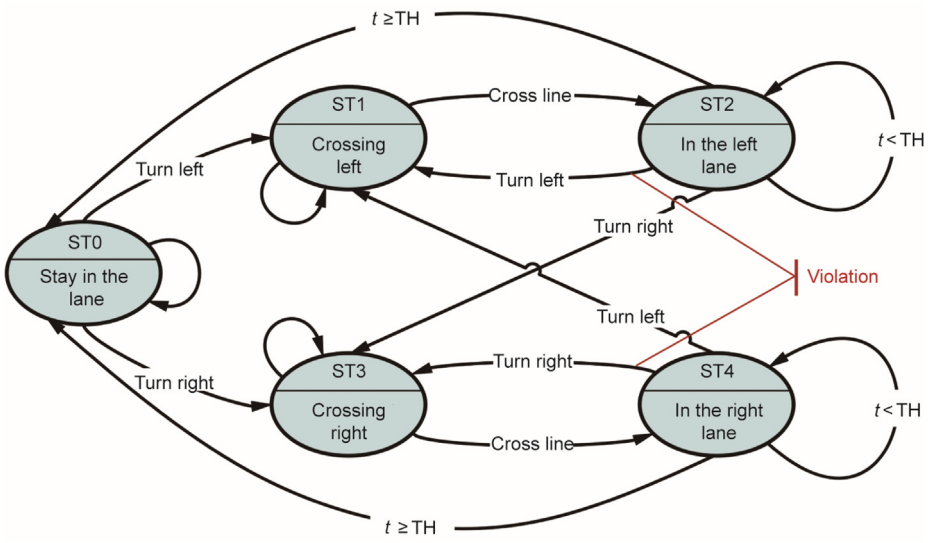


Fig. 6. The state machine for the traffic rule restricting lane-changing behavior. t: the length of time the vehicle stays in the current lane; TH: threshold; ST0: the state in the middle lane; ST1: the state in a left-hand crossing lane; ST2: the state in the left lane; ST3: the state in a right-hand crossing lane; ST4: the state in the right lane.

When the lane-changing scenario begins, the vehicle may be in one of three states: in a lane, in a left-hand crossing lane, or in a right-hand crossing lane. The three states respectively correspond to ST0, ST1, and ST3. In order to simplify the state machine, the relationship between the vehicle and the lane is primarily represented by three states: in the middle lane (ST0), which indicates that the vehicle can change lanes to the left or right; in the left lane (ST2), which indicates that the vehicle has made a left turn in the last T seconds; and in the right lane (ST4), which is the opposite situation of ST2. In accordance with the traffic rule, when the state is ST2 or ST4 and the vehicle's dwell time in the left/right lane exceeds T seconds, the vehicle is permitted to change lanes to the left and right; its state changes to ST0, and the lane it stays in the middle lane. Consequently, a rule violation can occur in one of two ways: either the vehicle stays in ST2 and crosses to the left again (ST2→ST1), or it stays in ST4 and crosses to the right again (ST4→ST3).

For example, if a vehicle remains in ST2, it has just made at least one lane change to the left. If it turns left a second time, it will cross the left lane line and be in the ST1 state of changing lanes to the left. This is a traffic violation, so when the signal changes from ST2 to ST1, the "breaking" parameter becomes true. After the completion of the action of crossing the lane line, the vehicle returns to ST2. Theoretically, when a vehicle crosses a lane line, it can change lanes to either the left or the right, so the state machine includes all lane-change-related cases. This rule does not apply to situations in which a vehicle drives straight ahead without changing lanes; we consider this to be driving in the lane, which falls under the ST0/ST2/ST4 conditions.

5.2. Properties definition

In the proposed verification procedure, the MTL model must be accompanied by a list of properties representing the requirements of the original natural language traffic rules. MTL is also utilized to express these properties of linear state sequences [17,29,42]. After they have been formally defined, the MTL models are presented for automated verification [29]. This section focuses on linear-time behavior and design properties using the state machine shown in Fig. 6.

Eq. (1) is the definition of the traffic rule according to Table 4. In addition, three properties are defined based on the state machine's description of the vehicle's behavior to ensure compliance with traffic regulations. These properties' implementations are respectively depicted in Eqs. (2)–(4).

$$\begin{aligned}\text{TrafficRule} \triangleq\ & \text{actCross(Left)} \vee \text{actCross(Right)} \\ & \rightarrow (\text{actCross(Left)} \wedge \mathrm{G}_{(-T,0)}(\neg\text{actCross(Left)} \\ & \vee \neg\mathrm{X}(\neg\text{actCross(Right)} \cup \text{actCross(Left)})) \\ & \vee (\text{actCross(Right)} \wedge \mathrm{G}_{(-T,0)}(\neg\text{actCross(Right)} \\ & \vee \neg\mathrm{X}(\neg\text{actCross(Left)} \cup \text{actCross(Right)}))\end{aligned} \tag{1}$$

5.2.1. Property 1: The right to switch lanes

Due to road structures and driving demands, a vehicle should have the opportunity to change lanes to the left or right, unless it is on the road's edge or is constrained by other traffic laws. Fig. 7 depicts a common driving need that is encountered in daily life.

Therefore, the MTL model should limit lane-change intervals rather than prohibiting continuous lane changes entirely. According to the FSM depicted in Fig. 6, it is permissible for the vehicle to return to the left or right lane after entering the ST2/ST4 state. The only restriction is that, when the vehicle makes a lane change, it must wait until the state changes from ST2/ST4 to ST0 before it is permitted to make the same lane change again.

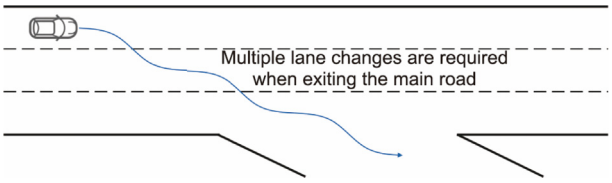


Fig. 7. The necessity for multiple lane changes.

