Towards a compositional verification approach for multi-agent systems: Application to Platoon system

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

Laboratoire Systèmes et Transports (SeT)
{firstname.lastname}@utbm.fr

Résumé. This paper presents a methodology for the verification of reactive multi-agent systems (RMAS). High level of confidence about a safety execution is a must in such systems. For this reason, model-checking appear as an adequate tool to the verification of such models. However, model-checking can be confronted with the problem of huge state space exploration. To avoid this kind of inconvenience, it is possible to apply methods like 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 some safety property. The application considered in this paper is a platoon with linear configuration. The safety property to be verified is the non-collision between platoon vehicles. The SAL toolkit has been adopted as a verification tool, by applying SAL model checkers.

Mots-Clés : Multi-agent systems, Model-checking, Compositional verification, Platoon application

1. Introduction

Reactive multi-agent systems (RMAS) [CGS+07] appears to be a promising approach for the designing of many reactive concurrent system. Such systems rely on reactive and autonomous agents, that behave
based on their own perceptions [Fer99]. In these systems, global intelligent behavior emerges as a result of individual autonomous behavior of each agent. Each agent behavior can be specified on the bases of its perceptions and interactions. RMAS shows some features like self-organization, robustness, simplicity, agents redundancy and also low cost agent design. Consequently, reactive multi-agent systems have been adopted in the design of many complex applications, as in simulation [DDHP10], [ER01], problem solving [FD92], positioning problems [MSK09] and road traffic problems [EHAD04].

However, to be considered as adequate, RMAS must be able to verify some safety conditions to be considered in each application.

Verification is on the way to become a very frequent stage in the design process for reactive applications. Verification consists in formally establishing the logical validity of some 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, by expressing that some property $F$ is invariant. $F$ is invariant if $F$ is satisfied by each reachable state of the model $M$. Two families of verification algorithms have produced efficient tools: model-checking [DMRS03] and theorem-proving [SSC08]. Model-checking are adequate tools to perform invariance verifications. On the other hand, model-checking suffers from two limitations. First, the non termination problem: model-checking is confronted to the exploration of infinite state if the model includes unbounded variables. To address this problem, new families of model checkers perform bounded depth exploration. If no counter-example is found an additional proof step based on induction can be added [DMRS03]. Of course, this induction is not guaranteed to succeed, but in many cases it succeeds in an infinite state-space exploration.

Second, even in case 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 rule to these auxiliary property in order to establish the global property.

This paper presents an approach to the compositional verification adopted in a large class of RMAS application. This applications are characterized by a high level of autonomy of the agents, and by a local perception and interaction 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.

The application we adopt to illustrate this verification approach is the linear Platoon system. Linear Platoons are a sets of vehicles that circulate while keeping a train configuration without any material coupling. System design bases on a decentralized, local approach. Each car represents an autonomous agent that behaves based only on its own perceptions capabilities. Each agent is characterized by a set of parameter.

The adopted approach 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. Those models include naturally unbounded real variable. Verification and specification is performed using SAL toolkit [BGL+00]. The safety condition to be verified is the non-collision between vehicle during platoon operation. SAL toolkits facilitate the application of the compositional verification approach by offering the possibility of introducing an auxiliary lemma. This paper is organized as follow: Section 2 gives a general definition of the compositional verification method. In section 3, a concrete application is described: Linear platoon system and the inspired physical model. Section 4 presents the SAL specification models of vehicle behavior and the application of the compositional verification. Finally in the conclusion some remarks and future work close to this presentation are mentioned.
2. A compositional verification method

In this paper, we are interested in the verification of the validity of a formula $F$, relatively to a behavioral model $M$. Relative validity means that every possible evolution of the system as determined by $M$ satisfies $F$.

The compositional rule is applied on the instance $M_i$ of the system model $M$. The instance $M_i$ results from $M_{i-1}$ by adding a new component $C_i$ (noted $M_i = C_i \parallel M_{i-1}$). In our case, an agent interacts only by perceptions, so there is no synchronized operations. For this reason $\parallel$ is considered as an asynchronous composition operator.

