BiAgent-based Model for IoT Applications
Case of a Collision Avoidance System

Souad Marir, Roumeissa Kitouni, Zakaria Benzadri, Faiza Belala

Department of Software Technologies and Information Systems,
University of Constantine 2 Abdelhamid Mehri
{souadmarir94,kitouni.romaissa,benzadri}@gmail.com
faiza.belala@univ-constantine2.dz

Abstract. The Internet of Things (IoT) consists in connecting every aspect of daily and professional life to a common infrastructure, in order to improve considerably the efficiency of otherwise unthinking objects. The huge scale on which they operate, as well as the lack of adequate standards and infrastructures makes the development of IoT applications a task of gradually growing complexity. The objective of this work is to define a formal model with BiAgents (Bigraphical Agents) for IoT applications, based on a suggested generic multi-layered architecture. We show how bigraphs support the structural aspects modelisation of these applications while the agents specify their analytical and decisional aspects. We proceed then to the edition and execution of our model using the bigraph implementation tool (RCTool*4*Bigraphs), and through the exploitation of its model-checker, we formally verify its most critical property. As a practical example, we study the case of a Collision Avoidance System.

Keywords: Advanced Driver Assistance Systems, BAM4IoT, Bigraphs, BiAgents, Collision Avoidance System, Formal Specification, Internet of Things, RCTool*4*Bigraphs.

1 Introduction

The Internet of Things (IoT) is the vision of a world where each entity has a physical or virtual representation, as well as a presence on existing or future interoperable networks. These entities interact through specific protocols in order to offer and consume services. They can generally perceive their environment and affect it.

The development of IoT applications becomes each year more complex and challenging. This is due to, among others, the need of ensuring interconnectivity and supporting a huge scale of interconnected devices. An important point to consider is that this type of application relies to a great extent on shared infrastructure and protocols (the Internet in the most common case). A poorly designed application may thus not only result in poor behaviour, but also in damaging the infrastructure or hindering other applications. To fully exploit the widely recognized mathematical formalism potential in analysing, designing and implementing IoT complex systems (with which human users and different devices interact), well-defined development approaches are required. To address this issue and specifically to model the IoT applications at different levels of abstraction, a new incremental approach is proposed. In fact, to date in literature, several approaches for the development of IoT applications have been proposed. It is possible to recognise some operational approaches based on multi-agent frameworks [1] to support the implementation of IoT systems. Nevertheless, no well-formalized approach able to support analysis, design and implementation phase of IoT applications development is currently available. In this paper, we address such issue by suggesting a systematic comprehensive approach.

In the beginning, we give a generic layered architecture for IoT applications allowing a separation of concerns mastering thus their complexity. Then, this architecture constitutes an intermediate model for the formal one. This latter is based on a judicious combination of bigraph model [2] and agents, allowing to describe either aspects of an IoT application with precision as well as a high level of abstraction. The defined model may support both IoT system's structure and its behaviour. Indeed, Bigraphs give a way to represent structural aspects according to two axis: locality (place graph) and connectivity (link graph). In addition, a mechanism is provided to express the evolution of Bigraphs, called the Reaction Rules. This makes it possible to describe the behaviour and the states of a dynamically evolving system. Aside from the two previously mentioned aspects of structure and behaviour, IoT applications are also characterized by cognitive aspects, like awareness of the environment and the ability to affect it. Intelligent agents may maintain these aspects, while providing an ideal tool for the specification of communication and context aware interactions. The paper contribution is twofold, on one hand we propose a formal approach for IoT applications in order to master their complexity, on the other hand we extend the BiAgents definition given in [3] for the modelling of this kind of systems.

The rest of this paper is structured as follows. In Section 2, we make a summary of the useful concepts for the comprehension of this work's main contribution. Section 3 introduces our multi-layered generic architecture, as well as our case study, the Collision Avoidance System. Section 4 presents our proposed BiAgent-based model for IoT applications while illustrating it with a realistic example. In Section 5, we establish a synthesis of related works that use formal methods to model IoT applications. Finally, conclusion and future direction of this work are drawn.

