An Experience with a Formal Modelling of a Multi-Agent System: the Platooning Problem

Samuel Colin, Arnaud Lanoix

An Experience with a Formal Modelling of a Multi-Agent System: the Platooning Problem ⋆

Samuel Colin[1] Arnaud Lanoix[2]

LORIA – DEDALE Team
Campus scientifique
F-54506 Vandoeuvre-Lès-Nancy, France

Abstract

In domains such as transportation or medical technology, applications are often developed as Multi-Agent Systems (MAS). The usage of formal methods in this context has not been much addressed. With our experience with the B modelisation of the platooning problem as a base, we aim at demonstrating how the development and the verification of such systems using formal methods can be supported.The successive versions of the B models tended towards a more generic architecture that experts of the domain matched with a well-known MAS model.

Keywords: B method, Formal method, Situated Multi-agent system, Influence/Reaction model, Platooning, Specification, Verification

1 Introduction

Multi-Agent Systems (MAS) are widely used to develop applications in the field of transportation, medical technologies or space exploration. The difficulty of designing and studying situated MAS comes from the autonomy of the agents and their interactions with a common environment. These systems are highly distributed, where agents evolve in parallel, and more generally work in a dynamic environment. Individually, the agents may be very complex. Their composition leads to the problem of preservation or the occurrence of emergent properties at the global level of the system. Some of them are not always predictable from the local level. It is then difficult to formally express such systems and simulate them to predict their global behaviour [6,8].

Due to the contexts these systems are used in, i.e. critical contexts, the problem of ensuring their safety arises. To that end, formalisation is needed, which has begun to receive a substantial amount of interest. The questions linked to this formalisation are such as: can my MAS be formalised ? If not, can at least a smaller part of it be formalised ? If on the

⋆ This work has been partially supported by the ANR (National Research Agency) in the context of the TACOS project, whose reference number is ANR-06-SETI-017 (http://tacos.loria.fr), and by the pôle de compétitivité Alsace/Franche-comté in the context of the CRISTAL project (http://ww.projet-cristal.org).

[1] Email: Samuel.Colin@loria.fr

[2] Email: Arnaud.Lanoix@loria.fr

contrary the system can be formalised, are the desired properties expressible ? If they are, can they be verified ? In case of success, what new knowledge can be tracted from the formalisation ? Can the model be adapted to other similar MAS ? If not, are there at least some experience that could be used to formalise them more easily ?

The first questions have begun to be answered. For instance, Hilaire et al [9] propose a general framework for modelling MAS based on Object-Z and statecharts. This framework focuses on organisational aspects in order to represent agents and their roles. Similarly, Regayeg et al [11] combine Z notations and linear temporal logic to specify the internal part of agents and the specification of the communication protocols between agents. They propose general patterns and the use of Z support tools to model-check their specifications. It is to be noticed that the proposed patterns do not deal with dynamics of physical worlds.

Inverno and Saunders [10] have developed a multi-agent approach for simulating the behaviour of stem cells. Their aim is to highlight which properties are required on components in order to maintain general properties. Their formal models, written in Z, are based on a layered technique in which physical, biological and chemical environment are considered separately.

Schneider et al [12] apply their framework based on CSP and B as a starting point for the simulation of a biomedicine MAS. They focus on the clotting behaviour of artificial platelets. We can also point out a recent work [4] involving the use of Event-B. It focuses on the coordination between agents and only specifies the interaction protocol. Some patterns for the B specification of fault-tolerance protocols are proposed in the case of agent communication.

We ourselves are interested in the so-called platooning problem presented in Sect. 2, where the goal is to have several vehicles travel in a convoy by defining simple rules for each of them. Moving in a convoy is thus their emergent behaviour. Thanks to collaboration with the MAIA team, we developped B models for this problem presented in Sect. 3. We more particularly were interested in the problems or advantages linked with the choice of the B method. The successive versions of the B models tended towards a more generic architecture that experts of the domain matched with the Influence/Reaction (I/R) model proposed by Ferber & Muller [7]. From the particular platooning model we thus extracted generic B patterns for the I/R model. We presented these patterns in [14] and succinctly remind them in Sect. 4. Section 5 concludes this paper and gives some perspectives.

2 An Example of MAS : a Platoon of Vehicles

The Cristal project involves the development of a new type of urban vehicle with new functionalities and services. One of the major cornerstones of Cristal is the platooning problem.

A platoon is defined as a set of autonomous vehicles which have to move in a convoy, i.e. following the path of the leader (possibly driven by a human being) in a row (or a platoon) and that should observe an ideal distance between each other. The control of a platoon involves the longitudinal control of the vehicles, i.e. maintaining a certain distance between each other, and their lateral control, i.e. each vehicle should follow the track of its predecessor. Those controls can be studied independently [5]; we will only focus on the longitudinal control.

