Xu J, Huang Y, Shi J *et al.* A multi-agent spatial logic for scenario-based decision modeling and verification in platoon systems. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 36(6): 1231–1247 Nov. 2021. DOI 10.1007/s11390-021-1565-8

A Multi-Agent Spatial Logic for Scenario-Based Decision Modeling and Verification in Platoon Systems

Jingwen Xu[1], *Member, CCF*, Yanhong Huang[1], Jianqi Shi[1,*], and Shengchao Qin[2], *Senior Member, ACM, IEEE*

[1] *National Trusted Embedded Software Engineering Technology Research Center, East China Normal University Shanghai 200062, China*

[2] *College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518061, China*

E-mail: 51194501198@stu.ecnu.edu.cn; {yhhuang, jqshi}@sei.ecnu.edu.cn; sqin@szu.edu.cn

Received May 6, 2021; accepted October 19, 2021.

Abstract To cater for the scenario of coordinated transportation of multiple trucks on the highway, a platoon system for autonomous driving has been extensively explored in the industry. Before such a platoon is deployed, it is necessary to ensure the safety of its driving behavior, whereby each vehicle's behavior is commanded by the decision-making function whose decision is based on the observed driving scenario. However, there is currently a lack of verification methods to ensure the reliability of the scenario-based decision-making process in the platoon system. In this paper, we focus on the platoon driving scenario, whereby the platoon is composed of intelligent heavy trucks driving on cross-sea highways. We propose a formal modeling and verification approach to provide safety assurance for platoon vehicles' cooperative driving behaviors. The existing Multi-Lane Spatial Logic (MLSL) with a dedicated abstract model can express driving scene spatial properties and prove the safety of multi-lane traffic maneuvers under the single-vehicle perspective. To cater for the platoon system's multi-vehicle perspective, we modify the existing abstract model and propose a Multi-Agent Spatial Logic (MASL) that extends MLSL by relative orientation and multi-agent observation. We then utilize a timed automata type supporting MASL formulas to model vehicles' decision controllers for platoon driving. Taking the behavior of a human-driven vehicle (HDV) joining the platoon as a case study, we have implemented the model and verified safety properties on the UPPAAL tool to illustrate the viability of our framework.

Keywords autonomous driving, decision-making model, platoon system, safety verification, timed automaton

1 Introduction

Autonomous driving technology has undergone tremendous development over the years [1]. It has been laid out in multiple applicable fields in the blueprint for the future. As a kind of safety-critical systems, ensuring the vehicle's safety during driving is a primary consideration. Since the decision-making function is the key to autonomous driving, many researchers have adopted formal methods to verify the reliability of driving decisions in different scenarios, such as lane-changing [2, 3], crossing [4], turning [5] maneuvers, car following [6], and vehicle interaction [7]. These studies only consider a single vehicle's driving behavior, and rarely focus on the collaborative control behavior of multiple vehicles. As one of the most promising commercial applications in the autonomous industry, automated truck transportation has attracted much attention. Hence, it is vital to ensure the safety of multi-vehicle cooperative driving behavior in platoon scenarios.

In our platoon driving scenario, multiple intelligent heavy-duty trucks complete unified transportation tasks via communication. The platoon composed of these trucks can be regarded as a multi-agent system [8].

Regular Paper

Special Section on Software Systems 2021—Theme: Dependable Software Engineering

This work was partially supported by the National Key Research and Development Program of China under Grant No. 2019YF-B2102602.

*Corresponding Author

These trucks are equipped with a 5G-based intelligent truck system developed by SAIC Motor. They drive across the East Sea Bridge in the platoon mode and complete such tasks with level-4 autonomous driving, centimeter-accuracy positioning, and interaction with automatic equipments. As Fig.1 shows, driving behaviors in the platoon mode include joining, leaving, splitting, merging, and maintaining. For these collaborative behaviors, a verifiable agent-based architecture [9] was proposed for deploying a safety platoon system. Other formalization work verified the correctness of the platooning protocol [10–12]. Without considering the impact of driving scenarios, their methods only focus on platoon vehicles' interactive behaviors, which is less expressive and limited in specific situations.

Considering the significance of driving scenes to decision-making, Hilscher *et al.* first proposed Multi-Lane Spatial Logic (MLSL) [2] to formalize the spatial properties of driving scenes and to verify the safety of the decision controller. It turns out to be an effective way because it uses an abstract model to capture the scenario's characteristics, and MLSL is conducive to logical reasoning and formal modeling. However, their work is limited to verifying driving behaviors from the perspective of a single-vehicle observation and cannot handle a platoon system with multi-vehicle observation.

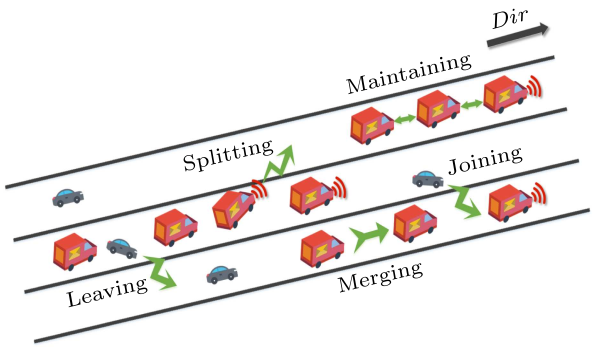


Fig.1. Different platoon driving behaviors on highways.

In this paper, we focus on the platoon transportation of smart heavy trucks on cross-sea expressways, and propose a formal modeling and verification approach to offer safety assurance for its scenario-based decision-making functions. Fig.2 shows our modeling and verification framework, and the contributions are as follows.

• We enhance an existing abstract model [2] to characterize the platoon driving scenario. Based on the

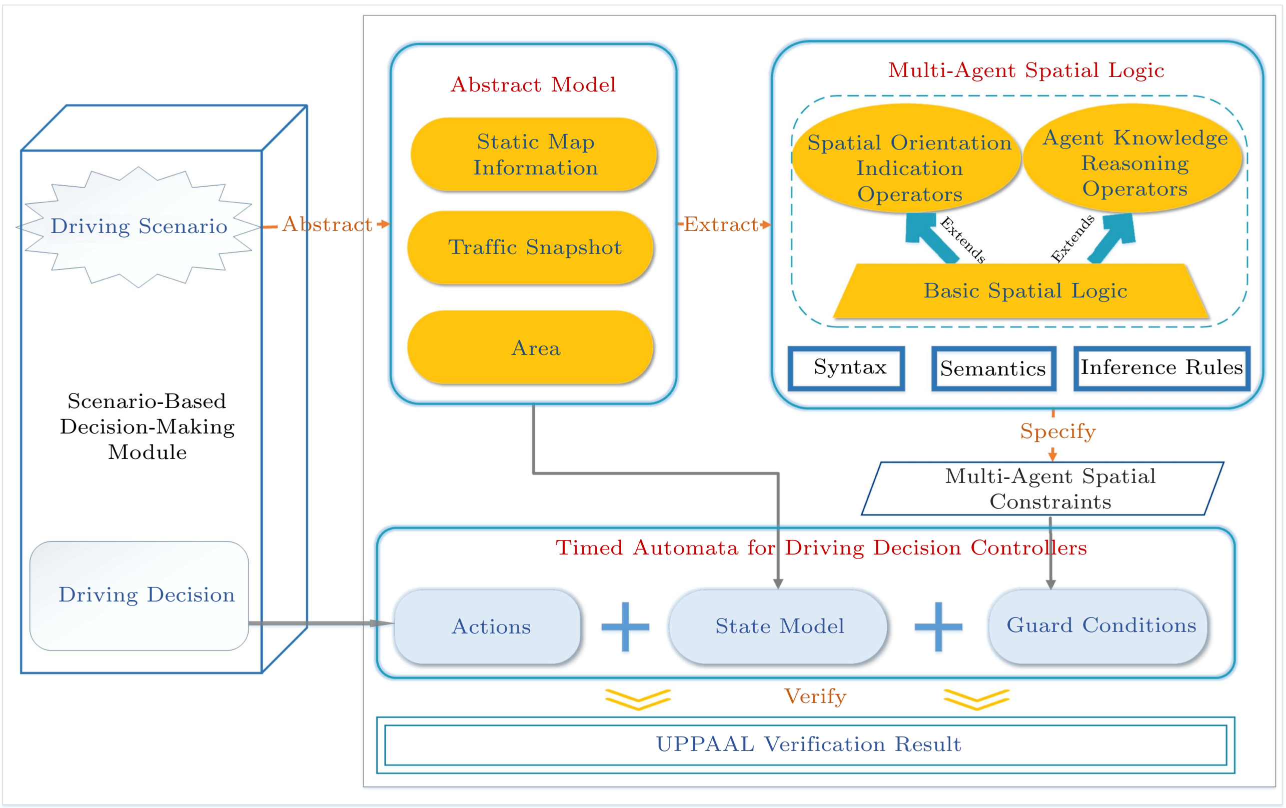

Driving Scenario
Abstract
Scenario-Based Decision-Making Module
Driving Decision
Abstract Model
Static Map Information
Traffic Snapshot
Area
Extract
Multi-Agent Spatial Logic
Spatial Orientation Indication Operators
Agent Knowledge Reasoning Operators
Extends
Extends
Basic Spatial Logic
Syntax
Semantics
Inference Rules
Specify
Multi-Agent Spatial Constraints
Timed Automata for Driving Decision Controllers
Actions
State Model
Guard Conditions
Verify
UPPAAL Verification Result

Fig.2. Formal modeling and verification framework.

scene constructed by the abstract model, MLSL is then modified into our basic spatial logic, which describes the spatial scene specification in general driving situations.

• We propose a Multi-Agent Spatial Logic (MASL) by extending the basic spatial logic from the perspective of relative orientation and multi-agent observation. The MASL formula specifying multi-agent spatial constraints in driving scenes is a formal expression of guard conditions for decision-making.

• We propose a formal modeling method applicable for platoon maneuvers, which adopts timed automata to construct the vehicle's decision controller. In the case of joining platoon, we can use MASL formulas as the model's guard conditions to determine whether a car can safely cut between two platoon members. By implementing the interactive vehicles' controllers as a network of timed automata model on the UPPAAL tool, we verify multi-vehicle cooperative driving behaviors, thereby obtaining a reliable decision-making system for the platoon.

