VerifCar: a framework for modeling and model checking communicating autonomous vehicles

Johan Arcile, Raymond Devillers, Hanna Klaudel

VERIFCAR: A framework for modeling and model checking communicating autonomous vehicles

Johan Arcile[1], Raymond Devillers[2], and Hanna Klaudel[1]

[1] IBISC, Univ Evry, Université Paris-Saclay, 91025 Evry, France,
{johan.arcile,hanna.klaudel}@univ-evry.fr
[2] ULB, Bruxelles, Belgium, rdevil@ulb.ac.be

Abstract This paper presents a framework, called VERIFCAR, devoted to the validation of decision policies of communicating autonomous vehicles (CAVs). The approach focuses on the formal modeling of CAVs by means of timed automata, allowing a formal and exhaustive analysis of the behaviors of vehicles. VERIFCAR supports a parametric modeling of CAV systems as a network of timed automata tailored for verification and limiting the well-known state space explosion. As an illustration, VERIFCAR is applied to check robustness and efficiency, as well as to asses the impact of communication delays on the decision algorithms of CAVs, on well chosen case studies representing real-life critical situations.

Keywords: Timed Automata, Formal Verification, Model Checking, Communicating Autonomous Vehicles

1 Introduction

Autonomous vehicles are rational sophisticated entities (sometimes called *agents*) that, as the term already suggests, act autonomously across open and distributed environments. They may have different perceptions of the environment because of the information they possess, and differing interests in terms of the goal to be accomplished. Inter-vehicle communications affect perceptions and, in turn, individual decisions and behaviors. A system of *communicating* autonomous vehicles (CAV system) is then both a multi–agent system [32] and a real-time system [2]. More precisely, a CAV system is a network of vehicles that communicate with their neighbors to fulfill a goal as fast as possible while complying to the traffic laws and avoiding crashes. Moreover, the correctness of such a system also requires to respect a set of time constraints.

Through some level of abstraction, computer simulations enable one to model the behavior of vehicles in a chosen environment so that various kinds of scenarios may be studied [6,28,33]. However, when vehicles present non-deterministic behaviors, simulation tools are generally not exhaustive since each simulation corresponds to a single path in the graph of all the possible behaviors. This is especially true in the context of communicating agents, where agents interact during non-deterministic time intervals, adding a new kind of non-determinism to the usual one, offering various possible actions at some point. It is therefore appealing to formally verify the core CAV behaviors in order to be confident in the integration of autonomous vehicles into the road traffic.

Formal modeling and verification of CAV systems require not only the definition of both the vehicle states and the road (the environment) but also a specification of interactions between vehicles and an expressive query language to check properties. Ideally, the resulting model of a CAV system should be accurate enough to capture the spatial and time aspects of the original system and should also provide a formal basis for the verification of properties like robustness to faults, effectiveness of maneuvers or the impossibility of collisions (safety), and for the calibration and assessment of decision policies. The language for stating properties should be expressive enough for the needs and appropriate for applying automatic verification techniques and tools.

A widely used automatic technique for system verification is model checking [10]. It provides algorithmic means for determining whether an abstract model – representing a hardware/software/mixed project (in our case a CAV system) – satisfies a formal specification (property) expressed as a temporal logic formula. Moreover, if the property does not hold, the method usually identifies a counterexample run that shows the/a source of the problem.

The main objective of this paper is to present a way to perform formal modeling and model checking of CAV systems, focusing on the impact of various types of communications on vehicles' safety and traffic fluidity. More specifically, we present a framework, called VERIFCAR, composed of a scalable model of a CAV system optimized for formal verification together with a method of calculating indicators allowing to evaluate the quality of a given autonomous vehicle decision policy. The framework is designed in particular to be exhaustive on the non-determinism induced by the latency, communication delays and concurrency features. To show the usefulness of our approach, we present various examples of impacts the communications may have on safety, efficiency or traffic fluidity.

The underlying modeling formalism that we use in VERIFCAR to specify the behavior of CAVs is a model of Timed Automata [2], which is a standard supported by several verification tools, e.g., the model checker UPPAAL [29]. The timed automata formalism is the most well-established model for the specification and verification of distributed real-time system designs. Among many advantages, it allows:

- to create a clear and concise abstract model of the considered CAV systems;
- to assess the robustness of a vehicle decision policy through a fault injection; and
- to apply model checking algorithms and tools, in particular the algorithms designed for timed properties expressed in the temporal logic TCTL [1].

To the best of our knowledge, this kind of formal approach does not seem to have been exploited up to now, except in our recent conference communication [4], of which the present paper is an extended and improved version.

The structure of the paper is as follows: Sections 1 and 2 introduce the motivation and objectives of this paper, and connect them with the related work in the field. Section 3 defines our CAV systems at an abstraction level adapted to our issues. Section 4 introduces our framework VERIFCAR as a parametric network of timed automata synchronizing through broadcast inter-vehicle communications. Two more elaborate communication schemes (negotiations and communications via road infrastructure) are also considered. Section 5 recalls briefly the temporal logic used in our framework and the verification process, including the indicators chosen for the verification purposes and the way they can be computed during model checking. Section 6 justifies our choices of parameters and variables to describe the system, together with a discretization required for verification, allowing the reduction of the resulting state space according to a desired precision. Section 7 shows several examples of how VERIFCAR may be used to study the behavior of autonomous vehicles in presence of inter-vehicle communications, with or without some forms of negotiation or vehicle-infrastructure communications. It includes a discussion on how a practitioner should use VERIFCAR. Section 8 concludes the paper by summarizing the contributions and highlights possible extensions of VERIFCAR allowing to ease its usage and to tackle more complex problems. Finally, an appendix details some algorithm evoked in the text.

2 Related work

The decision policies of CAVs can potentially impact safety, traffic fluidity and energy consumption. They often rely in practice on trajectory planning algorithms studied in 3D simulation, often in conjunction with on-road experiments [19,21,15,23,16]. While the reliability of such systems is a key concern of policy-makers [17], the formal verification of decision policies of CAVs appears to be under-explored.

Among the approaches dealing with formal verification in the context of autonomous vehicles, some use timed models, which seem suitable for studying non-determinism induced by message delays and latency in vehicles' communications. This is the case for example in [14], which aims at verifying the functional layer of mobile robots, *i.e.*, the low-level layer which interacts directly with sensors and actuators and transmits information to the decision layer. For this reason, it does not need to model several agents evolving in a given environment. Another example is [18], which addresses the soundness of vehicle platooning (enabling vehicles to travel as a group on the roads). It focuses on properties of the vehicle platoon, for instance, correct joining and leaving, and considers the representation of vehicles relatively to the platoon. Both these approaches focus on specific properties and their optimized models are not complete enough for our needs.

In [25] hybrid systems are used to model autonomous vehicles. With this formalism, combining both continuous state variables and discrete operating modes, their model achieves a realistic representation of vehicles physics, similar to those which may be found in simulations (slip angle, yaw rate, etc.). However, such a realism leads to low performances during the verification phase, as shown in the presented case study. Indeed, for a system with only two vehicles and a single one applying a decision policy generating limited non-determinism (n paths for n time steps) and the absence of communications between vehicles, the model checker dReach [20] already takes a few minutes for a full exploration of the state space.

Another work, concerning train controllers modeled by hybrid systems [26], proposes a similar representation of agents, taking into account their velocity and acceleration in order to model a realistic physical movement. Here, communications between trains are possible, but the state explosion phenomenon is even more present, since a simple case with two trains takes more than half an hour to give results.

The closest work to ours seems to be [27], involving robots evolving in a two dimensional grid and modeled with a formalism based on timed automata. The mobile robots are represented as agents evolving in a physical environment but at a very high level of abstraction and only basic actions can be performed (such as moving to an adjacent cell). In particular, the model does not include velocity or acceleration values, which are crucial for the realism of vehicles on a road. Furthermore, agents cannot communicate with each other, making it impossible to study the non-determinism induced by communications on a CAV system.

3 General view of a CAV system

The systems of CAVs we consider are composed of several lanes forming a portion of a motorway on which several vehicles can move in the same direction, each one realizing some goal. Each vehicle is defined by a set of constants such as its length or braking capacities. We assume that the vehicles are provided with various perception sensors allowing to observe the behavior of their neighbors. As our approach focuses on the impact of communications on decisions, we assume these sensors are perfect, in the sense that the information obtained from them is always accurate and immediate. Vehicles are also able to communicate and receive pieces of information, which are not directly observable, such as the planned lane change of the other vehicles. These communications are timed in order to realistically represent the delays between emissions and receptions of data. The behavior of each vehicle consists in repeating endlessly, at its own constant frequency:

- a computation allowing to make a decision on the immediate action to be performed (*i.e.*, execution of the decision algorithm) in the form of an acceleration (speed increase, braking, no change) and a direction (left, right, no change), followed by
- the communication of its intention (in the form of the trajectory it wishes to follow) to all vehicles able to receive this information. Received information coming from the other vehicles is stored in the database of the vehicle and used when running the decision algorithm.

The road. In our framework, we consider a road section composed of one or more unidirectional lanes the vehicles move on, and we store at any time the coordinates (position) of each vehicle on it. The current position (x, y) (as well as the initial one) is expressed as the distance from the beginning of the road section (x) and from one of the road borders (y). Due to verification purposes requiring discrete value domains, these distances are expressed using discrete values, but precise enough to satisfactorily model the vehicle progression on the road (see Section 6 for more details on the discretization aspects). As a consequence, the position of a vehicle is a point on a two-dimensional orthogonal grid on which the longitudinal and lateral gaps between adjacent points may be different.

