An Approach to Compositional Verification of Reactive Multiagent Systems

Jean-Michel CONTET, Pablo GRUER,
Franck GECHTER, Abderrafiaa KOUKAM

Abstract

This paper presents an approach to the verification of reactive multiagent system (RMAS) applications. Many of those applications require high levels of confidence about their safety of execution. In those cases, model-checking appears as an adequate verification tool. However, the complexity of RMAS models generally forbids the direct application of model-checking, due to the combinatory explosion problem. Avoiding this kind of inconvenience is frequently possible by applying methods such as abstraction or composition. This work presents a compositional method adapted to the verification of RMAS models belonging to a wide class of applications. The method is adapted to the verificationas of safety properties. In this paper, the approach is put to practice by considering a vehicle platoon application with linear configuration. The crucial safety property being verified is the absence of collisions between platoon vehicles. A formal specification model is written with the SAL (Symbolic Analysis Laboratory) transition system language and compositional verification is performed with the SAL toolbox, by applying SAL model checkers.

1 Introduction

Methods to tackle a wide range of applications by designing reactive concurrent systems have come to relative maturity. Among those methods, reactive multiagent systems (RMAS) [ZF94] appear as a promising approach: intelligent global behavior emerges as a result of the operation of individual, autonomous agents. We distinguish reactive agents from reactive modules because the former posses complete control autonomy. Consequently, interactions with other agents can limit to perceiving them in their environment, even if other, more explicit interaction mechanisms, such as communication are not formally excluded. Autonomy is in many cases advantageous, relatively to more centralized approaches, requiring communication. Moreover, individual behavior can

be specified on the base of restricted, local perceptions and interactions. Those positive aspects have led to the adoption of RMAS as an approach to the design of a wide range of applications. Among them, we can consider applications of mobile devices such as autonomous vehicles, situated in material environments (towns, airports, factories, ...). However, to be considered as adequate relatively to this application domain, RMAS based approaches have to offer the possibility to reinforce confidence about safety of operation. Indeed, unsafe operation of mobile devices is likely to provoke accidents with human or at least material damage. Verification by model-checking [dMRS03, GL91] appears to be an adequate approach to the satisfaction if this requirement. Model-checkers are well adapted to the verification of safety properties.

Verification is the activity dedicated to prove that a property, expressed by a logical formula, is satisfied by (or valid relatively to) an abstract representation or model of the system. In this paper, we interest in formulas which express a safety property, by expressing that some state formula F is an invariant. A state formula is a logical formula (generally first order) without temporal operators. A formula F is an invariant if F is satisfied by every reachable state of the system. Model-checkers are adequate tools to perform invariance verifications (furthermore, many model-checkers are not restricted to the case where F is a state formula). Model checkers have reached considerable maturity, but are confronted to two limitations:

- If the system model includes unbounded variables, model-checking is confronted to the exploration of an infinite state space. In that case, model-checking comes to termination only if a refutation or counter-example is found.
- Even in the case of finite state spaces there is the risk of a so called state space explosion. Systems composed of a huge number of components, each one of which could in turn be quite complex, 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.

New families of model-checkers have been proposed recently, which can perform bounded depth explorations of non-finite state spaces, in search for counter-examples. If the search does not succeed, an additional proof step can be added to extend the validation of the verified property to unbounded system evolutions. Frequently, this additional proof bases on some kind of induction [dMRS03] but of course, this induction is not guaranteed to succeed. Nevertheless, bounded model checkers can also be affected by the state-space explosion problem.

To overcome the state space explosion problem, compositional verification approaches [WS03, HLFS04, YZ09, MS08, ZAX08] have been proposed. Compo-

sitional verification consists in verifying separately a set of auxiliary properties relative to system components. Then, a compositional verification rule applies to auxiliary properties in order to establish the global property.

This work presents a compositional verification approach adapted to a wide class of RMAS applications. This class can be characterized by a high level of autonomy of the component agents, and by local perception and interaction patterns. Those particular patterns 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 later 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 the agent model.

To illustrate the verification approach, the vehicle platoon problem is presented, with requirements related to urban passengers transportation [CGGK10]. The adopted platoon geometry is linear: virtual trains of cars, without material coupling. System design bases on a decentralized and local control approach: each car is an agent which autonomously produces its own behavior when integrated within the train. A car interacts with the environment (including other cars) by means of its perception capabilities and its behavior is generated exclusively from those local perceptions.