The rest of the paper is organized as follows. In Section 2, we define an abstract model and the basic spatial logic. Then, we propose MASL and formally define its syntax, semantics, and inference rules in Section 3. Section 4 introduces a timed automata type to construct the vehicle's decision-making controller. Section 5 implements and verifies the network of timed automata model for multi-vehicle coordinated control based on a specific case. Section 6 reviews related work, with concluding remarks and future work in Section 7.

2 Basic Spatial Logic and Its Dedicated Abstract Model

The proposed MLSL adapted to a dedicated abstract model is a logic language expressing spatial properties on multi-lane motorways where all cars are driving in one direction[4]. Since MLSL can only verify vehicle behaviors from the perspective of a single vehicle's observation, it is not suitable for the platoon scenario with multi-vehicle collaborative control. To overcome this limitation, we improve the existing abstract model according to the platoon scene's characteristics on the cross-sea highway, and modify MLSL correspondingly into our basic spatial logic, in which the representation of spatial properties is similar to that in MLSL, but can be further extended in terms of relative orientation and multi-agent knowledge reasoning. In this section, we first define an abstract model depicting important features of road traffic. Using the abstract model as the carrier, we then define the basic spatial logic to formally represent driving scene spatial constraints.

2.1 Abstract Model

It is vital to characterize the driving scenarios so that the features can facilitate properties specification and reasoning. For the scene factors considered in the decision-making process of platoon control maneuver, we define the driving scenario as an abstract model, which consists of the following parts: static map information, traffic snapshot, and observation area.

2.1.1 *Static Map Information*

Mapping the real world's road structure to a two-dimensional coordinate system, static map information describes some fixed information such as road division, orientation, connectivity, and road range. As shown in Fig.3, the horizontal axis (x-axis) represents the discrete lanes, and the vertical axis (y-axis) indicates the road extension. A global map is defined as a structure $Map = (Lane, Dir, Begin, End)$.

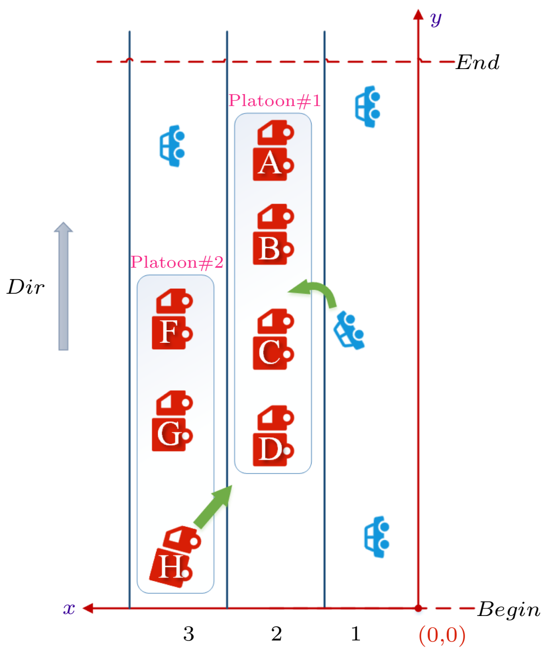


Fig.3. Vehicles driving on the two-dimensional map.

• *Lane* is the set of lanes on highways, represented as an integer interval. Note that our research scenario is a cross-sea bridge, which is a two-way six-lane highway. For convenience, we only consider the three straight lanes in one direction.

• Dir is the lane direction in which vehicles are allowed to travel. We have $Dir = \{+1, -1\}$ indicating positive and negative ordinate direction respectively. In addition, we define auxiliary functions $lft(lane, dir)$, $rgt(lane, dir)$, to calculate the left and the right adjacent lane of a specific lane $lane \in Lane$ in direction $dir \in Dir$, respectively. As shown in Fig.3, lane 2 satisfies: $lft(2, +1) = 3$, $rgt(2, +1) = 1$.

• $Begin : y = 0$ and $End : y = LEN$ are the start and the end of the lane respectively, which limit vehicles' driving range in the vertical direction. They are functions of the y-axis, and LEN is the road length.

2.1.2 Traffic Snapshot

Traffic snapshot records vehicles' traffic information on Map at a given point in time, where the data is collected from vehicles' cameras, lidar, gyroscope, and other sensing devices. Since the traffic information is continuously changing, we record it once every cycle and use vehicle-related functions to represent it.

Let $Vehl$ be the set of vehicles currently traveling on Map, including trucks in the platoon and non-queue cars. Traffic snapshot is a tuple composed of multiple functions. The function's parameter indicates a specific vehicle $C \in Vehl$, and the return value is the car's current traffic information TS, a tuple of the form $(res, pos, inplt, pre, suc, spd, acc, len, pltFunc)$.

• $res(C) : Vehl \to \mathcal{P}(Lane)$ returns the set of lanes currently occupied by car C. Vehicles traveling straight only occupy one lane, but during lane-changing, vehicles can occupy two lanes simultaneously.

• $pos(C) : Vehl \to \mathcal{R}$ returns the ordinate of car C's rear position.

• $inplt(C) : Vehl \to \{\text{true}, \text{false}\}$ returns true if car C is currently in the platoon; false otherwise.

• $pre(C) : Vehl \to Vehl$ returns the predecessor vehicle of car C in the platoon.

• $suc(C) : Vehl \to Vehl$ returns the successor vehicle of car C in the platoon.

• $spd(C) : Vehl \to \mathcal{R}$ returns the current speed of car C and its unit is m/s.

• $acc(C) : Vehl \to \mathcal{R}$ returns the current acceleration of car C and its unit is m/s^2.

• $len(C) : Vehl \to \mathcal{R}$ calculates the length of the vehicle's land occupation.

• $pltFunc(C) ::= join(C)|leave(C)|split(C)|...$ is an extensible function representing the platoon behaviors that car $C \in Vehl$ intends to perform. In different platoon driving modes, $pltFunc$ can be assigned to the corresponding function, which is defined according to the characteristics of the behavior. Since the case study in Section 5 is about the behavior of a human-driven vehicle (HDV) joining the platoon, here we let $pltFunc := join$.

$join(C) : Vehl \to Lane$ returns the lane number car C claims to change for joining the platoon.

2.1.3 Observation Area

The observation area is a rectangular observation segment with a fixed length and width on Map, which is described by a tuple $Area = (L, X)$, where

• $L = [l, n]$, $L \subseteq Lane$ is the set of lanes in the area's horizontal direction, represented as an integer interval; $|L|$ is the number of lanes occupied by the area;

• $X = [r, f]$, $r, f \in \mathbb{R}$, a real interval extending along the Dir direction. r and f are ordinates of the area's start point and endpoint respectively, and $|X|$ represents the length of the area.

We abbreviate area $A^{[l,n]}_{[r,f]}$ as A or $A^{[l,n]}$ or $A_{[r,f]}$, where the superscript indicates the lane interval and the subscript indicates the vertical road extension.

2.2 Basic Spatial Logic

Since we have formalized road traffic as an abstract model, we will use it as a carrier to study the spatial characteristics of general driving situations. In this subsection, we define the syntax and semantics of the basic spatial logic. Given the current traffic snapshot TS and a specific area $A^{[l,n]}_{[r,f]}$ that can be observed by the vehicle ego, we use the basic spatial logic formula ϕ to describe vehicles' spatial distribution in the area.

Definition 1 (Syntax of Basic Spatial Logic). *The syntax of the basic spatial logic formula ϕ is defined by*

$$\phi ::= \text{true}|\alpha = \beta|re(c)|cl(c)|\phi_1 \wedge \phi_2|\neg\phi|\phi_1 \frown \phi_2|^{\phi_1}_{\phi_2}|gap,$$

where α, β, c are variables of car identifiers from $Vehl$.

To define the semantics of the basic spatial logic, we construct a context model $\mathcal{M} = (TS, A, ego, \boldsymbol{v})$ concerning a traffic snapshot TS, an observer ego with its observation area A, and a valuation function $\boldsymbol{v}$. The valuation $\boldsymbol{v} : CVar \to Vehl$ is a function mapping car variables $\alpha, \beta, c, ... \in CVar$ to actual car identifiers in the set $Vehl$.

Definition 2 (Semantics of Basic Spatial Logic). *The satisfaction relation between a context model $\mathcal{M} = (TS, A, ego, \boldsymbol{v})$ and a basic spatial logic formula ϕ, $\mathcal{M} \models \phi$, is defined inductively over ϕ as follows*:

1) *true*: *it is satisfied under any context.*

$$\mathcal{M} \models true \quad for\ all\ (TS, A, ego, \boldsymbol{v}).$$

2) $\alpha = \beta$: *the values of variables α and β are equal.*

$$\mathcal{M} \models \alpha = \beta \quad \Leftrightarrow \quad \boldsymbol{v}(\alpha) = \boldsymbol{v}(\beta).$$

3) $re(c)$: *area $A^{[l,n]}_{[r,f]}$ is currently occupied by car c. I_A is the set of vehicles occupying area A, res_A is the reservation of a specific car in area A, and len_A is the intersection interval between the specified car's occupancy range and the ordinate of area A.*

$$\mathcal{M} \models re(c) \quad \Leftrightarrow \quad |L| = 1 \wedge |X| > 0 \wedge \boldsymbol{v}(c) \in I_A \\ \wedge\, res_A(\boldsymbol{v}(c)) = L \wedge len_A(\boldsymbol{v}(c)) = X.$$

4) $cl(c)$: *area $A^{[l,n]}_{[r,f]}$ is declared by car c for joining the platoon. I_A is the set of vehicles that declare to change location to area A, $join_A$ is a certain car's lane-change declaration for joining the platoon in area A, and len_A is the intersection interval between the car's declared position range and the ordinate of area A.*

$$\mathcal{M} \models cl(c) \quad \Leftrightarrow \quad |L| = 1 \wedge |X| > 0 \wedge \boldsymbol{v}(c) \in I_A \\ \wedge\, join_A(\boldsymbol{v}(c)) = L \wedge len_A(\boldsymbol{v}(c)) = X.$$

5) $\phi_1 \wedge \phi_2$: *ϕ_1 and ϕ_2 are both satisfied in area $A^{[l,n]}_{[r,f]}$.*

$$\mathcal{M} \models \phi_1 \wedge \phi_2 \quad \Leftrightarrow \quad \mathcal{M} \models \phi_1 \;\wedge\; \mathcal{M} \models \phi_2.$$

6) $\neg\phi$: *the given model does not satisfy formula ϕ.*

$$\mathcal{M} \models \neg\phi \quad \Leftrightarrow \quad \neg\, TS, A, ego, \boldsymbol{v} \models \phi.$$