The vehicles. Each vehicle is approximated on this orthogonal grid by a rectangle with a given length and width, centered on its position (x, y) as illustrated in Fig. 1. Its state

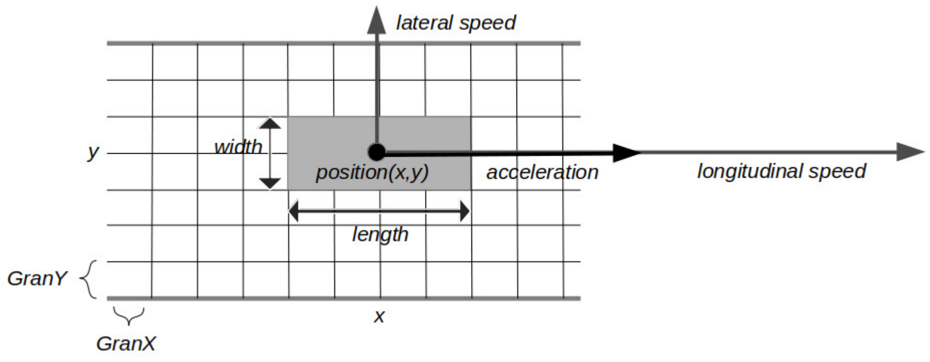

Figure 1. A portion of a road discretised in x-coordinate with a granularity Gran_x and in y-coordinate with Gran_y with a moving vehicle on position (x, y).

is thus described by a record containing its position (the rectangle's center), its current longitudinal and lateral speeds, its longitudinal acceleration (in a given range from negative to positive values) and its knowledge about other vehicles (for example in the form of timed trajectories). Note that the types[3] of these variables are critical and have to be chosen with care as they directly impact both the size of the state space and the modeling precision. This topic will be discussed in Section 6. Since we are not aiming at checking the control of vehicles (*i.e.*, the module responsible for producing a trajectory according to the decision choices), we abstract from the physical laws involved in a maneuver such as rotation or inertia. Hence, the rectangle representing the vehicle never rotates and the direction change is applied directly on the lateral speed value. In other words, turning the steering wheel impacts the speed value that will make the rectangle move on the y axis, while its longitudinal speed still makes it move on the x axis.

The decision. The vehicle decision algorithm follows a given decision policy. The latter may define in particular parameters such as safety distances to be respected in function of the speed. In our experiments, we use a policy common to all vehicles, but this limitation may easily be dropped. In order to make a decision, the algorithm takes into account its own state information including what it knows about its neighborhood (for example, a representation of timed trajectories of other vehicles) as well as its own route (goal). A route of the vehicle is defined as a sequence of positions the vehicle has to reach. More precisely, we consider a sequence of sets of positions to be reached, with the requirement to reach at least one position of each set. These sets may contain several adjacent lateral positions at the same distance from the origin. Concretely, it allows one to define the route as a parameter that will impact the decision choices according to the exit the vehicle wants to take.

Environment update At a constant frequency, the longitudinal speed and position of each vehicle is updated according respectively to the acceleration of the vehicle and

[3] A type corresponds here to a range of integers

longitudinal and lateral speeds. We call that process the environment update, as it updates the state of all agents on the system simultaneously. The environment update frequency should be higher than the frequency of the decision algorithm of each vehicle. This is useful in order to avoid that there is no update between two decisions of the same vehicle, which would mean the first one had no impact on the state of the vehicle. In general, in order to represent the actions of the vehicles in a realistic way, the environment update frequency should be as high as possible. However, the size of the state space grows exponentially with the increase of this frequency.

Timed trajectories. The decision algorithm aims at predicting future conflicts between vehicles, and choosing the most suitable maneuver regarding their needs. We provide for this purpose predicted trajectories but other specific algorithms might be used.

The predicted trajectories of vehicles are timed abstractions represented as sequences of positions the vehicle will reach at some dates up to some time horizon (typically 10 s), as shown in Fig. 2. They are meant to represent the intention of a vehicle at a given moment and can be communicated to other vehicles. In order to optimize memory space, these trajectories are never stored but they are computed on demand from the information of a vehicle and their neighbors, which has to be as compact as possible. To do so, we encode these trajectories with two variables with a limited range of possible values:

– The lane the vehicle is currently aiming at.
– The time delay planned by the decision algorithm before starting a lane change (if any).

The latter delay is expressed using a gap and a maximum duration. For instance, a gap of 100ms and a maximum duration of 5 seconds for the time delay would give 51 possible values ([0.0,0.1,...,4.9,5.0]) for the time delay variable. To indicate the next lane change, this yields 103 possibilities: 51 possibilities for a left change, 51 for a right change and 1 for no lane change.

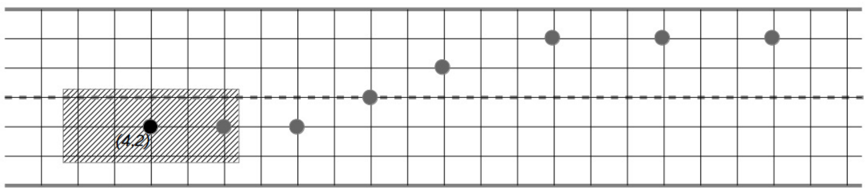

Figure 2. A two lanes road section where a vehicle (dashed rectangle) is centered in $(4,2)$ and its predicted timed trajectory $(6,2);(8,2);(10,3);(12,4);(15,5);(18,5);(21,5)$ for seven next time periods is shown by gray dots (the progression in x varies in function of the speed).

In the real life, the predicted trajectories would of course be much more accurate but, as in our modeling the timed precision of the environment depends on the environment update frequency, it is of no use to represent states which could never be observed, neither by the vehicles nor by the environment. As a consequence, we base our timed trajectories on the environment update frequency.

Cooperation aspects. As we aim to model cooperative vehicles, in addition to the vehicle-to-vehicle information exchange, we propose to study two other forms of communications that are likely to be used in the future for CAVs: negotiation and infrastructure-based decision. These aspects are novel with respect to our previous work [4].

Negotiation is modeled as a distributed algorithm in which agents can interact with each other, each agent trying to impact other agents for its personal benefit. Typically, it can be the case that a vehicle "requests" another one to wait for some time before doing an action in order to minimize the negative impact that this action may have on the safety and/or fluidity of the traffic. The decision algorithms take into account such information using broadcast communications.

Infrastructure-based decision consists in using terminals along the road, which observe the moves and intentions of vehicles, compute and send orders to vehicles, controlling traffic in a centralized way. Here, such terminals are modeled in the same way as the vehicles, *i.e.*, as agents which receive data broadcasted by vehicles and compute the orders to be sent at a given frequency.

Note that we designed a decision algorithm for CAV systems with several variants, with and without taking into account the above cooperation aspects. We needed them in order to illustrate our approach, make various experiments, and show how easily our modeling choices allow to modify the environment of the system. However, it must be understood that the present paper focuses on the modeling of such systems, to show how to assess the robustness of a given decision policy, but does not have any ambition to present and promote a concrete and efficient decision algorithm.

4 VERIFCAR: a timed automata based framework for CAVs

Communication delays between emission and reception of data might be one of the most critical and yet unpredictable parameters in the context of CAVs. Our objective is to make it possible to study the non-determinism induced by such communication delays. The timed automata formalism provides an efficient way to model such systems, leading to several tools that have proved their usability for verification, such as UPPAAL [29].

We present in this section our framework, called VERIFCAR, implementing the modeling ideas introduced in the previous section, using the timed automata formalism. The model will be expressed as a network (a set) of timed automata synchronizing using broadcast communications. The states of each automaton are called locations. One may travel between locations following a timed schedule constrained by the labels of the visited locations and those of the crossed transitions (arcs). Some locations are urgent, meaning that the time may not evolve when the corresponding automaton is there. To express timed information, special variables called clocks evolving with time are introduced. They can be reset and can be used in Boolean formulas defining invariants on locations or guards on transitions.

The functioning of the automata is as usual, *i.e.*, each automaton starts from its initial configuration (initial location and all clock values equal to zero), it may stay in a location when its *invariant* (a predicate labeling the location) is true, and a transition may occur when its *guard* (a predicate labeling the transition) is valid (and so is the invariant of the destination location).

Invariants and guards may use constants (usually in the form of parameters) and variables, either being global or specific to an automaton (like a component of a vector indexed for example by a vehicle i).

All clocks are assumed to progress together. A broadcast channel k has an emitter, denoted $k!$, and a receptor, denoted $k?$, each one associated with at least one transition. When a transition with an emitter is crossed, all available transitions in the network with a reception on the same channel must be crossed simultaneously (meaning that no other actions can happen in between and clocks value cannot change).

4.1 VERIFCAR components and their roles

Let n be the number of agents in our system. Our model thus comprises $n+1$ automata: the environment automaton A_0, and the agents automata A_i for $1 \leq i \leq n$. Agents can be either communicating vehicles or a communicating infrastructure (typically a set of road side units). Besides a set of constant parameters, the data structure of the model is encoded as a set of sub-structures, each of them modeling the state of a single vehicle. All the variables that must be known about that particular vehicle (position, speed, knowledge about the environment, etc.) are members of such a sub-structure. Note that in the timed automata formalism, the variables need to be integer numbers. Along with this data structure, we define $n+1$ clocks C_i, for $0 \leq i \leq n$, and a broadcast channel k. We also consider three functions :

- $update()$, which updates the state of all agents, *i.e.*, their longitudinal and lateral position, speed and acceleration;
- $decision(i)$, which computes the next acceleration and direction to be applied for agent i;
- $communicate(i)$, which sends information about agent i intentions to other agents.

The templates of the VERIFCAR model are then depicted in Fig. 3.

Agent automaton A_i (for $1 \leq i \leq n$) is associated with clock C_i. The role of this automaton is to trigger both the decision of agent i and the communication of this agent to other agents. It is composed of three locations s_0, s_1 and s_2. Location s_0 is the initial one: it is an urgent location (meaning that it must be left as soon as possible) and has an outgoing transition to s_1, which sets C_i to its initial value (defined as a parameter of the agent). Location s_1 is associated with an invariant $C_i \leq freq_i$ where $freq_i$ is a parameter defining the time interval between two decisions for agent i. There is a transition from s_1 to s_2 with a guard $C_i \geq freq_i$, which triggers the function $decision(i)$ and resets C_i. This allows for the decision to occur exactly every $freq_i$ time units. The transition is also associated with the broadcast channel receptor $k?$, whose role will be explained later. Location s_2 is associated with an invariant $C_i \leq MIN\_comm\_delay_i$ and has an outgoing transition to s_1 with a guard $C_i \geq MAX\_comm\_delay_i$, which triggers

