

Statistical Model Checking of Cooperative Autonomous Driving Systems

Cinzia Bernardeschi^[0000-0003-1604-4465], Giuseppe Lettieri^[0000-0003-1005-7441],
and Federico Rossi^[0000-0002-4906-6997]

Department of Information Engineering, University of Pisa, Pisa, Italy
{cinzia.bernardeschi, giuseppe.lettieri}@unipi.it,
federico.rossi@ing.unipi.it

Abstract. In automotive engineering, one of the ideas of vehicle cooperation is to improve convoy movements to increase the safety and efficiency of transportation systems. The complexity and stochastic nature of such cooperative systems pose difficulties for traditional model-checking techniques. Statistical model checking (SMC) provides an answer by using statistical inference to evaluate system features in a probabilistic way. In this work, we show how SMC offers a strong framework for assessing vehicle platooning systems in a range of operational scenarios by fusing statistical analysis and formal verification methodologies. Thanks to this approach, we managed to statistically prove a set of safety and functional properties of an autonomous vehicle platoon system.

Keywords: Cyber-physical systems · Statistical model checking, Vehicle dynamics · Vehicle platooning

1 Introduction

The development of vehicle platooning has emerged as a promising area in automotive engineering [19, 20, 24] in the pursuit of safer and more efficient transportation systems. The concept of platooning, which involves a convoy of vehicles moving in tandem and coordinating their movements, can completely transform the way that roads are used for transportation. Vehicle platooning is a strategy that attempts to improve traffic flow, minimize fuel consumption, and lessen the environmental effect of individual cars by utilizing modern communication and control systems [17, 18]. To ensure system correctness under a variety of operating circumstances, the complex interaction of several vehicles operating nearby necessitates strict verification and validation procedures.

Model checking [7] is an approach used in formal verification to examine system designs for correctness against predetermined attributes. Model checking provides important insights into the robustness of a system by thoroughly examining the state space of the model to see whether it meets desired behavioural specifications. Timed automata (TA)[1] are a formalism for model checking real-time systems. TA are a type of formal model designed to reflect the temporal

dynamics found in cyber-physical systems. By adding explicit timing restrictions and temporal logic specifications, these automata go beyond conventional finite-state machines and allow for the description and validation of complex timing-dependent characteristics. Model-checking techniques [6, 8] can efficiently traverse the state space of time automata to confirm the meeting of temporal features, such as deadlines, synchronisation, and temporal ordering of events, by embedding timing constraints as part of the system model. Because accurate timing coordination is critical to the safe and effective operation of cooperative autonomous driving systems, timed automata are also a vital formalism for the utilization of model-checking techniques in the study and validation of these systems.

In particular, Statistical Model Checking (SMC) [16] uses statistical inference to evaluate system attributes probabilistically. This allows for the assessment of cyber-physical systems across a range of circumstances and uncertainties [2, 13]. This approach extends the traditional timed automata with the rate of clocks given by general expressions that involve other clocks, obtaining the physical behaviour described by ordinary differential equations. Such automata are called hybrid automata [9]. To deal with invariants that contain derivatives that are not constant, e.g. in the vehicle platooning case study, the following vehicle speed derivative depends on the reference acceleration computed through the cooperative control algorithm, we use a statistical approach. SMC provides a methodical way to assess complex systems with stochastic behaviour by combining formal verification methods with statistical analysis. In particular, SMC adds increased scalability and applicability to classic formal verification approaches.

This paper presents an investigation into the application of SMC for vehicle platooning, addressing critical aspects of system design, analysis, and validation. The formalism is flexible and allows the modelling of vehicles able to join and leave the platoon dynamically with little effort. By integrating probabilistic reasoning with formal verification techniques, we aim to provide means for assessing the safety of platooning systems across a spectrum of operational conditions.

The paper is structured as follows: Section 2 illustrates related work and this work main contribution; Section 3 deepens on the technologies and concepts used in this work; Section 4 focuses on the modelling of the dynamic vehicle platoon; Section 5 shows the simulation and probabilistic verification steps; and Section 6 states the conclusions and possible future expansions.

2 Related Work

The use of model checking for autonomous driving systems' verification and validation was investigated thoroughly in the literature.

In particular, in [14] the authors presented a multi-agent system to formally verify the coordination of multiple autonomous vehicles into convoy or platoon formations. The approach explores the use of a so-called physical engine that comprises a Matlab/Simulink physical model of autonomous vehicles. Such a model is used to extract the characteristics of an abstract model implemented

with timed automata that models the algorithmic behaviour of platoon operations. The authors were able to verify the safety and functional properties of the system.

A similar approach was explored in [10] where the authors combined a multi-agent system comprised of a physical runtime and a model checking environment for formal verification for drone swarms.

In [3] the authors applied Statistical Model Checking and Petri nets to a single autonomous vehicle immersed in a traffic jam scenario. By doing so, the authors were able to provide probability bounds to safety properties such as collision probability and average distance before collision.

In [22] authors rigorously modeled the collision detection and avoidance infrastructure of an autonomous vehicle and applied Statistical Model Checking to assess key performance indicators of said collision avoidance system.

In [11, 21] authors presented an approach to model physical quantities and their derivatives as clocks in UPPAAL [4] and applied it to power electrical appliances.