7) $\phi_1 \frown \phi_2$: *area $A = A_{[r,f]}$ is vertically divided into two sub-areas $A_1 = A_{[r,k]}$ and $A_2 = A_{[k,f]}$ along the increasing direction of the ordinate. Sub-area A_1 satisfies ϕ_1, and sub-area A_2 satisfies ϕ_2, if and only if area A satisfies $\phi_1 \frown \phi_2$.*

$$\mathcal{M} \models \phi_1 \frown \phi_2 \quad \Leftrightarrow \quad \exists k \in \mathbb{R} : r \leqslant k \leqslant f \\ \wedge\, TS, A_{[r,k]}, ego, \boldsymbol{v} \models \phi_1 \\ \wedge\, TS, A_{[k,f]}, ego, \boldsymbol{v} \models \phi_2.$$

8) $\frac{\phi_1}{\phi_2}$: *area $A = A^{[l,n]}$ is horizontally divided into two sub-areas $A_1 = A^{[l,m]}$ and $A_2 = A^{[m+1,n]}$ along the lane direction. Sub-area A_1 satisfies ϕ_1, and sub-area A_2 satisfies ϕ_2, if and only if area A satisfies $\frac{\phi_1}{\phi_2}$.*

$$\mathcal{M} \models \frac{\phi_1}{\phi_2} \quad \Leftrightarrow \quad \exists m \in \mathcal{N} : l \leqslant m \leqslant n-1 \\ \wedge\, TS, A^{[l,m]}, ego, \boldsymbol{v} \models \phi_1 \\ \wedge\, TS, A^{[m+1,n]}, ego, \boldsymbol{v} \models \phi_2.$$

9) gap: *a blank interval in area A. If there is no car occupying or declaring this area, gap indicates the whole area is empty. If there exists one or more vehicles' occupancy/declaration, we use gap to indicate blank segments between its location to the front or rear car or both ends of the area.*

$$\mathcal{M} \models gap \quad \Leftrightarrow \quad \forall c \in CVar : \\ \mathcal{M} \models \neg re(c) \;\wedge\; \mathcal{M} \models \neg cl(c).$$

Fig.4 shows that vehicle spacing is divided into five levels according to the distance, and the standard of division refers to [13]. We define $gap ::= gap_1|gap_2|gap_3|gap_4|gap_5$, where $gap_1 = [0, 5]$ is a dangerous distance that requires rapid braking, $gap_2 = (5, 20]$ represents a short distance that requires slow braking, $gap_3 = (20, 40]$ represents a close distance, $gap_4 = (40, 80]$ represents a normal distance, and

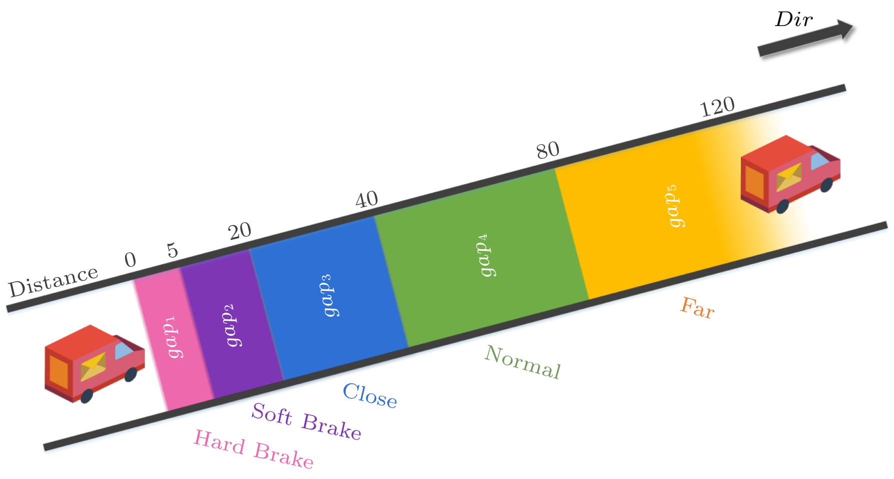


Fig.4. Different levels of gap division on the road.

$gap_5 = (80, 120]$ represents a far distance. The comparative relationship satisfies: $gap_5 > gap_4 > gap_3 > gap_2 > gap_1$.

In the following, we use a notation $\langle\phi\rangle$ to state that formula ϕ is satisfied somewhere in the considered area.

3 Multi-Agent Spatial Logic

To expand the basic spatial logic to MASL, we introduce the spatial orientation indication (abbreviated as SOI) operators and the agent knowledge reasoning (abbreviated as AKR) operators. SOI operators indicate orientation relative to the observation vehicle. AKR operators represent the individual knowledge, shared knowledge, and joint knowledge of the multi-agent system. In this section, we formally define the syntax and semantics of MASL and formulate inference rules to reason from single-agent observation information to multi-agent global knowledge.

3.1 Spatial Orientation Indication Operators

SOI operators indicate spatial orientations relative to the observation vehicle, including eight directions of east (E), south (S), west (W), north (N), southeast (SE), southwest (SW), northeast (NE), and northwest (NW). The formula composed of SOI operators followed by ϕ expresses the area's observation information, where the observation area is specified by relative orientation.

Definition 3 (Syntax of SOI Operators). *The syntax of spatial orientation indication operators is defined as follows*:

$$\psi ::= \Gamma\phi | \Gamma_i\phi,$$
$$\Gamma ::= N|S|W|E|NE|NW|SE|SW.$$

When considering the single-car perspective, we use Γ to represent its own relative observation direction. While focusing on the entire platoon system's observation angle, we use Γ_i to represent the observation direction relative to a certain platoon member.

For the Γ operator concerning single-car perspective, we construct a context model $\mathcal{M}_S = (TS, A, ego, \boldsymbol{v})$ to explain ψ. As Fig.5 shows, we take car C as the observer to detect regional traffic scenario in its northeast. Since the multi-agent perspective is a combination of all members' single-agent perspective, we define variable $P = (\alpha, \beta, c, ...)$ to indicate the platoon, where $\alpha, \beta, c \in CVar$ represent platoon members variables. Here we use P_i to represent the i-th member vehicle in platoon P, and thus P_1 represents the leading vehicle. We also define a distributed area $\mathbb{A}$ composed of all members' observation areas. Thus, to realize the transition from a single agent to the platoon system, we update the previous context model to obtain the multi-agent model $\mathcal{M}_P = (TS, \mathbb{A}, P, \boldsymbol{v})$, where the subscript of $\mathcal{M}_P$ represents the platoon variable P.

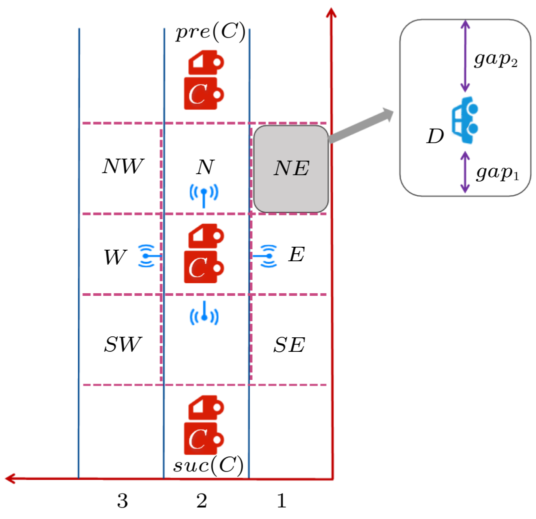


Fig.5. Regional traffic scene observed by car C.

Before giving the semantics of SOI operators, we introduce the following auxiliary functions, which are the calculation of the traffic snapshot's return value.

• $getArea(C)$ calculates car C's overall observation area, which covers its occupied segment and the surrounding detectable area extended in eight directions. It returns area-type $A^{[l,n]}_{[r,f]}$.

• $getAreas(P)$ is the combination of the respective observation areas of all members in platoon P.

• $F(C, \Gamma)$ calculates the observation area in a specific direction relative to the occupied segment of car C, where Γ specifies the orientation and the function returns area-type $A^{[l,n]}_{[r,f]}$.

Definition 4 (Semantics of SOI Operators). *Given a context model for a single agent $\mathcal{M}_S = (TS, A, ego, \boldsymbol{v})$ with $A = getArea(\boldsymbol{v}(ego))$, and a context model for multiple agents $\mathcal{M}_P = (TS, \mathbb{A}, P, \boldsymbol{v})$ with $\mathbb{A} = getAreas(P)$, then, the satisfaction of a formula with SOI operators is defined inductively as follows*:

1) $\Gamma\phi$: *the regional driving scene observed by car ego in its Γ direction can be expressed by formula $\Gamma\phi$, where the observation area A' is calculated by function $F(\boldsymbol{v}(ego), \Gamma)$*:

$$TS, A, ego, \boldsymbol{v} \models \Gamma\phi \Leftrightarrow TS, A', ego, \boldsymbol{v} \models \phi \wedge A' = F(\boldsymbol{v}(ego), \Gamma);$$

2) $\Gamma_i\phi$: *$TS, \mathbb{A}, P, \boldsymbol{v} \models \Gamma_i\phi$ is equivalent to the case of a single agent $TS, A, p, \boldsymbol{v} \models \Gamma\phi$, where observer p is the i-th member in platoon P, i.e., $p = P_i$, with its observation area $F(\boldsymbol{v}(p), \Gamma)$.*

$$TS, \mathbb{A}, P, \boldsymbol{v} \models \Gamma_i\phi \quad \Leftrightarrow \quad TS, A, p, \boldsymbol{v} \models \Gamma\phi$$
$$\wedge\ p = P_i \wedge A = getArea(\boldsymbol{v}(p)) \wedge A \in \mathbb{A}.$$

When vehicles in the multi-agent system share their observation results for communication, analyzing the specific meaning of the information helps infer important knowledge and discover problems. Thus, we introduce comparison operators $=, >, <, \neq$ to compare the relationship between different spatial logic formulas.