2 Basic concepts

This section introduces the different formal concepts we use to model the IoT.

2.1 Bigraphs

In Bigraphs theory [2], a *bigraph structure* facilitates the understanding and design of complex systems. Its formal notation guarantees a safety regarding the correction of the modelling. In addition, the *reaction rules* clearly specify the dynamics of the modelled system.

A bigraph structure is the combination of two graphs (see figure 1) :

– *The place graph* is in the form of a forest of trees each having a root called a "region". It has a control function that assigns each node a control[1]. It can also contain sites that are abstractions in which we can insert other bigraphs.
– *The link graph* is used to represent relationships and connectivity in a system. It has the structure of a hypergraph.

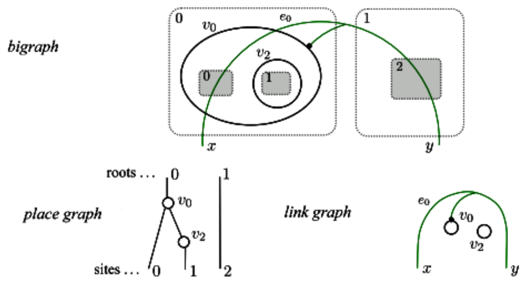


Fig. 1. Example of a bigraph [2]

Definition 1. ***A bigraph** [2] is a tuple of form : $G = (V, E, ctrl, prnt, link)$: $\langle m, X\rangle \to \langle n, Y\rangle$ where $\langle m, X\rangle$ and $\langle n, Y\rangle$ are the inner and outer faces of G. V and E are respectively, a final set of nodes and a final set of hyperedges. $ctrl$:$V \to \mathcal{K}$ is the control function which assigns a control to each node. $prnt$ represents the parent function. $link$ is the function that represents the different links contained in the bigraph.*

A reaction rule [4] has the form $R \to R'$ where R is known as the bigraph redex and R' the reactum one. If we write $B \to B'$, this means that there is a reaction rule that can be applied to B and give B', conversely if we write $B \not\to B'$, this means that there is no possible reaction rule for B to get B'.

Definition 2. ***A Bigraphical Reactive System (BRS)** is defined by the set of bigraphs representing the different states of the system obtained from the initial bigraph, and the set of reaction rules applied successively.*

2.2 BiAgents

Bigraphical Agents [1] are defined by a *physical structure* and a *logical structure*. The physical structure is modelled by the formalism of *bigraphs* and the logical one by *agents*.

[1] a number of ports

Definition 3. ***The physical structure*** *[3] of a BiAgent is defined as a tuple $\mathcal{B} = (\mathbb{B}, \mathcal{R}, \mathbb{U}, B_0, F)$ where $\mathbb{B}$ is the space of bigraphs, $\mathcal{R}$ is the set of reaction rules, $\mathbb{U}$ is the control space such as $\mathbb{U} \subseteq \mathcal{R} \times V_{\mathbb{B}}$, with $V_{\mathbb{B}}$ the set of bigraph nodes. B_0 is the initial bigraph. Before defining F, $dec3$ is a function which gives a valid decomposition of a bigraph into three bigraphs such as $dec3(B) = (B', B'', B''')$ and $B = B' \circ B'' \circ B'''$. V_R is the set of redex's nodes and $V_{R'}$ is the set of reactum's rules. F is the transition function that, from the current bigraph and from a control action, gives a new bigraph: $F : \mathbb{B} \times \mathbb{U} \to \mathbb{B}$. It is defined as follows by considering C as the context of R' and d the parameters of R' : $F(B, (R \to R', h)) = C \circ R' \circ d$ if $\exists$ $dec3$ such as $dec3(B) = (C, R, d)$ and $h \in V_R$ and $h \in V_{R'}$. If not, F is undefined.*

