A Multi-layered Bigraphical Modelling Approach for Context-Aware Systems

Ahmed Taki Eddine Dib, Ramdane Maamri

PII: S1319-1578(21)00211-1
DOI: https://doi.org/10.1016/j.jksuci.2021.08.008
Reference: JKSUCI 1081

To appear in: *Journal of King Saud University - Computer and Information Sciences*

Received Date: 9 May 2021
Revised Date: 5 August 2021
Accepted Date: 8 August 2021

A Multi-layered Bigraphical Modelling Approach for Context-Aware Systems

Ahmed Taki Eddine DIB

LIRE LABORATORY, UNIVERSITY OF CONSTANTINE II-ABDELHAMID MEHRI, ALGERIA

Ahmed.dib@univ-constantine2.dz

Ramdane MAAMRI,

LIRE LABORATORY, UNIVERSITY OF CONSTANTINE II-ABDELHAMID MEHRI, ALGERIA

RAMDANE.MAAMRI@UNIV-CONSTANTINE2.DZ

A Multi-layered Bigraphical Modelling Approach for Context-Aware Systems

Abstract

There exist several approaches proposed for building Context-Aware Systems (CAS). However, due to the continually changing environment, the large number of interrelated components, complexity and diversity of application domains make the modelling of context-aware systems a particularly challenging task. To address the increasing complexity of the modelling: i) It is critical to take into account the importance of the environment (operational context); and ii) rely on software engineering concepts such as abstraction and modularity in order to reduce the level of complexity. Also, a context-aware system may require intelligence and autonomy. These naturally lead us to apply intelligent agent-based engineering. This work introduces a formal layered design approach that combines intelligent control of multi-agent systems and bigraph's rigor to model context-aware systems. Bigraphical reactive systems are particularly compelling for their capacity to specify, simultaneously, the physical and logical distribution of system components and their interconnections using two distinct structures; that is: place graph and link graph. While the behaviour and dynamic evolution are expressed using defined reaction rules, and as a last step, the bigraph specifications are coded in the Maude language to allow their execution and the verification of their validity.

Keywords: Context-aware systems; Computing methodologies; Formal modelling; Multi-agent systems

Introduction

In today's world, context-aware systems (CAS) are everywhere, offering a diverse range of computing capabilities and services at any time and in any place. Applications should be aware of their environment and automatically adapt to evolving situations in order to provide adequate service to users. This is known as context awareness. Any detail that can be used to describe an entity's condition is referred to as context. An entity may be an individual, position, or thing that is considered important to a user's interaction with an application, such as: their location; time; activities; and preferences. If a system can adapt its behaviour to the actual context of use, it is context-aware. The most important purpose of such systems is to provide adequate services to the specific individuals, place, time, and event (Dey, 2001). The modelling and implementation of CAS in ubiquitous and pervasive environments poses new challenges that go way beyond conventional design and development methods. Context-aware systems may require a diverse set of resources that perform random tasks using pre-determined or ad-hoc communication protocols. These systems must have flexible and adaptive architectures to deal with continuous emerging requirements of the environment and users. There exist many formal and semi-formal modelling approaches proposed in the literature. However, these approaches do not address the environment as a modelling dimension at the design phase. This results in a pre-established often-rigid solution where interactions, users, and changes are fixed. Besides, they lack techniques and tools to validate the resulting design.

In this work, we consider a context-aware system as a multi-layered cyber-physical system. The modelling of context-aware systems considers general systems theory concepts, namely hierarchy, and emergence. Also, a context-aware system requires intelligence and autonomy. These naturally lead us to apply agent-based engineering. So, context-aware systems require these components:
- A solid concepts layers (context and context-aware system).
- Proactivity and adaptability to facilitate the sharing, management, and inference of a given context.
- Formal basis to ensure the robustness and the correctness of the system.

Formal modelling becomes essential in the development of computer systems in general. Relying heavily on mathematical definitions, formal methods may be applied in this context to provide a precise model for context-aware systems and ensuring the high evaluation assurance level 'EAL7' according to Common Criteria Standard (Common Criteria, 2021). Besides, it helps for the early integration of verification at the design phase, facilitating the early detection of the design defects. Further, formal methods are endowed with powerful tools used to automate various stages of the verification.

In this work, we propose a modelling approach that combines intelligent control of multi-agent systems and bigraph's rigor to model context-aware systems. Bigraphs are particularly appealing for their ability to specify, at the same time, the physical and logical distribution of system components and their interconnections using two distinct structures, namely the place and link graphs. While the behaviour and the dynamic evolution are expressed using a set of defined reaction rules. Indeed, BRS have shown their adequacy in several domain applications, such as context-aware systems (Sahnoun et al., 2017), networking (Boucebsi and Belala, 2020), Fog computing (Benzadri et al., 2021), Further, we rely on the Maude system to enable the autonomic executability and validation of the resulted specifications (Benzadri et al., 2021). This approach gathers the features presented above and shows how this model may be used in different scenarios and applications. The main features of the model are validated by implemented application.

Section 2 describes our general approach to the issue, while Section 3 describes the system's architecture, the implementation of context awareness, and the structure and behaviour. Section 4 lists the proof-of-concept applications and experiments that are parts of evaluating and validating the modelled approach. The fifth section summarizes and compares similar fieldwork. The final section ends with some suggestions for future research.

2- Our approach principle

A Context-aware system (CAS) is considered as a multi-layered cyber-physical system that focuses on explicit interaction and communication between the user, the system, and the external environment, employing a number of modalities. A CAS is composed of all of the system's components as well as the surrounding environment (sensors, actuators, intelligent appliances, smartphones and tablets, workstations, and servers) where devices are interconnected through a ubiquitous network that uses various protocols and mostly wireless transmission. The data is transferred from layer to layer in a uniform format employing standard protocols. The applicative layer offers the context-aware system, intelligent decision-making process, and interaction via a standard interface supporting communication between different entities. Therefore, the modelling of CAS must consider some additional aspects compared to classical software systems, making such systems hard to model and manage through time. Our goal is to make the models we create more useful i) by proposing a modelling approach to ease the process of model construction by bringing the solution that we offer closer to its respective domain ii) by relying on advanced software engineering concepts such as abstraction and modularity in order to reduce the level of complexity. Applying these principles broadens the scope of systems engineering beyond classical solutions by identifying common properties of hierarchical levels. In addition, to the SE concepts cited above. To address the increasing complexity of the modelling. **It is important to take into account the context as modelling dimension**. Further, it should explicitly and concisely describe operating environments. The environment layer represents the operational context in which the designed system is used. It is critical to achieving the desired goals. Advanced system performance is measured by understanding how both component reliability and component interactions are influenced by the conditions, settings, and circumstances in which they operate. The environment layer is a predominant endogenous modelling dimension in our approach. Building on this, the interface layer plays an important role between the environment layer and the software and hardware layers. It deals with both the operational context and system users by ensuring information transfer and allowing for a bidirectional causality. The information (data) is offered to the other layers in a uniform format and employing standard protocols. Engineered solutions within real-world systems involve working with decision-makers. The software layer plays an important role. First, it provides reactive behaviour through a domain-specific process for nominal user requests. Second, it provides a proactive action as a response for operational context change occurrences. This, independently from any human activity, promoting autonomy and preventing component reliability and interactions. It finally acts on the environment through physical effectors (the hardware layer). The aforementioned layer is built on agent technology as a building block. Where, Agents provide autonomy, reactivity, and proactivity, introducing intelligent behaviour by allowing anticipation. The interaction of the autonomous and cognitive agents collaborates as part of a multi-agent system. This fits well with CAS properties, like Context awareness, distribution, collaboration, and adaptation.

This study aims to create a bigraphical, multi-agent-based model to describe the configuration and behaviour of a context-aware system based on a collection of factors such as the system's user, the environment, the application, and the hardware. These parameters are intertwined and can affect one another. As a result, the system must adjust to its operational context to achieve its end objective. The system would satisfy pre-defined and desired requirements such as context-awareness, adaptation, usability, reliability, and robustness. The modelling of context-aware systems considering general systems theory concepts, namely hierarchy, and emergence. Which assumes a context-aware system as a multi-layered cyber-physical system composed of the factors cited above (the user of the system, the environment, the software, and the hardware). These factors represent the layers of our CAS. Where each layer contains several sublayers that constitute the local context of the actual layer. Besides, we introduce a second modelling dimension that operates across the sublayers of the main layers (the user of the system, the environment, the application, and hardware). This interleaving of layers and sublayers forms a micro/meso level of modelling (see Fig. 1.), representing a local context while the overall represents the global context of the system.

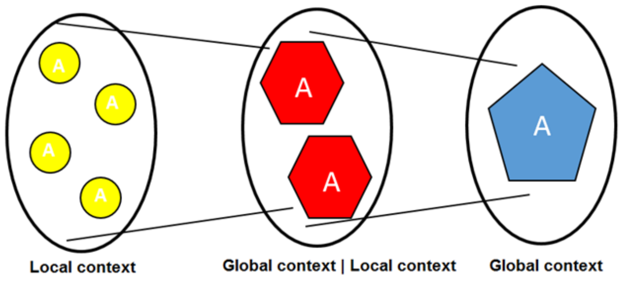


Figure 1. Micro/meso level of modelling

3- Formal specification of a context-aware system

The specification of the CAS system is achieved by the definition of the structure and behaviour of the layers described in section 2 (see Fig. 2.). The specification of the context-aware system we build provides mechanisms to manage the specific information of the global context in a hierarchical manner. The proposed approach is structured as follows; Each layer that constitutes our context-aware system is modelled as a bigraph. Bigraphical Reactive Systems were introduced by Milner (Milner, 2008) to provide a graphical, intuitive formal model capable of representing at the same time connectivity and locality concepts of distributed and highly dynamic systems such as Context-Aware System. Moreover, they provide a unification of existing process calculi for concurrency and mobility (such as π-calculus, Petri nets, λ calculus, and so on) (Milner, 2008).

