
Specifying in B the Influence/Reaction Model to Study Situated MAS

Application to vehicles platooning

Olivier Simonin ^{*}, Arnaud Lanoix ^{**}, Alexis Scheuer ^{*},
François Charpillat ^{*}

^{*} MAIA – LORIA – INRIA Lorraine, Nancy Universités
Campus Scientifique, BP 239, 54506 Vandœuvre lès Nancy, France
{Firstname.Lastname}@loria.fr

^{**} AeLoS – LINA, Université de Nantes
2, rue de la Houssinière, BP 92208, 44322 Nantes, France
Arnaud.Lanoix@univ-nantes.fr

Abstract.

This paper addresses the formal specification and verification of situated Multi-Agent Systems (MAS) that can be formulated within the Influence/Reaction model as proposed in 1996 by Ferber & Muller. In this model, our objective is to prove the correctness of reactive MAS with respect to a certain formal specification or property, using formal methods. This is an important step to bring MAS to high quality standards as required for critical applications encountered in domains such as transport systems. A generic B representation of systems instantiating the Influence/Reaction model is proposed, using patterns of specification. We illustrate our approach with a MAS to control unmanned land vehicles to form a platoon. The paper ends with considerations about further improvements of the framework, involving simulation and study of the properties of the system.

Keywords: *Situated MAS – Influence/Reaction model – B method – Design patterns – Platooning*

1. Introduction

We aim at defining a general approach to formally specify and study situated Multi-Agent Systems (MAS), *i.e.* systems composed of agents

Studia Informatica Universalis.

which interact in a physical environment (real or virtual). We are especially interested in addressing critical decentralised systems in which autonomous components interact with each other to form a complex system. The MAS approach allows the modeling of such systems with a bottom-up methodology. The components (agents), their behaviours, the way they interact with each other or with the environment define the local level. It gives rise to the collective behaviours of a system (the global level) whose properties are not always predictable from the local level. This approach permits the definition of flexible decentralised systems able to (self)organise. Such systems are appealing with respect to their faculty of adaptation but they are difficult to study. Specifying and verifying properties of such systems remains an open issue, which we attack in this paper.

Our approach consists in choosing the Influence/Reaction (I/R) framework [10], which is one of the few models expressing dynamics of situated MAS. It is also simple enough to make the challenge of verification reachable for simple properties.

In order to specify the I/R model, we adopt the B method, used for modelling and reasoning about systems. The B method already demonstrated its ability to verify industrial-strength software for autonomous systems (as *e.g.* French automatic subway [6]). We aim here at exceeding the software framework so as to specify the physical part of situated MAS, as the overall dynamics of such systems depend on the interactions between the agents and their environment, *i.e.* a physical world. B only works at the level of integers, hence it seems not adapted to the modelling of MAS which are intrinsically continuous systems. However we circumvent this limitation by abstracting over the granularity of the model, *i.e.* the physical units of the variables.

In this paper we propose B design patterns able (i) to express the main parts of situated MAS and their dynamics following the I/R definitions, and (ii) to help in verifying their local and global properties. Design patterns are a good way of communicating expertise by capturing the solutions to recurring design problems and re-using those solutions. We illustrate how our B patterns can be instantiated to study a specific MAS, by focusing on the platooning task.

The paper is organised as follows. Section 2 presents the I/R model in order to clearly express dynamics of situated MAS. Then we propose in section 3 a generic expression in B for the I/R model using design patterns. Section 4 shows how patterns can be instantiated to study a real complex case: a reactive MAS model for coordination of several autonomous vehicles. Eventually section 5 presents related works, and section 6 concludes and gives some perspectives.

2. The Influence/Reaction Model

The difficulty of designing and studying situated MAS comes from the autonomy of agents and their interactions within a common environment. They are highly distributed systems. It is then difficult to formally express/simulate such systems and to predict their global behaviour [9, 12].

In particular, most MAS models do not allow to express simultaneous actions. They propose often a sequential representation of actions, which is not generally equivalent. For instance two robots situated on both sides of a door, and trying to open and close it simultaneously, should fail as their forces are balanced. In a sequential representation of actions, the door will sequentially be open and closed, or vice versa.