Definition 4. ***The logical structure*** *[3] of a BiAgent is defined as a tuple $a = (\mathcal{O}, \mathcal{U}, host_0, obs, ctr, mgrt)$ where $\mathcal{O}$ is the agent's observation space such as $\mathcal{O} \subseteq \mathbb{B}$. $\mathcal{U}$ is the control space such as $\mathcal{U} = \mathcal{R}_a \times V_{\mathbb{B}}$. $host_0 \in V_{\mathbb{B}}$ is the node that hosts the agent initially. obs is the observation function, $obs : \mathbb{B} \times V_{\mathbb{B}} \to \mathcal{O}$ ctr is the control function with which an action can be executed, $ctr : \mathcal{O} \to \mathcal{U}$. $mgrt$ is the migration function that, with the host of the agent and an observation, provides the next host, $mgrt : V_{\mathbb{B}} \times \mathcal{O} \to V_{\mathbb{B}}$.*

2.3 Trace

Definition 5. *In a Bigraphical Reactive System,* ***a trace*** *[5] is a sequence of bigraphs $\langle a1, a2, ...\rangle$ such that for each a_i and a_{i+1}, There is a reaction rule $a_i \to a_{i+1}$. If there are two traces s and t and the last element of s is the redex of a reaction rule whose reactum is the first element of t, the composite trace exists and begins with all the elements of s followed by all the elements of t, in this case t is an extension of s. We denote $Tr(A)$ the set of all traces for a given BRS A.*

We use traces to make a history of the different events that happen in the system modelled. We can note $t = B_0 \overset{\text{R1}}{\prec} B_1 \overset{\text{R2}}{\prec} B_2 \overset{\text{R3}}{\prec} B_3$ with $R1, R2$ and $R3$ the reaction rules that allow the transition from a bigraph to another. On this basis, a BiAgent can be an agent with a memory by using, as an observation space, a set of *traces* of bigraphs instead of using a set of bigraphs. If we have a system that contains x agents, we can have x traces, each one representing the course of actions of a particular agent; we call each of these traces a *projection.*

Definition 6. *We represent the flow of agents in space through* ***a projection*** *t^a of a trace t defined as follows :*

$$t^a = (p_0^a, h_0^a) \prec (p_1^a, h_1^a) \prec ...$$

where each p_i^a is a projection of the bigraph B_i Which holds only the node in which the host is located h_i^a.

3 A Layered-Architecture for IoT applications

Nowadays, we are witnessing a radical evolution of the current Internet in a network of interconnected objects that not only collects information from the environment (detection) and interacts with the physical world (action / control), but also uses existing Internet standards to provide services for information transfer, analytic, applications and communications [6]. In this context, an IoT application is the set of software and hardware that enables a smart behaviour from an ordinary object. In other words, it is a functional system whose aim is to collect data, process it and emit some output that is relevant to its intended task. Specifically, we can summarize the main characteristics of IoT applications as follows [7,8]: *Interconnectivity, Heterogeneity, Dynamic changes, Scale, Security and Connectivity.*

In the present work, the software part of an IoT application will be our main focus; we give a multi-levels architecture of the Connected Objects applications and discuss its implication for the objects communications in terms of the traffic that will be generated. Figure 2 shows the given architecture knowing that an IoT application could be specified according to four layers:

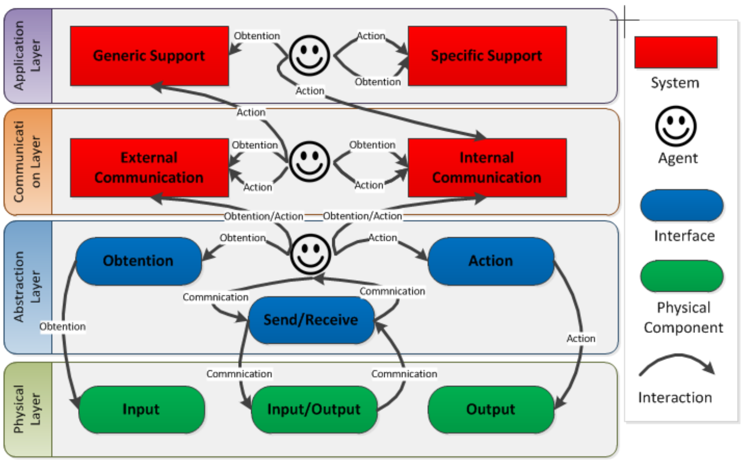


Fig. 2. Multi-layered architecture for an IoT application

Physical Layer It represents the system's hardware. In particular, we focus our interest on input and output hardware, as they are the mean to interact with and to receive signals from the environment.