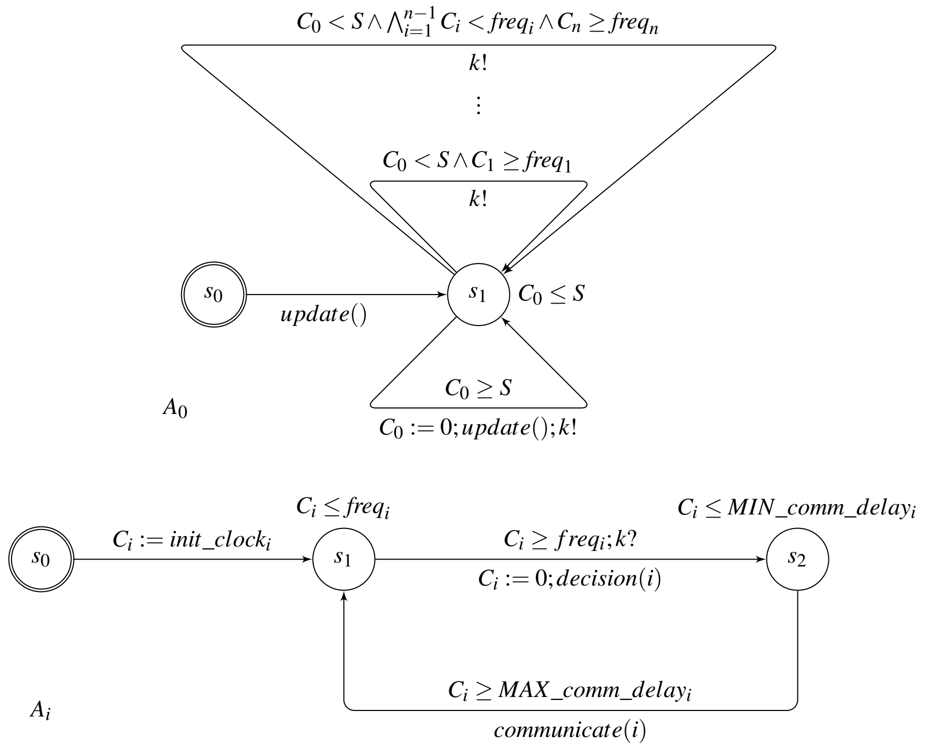

Figure 3. VERIFCAR: timed automata templates A_0 (for the environment) and A_i (for the agent i). The initial locations are urgent and have double borders.

the function $communicate(i)$, where $[MIN\_comm\_delay_i, MAX\_comm\_delay_i]$ is the non-deterministic time interval between data emission and reception by other agents. This model allows to reuse a single clock for two timed operations (decision and communication). However, it requires that $freq_i$ is greater or equal to $MAX\_comm\_delay_i$. This should not be an issue since latency delays in vehicle-to-vehicle protocols are expected to be less than 100ms [31,8,7] while decision modules cited in the literature often use a 10Hz frequency [13,22]. Note that the automaton is the same for vehicles and infrastructures, but the data structure and the actions performed by decision and communication are different.

Automaton A_0 is associated with clock C_0. Its role is to update on a regular time basis (given as parameter S) the state of all agents. It is composed of two locations s_0 (initial) and s_1. Location s_0 is an urgent location and has an outgoing transition to s_1, which triggers the function $update()$. This is used to make the first update that initializes the state of all agents. Location s_1 is associated with an invariant $C_0 \leq S$ and has a looping transition with a guard $C_0 \geq S$, which triggers the function $update()$ and performs a reset of clock C_0. This allows for the regular update of the system's state. This transition is also associated with the broadcast channel emitter $k!$. As the update emulates the continuous movement of objects, it should always have priority on all decision transitions available at the same time[4]. Otherwise, it would create paths in the state space, which only exist because of the loss of information due to the used abstractions. The role of the broadcast channel k is to deal with a form of irrelevant non-determinism: all transitions triggering the decision process are associated with a broadcast channel receptor $k?$ so that, in case of concurrency, the update always has to be triggered first.

Furthermore, thanks to the nature of the broadcast channel we use, all decision processes available in the same time unit will be triggered simultaneously, avoiding useless intermediary states in the system with local non-determinism without any impact on the system. Indeed, the decision made by an agent i is known by other agents only after i communicates, or by seeing the behavior change, which occurs after the update following the decision. Therefore, the order in which competing transitions triggering the decision process are executed is irrelevant.

These transitions must now wait for $k!$ to be triggered, but are not necessarily synchronized with the update sampling defined by S. A solution to this issue would be to add on A_0 a looping transition on s_1 with a guard $C_0 < S \wedge \bigvee_{i=1}^{n} C_i \geq freq_i$ that triggers $k!$. However, the disjunction operator $\bigvee$ is not supported by the tool we use. To emulate this behavior without introducing useless non-determinism, we added on A_0 one looping transition per agent (hence n loops), so that for each $i \in [1,n]$ there is a transition from s_1 to s_1 with a guard $C_0 < S \wedge \bigwedge_{j=1}^{i-1} C_j < freq_j \wedge C_i \geq freq_i$ that triggers $k!$. This particular modeling can be seen as a binary decision diagram. Indeed, if at least one agent can perform its action, there exists $i \in [1,n]$ such that $decision(i)$ is available and $\forall j \in [1,i-1]$ $decision(j)$ is not. Therefore, only one transition in A_0 is available at any moment, which induces a deterministic progression with this automaton.

5 VERIFCAR checking objectives and methodology

In order to use the chosen model checking tool for the VERIFCAR model, the queries must be expressed in the computational tree logic (CTL) [12]. CTL queries are formed of pairs of path quantifiers A or E (for Always and Exists, respectively) and path operators G or F (for Globally and Finally, respectively). For instance, formula $EF\ p$ (respectively $AF\ p$) means that there exists at least one state satisfying property p on at least one path (respectively on all paths) starting from the initial state, where p is an atomic formula or an implementation of a (even complex) Boolean function. For example, to check if there is a possibility that vehicle i reaches eventually lane number 2, one may use a query of the form $EF\ vehicle[i].lane = 2$.

5.1 Indicators for the analysis

As already mentioned, the goal of VERIFCAR is to make it possible to assess and compare decision algorithms for CAVs with respect to properties such as safety, efficiency,

[4] It only occurs for decision transitions, not for communication ones, for which actions are independent of the vehicles' state, and therefore of the update function.

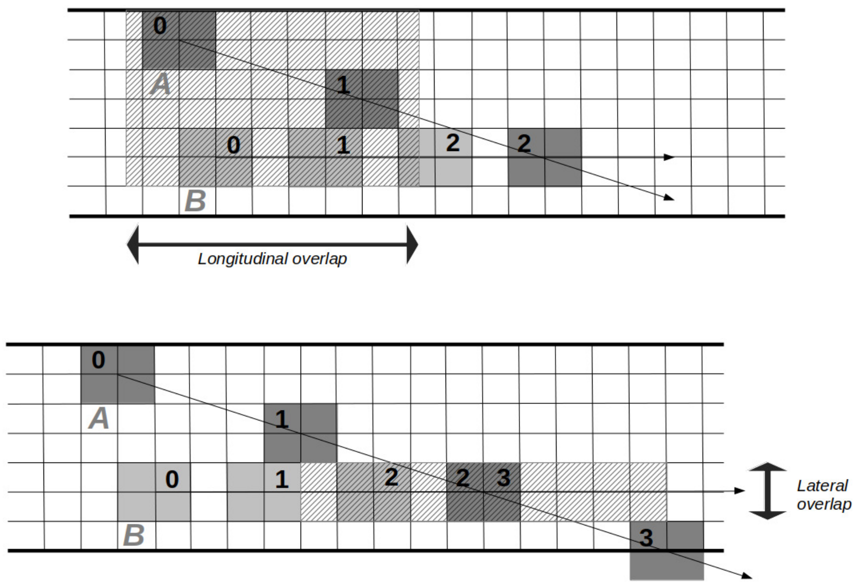

Figure 4. Illustration of overlaps for x-axis (top) and y-axis (bottom). Faster vehicle A is in dark gray while vehicle B is in light gray. The positions of the vehicles are shown at some chosen dates, indicated in top left corner for A and top right corner for B. Dashed areas corresponds to the overlap zones.

comfort or fairness. Here we provide a set of CTL queries to be used in our experiments, presented in the next section, and we shall try to generalize them in order to get an automated methodology.

Each of the mentioned aspects will be checked with various indicators. To check the safety aspect, we propose a Time-to-Collision indicator (TTC), which gives, for a given time t and two vehicles, the time before collision if both vehicles keep moving at the same constant speed. TTC is a commonly used indicator in the literature [30,24] when assessing safety for vehicles on a single lane. Here, we adapt it to a two-dimensional space.

For two moving points on an axis, the instant t_{meet} when they coincide is given by their present distance divided by the difference of their speeds. The algorithm computing the TTC we developed for our framework generalizes this idea for two vehicles of some size moving without rotation on a two dimensional grid: see Appendix A.

When a vehicle is followed at some distance by a faster one, for both longitudinal and lateral directions (*i.e.*, on the x and y axes), we may define a time interval $[t_{meet}, t^{meet}]$ corresponding to the overlap of the two vehicles: t_{meet} is the instant where the closest extremities (on the considered direction) coincide, and t^{meet} is the same for the most distant extremities. The TTC is then obtained by comparing the time intervals obtained for both directions:

– If the two resulting time intervals are non-empty and intersect, the left border of this intersection yields the TTC value. If that TTC value is zero, it means there is a collision presently occurring between the vehicles.
– If there is no intersection between these intervals, there is no possible collision presently expected, and by convention the TTC value is considered infinite.

As an example, consider the situation shown in Figure 4. Initially, vehicle A is in position (3,6) with a longitudinal speed value of 5 and a lateral speed value of -2, while vehicle B is in position (4,2) with a longitudinal speed value of 3 and a lateral speed value of 0. The vehicles lengths and widths have a value of 2 units. The time overlap on the x-axis is $[-0.5, 1.5]$ while the time overlap on the y-axis is $[1, 3]$. The intersection of these intervals ($[1, 1.5]$) is the time window where vehicles overlap on the two dimensional environment, see Figure 5. The left border value gives the TTC (in this case TTC$=1$).

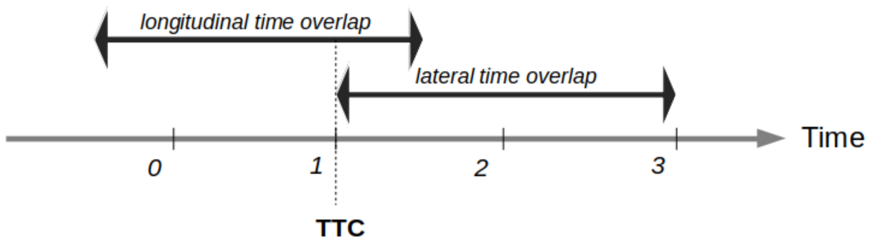

Figure 5. Illustration of TTC computation based on longitudinal and lateral time intervals.

When one considers a whole trajectory, one can compute the smallest TTC, and when considering all the trajectories, one gets a minimal and a maximal value for this smallest TTC. The complexity of that computation is in constant time, which is very interesting with respect to our concern to reduce the computation time.