The agent design approach adopted in this work characterizes by the following features: vehicle agent's behavior is specified on the basis of a physics inspired interaction mechanism. The inputs of this mechanism are vehicle's perceptions and the outputs are longitudinal and lateral control references, used as inputs to regulation and command stages. Those models naturally include dense, unbounded variables (real-number valued). The specification models are built using the SAL transition system language [BGL$^+$00]. Verification is performed to validate a condition for safe operation and passengers integrity, i. e., the impossibility of inter-vehicular collision during train operation. Verification by model-checking is performed with the SAL toolbox, which includes a bounded model-checker. SAL model-checkers facilitate the application of the compositional verification method proposed here, by offering an option consisting in the injection of an auxiliary lemma, an important aspect of our method. This work is organized as follows: next section presents in a general fashion the main concepts of the iterative compositional verification approach. Then, the linear platoon system is presented, the physics inspired interaction model is introduced and the SAL specification models of vehicle's behavior are described. The concrete application of our compositional verification approach in the frame of the SAL environment comes next, together with a comparative experience of direct versus compositional verification. Concluding remarks and comments on future work close this presentation.

2 A compositional verification method

In this work we interest in the verification of the invariance of a state formula F. A way to prove invariance is to prove the validity of formula $\Box F$, relatively to the system model M. Symbol $\Box$ represents a linear temporal operator. Formula $\Box F$ is satisfied by a causal and diligent sequence of states of M starting by an initial state, if F is satisfied by every state of the sequence. Formula $\Box F$ is valid relatively to M if it is satisfied by every causal and diligent sequence of states of M starting by an initial state.
We will note $M \vDash F_1, \cdots, F_n$ when properties F_i are valid relatively to system model M. Properties are evaluated over infinite sequences σ of states, representing causal and diligent evolutions of a system model M, which start at an initial state of M. Let us note $\mathcal{B}(M)$ the set of all sequences that M can produce, and $Sat(F)$ the set of sequences which satisfy property F. Then, $M \vDash F$ if and only if $\mathcal{B}(M) \subseteq Sat(P)$. We note $M, F_1 \vDash F_2$ if property F_2 is satisfied by model M under the assumption F_1. In this frame, the compositional verification method bases on the application of the following deduction rule:

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

Where C_i and C_{i+1} are system components, M_i is the system composed of i components, S_i is the safety property expressed relatively to a system with i components and T_i an auxiliary property about a system with i components. The integration of C_{i+1} to system M_i yields a new system noted $C_{i+1} \| M_i$. Auxiliary property T_i somehow represents the instance M_i of the system. The first premise of R_c indicates that T_i should be valid for C_i, under the assumption T_{i-1}. The second premise says that if T_i is valid, when component C_{i+1} is added, the safety property S_{i+1} will be valid.
It can be proved that rule R_c is sound under the assumption that adding component C_{i+1} does not modify the behavior of M_i. It is a strong condition indeed, but as we will see, it is satisfied by the non-trivial RMAS case study presented here. Furthermore, we consider the composition operator $\|$ as an asynchronous operator: the new behavior results from the interleaving of component transitions, together with synchronized transitions, if any. In our case, as component agents only interact by perceptions, there are not synchronized transitions.

Based on the previous deduction rule, the compositional verification approach applies to instances M_i of the RMAS model. Instance M_i results from M_{i-1} by addition of a new component agent A_i (noted $M_i = A_i \| M_{i-1}$). As already stated, an important assumption is that A_i and M_{i-1} interact only by means of local perception mechanisms, without any explicit agent-to-agent communication or synchronization. As a matter of fact, this assumption is part of a sufficient condition to the satisfaction of soundness, when applying rule R_c.
The verification process starts with a single component instance $M_0 = A_0$ of

the model. Properties to be verified at every verification step are:

- the specific safety property being analyzed, noted S_i, expressing a relationship between component agent A_i and system instance M_{i-1}.
- an auxiliary property T_i, relative to system M_i.