Abstraction Layer It is a low-level layer which implements all interaction protocols with the components within the Physical Layer (in the form of implementable driver modules). In addition, this layer offers an interface through which the application interacts with the physical components. The main reason we choose to represent this layer is the diversity of the used hardware for this kind of systems as well as the lack of standards. We need this layer to be provided with basic intelligence to analyse input at a low level and decide where to send it; we represent this by an agent that manages this particular layer: the Abstraction agent.

Communication Layer Since communication is an essential part of every IoT system, we choose to dedicate a separate layer to it. This layer is divided into two sublayers: Internal Communication (ICL), which is the set of protocols used for interaction between all layers within a system, and External Communication (ECL), which is the set of protocols used to communicate with separate, external systems. The intelligent entity managing this layer is the Communication agent, which verifies the correctness of package formats and sends them to the right destination.

Application Layer This most high-level layer is the actual IoT application. It is again divided into two sublayers: Generic Support for all non-functional aspects which are common to a large spectrum of applications, and Specific Support which is the handling of the most particular aspects of a system. Decisions on the global behaviour of the application are ultimately issued by this layer's smart entity: the Application agent.

Each of the previously mentioned agents are conceived according to a control-loop model as shown in figure 3.

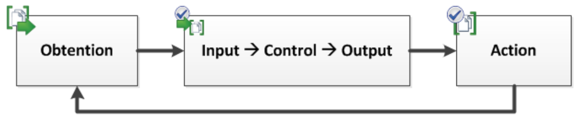


Fig. 3. Principle of an agent control loop

Illustrative Example

The case we will study in this section is that of a Collision Avoidance System (CAS) [9] as implemented in a car. It can detect a stationary natural obstacle and issue a warning to the driver and to an external server so as to warn those who might take the same road. It can also detect a moving obstacle (like a pedestrian) or receive a warning from an external server; in either of these cases, it will simply produce a warning for the driver.

By applying the proposed multi-layered architecture (of figure 2) to the CAS example of this section, we obtain the structure in figure 4;

– In the Physical Layer, one radar and two cameras (one in the front, one in the rear) are used to get raw information on the environment. A vibrator and speakers are used to issue warnings to the driver. A network card is used to communicate with other similar systems through a trusted server. To each kind of these components, an interface in the Abstraction Layer is associated.
– In the Communication Layer, we specify the subsystems that constitute the two sublayers. The verification subsystem in the ECL checks whether a received package is from a trusted server or needs to be destroyed.

– The Application layer was extended with all functional and non-functional high-level software components that specify the actual application behaviour.