Ferber & Muller proposed in [10] the Influence/Reaction model in order to express clearly the dynamics of such systems, considering a discretization of time. In this model, agents are described separately from the environment dynamics but connected to it by computing at each step which state they perceive (perceptions) and which influences they produce (actions). This dichotomy allows mainly to compute the result of simultaneous actions performed by different agents at a given time. Indeed, the new state of the system is defined as the combination of the different influences produced by the agents.

Figure 1 illustrates the I/R loop : perceptions of the environment release agent influences, which are actions modifying the whole system. The I/R model is formalised as shown in figure 2, where $s_i(t)$ is the internal state of the i^{th} agent at time t and $\sigma(t)$ the state of the environment at the same time t . P_i is a perception function from the environment to

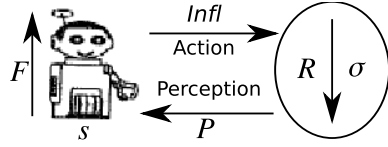


Figure 1: Interaction between an agent and its environment in the I/R model [9]

$$\left\{ \begin{array}{l} s_1(t+1) = F_1(s_1(t), P_1(\sigma(t))) \\ \dots \\ s_n(t+1) = F_n(s_n(t), P_n(\sigma(t))) \\ \sigma(t+1) = R(\sigma(t), \prod_i Infl_i(s_i(t+1))) \end{array} \right.$$

Figure 2: Formalisation of the I/R model [9]

the internal state of the i^{th} agent. F_i is a behavioural function that computes the new internal state of the i^{th} agent from its perceptions and its previous state. $Infl_i$ is an action function which produces a set of influences on the environment, for the i^{th} agent. At last, R is the reaction function computing the new state of the environment from its current state and the combination of all the influences produced by agents. Influences are combined thanks to the \prod operator. Details about definition of each function can be found in [9].

This coherent (and time discrete) expression of situated MAS give us the basis to formally specify their dynamics and properties. For this purpose, in the following, we propose a B expression of the I/R model.

3. Generic B Patterns of I/R Model

3.1. Specifying and Verifying with B

The B method is a formal software development method used to model and reason about systems [1]. It is based on set theory and relations. Software development in B supports abstractly specifying the requirements of the systems and then refining these requirements through several steps to create a concrete description of the system which can be automatically translated to code. Each B machine consists in variables representing the state of the model, operations representing the possible evolutions of this state and an invariant specifying the safety requirements. Each operation consists of a precondition part, which is a predicate conditioning the activation of the operation, and a substitution part specifying the effects of the considered operation.

The B method has been successfully applied in the development of several complex real-life applications, such as the Meteor project [6], the Roissy VAL [2] or the Coptilot project [15]. It is one of the few formal methods which has robust and commercially available support tools for the entire development life-cycle, from specification down to code generation.

Proofs of invariant consistency and refinement are part of each development. These *Proof Obligations* (POs) are generated from the model by applying certain semantic rules. Then, they can be proven with B support tools [19] or by any tool supporting first-order logic with set theory and axioms for the basic data types of B. Checking POs is an efficient and practical way to detect errors introduced during the development and to validate the correctness of the specified models.

3.2. B Patterns Architecture of the I/R Model

We propose a generic expression in B of the I/R model. The figure 3 shows a general rewriting of the I/R model using B concepts: (i) two B generic patterns express the *cycle* of the I/R model, (ii) one pattern expresses the agents behaviours, and (iii) one pattern expresses the environment evolution. Note that the given B models are not complete. They are only patterns and they have to be filled in for a specific system as it is illustrated in section 4.

To reduce the complexity of B patterns, we consider that agents are identical and defined by the same set of local variables and influences. However heterogeneity can be easily introduced by ignoring some variables in some agents.

The cycle Function of the I/R Model To formalise the I/R model in B, we express into the MAS machine (Fig. 3) an operation called cycle, which is the loop on all the perceptions, all the decisions (behaviours \rightarrow influences), and the global reaction. All these steps are characterised by a *step* variable valued by PERCEIVE, BEHAVE, INFL and REACT.

In order to use a “concrete” loop construction, the B method requires two levels of specifications. Into the implementation given Fig. 3, the loop is made explicit using a WHILE construction, that calls successively four “intermediate” operations perceptions, behaviours, influences and reaction corresponding to the four global steps of the I/R model.