Efficiency, comfort or fairness can be checked through indicators such as travel time, deceleration or waiting time. One can chose to check either extrema, mean value, or covariance between agents for such indicators. In general, it is possible to use in our framework any of the usual indicators used in the literature, such as the ones depicted in [9] (e.g. waiting time, fuel consumption, loss time, etc.).

5.2 Analysis methodology

Since we study complex scenarios involving non-determinism, different executions will often lead to different states. Therefore, we should not consider a single value for a given indicator but a set of possible values.

For each execution, there is a smallest value for some indicator *indic*. We call $indic_{min}$ the set of those smallest values (each for one execution). Let $k \overset{\text{df}}{=} \inf(indic_{min})$

and $k' \overset{\text{df}}{=} \sup(indic_{min})$, then k is the smallest possible value that satisfies

$$Q^{EF}_{min}(k) \overset{\text{df}}{=} EF\ indic \leq k$$

and k' is the smallest possible value that satisfies

$$Q^{AF}_{min}(k') \overset{\text{df}}{=} AF\ indic \leq k'.$$

It means $EF\ indic \leq k-1$ and $AF\ indic \leq k'-1$ must be false.

Similarly, we call $indic_{max}$ the set of the greatest values of *indic* for the various possible executions. Let $k \overset{\text{df}}{=} \inf(indic_{max})$ and $k' \overset{\text{df}}{=} \sup(indic_{max})$, then n is the smallest possible value that satisfies

$$Q^{AF}_{max}(k) \overset{\text{df}}{=} AF\ indic \geq k$$

and m is the greatest possible value that satisfies

$$Q^{EF}_{max}(k') \overset{\text{df}}{=} EF\ indic \geq k'.$$

It means $AF\ indic \geq k+1$ and $EF\ indic \geq k'+1$ must be false.

To find these extrema, we use the classical dichotomy algorithm in conjunction with model checking queries. Let $Q(n)$ be either $Q^{EF}_{min}(n)$ or $Q^{AF}_{min}(n)$ and $[i, j]$ be the range of possible values for n. Algorithm 1 describes the procedure to determine the smallest possible value that satisfies $Q(n)$. The maximal number of queries needed to find this value is $\log_2(j-i)$.

If $Q(n)$ is either $Q^{EF}_{max}(n)$ or $Q^{AF}_{max}(n)$, we use the same algorithm while adding a negation to the condition: $if(\neg\ Q(\frac{u+v}{2}))$ and return u instead of v.

Algorithm 1 Computation of the minimal value of indicator *indic* for which $Q(n)$ is true. $[i, j]$ is the range of possible values for n, with $i < j$. It is assumed that Q is monotonic, that $Q(j)$ is true and $Q(i)$ is not.

```
$u \leftarrow i$
$v \leftarrow j$
while $u \neq v-1$ do
  if $Q(\frac{u+v}{2})$ then
    $v \leftarrow \frac{u+v}{2}$
  else
    $u \leftarrow \frac{u+v}{2}$
  end if
end while
return $v$
```

In addition to these numerical indicators, arrival orders of vehicles can give additional information on their behaviors. We define a Boolean function denoted $before(x, y)$ taking two vehicles x and y as arguments and evaluating to true on a state if x has reached the end of the road and y has not. To find all the possible arrival orders, we check for every pair of vehicles A and B the following queries: $EF\ before(A,B)$ and $EF\ before(B,A)$. For each pair we have three possible results on the couple of queries:

– Both are true, A arrives first in some executions and B arrives first in others.
– Only one query is true, the order never changes for the pair in all possible executions.
– None is true, both vehicles always reach the end of the road at the same time unit.

Thanks to these simple reachability queries, we manage to have a picture of the possible behaviors that may occur. In some cases of highly non-deterministic behaviors, further investigation might be useful to check arrival orders between more than two vehicles. This case will be illustrated in Section 7.3.

6 Modeling choices, calibration and precision

In order the framework to be usable in practice, it is necessary to maintain a balance between the realism of the representation and the efficiency of the model checking. That implies having the smallest possible state space while losing the least amount of information. Since the available tools that are able to formally verify such systems can only handle integer data structures (basically: parameters, variables and arrays), we have to propose a satisfactory discretization for the various physical quantities describing the vehicles' behaviors. Indeed, the complexity of the verification procedures rapidly increases with the range of these integer variables, while the accuracy of the models requests an adequate granularity.

Our discrete representation of the state of vehicles will thus be encoded with the following discrete variables:

– its (longitudinal and lateral) positions x and y,
– its forward speed v with the corresponding forward acceleration a,
– and the direction of the vehicle $D \in \{-1, 0, 1\}$, corresponding respectively to a lateral move to the right, no change and to the left.

With the exception of D, we will denote by Gran_a, Gran_v, Gran_x, Gran_y the granularity of these quantities, *i.e.*, the gap between two consecutive values (this allows to normalize the data as integers), and by N (with the corresponding subscripts) the size of the needed data structure, called a *range*.

The updates of the forward speed and position values after a period are expressed by the usual formulas. However, as our objective is to be able to efficiently perform model checking, we need our modeling to be parametric, making it possible to preserve an adequate balance between the size of the state space to be analyzed and the desired level of realism. The latter directly depends on the period at which the system is observed, called the *sample*. Thus, the following parameters and constants will be used to describe the system:

– S is the sample period, in seconds (written as s);
– L is the length and R is the width of the road segment, in meters (written as m);
– V_{min} is the min and V_{max} is the max value of (longitudinal) speed expressed in meters per second (written as m/s);
– A_{min} is the min and A_{max} is the max value of acceleration expressed in meters per second squared (written as m/s^2);

- Gran_a is the granularity of the acceleration expressed in meters per second squared;
- W is the lateral speed during a lane change, expressed in meters per second.

As a consequence, the acceleration range is then $\mathsf{N}_a = 1 + (A_{max} - A_{min})/\mathsf{Gran}_a$, assuming that 0 is one of the possible accelerations and that A_{max}/Gran_a as well as A_{min}/Gran_a are integers. The acceleration is then expressed as $a = A \cdot \mathsf{Gran}_a$, the longitudinal speed as $v = V \cdot \mathsf{Gran}_v$, the longitudinal position as $x = X \cdot \mathsf{Gran}_x$, and the lateral position as $y = Y \cdot \mathsf{Gran}_y$, where A, V, X, Y are integers (normalized variables without dimensions). The interest of introducing those dimensionless variables will be to simplify the formulas for state updatings, when granularities are chosen adequately.

Then, the granularities and ranges for the longitudinal and lateral positions and speeds may be computed, as well as their normalized updates after one sample S:

For the longitudinal speed, the update after one sample is

$$v' = v + a \cdot \mathsf{S}.$$

In normalized variables this gives $V' = V + (A \cdot \mathsf{S} \cdot \mathsf{Gran}_a)/\mathsf{Gran}_v$ for a given granularity Gran_v, and $V' = V + A$ if we take the granularity $\mathsf{Gran}_v = \mathsf{Gran}_a \cdot \mathsf{S}$. The main advantage of the latter is that it does not introduce new losses and simplifies the formula. The resulting range is $\mathsf{N}_v = 1 + (V_{max} - V_{min})/(\mathsf{Gran}_a \cdot \mathsf{S})$, where we use a ceiling function if the division does not provide an integer.

For the longitudinal position, the update after one sample is

$$x' = x + v \cdot \mathsf{S} + a \cdot \mathsf{S}^2/2.$$

For a given granularity G_x, this leads in normalized variables to $X' = X + ((V \cdot \mathsf{S} \cdot \mathsf{Gran}_v) + (A \cdot \mathsf{S}^2/2 \cdot \mathsf{Gran}_a))/G_x = X + (2 \cdot V + A)(\mathsf{Gran}_a \cdot \mathsf{S}^2/2)/G_x$. In order to avoid additional losses, we should choose the granularity $G_x = \mathsf{Gran}_a \cdot \mathsf{S}^2/2$, which yields $X' = X + 2 \cdot V + A$. However this granularity will usually be uselessly small (for instance, if $\mathsf{Gran}_a = 1$ and $\mathsf{S} = 10^{-1}$, we get a granularity of 5 mm), leading to a huge range. In order to avoid a state space explosion during the verification process, we shall thus approximate x with a precision of $\mathsf{Gran}_x = p \cdot G_x$, with an adequate parameter p. Hence, the normalized update of x becomes $X' = X + (2V + A)/p$ (rounded) and the corresponding range is $\mathsf{N}_x = \mathsf{L}/\mathsf{Gran}_x$. In order to choose a suitable p, we may observe that Gran_x is the maximal longitudinal loss of precision we may face during a sample. Hence, we may introduce the normalized maximal loss of precision[5] during one second $\mathsf{Norm}_x = \mathsf{Gran}_x/\mathsf{S} = p \cdot \mathsf{Gran}_v/2$. The value of Norm_x may be fixed independently from the constants of the system and we get the corresponding $\mathsf{Gran}_x = \mathsf{Norm}_x \cdot S$ and $p = 2 \cdot \mathsf{Norm}_x/\mathsf{Gran}_v$.

For the lateral position update after one sample we have

$$y' = y + W \cdot \mathsf{S} \cdot D$$

with a corresponding granularity $\mathsf{Gran}_y = W \cdot \mathsf{S}$ since W is a constant (the lateral speed when there is a lateral move) and the direction of the vehicle $D \in \{-1, 0, 1\}$, corresponding respectively to a lateral move to the right, no change and to the left. The range

[5] Actually twice the loss of precision, thanks to rounding.

of the direction is thus $\mathsf{N}_D = 3$, while the range of the lateral position is $\mathsf{N}_y = \mathsf{R}/\mathsf{Gran}_y$. In the normalized variables, this becomes $Y' = Y + D$.

The size of the state space due to the variables is then obtained by multiplying all the above ranges, for each vehicle, hence it behaves like $\mathscr{O}(\alpha^n)$, where n is the number of vehicles and $\alpha = \mathsf{N}_a * \mathsf{N}_v * \mathsf{N}_x * \mathsf{N}_D * \mathsf{N}_y$. Clocks also take part to the state space, so that the size of the full state space is obtained by multiplying the above by an additional exponential, which relies in an intricate way on the number of intersections and differences of the various time intervals (some may be reduced to a single value) occurring in the *guards* and *invariants* of the system, as detailed in [3].

Fortunately, the formal tools do not necessarily construct the whole state space to analyze such systems. Of course, the complexity of the verification procedures also depends on the degree of non-determinism present in the specification and the difficulty to solve the queries.