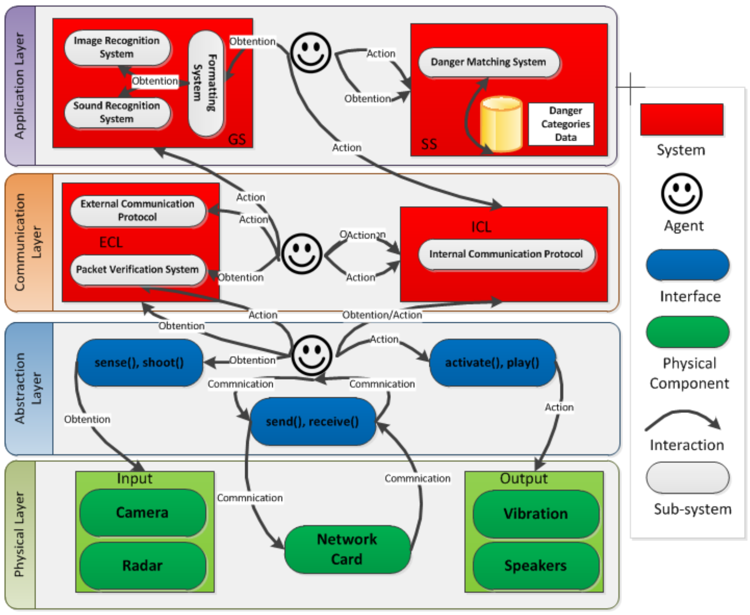


Fig. 4. Multi-layered Architecture applied to the Collision Avoidance System

The major components of our proposed architecture for CAS example are used for an easy management and development of this application by both users and programmers. The re-usability and genericity of this approach allow the designer to generate models for any IoT application. In the following we describe the choices that have been taken to satisfy these requirements; we are interested by the formal description and verification issues. Particularly, we show how we generate a formal definition of IoT applications, based on the theory of bigraphs.

4 BAM4IoT: A BiAgent-based model for IoT

The generic architecture presented earlier represents an intermediate phase towards the development of an appropriate formal model based on BiAgents [3]. Indeed, our model may have two distinct but complementary views: the physical view (the layers) that define the place,the connectivity of components of an IoT application and the logical view (the agents) that manages these layers by obtaining information, analysing it and take an action according to the given information. We detail in the following sections the proposed Bigraphical Agents Model for the IoT (BAM4IoT) according to its two structures (*physical structure* and *logical structure*).

Definition 7 (BAM4IoT Physical Structure). *The physical structure of BAM4IoT model is defined as following :*

$$\mathcal{B}_{IoT} = (\mathbb{B}, \mathcal{R}, \mathbb{U}, B_0, F)$$

- $\mathbb{B}$ *is the set of bigraphs modelling the chosen IoT application.*
- $\mathcal{R}$ *is the set of reaction rules describing its behaviour.*
- $\mathbb{U} \subset V_{\mathbb{B}} \times \mathcal{R}$ *is the set of controls representing the potential actions recording to a single node with* $V_{\mathbb{B}}$ *the set of nodes of every* $B \in \mathbb{B}$.
- $B_0 \in \mathbb{B}$ *is the bigraph representing the initial state of the IoT system modelled.*
- F *is the control function defining the transition between a bigraph and another according to a control* $u_i \in \mathbb{U}$.

Definition 8 (BAM4IoT Logical Structure). *The logical structure of BAM4IoT model is defined by a set of adaptive agents aIoT , each agent a(i)IoT has the following format:*

$$a^{(i)}_{IoT} = (\mathcal{O}, \mathcal{U}, \mathcal{D}, host_0, obs, an, ctr, mgrt)$$

with :

- $\mathcal{O} \subset \mathbb{B}$ *is the observation space.*
- $\mathcal{U}$ *is the control space which represents the possible agent's actions.*
- $\mathcal{D}$ *is the decision space obtained after the agent's analysis.*
- $host_0$ *is the agent's initial host.*
- *the function of observation obs which provides an observationo* $\in \mathcal{O}$ *using a bigraph and the host of the observant agent :* $obs(b, h) = o$.
- *the analysis function that, with an observation or a set of observations and a host, analyses this host or its sons and returns a positive or negative decision* :$an(o, h) = \alpha \in \mathcal{D}$.
- *The control function which gives the next succession of rules to be executed, each according to a node, using the result of an analysis :* $ctr(\alpha) = u \in \mathcal{U}$.
- *The migration function that provides the next host of the agent according to the current host and an observation* $mgrt(o, h) = h'$.

CAS Example Application

Let us take again the CAS example (Collision Avoidance System) and try to define their structures (physical and logical), as well as its behaviour while illustrating the BAM4IoT definitions.

The Physical Structure of CAS example is given by the tuple

$$\mathcal{B}_{CAS} = (\mathbb{B}, \mathcal{R}, \mathbb{U}, B_0, F)$$

- $\mathbb{B} = \{B_0, ..., B_{20}\}$ is the set of all bigraphs resulting from the application of the defined reaction rules.