Property 1. The traffic rule is violated if the time between lane changes is less than [T] seconds.

$$\text{Property 1} \triangleq \text{TrafficRule} == (t1 - t0 > \text{DeltaTime}) \tag{2}$$

Here, Eq. (1) defines TrafficRule. In Eq. (2), $t0$ and $t1$ are the times at which the vehicle crosses lanes in the same direction, and DeltaTime is the minimum interval between two lane-change behaviors, which is equal to T in Eq. (1). Due to the similarity between the left and right lane-change situations, only the left lane-change example is provided.

5.2.2. Property 2: Continuously change lanes

As depicted in Fig. 8, when a vehicle changes lanes in the same direction more than once within a limited amount of time, the traffic rule is violated for every lane change except the first. For example, the vehicle violates the rule twice if it changes lanes to the left three times in T seconds.

Property 2. The number of violations depends on the number of vehicle lane changes.

$$\begin{aligned}&\text{Property 2} \triangleq \forall t0 \in 1...\text{NOW}: \\ &\quad \text{IF} \wedge \text{ActCrossLeft}[t0] = \text{TRUE} \\ &\quad \wedge\ \forall t1 \in t0+1...\text{NOW: ActCrossRight}[t1] = \text{FALSE} \\ &\quad \wedge \forall t2 \in t0+1...\text{NOW: ActCrossLeft}[t2] = \neg\text{ActCrossLeft}[t2-1] \\ &\quad \text{THEN} \wedge \text{TrafficRule[NOW]} = \text{FALSE} \\ &\quad \wedge \forall t3 \in t0+2...\text{NOW: TrafficRule}[t3] = \neg\text{TrafficRule}[t3-1] \\ &\quad \text{ELSE TRUE}\end{aligned} \tag{3}$$

In this property, the vehicle violates the rule if it changes state from ST2/ST4 to ST1/ST3, regardless of how many times it has been in the ST2/ST4 state.

5.2.3. Property 3: Reverse lane change

What will occur when the vehicle first changes lanes to the left and then back? The FSM indicates that the vehicle does not violate this traffic regulation in this manner. As depicted in Fig. 9.

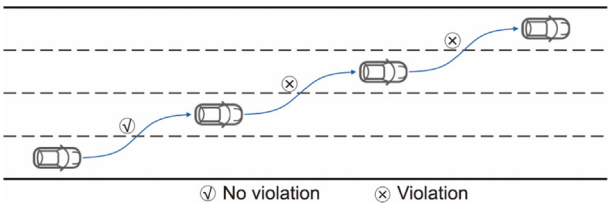


Fig. 8. Illustration of property 2.

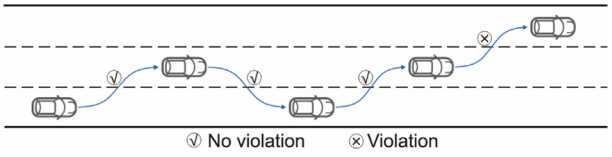


Fig. 9. Diagram of property 3.

Property 3. No violation occurs when an opposite lane change occurs between two identical lane changes.

$$\begin{aligned}\text{Property3} \triangleq\ & \text{IF ActCrossLeft}[\text{NOW}] = \text{TRUE}\\ & \text{THEN}\forall t0 \in 1...t-1:\\ & \text{IF} \wedge \text{ActCrossLeft}[t0] = \text{TRUE}\\ & \wedge \exists t1 \in t0+1...t-1: \text{ActCrossRight}[t1] = \text{TRUE}\\ & \wedge \forall t2 \in t0+1...t-1: \text{ActCrossLeft}[t2] = \text{FALSE}\\ & \text{THEN TrafficRule}[\text{NOW}] = \text{TRUE}\\ & \text{ELSE TRUE}\end{aligned} \tag{4}$$

According to the traffic rule, it is only when a vehicle crosses two lanes in the same direction that the vehicle violates the rule. Therefore, if the lane crossing is not continuous, the behavior is acceptable. This circumstance is easily ignorable, so we set this property to validate the MTL model.

We proposed the above three properties as model-checking examples. To verify each digital traffic rule, the designer of the digital traffic rules can create a variety of properties from various perspectives. Because each traffic rule corresponds to a unique scenario, it is essential to design properties and a state machine to ensure the correctness and semantic consistency of the digital rule.

5.3. Equivalent checking

Both the MTL and the FSM are used to describe the same traffic rule, but they are completely different. Their results, which indicate whether or not traffic laws have been broken, should be identical. Therefore, a method of equivalence checking can be used to ensure mutual correctness.

Definition 1. The outcome of the state machine and the MTL model must be identical.

$$\begin{aligned}\text{Consistency} \triangleq\ & \forall t \in 1...\text{SIMTIME: TrafficRule}[t]\\ & == \neg\text{Violation}[t]\end{aligned} \tag{5}$$

An implementation of Definition 1 is found in Eq. (5). The consistency of the state machine and the MTL model is defined as: At any time, their outputs indicate the same state. There is a logical negation symbol in Eq.(5), because when a violation occurs, the output "TrafficRule" in Eq.(1) is "false," while the state "violation" of the FSM is "true."

6. Evaluation of the formal verification method

Some formal verification tools use static analysis to prove or disprove that a program's behavior conforms to a formal specification [43]. Temporal Logic of Actions Plus (TLA$^+$) [44,45] and Z [46] are state-based descriptions that emphasize the characteristics and values of the system. Labelled Transition System Analyzer (LTSA) [47] is an event-based specification that describes a series of system events. SPIN [48], a generic verification system for models, is state-based but can describe a system that resembles imperative programming. PRISM [49] is also state-based but introduces a probabilistic Markov model. There are additional formal languages, each with its own benefits and drawbacks.

6.1. Evaluation technique and instruments

TLA$^+$ is a formal language of the highest level for modeling systems. For the formal verification of digital traffic rules, we employ TLA$^+$. In addition, TLC, a model checker, is the standard tool for model checking. Since the MTL model, FSM, and properties are all described by mathematical equations, formal verification with TLC is straightforward. Due to the invariance of the TLA$^+$ equation under stuttering, the Until and Next operators are unavailable in TLA$^+$. Eq. (1) is implemented in TLA$^+$ using the Any and Exist operators.

