Compositional verification for reactive multi-agent systems applied to platoon non collision verification

Madeleine EL-ZAHER, Jean-Michel CONTET, Pablo GRUER, Franck GECHTER, Abderrafiaa KOUKAM

IRTES-SeT, UTBM, 9010 Belfort cedex, France {firstname.lastname}@utbm.fr

Abstract. *This paper presents a methodology for the verification of reactive multi-agent systems (RMAS). High level of confidence about a safe operation is a mandatory in many reactive applications. Model-checking appear as an adequate tool for the verification of safety properties. However, model-checking can be confronted with the problem of intractable state space sizes. To avoid this kind of limitation, it is possible to apply verification methods based on abstraction or composition. This paper presents a compositional verification method adapted to a wide range of RMAS applications. This method is appropriate for the verification of safety properties. The application considered in this paper is a platoon of vehicles with linear configuration. The safety property to be verified is the non collision between platoon vehicles. The SAL tool-kit has been adopted as a verification tool, by applying SAL model checkers. The verification method bases on a compositional verification rule.*

Keywords: *Multi-agent systems, Model-checking, Compositional verification, Platoon application*

1. Introduction

Reactive multi-agent systems (RMAS) [CGS+07] appear to be a promising approach for the design of many reactive concurrent systems. Such systems rely on reactive and autonomous agents, that

behave based on their own perceptions [Fer99]. In RMAS, global intelligent behavior emerges as a result of individual autonomous behavior of each agent. Each agent behavior can be specified as a function of its perceptions and interactions. RMAS show some features like self-organization, robustness, component simplicity, redundancy and also low cost agent design. Consequently, reactive multi-agent systems have been adopted in the design of many complex applications, in fields such as simulation [DDHP10, ER01], problem solving [FD92], placement problems [MSK09] a road traffic modelling [EHD04].
However, to be considered adequate for the design of reactive embedded systems, RMAS must verify safety requirements that depend on the application.
Verification is on the way to become a current, if not mandatory stage in the design process of reactive applications. Verification consists in formally establishing the logical validity of some logic formula F, relatively to a behavioral model M of the application. Relative validity means that every possible evolution of the system, as determined by M, satisfies F. In this paper, we interest in formulas that express a safety property, generally by assessing that some logic formula F is an invariant of M. F is invariant if F is satisfied by every reachable state of the model M.
The goal of this paper is to show a verification approach applied to a reactive multi-agent system, version of a vehicle platoon.
This paper is organized as follow : In section 2 a state of the art about the different existing verification method is proposed. Section 3 gives a general definition of the compositional verification method. In section 4, a concrete application is described : Linear platoon system and its physics inspired interaction model. Section 5 presents the SAL specification models of vehicle behavior and the application of compositional verification. In section 6, we dress a comparison between compositional verification and direct verification. Finally in the conclusion some remarks and future work close to this presentation are mentioned.

2. State of the art

Two families of verification algorithms have produced efficient tools : model-checking [DMRS03] and theorem-proving [SSC08]. Model-checkers are adequate tools to perform invariance verifications, but suffer from two limitations. Firstly, model-checking is confronted to the exploration of infinite state spaces if the model includes unbounded variables. This situation leads non termination of the model-checking process. To address this problem, new families of model checkers are able to perform bounded depth exploration. If no counter-example is found, an additional proof step based on induction can be performed [DMRS03]. Of course, this induction step is not guaranteed to succeed, but in many cases it succeeds and allows model-checking termination with a result.
Secondly, even in the eventuality of finite state spaces, if the system is composed of a large number of components each one of which could in turn be quite complex, the system could have a very big global state space, resulting from the composition of all components. Direct verification on the global state space can be prohibitive, in terms of calculation time.
Compositional verification [WS03, MS08] appears as methodological approach addressing this kind of difficulties. Compositional verification consists on verifying separately a set of auxiliary properties relative to system components. Then, to apply a deduction rule to these auxiliary property in order to establish the global property.
Compositional reasoning can facilitate the specification of systems by decomposing the system into small parts and attributing to each one a property that characterizes its behavior. If the conjunction of the local properties implies the hole system specification, it will be sufficient to check each local property in the part that it describes and then verify if the system satisfies each local property. According to [BBNJ10] two main approaches for compositional verification exist : assume-guarantee and deductive approach.
The assume-guarantee-paradigm for compositional reasoning has been introduced by Amir Pnueli in [Pnu85] it was also detailed in [CGP99, GL91, AL95]. This technique relies on the decomposition of the system properties into two parts : An assumption about the

components behavior, the second is a property guaranteed by the components. The main problem of this approaches is to find the adequate assumptions for a particular decomposition.
The assume-guarantee approach of compositional verification was used in different application, for example in [CIP04] compositional verification was used to model check middleware-based software-architecture with respect to a subset of Linear Temporal Logic system properties. [JJ98] and [CMJT97], shows examples of the application of the compositional method for respectively the verification of multi-agent system, and the verification of knowledge-based systems, both uses the compositional modeling framework DESIRE [BDKJT95].
This paper presents an approach to the compositional verification adapted to a large class of RMAS application. This applications are characterized by a high level of autonomy of the agents, and by local perceptions and interactions that allow to assume that the required properties can be verified locally, by abstracting from the rest of the system, thereby justifying the compositional approach. Furthermore, the approach is well adapted to systems composed of similar agents. Different instances of this kind of RMAS result from successive incorporations of a new instance of an agent.