- $\mathcal{R} = \{R_0, R_1, R_2, R_3, R_4, R_5, R_6, R_7\}$. A summary of these reaction rules is given in table 1.
- $V_{\mathbb{B}}$ the set of nodes of every $B \in \mathbb{B}$ with $V_{\mathbb{B}} = V_{SS} \uplus V_I \uplus V_F \uplus V_E \uplus V_{ES} \uplus V_S$.
- $\mathbb{U}$ the set of every possible couple made of reaction rules and nodes : $\mathcal{R} \times V_{\mathbb{B}}$.
- $B_0 \in \mathbb{B}$ is the bigraph representing the system's initial state B_{SDC} (fig. 5).
- F is the control function that defines the transition from a bigraph to another through a particular control. If a rule isn't applicable to the node to which it is associated in the control $u_i \in \mathbb{U}$, F is undefined.

Table 1. Collision Avoidance System Reaction Rules

Rule ID	Description
R0	A sensor device (radar) determines the presence of an input (pedestrian).
R1	The information is sent to the ICL (Internal Communication Layer) in the purpose of beeing formatted.
R2	The information is formatted according to the Internal Communication Protocols (ICP) and its path is traced.
R3	The formatted information is transmitted to the system (GS).
R5	The system looks for internal data on known dangers that may correspond to the extracted information.
R6	An action corresponding to the detected danger is requested.
R7	The action is transmitted to the Internal Communication System (ICS) to be carried out on the Current System (CS).

The Logical Structure of CAS example is given by a set of three agents: AppAg, ComAg , AbsAg , each one manages changes of the corresponding root (layer) [10]. For lack of space, we explain only the following agent, which is judged the more relevant (see [10] for more details):

$$App^{Ag} = (\mathcal{O}^{App^{Ag}}, \mathcal{U}^{App^{Ag}}, \mathcal{D}^{App^{Ag}}, host_0^{App^{Ag}}, obs^{App^{Ag}}, an^{App^{Ag}}, ctr^{App^{Ag}}, mgrt^{App^{Ag}})$$

with:

- $\mathcal{O}^{App^{Ag}} = \{B_6, B_8, B_{10}\}$
- $\mathcal{U}^{App^{Ag}} = \{(ActionApp, IE), (ActionCom, IE)\}$
- $\mathcal{D}^{App^{Ag}} = \{\alpha_1^{App}\}$
- $host_0^{App^{Ag}} = 2$
- $obs^{App^{Ag}}(B, h) = \begin{cases} obs_1^{App} & \text{if } B = B_6 \text{ and } h = 2 \\ obs_2^{App} & \text{if } B = B_8 \text{ and } h = FA \\ obs_3^{App} & \text{if } B = B_8 \text{ and } h = FS \\ obs_4^{App} & \text{if } B = B_8 \text{ and } h = GS \\ obs_5^{App} & \text{if } B = B_{10} \text{ and } h = IE \end{cases}$
- $an^{App^{Ag}}(o, h) = \alpha_1^{App}$ if $o = \{obs_1^{App}, obs_2^{App}\}$ and $h = GS$
- $ctr^{App^{Ag}}(\alpha) = \{(ActionApp, IE), (ActionCom, IE)\}$ if $\alpha = \alpha_1^{App}$

$$- \ mgrt^{App^{Ag}}(o,h) = \begin{cases} FA & \text{if } o = obs_1^{App} \text{ and } h = 2 \\ FS & \text{if } o = obs_2^{App} \text{ and } h = FA \\ GS & \text{if } o = obs_3^{App} \text{ and } h = FS \\ IE & \text{if } o = obs_4^{App} \text{ and } h = GS \\ 2 & \text{if } o = obs_5^{App} \text{ and } h = IE \end{cases}$$

In addition to this definition, we can visualize a system's behaviour through the use of traces and projections. This will illustrate all state changes that happen alternatively to the physical structure (CAS reaction rules) and to the logical structure (an agent's migration). Describing an application through the use of this extended model serves the main purpose of preventing non-deterministic behaviour. As a matter of fact, there is no way to guarantee a consistent execution path with Bigraphical Reactive Systems alone. A reaction rule's applicability is decided through a pattern matching and could be applicable without it being semantically appropriate. The control function on the other hand always associates to a given analysis (of the current state) an applicable and semantically appropriate set of actions. Figure 5 shows a scenario of detection of pedestrian. After applying CAS reaction rules (R_0 to R_7), the system arrives at a final state representing the action performed.