The behaviour of the vehicle's controllers has been given to us through projects' col-

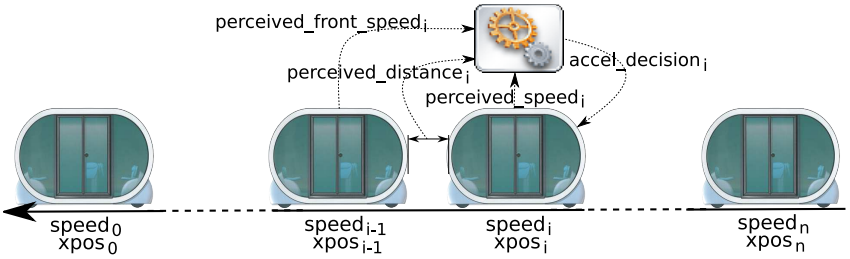

Figure 1. A platoon of vehicles

laboration by researchers of the MAIA team, which we will refer to as the experts of the domain. It can be summarised as follows, see Fig. 1:

- As we focus on the longitudinal control, the considered space is one-dimensional. Hence the position of the i^{th} vehicle is represented by a single variable $xpos_i$, its velocity by $speed_i$ [3] .
- $xpos_i$ and $speed_i$ are updated, depending on the current speed $speed_i$ of the vehicle and a decided instantaneous acceleration $accel\_decision_i$ passed to the engine.

$$new\_speed = speed_i(t) + accel\_decision_i(t+\Delta t).\Delta t$$

$$\text{if } new\_speed > Max\_speed, \begin{cases} xpos_i(t+\Delta t) = xpos_i(t) + \Delta t.Max\_Speed \\ speed_i(t+\Delta t) = Max\_Speed \end{cases} \tag{1a}$$

$$\text{if } new\_speed < 0, \begin{cases} xpos_i(t+\Delta t) = xpos_i(t) - \frac{speed_i(t)^2}{2.accel\_decision_i(t+\Delta t)} \\ speed_i(t+\Delta t) = 0 \end{cases} \tag{1b}$$

$$\text{otherwise}, \begin{cases} xpos_i(t+\Delta t) = \left(\begin{array}{l} xpos_i(t) + speed_i(t).\Delta t \\ + \frac{accel\_decision_i(t+\Delta t).\Delta t^2}{2} \end{array} \right) \\ speed_i(t+\Delta t) = new\_speed \end{cases} \tag{1c}$$

  These mathematical laws assume that the actuators of the engine are perfect. Moreover, three cases are distinguished, depending of the considered new speed.
- Each vehicle uses sensors for estimating its velocity $perceived\_speed_i$, the distance $perceived\_distance_i$ to its leading vehicle and the velocity $perceived\_front\_speed_i$ of its leading vehicle. The sensors are supposed to be perfect. Of course, the leader does not need the last two pieces of information as it has no preceding vehicle:

$$\begin{array}{r} perceived\_speed_i(t+\Delta t) = speed_i(t) \\ perceived\_distance_i(t+\Delta t) = xpos_{i-1}(t) - xpos_i(t), \text{ if } i > 0 \\ perceived\_front\_speed_i(t+\Delta t) = speed_{i-1}(t), \text{ if } i > 0 \end{array} \tag{2}$$

- Each vehicle can influence its speed by computing and pass to the engine an instantaneous acceleration $accel\_decision_i$. The acceleration can be negative, corresponding to the braking of the vehicle. $accel\_decision_i$ is defined according to the sensor values using a law similarly defined than the previous ones, but which cannot be given here for confidentiality reasons.

Note 1 *These laws come from the experts of the domain. Our goal is to develop a formal framework in order to implement them and prove properties of the obtained model. The properties we are looking forward to in this model are among the following:*

[3] No units are specified, thus the position or the speed could be based on millimetres as well as meters.

- *The model is sound bound-wise, i.e. none of the specified bounds are violated.*
- *No collision occurs between the vehicles.*
- *No unhooking occurs, i.e. the distance between vehicles cannot be infinitely long.*
- *No oscillation occurs, i.e. a phenomenon of a wave propagates from ahead of the platoon to its back, without never stabilising.*

We focus on the soundness of the model and the absence of collision in the remainder of this document, but the reader must be aware that it is still an ongoing work.

3 A B Model for the Platooning Problem

We briefly describe the B method before we present a B model of the platooning problem and difficulties of the proof of this model.

3.1 The B Method

B is a formal software development method used to model and reason about systems [1]. It is based on set theory and relations. Its key features are:

- First-order *set-theoretic* foundations, which are a well-understood domain of mathematics.
- *Modularity* for helping with the specification of big systems by means of separated development
- *Refinement* for a detail-wise incremental development

The state of a model is expressed through its variables and invariant. Its evolution is expressed through its methods. Verifying a model entails the verification the correctness of its methods w.r.t. its invariant, by taking into account the model it refines, if any.

The B method has proved its strength in the industry in complex real-life applications such as the Roissy VAL [3]. The B method is also supported by academic [4] and commercial [5] [6] support tools.