The basic idea embodied by the compositional verification method, which presents an iterative aspect, is to simplify the verification of S_i by replacing M_{i-1} by auxiliary property T_{i-1}, introduced as an assumption. The iterative process stops when the safety property is refuted for some instance M_i of the model, or verified for instance M_k, with k being a predefined number of agents. Another way of doing is to repeat the verification step up to the number k of agents for which verification fails. The verification process can be described as follows:

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

Even though the preceding formulation has an inductive flavor, it should not be consider to be an induction process. We do not try to establish the safety property for arbitrary values of i. As a matter of fact, we expect that for some value of i, the new auxiliary property T_{i+1} fails, and therefore, iterations stop. Formulation of the auxiliary properties T_i depends on the problem characteristics. In general it can be said that T_i somehow somehow represents how the rest of the system conditions the satisfaction of S_{i+1}, eventually in an indirect fashion. Additionally, a set of parameters could eventually intervene in the formulation of property T_i. Giving values to those parameters is another important aspect of the approach. Both the formulation of T_i and the valuation of its parameters are made easier by the specification approach applied in this work. Particularly, the physical inspiration of the interaction model, presented in the next section, gives useful indications.

3 The application

3.1 General description

The application presented in this work is a vehicle platoon system with linear platoon configurations, i.e., virtual trains of vehicles without material coupling. These are a promising approach to new transportation systems [HTV94], with innovative capabilities. Platoon systems, when applied to civil vehicles, have been mainly studied as a way to increase track density in highways. More

recently, linear platoons have been studied as the basic technology to implement new passenger transportation services in urban environments, with a high adaptability to user needs and safety improvement thanks to automated or semi-automated driving assistance (obstacle detection and avoidance, automatic car parking,...). A basic problem in platoon systems is the control of the vectorial inter-vehicle distance. Some of the more followed approaches are based on automatic control. In this frame, the control of global platoon geometry has been decomposed in different sub-problems: longitudinal control (distance regulation), lateral control (angle regulation), integrated lateral and longitudinal control and merge/split capabilities. Most of the lateral or longitudinal control proposals base on PID (Proportional, Integral, Derivative) control [IX94, MH90, DP96]. Integrated longitudinal and lateral control has also attracted research as in [WC01]. A reactive, autonomous longitudinal and lateral control approach, intended for urban area transportation, with stringent conditions: smaller curve radius and more constrained merge and split operations has also been developed [CGGK07, CGGK06]. Within the approach presented in this work, vehicle's behavior and interactions are specified from a physics inspired model, as described next. For the sake of a simpler presentation, we abstract from lateral control and focus the presentation on rectilinear train's displacement.

3.2 The Physics inspired interaction model

The platoon multiagent system presented in this paper is composed of a set of agents each one corresponding to a vehicle. The behavior of every vehicle agent depends only on its own perceptions. Two main vehicle behaviors have been defined: The header vehicle behavior and the follower vehicle behavior. The header vehicle behavior results from perception of the road. The intended goal of header agent is to follow a predefined routing and its perception is based basically on artificial vision. The follower vehicle behavior results exclusivelly from its perception of the preceding vehicle in the platoon. These interactions are mostly based on distance-measuring devices: stereo-vision and/or laser range finder.

In this paper we concentrate on the description of the follower vehicle behavior. The main use of the physics-inspired interaction model of follower vehicle agents is to specify the reactive behavior of one agent, as a result of agent's perceptions. The following section describes the interaction model adopted in this study.

3.3 Vehicle's interaction model

A train is composed of n vehicles $V_0, \cdots, V_{n-1}$. The first one, V_0 is assigned the functions of navigation. Vehicle V_i $(i > 0)$ measures the distance vector

to vehicle V_{i-1} (the preceding one) and calculates an interaction force based on the mechanical laws of a virtual spring damper place between V_i and V_{i-1}. The interaction forces intervene in the calculation of an acceleration vector to be applied to the vehicle. The virtual spring damper model bases on stiffness k, damping factor h and spring's un stretched length l_0. The forces involved are : spring force $\vec{F}_s$ and the damping force $\vec{F}_d$. Each vehicle V_i is represented by its position $\vec{X}_i = [x_i, y_i]$. The mass of the vehicle is denoted by m (we assume that each vehicle has the same mass). The distance between vehicles is $d = \|\vec{X}_{n+1} - \vec{X}_n\|$ (Cf. figure 1). Newton law of motion allows to calculate the vehicle acceleration in the preceding vehicle's reference:

Spring force	$\vec{F}_s = -k(\|\vec{X}_{n+1} - \vec{X}_n\| - l_0)\vec{u_{n+1\ n}}$
Damping force	$\vec{F}_d = -h(\dot{\vec{X}}_{n+1} - \dot{\vec{X}}_n)$

$$m * \vec{\gamma} = -k(\|\vec{X}_{n+1} - \vec{X}_n\| - l_0)\vec{u_{n+1\ n}} - h(\dot{\vec{X}}_{n+1} - \dot{\vec{X}}_n) \quad (1)$$