Fig. 10 is a schematic representation of a verification process. The simulation time refers to the number of states in each case. The parameter IT is used to represent the sampling interval, that is, the time interval between two states directly. In accordance with the real world, IT may be set to any period, such as 1 s. The status is updated every IT seconds. The interval time represents the length of time spent in the ST2/ST4 state. In addition, it corresponds to the minimum DeltaTime allowed between two consecutive lane changes.

During formal verification, TLA$^+$ automatically traverses all paths and state combinations to ensure that all possible cases are verified. When we apply the traffic rule to the state machine for formal verification, we can therefore verify the correctness of the traffic rule in a variety of situations. Case 6 is an S-shaped path in which the vehicle changes lanes first to the left and then to the right. Fig. 6's state machine demonstrates that this scenario does not result in a violation; however, if the verified traffic rule in Eq. (1) indicates a violation, the result of Eq. (5) is false, indicating that the traffic rule is inconsistent with the state machine and that the designed digital traffic rule is flawed.

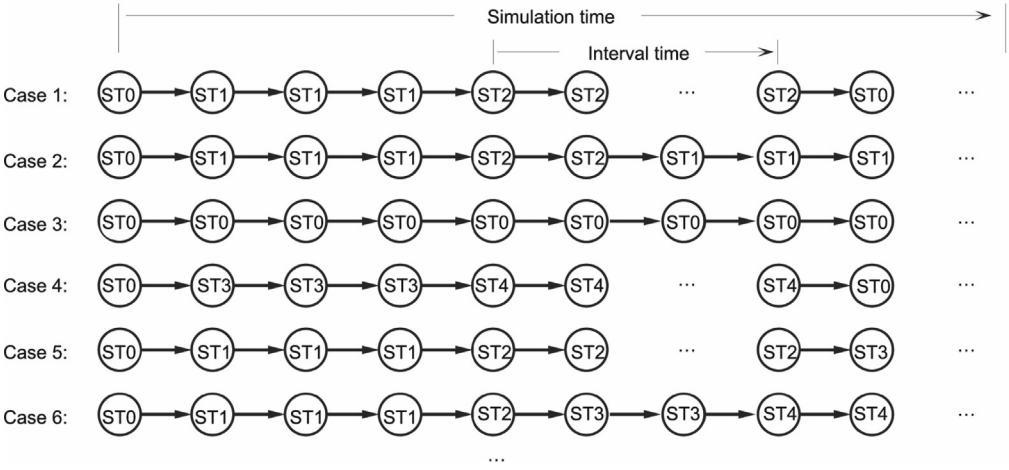


Fig. 10. Evaluation method.

6.2. Model implementation

The traffic rule model shown in Eq. (1) is implemented in TLC. The primary functions of the state machine are property definition and equivalent checking.

6.3. Evaluation results

We set IT to 1 s, the simulation time to 25 s, and the interval time to 10 s. The permitted interval time is the time between two lane changes in the same direction. The TLC is where the checking module is implemented. In this evaluation, a total of 16 666 625 states were discovered, and neither an error nor a warning was generated. In this verification flow, there were approximately 2 777 540 different possible states for ST1 and ST3 and approximately 5 553 740 different possible states for ST2 and ST4.

In this instance, the MTL model of traffic rule corresponds to the properties, and the output of the MTL model and the FSM are identical. The formal verification result indicates that the equivalence verification and model checking have been successfully completed. Eq. (1)'s digital traffic rule is consistent with the natural language traffic rule and can accurately describe violations.

6.4. Formal verification of the violation of an incorrect digital traffic rule

It is challenging to design the MTL equation correctly on the initial attempt. In this section, we provide an example of an incorrect digital traffic rule and then test its correctness using the formal verification method proposed in the previous section to determine whether this formal verification method can accurately identify problems with digitization in traffic rules.

$$\begin{aligned}\text{TrafficRule} \triangleq\ & \text{actCross(Left)} \vee \text{actCross(Right)} \\ & \rightarrow (\text{actCross(Left)} \wedge \mathrm{G}_{(-T,0)}(\neg\text{actCross(Left)})) \\ & \vee (\text{actCross(Right)} \wedge \mathrm{G}_{(-T,0)}(\neg\text{actCross(Right)})))\end{aligned} \tag{6}$$

Eq. (6) is another application of the traffic rule. When the vehicle crosses the lane line, the vehicle must ensure that a similar action has not occurred in the recent past.

When the verification flow depicted in Fig. 11 is executed, there are two conflicts: one between Eq. (6) and the state machine, and another between Eq. (6) and the aforementioned Property 3. This is because the property described by Eq. (4) is not satisfied by Eq. (6). When a vehicle changes lanes to the left lane and then to the right lane, moving to the left lane again should not be considered a continuous lane change, because a lane change to the right occurs between the two left turns. Property 3 describes this acceptable behavior well, but Eq. (6) reported a violation in step 5 of Fig. 11 due to the lack of consideration of the above situation.

This validation warning will not appear if Eq. (1) is used as the digital rule. The results of the formal verification utilizing these two digital rules are compared in Table 5.

This experiment demonstrates that the method proposed in this paper is effective. Using the formal verification ensures the consistency and correctness of traffic regulations.

7. Simulation experiment for the digitalization of a traffic rule

Once a digital traffic rule becomes available, it can be implemented on various computer systems, as follows: ① The autonomous driving simulation system can obtain the vehicle trajectory and environmental data in the simulation software, and then use the digital traffic rule to determine the legality of vehicle behavior. ② In the field test of a vehicle, the relevant data is recorded by the sensor in the field or on the vehicle, and the legality determination can be made on the remote server. ③ Using data from sensors and high-precision maps, an autonomous driving system can determine—while the vehicle is driving on the road—whether the driving trajectory or planned trajectory includes illegal behavior and can modify the driving behavior to prevent the occurrence of illegal behavior.

Using the aforementioned methods, we can evaluate whether the trajectory of an autonomous vehicle violates a particular traffic rule. This allows traffic authorities to determine whether a vehicle is roadworthy; the digital rules can also assist automobile

Table 5
Comparison of two equations.