These four operations are specified into the Scheduler B machine. perceptions (respectively behaviours, influences and reaction) expresses a loop by calling the perceive (respectively behave, infl and react) method in turn for each agent to the considered step. Moreover, to give a WHILE construction, we must use two levels of B specifications.

The Agents B Pattern Each agent is determined by its local variables, which characterise its perceptions and its memories, and by its influences. We specify all the agents in the same B pattern given in the machine Agents on Fig. 3.

- The $local_1, \dots, local_j, \dots, local_m$ variables are arrays associating each agent to its internal state, *i.e.* its perceptions or knowledge (memory). These arrays $local_j$ are represented in B by total functions defined between the numbers of agents (the indices) $0..MAX_AGENTS$ and the values contained thereinto.

- The $influence_1, \dots, influence_k, \dots, influence_n$ variables denotes intended actions produced by agents in order to change the system. Note that a communication is a particular influence produced by an agent and that can be perceived by one or several agents.

The INVARIANT part of Agents can also contain safety properties between the $local_j$ and $influence_k$ variables.

The perceive and react methods express the agent actions at the PERCEIVE and REACT steps of the I/R model. They are generic because their bodies correspond only to operations calls on the environment:

- perceive calls the perception method of Environment to perceive the environment data, and
- react calls the reaction method of Environment to propagate its influences into the environment.

The two methods to be filled in are *behave* and *infl*:

- *behave* is a B method which expresses the internal behaviour of each agent. It calls all the necessary $local_j = behave_j(local_1, \dots, local_m)$ to change its internal state;
- *infl* is the B method which produce the refresh influences using the necessary $influence_k = infl_k(local_1, \dots, local_m)$ functions.

The Environment B Pattern The Environment machine of Fig. 3 gives a general B model of the environment. To represent the state of the system, we define some $global_1, \dots, global_l$ variables. Each $global_i$ variable can model:

- specific data of each agent represented by arrays associating the data to the agents (for instance, their location);
- data not associated to any agent, but useful for modelling reason, as a global information necessary for all agents (*e.g.* location of other objects different from the agents). Note that the B pattern given in Fig. 3 focuses only on specific data of each agent.

The INVARIANT part of the generic model contains typing predicates on the $global_i$ variables.

The desired global safety properties of the system can also be expressed into the INVARIANT once the model has been instantiated for a specific problem. The proof of the B models will highlight what is required of the behavioural part to be able to ensure those properties.

The dynamics of the environment result from the “physical” interactions the agents have with it. These interactions consist of two methods, the first to perform agents *sensing*, the second to *combine* agents’ influences. These methods are called perception and reaction respectively in Fig. 3:

- *perception* is a method called by each agent to perceive its environment. Following B syntax, we call into perception the functions $local_j = perceive_j(global_1, \dots, global_l)$, which express the link between the perceptions $local_j$ and the environment variables. Each $local_j = perceive_j(global_1, \dots, global_l)$ correspond to a sensor update, which can imitate noise or sensor errors.

– reaction is a method called by each agent so that the influences are combined in order to update the agent global variables (its physical state). Objects are not considered but could be introduced as passive agents. The functions $global_i = react_i(global_1, \dots, global_i, influence_1, \dots, influence_n)$ linking environment data and influences have to be expressed here, by way of combining the influences. Noise or errors on the actuators can also be imitated here.

Moreover, the precondition part of both methods can be augmented to take into account safety properties, once the model has been instantiated for a specific problem.

4. Application to a Platooning Model

This section shows how to use the proposed B patterns to study decentralised platooning, *i.e.* convoys of autonomous vehicles following a leader [21]. As in [8], we suppose that lateral and longitudinal control are independent. We only consider here the longitudinal control, to simplify. This example is motivated by studying the properties of the model proposed by [8], especially the risk to have longitudinal collisions.

This example has been developed in the context of the CRISTAL project (French Pole de competitivite “Vehicule du Futur”) from a real case study.