Definition 1: A bigraph is formally defined by:

$$G = (V,E,ctrl,G^P,G^L):I \to J, I = \ < m,x > ,J = < n,y >$$

Where: V and E represent finite sets of nodes and edges respectively, ctrl : V → K ctrl: V→K is a control map that assigns control to each node. The signature K is a set of controls. $G^P = (V, ctrl, prnt):m \to n$ is the place graph of G which represents the forest where $prnt : m \uplus V \to V \uplus n$ is the acyclic parent map associating with each node its hierarchical parent. m and n are both finite ordinal numbers that represent sites and regions respectively, $G^L = (V, E, ctrl, link):X \to Y$ is the link graph of G which represents the hypergraph where $link : X \uplus P \to E \uplus Y$is the link map, X and Y represent respectively inner names and outer names, and P is the set of ports of G. The interfaces $I = \langle m,X \rangle$ and $J = \langle n, Y \rangle$ represent respectively the inner and outer faces of the bigraph G.

The structural evolution of each layer is done using the composition of bigraphs. The CAS overall structure is the result of the juxtaposition of the bigraphs of the different layers. While the dynamic evolution and system behaviour is modelled using bigraphical reaction rules.

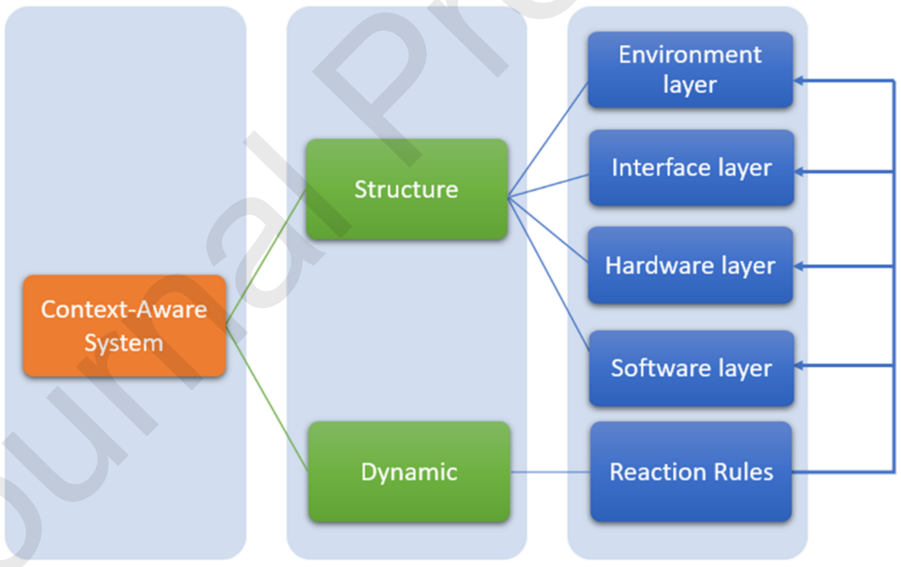


Figure 2. Context-aware system modelling

3.1 Bigraphical specification of a context-aware system structure

3.1.1 Bigraphical specification: environment layer

We start our specification with the environment layer. This layer represents a significant source of information. It is also shared and substituted in various heterogeneous systems like services, sensors, etc. To encode the environment layer, we use Milner's standard bigraph definition. Each node represents a specific entity of our environment (operational context), while edges and hyperedges represent the connections between these entities. This choice is driven by the genericity (abstraction) and the expressiveness of this conventional definition to encode many aspects of context-aware systems application domains and features. Our context model is based on the aim to model a specific set of upper-level entities that can help in adding specific concepts in different application domains. In this work, we consider the context representation as Illustrations with high-level concepts (Dey, 2001). For that purpose, we provide a mapping approach based on a domain-specific ontology. This choice is motivated by the fact that an ontology is a systematic representation of the information within a specific domain by a set of concepts and the relationships between those concepts (Lüddecke et al., 2014). Besides, there is an ascendant number of practical ontologies for CAS operational context like Context-Driven Adaptation of Mobile Services (CoDAMoS) (Preuveneers et al., 2004), Smart Space (Qin et al., 2007), and CACOnt (Xu et al., 2013). Further, we provide flexibility for the CAS designer to define the bigraph representing the environment layer (operational context) by **i)** relying on an existing ontology or **ii)** defining by extension the concepts of the current environment (see Table 1).

Table 1. **Mapping** between context-aware system, bigraph, and ontology concepts.

Context-aware system	**Formally**	**Conceptually**
Environment layer	Bigraph	Ontology
Component/parts	Nodes	Concepts
Interconnection	Edges, hyper-edges	Links

Definition 2: formally the environment layer is defined by the bigraph G^{ENV}.

$$\mathrm{G}^{\mathrm{ENV}} = \left(\mathrm{V}^{\mathrm{ENV}},\mathrm{E}^{\mathrm{ENV}}, \mathrm{ctrl},\mathrm{G}_{\mathrm{P}}^{\mathrm{ENV}},\mathrm{G}_{\mathrm{L}}^{\mathrm{ENV}}\right): \mathrm{I}\rightarrow\mathrm{N} \qquad \text{Where:}$$

- V^{ENV} represents the set of entities. These entities constitute the operational context of a specific domain; they can be virtual like (virtual machines, services, etc.), or physical parts such as (home architectural elements, roads, rails, sensors, etc.);
- E^{ENV} represents the set of relations between the concepts defined by the set of entities V^{ENV};
- $\mathrm{G}_{\mathrm{P}}^{\mathrm{ENV}}$ and $\mathrm{G}_{\mathrm{L}}^{\mathrm{ENV}}$ represent the topology and the interconnections between the concepts of the operational context.
- $I = \langle m,X\rangle$ and $J = \langle n, Y\rangle$ are the interfaces of the operational context and the context-aware system other layers.

3.1.2 Bigraphical specification: interface layer

After defining the environment layer, we now turn to the second step to make our context-aware system. To be precise, we describe the interface layer. It constitutes the information transfer interface between the user, the system (application and hardware), and the operational context. As promoted in the new generation cybernetic system wave, the user and the operational context are constituent parts within a system. There is a need to relax traditional assumptions of the boundary between the system and its operational context, making it merely a prolongation of the context-aware system. The need to better account for the importance of context is a crucial enabler for this strategic shift. The interface layer offers communication mechanisms and strategies to collect information from multiple sources, make it available to operators and decision-makers. The implementation of the interface layer is done through a linking (λ) that represents a node-free bigraph with no places. Linkings are generated using two primary bigraphical forms: elementary *substitutions* $^{y}/_{X}$ and elementary *closures* $/_{x}$.

Definition 3: the interface layer represented by linking (λ) is a map from inner names to outer names and edges to and from the environment layer bigraph, the user and the hardware and application layers. This is generated by the composition, product, and identities of the bigraphs of the different layers. The interface layer is a linking λ, where a substitution σ from X to Y is a tensor product of elementary substitutions.

$$\sigma \overset{\text{def}}{=} y_0/X_0 \otimes \cdots \otimes y_{n-1}/X_{n-1}, \text{ where } X = X_0 \uplus \cdots \uplus X_{n-1} \text{ and } Y = \{\vec{y}\}$$

First, the interface layer plays a key role as a bidirectional communication medium between the different context-aware systems parts (from the local to the global context and vice versa). Secondly, it provides a minimal common set (inner names and outer names) to construct and evolve the bigraphical context-aware system specification. Finally, through bigraphical composition and juxtaposition, making significant bigraphs from smaller ones.

3.1.3 Bigraphical specification: hardware layer

The hardware layer represents the physical parts of the context-aware system following the traditional view (i.e., excluding the operational context physical parts). It is defined as the containers (servers, terminals, smartphones, tablets, etc.), effectors, sensors, and wired communication medium. Effectors and sensors can be electronic and mechanical. The evolution in the hardware layer is a predominant factor. These parts can evolve through time (deployment of new devices) or due to change in technology. At the design time, the system is represented by a bigraph $\mathrm{G^H}$.

Definition 4: formally the bigraph $\mathrm{G^H}$ is defined as follows,

$$\mathrm{G^H} = \left(\mathrm{V^H},\mathrm{E^H}, \mathrm{ctrl},\mathrm{G^H_P},\mathrm{G^H_L}\right)\colon \mathrm{N}\to\mathrm{Z} \ \text{ where:}$$

- $\mathrm{V^H}$ is the set of devices of the CAS.
- $\mathrm{E^H}$ is the set of edges connecting the different devices.
- G_P^H the place graph represents the topology (physical distribution) and location while G_L^H represents the interconnection schema of the devices of our context-aware system.
- The interfaces $N = \langle m,X\rangle$ and $Z = \langle n, Y\rangle$represent the means of communication with the other layers by putting one bigraph on top of another (composition) or side-by-side (juxtaposition).

3.1.4 Bigraphical specification: Software layer

The software layer plays an important role. First, it provides reactive behaviour through a domain-specific process for nominal user requests. Second, it provides a proactive action as a response for operational context change occurrences. It finally acts on the environment through physical effectors (the hardware layer). The aforementioned layer may be built on agent technology as a building block (Dib et al., 2016) but not necessarily; It can possibly be based on cloud or fog computing services. Agents provide autonomy, reactivity, and proactivity, introducing intelligent behaviour by allowing anticipation. The interaction of the autonomous and cognitive agents collaborates as part of a multi-agent system. This fits well with CAS properties, like Context awareness, distribution, collaboration, and adaptation.