Digital rule	Property 3: pass or not? (Eq. (4))	Consistency: pass or not? (Eq. (5))
Eq. (1)	Yes	Yes
Eq. (6)	No	No

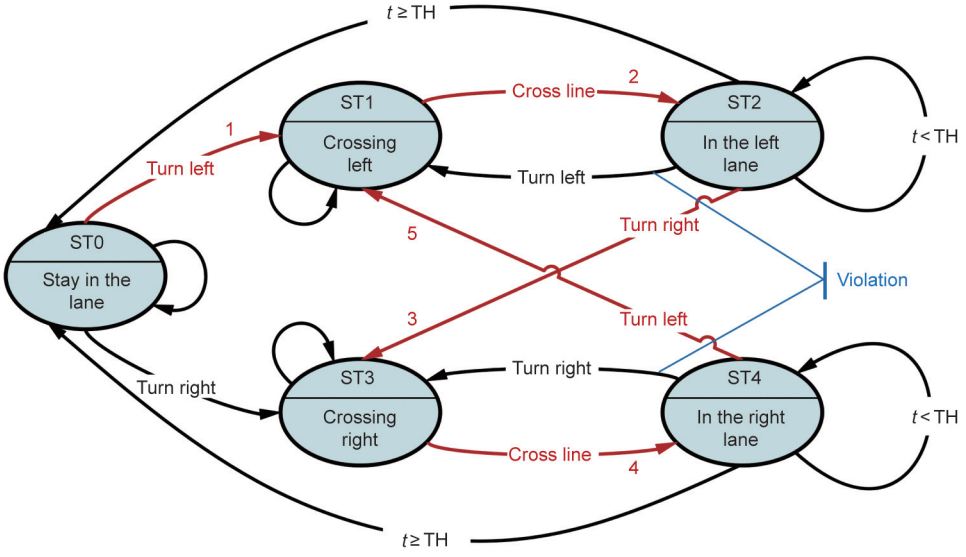


Fig. 11. The verification flowchart.

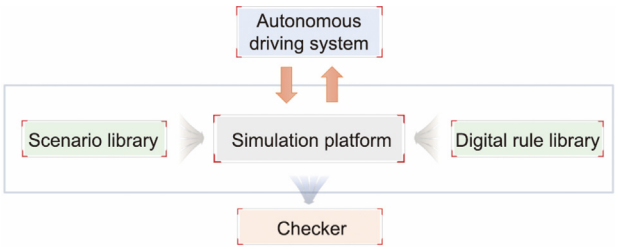


Fig. 12. The simulation system's framework.

manufacturers in determining whether their products comply with traffic regulations. In this section, we apply multiple digital rules to autonomous driving simulation experiments to determine whether the simulator-tested vehicle violates the rules.

7.1. The simulation system's framework

The architecture of the simulation system used in our experiments is depicted in Fig. 12. Based on the Virtual Test Drive (VTD) autonomous driving simulation software, we designed and developed a simulation platform for simulating vehicle behavior. As the test subject is connected to the simulation platform, an autonomous driving system was used to control the behavior of the ego vehicle in the simulation.

In addition, we created a scenario library containing various simulation scenarios, such as highway segments, intersections, and so forth. The scenario format was described in accordance with OpenDrive [50] and OpenScenario [51]. Several digital traffic rules were simultaneously stored in the rule library and provided to the simulation platform.

A software named Checker was designed for digitizing traffic rules in our experiments. It obtains the trajectory of each vehicle and environmental data from the simulation platform, analyzes

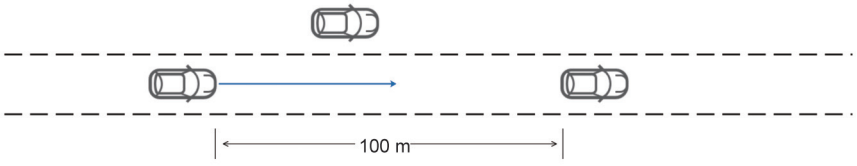
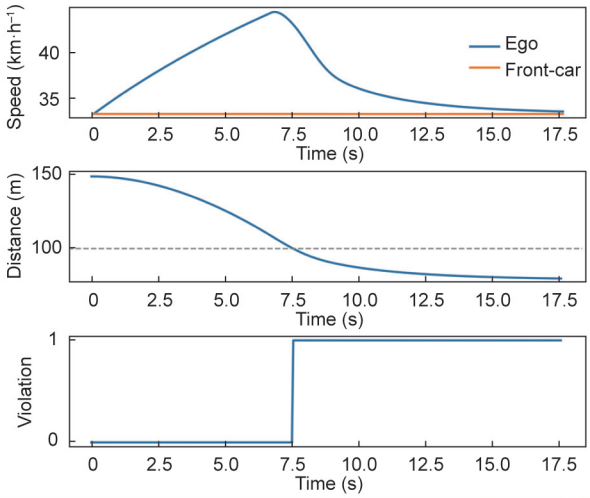
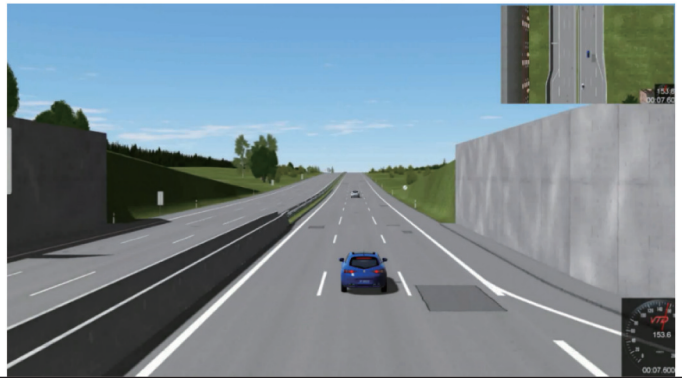
Step	Contents
Rule	For any two motor vehicles operating in the same driveway, the vehicle in the rear must maintain a safe distance sufficient for emergency braking measures [34]
MTL	onRoadType(Highway) ∧ stObjPosition(Obj, Ego, Ahead) ∧ stObjPosition(Ego, Obj, SameLane) → stProperDistance
Scenario	
Scenario description	The self-vehicle (denoted as the "ego") is faster than the vehicle in front of it; the distance between the self-vehicle and the vehicle in front of it in the same lane gradually decreases; when the distance is less than 100 m, it violates the traffic law
Checking result	
Screenshot of the violation	

Fig. 13. Simulation result 1.

the digital traffic rules, and determines whether the ego vehicle's behavior violates these rules. Checker parses the digital traffic rules and uses them to determine whether the simulation environment's vehicle behavior is illegal. Using this simulation system, we can determine whether a vehicle controlled by a planning algorithm in a self-driving vehicle violates a particular traffic regulation when operating in the simulation environment. This simulation system utilized a workstation (ThinkStation P720) with an Intel Xeon Gold 6134 CPU@3.2GHz and 128 GB memory.