Figure 1 – Platoon vehicles in a simulation area

The application we adopt to illustrate the verification approach is a linear Platoon system. Linear Platoons are sets of vehicles that circulate while keeping a train configuration without any material coupling. Our proposal bases on a decentralized, local approach to vehicle platoon organization. Each car is an autonomous agent that behaves based only

on its own perceptions capabilities. The approach adopted in this work is characterized by the following features : Agent's behavior is specified by a physical interaction model. Inputs of this interaction model are agent's perceptions, and the outputs are longitudinal and lateral control references, which are inputs for the regulation and command stages. The safety condition to be verified is the non-collision between vehicles during platoon operation. As those models include naturally unbounded real variables, specification and verification are performed using the SAL toolkit [BGL$^+$00].

3. A compositional verification method

In this paper, we are interested in the verification of the validity of a logic formula F, relatively to a behavioral model M. Relative validity means that every possible evolution of the system as determined by M satisfies F. Some notational conventions will be used here : given the properties $F_1, \cdots, F_n$, we will note $M \vDash F_1, \cdots, F_n$ when all the properties F_i are valid relatively to the system model. These properties are evaluated over an infinite sequences σ of model states. We will note $\mathcal{B}(M)$ the set of sequences produced by model M (its behavior) and $Sat(F)$ the set of sequences that satisfy the property F. Then, $M \vDash F$ if and only if $\mathcal{B}(M) \subseteq Sat(F)$. We note $M, F_1 \vDash F_2$ if property F_2 is satisfied by model M under the hypothesis F_1.

We formulate a compositional verification rule to be applied to an instance M_i of the system model M. Instance M_i results from instance M_{i-1} by adding a new component C_i (the composition is noted $M_i = C_i \| M_{i-1}$). In our case, each component C_i is an agent that interacts only through its own perceptions. There are no agent-to-agent communication or synchronisation operation. Consequently, $\|$ is considered to be an asynchronous composition operator.
The main idea with the compositional verification rule is to simplify the verification of logica formula S_i, expressing a safety property relative to model instance M_i, by replacing M_{i-1} by an auxiliary property T_{i-1}, introduced as an hypothesis. We emphasize the fact that hypothesis T_{i-1} referres to model M_{i-1}. The compositional verification rule can

then be expressed as follows :

$$R_c : \frac{C_i, T_{i-1} \vDash T_i \quad C_{i+1}, T_i \vDash S_{i+1}}{C_{i+1} \| M_i \vDash S_{i+1}} \tag{1}$$

which means that if, under hypothesis T_{i-1}, the new hypothesis T_i is valid relativelly to component C_i and if safety property S_{i+1} is valid relativelly to component C_{i+1}, under hypothesis T_i, then the safety property S_{i+1} is valid relativelly to the model $C_{i+1} \| M_i$. On the basis of this rule, a compositional verification process can be described as follow :

$$\begin{cases} \textbf{verify } C_0 \vDash T_0 \textbf{ and } C_1, T_0 \vDash S_1 \\ \textbf{for } i > 1 \textbf{ while } C_i, T_{i-1} \vDash T_i \textbf{ apply } R_c \textbf{ to } C_{i+1}, T_i \end{cases}$$

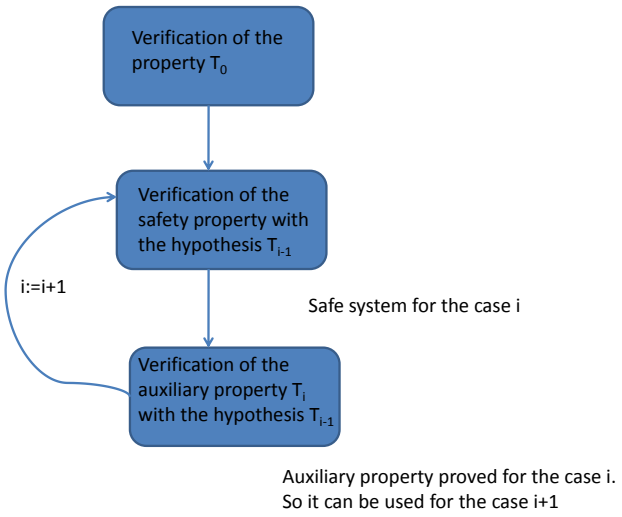

Figure 2 – Application of the compositional verification method

The verification process, illustrated by figure 2, begins with the instance $M_0 = C_0$. Then at each step of verification two properties

have to be verified :

- Safety property, noted S_i, relative to M_i, composed by the instance M_{i-1} and the new component C_i.
- An auxiliary property T_i, relative to the system M_i.