Definition 5: A bigraph G^A modelling the software layer of our context-aware system over a signature K takes the form:

$$\mathrm{G^A} = \left(\mathrm{V^A},\mathrm{E^A},\mathrm{ctrl^A},\mathrm{G^A_P},\mathrm{G^A_L}\right)\colon \mathrm{Z}\to\mathrm{J} \qquad \text{Where}$$

V^A is the set of abstract bigraphical nodes. Each node represents a computational entity (i.e., an object, a web service, or an agent). The set of nodes V^A is identified over a signature K using a new control map $\mathrm{ctrl}^A\colon V^A\to K$, assigning to each node $v_i \in V^A$ a control $k \in K$ with $P^A = \left\{(v \mid i) \,\middle|\, i \in ar\left(ctrl^A(v)\right)\right\}$ is the set of ports. Node's ports are communication points between the local context (computational entity) and the global context (hardware and the operational context layers). This abstraction allows the designer to choose the desired paradigm (oriented object, service-based or agent-oriented). E^A is the set of links between the computational entities $v_i \in V^A$. The semantic associated with the nodes and links are defined using bigraphical sorting.

Definition 6: formally, a bigraph G modelling a context-aware system is defined by

$G \overset{\text{def}}{=} \mathrm{G}^{\mathrm{A}} \circ \mathrm{G}^{\mathrm{H}} \circ \mathrm{G}^{\mathrm{ENV}}$ where $\mathrm{G}^{\mathrm{A}}, \mathrm{G}^{\mathrm{H}}$ and $\mathrm{G}^{\mathrm{ENV}}$are respectively the abstract bigraphs that continue our context-aware system layers. Where $\mathrm{G}^{\mathrm{ENV}} : I \rightarrow N,\ \mathrm{G}^{\mathrm{H}} : N \rightarrow Z$ *and* $\mathrm{G}^{\mathrm{A}} : Z \rightarrow J$ are bigraphs with $|\mathrm{G}^{\mathrm{A}}| \# |\mathrm{G}^{\mathrm{H}}| \# |\mathrm{G}^{\mathrm{ENV}}|$ are disjoints. **The context-aware system (bigraph *G*) is the result of the juxtaposition of the bigraphs of the different layers by layering one bigraph on the top of another. The bigraph *G* is defined by:**

$$\mathrm{G}^{\mathrm{A}} \circ \mathrm{G}^{\mathrm{H}} \circ \mathrm{G}^{\mathrm{ENV}} \overset{\text{def}}{=} \langle \mathrm{G}_{\mathrm{P}}^{\mathrm{A}} \circ \mathrm{G}_{\mathrm{P}}^{\mathrm{H}} \circ \mathrm{G}_{\mathrm{P}}^{\mathrm{ENV}}, \mathrm{G}_{\mathrm{L}}^{\mathrm{A}} \circ \mathrm{G}_{\mathrm{L}}^{\mathrm{H}} \circ \mathrm{G}_{\mathrm{L}}^{\mathrm{ENV}} \rangle : I \rightarrow J$$

We define an extended signature K defined by a set of elements called controls. For each control, the signature provides a finite set of ports and a finite set of attributes, where: port (K) denotes the control ports' names, and attribute (K) denotes the attributes of control. An attribute takes the form <name, type, value>; attributes are useful for describing the CAS devices. It can help to specify many device's features. For example, <Role, Group, identifier>, <protocol, specification, null>, etc. Based on these attributes and using tools, we can perform analysis and some manipulations. Finally, a signature K determines atomic and non-atomic controls and which control is active or passive. The sorting discipline $\sum = \{\Theta, K, \Phi\}$ over the abstract bigraphs $\mathrm{G}^{\mathrm{A}}, \mathrm{G}^{\mathrm{H}}$ *and* $\mathrm{G}^{\mathrm{ENV}}$ is a tree attribute structure where Φ is a set of formation rules, K is a signature and Θ is a set of sorts on places and links. Since we define a domain-specific modelling language for formal specification of context-aware systems. We adopt a sorting discipline over the abstract bigraphs $\mathrm{G}^{\mathrm{A}}, \mathrm{G}^{\mathrm{H}}$ *and* $\mathrm{G}^{\mathrm{ENV}}$. Such a discipline imposes structural constraints that make the bigraphs represent the model's formal entities. Example: bigraphical sorting of Petri nets (Milner, 2004) makes it possible to model Petri nets using bigraphs. Thus, each model is represented in bigraphs by three parameters: (i) defines processes syntactically (sorting); (ii) defining a signature K; (iii) and then presents its rules of interaction. The only bigraphs permitted are those that satisfy the defined sorting discipline.

3.2 Bigraphical specification of a context-aware system dynamics

A context-aware system (CAS), ***have to manage local context (the system) and global context (context of use; environment)*** to be aware of their state and the state of the environment. To this end, these systems require more adaptive features to deal with usually open-ended, highly parallel, and interactive components that often operate in a highly changing environment. The proposed conceptual framework defines mechanisms to deal with context-aware system dynamics at the local and global context based on bigraphical reaction rules. System behaviour and the environment changes constitute the local and global context dynamics. Besides, modelling context-aware systems consider the process of determining a system's actions as a result of its internal goals resolution or as an environmental change. To provide context-aware system key features such as self-awareness and self-adaptation. Self-awareness refers to the state (information) that a system has about its own behaviour and the environment, it can be distributed or held locally. Furthermore, self-adaptation provides capabilities for flexible adaptation and reconfiguration. To tackle these issues, we an engineering approach based on bigraphical reactive systems (bigraphs + reaction rules). To present the dynamics of processes by means of *reactions* of the form $r \rightarrow r'$ on a bigraph G. Iff $G: \langle m, X \rangle \rightarrow \langle n, Y \rangle$ is active at $i \in m$ if every ancestor node of site i is active. These reactions define the behaviour and the structural evolution. A BRS above the sorting discipline $\sum$ consists of $'\boldsymbol{BG}(\sum)$ enriched with a set $'\boldsymbol{R}$ of reaction rules. We denote the BRS by $'\boldsymbol{BG} \overset{\text{def}}{=} {'\boldsymbol{BG}}(\sum, {'\boldsymbol{R}})$.

4- Case Study

Nowadays, ubiquitous computing has become essential in our daily life as well as in the automotive world. Today's cars are endowed with safety and driver assistance systems that make it easier for drivers to make journeys. These systems are distributed and based on the technology of sensor and camera networks, which intercept danger and therefore reduce the accident rate. Such systems demand runtime capabilities for flexible adaptation and continuous reconfiguration. The purpose of this section is to apply our approach (presented previously) to an intelligent vehicle (driver assistance system) as a case study. But first, we describe the Intelligent driving assistance systems. Then, we present the resulted modelling using our BRS-based approach.

A driver assistance system is a complex system that contains several subsystems based on sensor networks and centralized IT. It is made up of a large number of heterogeneous and distributed physical devices. The system perceives the signals from distributed sensors that differ in their operation (line sensor, obstacle sensor, etc.) and works on a dynamic and changeable environment which aims to:

- Avoid a dangerous situation that could lead to an accident;
- Free the driver from many tasks that could reduce his vigilance;
- Assist the driver in his perception of the environment (overtaking detector, freezing risk detector, etc.).

One of the principles of context-aware systems is to provide services that focus on the ability to interact with the operational environment, as defined in (Dey, 2001) "a system is context-aware if it uses the context to provide relevant information and/or services to a user. Relevance, depends on the task of the user ". We can deduce that a driver assistance system is a perfect example of a context-aware system because the latter perceives information from its environment thanks to its sensors, then adapts its behaviour to its environment according to its intelligence (decision-making process). We have designed an intelligent driver assistance system to decrease the accident rate on urban roads and expressways. Our smart vehicle is equipped with a set of sensors, control modules, and actuators to analyse information from the environment and react by adapting its behaviour, on the traffic and on the car itself. The behaviour is presented as an action or maneuverer (set of actions), which may be the best or the least expensive.

4.1 Intelligent driving assistance system specification

In this section, we apply the bigraphical proposed approach to model our intelligent driving assistance system. We model the structural aspect of our driving assistance system using the Bigraph-CAS. We define the different possible scenarios of our intelligent vehicle system, which manages the different cases to avoid collusion and facilitate the driver's tasks. We instantiate a set of reaction rules on the Bigraph-CAS to model these scenarios.

4.1.1 Smart car assistant system structure

First, we specify the operational context using the bigraph $\mathrm{G}^{\mathrm{ENV}}$ which represents the set of contextual entities of our operational environment and their logical and physical location. Then, we model the intelligent driving assistance system hardware components through the bigraph G^{H}. Afterward, the bigraph G^{A} is used to model the intelligent decision-making process. In this case study, we rely on multi-agent systems to implement the application layer. Finally, the edges, hyperedges, and interfaces represent a middleware permitting (i) interaction and communication between the different entities and (ii) composition of the bigraphs to form our Bigraph-CAS. The contextual bigraph $\mathrm{G}^{\mathrm{ENV}}$ of our system is presented in Fig. 3. The nodes' description is classified in Table 2. As you can see, the nodes represent the contextual entities interconnected with edges and hyperedges, the root (0) represents the operational environment of our system, the sites represent the abstract elements. The nodes of type R are the plots of road, these plots contain lines for the trajectory; these lines can be continuous or discontinuous, O represents an obstacle, and V the intelligent vehicle. We present two contextual bigraph scenarios on this level of abstraction, the first one is on an "urban" road and the second on a "fast lane" represented in Fig 4.