7.2. Simulation

The same digital traffic rules as described in Table 4 are applied here. The results of the simulation experiment are listed below.

7.2.1. Digital rule 1: Safe distance requirements

The content and result of the simulation are listed in Fig. 13 [34]. To avoid danger, the driver must maintain a safe distance between the ego vehicle and the vehicle ahead in the same lane. In the screenshot at the bottom of Fig. 13, the blue vehicle represents the ego vehicle, and the distance to the car in front in the same lane decreases continuously from the start of the simulation. Starting at 7.6 s, the distance between the ego vehicle and the vehicle in front is less than 100 m.

Thus, checker is able to use the formalized digital traffic rule in Fig. 13 to determine whether the vehicle violates the traffic rule.

7.2.2. Digital rule 2: Continuous lane changes are prohibited

In this scenario, in order to reach the rightmost lane, the ego vehicle must make two lane changes. The experimental result is

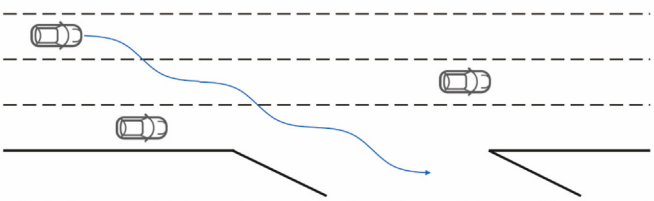
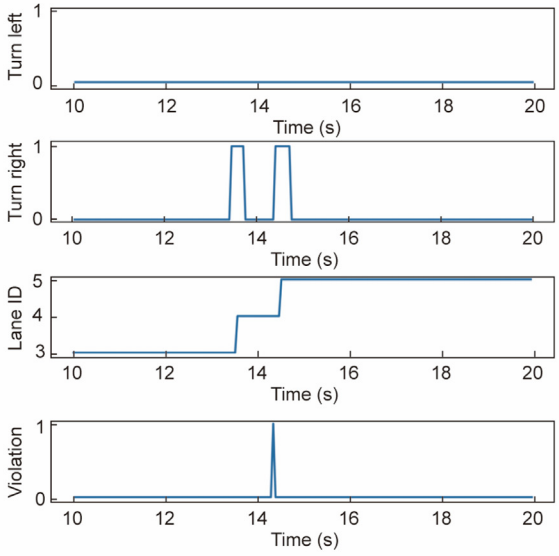
Step	Contents
Rule	When changing lanes, the vehicle shall not make two or more consecutive lane changes in a row
MTL	actCross(Left) ∨ actCross(Right) →(actCross(Left) ∧ $G_{(-7,0)}$(¬actCross(Left) ∨ ¬X(¬actCross(Right) ∪ actCross(Left))) ∨ (actCross(Right) ∧ $G_{(-7,0)}$(¬actCross(Right) ∨ ¬X(¬actCross(Left) ∪ actCross(Right)))
Scenario	
Scenario description	Initially, the ego vehicle was in the middle lane. The vehicle abruptly changes lanes to the right twice as it approaches the exit ramp, intending to exit the highway via the ramp. The lane identity document (ID) identifies the lane in which the ego vehicle is positioned
Checking result	
Screenshot of the violation	

Fig.14. Simulation result 2.

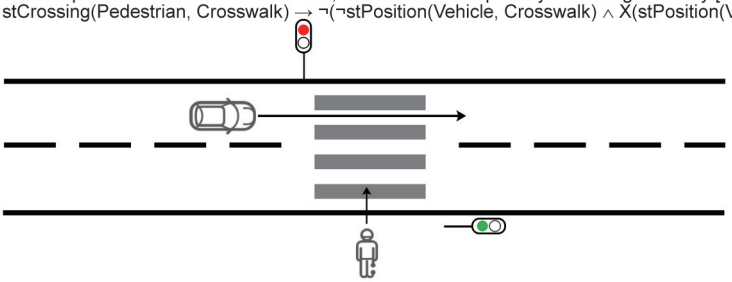
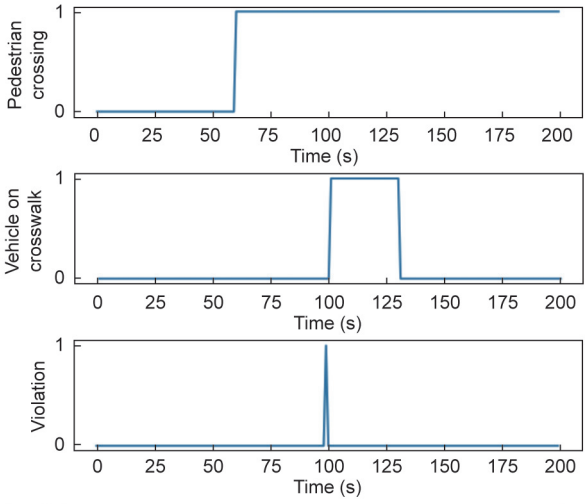
Step	Contents
Rule	When a pedestrian crosses a crosswalk, vehicles must stop and yield the right-of-way [34]
MTL	stCrossing(Pedestrian, Crosswalk) → ¬(¬stPosition(Vehicle, Crosswalk) ∧ X(stPosition(Vehicle, Crosswalk))
Scenario	
Scenario description	There is low visibility due to fog; the traffic light on the interstate is green, and the pedestrian crosses the street against the red light Contravening this traffic regulation, the vehicle does not yield to the pedestrian
Checking result	
Screenshot of the violation	

Fig. 15. . Simulation result 3.

depicted in Fig. 14. It is evident that the ego vehicle made two consecutive rightward lane changes, with only a 0.9 s gap between the lane changes. Checker is used to determine whether the vehicle violates the digital rule outlined at the top of the table. When the vehicle changed lanes for the first time, Checker did not note a violation; however, when the vehicle changed lanes for the second time at 14.35 s, Checker indicated that the traffic rule had been violated. The simulation results align well with our traffic rule definition.