It can be proved that rule (1) is consistent if the addition of a new component C_{i+1} does not influence the behavior of system M_i. A deduction rule is consistent if, whenever the premisses are satisfied the conclusion is also satisfied. This assumption is certainly strong, but many RMAS comply with it, specially those in which agent behavior bases on local perceptions, as our illustrative study will show. In what comes to the auxiliary property T_i, it should be noted that in general, it will not differ formally from the preceding instance T_{i-1}. Only the value of some parameter(s) appearing in the property will change. Farther discussion of this aspect will be presented in the case-study.

4. The application

4.1. *General description*

The application presented here is a vehicle platoon system. A platoon is a set of vehicles that circulate while keeping a specific configuration on terrain, without the intervention of any physical coupling mechanism. We are interested in linear platoons, that have a train configuration. Our approach consists in considering a platoon as a RMAS, each vehicle embodying an autonomous agent.
The platoon control problem consists in specifying vehicle-agent's behavior in order to obtain the intended global platoon's behavior. It can be decomposed in two sub-problems : longitudinal control (distance regulation) and lateral control (direction regulation). Many lateral or longitudinal control approaches base on the PID scheme (Proportional, Integral, Derivative) [IX94, DP96]. Some researches propose also an integrated longitudinal and lateral control as in [WC01]. A reactive approach with autonomous longitudinal and lateral control has been developed in [CGGK09]. In this paper, vehicle's behavior is specified using

a physically inspired interaction model, as described next.
The platoon multi-agent system proposed in this paper is composed of agents each one corresponding to a vehicle. The behavior of each agent is the result of its own perceptions. We can distinguish between two main behaviors. Firstly, the leader behavior concerns the leader vehicle, which perceives the environment and follows a predefined trajectory. Leader's perceptions base mainly on artificial vision. Secondly the follower behavior concerns follower agents. every follower agent perceives only its immediately preceding vehicle in the platoon. Follower perceptions are mostly based on distance-measuring devices : stereo vision, laser range finders ...
In this paper, we are interested in the follower vehicle behavior. A physics inspired interaction model for a follower agent is introduced, to specify the reactive behavior of the agent based on its own perceptions. This interaction model is described in the following section.

4.2. *Agent's interaction model*

A follower agent behaves as if it was connected to the platoon by a means a virtual, physics-inspired mechanism. The first vehicle is autonomous or driven by a human pilot. Each one of the follower vehicles follows the trajectory of its predecessor within the platoon. The predecessor is the nearest platoon vehicle in the direction of movement. A follower vehicle has perceptive capability to measure the distance vector (direction and magnitude) to its predecessor. From this measurement, it calculates and deduces interaction force, based on the virtual, physics inspired interaction model, composed of two springs and a damper, as shown by figure 3. The virtual coupling influences only on the follower vehicle, as a matter of fact, there is no mutual interaction as in the case of a material coupling mechanism. Model parameters are, k_1 and k_2, the stiffness of each one of the springs, h damping factor, L_v spring resting length and, m the vehicle's mass.

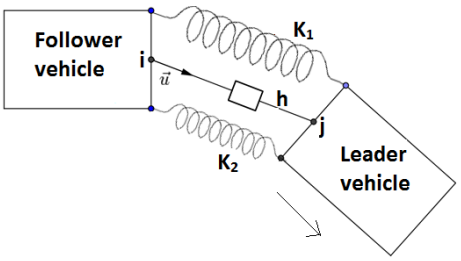

Figure 3 – Physical interaction model

Each follower agent measures the distances at three points :

– D : Length of the damper.

– d_1 : Length of the first spring (Spring on the left).

– d_2 : Length of the second spring (Spring on the right).

Note that when a platoon in a train-configuration is moving along a line, the three distances D, d_1 and d_2 are equals. Forces involved in this model are $\vec{F_1}$, the force of the first spring, $\vec{F_2}$, the force of the second spring and the damping force $\vec{F_h}$.
To calculate vehicle's acceleration in the follower vehicle reference frame, the Newton law of motion is used :

– **Force of each spring :** $\vec{F_i} = k_i(d_i - l_v)\vec{u}$ with $i \in \{1, 2\}$

– **Force of Damper :** $\vec{F_d} = h(||\frac{\Delta D}{\Delta t}||)\vec{u}$

Where $\vec{u} = \vec{ij}/\|ij\|$ (cf. figure 3).

$$m * \vec{\gamma} = (h\frac{\Delta D}{\Delta t} + k_1(d_1 - l_v) + k_2(d_2 - l_v))\vec{u} \tag{2}$$

In linear circulation this equation could be written as :

$$m * \vec{\gamma} = (h\frac{\Delta D}{\Delta t} + k(D - l_v))\vec{u} \tag{3}$$

By discrete integration, vehicle's state (speed, orientation and position) can then be determined and the command law can be computed. Equation 2 is the base for the specification of agent's behavior. This equation will be applied to calculate, at each agent's operation cycle, a new acceleration command reference to be sent to vehicle's controller.