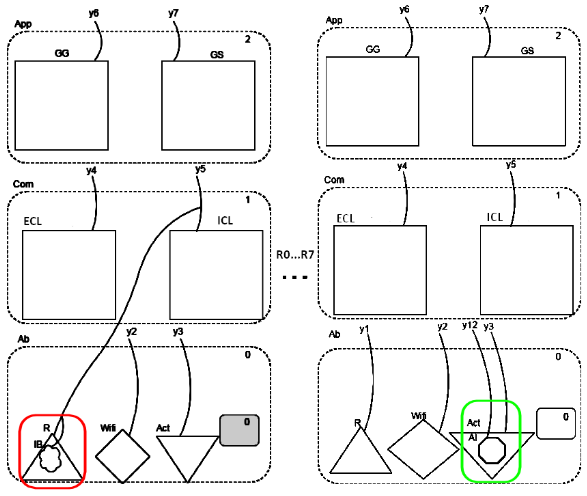
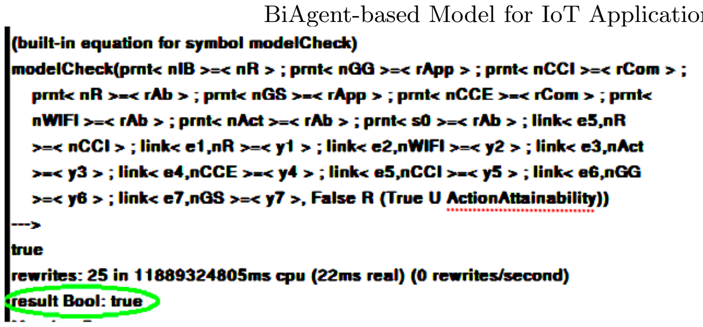


Fig. 5. The BAM4IoT of Collision Avoidance System B_{CAS}

For validation purpose, we exploit the RCTtool4Bigraphs (an extended and generic version of a tool that has been already proposed in [11] and [12]) to check that our BAM4IoT specification satisfies an important property. Figure 6 shows model checking results of the *Action-Attainability* property; we conclude that the entered specification verifies the desired property.



```
(built-in equation for symbol modelCheck)
modelCheck(prnt< nIB >=< nR > ; prnt< nGG >=< rApp > ; prnt< nCCI >=< rCom > ;
   prnt< nR >=< rAb > ; prnt< nGS >=< rApp > ; prnt< nCCE >=< rCom > ; prnt<
   nWIFI >=< rAb > ; prnt< nAct >=< rAb > ; prnt< s0 >=< rAb > ; link< e5,nR
   >=< nCCI > ; link< e1,nR >=< y1 > ; link< e2,nWIFI >=< y2 > ; link< e3,nAct
   >=< y3 > ; link< e4,nCCE >=< y4 > ; link< e5,nCCI >=< y5 > ; link< e6,nGG
   >=< y6 > ; link< e7,nGS >=< y7 >, False R (True U ActionAttainability))
--->
true
rewrites: 25 in 11889324805ms cpu (22ms real) (0 rewrites/second)
result Bool: true
```

Fig. 6. Model Checking results of the Action-Attainability property.

5 Related Work

In their paper about high-level application development in IoT [13], authors mention three main approaches for developing IoT applications: Middleware, Programming framework and Model-Driven Development (MDD). The latter, which is most relevant for our current work, consists of describing the applications as high-level abstract models, and then using these models according to some research ways (code generation, formal analysis, etc.).