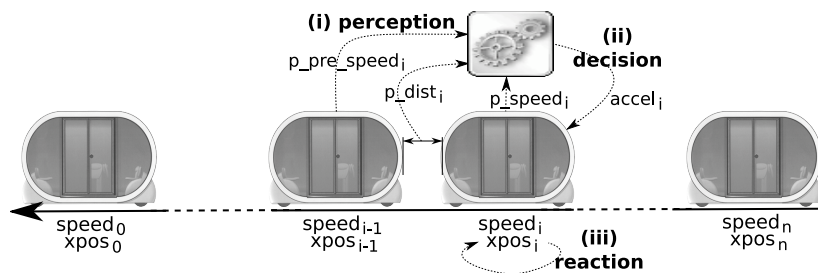


Figure 4: Platooning

A one dimensional platooning As depicted in Fig. 4 each vehicle’s state is given by an integer couple $(xpos_i, speed_i), i \in$

0..*MAX_VEHICLES*: $xpos_i$ is the vehicle's position, and $speed_i$ its bounded velocity. Distance and time scales do not have to be specified: they can be as small as needed.

Each autonomous vehicle (not the leader) selects its acceleration $accel_i$ according to its sensors' values: its velocity p_speed_i , the distance p_dist_i to the previous vehicle and the velocity $p_l_speed_i$ of this vehicle. To simplify, perceptions are supposed errorless.

The following platooning laws are taken from [8]:

- the leader aims to reach a velocity $IdealSpeed(t)$ which is either constant or computed online: $accel_0(t) = (IdealSpeed(t) - p_speed_i(t))/\Delta t$;
- the other vehicles aims to maintain a distance $IdealDist_i(t)$ (once again, either constant or computed online) between the vehicles: $accel_i(t) = (p_dist_i(t) - IdealDist_i(t))/(2\Delta t^2) + (p_l_speed_i(t) - p_speed_i(t))/\Delta t$.

These computed accelerations are limited by fixed bounds. The resulting acceleration is then used to compute the new state of each vehicle, using classical mechanics' laws [18, § III.A].

B Specification of Platooning The architecture of the B models follows the B patterns given in section 3, with a few renaming: Environment becomes PhysicalVehicles, Agents becomes VehiclesControllers (as depicted Fig. 5). Scheduler and MAS are kept.

The PhysicalVehicles machine contains the modelling of the physical world and its laws. The state of the vehicles are represented by two arrays, $xpos$ and $speed$, associating to each vehicle number the corresponding position or speed, respectively. It contains perception and reaction operations, to react to its controllers.

The global safety property we are interested in with this model is to establish that no collision can occur for instance. This can be done by adding into the INVARIANT of PhysicalVehicles an expression like:

$$\forall(ii).($$

$$(ii \in \text{dom}(xpos) - \{0\} \wedge xpos_updated(ii) = \text{TRUE} \wedge xpos_updated(ii-1) = \text{TRUE})$$

$$\Rightarrow xpos(ii-1) - xpos(ii) \geq \text{CRITICAL_DISTANCE})$$

which expresses that the distance between two updated positions of successive vehicles must be higher than `CRITICAL_DISTANCE`.

Figure 5 also shows the `VehiclesControllers` machine, with 3 local variables representing the perceptions (`p_speed`, `p_dist` and `p_l_speed`) and one influence (`accel_decision`). It contains all the internal decision mechanism of the agents, *i.e.* the `infl` method, which produces a new acceleration. As established in section 3, only two methods have to be explicitly written in `VehiclesControllers` (`perceive` and `react` already call their respective counterparts in the `PhysicalVehicles` model): `behave` which is bypassed because the agents are completely reactive, and `infl`.

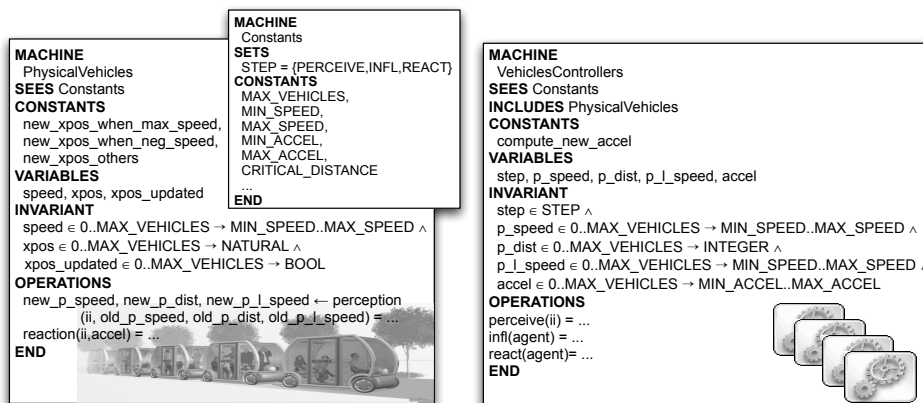


Figure 5: Part of the B specification of platooning

Soundness of the Model, Properties B files once written, current Proof Obligations (POs) for the platooning example are presented in Table 1. Most of them are automatically solved by the Atelier B, the others are proved interactively. An interactive proof means that the automatic part of the prover did not go far enough into the proof tree. We have to add new hypotheses and introduce sub-goals to guide the prover. Other difficulties comes from complex arithmetic, which is a hard point for theorem provers, but this tends to be less true over time.