Contributions. The contributions of this work are:

- We embedded physical vehicle and control dynamics all within the same UPPAAL model-based environment for Statistical Model Checking;
- We integrated the platoon application behaviour with the rest of the UPPAAL model;
- We developed a system model that can be easily extended to expand the details of the model (e.g. further complexity in the models of vehicle dynamics, interaction between vehicles and surrounding environment, and inter-vehicle communication).

With respect to previous approaches, we could fully exploit the Statistical Model Checking capabilities of UPPAAL to statistically verify the whole system, from the vehicle dynamics and controls to the platoon behaviour. Moreover, using SMC we could insert stochastic variability inside the platoon behavior without the need to explicitly model randomness in the model. Finally, the use of SMC also helped us to reduce the number of required simulations to reach the desired probability bounds for the assessment of safety and functional properties.

3 Background

This section illustrates basic knowledge of the technologies and concepts used in this work.

3.1 Statistical Model Checking

SMC [16] is a method that aims to study a system’s behaviour by using statistical analysis rather than precise model analysis. Known by another name, Monte

Carlo simulation, it works particularly effectively with stochastic and probabilistic models [15]. The technique entails running a sufficient number of independent simulations of the model’s behaviour to produce a prediction of the behaviour that is statistically valid. The system properties are not completely guaranteed by the SMC technique. However, there is no limit to how much the results’ confidence can be raised. More tests can be carried out if a higher level of assurance is required. The number of simulations rises sublinearly with model size and linearly with certainty. Discrete and nonlinear phenomena are also naturally included in the same model by SMC.

3.2 The UPPAAL Model Checker For Timed Automata

One integrated tool environment for modelling and analyzing system behaviour is UPPAAL [4]. UPPAAL is built on timed automata (TA). It is possible to simulate and evaluate the timed behaviour of different systems using TA. However, when conventional numerical methods are applied, we end up with the state-space explosion problem, where the problem’s size grows exponentially with the number of inputs. This can be addressed within UPPAAL SMC using priced TAs [5] and SMC.

A timed automaton [1] is a directed graph over a set of vertices with a unique vertex known as the initial state. We conventionally refer to vertices as locations. When the automaton is in a certain location, we define that location as active. Clock variables are used to model progressively increasing time. Each location can have an invariant which is a condition on a clock that is deemed to be true as long as the location is active. Exactly one location is active at any time. Locations are connected by edges and for each edge we can have a reset set, an action, and a guard. Guards are conditions on clocks, resets are used to update clocks upon transition from one location to the other and actions can apply updates to the automaton state. An edge is enabled if it starts from an active location, the associated guard is satisfied and the arriving location invariant is satisfied. Keep in mind that following edges happens instantly and doesn’t require any time. Edges and locations can be associated with costs and cost rates, respectively, to create a priced timed automaton.

UPPAAL is particularly suitable for modelling hybrid automata [12], where analogue physical processes and digital computer processes interact.

3.3 Cooperative Adaptive Cruise Control and Platooning

Let us consider a system where a platoon of vehicles is led by a *leader* car that is followed by a certain number of vehicles. The goal is to form a platoon of vehicles that move in unison, maintaining a fixed safety distance without colliding.

The Cooperative Adaptive Cruise Control (CACC) algorithm [19, 24] raises the bar for cruise control by allowing cars to react to the car in front of them in their lane by using a mix of sensors and vehicle-to-vehicle communication. Additionally, the CACC system can react to speed changes made by the car in front of you and by other cars that are farther away and out of your field

of vision faster. It has been demonstrated that the CACC algorithm satisfies the property of string stability [25], a fundamental prerequisite for guaranteeing the safety of the system. For platoons to specifically attenuate either distance error, velocity, or acceleration along the upstream direction, string stability is a desirable quality. Indeed, if the string stability cannot be guaranteed, error signals will be amplified along the upstream direction even if the closed-loop system is internally stable, eventually resulting in the collision of two consecutive vehicles.

Modeling of a vehicle. For simplicity, vehicles can be modelled as material points of mass m , each of them described by a state vector $\langle x, \dot{x}, \ddot{x} \rangle$. Aerodynamic drag interaction can be modelled with a coefficient k that multiplies the speed ($k \cdot \dot{x}_i$). Furthermore, the motion is controlled by the reference acceleration that provides the longitudinal force $F_{lon}^{(i)}$ (we assume a perfect slip ratio between the wheel and the road). Hence we can write the dynamic equation that regulates the vehicles' motion.

$$m \cdot \ddot{x}_i = F_{lon} - k \cdot \dot{x}_i. \quad (1)$$

We can then model the vehicle communication delay of the reference acceleration with a first-order control system. Let a be the vehicle acceleration ($a_i = \ddot{x}_i$), the communication delay behaviour can be expressed as in Equation (2), with τ being the time constant of the system (i.e. from control theory, the time it takes for the step response to rise to 63%).

$$\dot{a}_i = \frac{a_{ref}^{(i)} - a_i}{\tau}. \quad (2)$$

CACC control law. The leader state can be identified by its longitudinal position, speed, and acceleration $\langle x_{leader}, \dot{x}_{leader}, \ddot{x}_{leader} \rangle$ and follows a possible pre-determined path. The same holds for each of the following vehicle i , $\langle x_i, \dot{x}_i, \ddot{x}_i \rangle$. Suppose that vehicles are numbered from 1 to N , with the vehicle 1 being the closest to the leader. For each vehicle i , the desired acceleration $a_{ref}^{(i)}$ is expressed by Equation (3).