7.2.3. Digital rule 3: Pedestrian first

In this scenario, the pedestrian begins crossing the street at the 3 s mark. Instead of evading, the vehicle continues straight forward. The experimental result is depicted in Fig. 15. The simulation demonstrates that, when the vehicle enters the pedestrian crossing, it is in violation of traffic regulations. The results of the experiment indicate that Checker and the digital traffic rules can correctly identify violations and output the violation's time and content in these scenarios.

8. Discussion

An authorized organization can use digital traffic rules to evaluate the compliance of a tested vehicle's behavior. In such a case, the vehicle enterprise is required to provide the authorized organization with the vehicle's trajectory and environmental data for offline evaluation. Checker of digital traffic rules can also be used as a module in an autonomous driving system to evaluate in real time whether a vehicle's trajectory or a path generated by a planning module complies with traffic rules. In this situation, the vehicle's compliance result may be affected by measurement errors and uncertainties in the planning and control module. Here, we briefly discuss these situations.

8.1. Influence of measurement errors

Inaccurate measurement may result in mistakes when enforcing traffic regulations. When calculating the condition of a vehicle, precise data such as position, heading, sideslip, and longitudinal speed are frequently required. Filters can be used to remove certain data fluctuations, such as position or speed jumps caused by measurement errors. However, filters are ineffective against some persistent errors because their characteristics are obscure. For example, to determine if a vehicle is running on the lane line by calculating whether the tire edge exceeds the center of the lane line, the positions of the vehicle and the lane line are needed. Unfortunately, the positioning error (e.g., 0.10 m) is frequently greater than half the width of the lane line (e.g., 0.13 m), making misjudgment possible.

The development of perception and positioning technology will diminish the evaluation error. For example, enhancing the accuracy of vehicle attitude estimation [52] and sideslip angle estimation [53] results in more precise vehicle locations and more sophisticated high-definition maps, particularly in Global Positioning System (GPS)-denied regions or dense urban areas. This permits a more precise relative position relationship between the vehicle and the lane line to be obtained.

8.2. Error dynamics and settling time

When an autonomous driving system controls a vehicle, deviations will occur between the actual motion and the desired motion, due to various errors and uncertainties. Error dynamics are used to describe the development of this controller deviation [54]. When an intelligent vehicle starts up or enters the automatic driving mode, it typically takes a while for the controller to reach a steady state. This period is known as the settling time. The settling time of a controller indicates how quickly it can maintain the steady-state error within a specified range.

When evaluating the behavior of a vehicle using digital traffic rules, the vehicle's steady state is typically not a concern, because we are more concerned with whether the behavior of a vehicle violates the rules than with why it may do so. Therefore, error dynamics are not a consideration when designing digital traffic rules. We can assume that error dynamics are stable in experiments.

However, error dynamics can be problematic when a large number of discrete scenarios are tested in simulation software or on a test field. Usually, we only care about a vehicle's violation of traffic laws when it is in a steady state. It is not necessary to start the evaluation before this condition has been reached.

To avoid the influence of error dynamics, we begin the compliance evaluation in the experiments a short time (usually 3 s) after the simulation has begun. This time can be used by the vehicle's controller to reach a steady state. During this period, no violations resulting from error responses will be detected.

8.3. Uncertainty in planning

Planning is a complex module of the autonomous driving system that requires precise perception of the environment, comprehension of the intentions of traffic participants, and the capacity to ensure stable and safe driving in a variety of scenarios. Under actual conditions, it is necessary to deal with a substantial amount of uncertainty related to the front and rear modules; moreover, the limitations of perception, the contingency of behavior prediction, and the interaction with control make the implementation of planning more difficult.

The solutions to deterministic decision problems with known alternatives in each state are relatively straightforward. In the real world, however, the behavior of other vehicles may be subject to a variety of uncertainties, so actions may result in a variety of new states with a certain probability. As a result of the bifurcation caused by the random state, the random search tree is exponentially more complex than the deterministic search tree. The performance of bounded uncertainty becomes an important issue that the decision module must address.

In various scenarios, the effect of this uncertainty on the performance and digitization of traffic rules can be analyzed. Digital traffic rules have two primary applications. The first application is for traffic administrators to test and monitor vehicle infractions. In this scenario, vehicle status data is uploaded to the cloud or a designated server, and the traffic manager calculates digital traffic rules offline. The absence of real-time requirements reduces the importance of performance.

Another application is for determining whether the ego vehicle has violated or is about to violate traffic regulations. For the former, calculations can be performed rapidly, because the vehicle's behavior is determined by the past; for the latter, since the decision tree may contain numerous action sequences, it is challenging to evaluate each sequence for violations. To improve performance, we can utilize the prior and reduce the sampling space for rules. For example, based on the current environment, we estimate the traffic rules that the vehicle is most likely to violate and only evaluate those rules.

In addition, the planning module must take into account the unpredictability of the interaction between traffic participants. Considering the unpredictability of pedestrian trajectories, ensuring the safety of pedestrians on the road, for example, is an important concern. The Road Traffic Safety Law of the People's Republic of China stipulates that vehicles must stop and yield to pedestrians using crosswalks [34]. As demonstrated in Section 7.2.3, we set corresponding terms when designing the digital traffic rules. To avoid a dangerous situation, an automatic driving system can incorporate this digital rule as a high cost into the planning module. To prevent the autonomous driving system from making dangerous decisions, a severe penalty will be imposed if this rule is violated during an action sequence.

Due to data measurement errors and other uncertainties, the evaluation results will fluctuate when digital traffic rules are implemented. How to avoid this type of influence, improve the robustness of traffic rules evaluation, and reduce the result's sensitivity to the input data is a very important and worthwhile topic for in-depth study. In the future, we will conduct additional research in this field.