Table 2. Classification of the contextual nodes bigraph $\mathrm{G}^{\mathrm{ENV}}$

Node	Description	Control	Arity	Graphic form
V	Vehicle	V	1	◯
O	Obstacle	O	1	◇
LG	Road Line	L	1	▭
R	Road	R	4	▢

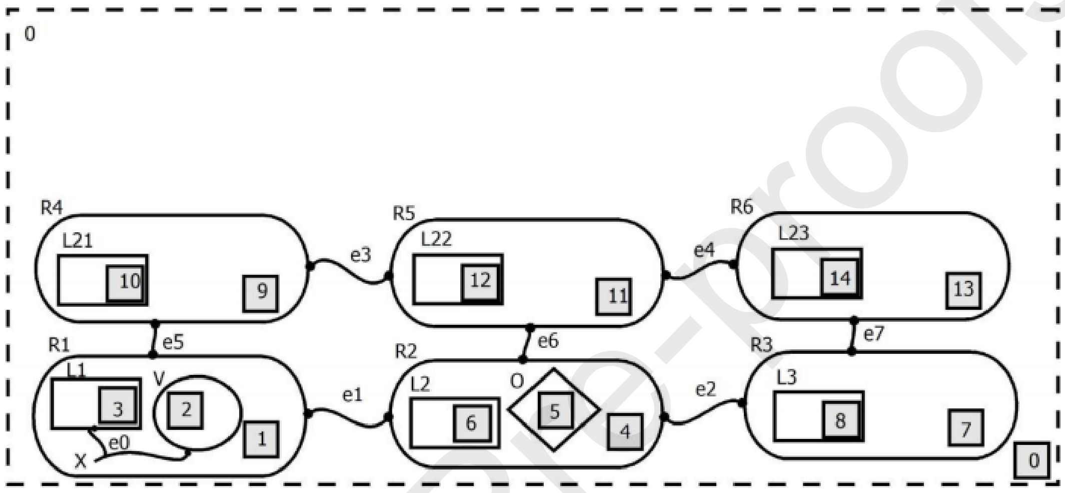


Figure 3. Contextual bigraph: "urban" road scenario

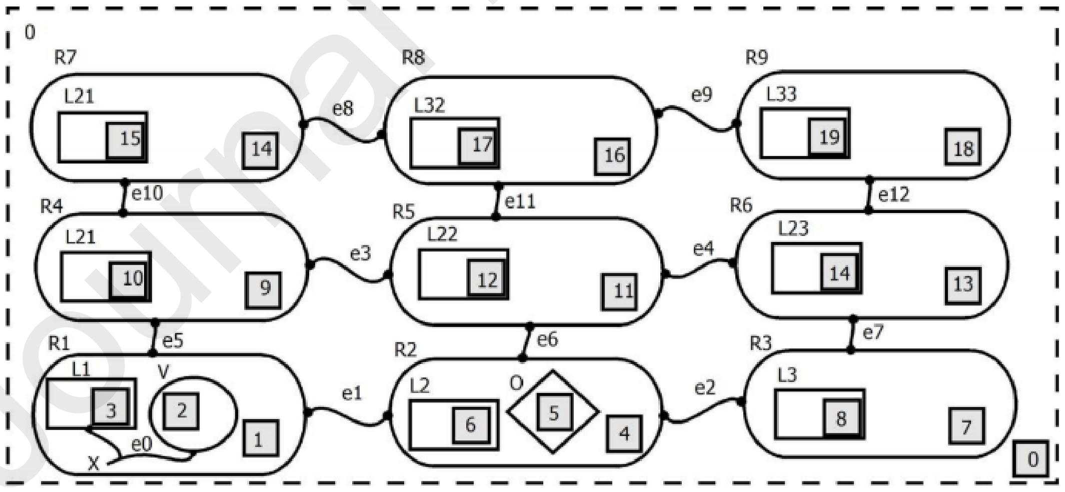


Figure 4. Contextual bigraph: " fast lane " road scenario

G^{ENV} is a bigraph modelling the operational environment of the diver assistance system over a signature K, it takes the form:

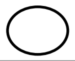
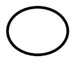
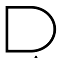
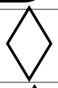
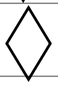
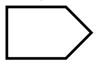
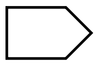
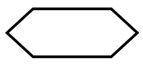
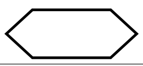
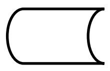
$$G^{ENV} = \left(V^{ENV}, E^{ENV}, \mathrm{ctrl}, G_P^{ENV}, G_L^{ENV}\right): I \rightarrow N$$

– V^{ENV} represents the set of contextual nodes {R1, R2, R3, R21, R22, R23, R31, R32, R33, V, L1, L2, L3, L21, L22, L23, L31, L32, L33, O} .

– E^{ENV} represents the set of links = {e1, e2, e3, e4, e5, e6, e7, e8, e9, e10, e11, e12}.

– the set of controls = {R, R2, R3, V, LG {L, L2, L3}, O}.
– G_P^{ENV}= (V^{ENV} , $ctrl^{ENV}$,$prnt^{ENV}$): $m \to n$.
– $prnt^{ENV}$= $m \uplus V^{ENV} \to V^{ENV} \uplus n$. It is a transformation function that assigns to each node its parent node.
– G_L^{ENV}= (V^{ENV},E^{ENV},$ctrl^{ENV}$, $link^{ENV}$): I→N .
– $link^{ENV}$=: $I \uplus E^{ENV} \to E^{ENV} \uplus N$.
– Set of inner names = {x}.
– Set of outer names= {∅} .
– m= 20; sites number.
– n= 1; region number.
Then, we start with the specification of the hardware layer of the system through the bigraph G^H . This bigraph represents a lower level of abstraction (a Meso level). It specifies the hardware components of the smart car and their interconnections. The vehicle node (V), this bigraph is represented in Fig 5, where Table 3 represents the description of the nodes of the G^H graph and their graphical forms:

Table 3. Classification of the hardware nodes bigraph G^H

Node	**Description**	**Control**	**Arity**	**Graphic form**
GE	Event manager	GE	2	
GM	Engine manager	GM	2	
U	Obstacle sensor	U	2	
UG	Obstacle sensor left	UG	2	
UD	Obstacle sensor Right	UD	2	
LDG	Discontinuous line sensors left	LDG	2	
LDD	Discontinuous line sensors right	LDD	2	
SG	Line sensors left	SG	2	
SD	Line sensors right	SD	2	
CG	Flashing lamps left	CG	2	
CD	Flashing lamps right	CD	2	
S	Stoplights	S	2	

Each node represents a physical element of the intelligent vehicle in the bigraph G^H, here is a description of the roles of each physical element:
(GE) Event manager: this element manages the different incoming signal flows from the sensors and outgoing towards other elements like engine, lamps, etc.; in our vehicle, this element plays the role of the driver assistance system memory.
(U, UD, UG) Obstacle sensors: these elements detect objects on the road; these sensors are based on the principle of "ultrasound," reflecting on an obstacle and returning to its starting point by producing

an echo. In our vehicle, these sensors (U, UD, UG) are located in the front of the vehicle, right mirror, and left mirror.

(LDG, LDD) Discontinuous line sensors: represents the elements that detect the state of the two parallel lines (right and left) that form the trajectory of the road; these sensors are based on the principle of "phototransistors" they will illuminate the ground to catch the lines. In our vehicle, these sensors (LDG, LDD) are located respectively in the middle of the vehicle's right and left edges.

(SLG, SLD) Line sensors: same principle as discontinuous line sensors (LDG, LDD) except that they perceive a trajectory exit. These sensors are located in the front left and front right corners of the vehicle.

(CG, CD) Flashing lamps: located at the corners of the vehicle both light up during emergency braking, to the right when passing to the right, and the left when passing to the left.

(S) Stoplights: which comes on when braking. It is located at the back.

(GM) Engine manager: responsible for braking, acceleration, and also steering of the vehicle.

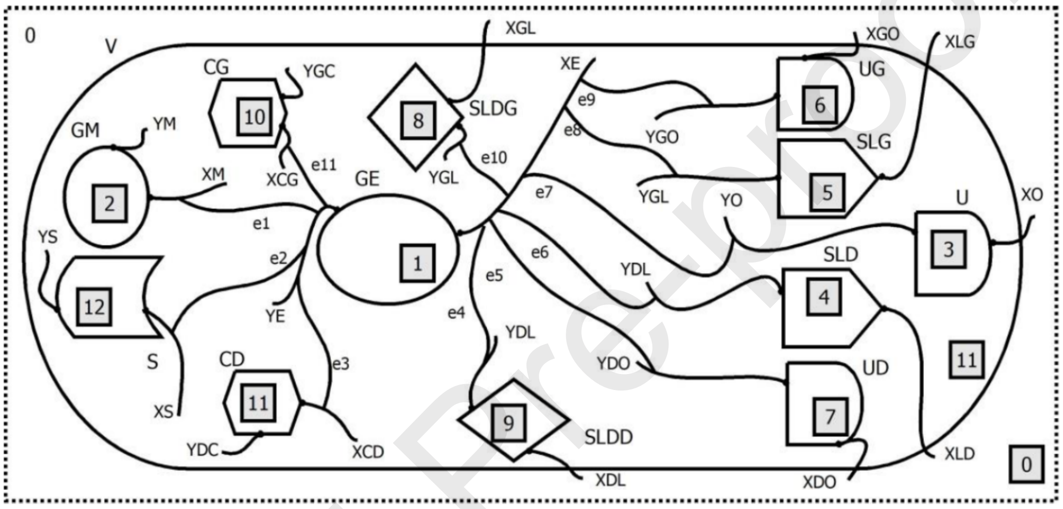


Figure 5. Bigraphical representation of the hardware layer (bigraph G^H)

The G^A bigraph represented in Fig. 6., models the multi-agent level structure of our driver assistance system. The description of the G^A bigraph nodes are classified in Table 4. The agents are represented with a set of nodes. Arcs represent the interconnection links representing the communication between the agents, the root (0) models the SMA environment. In contrast, the sites represent the most abstract elements:

Table 4. Description of the G^A bigraph for driving assistance systems and graphical forms

Node	Description	Control	Arity	Graphic form
AGOB	Obstacle Agent	O	2	◯
AGSL	Line Following Agent	S	2	◯
AGD	Overtaking agent	DG	2	◯
AGP	Pilot Agent	P	2	◯

In our multi-agent system, the agents are distributed according to their roles. Here are the role descriptions of each agent in our system. See Fig. 6.