$$a_{ref}^{(i)} = c_1 \cdot \ddot{x}_{leader} + (1 - c_1) \cdot \ddot{x}_{i-1} - k_1 \cdot (\dot{x}_i - \dot{x}_{leader}) - k_2 \cdot (x_i - x_{i-1} + d_{safe}), \quad (3)$$

where $\langle x_i, \dot{x}_i, \ddot{x}_i \rangle$ is the state of the preceding vehicle and d_{safe} is the desired fixed distance between vehicles in the platoon. The coefficients c_1 , k_1 , k_2 tune the control response. The value c_1 weights the acceleration value received by the leader and the front vehicle. The higher the value of c_1 , the less sensible to fluctuations of the front vehicle is the control. The values k_1 and k_2 model two gains on the control of speed and position, respectively.

Figure 1 shows the functional scheme of a platoon with a leader and two following vehicles. In the Figure, the CACC function implements the reference acceleration evaluation according to the CACC algorithm in Equation (3).

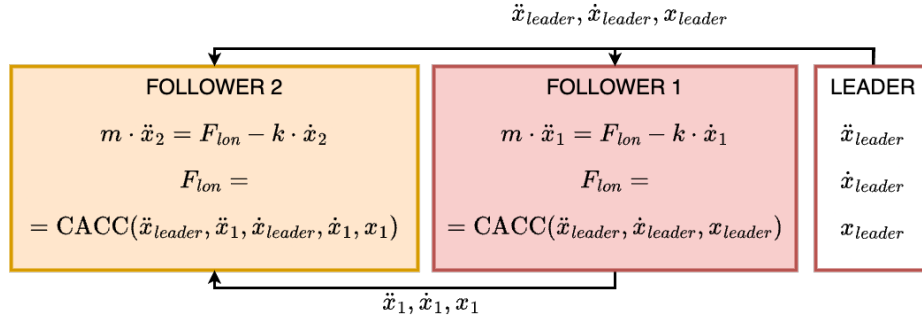


Fig. 1: Schematization of the platoon with 2 followers and 1 leader and vehicle-to-vehicle communications.

4 Platoon Model In UPPAAL

We define four templates that model the components of the system:

- **LeaderDynamic**: models the leading vehicle behaviour.
- **FollowerDynamic**: models the follower vehicle behaviour.
- **PlatoonCommand**: models the reference commands issued to the platoon leader, i.e., the reference acceleration.
- **Simulation**: models the simulation progress for the concrete simulator inside UPPAAL.

The physical quantities, i.e. the system state $\{x_i, \dot{x}_i, \ddot{x}_i\}$, $i = 0 \dots N - 1$, are:

```
clock positions[NUM_VEHICLES];
clock speeds[NUM_VEHICLES];
clock accelerations[NUM_VEHICLES];
```

The leader occupies Index 0, while the follower vehicles occupy the others, corresponding to the platoon order. All these quantities are defined as clocks and their dynamic will be detailed in the following Subsections. The reference acceleration `leader_acceleration` is passed by reference to the **Leader** and **Command** processes. Then, both **Leader** and **Follower1**, **Follower2**, ... vehicles are constructed by passing them by reference the corresponding position, speed, and acceleration variable from the system declarations.

The leader is constructed at lines 26-32 in Listing 1.1.

To ease the construction of followers we used a *Follower(const int rank, const bool leaving, double& leave_time)* generator function to construct a **FollowerDynamic** process from its rank as in Listing 1.1, lines 12-24. The last two arguments are used later to model a leaving vehicle. A follower in position i ($i \neq 0$) is constructed as

Listing 1.1: Composition of the platoon system

```
1 clock t;
2 const int NUM_VEHICLES = N;
3
4 // Reference acceleration from the platoon command
5 double leader_acceleration = 0;
6
7 // Vehicles physical quantities
8 clock positions[NUM_VEHICLES];
9 clock speeds[NUM_VEHICLES];
10 clock accelerations[NUM_VEHICLES];
11
12 // Generator function for the follower's dynamic
13 Follower(const int rank, const bool leaving, double& leave_time) =
14     VehicleDynamic(
15         t, // global clock time
16         accelerations[0], // platoon leader acceleration
17         speeds[0], // platoon leader speed
18         positions[rank-1], // Position of the front vehicle
19         speeds[rank-1], // Speed of the front vehicle,
20         accelerations[rank-1], // Acceleration of the front vehicle
21         positions[rank], // Inout var for the vehicle position
22         speeds[rank], // Inout var for the vehicle speed
23         accelerations[rank], // Inout var for the vehicle acceleration
24     )
25
26 Leader = LeaderDynamic(
27     t,
28     leader_acceleration,
29     positions[0],
30     speeds[0],
31     accelerations[0]
32 );
33 Follower1 = Follower(1);
34 Follower2 = Follower(2);
35 ...
36 FollowerNm1 = Follower(N-1);
37
38
39 Command = PlatoonCommand(t, leader_acceleration);
40
41 Sim = Simulation(s);
42
43 system Sim, Leader, Command, Follower1, Follower2, ..., FollowerNm1;
```

```
Follower1 = Follower(i);
```

Listing 1.1 shows the composition of the system to form the platoon inside UPPAAL.