The verification process begins with the instance $M_0 = C_0$. Then at each step of verification two properties have to be verified:

- Safety property, noted $S_i$, it is the relation between the instance of the system $M_{i-1}$ and the new component $C_i$.
- An auxiliary property $T_i$, relative to the system $M_i$.

In the Compositional verification method, the main idea is to simplify the verification of $S_i$ by replacing $M_{i-1}$ with the auxiliary property $T_{i-1}$, introduced as an hypothesis. The process stops when arrived to an instance $M_k$ that does not satisfy the auxiliary property $T_k$.

Given the properties $F_1, \ldots, F_n$, we will note $M \vDash F_1, \ldots, F_n$ when all the properties $F_i$ are valid relatively to the system model. These properties are evaluated over an infinite sequence of state $\sigma$. We will note $B(M)$ the set of sequences produced by $M$ and $\text{Sat}(F)$ the set of sequences that satisfy the property $F$. Then, $M \vDash F$ if and only if $B(M) \subseteq \text{Sat}(F)$. We note $M, F_1 \vDash F_2$ if property $F_2$ is satisfied by model $M$ under the hypothesis $F_1$. In this frame following rule is applied for the compositional verification:

\[
R_c : \frac{C_i, T_{i-1} \vDash T_i}{C_{i+1}, T_i \vDash S_{i+1}}
\]

\[
C_{i+1} \parallel M_i \vDash S_{i+1}
\]
On the basis of this rule, the verification process can be described as follow:

\[
\begin{align*}
\text{verify } & C_0 \models T_0 \text{ and } C_1, T_0 \models S_1 \\
\text{for } i > 1 \text{ while } & C_i, T_{i-1} \models T_i \text{ apply } R_c \text{ to } C_{i+1}, T_i
\end{align*}
\]

Formulation of the auxiliary property depends on the problem characteristics. Figure 1, shows the application of the compositional verification rule.

Figure 1 – Application of the compositional verification method

3. The application

3.1. General description

Multi-agent system application presented here is a vehicle platoon system. A platoon is a set of vehicles that circulates while keeping a
specific configuration without any physical coupling. We are interested in linear platoons: platoon that has a train configuration.

The platoon control problem can be decomposed in two platoon system sub-problems: longitudinal control (distance regulation), lateral control (angle regulation). Most of lateral or longitudinal control approaches are based on the PID control (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.

3.2. Physics inspired interaction model

The platoon 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. (1) Leader behavior: Concerns the leader vehicle, this agent percept only the road or follows a predefined trajectory and its perceptions are based on artificial vision. (2) Follower behavior: Concerns follower vehicles, this agent percepts only the preceding vehicle in the platoon. These interaction are mostly based on distance measuring-devices: stereo vision, laser range finder...

In this paper, we are interested in the follower vehicle behavior. The main use of physics inspired interaction model of follower agent is to specify the reactive behavior of the agent based on the agent’s perceptions. The interaction model is described in the following section.

3.3. Vehicle’s interaction model

Vehicle agents are inter-connected by means a virtual physics-inspired interaction model. First vehicle is driven by a human driver or by software. Each one of the follower vehicles follows its predecessor’s trajectory, and calculates the inter-vehicle vector distance, and deduces an interaction force, using the virtual physical interaction model. This interaction model is composed of two springs and a damper connecting
two following vehicles like shows figure 2. This virtual coupling influences only on the following vehicle, there is no mutual interaction as in regular physics. Model parameters are, \( k_1 \) stiffness of the first spring, \( k_2 \) stiffness of the second spring, \( h \) damping factor, \( L_v \) spring resting length and \( m \) the vehicle mass.

![Physical interaction model](image)

**Figure 2 – Physical interaction model**

Each follower agent measures the distances at three key points:

- \( D \) : Length of the damper.
- \( d_1 \) : Length of the first spring.
- \( d_2 \) : Length of the second spring.