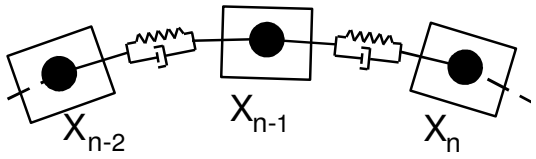

Figure 1: Simplification of forces applied to the vehicle X_{n-1}

By discrete integration, speed and vehicle's state (position and orientation) can then be determined and the command law can be computed. In this case, it consists on vehicle's direction and speed. The choice of a command law and the law's parameter values takes into account the characteristics of the test vehicle used in our laboratory to perform experimentations. Other constraints derive from passenger transportation regulations.

Formula 1 is the base for the definition of agent's control functions. Those functions will calculate, at each agent's operation cycle, a new acceleration command reference to be sent to the vehicle's controller. It should be noted that the interaction model presented here is a simplified version. In practice, the interaction model also takes into account the lateral deviation, related to the angle formed by speed vectors of a vehicle and of its predecessor.

4 Compositional verification

4.1 The verification framework

The SAL toolkit[1] is used as verification framework. First of all, a behavioral model of the verified system has to be defined by using the SAL transition systems language. A SAL model is encapsulated by 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 from and M_2.
Among the model checkers include in SAL, the bounded model checker (BMC) has been chosen, due to the presence of real-valued variables in the specification model. This is a model checker for infinite state systems based on SAT solving. The exploration depth is bounded as a means to avoid non terminating search. Consequently the SAL bounded model checker performs verification by refutation i.e., search for a counterexample within the exploration bound. When a counter-example is not found, the model checker is able, in some particular cases, to extend the result to general validity, by applying an induction step. Otherwise, if the model is infinite, non-termination can occur [dMRS03, Pik07]. The SAL BMC was applied because the models presented here naturally include real variables.

In this paper, we present a simpler presentation of the formal model. We focus the presentation on the vehicle control represent by the vehicle's interaction model.

4.2 The *Vehicle* context

The system presented in this work is a RMAS composed of vehicle agents. Modeling begins by building the behavioral model of an agent, encapsulated in the SAL *Vehicle* context, including three basic modules. Each one of those modules models the behavior of a component of the *Vehicle* agent. The *PhysicalModel* module models physical characteristic of the vehicle, the *VehicleControl* module calculates new acceleration references from vehicle's perceptions (thaks to the Vehicle's interaction model presented before), as produced by the *Perception* module. The *VehicleBehavior* module specifies the asynchronous composition of those component modules modules. Asynchronous composition was chosen

[1] `http://sal.csl.sri.com/index.shtml`

because only one component is active at any instant, so that transition interleaving perfectly describes system evolution.

```
Vehicle{DeltaSpeedMin : REAL, DeltaSpeedMax : REAL}: CONTEXT =
BEGIN
Constants definition
Types definition
Variables declaration
Functions definition
PhysicalModel: MODULE = ···
VehicleControl: MODULE = ···
Perception: MODULE = ···
vehicleBehavior: MODULE = PhysicalModel [ ] VehicleControl [ ] Perception;
END;
```

A context includes a series of sections, among which we can consider the following:

- Constants definition: this section associates values to constant names used in all calculations:

```
%% Constants definition
safetyDistance: REAL = 0.3;
minimumAcceleration: REAL = -3;
maximumAcceleration: REAL = 1.3;
masse: REAL = 500;
.....
```

- Types definition: the SAL transition-system language includes a rich set of type constructors. In this work, three enumerated types are defined, to represent the internal state of each one of the component modules:

```
%% types definition
PerceptionState: TYPE = {Pidle,Pwait,Read};
VehicleControlState: TYPE = {Ridle,Rwait,Rcompute,Stop};
PhysicalModelState: TYPE = {Pwait,PCompute,Speedlimits};
```

- Functions definition: associates a function name to an expression. Here the *computeNewAcceleration* function compute the new value of acceleration thanks to the vehicle's interaction model :

```
%% Functions definition
computeNewAcceleration(_distance: REAL): REAL = (k*(_distance-regularDistance)
- (h*_distance)/delay)/masse;
....
```

- Modules definition: each component of the vehicle agent yields a module. As an illustration, consider the *VehicleControl* module below. Variable *vehState* keeps track of the internal state of the module. Component modules synchronize and communicate by means of shared variables, which are

translated to either *INPUT*, *OUTPUT* or *GLOBAL* module variables. Boolean variables are treated as events, used to synchronize the behaviors of the three components. State transformations produced by transitions are described using the well known "after" (primed variables), "before" (unprimed variables) expressions.