The shaded blocks shown in Fig.6 are the overlapping areas that can be observed by the front and rear cars simultaneously. The rear car's NE position is equivalent to the SE position of the previous car, and other situations are defined below. $\psi_1 = \psi_2$ is satisfied if and only if the scene information for the same observation area is consistent.

$$TS, \mathbb{A}, P, \boldsymbol{v} \models \ \psi_1 = \psi_2 \quad \Leftrightarrow$$
$$\psi_1 = NE_i\phi_1 \wedge \psi_2 = SE_j\phi_2 \wedge \phi_1 = \phi_2 \wedge i = j + 1 \text{ or}$$
$$\psi_1 = NW_i\phi_1 \wedge \psi_2 = SW_j\phi_2 \wedge \phi_1 = \phi_2 \wedge i = j + 1 \text{ or}$$
$$\psi_1 = N_i\phi_1 \wedge \psi_2 = S_j\phi_2 \wedge \phi_1 = \phi_2 \wedge i = j + 1.$$

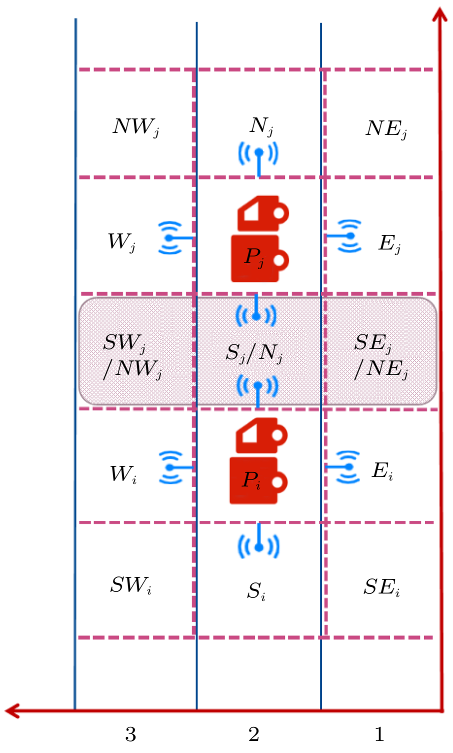


Fig.6. Overlapping area of both cars' view.

If the observation information of the same area is inconsistent, we believe that the perception of distance by different observers has been biased. To correct this inconsistency, we stipulate that the closer the perceived vehicle spacing, the higher the risk of collision. Therefor the closer distance is used as the correction standard. Thus, when comparing the observation formulas of different vehicles on the same area, it is necessary to compare the level of *gap* in the formulas. The smaller the *gap* level, the smaller the corresponding formula ψ, which means a more urgent situation.

$$TS, \mathbb{A}, P, \boldsymbol{v} \models \ \psi_1 < \psi_2 \quad \Leftrightarrow$$
$$\psi_1 = NE_i\phi_1 \wedge \psi_2 = SE_j\phi_2 \wedge \phi_1 < \phi_2 \wedge i = j + 1 \text{ or}$$
$$\psi_1 = NW_i\phi_1 \wedge \psi_2 = SW_j\phi_2 \wedge \phi_1 < \phi_2 \wedge i = j + 1 \text{ or}$$
$$\psi_1 = N_i\phi_1 \wedge \psi_2 = S_j\phi_2 \wedge \phi_1 < \phi_2 \wedge i = j + 1,$$

where ϕ_1 and ϕ_2 both contain *gap*, and the relationship of *gap* satisfies: $gap_1 < gap_2 < gap_3 < gap_4 < gap_5$.

If the observation information provided by two adjacent vehicles points to two different regions, then $\psi_1 \neq \psi_2$.

3.2 Agent Knowledge Reasoning Operators

Relying on the vehicle-to-vehicle (V2V) communication technology, vehicles in the multi-agent system can mutually confirm the observation information of other members and collect it into global knowledge. In this subsection, we define AKR operators to indicate the knowledge acquisition degree of different platoon members, where the knowledge refers to spatial observation information. We specifically define three operators to represent individual knowledge, shared knowledge, and joint knowledge of platoon, respectively.

Definition 5 (Syntax of AKR Operators). *The syntax of agent knowledge reasoning operators is defined as follows*:

$$\sigma ::= \Sigma\psi,$$
$$\Sigma ::= K_i | E_G | D_G.$$

Definition 6 (Semantics of AKR Operators). *Given a context model for multiple agents $\mathcal{M}_P = (TS, \mathbb{A}, P, \boldsymbol{v})$, the satisfaction of a formula with AKR operator is defined inductively as follows*:

1) *$K_i\psi$ indicates the individual knowledge, where the spatial information ψ is recognized or obtained by platoon member P_i within its own field of vision.*

$$TS, \mathbb{A}, P, \boldsymbol{v} \models K_i\psi \quad \Leftrightarrow \quad TS, A, p, \boldsymbol{v} \models \psi$$
$$\wedge\ p = P_i \wedge \psi = \Gamma_i\phi \wedge A = getArea(\boldsymbol{v}(p)) \wedge A \in \mathbb{A}.$$

2) *$E_G\psi$ represents the shared knowledge, where the spatial information ψ can be observed simultaneously by each agent in sub-queue G and G is formed by several consecutive vehicles in platoon P. When two adjacent vehicles share their information, they reach a consensus only when the observation area is consistent and the information is the same.*

$$TS, \mathbb{A}, P, \boldsymbol{v} \models E_G\psi \quad \Leftrightarrow \quad \forall p \in G: \\ TS, \mathbb{A}, P, \boldsymbol{v} \models K_i\psi \ \wedge \ p = P_i \wedge G \subseteq P.$$

3) *$D_G\psi$ indicates the joint knowledge. It is an aggregation of knowledge that all agents acquire in sub-queue G. Through communication and information integration, all the platoon members can finally obtain global knowledge. Thus, the decision controller can direct vehicles to drive cooperatively based on joint knowledge.*

$$TS, \mathbb{A}, P, \boldsymbol{v} \models D_G\psi \quad \Leftrightarrow \quad \forall p, q \in G, p = P_i, q = P_j: \\ TS, \mathbb{A}, P, \boldsymbol{v} \models K_i\psi \quad \rightarrow \quad TS, \mathbb{A}, P, \boldsymbol{v} \models K_j\psi.$$

3.3 Inference Rules

Vehicles' observation information needs to be integrated into unified knowledge to facilitate systemic decision-making. We regard the members of the platoon as different agents. To reason from a single agent's local information to the multi-agent system's global knowledge, we formulate the following inference rules.

The conjunction rule realizes the reasoning from individual knowledge to shared knowledge. Supposing that the platoon member P_i and its front vehicle P_j satisfy $K_i(NE_i\phi)$, $K_j(SE_j\phi)$, respectively, they can recognize each other's knowledge acquisition via communication: $K_iK_j(SE_j\phi)$, $K_jK_i(NE_i\phi)$. Since their information of the same observation area is consistent, they reach a consensus on the area's scene information, which is called shared knowledge.

$$K_i(\psi_1) \wedge K_j(\psi_2) \wedge (\psi_1 = \psi_2) \wedge (i = j+1) \\ \wedge K_iK_j(\psi_2) \wedge K_jK_i(\psi_1) \implies E_G\psi,$$

where $G = (P_j, P_i)$, $\psi = \langle\psi_1\rangle$, or $G = (P_j, P_i)$, $\psi = \langle\psi_2\rangle$.

The disjunction rule infers from individual knowledge to joint knowledge. Given that the platoon member P_i and its front vehicle P_j satisfy $K_i\psi_1, K_j\psi_2$, respectively, it is necessary to determine whether their observation information points to the same area during communication. If their information is consistent, it can be used as shared knowledge or as joint knowledge. However, if their respective observation points to different regions, their observation information will be merged into joint knowledge.

1) $K_i\psi_1 \wedge K_j\psi_2 \wedge (\psi_1 = \psi_2) \wedge (i = j+1)$
$$\wedge K_iK_j\psi_2 \wedge K_jK_i\psi_1 \implies D_G\psi,$$

where $G = (P_j, P_i)$, $\psi = \langle\psi_1\rangle$ or $\psi = \langle\psi_2\rangle$.

2) $K_i\psi_1 \wedge K_j\psi_2 \wedge (\psi_1 \neq \psi_2) \wedge (i = j+1)$
$$\wedge K_iK_j\psi_2 \wedge K_jK_i\psi_1 \implies D_G\psi,$$

where $G = (P_j, P_i)$, $\psi = \langle\psi_1 \wedge \psi_2\rangle$.

When the observation results of different vehicles in the same area are inconsistent, the error correction rule helps to find faults in these vehicles' perceptions and correct them in time. According to the rule of the comparison operators, observational information with a higher risk can be used as a calibration standard. Thus, by modifying another vehicle's observation result, these vehicles can maintain information consistency.

1) $K_i(\psi_1) \wedge K_j(\psi_2) \wedge (\psi_1 < \psi_2) \wedge (i = j+1)$
$$\wedge K_iK_j(\psi_2) \wedge K_jK_i(\psi_1) \implies K_j(\psi) \wedge \psi = \langle\psi_1\rangle.$$

2) $K_i(\psi_1) \wedge K_j(\psi_2) \wedge (\psi_1 > \psi_2) \wedge (i = j+1)$
$$\wedge K_iK_j(\psi_2) \wedge K_jK_i(\psi_1) \implies K_i(\psi) \wedge \psi = \langle\psi_2\rangle.$$

In summary, our proposed MASL can always formally describe the vehicle's occupancy on the road space under any platoon driving behavior, thereby specifying driving scene constraints. Moreover, inference rules simulate the process of information exchange and integration between vehicles, which apply to the communication stages of different platoon modes.

4 Timed Automata for Decision Controller

In this section, we introduce a timed automata type supporting MASL formulas to model the vehicle's decision controller for platoon maneuvers. Although different vehicles will make corresponding driving decisions in different platoon modes, their formal modeling methods are still the same.

Definition 7 (Timed Automaton with MASL). *A timed automaton for vehicle's decision control supporting our Multi-Agent Spatial Logic is defined by a seven-tuple structure*: $TA = (Agt, Q, I, Act, G, T, q_0)$.