7 Analysis of CAV systems with VERIFCAR.

In this section, we will illustrate the process of studying the decisions of CAVs with VERIFCAR. We focus on two non-deterministic scenarios, which are well suited to exhibit how communication parameters (and especially temporal ones) are involved in the behavior of the vehicles. First, we point out the impact of communication delays on the non-determinism of the system, *i.e.*, delays between the broadcast and the reception. Then, by injecting faults in the operation of emitters and receptors, we study the robustness of a given decision policy facing such failures. Finally, we compare three decision policies: using only vehicle-to-vehicle communications, using negotiation and using communications via intelligent road infrastructures, for safety and efficiency, *i.e.*, the time that the vehicles need to reach their goal.

The results of our experiments, obtained using the queries introduced in Section 5, are reported in the tables below with the following meaning:

- The arrival order between two vehicles tells which one arrives first. In case of non-determinism, this may give a better insight on the vehicles behaviors than travel times alone, as this shows if a vehicle is able or not to overtake another one.
- The travel times of a vehicle (*i.e.*, the instant when the vehicle leaves the road portion) show the minimal and maximal possible values for all the possible scenarios.
- The worst TTC for two vehicles is the minimal time-to-collision value for some trace. For all the possible traces starting from the same initial state, a minimal and a maximal value of the worst TTC are computed. The minimal one corresponds to the most dangerous scenario and the maximal one to the safest scenario.

The scenarios we use in the experiments involve three vehicles evolving on a three-lane motorway section of $\mathsf{L} = 500$ m long and $\mathsf{R} = 10.5$ m large with two lanes (left and right) and a junction lane which starts at the beginning of the section, joins the right lane after 200 m and ends 200 m later. The constraints defining the velocity of the vehicles are $V_{min} = 0$ m/s and $V_{max} = 40$ m/s (= 144 km/h); $A_{min} = -5$ m/s^2 and $A_{max} = 3$ m/s^2, and $W = 1$ m/s. We fix $\mathsf{Gran}_a = 1$ m/s^2 and the environment sample $\mathsf{S} = 0.1$ s (10 Hz). Such an S allows to monitor the system's behavior in a satisfactory way and such a Gran_a yields a sufficient number of acceleration choices for the decision algorithms of vehicles. One may notice that it is a wise choice as it also leads to good integer divisions in the formulas we expressed in Section 6. With such parameters and the granularity guaranteeing no further loss of information, *i.e.*, $G_x = 0.005$ m, the complexity of the data structure per vehicle is of the order of 10^{11} and it becomes $(10^{11})^3$ for our 3 cars scenario, which might be problematic for the verification tools. We then fix a reasonable normalized accuracy $\mathsf{Norm}_x = 1$ (meaning the loss of precision on the longitudinal position of the vehicle in one second is always less than 1 meter, down to 0.5 meters thanks to rounding) and get $p = 2 \cdot q/\mathsf{Gran}_v = 20$. Therefore, the granularity on x becomes $\mathsf{Gran}_x = p \cdot G_x = 0.1$ m and this approximation allows to divide the state space by $p^3 = 8000$ while not significantly impacting the behavior of the model.

7.1 Decision algorithm

The objective of our decision algorithm is that a vehicle follows its route as fast as possible while complying to the code rules and avoiding collisions with other vehicles. The decision algorithm is local (*i.e.*, each vehicle runs its decision on its own) and takes as an input the global knowledge the vehicle possesses on itself and on other vehicles. It computes timed trajectories of its neighbors, allowing in turn to compute as an output its own new acceleration and direction. The algorithm has to avoid future collisions with vehicles in front of the vehicle, but not behind it as long as there is no immediate danger (the vehicles in front have priority with respect to the ones behind). The expected emergent behavior is that vehicles adapt to the actions of those who precede them.

Algorithm 2 proposes a high level version of the main decision function where $[MinAcc, MaxAcc]$ are all possible acceleration values, $[1, NbLanes]$ are all possible lanes and $[0, MaxDelay]$ are all possible delay values. The decision function tries to find a suitable trajectory with the following objectives (by order of priority):

- going as fast as possible,
- being as close as possible to the lane defined by its route,
- delaying as little as possible its direction changes

Intuitively, a non-zero delay means that the vehicle needs to change lane, but its best option is to do it later: the decision is thus to go straight until the next decision is taken.

The full implementation, together with all source material, can be found at `https://forge.ibisc.univ-evry.fr/jarcile/VerifCar/`. This algorithm has been implemented for the case studies and is given for a better understanding of the present section, but should not be considered as one of our main contributions: our goal is to illustrate the methodology, not to promote a clever decision making algorithm.

7.2 Impact of vehicle-to-vehicle communication on the behavior of CAVs

In this section, we focus on a decision process not involving cooperation (without negotiation nor infrastructure). Thus, each vehicle broadcasts its planned trajectory without trying to impact explicitly the behaviors of the other vehicles. Of course, it also receives the information from its neighbors and uses it in its decision process.

Algorithm 2 High level pseudo-code for the decision function

```
Compute the set of timed trajectories of vehicles in front
Define the lane $L$ to reach w.r.t. the route
for $Acceleration \leftarrow MaxAcc$ to $MinAcc$ do
    for $Lane \in [1, NbLanes]$, starting from $L$ and exploring the neighborhood of $L$ do
        for $Delay \leftarrow 0$ to $MaxDelay$ do
            if Chosen behavior does not generate conflict with vehicles in front then
                if $Delay = 0$ (Waiting not needed) then
                    Define $Direction$ w.r.t. the chosen $Lane$
                else
                    $Direction \leftarrow 0$ (Vehicle goes straight because it is waiting)
                end if
                Return new $Acceleration$ and $Direction$ values
            end if
        end for
    end for
end for
Emergency behavior
```

We use Scenario 1, in which initially vehicle A is on the right lane at position 50 m with a speed of 20 m/s, vehicle B is on right lane at position 0m with a speed of 35 m/s and vehicle C is on the junction lane at position 20 m with a speed of 28.2 m/s. All the vehicles aim to reach the right lane at the end of the road portion, cf. Figure 6.

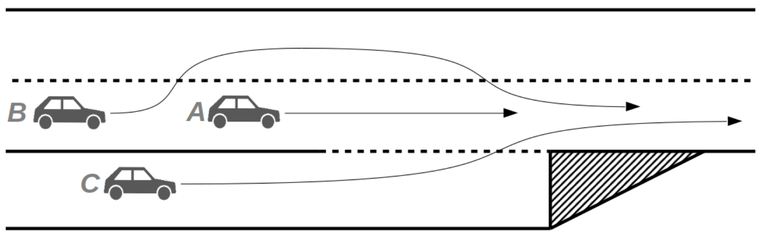

Figure 6. Initial position and possible trajectories of CAVs in Scenario 1.

We assume that vehicle-to-vehicle communications in the default case take between 30ms and 40ms. All the decisions of vehicles have an activation frequency of 10 Hz. In order to avoid unrealistic synchronous behaviors of vehicles, the clocks of agents A, B and C are initialized respectively to 0 ms, 30 ms and 70 ms. This kind of lag between clocks (which is a real life situation where vehicles do not synchronize their clocks) induces non-determinism as a vehicle might take a decision after or before receiving a critical information. Note that since all vehicles have the same decision activation frequency, a scenario where clocks would be initialized with the same value would have a deterministic behavior, as the reception of data would always happen between two given decisions, no matter the moment it happened in the communication delay interval.

Indicator		Variant of communication delay		
		$[30,40]$ms	$[40,40]$ms	$[0,90]$ms
Arrival order	A vs B	B	B	B
	A vs C	C	C	C
	B vs C	$B\|C$	B	$B\|C$
Travel time [s]	A	$(13.1,13.2)$	$(13.1,13.1)$	$(13.0,13.2)$
	B	$(12.7,13.0)$	$(12.7,12.7)$	$(12.7,13.0)$
	C	$(12.6,12.8)$	$(12.8,12.8)$	$(12.6,12.8)$
Worst TTC [s]	A and B	$(1.03,2.90)$	$(2.90,2.90)$	$(1.03,2.90)$
	A and C	(∞,∞)	(∞,∞)	$(2.80,\infty)$
	B and C	$(1.50,\infty)$	(∞,∞)	$(1.50,\infty)$
Full state space exploration [s]		11	4	53

Table 1. Comparison of arrival order, travel time and TTC in Scenario 1 for three variants of communication delay intervals. For arrival order, X vs Y in column 2 indicates that we check who between X and Y is faster, while the winner is indicated in columns 3-5. For travel time and TTC, the pairs in columns 3-5 correspond to the resulting inf and sup values.

Impact of communication delays In Table 1, one can see the differences on Scenario 1 when using various time intervals for the communication delays. The full state space exploration indicates the time corresponding to build and fully explore the state space once, *i.e.*, for a single UPPAAL run. Some indicators such as Travel time or Worst TTC require dichotomy searches, and so several UPPAAL runs, however the number of such runs does not rely on the size of the state space and is generally small. Furthermore, the state space is memorized and not all runs need to fully explore it. For example, the computation of the travel time (which is the most costly of the indicators we considered, in terms of computation time) takes about 45 s to obtain the inf and sup values of the travel time when the full exploration time equals 11 s; similarly, it is about 180 s when the full exploration time equals 53 s, corresponding roughly to the same magnitude (factor 4).

The default case, where the communication delay is in [30,40] ms, illustrates a non-deterministic behavior since several indicators show different possible values, meaning there exists different paths leading to different results. By comparing the travel time values, we can see that travel time for A is always longer than for B or C, meaning it will always be behind the other two at the end of the road. This is confirmed by the arrival order indicator. The inf Travel time value for B is between the inf and sup travel time values of C. However, the inf and sup values for B are greater than respectively the inf and sup values for C. Therefore, it could be the case that B is always behind C. This can be checked thanks to the arrival order between B and C, which indicates that there exist paths where B is before C at the end of the road and some paths where it is the opposite. Finally the worst TTC value gives us indications about safety (an ∞ value means there was never any danger of collision, otherwise a low value means we were close to a collision). It is useful when analyzing the impact of decision parameters, or comparing decision policies.