3.2 Organisation of the B Model for the Platooning Problem

The components of our model and their organisation is depicted Fig. 2, where:

Constants holds the constants used throughout the model and the hypotheses on these constants.

PhysicalVehicles models the interactions between vehicles and the environment, i.e. their perceptions and their actions.

VehiclesControllers models the steps a vehicle can be in: a vehicle can perceive, decide for an acceleration or attempt to move.

Scheduler schedules the steps all the vehicles go through: all vehicles do their perceptions, then they all do their decisions and finally their actions. Due to B expression, Scheduler

[4] B4free: http://www.b4free.com/
[5] AtelierB: http://www.atelierb.eu/
[6] B-toolkit: http://www.b-core.com/btoolkit.html

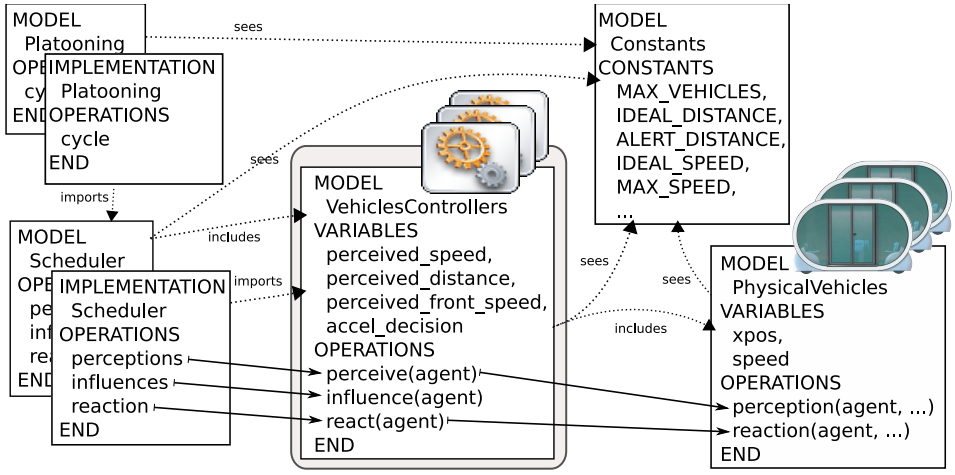

Figure 2. Organisation of the platooning B model

is a so-called *implementation* model because loops can only be expressed in them. Hence Scheduler refines the abstract expression of scheduling, a model called Scheduler_abs.

Platooning schedules each global step expressed in Scheduler, i.e. it schedules in a loop all the perceptions, all the decisions and all the actions. For the same reason as above, Platooning is an implementation which refines Platooning_abs.

Note 2 *Instead of having n identical B machines, each one associated with a vehicle, we have a single B machine representing all the n vehicles. We have abstracted the variables of each vehicle into arrays (or functions, in B terms) indexed by the index of the vehicle and whose contents is the purpose of the original variable. For instance, $speed_i$ will not be a single variable of the hypothetical B machine PhysicalVehicles_i but rather the value at the i^{th} index of the speed array.*
The same goes for VehiclesControllers.

Remark 3.1 The mathematical model presented Sect. 2 relies on the continuous domain of real numbers. The B method only allows integer numbers. We thus assume that the units of the models are precise enough to "absorb" the errors introduced by the use of integers. For instance, the distances will correspond to millimetres instead of meters. The problem of errors introduced by the digital representation of numbers is bound to happen anyway, when using floating-point real numbers in the implementation step, for instance.

3.3 B Model for the Environment

As stated in Sect. 3.2, the environment is modelled by PhysicalVehicles. Section 2 introduced the various laws to be used for modelling the platooning problem. The laws involved in the environmental part of the model concern the perceptions and the reaction, each one found in its respective method as follows.

The perception method models the perceptions a vehicle has about its environment, as expressed by (2). These perceptions are supposed perfect: the returned distance corresponds to the difference between the position of a vehicle and its leading one. The leader vehicle is a specific case explicited in the conditional.

```
|| IF i = 0
   THEN
     new_perceived_distance := old_perceived_distance ⩤ { 0 ↦ 0 }
   || new_perceived_front_speed := old_perceived_front_speed ⩤ { 0 ↦ 0 }
   ELSE
     new_perceived_distance := old_perceived_distance ⩤ { i ↦ (xpos(i−1) − xpos(i)) }
   || new_perceived_front_speed := old_perceived_front_speed ⩤ { i ↦ speed(i−1) }
   END
```

The reaction method models the update of the environment w.r.t. the acceleration decided by a given vehicle, as expressed by (1). The three cases described Sect. 2 are expressed in the conditional structure of the method: either the speed-to-be is over the maximum speed, or less than zero or between the defined bounds.

```
IF (considered_speed > MAX_SPEED)
THEN
  xpos(i)  := new_xpos_when_max_speed(xpos(i)) ||
  speed(i)   := MAX_SPEED
ELSE
  IF (considered_speed < 0)
  THEN
    xpos(i)  := new_xpos_when_neg_speed(xpos(i),speed(i),acceleration(i)) ||
    speed(i)   := 0
  ELSE
    xpos(i)  := new_xpos_others(xpos(i),speed(i),acceleration(i)) ||
    speed(i)   := considered_speed
  END
END
```