Current unproved POs correspond to two cases:

Component	POs	Proved	Unproved	%Pr
Constants	1	1	0	100
PhysicalVehicles	31	24	7	77
MAS_abs	0	0	0	-
MAS	169	169	0	100
Scheduler_abs	0	0	0	-
Scheduler	64	64	0	100
VehiclesControllers	21	17	4	80
TOTAL	286	275	11	96

Table 1: Proof results for the current B model of platooning

– some of them are not yet proved interactively with the tool, because of the previously mentioned difficulties, but they seem true and some are done by pen and paper;

– other unproved POs and the difficulties to prove them pinpoint mistakes on the model which correspond to know limitations of the considered laws for the longitudinal control, *i.e.* collision may occur in some experimental conditions [18, § IV].

More generally, during the development process, the proof process has unlighted lacks of hypotheses about some constants of the system, such as the relationship between the bounds for the acceleration or between the acceleration and the maximal speed of the system. 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.

5. Related Works

Works on the specification and verification of MAS generally use as verification methods model-checking techniques. They can be divided in two main categories focusing on logical agents (*e.g.* [14, 7, 17, 16]) and deliberative/communicating multi-agent organisations (*e.g.* [20]). However these models are not adapted to study situated MAS, due in part to the combinatorial explosion generated by their complexity. Theorem proving approaches abstract this explosive number of states into predicates describing their general properties, thus theorem proving helps dealing with systems having an infinite, or really big, number

of states. The B method more particularly uses set theory, integer numbers and the theorems thereof to achieve this result.

We can nonetheless point out situated MAS designs based on model-checking. Hilaire *et al* [13] propose a general framework for modelling MAS that focuses on organisational aspects. Similarly, Regayeg *et al* [17] defined a new language based on the Z notation and linear temporal logic allowing specifications of the internal part of agents and the specification of the interaction protocol (communications) between the agents. At last, [3] formalise situated MAS using coloured Petri net. Once again this formalism is limited by the space explosion which requires some simplification of the model.

More closely to the B method, we can point out recent works [5, 4, 11] involving the use of Event-B (an adaptation of the B method for modelling reactive systems). These models focus on the coordination between agents and only specifies the interaction protocol that structures the agent negotiation and decision making.

6. Conclusion

We have proposed in this paper a formal pattern for the Influence/Reaction model proposed by Ferber & Muller [10]. This pattern is expressed with the B method, ensuring that a fully instantiated model fits the high quality standards of the domain of critical software. This pattern is also a generic expression of the Influence/Reaction model, allowing in particular to take into account simultaneous actions generated by agents. Indeed, any Multi-Agent System expressed through this model and defined over integer numbers, or that can be approximated with them, can be instantiated in this B pattern.

We furthermore illustrated the adequacy of our generic I/R formal model with the problem of platooning defined as a reactive MAS. We also obtained a platooning model that is sound up to typing constraints. We drew a general schema for adding and proving more general properties to an instantiated MAS and how the process unrolls when doing so.

The strength of our approach resides in the fact that we consider a formal model covering a whole system of agents. We are able to easily express the local properties of agents in the relevant part of the design pattern. We can easily match these local properties with desired global properties expressed in the environmental part of the design pattern. In other approaches, the design is focused on a single agent which is instantiated several times, and global properties are tentatively extracted, sometimes with difficulty, from the interaction of those agents. Our approach forces the developer to take into account at design time global properties, even if these are trivial. The global properties can be enriched later on without having to restart the design from the beginning.