Few works in the literature have focused on the use of formal methods to design and specify such applications by supporting the characteristics of complex, distributed and auto-adaptive systems. For instance, authors of [1] based their framework for the IoT on agents, focusing on self-adaptive and self-organizing aspects of the applications. On the other hand, graph-based formalisms are usually favoured to support the description of interrelation as well as dynamic configuration. The work cited in [14] tried to combine the advantages of both formalisms and proposed a hybrid one using complex networks, a graph-based formalism, as well as agent-based modelling for the IoT. The idea of creating a formal model out of many different methods stems from their individual failure to keep up with the ever-growing complexity of modern systems, and the decreasing accuracy with which they can describe IoT applications.

In the same thought, our proposed approach combines: (1) Bigraphs to describe geographical dispersion, connectivity and dynamic changes; and (2) Agents to model communications and context-aware interactions in IoT applications. In particular, the proposed formal model (BAM4IoT) gives an unified way to describe IoT applications so as to result in a deterministic, understandable specification. Moreover, we used a practical tool (RCTool4Bigraphs [11], [12]) to verify an IoT-related property (*Action-Attainability property*).

6 Conclusion

A recurring challenge in the development of IoT applications is managing their complexity. To overcome this, we have set ourselves the main objective of applying one of the most appropriate mathematical formalisms. Our attention has

turned to the bigraphs and the agents. A judicious combination of these two formalisms allowed to consider the two aspects inherent to this type of applications; the associated *physical object/virtual intelligence* pair.
We have chosen to develop first, a layered architecture to simplify the description of IoT applications and its instantiation according to a case study: Collision Avoidance System (CAS). Then, we were able to validate the physical structure of the extended formal model BAM**4**IoT with a model-checking tool of bigraphs (RCTool*4*Bigraphs).
We propose, in the future works, to give our model a way for objects (agents) to communicate by extending the definition of the agents in BAM**4**IoT with a suitable function. Obviously, a tool around the bigraphs that supports the virtual or logical aspect of the systems is an unavoidable solution.

References

1. do Nascimento, N.M., de Lucena, C.J.P.: Fiot: An agent-based framework for self-adaptive and self-organizing applications based on the internet of things. Information Sciences **378** (2017)
2. Milner, R.: The space and motion of communicating agents. Cambridge University Press (2009)
3. Pereira, E., Kirsch, C., Sengupta, R.: Biagents–a bigraphical agent model for structure-aware computation. Cyber-Physical Cloud Computing Working Papers, CPCC Berkeley (2012)
4. Perrone, G.: Domain-Specific Modelling Languages in Bigraphs. PhD thesis, IT University of Copenhagen Copenhagen, Denmark (2013)
5. Perrone, G., Debois, S., Hildebrandt, T.: Bigraphical refinement. arXiv preprint arXiv:1106.4091 (2011)
6. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of things (iot): A vision, architectural elements, and future directions. Future generation computer systems **29**(7) (2013)
7. Patel, K., Patel, S.: Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges. IJESC (2016)
8. Vermesan, O., Friess, P.: Internet of Things – From Research and Innovation to Market Deployment. River Publishers (2014)
9. : Advanced Driver Assistance Systems (ADAS) — TOSHIBA Storage & Electronic Devices Solutions Company. (2017)
10. Marir, S., Kitouni, R.: Combinaison des agents et des bigraphes pour la modélisation des applications iot. Master's thesis, Université Constantine 2 Abdelhamid Mehri (2017)
11. Benzadri, Z.: Spécification et Vérification Formelle des Systèmes Cloud. Thesis, Université Constantine 2 - Abdelhamid Mehri (October 2016)
12. Benzadri, Z., Bouanaka, C., Belala, F.: Big-caf: a bigraphical-generic cloud architecture framework. International Journal of Grid and Utility Computing **-**(-) (2016) –
13. Patel, P., Cassou, D.: Enabling high-level application development for the internet of things. Journal of Systems and Software **103** (2015)
14. Batool, K., Niazi, M.A.: Modeling the internet of things: a hybrid modeling approach using complex networks and agent-based models. Complex Adaptive Systems Modeling **5**(1) (Mar 2017)