(AGOB) Obstacle Agent: the role of this agent is to perceive the emergence of an obstacle during the movement of our intelligent vehicle. When an obstacle is detected, the latter evaluates the obstacle's distance and then communicates it to the pilot agent. This agent is qualified as a reactive agent.
(AGSL) Line Following Agent: its role is to perceive the two lines which form the trajectory of a parcel of road, so the vehicle will move forward by following these lines when the latter tilt to the right or the left the agent will Inform the pilot agent so he can adjust the trajectory of the vehicle. The type of this agent is reactive
(AGDD) Overtaking agent: its role is to collect information so that the vehicle can overtake on the right or the left. This information is the state of the left or right line, is it discontinuous, and the 180 ° left field or the 180 ° right field is free, that is to say, that there is an object or a vehicle in the right or left lane. Then the agent will choose the free lane, which allows him to confirm the overrun action (if the two lanes are free, the priority is for overtaking on the right). The result of the perception and the union of all this information will be communicated with the pilot agent. This agent is cognitive.
(AGP) Pilot Agent: this agent communicates with all the previous agents to decide (decision) what action to choose (reaction). This agent is cognitive.

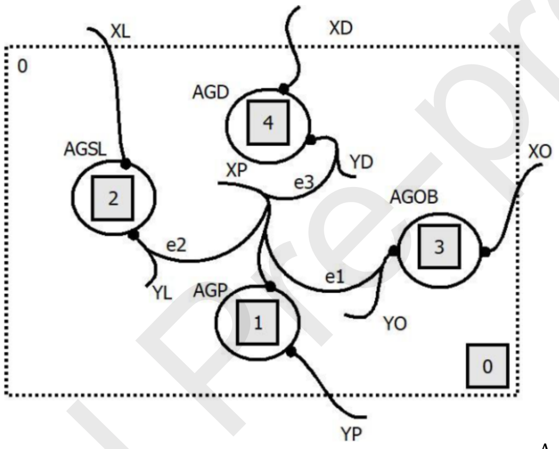


Figure 6. Assistance driving system, Bigraph G^A.

In what follows, and using bigraphical composition. We deploy the software agents (multi-agent system) of the application layer specified by the bigraph G^A on the hardware layer defined by the bigraph G^H. The resulted bigraph is represented in Fig. 7.

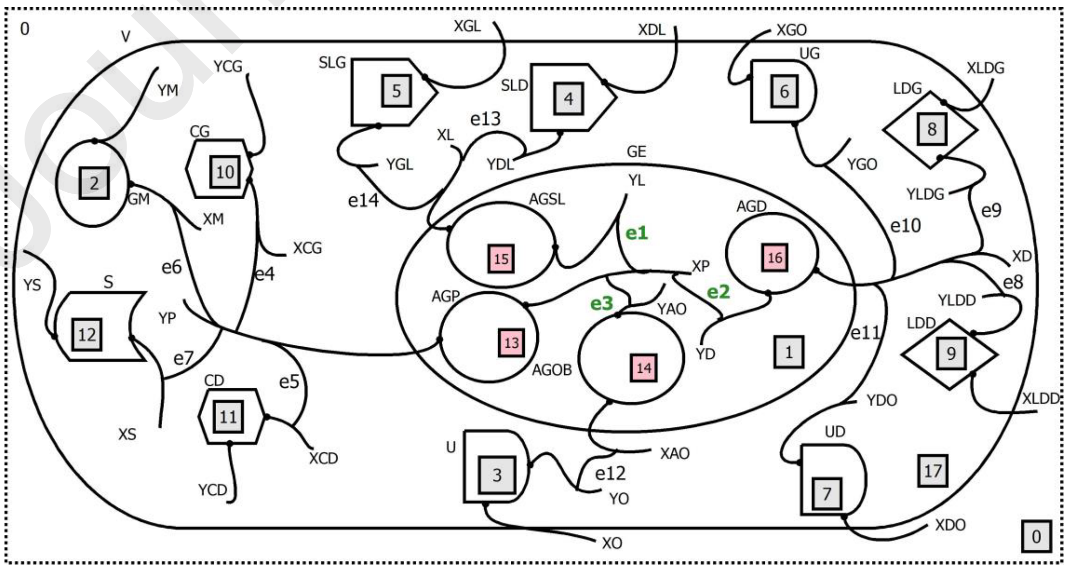


Figure 7. Bigraphical representation of the composition of the hardware and software bigraphs

The algebraic specification is as follows:

$$G^{CAS} = \left(V^{CAS}, E^{CAS}, cntrl^{CAS}, G_P^{CAS}, G_L^{CAS} \right) : (m, Z) \to (n, J)$$

– Set of agent Nodes V^{CAS} = {AGP, AGOB, AGSL, AGDGE, U, UG, UD, SLG, SLD, LDG, LDD, CG, CD, GM, S, V}. It represents the union of a disjoint set of the bigraphs.
– Set of arcs = {e1, e2, e3, e4, e5, e6, e7, e8, e9, e10, e11, e12, e13, e14}.
– set of controls = {P, OB, SL, D, GE, U, UG, UD, SLG, SLD, LDG, LDD, CG, CD, GM, S}.
– $\mathrm{G}_{\mathrm{P}}^{CAS}$= ($\mathrm{V}^{CAS}$, ctrl^{CAS} ,$prnt^{CAS}$): $m \to n$
– $prnt^{CAS}$= $m \uplus \mathrm{V}^{CAS} \to \mathrm{V}^{CAS} \uplus n$. It is a transformation function that assigns to each node its parent node.
– $\mathrm{G}_{\mathrm{L}}^{CAS}$= (V^{CAS},E^{CAS},ctrl^{CAS}, link^{CAS}): Z→J
– link^{CAS}=: $\mathrm{I} \uplus \mathrm{E}^{CAS} \to \mathrm{E}^{CAS} \uplus \mathrm{N}$.
– $Z = \{xP, xD, xAO, xL, xE, xM, xGO, xDO, xO, xDL, xGL, xLDG, xLDD, xCD, xCG, xS\}$ is a set of inner names
– $J = \{ yP, yD, yAO, yL, yE, yM, yGO, yDO, yO, yDL, yGL, yLDG, yLDD, yCD, yCG, yS\}$ is a set of outter names.
– m= 18; site number.
– n= 1; region number.

We mentioned previously that a CAS bigraph is the composition of tree bigraphs; operational context $\mathrm{G}^{\mathrm{ENV}}$, the hardware G^{H} and decision making (application) G^{A} bigraphs which results in our smart driving assistance system.

4.1.2 Smart car assistant system dynamics

The smart car assistant system dynamics are given as bigraphical reactive rules that express the dynamicity and reconfiguration of the system. In this section, we define a set of reaction rules at different levels of reconfiguration. Table 5 gives the defined reaction rules expressing a set of possible actions that can be applied over a smart car assistant system part. An applied reaction rule, defines a redex bigraph to be transformed into a reactum bigraph. As both redex and reactum bigraphs respect the formation rules ΦCS, the reaction rules continuously create setups that are structurally correct by definition. Reactions will not execute if a bigraph is distorted.

Table. 5. Smart car assistant system dynamics, reaction rules description

Layer	Rule	Algebraic description
Environment level	RL1: Lines detection	$\mathrm{R}.(\mathrm{L}.(d_1)\|\mathrm{V}.(d_2)\| d_0) \to \mathrm{R}.(L_{e0}.(d_1)\|\mathrm{V}_{e0}.(d_2)\| d_0)$
	RL2: Move forward (vehicle)	$R.(L_{e0}.(d_1)\|V_{e0}.(d_2)\|d_0)\|R2.(L2(d_4)\|d_3) \to R.(L.(d_1)\|d_0)\|R2.(L2_{e0}(d_4)\|V_{e0}$
	RL3: obstacle detection (frontal)	$\mathrm{R}.(L_{e0}.(d_1)\|V_{x\,e0}.(d_2)\|\mathrm{O}.(d_3)\mid d_0) \to \mathrm{R}.(L_{e0}.(d_1)\|V_{x\,e0\,e13}.(d_2)\|\mathrm{O}_{e13}.(d_3)\mid$