Note that in linear cases, \( d_1 = d_2 = D \). In turning cases, \( d_1 = D + \frac{DL}{2r} \) and \( d_2 = D - \frac{DL}{2r} \) (or vice versa depending on whether it is a left or right turning case). Where \( L \) is the vehicle width and \( r \) is the curve radius.

Forces involved in this system are \( \vec{F}_1 \) force of the first spring, \( \vec{F}_2 \) force of the second spring and the damping force \( \vec{F}_h \).

To calculate the vehicle acceleration in the preceding vehicle’s reference 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** : \( \vec{F}_h = h(||\Delta \vec{D}||)\vec{u} \)

Where \( \vec{u} = \vec{i}_j / ||i_j|| \) (cf. figure 2).

\[
m \ast \gamma = (h \frac{\Delta D}{\Delta t} + k_1(d_1 - l_v) + k_2(d_2 - l_v))\vec{u} \quad (1)
\]
In linear circulation this equation could be written like:

\[ m \gamma = (h \frac{\Delta D}{\Delta t} + k(D - l_v)) \vec{u} \]  

(2)

In turning cases it could be written like:

\[ m \gamma = (h \frac{\Delta D}{\Delta t} + k(D - l_v) + \frac{DL}{2r} (k_1 - k_2)) \vec{u} \]  

(3)

By discrete integration vehicle’s state (speed, orientation and position) can then be determined and the command law can be computed. Equation 1 is the base for the definition of agent’s behavior. This equation will calculates at each agent’s operation cycle, a new acceleration command reference to be sent to vehicle’s controller.

4. Compositional verification

What is presented now is a verification case-study of a safety property in the platoon MAS: Non-collision of a follower vehicle with its leader during the platoon circulation with a constant speed. SAL 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. 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, 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.

4.1. 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 $M_1 \parallel M_2$ is a module that takes $M_1$ and $M_2$ transitions in a lock step, or asynchronously, where $M_1 \mathbin{\llbracket} M_2$ is a module that takes an interleaving of transitions from $M_1$ from and $M_2$. In our case, a Vehicle context represents the behavioral model of the follower vehicle agent and includes three modules: Perception, Vehicle Control and Physical Model. Each one of these modules models a component of the agent behavior. 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 behavior of the Vehicle context results from the asynchronous composition of the three modules. 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.

Here is a sample code snippet for the Vehicle context:

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

Some detail is given now about model elements:

- **Constante definition**: This section assigns a value to each physical constant.

```
safetyDistance : REAL = 0.3 ;
minAcceleration : REAL = -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.

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

– **Function definition**: This section defines the different functions used for modules’ calculations. For example, the "computeSpeed" function calculates the speed of the vehicle, from its acceleration during time period.