Here, in addition to the default case, we check two variants of the communication delays: one where we reduce the interval to [40,40] ms (deterministic delays), and one where we extend it to [0,90]ms. As expected, all the inf and sup values obtained when we reduce the interval are bounded by the values for the default case, whereas when we extend the interval, all the values for the default case are bounded by the new inf and sup values. This is consistent with the time needed to fully explore the resulting state space, which increases proportionally to the interval, as indicated by the last row. More precisely, the [40,40] ms interval seems to result on a deterministic scenario with only one path. It is worth to be mentioned that using a single possible delay value greatly reduces the non-determinism, but does not necessarily suppress it. Actually, there might still be states where there is concurrency between actions. For instance, if both a communication and a decision are available at the same time, the order in which the actions are performed may lead to different states.

Indicator		A's receptor disabled		B's receptor disabled	
		$LSD = 0.5$m	$LSD = 1$m	$LSD = 0.5$m	$LSD = 1$m
Arrival order	A vs B	B	B	B	B
	A vs C	C	C	C	C
	B vs C	C	C	$B\|C$	$B\|C$
Travel time [s]	A	$(13.3, 13.3)$	$(13.4, 13.4)$	$(13.3, 13.3)$	$(13.3, 13.3)$
	B	$(13.0, 13.0)$	$(13.0, 13.0)$	$(12.7, 12.9)$	$(12.7, 12.9)$
	C	$(12.6, 12.6)$	$(12.6, 12.6)$	$(12.6, 13.0)$	$(12.6, 13.0)$
Worst TTC [s]	A and B	$(3.00, 3.00)$	$(3.00, 3.00)$	$(\mathbf{0.00}, \mathbf{0.00})$	$(2.80, 2.80)$
	A and C	(∞, ∞)	(∞, ∞)	$(2.86, \infty)$	$(5.04, \infty)$
	B and C	$(1.50, 1.50)$	$(1.50, 1.50)$	$(0.15, \infty)$	$(0.40, 1.35)$
Full state space exploration [s]		5	5	10	11

Table 2. Comparison of arrival order, travel time and TTC on Scenario 1 with disabled communication receptors and variations of lateral safety distance (LSD). The other notations are as in Table 1.

Assessing robustness of decision through fault injection Table 2 and Table 3 show the results of model checking with fault injection, by disabling the receptor or the emitter, respectively, on one of the vehicles. Note that the vehicle sensors are still working, meaning the vehicle gets all other perceptible information (such as position, speed,

Indicator		A's emitter disabled		B's emitter disabled		
		LSD=0.5 m	LSD=1 m	LSD=0.5 m	LSD=1 m	LSD=1.6 m
Arrival order	*A* vs *B*	*B*	*B*	*B*	*B*	*A*
	A vs *C*	*A*	*A*	*C*	*C*	*A*
	B vs *C*	*B*	*B*	$B\|C$	$B\|C$	*B*
Travel time [s]	*A*	$(13.3, 13.3)$	$(13.3, 13.3)$	$(13.4, 13.4)$	$(13.4, 13.4)$	$(13.0, 13.0)$
	B	$(12.7, 12.7)$	$(12.7, 12.7)$	$(12.7, 13.1)$	$(12.7, 13.1)$	$(13.1, 13.1)$
	C	$(13.5, 13.5)$	$(13.5, 13.5)$	$(12.6, 12.8)$	$(12.6, 12.8)$	$(13.4, 13.4)$
Worst TTC [s]	*A* and *B*	$(\mathbf{0.00}, \mathbf{0.00})$	$(2.80, 2.80)$	$(0.30, 0.30)$	$(2.09, 3.00)$	$(1.50, 1.50)$
	A and *C*	$(2.06, 2.06)$	$(2.06, 2.06)$	(∞, ∞)	(∞, ∞)	$(2.22, 2.22)$
	B and *C*	(∞, ∞)	(∞, ∞)	$(\mathbf{0.00}, \infty)$	$(\mathbf{0.00}, \infty)$	$(4.30, 4.30)$
Full state space exploration [s]		6	6	12	12	10

Table 3. Comparison of arrival order, travel time and TTC on Scenario 1 with disabled communication emitters and variations on the length of lateral safety distance (LSD). The notations are as in Table 1.

acceleration, etc...) of other vehicles. The non-perceptible information concerns the intention of the vehicles (their planned trajectories).

When we disable the receptor on vehicle A, the behavior seems deterministic, with C always being first at the end of the road. When we disable the receptor on vehicle B, the behavior is close to the default case; however one can observe that the worst TTC between A and B becomes 0.0 s meaning that there is a collision between these two vehicles. Also, a state where worst TTC between B and C is 0.15 s can be reached, indicating a serious danger. It may be due to the lack of information vehicle B has on other vehicles intention when doing its overtaking. To force B to be more careful, we increase the lateral safety distance (LSD) in the decision process (initially at 0.5m) to 1m. This is intended to increase safety at the expense of efficiency. In the case of the disabled receptor on vehicle A, one can indeed see there is a slight loss of efficiency (vehicle A now takes slightly longer to reach the end of the road). In the case of the disabled receptor on vehicle B, we do not observe a loss of efficiency, but an overall increase of the minimal worst TTC value, including the null ones.

The same checks are performed while disabling the data emitter instead of the receptor. When disabling the emitter on vehicle A, we observe a collision between vehicles A and B, which no longer occurs with the LSD of 1 m. When disabling the emitter on vehicle B, we get a worst TTC of $(0.00, \infty)$, meaning there is both a path leading to a collision between vehicles B and C, and another one without any danger. The increase of LSD to 1 m does not prevent the collision. This motivates us to search the minimal increase of LSD needed to deal with this collision. To do so we proceeded by dichotomy, like for other indicators. As a result we obtained that we need a minimum LSD of 1.6m, which does avoid collision but at the cost of a great loss of efficiency. Indeed, in that scenario, one can observe that none of the vehicles can now overtake vehicle A (the slow one), leading to the worst overall efficiency in our results.

7.3 Impact of cooperation on the behavior of vehicles

In this section, we study the decision policy implementing the forms of cooperation defined in Section 3, namely negotiation and infrastructure. We first describe how they have been implemented.

Implementation of negotiations In this case, each vehicle will try to negotiate whenever its wished trajectory is in conflict with the predicted trajectory of some other vehicle and whenever that vehicle is changing lane. Basically, it happens when a slower vehicle changes lane in front of another one. To keep the state space as small as possible, the request for negotiation (or its absence) is simply indicated by broadcasting a Boolean variable. When the targeted vehicle receives the information, it delays its intention (stopping its maneuver for a while) if this delay does not impact the maneuver it is actually performing and allows the faster vehicle to overtake it. The objective is that none of the vehicles should slow down. And indeed, if we consider Algorithm 2, the vehicle initiates the negotiation after getting new acceleration and direction values, while the target vehicle computes the decision regarding the request, after having found a safe trajectory for itself.

Implementation of infrastructures The road terminal in our infrastructure-based decision performs the following actions at a fixed frequency: First, it computes the predicted and wished trajectories for all vehicles, using the information received from the broadcasted data. Then, it identifies conflicts between vehicles, *i.e.*, situations where the predicted timed trajectory of a vehicle is in conflict with the wished trajectory of another one. Then, it decides which actions should be applied to each vehicle in order to maximize the sum of the accelerations of the vehicles. Finally, the terminal sends to each chosen vehicle the corresponding orders. The impact on Algorithm 2 is that, instead of trying to be as close as possible to the maximal acceleration and to the direction matching the route of the vehicle, it now tries to be as close as possible to the acceleration and direction values sent by the terminal.

Testing scenario Our aim is to compare the above decision policies, namely:

- the base variant of decision without negotiation nor infrastructure (and without fault),
- the variant using negotiation;
- the variant using an intelligent road infrastructure.

We will use for that the Scenario 2 (cf. Figure 7) which is as follows. Initially, vehicle A is on the right lane at position 0 m with a speed of 30 m/s, vehicle B is on the left lane at position 30 m with a speed of 15 m/s and vehicle C is on the junction lane at position 40 m with a speed of 20 m/s. They all aim to be on the right lane at the end of the road portion.

We assume that communications from the terminal to other agents take between 50 ms and 100 ms and communications from vehicles to other agents take between 30 ms

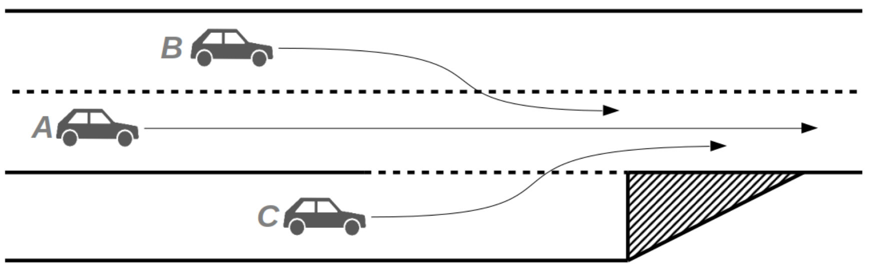

Figure 7. Initial position and possible trajectories of CAVs in Scenario 2.

and 40 ms. The terminal decision has an activation frequency of 2 Hz and its clock is initialized at 0 ms. The decision of each vehicle has an activation frequency of 10 Hz. We also consider two variants, where the agent clocks are given different initial values. In initialization variant 1, clocks of agents A, B and C are initialized respectively after 70 ms, 30 ms and 0 ms, while in initialization variant 2, clocks of agents A, B and C are initialized respectively at 0 ms, 30 ms and 70 ms.

Scenario		Init. variant 1			Init. variant 2		
		Base	Negotiation	Infrastructure	Default case	Negotiation	Infrastructure
Arrival order	*A* vs *B*	*B*	*A*|*B*	*A*	*B*	*A*	*A*
	A vs *C*	*C*	*A*|*C*	*A*	*C*	*A*|*C*	*A*
	B vs *C*	*C*	*C*	*C*	*C*	*C*	*C*
Travel time	*A*	$(14.6, 14.6)$	$(13.0, 15.9)$	$(13.2, 13.3)$	$(14.6, 14.6)$	$(13.0, 13.6)$	$(13.2, 13.4)$
	B	$(14.4, 14.4)$	$(14.4, 14.4)$	$(14.4, 14.4)$	$(14.4, 14.4)$	$(14.4, 14.4)$	$(14.4, 14.4)$
	C	$(13.2, 13.2)$	$(13.2, 13.2)$	$(13.8, 13.8)$	$(13.2, 13.2)$	$(13.2, 13.2)$	$(13.8, 13.8)$
Worst TTC	*A* and *B*	$(0.90, 0.90)$	$(\mathbf{0.00}, 1.56)$	$(1.40, 1.44)$	$(0.95, 0.95)$	$(0.60, \infty)$	$(1.36, 1.41)$
	A and C	(∞, ∞)	$(1.14, \infty)$	(∞, ∞)	(∞, ∞)	$(1.14, \infty)$	(∞, ∞)
	B and C	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
Full state space exploration [s]		5	379	27	3	299	36

Table 4. Comparison of arrival order, travel time and TTC for default case decision, decision with negotiation and decision with infrastructure on the two initialization variants of Scenario 2.