4.1 Leader Dynamic



```
acceleration' == (a_reference - acceleration)/T &&
speed' == (acceleration*tt-alpha*speed)/mass &&
position' == speed
```

Fig. 2: Hybrid automaton that describes leader dynamics and behaviour

In the model, the dynamics of the leader vehicle follow the exact behaviour expressed in Equation (1). Figure 2 shows the hybrid automaton that describes the leader vehicle dynamics. The automaton has only one state, the initial one, with an invariant that follows a reference acceleration `a_reference`, passed as a reference parameter to the process. The reference is followed by a first-order control system with a time constant `T`, with the behaviour described by the invariant `acceleration' == (a_reference - acceleration)/T`. Then the rest of the dynamic interaction with the road is modelled by the invariant `speed' == acceleration - alpha*speed/mass`, where `mass` is the vehicle mass and `alpha` is the dynamic friction coefficient with the road.

4.2 Platoon Commands

The leader reference acceleration can be modelled with another automaton that simply behaves as an external actor issuing different trajectories to the entire platoon. Figure 3 shows an example automaton that updates the reference acceleration `a_leader`, read by the platoon leader. In this particular example, from the initial location, the first transition is enabled at `t>=10` and updates the acceleration `a_leader = 0.5`. After that, another local clock `local_t` is used to periodically update the acceleration. When `t>=10` the automaton transitions to the next location updating the acceleration to `a_leader = 0` (i.e. makes the platoon move with the same speed for a while) and resetting the clock to `local_t = 0`. In the next location, when `t>=10`, the acceleration is updated to `a_leader = -0.5` (i.e. the platoon decelerates) and finally, in the next location, when `t>=5` the transition to the `CYCLE` state sets back the acceleration to `a_leader = 0.5`.

4.3 Follower Dynamic

We want to allow vehicles to join and leave the platoon. Two models for follower vehicles are defined. We assume that a vehicle that has left the platoon it cannot join it again.

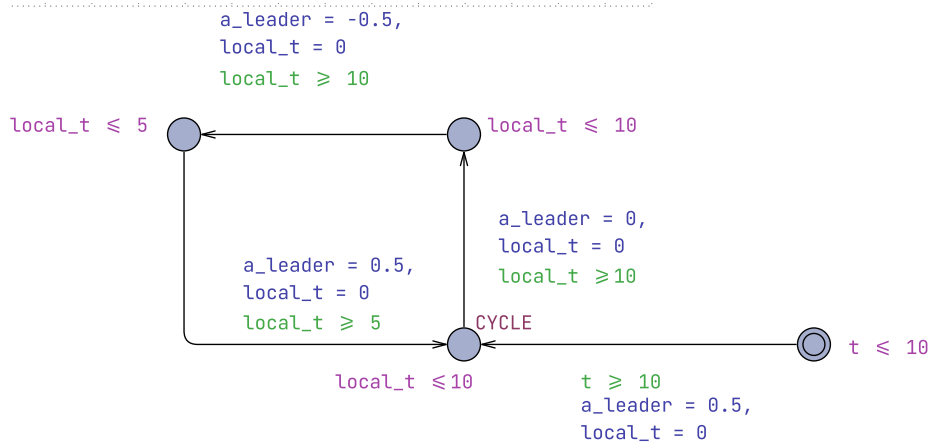


Fig. 3: Hybrid automaton that models commands issued to the platoon leader

Basic Follower Figure 4 shows the basic follower dynamics and controls. The reference acceleration is computed using the CACC control equation (see Equation (3)); the rest of the dynamic is the same as the one described in Equation (1). Therefore the invariant for the derivative of the acceleration is defined as `acceleration' == (eval_accel() - acceleration)/T`. The acceleration is evaluated by the function `eval_accel()` in Listing 1.2.



```

acceleration' = ( eval_accel() - acceleration)/T &&
speed' = (eval_accel()*tt- alpha*speed)/mass &&
position' = speed/T

```

Fig. 4: Hybrid automaton that describes simple follower dynamics and behaviour

Joining and Leaving The Platoon In the case of join and leave, a follower vehicle has three possible locations as shown in Figure 5, depending on the role in the platoon:

- **NOT_JOINED**: the vehicle is moving independently and it is not part of the platoon already ;
- **JOINED**: the vehicle joined the platoon and it is following the front vehicle according to the CACC Equation (3).
- **LEFT**: the vehicle abandoned the platoon and it is no longer part of the convoy (i.e. moved into another lane).

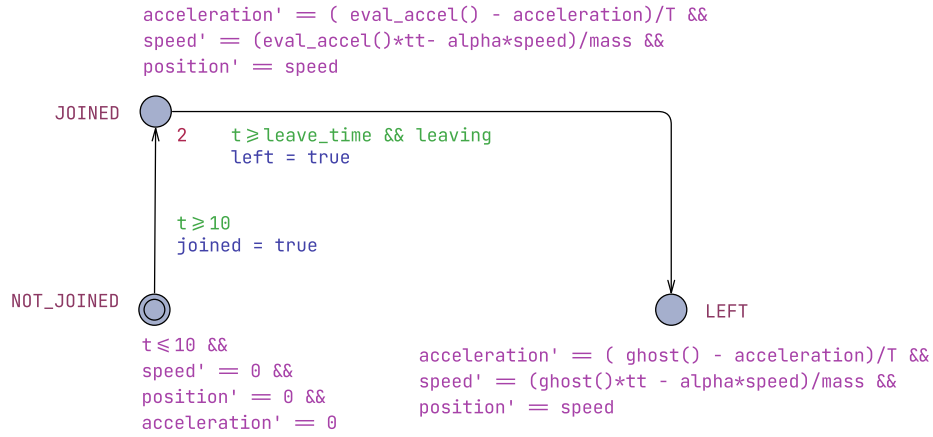


Fig. 5: Hybrid automaton that describes follower dynamics and behaviour with join and leave