Formulas for calculating the new position are enclosed into functions for readability purposes. Equation (1a) in charge of calculating the new position when the considered speed is over the maximum speed is defined as follows:

$$\begin{array}{l} \text{new\_xpos\_when\_max\_speed} \in \mathbb{Z} \rightarrow \mathbb{Z} \\ \wedge\ \text{new\_xpos\_when\_max\_speed} = \%(\text{xpos}).(\text{xpos} \in \mathbb{N} \mid \text{xpos} + \text{MAX\_SPEED} \times \text{TIME\_STEP}) \end{array}$$

Properties are specified in the invariant. For instance, the fact that the vehicles never go backwards and don't violate the maximum speed bound is expressed as:

$$\text{speed} \in 0..\text{MAX\_VEHICLES} \rightarrow 0..\text{MAX\_SPEED} \wedge \text{xpos} \in 0..\text{MAX\_VEHICLES} \rightarrow \mathbb{N}$$

Similarly, the safety property of no collision is expressed as the position between two consecutive vehicles should be greater than a minimal distance determined as being the distance of collision.

$$\forall i \,.(\, i \in \text{dom}(\text{xpos}) - \{0\} \Rightarrow \text{xpos}(i-1) - \text{xpos}(i) \geq \text{CRITICAL\_DISTANCE})$$

Proof and its impact on the model

Properties expressed in the invariant of PhysicalVehicles have various influences on its methods.

- The perception method builds perceptions upon the true variables of the environment, hence properties for the perceptions derive directly from properties of the environment. While obvious and seemingly innocuous, this remark is of great importance w.r.t. the properties of the agents.
- The reaction method as-is ensures the property of not violating the bounds. The absence of collision requires a strengthening of the precondition: we decided to add the statement that the distance between the involved vehicles is greater than a certain dangerous dis-

tance called ALERT_DISTANCE. Adding this requirement highlighted at the proof step that the model missed hypotheses between the constants representing the key distances, or had wrong hypotheses which did not permit to achieve the proof. For instance the relationship between ALERT_DISTANCE and CRITICAL_DISTANCE became:

```
ALERT_DISTANCE > MAX_SPEED × MAX_SPEED / (−MIN_ACCEL)
                 + MAX_SPEED × TIME_STEP + CRITICAL_DISTANCE
```

3.4 B Model for the Vehicle Controllers

The vehicles interact with their environment by perceiving it and by applying their decisions. The methods perceive and react are not shown here as they are just wrappers calling the relevant methods of PhysicalVehicles.

The vehicles must also decide which acceleration they apply based on their perceptions, as reflected by the influence method. The specific case of the leader is taken into account. For readability purpose, the computation of the decision in the non-leader case has been defined into a specific compute_new_accel function which is not shown here for confidentiality reasons. Moreover, the resulting acceleration is filtered so as to stay between the set bounds of the system.

```
IF i = 0
THEN
  ANY new_accel
  WHERE new_accel = ( IDEAL_SPEED − perceived_speed(i) ) / TIME_STEP
  THEN
    accel_decision(i) := min({MAX_ACCEL, max({MIN_ACCEL, new_accel})} )
  END
ELSE
  IF (perceived_distance(i) < ALERT_DISTANCE)
  THEN
    accel_decision(i) := MIN_ACCEL
  ELSE
    ANY new_accel
    WHERE new_accel = compute_new_accel(perceived_distance(i),
                                        perceived_front_speed(i),
                                        perceived_speed(i))
    THEN
      accel_decision(i) := min( {MAX_ACCEL, max({MIN_ACCEL, new_accel})} )
    END
  END
END
```

Proof and its impact on the model

The proofs for VehiclesControllers involve proving the soundness of the computation, i.e. the filtering of the acceleration mentioned above, and the fact that the position of two consecutive vehicles are distant of more than ALERT_DISTANCE. At the moment, absence of collision can not be proved because of difficulties for the tool to prove arithmetically-heavy formulas. Nonetheless earlier experiments for simpler and less relevant properties show us the full process of completing the model with new properties.

We stated that in the PhysicalVehicles model, the precondition of reaction could be strengthened so as to ensure the invariant of the model. This strengthening in turn has an impact on the VehiclesControllers. Indeed, the reaction method is called by the react method, hence any property stated in the precondition of reaction has to be ensured by react. Hence its precondition has to be strengthened as well. This effect propagates back and forth in the invariant and the methods of VehiclesControllers:

- The strengthened precondition of react requires new statements about the influences in the invariant
- These new statements have to be ensured by the influence method, hence its precondition is strengthened
- The influences being a consequence of the perceptions, new properties about the perceptions are stated in the invariant
- The perceive method has to ensure these new properties about perceptions

At this point, strengthening the precondition of perceive is useless, because perceptions are a direct consequence of the variables of PhysicalVehicles. Hence we can see here the importance of the relationship between properties of the environment and perceptions (see Para. 3.3) : any new property comes back to itself through the entire model, from the reaction method, through the various steps of the vehicles in reverse order, to the perception method.