Table 4 shows the results obtained for Scenario 2. First, let us consider initialization variant 1. Our default case decision policy seems to lead to a deterministic behavior, where C manages to reach the right lane ahead of vehicle B while A has to brake to avoid a collision with vehicle B. This can be confirmed by looking at the worst TTC between A and B, which is less than 1 second. When adding the negotiation aspect, one can observe a change in the behavior, which is now non-deterministic. While the travel times for B and C have not changed and are all the same (a unique value) for all possibles executions, this is no longer the case for A, for which the minimal travel time is now shorter than for other vehicles while its maximal travel time has increased. The arrival orders give us the information that vehicle A can sometimes arrive ahead of B (respectively C), sometimes not. This information is not sufficient to get a perfect picture of the possible scenarios. As C always arrives before B, one can be sure that arrival orders, where A arrives as the first or the last are possible. However, to know if A can arrive between B and C one should check the following property: $\mathit{before}(C,A) \rightarrow \mathit{before}(B,A)$, which means[6] that when A is not first, it must be last, and therefore a case where it arrives between the two other vehicles is not possible. Here, the above property is false, and the model checker is able to give us traces where vehicle A arrives between C and B.

Concerning safety, one can see that the worst TTC between A and B can be zero, meaning that there are possible scenarios. with a collision between these two vehicles. The potential gain in overall travel time is thus paid here by a lack of safety.

Finally, when using an infrastructure-based decision policy, one can observe that vehicle A always arrives first and vehicle C manages to reach the right lane ahead of vehicle B, as previously. The comparison between the travel times shows us that with the infrastructure, the efficiency is not as good as with negotiation in the best case (vehicle A and C being slower), however the non-determinism is not significant (unlike with negotiation where there exists a possibility of collision) and the worst TTC is even better than for the default case decision policy.

Looking at the results obtained with initialization variant 2, one can see there are almost no differences for default case and infrastructure based decision policies. However, in this scenario, the negotiation does not lead to collision anymore. Vehicle B now always arrives as the last, vehicle A and C being either first or second. The worst TTC is smaller than for the default case but we gain in overall efficiency.

Considering the state space exploration time, one can see that the non-determinism induced by negotiation in this particular example can lead to a huge time increase (about a hundred times longer) when compared to our default case. Note that this is not due to the negotiation itself but to the non-determinism. Finally, infrastructure-based decision, despite showing a deterministic behavior, takes longer than the default case for the state space exploration (about 10 times longer). This is due to the fact that the road infrastructure is implemented as yet another agent, which impacts the size of the resulting state space. As one can see, the time needed to perform the verification process is however maintained quite short thanks to our modeling choices.

7.4 Limitations and parameter management

Through several additional experiments, we also evaluated the impact of our main parameters on the computation time of UPPAAL. To study this impact, we measured the full exploration time while varying several scenarios with non-deterministic behaviors. Overall results are given in Table 5. The environmental parameters, such as the length of the road portion, the size of the variables used for timed trajectories or to represent vehicles behaviors, show a linear impact on the computation time, even with complex scenarios. This is also true for the normalized maximal loss of precision Norm_x. For instance, for a scenario with a full exploration time of more than 20 minutes when no further approximation is performed[7] the use of a value Norm_x=1 m/s allows to reduce

[6] The operator $\rightarrow$, called "leads to", supported by the UPPAAL model checker is equivalent to $AG\ (\neg\ \mathit{before}(C,A) \vee AF\ \mathit{before}(B,A))$.

[7] *i.e.*, when $\mathsf{Norm}_x = \mathsf{Gran}_v/2$: in this example $\mathsf{Norm}_x = 0.05$ m/s.

this time to about 6 minutes. These results indicate that we can increase the accuracy of the modeled environment with limited cost.

In contrast, we are limited in the degree of realism, especially for scenarios involving a lot of non-determinism, where variations on the sampling period S and decision frequency show an exponential impact on the computation time. In particular, since the frequency induced by the sampling period is required to be smaller than or equal to any other frequency in the system, the use of a given decision frequency implies to scale accordingly the sampling period. Therefore, increasing the decision frequency of vehicles in a system can lead to a state space explosion. However, it is still possible to manage to go up to 20 Hz with a reasonable computation time (a few hours) for complex scenarios involving a lot of non determinism, which as discussed in Section 4 is enough for decision modules mentioned in the literature. For simpler scenarios, however, a higher decision frequency can be used.

	Norm_x[m/s]			Gran_a[m/s^2]			L[m]		Delay range		S[s]			$freq_i$ $i \in [1,n]$[Hz]	
Value	0.5	0.1	0.05	0.5	0.2	0.1	750	1000	100	500	0.05	0.02	0.01	20	33.3
Time [s]	11	11	11	16	17	52	15	18	14	31	16	58	90	45	166

	Norm_x[m/s]			Gran_a[m/s^2]		S[s]		$freq_i$ $i \in [1,n]$[Hz]
Value	0.5	0.1	0.05	0.5	0.2	0.05	0.03	20
Time [m]	11.1	15.6	20.6	9.1	18.2	19.0	113.6	500.3

Table 5. Top: Full exploration time (in seconds) for Scenario 1 with altered parameters. For comparison, the full exploration of the default case takes 11 seconds. Bottom: Full exploration time (in minutes) for initialization variant 1 of Scenario 2 using negotiation, with altered parameters. For comparison, the full exploration of the default case takes 6.3 minutes.

Finally, it must be observed that the number of vehicles does not necessarily impact the computation time: it is the resulting non-determinism of the evolutions that does, much more than the increased number of variables. This is illustrated in Table 6, where vehicles have been added to the previously studied cases, with various random position and speed values. A look at the smallest exploration times indicates that the increasing size of the data structure seems to have a linear impact on the system. On the other hand, one can see that the generated state space can be drastically different between systems with the same number of vehicles but with different initial parameters.

Note that the maximum reasonable number of vehicles is therefore mainly case-related. Indeed, adding vehicles can sometimes reduce the non-determinism of the system and sometimes increase it. This result in particular indicates that our framework could hardly support erratic human-like drivers (*i.e.* decision algorithm with random choices) but is well suited for checking interaction behaviors between autonomous vehicles characterized by well-defined decision protocols.

To study the behavior resulting from the use of a given decision algorithm, one should implement it within the UPPAAL language, which only supports discrete vari-

	Scenario 1					Scenario 2 (negotiation) Init. variant 1	
Number of added vehicles	1	2	3	4	5	1	2
Smallest Time [s]	13	24	30	40	52	347	505
Longest Time [s]	17	50	151	142	167	817	1798

Table 6. Full exploration time for Scenario 1 and variant 1 of Scenario 2 using negotiation, while vehicles are added with random initial position and speed values. A few random initializations are performed each time and the smallest and longest observed exploration times are given.

ables. Therefore, scaling variables and the use of rounding on divisions is recommended to avoid or at least reduce error accumulations. Also, if there are variables needed for the decision or communication protocol, they should be added in the data structure. Their ranges must be defined, and it is recommended to keep those ranges as small as possible. The main limitation arises from "time-sensible" parameters, mainly the frequency of decision making, which can hardly exceed 20 Hz. However, as mentioned previously in Section 4, this limitation seems fair enough to model most of the decision protocols described in the literature. Concerning calibration, we need to find a balance between the precision on the longitudinal position (*i.e.*, the normalized maximal loss of precision parameter, Norm_x) and the frequency of the updates. The former allows to decrease the computation times at a cost in precision and the latter to gain accuracy on the system's monitoring at the cost of increasing computation times. In order to estimate an acceptable loss of precision, one may first proceed by simulation, which can also be done with UPPAAL. One may study a given indicator using several Norm_x values, then compare them to the result obtained with the Norm_x that guarantees no loss of information (*i.e.*, with $p = 1$ if we refer to section 6). The user has to choose the greatest Norm_x for which the observed error is still acceptable (this may require an expertise in the domain).

Finally, we summarize all the assumptions and abstractions that are inherent to VERIFCAR:

- Sensors are perfect (accurate and immediate);
- Decision is taken on a constant accurate frequency;
- Communication latency occurs in a defined and finite time interval;
- Frequency of decision for a given vehicle is greater or equal to the maximal communication latency for the same vehicle;
- Physical laws are simplified (rotation, friction, inertia do not exist);
- Environment is flat (only two dimensions are considered).

8 Conclusion

We presented VERIFCAR, a framework based on timed automata and dedicated to modeling and verifying CAVs. VERIFCAR is suitable for studying the behavior of CAVs at a rather high level of abstraction and addresses questions of safety, efficiency and robustness with a specific focus on the impact of latencies, communication delays and failures on the behavior of CAVs. It is particularly well adapted for the exhaustive analysis of critical situations involving a few number of vehicles, such as overtaking, insertion lanes, crossroads or traffic roundabouts.

VERIFCAR builds on general concepts borrowed from [4] to model such systems of CAVs but brings several novel ideas and improvements. First, it implements a totally different timed automata model, using broadcast synchronizations instead of handshake, which greatly simplifies the automata and contributes to speed up the verification processes. Thanks to suitable discretization and approximations, the exhaustive techniques of model checking can be used while limiting the state explosion phenomena. Next, it supports two additional forms of cooperation: negotiations between CAVs and communications with an intelligent road infrastructure. Finally, it provides pertinent indicators for the analysis of behaviors, together with a computation methodology using model checking. As shown in a companion paper [5], this method can be used jointly with simulations, contributing to give more insight on the studied systems.

As an illustration, we applied VERIFCAR to assess and compare decision policies for CAV systems on well chosen scenarios. To do so we implemented decision algorithms (including negotiations and infrastructure), computed the associated indicators and discussed the obtained results. We also checked the robustness through faulty environments, pointing out vulnerabilities and thus illustrating how VERIFCAR could be used to detect and thwart those vulnerabilities. All source material is available at `https://forge.ibisc.univ-evry.fr/jarcile/VerifCar/`.

As a future work, we plan to implement interfaces for VERIFCAR in order to allow an automated generation of models and indicators. Also, in order to verify more complex situations (with more vehicles, more complex road configurations, more involved interactions, etc) while facing successfully the classical explosion problems, we could consider abstraction techniques (like in [11]) and dedicated verification algorithms to reduce the computation time.