Listing 1.2: Reference acceleration computation according to CACC Equation (3).

```

1 double eval_accel() {
2     double accel = c*a_leader+(1-c)*a_front - K1*(speed-v_lead) -
      K2*(position-x_front+dsafe);
3     if(speed <= 0)
4         return fmax(0, accel);
5     else
6         return accel;
7 }
  
```

In all the locations, the follower dynamics is expressed again by Equation (1). When the follower is in the **JOINED** location the reference acceleration is computed using the CACC control equation (see Equation (3)); the rest of the dynamic is the same as the one described in Equation (1). We slightly modified the algorithm not to allow vehicles to go in reverse when they are closer than the safety distance and with a null speed.

We want vehicles to be able to dynamically join and leave the platoon. In particular, we look for vehicles to leave the platoon from any position and to join into the back of the platoon queue. If we look at Figure 5 the **NOT_JOINED** state serves this purpose. Initially, no vehicle is inside the platoon. After a certain given time in the simulation, vehicles can start to join the platoon, transitioning into state **JOINED**. Note that the enabled transitions are not evaluated together but according to the probabilistic approach of SMC, each simulation will have a different order of vehicles joining the platoon.

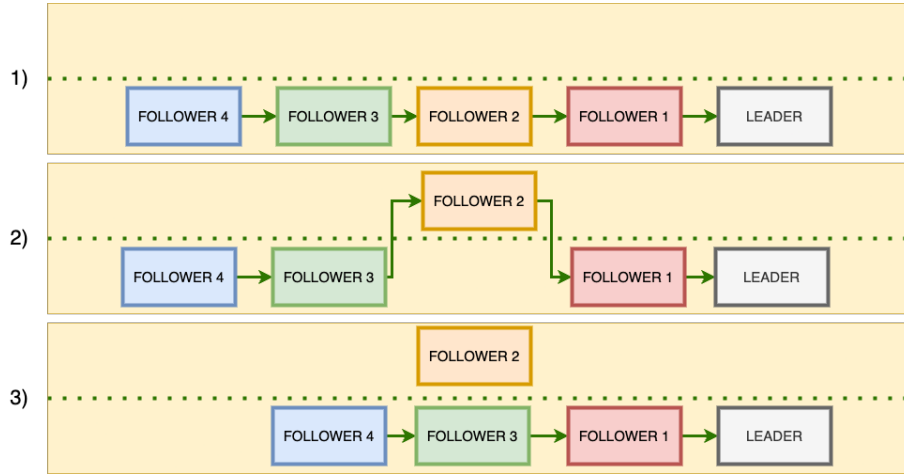


Fig. 6: Three-step evolution of the platoon when a vehicle leaves the formation

We introduce another location named **LEFT** to model a vehicle that left the platoon. For simplicity, we can assume that the leaving vehicle moves away from the longitudinal axis of the platoon, making space for the following vehicle. To model this behaviour, we imagine the leaving vehicle as a *ghost* vehicle that coincides with the front one. By doing so, the following vehicle can close up the distance with its new front vehicle. Figure 6 visualizes the operation of a vehicle (namely, **FOLLOWER 2**) leaving the platoon. We obtain this behavior by changing the evaluation function in the location **LEFT** invariant, with a new *ghost()* function.

```
double ghost() {
    return c*a_leader+(1-c)*a_front - K1*(speed-v_lead) -
        K2*(position-x_front);
}
```

The new function differs just for the last term of the CACC Equation (3); indeed, instead of maintaining a distance **dsafe** from the preceding car, the ghost car will have the preceding car position itself as a reference, hence making it possible for the following car to get close.

4.4 Simulation Control

Since invariants are constantly evaluated, to control the granularity of the enabled transitions we can introduce an automaton that enables a transition with a given frequency. Figure 7 on the left shows an example automaton that has a single initial location, an invariant $t \leq 1$, a guard $t >= 1$, and a reset $t = 0$. This results in a transition exactly every 1 second. If we remove the invariant $t \leq 1$ the transition is enabled every time $t >= 1$ but not exactly at 1 as in Figure 7



Fig. 7: Simulation automaton to control enabling transitions. On the left with a constant exact transition time of 1s and on the right with an exponential distribution of transition times with $\lambda = 1$.

on the right. The actual time will follow a certain probability distribution. We can also specify the rate of the exponential time distribution for the transitions, which has the following density function:

$$\lambda \cdot e^{-\lambda \cdot t}, \quad t \geq 0. \quad (4)$$

5 Simulation and Probabilistic Verification

In this Section, we proceed to first simulate the platoon system under different conditions to visually evaluate and validate the model. We will refer to a platoon configuration with 4 vehicles, where the first one is the leader. Then we apply SMC to provide probabilistic guarantees on the safety properties of the system. The overall system can be configured with several parameters, to explore different initial scenarios and requirements: i) Initial position and speed; ii) Desired safety distance between vehicles; iii) Control gain parameters; iv) Leave time of the vehicles and whether vehicles can leave or not the platoon. We set up initial positions so that the vehicles are spaced by 100 m and initial speeds so that the vehicles start still with null speed.