	RL4: obstacle detection (side)	$\mathrm{R}.(L_{e0}\,.(d_1\,)\|\|V_{x\,e0}\;.(d_2\,)\|d_0\,)\|Rj.(\mathrm{O}.(d_3\,)\|\;Lj.(d_4)\|d_5)\;\rightarrow\mathrm{R}.(L_{e0}\,.(d_1\,)\|\|V_{x\,e0\,e13}\,.(d_2\,)\|d_0\,)\|$ $_4)\|d_5)$
	RL5: overtake on the left	$R_i.(L_{i\,e0}\,.(d_1)\|V_{x\,e0\,e13}\,.(d_2\,)\|O_{e13}\,.(d_3)\|d_0)\|R_{i\,+\,1}\,.(L_{i\,+\,1}\,(d_5)\|d_4)\rightarrow R_i.(L_i\,.(d_1)\|O_{e13}\,.(d_3)\|$ $)\|V_{x\,e0\,e13}\,.(d_2\,)\|d_4)$
	RL6: overtake on the right	$R_i.(L_{i\,e0}.(d_1)\|V_{x\,e0\,e13}.(d_2)\|\;O_{e13}.(d_3)\|d_0\|\;R_{i\,-\,1}.(L_{i\,-\,1}.(d_5)\|d_4)\rightarrow R_i.(L_i.(d_1)\|O_{e13}.(d_3)\|d_0\|$ $_{x\,e0}.(d_2)\|d_4)$
	RL7: Stop the vehicle	$R_i.(L_{e0}.(d_1)\|V_{x\,E0}.(d_2)\|d_0))\;\|\;R_j.(d_4)\rightarrow R_i.(L_{e0}.(d_1)\|V_{x\,e0\,e13\,e14}.(d_2)\|O_{e13}.(d_3)\;\|\;d_0)\|R_j.(O2$
Hardware-level	RL 8: brake lights (on)	$V.(AGP_{xy}.(d_1)\|S_{xy}.(d_2)\|d_0)\rightarrow V.(AGP_{x\,y\,e7}.(d_1)\|S_{x\,y\,e7}.(d_2)\|d_0)$
	RL 9: Blinker (on)	$V.(AGP_{xy}.(d_1)\|CN_{xy}.(d_2)\|d_0)\rightarrow V.(AGP_{x\,y\,e4}.(d_1)\|CN_{x\,y\,e4}.(d_2)\|d_0)$
	RL 10: Blinker (off)	$V.(AGP_{x\,y\,e4}.(d_1)\|CN_{x\,y\,e4}.(d_2)\|d_0)\rightarrow V.(AGP_{xy}.(d_1)\|CN_{xy}.(d_2)\|d_0)$
	RL 11: turn on the engine	$V.(AGP_{xy}.(d_1)\|GM_{xy}.(d_2)\|d_0)\rightarrow V.(AGP_{xye6}.(d_1)\|GM_{xye6}.(d_2)\|d_0)$
	RL 12: Turn off the engine	$V.(AGP_{xye6}.(d_1)\|GM_{xye6}.(d_2)\|d_0)\rightarrow V.(AGP_{xy}.(d_1)\|GM_{xy}.(d_2)\|d_0)$
MAS level	RL13 : : Lines detection	$V.(AGP_{xy}.(d_1)\|AGOB_{xy}.(d_2)\|AGSL_{xy}.(d_3)\|AGD_{xy}.(d_4)\|d_0)\rightarrow V.(AGP_{xye1}.(d_1)\|AGOB_{xy}.(d$ $AGD_{xy}.(d_4)\|d_0)$
	RL14: obstacle detection (SMA)	$V.(AGP_{xy}.(d_1)\|AGOB_{xy}.(d_2)\|AGSL_{xy}.(d_3)\|AGD_{xy}.(d_4)\|d_0)\rightarrow V.(AGP_{xye3}.(d_1)\|AGOB_{xye3}.$ $AGD_{xy}.(d_4)\|d_0)$
	RL15: No obstacle detected	$V.(AGP_{xye1}.(d_1)\|AGOB_{xy}.(d_2)\|AGSL_{xye1}.(d_3)\|AGD_{xy}.(d_4)\|d_0)\rightarrow V.(AGP_{xye1e2}.(d_1)\|AGOB_x$ $)\|AGD_{xye2}.(d_4)\|d_0)$
	RL16: obstacle detected	$V.(AGP_{xye1e2}.(d_1)\|AGOB_{xy}.(d_2)\|AGSL_{xye1}.(d_3)\|AGD_{xye2}.(d_4)\|d_0)\rightarrow V.(AGP_{xye1}.(d_1)\|AGO$ $d_3)\|AGD_{xy}.(d_4)\|d_0)$
	RL17: No obstacle detected	$V.(AGP_{xye3}.(d_1)\|AGOB_{xye3}.(d_2)\|AGSL_{xy}.(d_3)\|AGD_{xy}.(d_4)\|d_0\rightarrow V.(AGP_{xy}.(d_1)\|AGOB_{xy}.$ $AGD_{xy}.(d_4)\|d_0)$
MAS/Hardware level	RLCi: agent / sensors communication	$V.(AGi_{xy}.(d_1)\|Ci_{xy}(d_2)\|d_0)\rightarrow V.(AGi_{xy\,ei}.(d_1)\|Ci_{xy\,ei}(d_2)\|d_0)$
	RLD-Ci: agent/sensors communication	$V.(AGi_{xy\,ei}.(d_1)\|Ci_{xy\,ei}(d_2)\|d_0)\rightarrow V.(AGi_{xy}.(d_1)\|Ci_{xy}(d_2)\|d_0)$

Finally, we will apply all the rules defined in Table 5 to some scenarios of our driving assistance system. Here are some scenarios of our system modelled by the application of rules on the bigraph CAS.

Scenario 1 (following lines/trajectory): the vehicle will pick up the parcel lines of road, then the vehicle moves forward on this parcel. To achieve this scenario, you need to apply the following rules:

$$\text{SC1} = CAS_0 \xrightarrow{RL1} CAS_1 \xrightarrow{RLLFR} CAS_2 \xrightarrow{RLLFL} CAS_3 \xrightarrow{RL13} CAS_4 \xrightarrow{RL11} CAS_F$$

The initial state of our CAS bigraph is represented in Fig. 8. and Fig. 9.

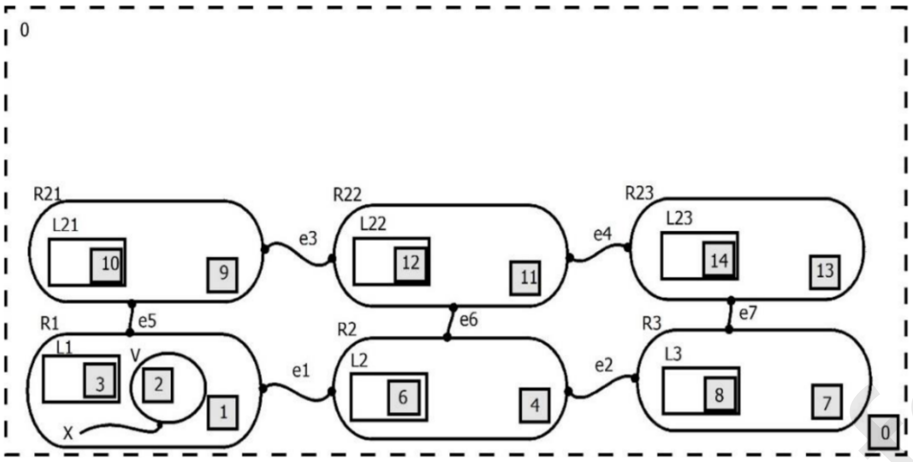


Figure 8. CAS_0 initial state of scenario 1 (urban road level)

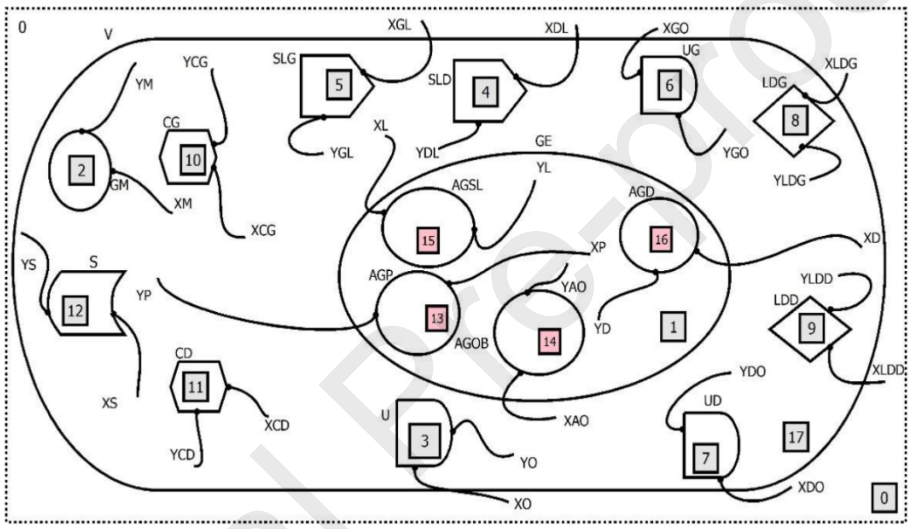


Figure 9. CAS_0 initial state of scenario 1 (vehicle level)

The vehicle will detect the road parcel lines. In this case, we will apply the RL1 rule at the road level and RLSLG, RLSLD rules. The resulted bigraphs are depicted in Fig. 10. and Fig. 11.