Further developments of our proposition include elaborating on the expression of agents heterogeneity when considering one system. We also want to ease the specification and verification of global properties in the proposed framework. More generally, we started to develop an application that aims at automatically generating complete B patterns from agent behaviours and physical laws specified through an interface. We also intend to lift the limitation on integer numbers by defining the same kind of pattern for other formal methods supporting real numbers. Eventually, we plan to express in the platooning model more complex properties than presented, and to study other kinds of situated multi-agent systems.

Acknowledgements We address our many thanks to Samuel Colin, for common effort about the B specifications of the platooning problem in the context of the CRISTAL project.

References

- [1] J.-R. Abrial. *The B Book*. Cambridge University Press, 1996.
- [2] F. Bateau 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 Int. Conf. of B and Z Users*, volume 3455 of *LNCS*, pages 334–354. Springer-Verlag, 2005.
- [3] I. Bakam, F. Kordon, C Le Page, and F Bousquet. Formalization of a spatialized multi-agent system using coloured petri nets for the study of a hunting management system. In *FAABS 2000, Lecture Notes in Artificial Intelligence*, number 1871, pages 123–132, 2001.
- [4] E. Ball. *An Incremental Process for the Development of Multi-agent Systems in Event-B*. PhD thesis, University of Southampton, 2008.

- [5] E. Ball and M. Butler. Event-b patterns for specifying fault-tolerance in multi-agent interaction. In *Workshop on Methods, Models and Tools for Fault Tolerance*, pages 4–13, 2007.
- [6] P. Behm, P. Benoit, and J.M. Meynadier. METEOR: A Successful Application of B in a Large Project. In *Integrated Formal Methods, IFM'99*, volume 1708 of *LNCS*, pages 369–387. Springer Verlag, 1999.
- [7] F. Brazier, C. M. Jonker, and J Treur. Principles of component-based design of intelligent agents. *Data Knowledge Engineering*, 41(1):1–27, 2002.
- [8] P. Daviet and M. Parent. Longitudinal and lateral servoing of vehicles in a platoon. In *Proceeding of the IEEE Intelligent Vehicles Symposium*, pages 41–46, 1996.
- [9] J. Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-wesley Professional, 1999.
- [10] J. Ferber and J.P. Muller. Influences and reaction : a model of situated multiagent systems. In *2nd Int. Conf. on Multi-agent Systems*, pages 72–79, 1996.
- [11] H.-J. Gao, Z. Qin, L. Lu, L.-P. Shao, and X.-C. Heng. Formal specification and proof of multi-agent applications using event b. *Information Technology Journal*, 6(7):1181–1189, 2007.
- [12] A. Helleboogh, G. Vizzari, A. Uhrmacher, and F. Michel. Modeling dynamic environments in multi-agent simulation. *Autonomous Agents and Multi-Agent Systems*, 14(1):1–27, 2007.
- [13] V. Hilaire, 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)*. in press, 2006.
- [14] C. M. Jonker and J Treur. Compositional verification of multi-agent systems: A formal analysis of pro-activeness and reactiveness. *Lecture Notes in Computer Science*, 1536:350–380, 1998.
- [15] F. Patin, G. Pouzancre, and Servat T. A formal approach in the implementation of a safety system for automatic control of platform doors. In *4th AFIS Conference on System Engineering*, 2006.
- [16] F. Raimondi and A. Lomuscio. Mcmas: a tool for verifying multi-agent systems. In *12th int. conf. on tools and algorithms for the construction and analysis of system (TACAS'06)*, volume 3920 of *LNCS*. Springer Verlag, 2006.
- [17] A. Regayeg, A. H. Kacem, and M. Jmaiel. Specification and verification of multi-agent applications using temporal z. In *Intelligent Agent Technology Conf. (IAT'04)*, pages 260–266. IEEE Computer Society, 2004.
- [18] Alexis Scheuer, Olivier Simonin, and François Charpillet. Safe longitudinal platoons of vehicles without communication. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 70–75, Kobe (JP), May 2009.
- [19] Steria – Technologies de l’information. *Proof Obligations: Reference Manual, version 3.0*, 1998.
- [20] M. Wooldridge, N. R. Jennings, and D. Kinny. A methodology for agent-oriented analysis and design. In *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 69–76. Seattle, WA, USA, 1999.
- [21] Jano Yazbeck, Alexis Scheuer, Olivier Simonin, and François Charpillet. Decentralized local approach for lateral control of platoons. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Shanghai (CN), September 2011. To appear.