The control Equation (3) was set up with $c_1 = 0.1$, $k_1 = 1$, $k_2 = 1$, $d_{safe} = 50$ m. Figure 8 shows the simulation outcome with a set desired distance of 50 m with a leaving vehicle after $t \geq 30$. Instead, Figure 9 shows the simulation outcome with a different value of k_1 , and k_2 so that $c_1 = 0.1$, $k_1 = 0.1$, $k_2 = 0.1$ and $d_{safe} = 50$ m. From the figure, it is clear that this configuration leads to an unsafe state due to the poor reactivity of the control system; indeed the parameters k_1 and k_2 control the adjustments in the CACC equation. The higher k_1 the higher the acceleration adjustment due to the speed error. The higher k_2 the higher the adjustment to the position error, as expressed hereafter:

$$\begin{aligned} & k_1 \cdot (\dot{x}_i - \dot{x}_{leader}) \\ & k_2 \cdot (x_i - x_{i-1} + d_{safe}). \end{aligned}$$

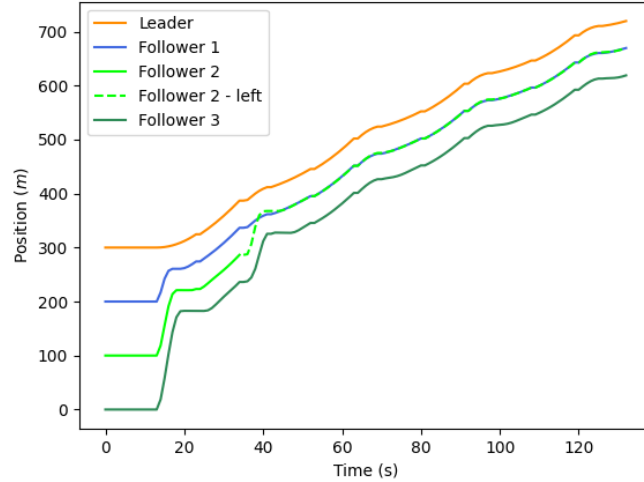


Fig. 8: Position of the 4 platoon vehicles with `Follower2` leaving at time $t \geq 30$, becoming a ghost reference.

5.1 Functional and Safety Properties

In this Section, we introduce a set of safety and functional properties to be proven with SMC. In UPPAAL a property that must always hold can be defined as $\text{Pr}[\leq \text{simulation_time}] (\Box \text{expr})$, where `simulation_time` indicates how much time the system must be simulated for, \Box indicates that `expr` must always hold. Moreover, a property that must eventually be held is defined as $\text{Pr}[\leq \text{simulation_time}] (\langle \Box \text{expr} \rangle)$. The term `expr` is a boolean expression that can contain either a global scope variable (e.g. `positions[0]`) or a process scope variable (e.g. `Follower1.joined`). We used the following parameters for the verification: i) Desired safety distance: 50 m; ii) $k_1 = 1$, $k_2 = 2$, $c_1 = 0.1$; iii) Initial speeds for all vehicles: 0 m/s and iv) Initial positions: vehicles spaced by 100 m.

Table 1 summarises the safety properties we want to prove using statistical model checking. In detail, we want to assess the probability of the vehicle positions not overlapping during the platoon lifetime in the simulation. Furthermore, Table 2 details the properties for the convergence of the platoon positions to the desired distance. The variables `Sim.distances[0]`, `Sim.distances[1]`, `Sim.distances[2]` contains the rolling average distance between pairs of vehicles in the platoon. For example, with a safety distance of 50 m, we want to prove that, eventually, the average distance between the first and the second vehicle in the platoon will be within 5% of the safety distance, i.e. `Sim.distances[0]` $\in [47.5, 52.5]$.

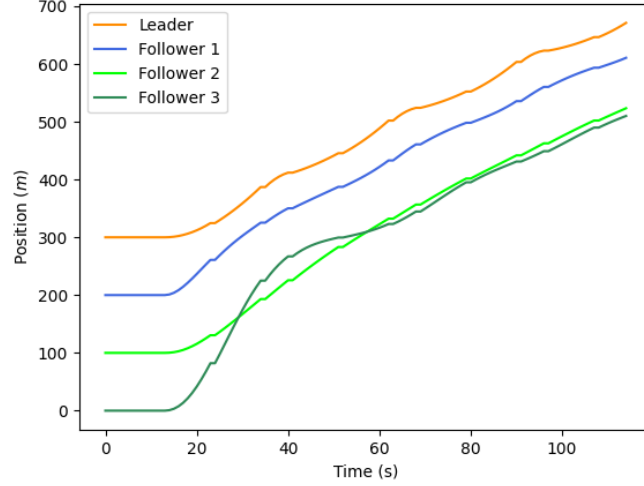


Fig. 9: Position of the 4 platoon vehicles with a crash due to slow reactivity from the control