Remark 3.2 Any new property express in the model must have a lot of thought put in it, so as to be not too strong nor too weak, because it is self-referential through one step of functioning of the model. It also exhibits an intrinsic control loop in the model which can be made explicit as explained in the next section.

3.5 B Models for the Global Control Loop

The VehiclesControllers model has an implicit execution trace: preconditions of its various methods include an indication on the step the methods have to be "computed" at. Thus PhysicalVehicles, VehiclesControllers and Constants are enough for describing the whole model.

Nonetheless, it is possible to make this implicit scheduling of operations explicit by adding loops to the system. As depicted in the general architecture of the model presented Fig. 2, Scheduler specifies how to achieve each global step (perceptions, decisions and actions). The corresponding methods are WHILE loops over the vehicles in turn from the last vehicle up to the leader. Similarly, the global loop expressed in Platooning makes the perceptions, influences, reaction sequence explicit.

3.6 Verification of the Model

As for the model with the sole property of soundness, the generated proof obligations consisted of 614 obvious proof obligations and 103 non-obvious proof obligations. Validation of the obvious ones was immediate (hence their name "obvious"). The verification of the non-obvious formulas was automatic for 95 of them, leaving 8 formulas to be proved interactively, even if for some of them it is still an overstatement. The precise distribution of proof obligations is given Table 1.

The formulas to be proved interactively were divided between simple and complex statements. By "simple statements", we mean formulas such as the following predicate:

$$
\begin{array}{l}
\quad i \in \mathbb{Z} \land j \in \mathbb{Z} \land k \in \mathbb{Z} \\
\land\ i \leq j \\
\Rightarrow \\
\quad \min(\{j, \max(\{i,k\})\}) \in i \mathinner{..} j
\end{array}
$$

Component	Obvious	Proofs	Automatic	Interactive
Constants	1	0	0	0
PhysicalVehicles	15	16	11	5
Platooning_abs	3	0	0	0
Platooning	373	6	6	0
Scheduler_abs	14	0	0	0
Scheduler	90	59	59	0
VehiclesControllers	118	22	19	3
TOTAL	614	103	95	8

Table 1
Proof results for the B model of platooning

This statement coming from the VehiclesControllers is easy to prove by pen and paper as well as with the proof tool: commanding it to try using more in-depth search through a single command immediately proves this formula. This is the reason why we wrote that using "interactive proving" for such a formula is a bit of an overstatement.

Still, we are also left with more thorough formulas to prove. For instance, we have to prove that the calculations in the reaction method keep the speed and position in the specified bounds. For instance, in case the vehicle attempts to go backwards:

$$
\begin{array}{l}
\quad ... \\
\quad \wedge\ 0 \leq -1-\text{speed(i)}-\text{acceleration(i)} \\
\quad \wedge\ -1-\text{MAX\_SPEED}+\text{speed(i)}+\text{acceleration(i)}+1 \leq 0 \\
\Rightarrow \\
\quad \text{xpos(i)}-\text{speed(i)} * \text{speed(i)}/(2 * \text{acceleration(i)})\ \in \mathbb{N}
\end{array}
$$

The formula itself is longer but we left out some of the hypotheses for clarity. Knowing that the speed is always positive, we can infer that acceleration is negative, hence the right term of the subtraction is positive. As position is a natural number, we can deduce that the whole term is positive, thus the formula is true.

The proof tool can not infer all this knowledge by itself, that is why we have to hint it into proving the positiveness of the whole term by proving the positiveness of the various subterms. The difficulty of proving this last formula thus comes from the inability of the proof tool to find the relevant theorems. This point is the sole difficulty in verifying the whole model with the soundness property.

As for the model with the absence of collisions, we run into new difficulties which require an evolution of the model. These difficulties come from the lack of hypotheses. Let us analyse for instance one of the proof obligation for the reaction method when ensuring that no collision occur:

$$
\begin{array}{l}
\quad \forall \text{i}\ .(\ \text{i} \in \text{dom(xpos)} \wedge\ \neg(\text{i} = 0) \\
\qquad \Rightarrow \\
\qquad\quad \text{CRITICAL\_DISTANCE} \leq \text{xpos(i}-1) - \text{xpos(i)}\ ) \\
\quad \wedge\ ... \\
\quad \wedge\ 0 \leq -1-\text{MAX\_SPEED}+\text{speed(i)}+\text{acceleration(i)} \\
\Rightarrow \\
\quad 0 \leq -\text{MAX\_SPEED}-\text{CRITICAL\_DISTANCE}+\text{xpos(i}-1)-\text{xpos(i)}
\end{array}
$$

This formula corresponds to the case where the considered speed is over the maximum speed, hence the new position of the vehicle depends solely on the distance covered at maximum speed. As the irrelevant hypotheses have been left out, we can see that it's not possible to prove this goal: intuitively stated, for the vehicles not to enter into a collision, they must actually not already be in a collision.