• *Agt identifies the agent that controls the execution of TA. In a multi-agent system composed of multiple vehicles, $Agt = P$ indicates that the executor is the platoon system, $Agt = P_i$ represents a platoon member,* *and $Agt = X$ represents the HDV that interacts with the platoon.*

• *Q is the set of states. Each state is mapped from a traffic snapshot TS at a specific moment.*

• *$I : Q \to \Phi$ is the labeling function which assigns an invariant $I(q)$ to state q. $\Phi ::= \Phi_{TS}|\Phi_D|\Phi_C|\Phi_M|\Phi_1 \wedge \Phi_2|\neg\Phi$, where Φ_{TS} is the set of the traffic snapshots TS, Φ_D is the set of data variables of other driving scene information, Φ_C is the set of clocks ranging over $\mathbb{R}$, and Φ_M is a set of MASL formulas representing multi-agent spatial constraints.*

• *Act is the set of actions directed by the decision controller. The occurrence of an action will trigger a state transition and value update of relevant variables.*

• *G is the set of guard conditions, including Φ_{TS} and Φ_D's data constraints, clock constraints of Φ_C and multi-agent spatial constraints of Φ_M expressed by MASL formulas.*

• *T is the set of transitions of TA. Given a state Q representing current traffic snapshot TS, when the guard condition g is satisfied, the transition relation that changes TS to a post-state TS' with an action $act \in Act$ is denoted as: $TS \xrightarrow{g/act} TS'$.*

• *$q_0 \in Q$ is the initial state that corresponds to the traffic snapshot in the initial scene TS_0.*

Given a state Q representing its corresponding traffic snapshot TS, when guard condition g is satisfied, the transition relation that changes TS to a post-state TS' with an action $act \in Act$ is denoted as: $TS \xrightarrow{g/act} TS'$. The actions in Act vary depending on the platoon driving mode, since driving strategies in different scenarios are inconsistent. In this way, it is necessary to adaptively customize the state transition caused by the action execution in the considered scenario. Nevertheless, the vehicle's decision controller in any platoon driving mode can be modeled concerning the timed automata type defined above. In the following, we take the joining mode as an example to formally define how to update the value of TS when the action occurs.

1) $TS \xrightarrow{go(t)} TS'$: after t seconds, car C will travel a certain distance according to the previous speed and acceleration, thereby the values of spd and pos need to be updated.

$$\begin{aligned}&TS' = (res, pos', inplt, pre, suc, spd', acc, len, join)\\&\quad\wedge pos'(C) = pos(C) + spd(C) \times t + \frac{1}{2} \times acc(C) \times t^2\\&\quad\wedge spd'(C) = spd(C) + acc(C) \times t.\end{aligned}$$

2) $TS \xrightarrow{acc(C,a)} TS'$: car C's current acceleration is assigned to a. $a > 0$ means acceleration, $a < 0$ means deceleration, and $a = 0$ means a constant speed.

$$\begin{aligned}&TS' = (res, pos, inplt, pre, suc, spd, acc', len, join)\\&\quad\wedge acc' = acc \oplus \{C \mapsto a\}.\end{aligned}$$

3) $TS \xrightarrow{clm\_j(C,n)} TS'$: car C declares to change lanes to join the platoon, and n is the lane number where the platoon is located.

$$\begin{aligned}&TS' = (res, pos, inplt, pre, suc, spd, acc, len, join')\\&\quad\wedge join' = join \oplus \{C \mapsto \{n\}\}.\end{aligned}$$

4) $TS \xrightarrow{wd\_j(C)} TS'$: car C withdraws the request to join and continues to drive as usual.

$$\begin{aligned}&TS' = (res, pos, inplt, pre, suc, spd, acc, len, join')\\&\quad\wedge join' = join \oplus \{C \mapsto \emptyset\}.\end{aligned}$$

5) $TS \xrightarrow{in\_j(C)} TS'$: car C is changing lanes to join the platoon. It occupies two lanes simultaneously.

$$\begin{aligned}&TS' = (res', pos, inplt, pre, suc, spd, acc, len, join')\\&\quad\wedge res' = res \oplus \{C \mapsto res(C) \cup join(C)\}\\&\quad\wedge join' = join \oplus \{C \mapsto \emptyset\}.\end{aligned}$$

6) $TS \xrightarrow{fin\_j(C,A,B)} TS'$: car C has completed the lane change and entered the platoon. After joining, car C is in front of car B, followed by car A.

$$\begin{aligned}&TS' = (res', pos, inplt', pre', suc', spd, acc, len, join)\\&\quad\wedge res' = res \oplus \{C \mapsto \{n\}\}\\&\quad\wedge inplt' = inplt \oplus \{C \mapsto \text{true}\}\\&\quad\wedge pre' = pre \oplus \{C \mapsto A\} \wedge suc' = suc \oplus \{C \mapsto B\}\\&\quad\wedge pre' = pre \oplus \{B \mapsto C\} \wedge suc' = suc \oplus \{A \mapsto C\}.\end{aligned}$$

Therefore, to formally model the vehicle's decision controller, we first map the traffic snapshot perceived by vehicles at different moments to TA's state model. Then, the observed multi-agent spatial constraint can be specified as the MASL formula, which is TA's guard condition for decision-making. In this way, the vehicle can issue commands to perform correct actions based on its driving strategy. These behavioral decisions for different platoon scenarios constitute TA's action set. As a general method, the timed automata model is also suitable for modeling decision controllers in other platoon scenarios. When verifying other driving behaviors, we could adjust the action set according to the driving strategy, and then obtain the state update formula according to the characteristics of the action.

5 Model Implementation & Verification

Based on the modeling specifications proposed above, we will take the behavior of HDV joining the platoon as a specific case. By modeling decision controllers of the platoon and HDV that interacts with it, we explain how vehicles perform scenario-based decision-making and collaborative control during driving, thereby ensuring the safety of the platoon control maneuver.

5.1 Case Description

On a three-lane highway with a total length of LEN, there is a platoon P consisting of five intelligent heavy trucks driving on lane n at a constant speed of 60 km/h. It is stipulated that the distance between vehicles in the platoon must be controlled within 20 meters. At a certain moment, an HDV identified as car X is driving on lane m. It is adjacent to platoon P and intends to temporarily cut in the platoon to change to lane n.

First, car X sends a signal to platoon P to declare its expected position for joining the platoon. After receiving the signal, each truck will search for the specific location of car X within its perspective and reports its observation information to the entire platoon system. Through information sharing and knowledge inference, all trucks further confirm car X's specific location relative to the entire platoon. Thus, the platoon leader can make decisions based on global information. It will interact with car X on behalf of the platoon system and send corresponding control signals to command members to drive cooperatively.

Suppose that car X is allowed to directly join the platoon, or after several formation adjustments, an appropriate cut-in position is vacated for it. In this case, the platoon will send a permit signal, allowing car X to perform lane change operations within the specified time period. However, if car X is not allowed to join, the platoon will send it a rejection signal, making it continue to drive as usual.

5.2 UPPAAL Implementation

UPPAAL is an integrated tool for modeling, simulation and verification suitable for real-time systems, which applies to the verification of timing properties. In addition, UPPAAL is based on the theory of timed automata[14], and our modeling method utilizes MASL-based timed automata to construct the vehicle's decision controller. UPPAAL can also use signals to control the parallel execution of models in the network, which conforms to the characteristics of multi-vehicle cooperative driving in the platoon system. As a result, we choose UPPAAL as our experimental tool.

In this subsection, we instantiate the decision controller models of different vehicles on UPPAAL, including the platoon system, HDV, and cycle controller shown in Fig.7. These agents can interact with each other by sending and receiving signals, thereby constructing a network of timed automata model of multi-vehicle cooperative control.

5.2.1 Data Structure

Since the UPPAAL model's simulation relies on the execution of functions and the calculation of related variables, it is necessary to define the model's data structures as its data basis. Moreover, UPPAAL uses C-like code as the programming language but does not directly support MASL we proposed. Therefore, we define the data structure and related functions that represent the semantics of MASL in the form of C code. This self-defined method can reflect the structure and connotation of MASL and is more flexible.

We first define a structure called VelInfo, as shown in Fig.8, which corresponds to the traffic snapshot tuple. In the data structure, *index* represents the vehicle's ID number, and thus VelInfo applies to both platoon cars and non-queue cars. *inplt* is a boolean value indicating whether the vehicle is in the platoon. *pre* is the ID number of the preceding car in the platoon, and *suc* represents the succeeding car's ID number. Array *res* represents the lane number currently occupied by the vehicle since it can occupy at most two lanes simultaneously. *join* represents the lane where the car declares to change for joining the platoon. *pos* represents the position of the vehicle, i.e., the ordinate on the map. *spd*, *acc*, and *len* represent the vehicle's speed, acceleration, and body length, respectively.

To represent the driving scene information observed from the single-vehicle perspective, we define a structure named BiPerspect, as shown in Fig.9. It consists of the following parts: *observer* is the ID number of the observer vehicle, array $occIndicator$ indicates whether the observer has discovered HDV in a certain direction, and array $spatialInfo$ represents the spatial logic information in the discovered area, which is located in the direction indicated by $occIndicator$. The basic spatial logic formula uses $\frown$ operator to concatenate the spatial information on each sub-region, which corresponds to each element in the array $spatialInfo$.