Table 1: Safety properties on the position of the vehicles in the platoon

Purpose	Query
S1 Follower1 position check	$\text{Pr}[\leq 200]$ $(\lceil \text{positions}[0] > \text{positions}[1]$
S2 Follower2 position check	$\text{Pr}[\leq 200]$ $(\lceil \text{Follower2.left}$ $\parallel \text{positions}[1] > \text{positions}[2])$
S3 Follower3 position check	$\text{Pr}[\leq 200]$ $(\lceil \text{positions}[2] > \text{positions}[3])$

Table 2: Functional properties on maintaining the correct fixed distance between the vehicles in the platoon

Purpose	Query
F1 Follower1 distance	$\text{Pr}[\leq 200]$ $(\langle t \rangle \geq 100 \ \&\&$ $(\text{Sim.distances}[0] > \text{Follower1.dsafte} * 0.95 \ \&\&$ $\text{Sim.distances}[0] < \text{Follower1.dsafte} * 1.05))$
F2 Follower2 distance	$\text{Pr}[\leq 200]$ $(\langle t \rangle \geq 100 \ \&\&$ $(\text{Sim.distances}[1] > \text{Follower2.dsafte} * 0.95 \ \&\&$ $\text{Sim.distances}[1] < \text{Follower2.dsafte} * 1.05))$
F3 Follower3 distance	$\text{Pr}[\leq 200]$ $(\langle t \rangle \geq 100 \ \&\&$ $(\text{Sim.distances}[2] > \text{Follower3.dsafte} * 0.95 \ \&\&$ $\text{Sim.distances}[2] < \text{Follower3.dsafte} * 1.05))$

Table 3 shows the statistical model checking verification results for the properties above. In detail, we report the probability bounds (Probability in the table) of the property being true, the confidence interval (C.I. in the table) of this bound and the number of runs needed by the statistical model checker to obtain such probability bound. As we can see, we managed to prove the safety properties with a probability of at least 99% with a confidence interval of 95%.

Table 3: Statistical model checking proof of the properties

Property	C.I.	Probability	Runs
S1	95%	[0.99, 1)	368
S2	95%	[0.99, 1)	368
S3	95%	[0.99, 1)	368
F1	95%	[0.99, 1)	368
F2	95%	[0.99, 1)	368
F3	95%	[0.99, 1)	368

6 Conclusion

This work has shown how statistical model checking can be used for the analysis of the behaviour of cooperative cyber-physical systems, where physical processes and digital control are strictly interconnected. In particular, SMC is shown to be a strong framework that can evaluate system characteristics in a probabilistic manner in a variety of operational scenarios. With respect to studies based on simulation [23], not only can the incorporation of SMC improve the comprehension of system behaviour, but it also promotes well-informed choices during the design and implementation phases of the system. Further work will consider the application of SMC to analyse safety under complex physical aspects of the vehicle dynamic, critical road surfaces (e.g., wet or icy), and more realistic cooperative autonomous driving systems.

Acknowledgments. This study received funding from the European Union - Next-GenerationEU - National Recovery and Resilience Plan (NRRP) – MISSION 4 COMPONENT 2, INVESTMENT N. 1.1, CALL PRIN 2022 PNRR D.D. 1409 14-09-2022 – (FORESEEN) CUP N.P2022WYAEW and from the Italian Ministry of University and Research (MUR) in the framework of the Crosslab and FoReLab projects (Departments of Excellence).