This proof obligation had us add a new precondition in the reaction method:

$$(\ i \neq 0 \Rightarrow xpos(i-1) - xpos(i) \geq ALERT\_DISTANCE)$$

Consequences of this strengthening was explained Para. 3.4. Moreover, the new precondition required knowledge of the relationship between ALERT_DISTANCE and CRITICAL_DISTANCE (see Para. 3.3). With the addition to the precondition of the reaction method and this new hypothesis, we were able to prove interactively the proof obligation depicted earlier.

Remark 3.3 We can thus see that for more complex properties, knowledge of the domain becomes mandatory to be able to solve the proof obligations. This leads us to state a well-known yet important point in the use of formal methods. When the properties to be proved are linked to the building of the model (as the soundness for the platooning model), the verification merely depends on the intrinsic complexity of the formulas. When the properties to be proved are linked to the model itself (as the absence of collisions), domain knowledge becomes mandatory, hence a dialogue has to be kept with the experts.

For reference, the current version of the model with the absence of collision requires the verification of 110 proof obligations, 99 of which are proved automatically, 8 interactively and 3 still to be proved.

3.7 B Models Development History

The process of developing a satisfactory model for the platooning problem has required several versions converging to the model presented before:

- Version 1 & 2 introduce a single B machine to model waggons along a rail. Indeed, the longitudinal control of the platooning is very close to that kind of model
- Version 3 splits the model into two parts, one for the mathematical representation of the physical environment of the vehicles, the other for the initiatives pertaining to the vehicles, i.e. deciding for an acceleration and moving
- Version 4 refines the vehicles model by separating their possible states: a vehicle could perceive its environment, decide for its acceleration and apply this acceleration. In this version, the identified constants, such as the minimum and maximum bounds for speed or the acceleration, are put in a separate machine
- Version 5 & 6 introduce the expression of the global loops of the model, i.e. how the various steps are scheduled for each vehicle
- Version 7 is the version presented in the previous section.

This evolution has our collaboration with the MAIA team as a background. The various models have evolved thanks to the expertise of the MAIA team brought at each difficulty encountered while modelling and proving.

Remark that the vehicles have been abstracted by one single model since the beginning of the development process. As stated Note 2, all variables are actually arrays indexed by the number of vehicles. We had to resort to this modelling because the B method does not allow the reasoning over an arbitrary yet unknown number of modules. Such reasoning could be done in formal methods suited for distributed models, such as process algebras for instance.

4 B Patterns for the Influence/Reaction Model

The successive versions of the B models tended towards a more generic architecture that our collegues from the MAIA team matched with the Influence/Reaction (I/R) model proposed by Ferber & Muller [7]. From the particular platooning model we thus extracted generic B patterns for the I/R model presented succinctly here. For a more complete description, please see [14].

4.1 The Influence/Reaction Model

Ferber & Muller [7,6] proposed the I/R model in order to clearly express the dynamics of situated MAS. In this model, agents are described separately from the environment. The connection is done by computing, at each step Δt, which state each agent perceives and which influences it produces. The new state of the environment is defined as the combination of the various influences produced by the agents where:

- A set of variables, called *global variables* and denoted $global_1, \ldots, global_l$, represent the global state of the environment. ;
- Internal variables of each agent, called *local variables*, express its internal state, which are perceptions and memories, denoted $local_1, \ldots, local_m$;
- Influences express the actions produced by agents (e.g. acceleration) in order to change the system, denoted $influence_1, \ldots, influence_n$.

The evolution of an I/R model is computed following a specific cycle as shown Fig. 3: (i) all the agents perceptions are done, (ii) all the local behaviours are computed (iii) all influences are decided, and (iv) all the influences are combined to compute the reaction of the environment.

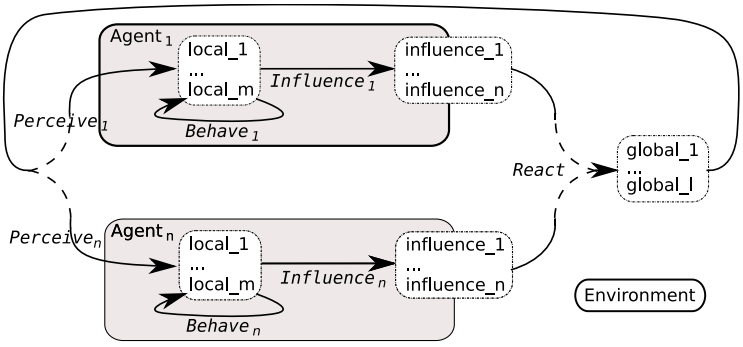

Figure 3. I/R model

The evolution is formalised by some specific laws given by functions on the variables of the model :

- $Perceive_i$ are perception functions from the environment to the internal state of the i^{th} agent;
- $Behave_i$ are behavioural functions computing a new internal state of the i^{th} agent from its perceptions and its previous state, corresponding to memorisation of information ;
- $Influence_i$ are action functions that produce a set of influences on the environment for the i^{th} agent;

- *React* is the reaction function computing the new state of the environment from its current state and the combination of all the influences produced by all the agents. To combine these influences, we sequentially take them into account by considering a $React_i$ function for each agent.