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

4.3 Compositional verification with SAL

As already stated, the safety property to be verified is non collision (i.e inter-vehicle distance greater than a predefined safety distance). The safety condition is expressed by the following SAL statement:

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

The *SafetyCondition* theorem has been expressed using the temporal operator G that expresses invariance : $G(P)$ means that P is satisfied by every state. Invariance is an alternative way to express validity of P, relatively to system evolutions.

The compositional rule R_c can be applied under the assumption that adding component C_{i+1} does not modify the behavior of M_i. The system presented

in this study satisfies the assumption since adding vehicle V_{i+1} to the train does not change the behavior of the preceding vehicle V_i and other preceding vehicles. This property is due to the forward local platoon control strategy adopted in this application: any vehicle agent only perceives the distace to the preceding vehicle, and does not perceive any one of the following vehicles. The compositional iterative verification method is based on the use of auxiliary property T_i and T_{i+1}. In this case study, the auxiliary properties expresses a condition relative to the speed increments of vehicle V_i. The auxiliary properties are expressed, by writing the following theorem statements:

```
AuxiliaryPropertyPre: THEOREM Vehicle |- G (deltaSpeed >= DeltaSpeedMinPre
AND deltaSpeed <= DeltaSpeedMaxPre);
AuxiliaryPropertyPost: THEOREM Vehicle |- G (deltaSpeed >= DeltaSpeedMinPost
AND deltaSpeed <= DeltaSpeedMaxPost);
```

where *DeltaSpeedMinPre*, *DeltaSpeedMaxPre*, *DeltaSpeedMinPost* and *DeltaSpeedMaxPost* have been declared as constants. The verification of the premise $C_{i+1}, T_i \vDash S_{i+1}$ of deduction rule R_c, is performed by the following SAL command:

```
sal-inf-bmc -depth=50 -induction -l AuxiliaryPropertyPre Vehicle SafetyCondition
```

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

The first step consists in verifying the auxiliary property T_0 which correspond to the *AuxiliaryPropertyPre* with *DeltaSpeedMin* = -0.1 m/s^2 and *DeltaSpeedMax* = 0.13 m/s^2. These values have been deduced from the characteristics of laboratory's experimental vehicle. In this case, the lemma is verified because the parameter values of vehicle model are taken from the real vehicle. Then, the iterative verification process can be started using the SAL verification command line:

```
sal-inf-bmc -depth=50 -induction -l AuxiliaryPropertyPre Vehicle AuxiliaryProperty-
Post
```

which establishes the validity of the auxiliary property to be used next. If this run of the model-checker succeeds, the value of constants *DeltaSpeedMinPost* and *DeltaSpeedMaxPost* is transferred to constants *DeltaSpeedMinPre* and *DeltaSpeedMaxPre*. The sal-inf-bmc command launches the bounded model checker. State space exploration is bounded (–depth=50) to avoid non terminating search. The $-$*induction* option triggers an induction step if no counter-example is found. The verification has been performed for up to five vehicles and the validity of the safety condition has been verified by the model checker. It

should be noted that during the verification process, only the behavioral model of a vehicle agent was used. There was no need for an explicit model of the complete system, thereby minimizing the risk of combinatorial explosion.

4.4 Comparative study of verification approaches

The impact of compositional verification was estimated by making a comparison with non-compositional verification, applied to a global model resulting from the composition of n vehicle agents, obtained by indexing the model of an agent, based on the module array construct of the SAL transition system language. The global behavior of an RMAS of n vehicles can be modeled by:

```
Platoon: CONTEXT =
BEGIN
PlatoonBehavior: MODULE = Vehicle(1) [ ] ... [ ] Vehicle(n);
END;
```

Regarding the non-compositional verification, we note that from four vehicles on, model-checking does not seem to be able to come to termination (in this experience, the model-checker has run during five full days without stopping and giving an answer).

Number of vehicles	Number of nodes explored direct verification	Number of nodes explored compositional verification
2	34912	34912
3	70242	34273
4	non termination	34589
5	non termination	34380

concerning compositional verification, we can see that the number of nodes visited is relatively constant. This is explained by the fact that every case of verification, model checker explores the same behavioral model, since the rest of the system is taken into account by the auxiliary property.