```plaintext
computeAcceleration(_distance : REAL) : REAL = \frac{\left(k*(distance-regularDistance) - \left(h*_distance)/delay/masse \right)}{masse};
```

– **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 `VehicleControl` module is describe:
VehicleControl : MODULE =
BEGIN
  LOCAL vehicleState : VehicleControlState
  GLOBAL start, endAction : BOOLEAN
  GLOBAL endPercept, endPhysicalModel : BOOLEAN
  INPUT measuredDistance : REAL
  OUTPUT acceleration : REAL
  INITIALIZATION acceleration = 0; vehState = Idle
  TRANSITION
    | vehicleState = Idle \rightarrow
    | start' = TRUE; vehicleState' = Wait
    | vehicleState = Wait AND endPercept AND
    | measuredDistance < safetyDistance \rightarrow
    | acceleration' = maxDeceleration; vehicleState' = Act
    | endPercept' = FALSE
    | vehicleState = Wait AND endPercept AND
    | measuredDistance >= safetyDistance \rightarrow
    | acceleration' = computeAcceleration ?(measuredDistance); vehicleState' = Act
    | endPercept' = FALSE
    | vehicleState = Act \rightarrow
    | endAction' = TRUE; vehicleState' = Wait
END;

4.2. Compositional verification with SAL : Linear circulation

As mentioned before, the case considered here is linear platoon in a longitudinal circulation (along a line) where $d_1 = d_2 = D$ (cf. figure 2), and the safety condition to be verified is the non-collision between two successive vehicles (i.e. inter-vehicle distance must be greater than a predefined safety distance). Following SAL statement express this safety condition.

SafetyCondition : THEOREM Vehicle |\neg G(measuredDistance >= safetyDistance);

Where $G$ is a temporal operator that express 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.

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. Platoon system satisfy this assumption, since adding a new
vehicle to the train does not modify the behavior of the other prece-
ding 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 property $T_i$ and $T_{i+1}$. In this case study, the auxiliary pro-
erties expresses a condition relative to the speed increments of vehicle $V_i$.
The auxiliary properties are expressed, by writing the following theo-
rem statements:

CurrentAuxiliaryProperty : THEOREM Vehicle $\neg G \left( \text{intervalSpeed} = \text{IntervalSpeedMinPre} \right. \left. \text{AND intervalSpeed} < \text{IntervalSpeedMaxPre} \right) ;$

NextAuxiliaryProperty : THEOREM Vehicle $\neg G \left( \text{intervalSpeed} = \text{IntervalSpeedMinPost} \right. \left. \text{AND intervalSpeed} < \text{IntervalSpeedMaxPost} \right) ;$

where $\text{IntervalSpeedMinPre}$, $\text{IntervalSpeedMaxPre}$, $\text{IntervalSpeedMinPost}$ and $\text{IntervalSpeedMaxPost}$ have been de-
clared as constants. The verification of the premise $C_{i+1}, T_i \models S_{i+1}$ of
the compositional verification rule $R_c$, is performed by the following
SAL command:

```
# CurrentAuxiliaryProperty : THEOREM Vehicle $\neg G \left( \text{intervalSpeed} > \text{IntervalSpeedMinPre} \right. \left. \text{AND intervalSpeed} < \text{IntervalSpeedMaxPre} \right) ;

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

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

The first step consists in verifying the auxiliary property $T_0$ which
correspond to the CurrentAuxiliaryProperty. Then, the iterative ve-
ification process can be started using the SAL verification command line :

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

which establishes the validity of the auxiliary property to be
used next. If this execution of the model-checker succeeded, $\text{IntervalSpeedMinPost}$ and $\text{IntervalSpeedMaxPost}$ became the
constants $\text{IntervalSpeedMinPre}$ and $\text{IntervalSpeedMaxPre}$. sal-
inf-bmc launches the bounded model checker. Space exploration is
bounded (depth=50) to avoid non terminating search. If no counter
example is found, an induction step is added using the induction com-
mand. The validity of the safety condition have been verified up to pla-
toon with 5 vehicles.

Figure 3 shows the procedure of the compositional verification process.
5. Conclusion

Verification by model checking of platoon models was in most cases limited to platoon with two vehicles. To perform verification to a complete train some experiences use theorem-proving methods applied to the global state space. In order to have a manageable state space, abstractions was introduced sometimes, but to much abstraction increases distance to reality. Incorporation of physical inspired model, as described in this paper, introduces some reality 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 way for the verification of distributed reactive applications. This method of verification is well adapted to non trivial systems such as linear vehicle platoon in a linear circulation. We are now working on using this verification method in case of non linear circulation of the platoon. The proposed iterative compositional method allows to prove the safety property in several scenarios of vehicle train composition. Another subject to work on
is the role and the nature of auxiliary property. In this work we choose the property about increasing and decreasing of the value of the vehicle speed at the end of the train. This property seems to be acceptable in this platoon model since speed variation have an essential role in the satisfaction of the safety property, the non collision between vehicles. A work to be done is to try to demonstrate that a given auxiliary property is a sufficient condition for the validity of safety property.

Références