In the next section, we show a generic B writing of the I/R model that implements this cycle function.

Note 3 *Two kind of situated MAS can be distinguished : the* hysteretic *agents which can memorise information and the* tropistic *agents which have no internal behaviours. In the latter case* $Behave_i$ *can be removed.*

4.2 B Design Patterns of the Influence/Reaction Model

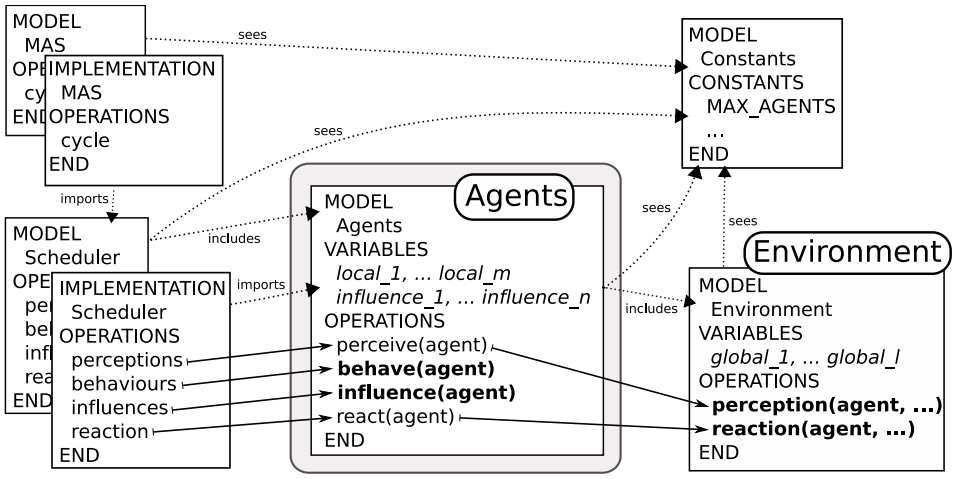

Figure 4. B patterns of the I/R model

All situated MAS that can be expressed with the I/R model can be written as an instantiation of B design patterns given Fig. 4 (see also [14]). More precisely:

(i) The two B models MAS and Scheduler express the evolution of an I/R model, i.e. the loop on all the perceptions, the decisions (behaviours and influences) and the global reaction. These B patterns are completely generic. They are only parametrised by the number of agents MAX_AGENTS and assume the existence of the tree other B models.

(ii) The B model Constants specifies the number of agents MAX_AGENTS, and can be completed with other specific constants or axioms required by the example.

(iii) The B pattern Agents specifies the perceptions and the memories of all the agents. Perceptions and memories are denoted $local_1, \ldots, local_m$ and the influences are denoted $influence_1, \ldots, influence_n$. All these variables are represented as arrays expressed in B by total functions between the numbers of agents (the indices) 0..MAX_AGENTS and their associated values.

The INVARIANT part of Agents contains typing predicate on the variables of the model. It can also contain safety properties between the $local_j$ and $influence_k$ variables, once the model has been instantiated for a specific problem.

The dynamics of the agents follow the cycle evolution of an I/R model. Four methods – one for each step – are managed by the Scheduler model.

- Methods perceive and react are generic because their bodies only correspond to operation calls on the environment methods perception and reaction;
- the method behave implements the local behavioural function $Behave_i$ of an agent, i.e. computations and the memorisation before the agent compute its influences.
- the method influence computes the influences which have to be passed to the environment from the considered agent, by implementing the $Influence_i$ function.

(iv) The B pattern Environment gives a general B model of the environment, i.e. a model of the laws of the world the agents evolve in. It contains all the variables $global_1$, ..., $global_l$ used to state the environment and represented by total functions between 0..MAX_AGENTS and their associated values.

The INVARIANT part of the generic model contains only typing predicates on the variables of the model. Safety properties can also be expressed here.

The dynamics of the environment result from its "physical" interactions with the agents. These interactions can be instantiated with the two methods:

- perception is a method called by each agent to perceive its environment. It must be an B implementation of the $Perceive_i$ functions.
- reaction is a method called by each agent so that influences are combined in order to perform its actions. It takes as parameters the influences decided by the agents and implements the $React_i$ function.