References

1. Alur, R., Dill, D.L.: A theory of timed automata. *Theoretical computer science* **126**(2), 183–235 (1994)

2. Barbier, M., Renzaglia, A., Quilbeuf, J., Rummelhard, L., Paigwar, A., Laugier, C., Legay, A., Ibañez-Guzmán, J., Simonin, O.: Validation of perception and decision-making systems for autonomous driving via statistical model checking. In: 2019 IEEE Intelligent Vehicles Symposium (IV). pp. 252–259 (Jun 2019). <https://doi.org/10.1109/IVS.2019.8813793>, <https://ieeexplore.ieee.org/abstract/document/8813793>
3. Barbot, B., Bérard, B., Duploux, Y., Haddad, S.: Statistical model-checking for autonomous vehicle safety validation. In: Conference SIA Simulation Numérique. Société des Ingénieurs de l’Automobile, Montigny-le-Bretonneux, France (Mar 2017), <https://hal.science/hal-01491064>
4. Behrmann, G., David, A., Larsen, K.G.: A Tutorial on Uppaal. In: Bernardo, M., Corradini, F. (eds.) Formal Methods for the Design of Real-Time Systems: International School on Formal Methods for the Design of Computer, Communication, and Software Systems, Bertinora, Italy, September 13-18, 2004, Revised Lectures. pp. 200–236. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30080-9_7, https://doi.org/10.1007/978-3-540-30080-9_7
5. Behrmann, G., Larsen, K.G., Rasmussen, J.I.: Priced timed automata: Algorithms and applications. In: Formal Methods for Components and Objects: Third International Symposium, FMCO 2004, Leiden, The Netherlands, November 2–5, 2004, Revised Lectures 3. pp. 162–182. Springer (2005)
6. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.* **8**(2), 244–263 (apr 1986). <https://doi.org/10.1145/5397.5399>, <https://doi.org/10.1145/5397.5399>
7. Clarke, E.M.: Model checking. In: Ramesh, S., Sivakumar, G. (eds.) Foundations of Software Technology and Theoretical Computer Science. pp. 54–56. Springer, Berlin, Heidelberg (1997). <https://doi.org/10.1007/BFb0058022>
8. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching time temporal logic. In: Kozen, D. (ed.) Logics of Programs. pp. 52–71. Springer, Berlin, Heidelberg (1982). <https://doi.org/10.1007/BFb0025774>
9. David, A., Larsen, K.G., Legay, A., Mikučionis, M., Poulsen, D.B.: Uppaal SMC tutorial. *International Journal on Software Tools for Technology Transfer* **17**(4), 397–415 (Aug 2015). <https://doi.org/10.1007/s10009-014-0361-y>
10. Desai, A., Dreossi, T., Seshia, S.A.: Combining model checking and runtime verification for safe robotics. In: 17th International Conference on Runtime Verification (RV). pp. 172–189 (2017)
11. Filipovikj, P., Mahmud, N., Marinescu, R., Seceleanu, C., Ljungkrantz, O., Lönn, H.: Simulink to UPPAAL Statistical Model Checker: Analyzing Automotive Industrial Systems. In: Fitzgerald, J., Heitmeyer, C., Gnesi, S., Philippou, A. (eds.) FM 2016: Formal Methods. pp. 748–756. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-48989-6_46
12. Henzinger, T.A.: The theory of hybrid automata. In: Proceedings 11th Annual IEEE Symposium on Logic in Computer Science. pp. 278–292. IEEE (1996)
13. Huang, Z., Chu, D., Wu, C., He, Y.: Path planning and cooperative control for automated vehicle platoon using hybrid automata. *IEEE Transactions on Intelligent Transportation Systems* **20**(3), 959–974 (2019). <https://doi.org/10.1109/TITS.2018.2841967>
14. Kamali, M., Dennis, L.A., McAree, O., Fisher, M., Veres, S.M.: Formal verification of autonomous vehicle platooning. *Science of Computer Programming* **148**, 88–106 (2017). <https://doi.org/https://doi.org/10.1016/j.scico.2017.05.006>, <https://www.sciencedirect.com/science/article/pii/S0167642317301168>

15. Katoen, J.P.: The probabilistic model checking landscape. In: Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science. pp. 31–45. LICS '16, Association for Computing Machinery, New York, NY, USA (2016). <https://doi.org/10.1145/2933575.2934574>, <https://doi.org/10.1145/2933575.2934574>
16. Legay, A., Delahaye, B., Bensalem, S.: Statistical model checking: An overview. In: Barringer, H., Falcone, Y., Finkbeiner, B., Havelund, K., Lee, I., Pace, G., Roşu, G., Sokolsky, O., Tillmann, N. (eds.) Runtime Verification. pp. 122–135. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16612-9_11
17. Li, S.E., Zheng, Y., Li, K., Wang, J.: An overview of vehicular platoon control under the four-component framework. In: 2015 IEEE Intelligent Vehicles Symposium (IV). pp. 286–291. IEEE (2015)
18. Liang, K.Y., Mårtensson, J., Johansson, K.H.: Heavy-duty vehicle platoon formation for fuel efficiency. *IEEE Transactions on Intelligent Transportation Systems* **17**(4), 1051–1061 (2016). <https://doi.org/10.1109/TITS.2015.2492243>
19. Lu, X.Y., Hedrick, J., Drew, M.: ACC/CACC-control design, stability and robust performance. In: Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301). vol. 6, pp. 4327–4332 vol.6 (2002). <https://doi.org/10.1109/ACC.2002.1025325>
20. Maiti, S., Winter, S., Kulik, L.: A conceptualization of vehicle platoons and platoon operations. *Transportation Research Part C: Emerging Technologies* **80**, 1–19 (2017)
21. Novak, M., Nyman, U.M., Dragicevic, T., Blaabjerg, F.: Statistical model checking for finite-set model predictive control converters: A tutorial on modeling and performance verification. *IEEE Industrial Electronics Magazine* **13**(3), 6–15 (Sep 2019). <https://doi.org/10.1109/MIE.2019.2916232>
22. Paigwar, A., Baranov, E., Renzaglia, A., Laugier, C., Legay, A.: Probabilistic collision risk estimation for autonomous driving: Validation via statistical model checking. In: 2020 IEEE Intelligent Vehicles Symposium (IV). pp. 737–743 (Oct 2020). <https://doi.org/10.1109/IV47402.2020.9304821>, <https://ieeexplore.ieee.org/abstract/document/9304821>
23. Palmieri, M., Quadri, C., Fagiolini, A., Bernardeschi, C.: Co-simulated digital twin on the network edge: A vehicle platoon. *Computer Communications* pp. 35–47 (2024). <https://doi.org/10.1016/j.comcom.2023.09.019>
24. Sheikholeslam, S., Desoer, C.A.: Longitudinal control of a platoon of vehicles. In: 1990 American Control Conference. pp. 291–296 (1990). <https://doi.org/10.23919/ACC.1990.4790743>
25. Swaroop, D., Hedrick, J.: String stability of interconnected systems. In: Proceedings of 1995 American Control Conference - ACC'95. vol. 3, pp. 1806–1810 vol.3 (1995). <https://doi.org/10.1109/ACC.1995.531196>