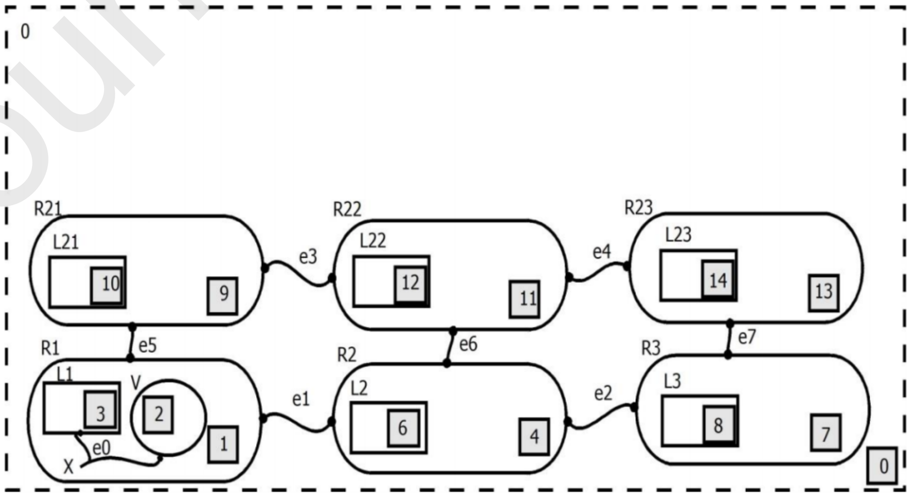


Figure 10. CAS_1 state of scenario 1 (urban road level)

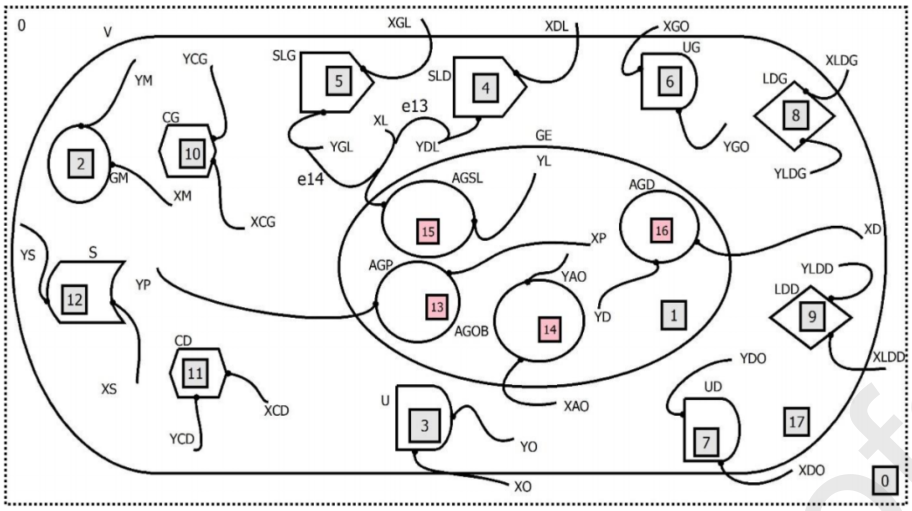


Figure 11. CAS_3 state of scenario 1 (vehicle level).

The AGSL line follower agent will then establish communication with the agent. AGP pilot, therefore, rule RL13 must be applied. This results in the bigraph of Fig. 12.

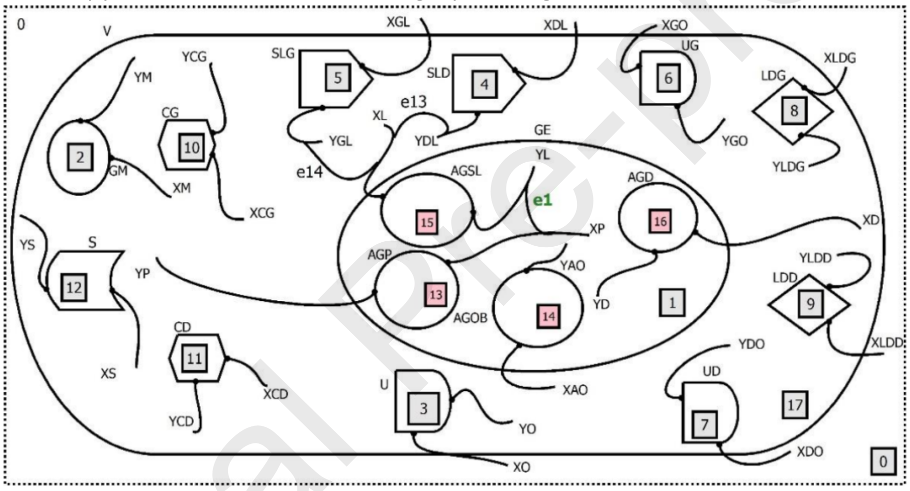


FIGURE. 12. CAS_4 state of scenario 1 (vehicle level)

Finally, the pilot agent will communicate with the GM engine manager agent. The latter makes the vehicle move forward, we will apply the rule RL11 to obtain the bigraph CAS_F.

Scenario 2 (Overtaking an obstacle):

The vehicle is in running mode. It will perceive an obstacle in front. It checks if he can pass; the overtaking can be done either to the right or to the left (priority is set to the left). Therefore, the vehicle will move from a road parcel to the right or left parcel and switch on the left or right flashing lights. This scenario dynamic is specified by the following set of reaction rules:

$$SC2 = CAS_0 \xrightarrow{RL3} CAS_1 \xrightarrow{RLU} CAS_2 \xrightarrow{RL14} CAS_3 \xrightarrow{RL9} CAS_4 \xrightarrow{RL5} CAS_5 \xrightarrow{RLD-U} CAS_6 \xrightarrow{RL17} CAS_7 \xrightarrow{RLD-LDG} CAS_8 \xrightarrow{RLLDD} CAS_9 \xrightarrow{RL4} CAS_{10} \xrightarrow{RLUD} CAS_{11} \xrightarrow{RL16} CAS_{12} \xrightarrow{RL2} CAS_{13} \xrightarrow{RLD-UD} CAS_{14} \xrightarrow{R16} CAS_{16} \xrightarrow{RL2} CAS_{13} \xrightarrow{RLD-UD} CAS_{14} \xrightarrow{RL15} CAS_{15} \xrightarrow{RL6} CAS_{16} \xrightarrow{RLD-LDD} CAS_{17} \xrightarrow{RLLDG} CAS_{18} \xrightarrow{RL10} CAS_F$$

4.2. Executability and formal verification of smart car assistance system

Software verification becomes essential in the development of computer systems in general. To verify and validate the correctness of the proposed BRS-based model. It is necessary to choose the appropriate verification technique along with the corresponding automated tool. There are many formal verification techniques in the literature, and model-checking is the most popular one. Its main benefit is providing a counter-example whenever the desired property does not hold in the actual system model. BigraphER (Sevegnani and Calder, 2016), BPL Tool (Gassara et al., 2017), and BigMC (Perrone et al., 2012) are some of the verification tools build around BRS. However, none of these tools meet our expectations, as they are limited and only suitable for some specific application domains. Also, possible verifications rely on limited pre-defined predicates.

In this work, we choose the Maude LTL model-checker to verify and validate the correctness of the proposed BRS-based model. The Maude programming language (Manuel Clavel et al., 2007) is used to implement the obtained CAS bigraphical models. Maude is a high-level programming language and high-performance system that supports executable specifications. It will be used to define formal specifications for the case study scenarios that are executable, analysable and verifiable. The verification process is fulfilled through the case study (scenario 1). As a first step, we encode the structural aspect (i.e., nodes and their signature and outer and inner interfaces) then the dynamic aspect (i.e., reaction rules) of our BRS-based model into Maude language (Listing 1). Secondly, we formulate the properties that we would like to verify. Finally, we will analyse and validate the resulted output given by Maude.

Listing 1. The Maude implementation of the bigraph CAS concepts.

```
mod BigraphCAS is

        protecting BOOL .
        protecting NAT .

----/////////////////////////    ----//Interface Level2
        sort Big .
        sort Edge .

 op e_ : Nat -> Edge [ctor] .
 op - : -> Edge [ctor] .
 op _|_ : Big Big -> Big [ctor assoc comm id: null] .
 op _||_ : Big Big -> Big [ctor assoc comm id: null] .
        op null : -> Big [ctor] .

----///////////////////----//Environment Level1

        op V_[_].{ _ } : Nat Edge Big -> Big [ctor] .
        op O_[_].{ _ } : Nat Edge Big -> Big [ctor] .
        op L_[_].{ _ } : Nat Edge Big -> Big [ctor] .
        op R_[_].{ _ } : Nat Edge Big -> Big [ctor] .

----/////////////////////////////----//Hadware Level3

        op GE_[_].{ _ } : Nat Edge Big -> Big [ctor] .
        op GM_[_].{ _ } : Nat Edge Big -> Big [ctor] .
        op U_[_].{ _ } : Nat Edge  Big -> Big [ctor] .
        op UG_[_].{ _ } : Nat  Edge Big -> Big [ctor] .
        op UD_[_].{ _ } : Nat  Edge Big -> Big [ctor] .
        op LDG_[_].{ _ } : Nat Edge Big -> Big [ctor] .
        op LDD_[_].{ _ } : Nat Edge Big -> Big [ctor] .
        op SLG_[_].{ _ } : Nat  Edge Big -> Big [ctor] .
        op SLD_[_].{ _ } : Nat  Edge Big -> Big [ctor] .
        op CG_[_].{ _ } : Nat  Edge Big -> Big [ctor] .
        op CD_[_].{ _ } : Nat  Edge Big -> Big [ctor] .
```

```
        op S_[_].{ _ } : Nat  Edge Big -> Big [ctor] .

----///////////////  ----//Application(MAS) Level4

        op AGOB_[_].{ _ } : Nat  Edge Big -> Big [ctor] .
          op AGSL_[_].{ _ } : Nat  Edge Big -> Big [ctor] .
           op AGD_[_].{ _ } : Nat  Edge Big -> Big [ctor] .
           op AGP_[_].{ _ } : Nat  Edge Big -> Big [ctor] .

     ------/////////////       ------///Reaction Rules

vars d0 d1 d2 d22 d222 d2222 d3 d11 d12 d711 d700
d011 d0117 : Big .

rl [LinesDetection] :
R 1[-].{ L 1[-].{ d11 } | V 1[-].{ d2 } | d0}
 =>
R 1[-].{ L 1[e 0].{ d11 } | V 1[e 0].{ d2 } | d0} .

rl [AgentSensorsCommunication1] :
V 1[e 0].{ AGSL 1[-].{ d1 } | SLG 1[-].{ d2 } | d0} | d3
 =>
V 1[e 0].{ AGSL 1[e 14].{ d1 } | SLG 1[e 14].{ d2 } | d0}
 | d3 .

rl [AgentSensorsCommunication2] :
V 1[e 0].{ AGSL 1[e 14].{ d1 } | SLD 1[-].{ d22 } | d0} |
d3
 =>
V 1[e 0].{ AGSL 1[e 13].{ d1 } | SLD 1[e 13].{ d22 } |
d0} | d3 .

rl [LinesProcessing] :
```

```
V 1[e 0].{ AGSL 1[e 13].{ d1 } | AGD 1[-].{ d22 } |
AGOB 1[-].{ d222 } | AGP 1[-].{ d2222 } | d0} | d3
 =>
V 1[e 0].{ AGSL 1[e 1].{ d1 } | AGD 1[-].{ d22 } | AGOB
1[-].{ d222 } | AGP 1[e 1].{ d2222 } | d0} | d3 .


rl [NoObsDetected] :
V 1[e 0].{ AGSL 1[e 1].{ d1 } | AGD 1[-].{ d22 } | AGOB
1[-].{ d222 } | AGP 1[e 1].{ d2222 } | d0} | d3
 =>
V 1[e 0].{ AGSL 1[e 1].{ d1 } | AGD 1[e 2].{ d22 } |
AGOB 1[-].{ d222 } | AGP 1[e 12].{ d2222 } | d0} | d3 .



rl [TurnOnEngine] :

V 1[e 0].{ AGSL 1[e 1].{ d1 } | AGD 1[e 2].{ d22 } |
AGOB 1[-].{ d222 } | AGP 1[e 12].{ d2222 } | d0} | GM
1[-].{ d12 } | d3
 =>
 V 1[e 0].{ AGSL 1[e 1].{ d1 } | AGD 1[e 2].{ d22 } |
AGOB 1[-].{ d222 } | AGP 1[e 6].{ d2222 } | d0} | GM
1[e 6].{ d12 } | d3 .

rl [MoveForwardVehicle] :
R 1[-].{ L 1[e 0].{ d11 } | V 1[e 0].{ AGSL 1[e 1].{ d1 } |
AGD 1[e 2].{ d22 } | AGOB 1[-].{ d222 } | AGP 1[e 6].{
d2222 } | d0} | GM 1[e 6].{ d12 } | d0} | R 2[-].{ L 2[-].{
d711 } | d700}
 =>
R 2[-].{ L 2[e 0].{ d11 } |  V 1[e 0].{ AGSL 1[e 1].{ d1 } |
AGD 1[e 2].{ d22 } | AGOB 1[-].{ d222 } | AGP 1[e 6].{
d2222 } | d0} | GM 1[e 6].{ d12 } | d0} | R 1[-].{ L 1[-].{
d711 } | d700} .



 endm
```