Note 4 *After a MAS has been instantiated with the previous patterns, safety properties can be introduced into the invariant of the models. Their proofs can require the strengthening of the methods' preconditions as presented Para 3.4.*

Example 4.1 Please notice that the platooning problem is an instantiation of these B design patterns to a specific case study where:

- the environment of all the vehicle's controllers is defined by $xpos_i$ and $speed_i$;
- the perceptions of each agent are $perceived\_speed_i$, $perceived\_distance_i$ and $perceived\_front\_speed_i$;
- the only influence decided by the agents is $accel\_decision_i$;
- the law of evolutions are given by equations, more precisely (2) gives the perception functions and (1) gives the reaction function.

No information have to be memorised, so we can remove the $Behave_i$ function as said Note 3.

5 Conclusion

We have proposed a B model for the platooning problem whose soundness has been verified with the help of B proof tools. Expressing additional properties into the model helped us to quickly pinpointing weaknesses in the assessment of the hypotheses. Not surprisingly from a software engineering point of view, knowledge of the experts of the domain was required for completing the hypotheses of the system. The advantage of using a formal method here was to avoid resorting to lengthy experimentations in order to understand where the model had flaws: the proof process helped identifying them easily.

The evolution of this model led us naturally to abstract the architecture from the problem itself. The obtained patterns matched very closely the well-known Influence/Reaction

model of the Multi-Agent community [6]. We completed this small gap by proposing generic B patterns suitable for instantiating any MAS expressed with the I/R model. With the use of a formal method, the originality of these patterns is to provide a framework for expressing soundness of the instantiated MAS and specifying additional emerging properties in a tool-supported environment. The expectable advantages of this approach were already stated for the particular model of the platooning problem.

Further work includes the study of the same platooning problem with related formalisms such as CSP||B [13] and Event-B [2]. It also includes understanding better the self-referential nature of some emerging properties in the platooning problem: absence of collision, unhooking, oscillation. The goal there would be to see if there are evolution and proof patterns linked to the expression of additional properties in the model.

Acknowledgement

We address our many thanks to Olivier Simonin, Alexis Scheuer and François Charpillet, from the MAIA team of the LORIA, for common efforts and fruitful discussions in the context of the TACOS and CRISTAL projects.

References

[1] Abrial, J.-R., "The B Book," Cambridge University Press, 1996.

[2] Abrial, J.-R. and S. Hallerstede, *Refinement, decomposition, and instantiation of discrete models: Application to Event-B*, Fundamenta Informaticae **77** (2007), pp. 1–28, special issue on ASM'05.

[3] Badeau, F. and A. Amelot, *Using B as a high level programming language in an industrial project: Roissy VAL*, in: *ZB 2005: Formal Specification and Development in Z and B, 4th International Conference of B and Z Users*, LNCS **3455** (2005), pp. 334–354.

[4] Ball, E. and M. Butler, *Event-B patterns for specifying fault-tolerance in Multi-Agent interaction*, in: *Proceedings of Methods, Models and Tools for Fault Tolerance*, Oxford, UK, 2007.

[5] Daviet, P. and M. Parent, *Longitudinal and lateral servoing of vehicles in a platoon*, in: *Proceeding of the IEEE Intelligent Vehicles Symposium*, 1996, pp. 41–46.

[6] Ferber, J., "Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence," Addison-wesley Professional, 1999.

[7] Ferber, J. and J. P. Muller, *Influences and reaction : a model of situated multiagent systems*, in: *2nd Int. Conf. on Multi-agent Systems*, 1996, pp. 72–79.

[8] Helleboogh, A., G. Vizzari, A. Uhrmacher and F. Michel, *Modeling dynamic environments in multi-agent simulation*, Autonomous Agents and Multi-Agent Systems **14** (2007), pp. 1–27.

[9] Hilaire, V., P. Gruer, A. Koukam and O. Simonin, *Formal specification approach of role dynamics in agent organisations: Application to the Satisfaction-Altruism Model*, in: *Int. Jour. of Software Engineering and Knowledge Engineering (IJSEKE)* (2006).

[10] Inverno, M. and R. Saunders, *Agent-based modelling of Stem Cell organisation in a Niche*, in: S. A. Brueckner, G. Di Marzo, S. A. Karageorgos and R. Nagpal, editors, *Engineering Self-Organising Systems : Methodologies and Applications*, LNAI (2005).

[11] Regayeg, A., A. H. Kacem and M. Jmaiel, *Specification and verification of multi-agent applications using temporal z*, in: *Intelligent Agent Technology Conf. (IAT'04)* (2004), pp. 260–266.

[12] Schneider, S., A. Cavalcanti, H. Treharne and J. Woodcock, *A layered behavioural model of platelets*, in: *11th IEEE International Conference on Engieerging of Complex Computer Systems, ICECCS*, 2006.

[13] Schneider, S. A., H. E. Treharne and N. Evans, *Chunks: Component verification in CSP||B*, in: *IFM'05* (2005).

[14] Simonin, O., A. Lanoix, S. Colin, A. Scheuer and F. Charpillet, *Generic Expression in B of the Influence/Reaction Model: Specifying and Verifying Situated Multi-Agent Systems*, INRIA Research Report 6304, INRIA (2007).
URL http://hal.inria.fr/inria-00173876/en/