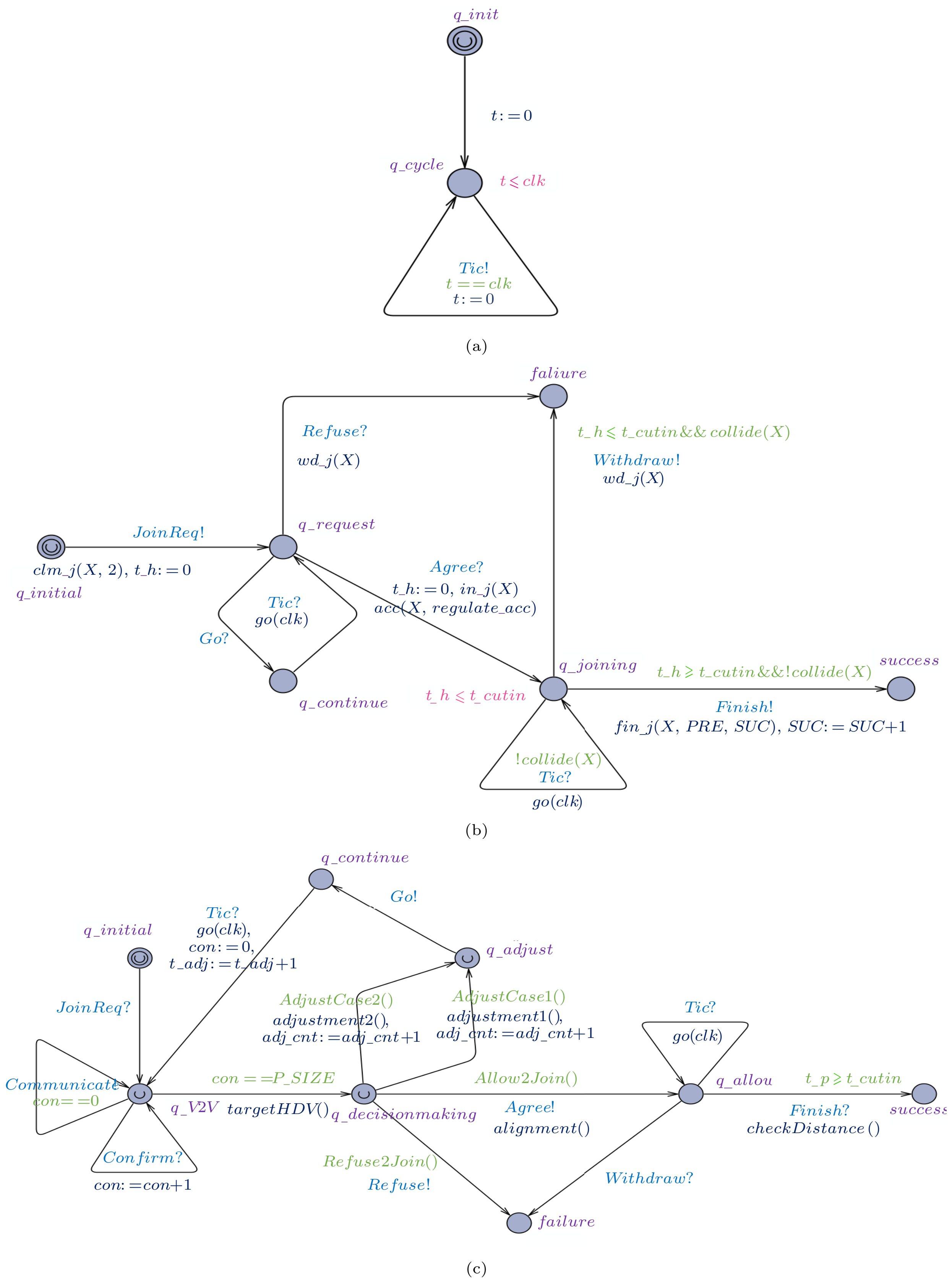


Fig.7. Decision controller models of different agents. (a) Cycle controller. (b) HDV controller. (c) Platoon controller.

For instance, the observer finds that HDV appears in its northeast, and the basic spatial logic formula from its own perspective for this area is expressed as $\phi = \langle gap_1 \frown re(X) \frown gap_2 \rangle$. Thus, the value of BiPerspect satisfies: $occIndicator[N\_E] = 1$, $spatialInfo = (gap_1, re(X), gap_2)$.

```
typedef struct {
        velid_t index;
        bool inplt;
        velid_t pre;
        velid_t suc;
        laneid_t res[2];
        laneid_t join;
        mapY_t pos;
        int spd;
        int acc;
        int len;
} VelInfo;
```

Fig.8. Definition of structure VelInfo.

```
typedef struct {
        velid_t observer;
        int occIndicator[9];
        int spatialInfo[5];
} BiPerspect;
```

Fig.9. Definition of structure BiPerspect.

The inference rules can merge the single-vehicle perspective of all platoon vehicles into the entire platoon's multi-vehicle perspective. Therefore, we define the structure MulPerspect shown in Fig.10. In the data structure, $PPRE$ represents the preceding vehicle relative to HDV, i.e., the $PPRE$-th platoon member is at the front side of HDV. Similarly, the $PSUC$-th member is at the rear side. *indicator* represents the position of HDV relative to the entire platoon, to be precise, the position relative to $PPRE$ vehicle. Array $spatialInfo$ maintains the spatial scene information of the area where HDV is located under the multi-vehicle perspective. For example, from the platoon's perspective, the MASL formula for the area where HDV is located is $\psi = \langle SW_2\phi\rangle \wedge \phi = \langle gap_1 \frown re(X) \frown gap_2\rangle$. Hence, the value of MulPerspect satisfies: $PPRE = 2$, $PSUC = 3$, $indicator = S\_W$, $spatialInfo = (gap_1, re(X), gap_2)$.

```
typedef struct {
        int PPRE;
        int PSUC;
        ORIid_t indicator;
        int spatialInfo[5];
} MulPerspect;
```

Fig.10. Definition of structure MulPerspect.

5.2.2 Cycle Controller

We start to model the cycle controller to control the elapse of time. The automaton sends a broadcast signal *Tic* to all vehicles every cycle, to make all automata run synchronously, and vehicles will take prescribed actions to drive a certain distance.

5.2.3 Platoon Controller

In a platoon system, the leading vehicle is responsible for making decisions and sending correct instructions to members for coordinated driving. After receiving signal *JoinReq* sent by HDV, the platoon will notify all members to detect the relative position of HDV within their respective field of view. The member vehicles report their observation information through V2V communication. By executing function $targetHDV()$, each vehicle's individual knowledge under the single-vehicle perspective is reasoned into the joint knowledge of the entire platoon. As a formal description of the platoon driving scene, the joint knowledge implies the spatial information of HDV relative to the platoon, which is the basis for decision-making.

Case 1. Let us assume that there is no collision after HDV cutting into the platoon according to the initially declared space, and the new platoon can still maintain a reasonable spacing. In that case, the automaton satisfies guard condition $Allow2Join()$. Hence, the leader will send out the signal *Agree*, allowing HDV to join the platoon directly without making any adjustments. The MASL formula of the judgment function $Allow2Join()$ is expressed as:

$$\begin{aligned} & TS, \mathbb{A}, P, \boldsymbol{v} \models E_P\psi \\ & \wedge\ \psi = \langle SW_i\phi\rangle \vee \langle SE_i\phi\rangle \\ & \wedge\ \phi \geqslant \langle gap_2 \frown re(X) \frown gap_2\rangle\,. \end{aligned}$$

We use C code to synonymously implement the MASL formula in UPPAAL based on the data structures, which is shown in Fig.11.

```
bool Allow2Join(){
  ORIid_t indt = MulPS.indicator;
  if (i==MulPS.PPRE&&(indt==S_W||indt==S_E)){
     int s0 = MulPS.spatialInfo[0];
     int s1 = MulPS.spatialInfo[1];
     int s2 = MulPS.spatialInfo[2];
     if (s0>=gap2 && s1==X && s2>=gap2)
       return true;
  }
  return false;
}
```

Fig.11. Implementation of function *Allow2Join*.

Case 2. Suppose that the position declared by HDV is too close to or intersects with the preceding vehicle in the platoon, which satisfies guard condition $AdjustCase1()$. In this case, if HDV directly joins the platoon without any adjustments, it will collide with the vehicle in front. Thus, the platoon executes function $adjustment1()$, controlling the preceding car to accelerate appropriately to provide a suitable space for HDV to cut in. The MASL formula of the judgment function $AdjustCase1()$ is expressed as:

$$\begin{aligned}&TS, \mathbb{A}, P, \boldsymbol{v} \models E_P\psi\\ \wedge\ &\psi = \langle SW_i\phi\rangle \vee \langle SE_i\phi\rangle\\ \wedge\ &\phi \leqslant \langle gap_2 \frown re(X) \frown gap_1\rangle\,.\end{aligned}$$

The code implementation of the MASL formula in UPPAAL is shown in Fig.12.

```
bool AdjustCase1(){
  ORIid_t indt = MulPS.indicator;
  if(i==MulPS.PPRE&&(indt==S_W||indt==S_E)){
     int s0 = MulPS.spatialInfo[0];
     int s1 = MulPS.spatialInfo[1];
     int s2 = MulPS.spatialInfo[2];
     if(s0<=gap2 && s1==X && s2<=gap1)
       return true;
  }
  return false;
}
```

Fig.12. Implementation of function *AdjustCase*1.

Case 3. Similar to $AdjustCase1()$, guard condition $AdjustCase2()$ is satisfied when the position declared by HDV is too close to or intersects with the rear vehicle. In this case, the platoon executes function $adjustment2()$ to make the rear car decelerate appropriately so that the distance can be increased to facilitate HDV cut-in. The judgment function $AdjustCase2()$ is equivalent to the MASL formula:

$$\begin{aligned}&TS, \mathbb{A}, P, \boldsymbol{v} \models E_P\psi\\ \wedge\ &\psi = \langle SW_i\phi\rangle \vee \langle SE_i\phi\rangle\\ \wedge\ &\phi \leqslant \langle gap_1 \frown re(X) \frown gap_2\rangle\,.\end{aligned}$$

The code implementation of function $AdjustCase2()$ is similar to that of $AdjustCase1()$.

Case 4. When one of the following conditions is satisfied, function $Refuse2Join()$ returns true. 1) HDV is not allowed to join the platoon due to its location. 2) None of the above three cases have been met. 3) The platoon has adjusted its formation too frequently. At this time, the leader will send signal *Refuse* to HDV to refuse its joining.

5.2.4 HDV Controller

To change lanes and join the platoon, HDV first sends signal *JoinReq* and declares the expected position in the platoon. When receiving signal *Go* for the platoon's formation adjustment, HDV could not cut in currently and can only drive forward synchronously. If receiving signal *Refuse*, HDV is prohibited from joining the platoon and cancels its declaration. However, if HDV receives signal *Agree*, it will manage to change lanes and adjust its speed to be consistent with the platoon during the period. Only when the action is completed within a limited time *t_cutin* and HDV ensures collision freedom with any platoon member can HDV be considered to have successfully joined the platoon.

When HDV is joining the platoon, it is necessary to ensure that HDV will not collide with any member vehicle. Therefore, we define collision detection function $collide()$ as the guard condition. Once a collision is detected during lane changing, it is determined that HDV failed to join the platoon. We also define an auxiliary function $intersect()$ to detect collisions by checking whether the positions of two cars overlap. The code implementation of the functions is shown in Fig.13.

```
bool collide(){
  for (int i=1;i<=P_SIZE;i++){
    VelInfo p1 = Plt[i];
    VelInfo p2 = HDV;
    if (intersect(p1, p2) == true)
      return true;
  }
  return false;
}

bool intersect(VelInfo p1, VelInfo p2){
  int p1_begin = p1.pos;
  int p1_end = p1.pos + p1.len;
  int p2_begin = p2.pos;
  int p2_end = p2.pos + p2.len;
  if (p1.pos > p2.pos){
    p1_begin = p2.pos;
    p1_end = p2.pos + p2.len;
    p2_begin = p1.pos;
    p2_end = p1.pos + p1.len;
  }
  if (p2_begin <= p1_end)
    return true;
  return false;
}
```

Fig.13. Implementation of functions *collide* and *intersect*.

The model implementation of this case is quite complicated and contains many details. To ensure the model's semantic consistency and normal execution on UPPAAL, we have also defined and utilized many temporary variables, auxiliary functions, signals, and controllers that execute in the background. Due to space limitations, these details are not specifically listed. All model files about this case study are available on the online repository[1].