5. Compositional verification

In this section, a verification case-study is presented. It consists in verifying a safety property relative to the platoon RMAS : non-collision of any follower vehicle with its preceding vehicle, during platoon operation, with constant speed, along a linear trajectory.
The behavior of each vehicle in the platoon can be described as shown in figure 4, a cyclic combining of the three sub-behaviors :

– *Perception* : Perceives the inter-vehicle distance (D), d_1 and d_2 . These distances are used in the calculation of the vehicle references.

– *Vehicle Control* : Regulates the inter-vehicle distance to be to be greater then a safety distance (Assumed to be the minimal inter-vehicle distance that could be reached without having risk of collision between vehicles). It computes vehicle acceleration and vehicle references (speed and orientation) based on the interactional model defined above.

– *Physical Model* : Computes vehicle's reaction as a function of its dynamic characteristics and speed. As a matter of fact, this sub-behavior is added for verification purposes, as described later.

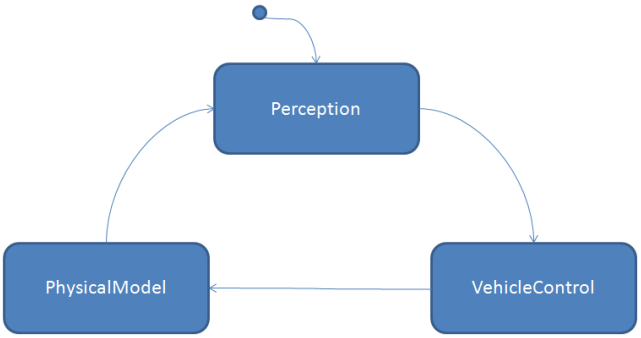

Figure 4 – Behavior cycle of a vehicle agent

SAL [1] toolkit was adopted as a verification and modeling framework. SAL includes a modeling language for the construction of behavioral models based on the transition-system paradigm. The language includes temporal logic expressions adapted to the formulation of properties.

1. http ://sal.csl.sri.com/

SAL also includes a set of model-checkers with different characteristics. For this case-study, the SAL bounded model-checker (bmc) has been adopted, because it can be applied to system models that include real variables. The presence of real variables induces infinite (and even non-enumerable) state spaces, introduces the possibility of systematic non-termination and requires specific model-checking algorithms. SAL bmc uses Yices[2], a SMT (Satisfiability Modulo Theories) solver that decides the satisfiability of arbitrary formulas. Yices partially avoids the non-termination problem by a mechanism of k-induction, where k is an integer representing an exploration depth. The induction mechanism, of course, is not guaranteed to terminate in all cases, but avoids systematic non-termination.

5.1. *Vehicle context*

A SAL model includes a number of transition-system modules, encapsulated in a context, that contains definitions of constants, types, functions and modules. A basic SAL *module* is a state-transition system where the state consists of input, output, local, and global variables. SAL modules can be composed synchronously, so that $M_1 \| M_2$ is a module that takes M_1 and M_2 transitions in a lock step, or asynchronously, where $M_1 [\,] M_2$ is a module that takes an interleaving of transitions from M_1 and from M_2. In our case, a $Vehicle$ context represents the behavioral model of the follower vehicle agent and includes three modules : *Perception*, *VehicleControl* and *PhysicalModel*. Each one of which corresponds to one of the three sub-behaviors described in figure 4. With this composition mechanism, model behavior results from the interleaving of individual module transitions. Our system complies with this mechanism because of cyclic operation : only one component is active a time.

2. http ://yices.csl.sri.com/

```
Vehicle : CONTEXT =
BEGIN
Constants definition
Types definition
Variables declaration
Functions definition
PhysicalModel : MODULE = ···
VehicleControl : MODULE = ···
Perception : MODULE = ···
vehicleBehavior : MODULE = PhysicalModel [ ] VehicleControl [ ] Perception ;
END ;
```

The SAL script shown here summarizes the vehicle context. This context contains the definitions of the constants, types, variables, functions and modules used in the model. It also precise the module composition type. The asynchronous composition of the vehicle three subbehaviors is expressed in the last line of the script (vehicleBehavior : MODULE = PhysicalModel [ ] VehicleControl [ ] Perception).
Some detail is given now about model elements :

– **Constants definition :** This section assigns a value to each physical constant.

```
safetyDistance : REAL = 1.2 ;
minAcceleration : REAL = -1.3 ;
maxAcceleration : REAL = 1.3 ;
mass : REAL = 500 ;
.....
```

– **Types definition :** This section introduces three enumerated types to represent internal states of each one of the modules.

```
PerceptionState : TYPE = {Pidle,Pwait,Pread} ;
VehicleControlState : TYPE = {Ridle,Rwait,Rcompute,Rstop} ;
PhysicalModelState : TYPE = {Pwait,PCompute,Plimits} ;
```

– **Functions definition :** This section defines the different functions used for modules' calculations. For example, the *"computeAcceleration"* function calculates the acceleration of the vehicle, based on the second law of Newton (cf. equation 2). The constant "*regularDistance*" corresponds to the spring resting length, it also represents the desired inter-vehicle distance.

computeAcceleration(_distance : REAL, _d_1 : REAL, _d_2 :REAL) : REAL = (k_1(_d_1-regularDistance)+ (k_2*(_d_2-regularDistance) - (h*_distance)/delay)/mass ;*
....