5 Conclusion

Most of the verification experiences in similar platoon projects limit to model-checking a system composed of two vehicles abstracted from the complete train

of vehicles. Some other experiences have performed verification of complete vehicle trains, by theorem-proving methods applied to the the global state space [SSC08]. In order to have a tractable state space, a high degree of abstraction has been introduced. But too much abstraction increases the distance to reality and consequently reduces confidence in the results. Alternatively, the incorporation of a physical model, as presented in this work, is a way to introduce more realistic details, into the verification process, thereby allowing greater confidence in results of verification. Nevertheless, this approach has a cost in terms of state space complexity and may require more sophisticated verification methods, such as compositional verification. The goal of this paper was to show that compositional verification can be a viable verification method, adapted to distributed reactive applications. The approach presented here is well adapted to non trivial systems such as linear vehicle platoons. The proposed compositional iterative verification allows to prove the platoon safety property to several scenario of vehicle train composition. The consistency condition for the application of the verification rule appears to be quite strong. How to weaken the consistency condition is an interesting work to do, because, as a consequence, the size of the set of applications to which the compositional verification method can be applied could increase. Another subject for reflection is the role and the nature of the auxiliary property. In this work, we have chosen a property about possible positive or negative increments of speed for the vehicle at the end of the train. It seems to be intuitively acceptable that this property plays an essential role in the satisfaction of the non collision property. A work to be done is to try to establish some way of doing to be able to show that a given auxiliary property is a sufficient condition for the validity of safety property.

References

[BGL+00] Saddek Bensalem, Vijay Ganesh, Yassine Lakhnech, César Mu noz, Sam Owre, Harald Rueß, John Rushby, Vlad Rusu, Hassen Saïdi, N. Shankar, Eli Singerman, and Ashish 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.

[CGGK06] Jean-Michel Contet, Franck Gechter, Pablo Gruer, and Abder Koukam. Multiagent system model for vehicle platooning with merge and split capabilities. *Third International Conference on Autonomous Robots and Agents*, pages 41–46, 2006.

[CGGK07] Jean-Michel Contet, Franck Gechter, Pablo Gruer, and Abder Koukam. Physics inspired multiagent model for vehicle platooning. *International Conference on Autonomous Agents and Multiagent Systems AAMAS*, 2007.

[CGGK10] J.M. Contet, F. Gechter, P. Gruer, and A Koukam. Car-driving assistance using organization measurement of reactive multi-agent system. *International Conference on Computational Science 2010 (ICCS 2010)*, 2010.

[dMRS03] Leonardo de Moura, Harald Rueß, and Maria Sorea. Bounded model checking and induction: From refutation to verification. 2725, 2003.

[DP96] Pascal Daviet and Michel Parent. Longitudinal and lateral servoing of vehicles in a platoon. *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 41 – 46, 1996. Automatic driving;Platooning techniques;.

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

[HLFS04] P.-A. Hsiung, T.-Y. Lee, J.-M. Fu, and W.-B. See. Formal verification of real-time embedded software in an object-oriented application framework. *Computers and Digital Techniques, IEE Proceedings* -, 151(6):417 – 434, nov. 2004.

[HTV94] J.K. Hedrick, M. Tomizuka, and P. Varaiya. Control issues in automated highway systems. *IEEE Control Systems Magazine*, 14(6):21 – 32, 1994.

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

[MH90] John J. Moskwa and J. Karl Hedrick. Nonlinear algorithms for automotive engine control. *IEEE Control Systems Magazine*, 10(3):88 – 93, 1990.

[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.

[Pik07] Lee Pike. Model checking for the practical verificationist: a user's perspective on SAL. 2007.

[SSC08] Alexis Scheuer, Olivier Simonin, and François 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] Kirsten Winter and Graeme Smith. Compositional verification for object-z. pages 280–299, 2003.

[YZ09] Haiqiong Yao and Hao Zheng. Automated interface refinement for compositional verification. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(3):433 –446, march 2009.

[ZAX08] Hao Zheng, J. Ahrens, and Tian Xia. A compositional method with failure-preserving abstraction for asynchronous design verification. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(7):1343 –1347, july 2008.

[ZF94] K. Zeghal and J. Ferber. A reactive approach for distributed air traffic control. *proceedings of Avignon94*, pages 381–390, 1994.