9. Conclusions

The digitization of traffic rules is a crucial field for the compatibility of autonomous vehicles with traffic rules. Using incorrect digital traffic rules will have a severe impact on traffic safety. Consequently, the premise for applying digital rules is that the digital rules must be consistent with the natural language traffic rules for human drivers. The issue of how to ensure the consistency and correctness of digital traffic regulations is therefore crucial.

To address this issue, we propose a novel approach that combines equivalence verification with model checking. At the same time, we describe the whole design and application process for digital traffic rules, including digitization and formal verification. The designed digital rules are then implemented in a simulation environment. The evaluation and experiment results indicate that natural language traffic rules can be translated systematically into computer-understandable TL. We also demonstrate the semantic coherence and correctness of the MTL traffic rules. This work

facilitates the compatibility of autonomous vehicles with traffic regulations.

The proposed method of digitization, verification, and application experimentation contributes to current advancements in autonomous driving and expedites the realization of autonomous vehicles. Theoretically, the methodology can also be applied to other types of international traffic and driving rules.

Acknowledgments

This research was conducted jointly by the Research Institute for Road Safety of the Ministry of Public Security and Huawei Technologies Co., Ltd.

Compliance with ethics guidelines

Lei Wan, Changjun Wang, Daxin Luo, Hang Liu, Sha Ma, and Weichao Hu declare that they have no conflicts of interest or financial conflicts to disclose.