– **Module definition :** Each vehicle's sub behavior is described by a module. Modules use shared variables. Module variables can be *LOCAL*, *GLOBAL*, *INPUT* or *OUTPUT*. Boolean variables simulating events are used to synchronize module executions. State transformations are described with "after/before" predicates, where variable value after the transformation is represented by primed identifiers and variable value before the transformation is represented by unprimed identifiers. Below the $Perception$ and the $VehicleControl$ module are described :

perception : MODULE =
BEGIN
LOCAL vehicleState : PerceptionState
LOCAL Deltadistance, DeltaSpeedPre, intervalSpeed :REAL
GLOBAL start, endAction : BOOLEAN
GLOBAL endPercept, endPhycalModel : BOOLEAN
INPUT speed : REAL
OUTPUT D, d_1, d_2 : REAL
Definition DeltaSpeedPre IN {x : REAL | x >= DeltaSpeedMinPre
AND x<= DeltaSpeedMaxPre}
INITIALIZATION vehState = Pidle
INITIALIZATION endPercept = FALSE
INITIALIZATION START = FALSE ; endPhysicalModel = TRUE
INITIALIZATION D IN {x : REAL | x >= SafetyDistance
AND x<= limitDistance}
TRANSITION
[
vehicleState = Pidle And Start −− >
start' = FALSE ;
vehicleState' = Pwait
[]
vehicleState = Pwait AND endPhysicalModel
vehicleState'= Pread
endPhysicalModel' = FALSE
[]
vehicleState = Pread
D' = D + ComputeDistance ?(speed, DeltaSpeed)
$d_1' = D$
$d_2' = D$
DeltaDistance' = ComputeDistance ?(speed, DeltaSpeed)
intervalSpeed' = DeltaDistance / delay
vehicleState' = Pwait
endPercept' = TRUE
]
END ;

Goal of the $Perception$ module is to calculate the three distances D, d_1 and d_2, to be used in the computation of the acceleration in the

$VehicleControl$ module. At the first iteration, D is chosen between *SafetyDistance* and the *limitDistacne* (limitDistance corresponds to the maximal range of perception). As we are in linear circulation d_1 and d_2 are equal to D. The variable *Speed* represents the speed of the follower vehicle, this variable is an input in the $Perception$ module, deduced from the $PhysicalModel$ module. The variable *DeltaSpeed* represents the variation of speed between the follower vehicle and its previous, so the speed of the leader vehicle is the sum of the speed of the follower vehicle and the variation of speed. *DeltaSpeed* is chosen between *DeltaSpeedMinPre* and *DeltaSpeedMaxPre* that correspond to the maximal and minimal speed variation of the vehicle.

```
VehicleControl : MODULE =
BEGIN
    LOCAL vehicleState : VehicleControlState
    GLOBAL start, endAction : BOOLEAN
    GLOBAL endPercept, endPhycalModel : BOOLEAN
    INPUT D, d1, d2 : REAL
    OUTPUT acceleration : REAL
    INITIALIZATION acceleration = 0 ; vehState = Ridle
    TRANSITION
    [
        vehicleState = Ridle −− >
            start' = TRUE ;
            vehicleState' = Rwait
    []
        vehicleState = Rwait AND endPercept AND
        D < safetyDistance −− >
            acceleration' = maxDeceleration
            vehicleState'= Act
            endPercept' = FALSE
    []
        vehicleState = Rwait AND endPercept AND
        D >= safetyDistance −− >
            acceleration' = computeAcceleration ?( D,d1, d2)
            vehicleState'= Act
            endPercept' = FALSE
    []
            vehicleState = Act −− >
            endAction' = TRUE
            vehicleState'= Rwait
    ]
END ;
```

VehicleControl module calculates the acceleration of the vehicle, based on the three measured distances produced by the *Perception* module. The acceleration is computed using the function *ComputeAcceleartion* that applies the equation 3.

5.2. *Compositional verification with SAL*

As mentioned before, the case considered here is train-configuration platoon moving along a line, where $d_1 = d_2 = D$ (cf. figure 3), and the safety condition to be verified is the non collision between two successive vehicles (i.e inter-vehicle distance must be greater then a predefined safety distance). Following SAL statement expresses this safety condition.

```
SafetyCondition : THEOREM Vehicle |- G( D >= safetyDistance) ;
```