Finally, we plan to check if the techniques we developed to perform efficient modeling and model checking may be transposed to other application domains, different from CAV systems.

References

1. R. Alur, C. Courcoubetis, and D. Dill. Model checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.
2. R. Alur and D. Dill. Automata for modelling real-time systems. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP'90)*, volume 443 of *LNCS*, pages 322–335. Springer-Verlag, 1990.
3. R. Alur and D. Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.
4. J. Arcile, R. Devillers, H. Klaudel, W. Klaudel, and B. Woźna-Szcześniak. Modeling and checking robustness of communicating autonomous vehicles. In Sigeru Omatu, Sara Rodríguez, Gabriel Villarrubia, Pedro Faria, Paweł Sitek, and Javier Prieto, editors, *Distributed Computing and Artificial Intelligence, 14th International Conference*, pages 173–180. Springer International Publishing, 2018.

5. J. Arcile, J. Sobieraj, H. Klaudel, and G. Hutzler. Combination of simulation and model-checking for the analysis of autonomous vehicles' behaviors: a case study. In *Multi-Agent Systems and Agreement Technologies*. Springer International Publishing, 2018.
6. F. Bai and H. Krishnan. Reliability analysis of DSRC wireless communication for vehicle safety applications. In *IEEE Intelligent Transportation Systems Conference*, pages 355–362, Sept 2006.
7. K. Bilstrup, E. Uhlemann, E. G. Strom, and U. Bilstrup. Evaluation of the ieee 802.11p mac method for vehicle-to-vehicle communication. In *IEEE Vehicular Technology Conference*, pages 1–5, Sept 2008.
8. S. Biswas, R. Tatchikou, and F. Dion. Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety. *IEEE Communications Magazine*, 44(1):74–82, Jan 2006.
9. R. J. Blokpoel, D. Krajzewicz, and R. Nippold. Unambiguous metrics for evaluation of traffic networks. In *IEEE Intelligent Transportation Systems Conference*, pages 1277–1282, Sept 2010.
10. E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.
11. R. Devillers and H. Klaudel. Abstraction strategies for computing travelling or looping durations in networks of timed automata. In Martin Fränzle and Nicolas Markey, editors, *14th International Conference, FORMATS 2016, Proceedings*, volume 9884 of *LNCS*, pages 140–156. Springer, 2016.
12. E. Allen Emerson. Handbook of theoretical computer science (vol. b). chapter Temporal and Modal Logic, pages 995–1072. MIT Press, Cambridge, MA, USA, 1990.
13. C. Urmson et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics Special Issue on the 2007 DARPA Urban Challenge, Part I*, 25(8):425–466, June 2008.
14. M. Foughali, B. Berthomieu, S. Dal Zilio, F. Ingrand, and A. Mallet. Model Checking Real-Time Properties on the Functional Layer of Autonomous Robots. In *International Conference on Formal Engineering Methods (ICFEM 2016)*, Tokyo, Japan, November 2016.
15. A. Furda and L. Vlacic. Enabling safe autonomous driving in real-world city traffic using multiple criteria decision making. *IEEE Intelligent Transportation Systems Magazine*, 3(1):4–17, Spring 2011.
16. S. Glaser, B. Vanholme, S. Mammar, D. Gruyer, and L. Nouveliere. Maneuver-based trajectory planning for highly autonomous vehicles on real road with traffic and driver interaction. *IEEE Transactions on Intelligent Transportation Systems*, 11(3):589–606, Sept 2010.
17. K.D. Stanley P. Sorensen C. Samaras J.M. Anderson, N. Kalra and T. A. Oluwatola. *Autonomous Vehicle Technology. A Guide for Policymakers*. Research Reports. RAND Corporation, 2016. ISBN: 978-0-8330-8398-2.
18. M. Kamali, L. A. Dennis, O. McAree, M. Fisher, and S. M. Veres. Formal verification of autonomous vehicle platooning. *Science of Computer Programming*, 148:88 – 106, 2017. Special issue on Automated Verification of Critical Systems (AVoCS 2015).
19. C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. *Transportation Research Part C: Emerging Technologies*, 60:416 – 442, 2015.
20. S. Kong, S. Gao, W. Chen, and E. Clarke. dreach: δ-reachability analysis for hybrid systems. In Christel Baier and Cesare Tinelli, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 200–205, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.
21. Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, 17(5):1105–1118, Sept 2009.

22. J. Levinson and S. Thrun. Robust vehicle localization in urban environments using probabilistic maps. In *IEEE International Conference on Robotics and Automation*, pages 4372–4378, May 2010.
23. M. Likhachev and D. Ferguson. Planning long dynamically feasible maneuvers for autonomous vehicles. *The International Journal of Robotics Research*, 28(8):933–945, 2009.
24. M. Minderhoud and P. Bovy. Extended time-to-collision measures for road traffic safety assessment. *Accident Analysis & Prevention*, 33(1):89 – 97, 2001.
25. M. O'Kelly, H. Abbas, and R. Mangharam. APEX : Autonomous vehicle plan verification and execution. In *SAE World Congress*, 2016.
26. A. Platzer and J.-D. Quesel. European train control system: A case study in formal verification. In Karin Breitman and Ana Cavalcanti, editors, *Formal Methods and Software Engineering*, pages 246–265, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
27. M. M. Quottrup, T. Bak, and R. I. Zamanabadi. Multi-robot planning : a timed automata approach. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 5, pages 4417–4422 Vol.5, April 2004.
28. M. Treiber and A. Kesting. *Trajectory and Floating-Car Data*, pages 7–12. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
29. Uppaal. `http://www.uppaal.org/`.
30. K. Vogel. A comparison of headway and time to collision as safety indicators. *Accident Analysis & Prevention*, 35(3):427 – 433, 2003.
31. T. L. Willke, P. Tientrakool, and N. F. Maxemchuk. A survey of inter-vehicle communication protocols and their applications. *IEEE Communications Surveys Tutorials*, 11(2):3–20, Second 2009.
32. M. Wooldridge. *An introduction to multi-agent systems - Second Edition*. John Wiley & Sons, 2009.
33. S. Zhang, W. Deng, Q. Zhao, H. Sun, and B. Litkouhi. Dynamic trajectory planning for vehicle autonomous driving. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 4161–4166, Oct 2013.

A TTC Algorithm

Algorithm 3 formalizes the computation of the TTC value used in our experiments. px_i (respectively py_i) is the longitudinal (respectively lateral) position of vehicle i, and *Long* (respectively *Lat*) is the longitudinal (respectively lateral) gap below which there is a collision. Similarly, vx_i (respectively vy_i) is the longitudinal speed (respectively lateral speed) value of vehicle i. *Long* and *Lat* depend on each vehicle length and width: typically, as the position is centered on the vehicle, *Long* will be the mean of the length of the two vehicles, while *Lat* will be the mean of the width of the two vehicles.

This algorithm consists in computing the starting time X_{in} and the ending time X_{out} of the longitudinal time overlap (respectively Y_{in} and Y_{out} for the lateral time overlap). If no overlapping (either longitudinal or lateral) is ever going to occur, the starting time will be given the value ∞ and the ending time will be set to 0. Afterwards, the computation of TTC is performed by returning the smallest value in the intersection between longitudinal and lateral time overlaps.

Algorithm 3 Computes TTC between vehicles A and B

```
                    {Computation of $X_{in}$}
if $(px_b \geq px_a$ and $vx_b < vx_a)$ or $(px_a \geq px_b$ and $vx_a < vx_b)$ then
  $X_{in} = \frac{|px_b - px_a| - Long}{|vx_a - vx_b|}$        {A and B getting closer to each other}
else if $|px_b - px_a| < Long$ then
  $X_{in} = 0$        {Already overlapping}
else
  $X_{in} = \infty$        {Will never overlap}
end if
                    {Computation of $X_{out}$}
if $(px_b \geq px_a$ and $vx_b < vx_a)$ or $(px_a \geq px_b$ and $vx_a < vx_b)$ then
  $X_{out} = \frac{|px_b - px_a| + Long}{|vx_a - vx_b|}$        {A and B getting closer to each other}
else if $vx_a \neq vx_b$ then
  $X_{out} = \frac{||px_b - px_a| - Long|}{|vx_a - vx_b|}$        {A and B moving away from each other}
else if $|px_b - px_a| < Long$ then
  $X_{out} = \infty$        {Already overlapping}
else
  $X_{out} = 0$        {Will never overlap}
end if
                    {Computation of $Y_{in}$}
if $py_b \geq py_a$ and $vy_b < vy_a$ then
  $Y_{in} = \frac{py_b - py_a - Lat}{vy_a - vy_b}$        {B on the left of A, getting closer to each other}
else if $py_a \geq py_b$ and $vy_a < vy_b$ then
  $Y_{in} = \frac{py_a - py_b - Lat}{vy_b - vy_a}$        {A on the left of B, getting closer to each other}
else if $|py_b - py_a| < Lat$ then
  $Y_{in} = 0$        {Already overlapping}
else
  $Y_{in} = \infty$        {Will never overlap}
end if
                    {Computation of $Y_{out}$}
if $py_b \geq py_a$ and $vy_b < vy_a$ then
  $Y_{out} = \frac{py_b - py_a + Lat}{vy_a - vy_b}$        {B on the left of A, getting closer to each other}
else if $py_b \geq py_a$ and $vy_a \neq vy_b$ then
  $Y_{out} = \frac{py_b - py_a - Lat}{vy_a - vy_b}$        {B on the left of A, moving away from each other}
else if $py_a \geq py_b$ and $vy_a < vy_b$ then
  $Y_{out} = \frac{py_a - py_b + Lat}{vy_b - vy_a}$        {A on the left of B, getting closer to each other}
else if $vy_a \neq vy_b$ then
  $Y_{out} = \frac{py_a - py_b - Lat}{vy_b - vy_a}$        {A on the left of B, moving away from each other}
else if $|py_b - py_a| < Lat$ then
  $Y_{out} = \infty$        {Already overlapping}
else
  $Y_{out} = 0$        {Will never overlap}
end if
                    {Computation of TTC}
if $X_{in} \leq Y_{in}$ and $Y_{in} \leq X_{out}$ then
  return $Y_{in}$        {Lateral time overlap starting during longitudinal time overlap}
else if $Y_{in} \leq X_{in}$ and $X_{in} \leq Y_{out}$ then
  return $X_{in}$        {Longitudinal time overlap starting during lateral time overlap}
else
  return $\infty$        {No intersection between time overlaps}
end if
```