5.3 Simulation & Verification

Through modeling and function implementation, we have constructed a network of timed automata of multi-agent interaction and control, and have simulated the complete process on the UPPAAL simulator. According to the simulation trace and variables' value change in Fig.14, it is shown that it took five seconds for HDV from declaration to finally join the platoon. During this period, it has undergone formation adjustments twice and the lane-changing operation for three cycles. Finally, HDV cut in the position located between the previous second and third members.

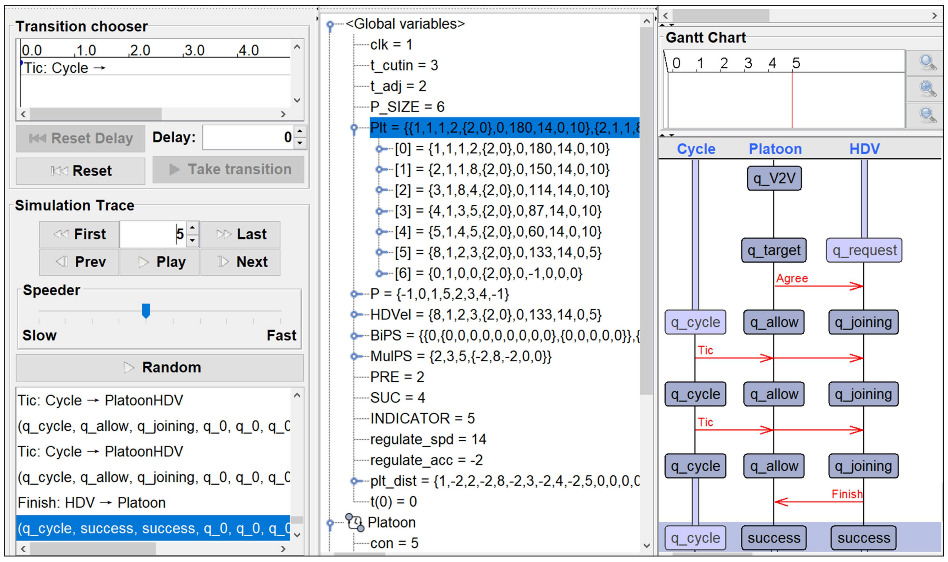
To verify the safety of vehicles' spacing, we also introduce automaton *Observer*, checking whether the distance between any adjacent cars is within the specified safety range $(5, 20]$. When detected to be out of the safe range, the automaton will transfer to the *unbalance_spacing* state. Otherwise, it will always remain in the *maintain_fine_spacing* state to indicate a safe distance.

Fig.15 shows the properties verification results, where these properties correspond to the specifications of the platoon decision controllers. P1 proves that our model is deadlock-free and bug-free. P2 and P3 prove that HDV will not fail to join the platoon due to unsatisfied conditions for joining or collision with the platoon. P4 and P5 prove that HDV can successfully join the platoon under correct decision control. P6 verifies that the platoon can always maintain a safe distance, and P7 shows that the new platoon can still keep a reasonable intra-platoon spacing after HDV cuts in. P8 indicates that the total time for HDV to successfully join the platoon will not exceed the formation adjustment time plus the lane change time. These verification results show that the vehicles' decision controller can ensure normal operation and the safety of driving behaviors at the same time.

6 Related Work

6.1 Developing Reliable Autonomous Systems

In recent years, many formal solutions have been proposed to develop trustworthy autonomous driving systems. Nyberg *et al.*[15] applied a verification technique to embedded safety-critical code, and summarized the application experience of Scania software development. Both Selvaraj *et al.*[16] and Todorov *et al.*[17] introduced static analysis, model checking, and deductive verification techniques into the development of automotive embedded software, bringing higher reliability, robustness, and confidence. Sifakis[18] proposed a computational model consisting of a system architecture model and an agent model, which combined



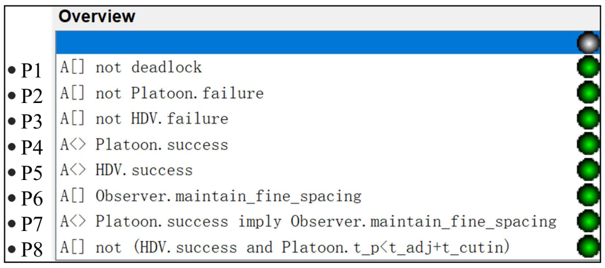
```
Tic: Cycle → 
Tic: Cycle → PlatoonHDV
(q_cycle, q_allow, q_joining, q_0, q_0, q_0
Tic: Cycle → PlatoonHDV
(q_cycle, q_allow, q_joining, q_0, q_0, q_0
Finish: HDV → Platoon
(q_cycle, success, success, q_0, q_0, q_0
```

```
<Global variables>
    clk = 1
    t_cutin = 3
    t_adj = 2
    P_SIZE = 6
    Plt = {{1,1,1,2,{2,0},0,180,14,0,10},{2,1,1,8
        [0] = {1,1,1,2,{2,0},0,180,14,0,10}
        [1] = {2,1,1,8,{2,0},0,150,14,0,10}
        [2] = {3,1,8,4,{2,0},0,114,14,0,10}
        [3] = {4,1,3,5,{2,0},0,87,14,0,10}
        [4] = {5,1,4,5,{2,0},0,60,14,0,10}
        [5] = {8,1,2,3,{2,0},0,133,14,0,5}
        [6] = {0,1,0,0,{2,0},0,-1,0,0,0}
    P = {-1,0,1,5,2,3,4,-1}
    HDVel = {8,1,2,3,{2,0},0,133,14,0,5}
    BiPS = {{0,{0,0,0,0,0,0,0,0,0},{0,0,0,0,0}},{
    MulPS = {2,3,5,{-2,8,-2,0,0}}
    PRE = 2
    SUC = 4
    INDICATOR = 5
    regulate_spd = 14
    regulate_acc = -2
    plt_dist = {1,-2,2,-2,8,-2,3,-2,4,-2,5,0,0,0,0
    t(0) = 0
Platoon
    con = 5
```

Fig.14. Model simulation results.

[1]https://gitee.com/ashley-ecnu-2021/platoon-modeling-verification, Sept. 2021.

data-driven and model-driven methods to design autonomous systems. By applying a formal verification approach to the vehicle's supervisor functions, Yasmine *et al.*[19] validated the design model and ensured that the developed software is compliant with functional requirements. Kinoshita *et al.*[20] constructed a communicating sequential process (CSP)[21] model to analyze the interaction between the driver and the driving system, to guarantee that the system can always capture the driver's status to hand over control in an emergency. Roohi *et al.*[22] explained that there is currently a lack of automatic verification tools to support developing autonomous driving systems. There are still great prospects for exploration in the formal methods and automatic verification tools of autonomous driving technology.


Overview

- P1 `A[] not deadlock` 🟢
- P2 `A[] not Platoon.failure` 🟢
- P3 `A[] not HDV.failure` 🟢
- P4 `A<> Platoon.success` 🟢
- P5 `A<> HDV.success` 🟢
- P6 `A[] Observer.maintain_fine_spacing` 🟢
- P7 `A<> Platoon.success imply Observer.maintain_fine_spacing` 🟢
- P8 `A[] not (HDV.success and Platoon.t_p<t_adj+t_cutin)` 🟢

Fig.15. Properties verification results.

In this paper, we focus on the critical function of the autonomous driving system, the decision-making module. We utilize timed automata modeling and model checking methods to verify how the decision-making function is executed to ensure safe driving from the perspective of system modeling.

6.2 Modeling of Vehicle's Decision Controller

For the safety verification of vehicle decision control, most of the existing work only considers a single vehicle's behaviors. An abstract model is utilized to prove the lane-change controller's safety on highways, where the controller model uses MLSL formulas as guards and state invariants[2,23]. Hilscher and Schwammberger expanded MLSL to Urban Multi-Lane Spatial Logic (UMLSL)[4] by introducing the network topology of urban roads to deal with urban traffic's crossing maneuvers. For driving behaviors on complex road conditions such as T-junction, blocked roads, and roundabout, Xu *et al.*[24,25] defined the basic scene structure for dividing driving tasks. They also considered spatio-temporal constraints of driving scenes in limited time and space, and behavior estimation of surrounding vehicles for quantitative safety verification. Ro *et al.*[6] constructed a car-following model based on Hybrid Input Output Automata (HIOA)[26,27] to compute more accurate vehicle spacing.

The above work has simulated the driving behaviors of a single vehicle in various scenarios. However, with the advancement of IoV (Internet of Vehicles) and 5G technology, vehicles can be interconnected to achieve collaborative driving. It is necessary to extend single-vehicle control to multi-vehicle coordinated control. To this end, we extend the scenario observed from the single-vehicle perspective to multi-vehicles, and model different vehicles' behavioral decisions, thereby simulating coordinated control through signal interaction.

6.3 Safety Verification of Platoon Behaviors

When multiple vehicles manage to complete a unified task through collaboration, it is necessary to adopt methods applicable for multi-vehicles to ensure their safety. Volker *et al.*[28] modeled the roadway and cooperative vehicles' behaviors as automata, proving the safety of the local traffic system. Regarding the platoon as a multi-agent system, an agent-based architecture[9] ensures that the deployed platoon system meets safety requirements. Mallozzi *et al.*[10] and Van and Geihs[11] used the UPPAAL[29] model checker to guarantee the platoon's functions never violate safety properties, including platoon forming, following, leaving, and negotiation behaviors. Focusing on platoon's internal and external safety uncertainties, Hyun *et al.*[30] proposed a statistical verification framework to automatically generate scenario configurations with uncertain factors, bypassing the state explosion problem.

In addition to modeling and verifying platoon behaviors, we also consider the impact of driving scenarios on decision-making, which is the difference from the above work. The formal specification language MASL is proposed to specify spatial scenes as the decision model's guard condition. In this way, the importance of the scenario is reflected, and the perception and the decision-making process of the autonomous driving system are well-connected.

7 Conclusions

In this paper, we proposed a formal framework for the safety assurance of platoon driving on cross-sea highways. The abstract model in the framework can formally describe the spatial characteristics of the driving scene, and MASL can be used to specify the spatial constraints of platoon driving scenes. In addition, the modeling method based on time automata can solve the problem of modeling decision controller, and then combine the MASL specification to define the model's guard conditions. The experimental results proved that our formal solution can effectively model and verify the safety properties of the decision controller of platoon vehicles, thereby obtaining a safety-guaranteed platoon control system.