Where G is a temporal operator that express invariance : $G(P)$ means that P is satisfied by every state. Invariance is a way to express validity of P, relatively to system evolutions.
As already stated, the compositional rule can be applied under the assumption that adding a new component to a model M does not modify its behavior. The platoon system presented here satisfies this assumption, since adding a new vehicle to the train does not modify the behavior of the other preceding vehicles. This property is due to the local platoon control strategy, where each agent perceives only the distance to the preceding vehicle.
The compositional iterative verification method is based on the use of auxiliary properties T_i. There is no general principle to formulate this property, its choice depends on the nature of the model. In this case study, vehicle's speed has the major influence on risk of collision. For this reason, we adopt an auxiliary property that expresses a condition relative to the speed increments of vehicle V_i, the property is given by the following equation :

$$T_i \equiv DeltaSpeedMin_i \leq deltaSpeed_i \leq DeltaSpeedMax_i \quad (4)$$

Values of $DeltaSpeedMin_0$ and $DeltaSpeedMax_0$ that correspond to T_0, depends on the vehicle characteristics. To express the instances T_i and T_{i+1} of the auxiliary property, a set of constants has been declared :

```
%% Constants definition
DeltaSpeedMinPre : REAL = ... ;
DeltaSpeedMaxPre : REAL = ... ;
DeltaSpeedMinPost : REAL = ... ;
DeltaSpeedMaxPost : REAL = ... ;
%% Global variable declaration
deltaSped : REAL ;
```

Values of constants *DeltaSpeedMinPre* and *DeltaSpeedMaxPre* are used in the formulation of T_i, and values of *DeltaSpeedMinPost* and *DeltaSpeedMaxPost* in the formulation of T_{i+1}. The auxiliary properties are expressed, by writing the following theorem statements :

CurrentAuxiliaryProperty : THEOREM Vehicle |− G ($Deltapeed >= DeltaSpeedMinPre$ *AND* $DeltaSpeed <= DeltaSpeedMaxPre$*);*
NextAuxiliaryProperty : THEOREM Vehicle |− G ($DeltaSpeed >= DeltaSpeedMinPost$ *AND* $DeltaSpeed <= DeltaSpeedMaxPost$*);*

Where *CurrentAuxiliaryProperty* and *NextAuxiliaryProperty* represents respectively T_i and T_{i+1}. The verification of the premise $C_{i+1}, T_i \vDash S_{i+1}$ of the compositional verification rule R_c, is performed by the following SAL command :

sal-inf-bmc –depth=50 –induction -l CurrentAuxiliaryProperty Vehicle SafetyCondition

where the $-l$ option introduces the auxiliary property as an assumption (hypothesis).

The first step consists in verifying the auxiliary property T_0 which corresponds to the $CurrentAuxiliaryProperty$. Then, the iterative verification process can be started. Model checking is launched using the SAL verification command line :

sal-inf-bmc –depth=50 –induction -l CurrentAuxiliaryProperty Vehicle NextAuxiliaryProperty

to verify the validity of the auxiliary property to be used next. If this verification succeeds, $DeltaSpeedMinPost$ and $DeltaSpeedMaxPost$ became the constants $DeltaSpeedMinPre$ and $DeltaSpeedMaxPre$ of the next iteration. Space exploration is bounded to avoid non terminating search. If no counter example is found, an induction step is added using the $induction$ option. The validity of the safety condition has been verified up to five vehicles platoon.
Figure 5 illustrates a possible perception of the verification process.