References

[1] Claybrook J, Kildare S. Autonomous vehicles: no driver...no regulation? Science 2018;361(6397):36–7.
[2] Cummings ML, Britton D. Regulating safety-critical autonomous systems: past, present, and future perspectives. In: Pak R, de Visser EJ, Rovira E, editors. Living with robots. London: Academic Press; 2019.
[3] Nair GS, Bhat CR. Sharing the road with autonomous vehicles: perceived safety and regulatory preferences. Transp Res Part C 2021;122:102885.
[4] Otter DW, Medina JR, Kalita JK. A survey of the usages of deep learning for natural language processing. IEEE Trans Neural Netw Learn Syst 2021;32(2):604–24.
[5] Kumar E. Natural language processing. New Delhi: IK International Pvt Ltd; 2011.
[6] Grishman R. Computational linguistics: an introduction. Cambridge: Cambridge University Press; 1986.
[7] Liu X, Wu D. From natural language to programming language. In: Goschnick S, editor. Innovative methods, user-friendly tools, coding, and design approaches in people-oriented programming. Hershey: IGI Global; 2018.
[8] Manna Z, Pnueli A. The temporal logic of reactive and concurrent systems. Berlin: Springer; 1992.
[9] Pnueli A. The temporal logic of programs. In: Proceedings of the 18th Annual Symposium on Foundations of Computer Science (sfcs 1977); 1977 Oct 31–Nov 1; Providence, RI, USA. New York City: IEEE; 1977. p. 46–57.
[10] Clarke EM, Emerson EA. Design and synthesis of synchronization skeletons using branching time temporal logic. In: Proceedings of the Workshop on Logic of Programs; 1981 May 4–6; New York City, NY, USA. Berlin: Springer; 1981. p. 52–71.
[11] Emerson EA, Halpern JY. "Sometimes" and "not never" revisited: on branching versus linear time temporal logic. J Assoc Comput Mach 1986;33(1):151–78.
[12] Alur R, Henzinger TA. A really temporal logic. J Assoc Comput Mach 1994;41(1):181–203.
[13] Alur R, Henzinger TA. Real-time logics: complexity and expressiveness. Inf Comput 1993;104(1):35–77.
[14] Koymans R. Specifying real-time properties with metric temporal logic. Real Time Syst 1990;2(4):255–99.
[15] Alur R, Feder T, Henzinger TA. The benefits of relaxing punctuality. J Assoc Comput Mach 1996;43(1):116–46.
[16] Ouaknine J, Worrell J. On the decidability of metric temporal logic. In: Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science (LICS'05); 2005 Jun 26–29; Washington, DC, USA. New York City: IEEE; 2005. p. 188–97.
[17] Konur S. A survey on temporal logics for specifying and verifying real-time systems. Front Comput Sci 2013;7(3):370–403.
[18] Rizaldi A, Althoff M. Formalising traffic rules for accountability of autonomous vehicles. In: Proceedings of the 2015 IEEE 18th International Conference on Intelligent Transportation Systems; 2015 Sep 15–18; Gran Canaria, Spain. New York City: IEEE; 2015. p. 1658–65.
[19] Esterle K, Gressenbuch L, Knoll A. Formalizing traffic rules for machine interpretability. In: Proceedings of the 2020 IEEE 3rd Connected and Automated Vehicles Symposium (CAVS); 2020 Oct 4–5; Victoria, BC, Canada. New York City: IEEE; 2020. p. 1–7.
[20] Maierhofer S, Rettinger AK, Mayer EC, Althoff M. Formalization of interstate traffic rules in temporal logic. In: Proceedings of the 2020 IEEE Intelligent Vehicles Symposium (IV); 2020 Oct 19–Nov 13; Las Vegas, NV, USA. New York City: IEEE; 2020. p. 752–9.
[21] Karimi A, Duggirala PS. Formalizing traffic rules for uncontrolled intersections. In: Proceedings of the 2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS 2020); 2020 Apr 21–25; Sydney, NSW, Australia. New York City: IEEE; 2020. p. 41–50.
[22] Beck H, Eiter T, Krennwallner T. Inconsistency management for traffic regulations: formalization and complexity results. In: Proceedings of the 13th European Workshop on Logics in Artificial Intelligence; 2012 Sep 26–28; Toulouse, France. Berlin: Springer; 2012. p. 80–93.
[23] Krasowski H, Althoff M. Temporal logic formalization of marine traffic rules. In: Proceedings of the 2021 IEEE Intelligent Vehicles Symposium (IV); 2021 Jul 11–17; Nagoya, Japan. New York City: IEEE; 2021. p. 186–92.
[24] Esterle K, Aravantinos V, Knoll A. From specifications to behavior: maneuver verification in a semantic state space. In: Proceedings of the 2019 IEEE Intelligent Vehicles Symposium (IV); 2019 Jun 9–12; Paris, France. New York City: IEEE; 2019. p. 2140–7.
[25] Hannah D, Edwards P, Khastgir S. Extended updated proposal for an approach to defining rules of the road. In: Proceedings of the 27th UNECE Functional Requirements for Automated and Autonomous Vehicles (FRAV) Session; 2022 Apr 19–20; online. Coventry: The University of Warwick; 2022.
[26] Bjesse P. What is formal verification? ACM SIGDA Newsletter 2005;35(24):1–34.
[27] Paulson LC. The foundation of a generic theorem prover. J Autom Reason 1989;5(3):363–97.
[28] Clarke EM, Grumberg O, Kroening D, Peled D, Veith H. Model checking. 2nd ed. Cambridge: The MIT Press; 2018.
[29] Baier C, Katoen JP. Principles of model checking. Cambridge: The MIT Press; 2008.
[30] Visser W, Havelund K, Brat G, Park SJ, Lerda F. Model checking programs. Autom Softw Eng 2003;10(2):203–32.
[31] Huang SY, Cheng KTT. Formal equivalence checking and design debugging. Berlin: Springer Science & Business Media; 2012.
[32] Alur R, Henzinger TA. Logics and models of real time: a survey. In: Proceedings of the Symposium of the REX Project (Research and Education in Concurrent Systems); 1991 Jun 3–7; Mook, The Netherlands. Berlin: Springer; 1991. p. 74–106.
[33] Ouaknine J, Worrell J. Some recent results in metric temporal logic. In: Proceedings of the 6th International Conference on Formal Modeling and Analysis of Timed Systems; 2008 Sep 15–17; Saint Malo, France. Berlin: Springer; 2008. p. 1–13.
[34] National People's Congress. [Road traffic safety law of the People's Republic of China]. Beijing: Standing Committee of the National People's Congress; 2003. Chinese.
[35] State Council of the People's Republic of China. [Regulations on the implementation of the road traffic safety law of the People's Republic of China]. Beijing: Standing Committee of the National People's Congress; 2004. Chinese.
[36] The Ministry of Public Security of the People's Republic of China. GA/T 1773.1-2021: Operating specifications for safe and civilized motor vehicle drivers. Beijing: The Ministry of Public Security of the People's Republic of China; 2021. Chinese.
[37] Ulmasovna ES. The principal use of propositional and suppositional terms in the sentences. JournalNX 2020;6(11):242–3.
[38] Scholtes M, Westhofen L, Turner LR, Lotto K, Schuldes M, Weber H, et al. 6-layer model for a structured description and categorization of urban traffic and environment. IEEE Access 2021;9:59131–47.
[39] Van Winsum W, Heino A. Choice of time-headway in car-following and the role of time-to-collision information in braking. Ergonomics 1996;39(4):579–92.
[40] Ayres TJ, Li L, Schleuning D, Young D. Preferred time-headway of highway drivers. In: Proceedings of the ITSC 2001. 2001 IEEE Intelligent Transportation Systems; 2001 Aug 25–29; Oakland, CA, USA. New York City: IEEE; 2001. p. 826–9.
[41] Rizaldi A, Keinholz J, Huber M, Feldle J, Immler F, Althoff M, et al. Formalising and monitoring traffic rules for autonomous vehicles in Isabelle/HOL. In: Proceedings of the International Conference on Integrated Formal Methods; 2017 Sep 20–22; Turin, Italy. Turin, Italy. Berlin: Springer; 2017. p. 50–66.
[42] Schnoebelen P. The complexity of temporal logic model checking. In: Proceedings of the 4th Advances in Modal Logic Conference (AiML 2002); 2002 Sep 30–Oct 2; Toulouse, France. London: King's College Publications; 2003.
[43] Habrias H, Frappier M. Software specification methods—an overview using a case study. Berlin: Springer; 2012.
[44] Kuppe MA, Lamport L, Ricketts D. The TLA$^+$ toolbox. 2019. arXiv:1912.10633.
[45] Lamport L. Specifying systems: the TLA$^+$ language and tools for hardware and software engineers. Boston: Addison-Wesley Professional; 2002.
[46] O'Regan G. Concise guide to formal methods. New York City: Springer; 2017.
[47] Foster H, Uchitel S, Magee J, Kramer J. LTSA-WS: a tool for model-based verification of web service compositions and choreography. In: Osterweil LJ, Rombach HD, Soffa ML, editors. Proceedings of the 28th International Conference on Software Engineering; 2006 May 20–28; Shanghai, China; 2006. p. 771–4.
[48] Holzmann GJ. The model checker SPIN. IEEE Trans Softw Eng 1997;23(5):279–95.
[49] Kwiatkowska M, Norman G, Parker D. PRISM: probabilistic symbolic model checker. In: Proceedings of the International Conference on Modelling Techniques and Tools for Computer Performance Evaluation; 2002 Apr 14–17. London, UK. Berlin: Springer; 2002. p. 200–4.

[50] Association for Standardization of Automation and Measuring Systems (ASAM e.V.). ASAM OpenDRIVE® V1.8.0 [Internet]. Hoehenkirchen: ASAM e.V.; 2021 Aug 3 [cited 2023 May 13]. Available from: https://www.asam.net/standards/detail/opendrive/
[51] Association for Standardization of Automation and Measuring Systems (ASAM e.V.). ASAM OpenSCENARIO® V1.2.0 [Internet]. Hoehenkirchen: ASAM e.V.; 2022 May 13 [cited 2023 May 13]. Available from: https://www.asam.net/standards/detail/openscenario/
[52] Wu Z, Yao M, Ma H, Jia W. Improving accuracy of the vehicle attitude estimation for low-cost INS/GPS integration aided by the GPS-measured course angle. IEEE Trans Intell Transp Syst 2013;14(2):553–64.
[53] Xia X, Hashemi E, Xiong L, Khajepour A, Xu N. Autonomous vehicles sideslip angle estimation: single antenna GNSS/IMU fusion with observability analysis. IEEE Internet Things J 2021;8(19):14845–59.
[54] Lynch KM, Park FC. Modern robotics. Cambridge: Cambridge University Press; 2017.