In the future, we will explore other significant properties of SOI operators and AKR operators in MASL to establish a complete reasoning system. We can comprehensively consider more factors in the platoon driving scenario to expand MASL, such as probability estimation, communication delay, and failure events. Thus, MASL will be able to characterize richer scene features. It is also necessary to optimize our solution and develop tools to verify the decision model's safety in different scenarios automatically.

References

[1] Okuda R, Kajiwara Y, Terashima K. A survey of technical trend of ADAS and autonomous driving. In *Proc. the 2014 International Symposium on VLSI Technology, Systems and Application*, Apr. 2014. DOI: 10.1109/VLSI-TSA.2014.6839646.

[2] Hilscher M, Linker S, Olderog E R, Ravn A P. An abstract model for proving safety of multi-lane traffic manoeuvres. In *Proc. the 13th International Conference on Formal Engineering Methods*, Oct. 2011, pp.404-419. DOI: 10.1007/978-3-642-24559-6_28.

[3] Zita A, Mohajerani S, Fabian M. Application of formal verification to the lane change module of an autonomous vehicle. In *Proc. the 13th IEEE Conference on Automation Science and Engineering*, Aug. 2017, pp.932-937. DOI: 10.1109/COASE.2017.8256223.

[4] Hilscher M, Schwammberger M. An abstract model for proving safety of autonomous urban traffic. In *Proc. the 13th International Colloquium on Theoretical Aspects of Computing*, Oct. 2016, pp.274-292. DOI: 10.1007/978-3-319-46750-4_16.

[5] Xu B, Li Q. A spatial logic for modeling and verification of collision-free control of vehicles. In *Proc. the 21st International Conference on Engineering of Complex Computer Systems*, Nov. 2016, pp.33-42. DOI: 10.1109/ICECCS.2016.014.

[6] Ro J W, Roop P S, Malik A, Ranjitkar P. A formal approach for modeling and simulation of human car-following behavior. *IEEE Transactions on Intelligent Transportation Systems*, 2018, 19(2): 639-648. DOI: 10.1109/TITS.2017.2759273.

[7] An D, Liu J, Zhang M, Chen X, Chen M, Sun H. Uncertainty modeling and runtime verification for autonomous vehicles driving control: A machine learning-based approach. *Journal of Systems and Software*, 2020, 167: Article No. 110617. DOI: 10.1016/j.jss.2020.110617.

[8] El-Zaher M, Gechter F, Gruer P, Hajjar M. A new linear platoon model based on reactive multi-agent systems. In *Proc. the 23rd International Conference on Tools with Artificial Intelligence*, Nov. 2011, pp.898-899. DOI: 10.1109/ICTAI.2011.146.

[9] Kamali M, Dennis L A, Mcaree O, Fisher M, Veres S M. Formal verification of autonomous vehicle platooning. *Science of Computer Programming*, 2017, 148: 88-106. DOI: 10.1016/j.scico.2017.05.006.

[10] Mallozzi P, Sciancalepore M, Pelliccione P. Formal verification of the on-the-fly vehicle platooning protocol. In *Proc. the 13th International Workshop on Software Engineering for Resilient Systems*, Sept. 2016, pp.62-75. DOI: 10.1007/978-3-319-45892-2_5.

[11] Van Nguyen T, Geihs K. Formal verification of multi-agent plans for vehicle platooning. In *Proc. the 9th EAI International Conference on Context-Aware Systems and Applications, and the 6th EAI International Conference on Nature of Computation and Communication*, Nov. 2020, pp.3-15. DOI: 10.1007/978-3-030-67101-3_1.

[12] Rashid A, Siddique U, Hasan O. Formal verification of platoon control strategies. In *Proc. the 16th International Conference on Software Engineering and Formal Methods*, Jun. 2018, pp.223-238. DOI: 10.1007/978-3-319-92970-5_14.

[13] Peng C, Bonsangue M M, Xu Z. Model checking longitudinal control in vehicle platoon systems. *IEEE Access*, 2019, 7: 112015-112025. DOI: 10.1109/ACCESS.2019.2935423.

[14] Alur R, Dill D. Automata for modeling real-time systems. In *Proc. the 17th International Colloquium on Automata, Languages, and Programming*, Jul. 1990, pp.322-335. DOI: 10.1007/BFb0032042.

[15] Nyberg M, Gurov D, Lidström C, Rasmusson A, Westman J. Formal verification in automotive industry: Enablers and obstacles. In *Proc. the 8th Leveraging Applications of Formal Methods, Verification and Validation. Industrial Practice*, Nov. 2018, pp.139-158. DOI: 10.1007/978-3-030-03427-6_14.

[16] Selvaraj Y, Ahrendt W, Fabian M. Verification of decision making software in an autonomous vehicle: An industrial case study. In *Proc. the 24th Formal Methods for Industrial Critical Systems*, Aug. 2019, pp.143-159. DOI: 10.1007/978-3-030-27008-7_9.

[17] Todorov V, Boulanger F, Taha S. Formal verification of automotive embedded software. In *Proc. the 6th International FME Workshop on Formal Methods in Software Engineering*, June 2018, pp.84-87. DOI: 10.1145/3193992.3194003.

[18] Sifakis J. Autonomous systems — An architectural characterization. arXiv: 1811.10277, 2018. https://arxiv.org/abs/1811.10277, Sept. 2021.

[19] Yasmine A, Rabea A B, Patricia G O. Towards formal verification of autonomous driving supervisor functions. In *Proc. the 10th European Congress on Embedded Real Time Software and Systems*, Jan. 2020.

[20] Kinoshita S, Yun S, Kitamura N, Nishimura H. Analysis of a driver and automated driving system interaction using a communicating sequential process. In *Proc. the 2015 International Symposium on Systems Engineering*, Sept. 2015, pp.272-277. DOI: 10.1109/SysEng.2015.7302769.

[21] Brookes S D, Hoare C A R, Roscoe A W. A theory of communicating sequential processes. *Journal of the ACM*, 1984, 31(3): 560-599. DOI: 10.1145/828.833.

[22] Roohi N, Kaur R, Weimer J, James S, Sokolsky O, Lee I. Self-driving vehicle verification towards a benchmark. arXiv: 1806.08810, 2018. https://arxiv.org/abs/1806.08810, Sept. 2021.

[23] Schwammberger M. Introducing liveness into multi-lane spatial logic lane change controllers using UPPAAL. *Electronic Proceedings in Theoretical Computer Science*, 2018, 269: 17-31. DOI: 10.4204/eptcs.269.3.

[24] Xu B, Li Q. A bounded multi-dimensional modal logic for autonomous cars based on local traffic and estimation. In *Proc. the 2017 International Symposium on Theoretical Aspects of Software Engineering*, Sept. 2017. DOI: 10.1109/TASE.2017.8285637.

[25] Xu B, Li Q, Guo T, Du D. A scenario-based approach for formal modelling and verification of safety properties in automated driving. *IEEE Access*, 2019, 7: 140566-14058. DOI: 10.1109/ACCESS.2019.2943184.

[26] Lygeros J, Sastry S. Hybrid systems: Modeling, analysis and control. Technical Report, Electronic Research Laboratory, University of California, 2008. https://www2.eecs.berkeley.edu/Pubs/TechRpts/1999/ERL-99-34.pdf, Sept. 2021.

[27] Alur R, Courcoubetis C, Henzinger T A, Ho P H. Hybrid Automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, Grossman R L, Nerode A, Ravn A P, Rischel H (eds.), Springer, 1992, pp.209-229. DOI: 10.1007/3-540-57318-6_30.

[28] Völker M, Kloock M, Rabanus R, Alrifaee B, Kowalewski S. Verification of cooperative vehicle behavior using temporal logic. *IFAC-PapersOnLine*, 2019, 52(8): 99-104. DOI: 10.1016/j.ifacol.2019.08.055.

[29] Behrmann G, David A, Larsen K G. A tutorial on UPPAAL. In *Proc. the International School on Formal Methods for the Design of Computer, Communication, and Software Systems*, Sept. 2004, pp.200-236. DOI: 10.1007/978-3-540-30080-9_7.

[30] Hyun S, Song J, Shin S, Bae D H. Statistical verification framework for platooning system of systems with uncertainty. In *Proc. the 26th Asia-Pacific Software Engineering Conference*, Dec. 2019, pp.212-219. DOI: 10.1109/APSEC48747.2019.00037.

Jingwen Xu received her B.S. degree in software engineering from Fuzhou University, Fuzhou, in 2019. She is currently a Master student in software engineering from East China Normal University, Shanghai. Her current research interests include model-driven development, formal methods, model checking, and related applications for autonomous driving systems.

Yanhong Huang received her B.S. degree in software engineering and Ph.D. degree in computer science from East China Normal University, Shanghai, in 2009 and 2014, respectively. She is an associate researcher with National Trusted Embedded Software Engineering Technology Research Center, East China Normal University, Shanghai. Her current research interests include formal method, semantics theory, and analysis and verification of embedded systems and industry software. Dr. Huang was the recipient of National Scholarship Award in 2013, IBM China Excellent Students in 2013, and Shanghai Excellent Graduates in 2009 and 2014.

Jianqi Shi received his B.S. degree in software engineering and Ph.D. degree in computer science from East China Normal University, Shanghai, in 2007 and 2012, respectively. He is currently an associate researcher with National Trusted Embedded Software Engineering Technology Research Center, East China Normal University, Shanghai. His current research interests include formal method, formal modeling and verification of real-time or control systems, and IEC 61508 and IEC 61131 standards. Dr. Shi was the recipient of Shanghai Science and Technology Committee Rising-Star Program in 2018, and ACM and CCF nomination of Excellent Doctor in Shanghai in 2014.

Shengchao Qin got his Ph.D. degree in applied mathematics from Peking University, Beijing, and he also worked as a postdoctoral research fellow in National University of Singapore under the Singapore-MIT Alliance program, Singapore, before moving his job to UK. While in UK, he worked as a university lecturer in Durham University, Bedfordshire, and reader in Teesside University, Middlesbrough, in 2011. His research interests lie mainly in formal methods, software engineering and programming languages, in particular, formal specification and modelling, program analysis and verification, theories of programming, and program logic such as separation logic. To this date he has published over 130 papers in international journals and peer-refereed international conferences. He is a senior member of ACM and IEEE. He serves as a full member of EPSRC peer review college and a member of UKRI Future Leaders Fellowship peer review college.