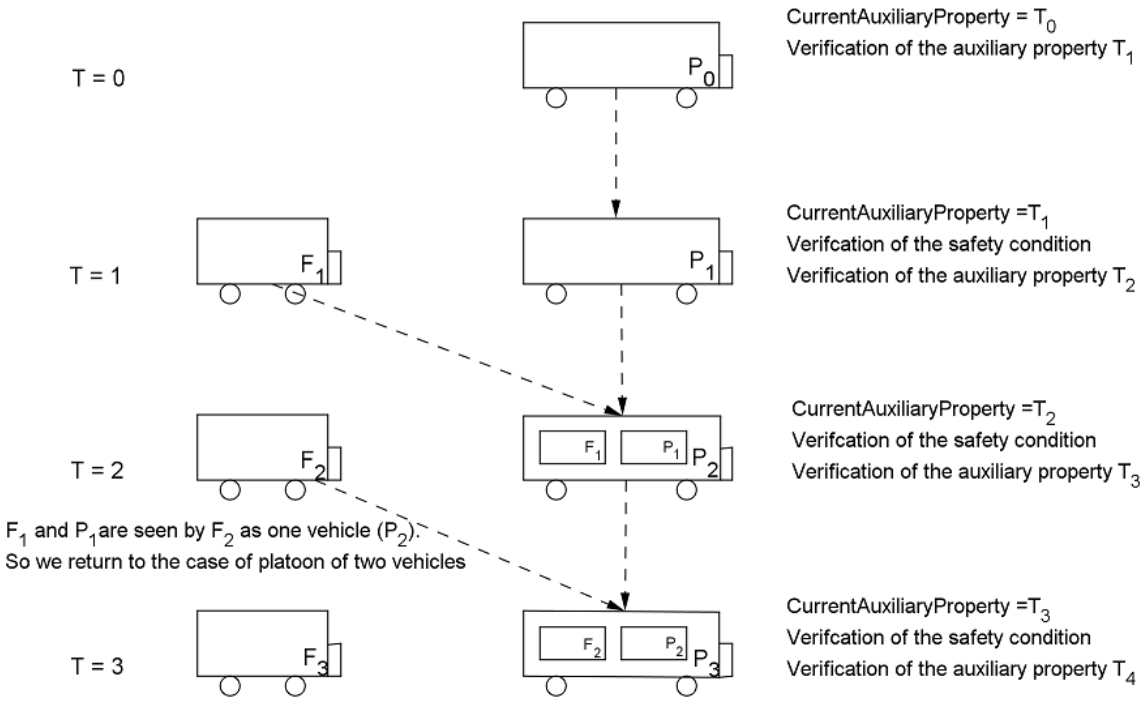

Figure 5 – Steps of the verification process

The compositional verification process consists on verifying at each iteration(i), the validity of the safety property : no collision between F_i and P_i.
At the first iteration (T=0), the model checking starts by the first component P_0, the safety condition is valid for this case.
At the second iteration (T=1), the vehicle P_0 is now called P_1, a new component (vehicle) is added to the system (Platoon), the verification of this case consists on verifying that there is no collision between F_1 and P_1 under the assumption T_1 : The speed variation of F_1 relatively to P_1 is between the $DeltaSpeedMin_1$ and $DeltaSpeedMax_1$, deduced from the previous step.
At each new iteration i, a new vehicle F_i is added to the platoon, and the vehicle P_i represents the combination of P_{i-1} and F_{i-1}, so P_i is seen by F_i as only one vehicle. The verification consists on considering F_i and P_i as a two vehicles system, and to verify that there is no collision bet-

ween F_i and P_i, under the assumption T_i : The speed variation of F_i relatively to P_i is between the $DeltaSpeedMin_i$ and $DeltaSpeedMax_i$.

6. Compositional VS. global verification

In order to evaluate the gain resulting from compositional verification, a comparison with direct, non compositional verification was done. This comparison consists on verifying a model composed of n agents vehicle.
In both verification cases (compositional and non compositional) the used vehicle has the same characteristics. These characteristics with the physical model parameters are summarized in table 1.

Number of vehicles	5
Vehicle mass	500 kg
Vehicle speed limit	4 m/s
Vehicle acceleration limit	1 m/s^2
Vehicle deceleration limit	-1 m/s^2
Desired inter-vehicle distance (l_v)	2 m
Safety distance	1 m

Tableau 1 – Parameters of the vehicles used in experimentations

6.1. *Non compositional verification*

The verification model is obtained by indexing the model of one agent. So, the behavior of the system can be modeled using :

SMAVehicle : ***CONTEXT***
BEGIN
SMAVehicleBehavior : ***MODULE*** *= Vehcile(1)[]...[]Vehicle (n)*
END ;

Table 2 shows the number of nodes explored during a direct verification of the safety property. The number of explored nodes is a statistical value given by the model checker at the end of the verification process. It represents the number of nodes that have been computed, until a node

violating the safety property has been found or the entire state space has been checked. As we can see, from four vehicles on, the model checker seems not to be able to terminate the verification, the model checker has been let to run for five days without stopping and giving an answer.

Number of vehicles	Number of explored nodes
2	34912
3	70242
4	Non termination
5	Non termination

Tableau 2 – Direct verification results

6.2. *Compositional verification*

The compositional verification bases on the approach presented in section 5. Table 3 shows the number of nodes explored during the compositional verification of the safety property. As we can see, the number

Number of vehicles	Number of explored nodes
2	34912
3	34273
4	34589
5	34380

Tableau 3 – Compositional verification results

of visited nodes is relatively constant. This is due to the fact that in each one of the verification cases, the model checker explores the same behavioral model that corresponds to only one vehicle. The rest of the system is taken into account through the auxiliary properties.

7. Conclusion

Verification by model checking of platoon models was in most cases limited to platoons with two vehicles. To perform verification

of a complete train some experiences use Theorem-proving methods applied to the global state space. In order to have a manageable state space, abstractions were introduced, but to much abstraction increases distance to reality.

Incorporation of a physicals inspired model, as described in this paper, introduces some details to the verification process and gives more confidence about the result. The goal of this paper was to show that compositional verification can be an efficient approach for the verification of distributed reactive applications. This method of verification is well adapted to non trivial systems such as linear vehicle platoon. It consists in decomposing the system into small components, where an auxiliary property is associated of each of these components. The global property to verify is deduced from the auxiliary properties.
In this paper, the compositional verification of linear platoon was limited in the verification of non collision between two following vehicles during a linear circulation. Verifying the platoon system along an arbitrary trajectory, will be the next task to work on.
Another subject of work on is to determine the role and the nature of the auxiliary property. Till now the auxiliary property is chosen depending on the nature of the system to be verified. Finding a method to prove that a given auxiliary property is a sufficient condition for the validity of safety property, will be an interest subject to work on.
Works exposed in this paper are done with the support of the French ANR (National research agency) through the ANR-VTT SafePlatoon[3] project (ANR-10-VTT-011).

Références

[AL95] M. Abadi and L. Lamport. Conjoining specifications. *ACM Trans. Program. Lang. Syst.*, 17 :507–535, May 1995.

[BBNJ10] S. Bensalem, M. Bozga, T.H. Nguyen, and Sifakis J. Compositional verification for component-based systems and

3. http ://web.utbm.fr/safeplatoon/

application. *Software, IET*, 4(3) :181–193, 2010.

[BDKJT95] F. Brazier, B. Dunin Keplicz, N.R. Jennings, and J. Treur. Formal specification of multi-agent systems : a real-world case, 1995.

[BGL+00] S. Bensalem, V. Ganesh, Y. Lakhnech, C. Muñoz, S. Owre, H. Rueß, J. Rushby, V. Rusu, H. Saïdi, N. Shankar, E. Singerman, and A. Tiwari. An overview of sal. In C. Michael Holloway, editor, *LFM 2000 : Fifth NASA Langley Formal Methods Workshop*, pages 187–196, Hampton, VA, jun 2000. NASA Langley Research Center.

[CGGK09] J.M. Contet, F. Gechter, P. Gruer, and A. Koukam. Bending virtual spring-damper : A solution to improve local platoon control. In *ICCS (1)*, pages 601–610, 2009.

[CGP99] E.M. Clarke, Jr., O. Grumberg, and D. A. Peled. *Model checking*. MIT Press, Cambridge, MA, USA, 1999.

[CGS+07] H.N. Chu, A. Glad, O. Simonin, F. Sempé, A. Drogoul, and F. Charpillet. Swarm approaches for the patrolling problem, information propagation vs. pheromone evaporation. In *ICTAI (1)*, pages 442–449, 2007.

[CIP04] M. Caporuscio, P. Inverardi, and P. Pelliccione. Compositional verification of middleware-based software architecture descriptions. In *Proceedings of the 26th International Conference on Software Engineering*, ICSE '04, pages 221–230, Washington, DC, USA, 2004. IEEE Computer Society.

[CMJT97] F. Cornelissen, C.M. M. Jonker, and J. Treur. Compositional verification of knowledge-based systems : a case study for diagnostic reasoning, 1997.

[DDHP10] S. David, A. Drogoul, S.L. Hickmott, and L. Padgham. An architecture for modular distributed simulation with agent-based models. In *AAMAS*, pages 541–548, 2010.

[DMRS03] L. De Moura, H. Rueß, and M. Sorea. Bounded model checking and induction : From refutation to verification. 2725, 2003.

[DP96] P. Daviet and M. Parent. Longitudinal and lateral servoing of vehicles in a platoon. *IEEE Intelligent Vehicles Symposium, Proceedings*, 1 :41 – 46, 1996.

[EHD04] S. El Hadouaj and A. Drogoul. A study of coordination within a road traffic environment. In *IAT*, pages 491–495, 2004.

[ER01] P.A.M. Ehlert and L.J.M. Rothkrantz. A reactive driving agent for microscopic traffic simulation. *Proc. of the 15th European Simulation Multiconference (ESM2001)*, 2001.

[FD92] J. Ferber and A. Drogoul. Using reactive multi-agent systems in simulation and problem solving. In N. M. Avouris and L. Gasser, editors, *Distributed Artificial Intelligence : Theory and Praxis*, pages 53–80. Kluwer, Dordrecht, 1992.

[Fer99] J. Ferber. *Multi-agent systems - an introduction to distributed artificial intelligence*. Addison-Wesley-Longman, 1999.

[GL91] O. Grumberg and D.E. Long. Model checking and modular verification. *ACM Transactions on Programming Languages and Systems*, 16, 1991.

[IX94] P. Ioannou and Z. Xu. Throttle and brake control systems for automatic vehicle following. *IVHS Journal*, 1(4) :345 –, 1994.

[JJ98] C.M. Jonker and Treur J. Compositional verification of multi-agent systems : a formal analysis of pro-activeness and reactiveness. In *International Journal of Cooperative Information Systems*, pages 51–92. Springer Verlag, 1998.

[MS08] P. Manolios and S.K. Srinivasan. A refinement-based compositional reasoning framework for pipelined machine verification. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 16(4) :353 –364, april 2008.

[MSK09] S. Moujahed, O. Simonin, and A. Koukam. Location problems optimization by a self-organizing multiagent approach. *Multiagent and Grid Systems*, 5(1) :59–74, 2009.

[Pnu85] A. Pnueli. In transition from global to modular temporal reasoning about programs. pages 123–144, 1985.

[SSC08] A. Scheuer, O. Simonin, and F. Charpillet. Safe Longitudinal Platoons of Vehicles without Communication. (RR-6741) :24, 2008.

[WC01] Myung Jin Woo and Jae Weon Choi. A relative navigation system for vehicle platooning. *SICE 2001. Proceedings of the 40th SICE Annual Conference. International Session Papers (IEEE Cat. No.01TH8603)*, pages 28 – 31, 2001.

[WS03] K. Winter and G. Smith. Compositional verification for object-z. pages 280–299, 2003.