Listing 2 represents the initial stat of the case study (scenario 1). The Maude command *red modelCheck(initial1, <> [] ~ GoalNotAcheiv)* as transcribed here states that there will be a future state reachable by a step of reaction rule from the current one where a goal will always be achieved. The result of the execution is depicted in Fig. 13. It shows that after 13 steps, the Maude model checker has used all the reaction rules and successfully reached the desired state. (The car moved forward and arrived at the desired destination) without reporting any violation, which makes the model free of non-compliant execution.

Listing 2. Initial state and property check of the case study (scenario 1) encoded in Maude language.

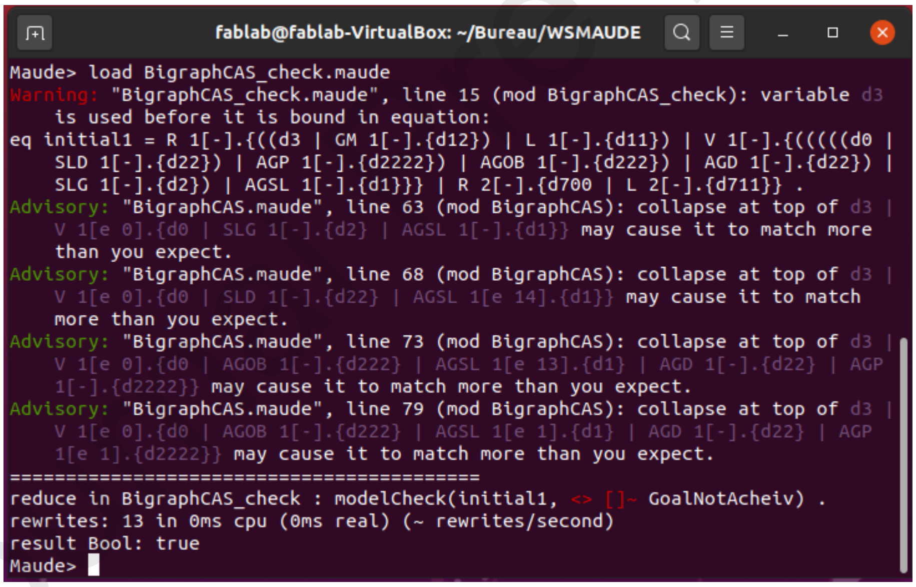
```
 mod BigraphCAS_check is
   protecting BigraphCAS_Prop .
   including MODEL-CHECKER .
   including LTL-SIMPLIFIER .

   ops initial1 : -> Big .

   vars d0 d1 d2 d3 d22 d222 d2222 d11 d12 d711 d700 d011 d0117 : Big .

   eq initial1 = R 1[-].{ V 1[-].{ AGSL 1[-].{ d1 }
                                                  | SLG 1[-].{ d2 } | AGD 1[-].{ d22 }  | AGOB 1[-].{ d222 } | AGP 1[-].{ d2222 }
                                                  | SLD 1[-].{ d22 } | d0}  | L 1[-].{ d11 } | GM 1[-].{ d12 }  | d3 }
                                                  | R 2[-].{ L 2[-].{ d711 } | d700}  .
  endm

 red modelCheck(initial1, <> [] ~ GoalNotAcheiv) .
```



```
Maude> load BigraphCAS_check.maude
Warning: "BigraphCAS_check.maude", line 15 (mod BigraphCAS_check): variable d3
    is used before it is bound in equation:
eq initial1 = R 1[-].{((d3 | GM 1[-].{d12}) | L 1[-].{d11}) | V 1[-].{((((((d0 |
    SLD 1[-].{d22}) | AGP 1[-].{d2222}) | AGOB 1[-].{d222}) | AGD 1[-].{d22}) |
    SLG 1[-].{d2}) | AGSL 1[-].{d1}}} | R 2[-].{d700 | L 2[-].{d711}} .
Advisory: "BigraphCAS.maude", line 63 (mod BigraphCAS): collapse at top of d3 |
    V 1[e 0].{d0 | SLG 1[-].{d2} | AGSL 1[-].{d1}} may cause it to match more
    than you expect.
Advisory: "BigraphCAS.maude", line 68 (mod BigraphCAS): collapse at top of d3 |
    V 1[e 0].{d0 | SLD 1[-].{d22} | AGSL 1[e 14].{d1}} may cause it to match
    more than you expect.
Advisory: "BigraphCAS.maude", line 73 (mod BigraphCAS): collapse at top of d3 |
    V 1[e 0].{d0 | AGOB 1[-].{d222} | AGSL 1[e 13].{d1} | AGD 1[-].{d22} | AGP
    1[-].{d2222}} may cause it to match more than you expect.
Advisory: "BigraphCAS.maude", line 79 (mod BigraphCAS): collapse at top of d3 |
    V 1[e 0].{d0 | AGOB 1[-].{d222} | AGSL 1[e 1].{d1} | AGD 1[-].{d22} | AGP
    1[e 1].{d2222}} may cause it to match more than you expect.
==========================================
reduce in BigraphCAS_check : modelCheck(initial1, <> []~ GoalNotAcheiv) .
rewrites: 13 in 0ms cpu (0ms real) (~ rewrites/second)
result Bool: true
Maude>
```

Figure 13. Execution results of the case study (scenario 1).

5- Discussion and related works

Context-aware computing helps establish smart ecosystems in Ambient Intelligence, Internet of Things, Mobile Computing, Pervasive and Ubiquitous Computing. However, context-aware system design is a significant issue. There exist several context modelling approaches that are worth considering. For a systematic review, this section investigates context-aware systems according to context representations and system modelling. In (Bettini et al., 2010) the authors gave a general viewpoint and categorised state of the art according to the scheme of exchange of contextual information: graphic-based, logic-based and ontology-based models. Besides, the authors highlighted that except for the ontological approaches that provide logical reasoning, the other approaches draw a general modelling schema of the context-aware system without making any distinction between the context representation and the system modelling.

In (Rakib and Faruqui, 2021), the authors proposed an ontology-based approach for the specification and verification of multi-agent systems in which processes are represented as agents while a message-passing technique performs communication. For the specification of the behaviour, the authors proposed an abstract model based on strategies. The resulting specification is translated into LTL formulas that enable verification of system properties using a model checking tool.

In (Hsieh, 2021), the authors proposed a multi-agent system workflow for context-aware systems. The proposed solution focuses on designing context-aware workflow management systems for CPS in an IoT-enabled manufacturing environment where the system is divided into two parts, the physical world and the cyber world. The Cyber World is modelled by agents specified using discrete timed Petri nets while the physical world is reduced as contextual information. To validate the correctness of the proposed approach, a series of experiments (simulations) have been conducted.

In (Mahfooz Ul Haque et al., 2021), the authors suggested a specific context-aware framework for modelling and verifying smart parking systems in urban cities. The proposed approach promotes flexible decisions and a decentralized environment to find parking slots dynamically while moving and/or arriving at a given destination. The Uppaal model checker is used to analyse and verify system properties formally.

In (van Engelenburg et al., 2019), the authors suggested a method for designing a context-aware system dedicated to information sharing in the container shipping domain. The modelling is specified using predicate logic. To illustrate the proposed system handling, the authors provided several definitions and steps to guide designers in their tasks.

Among the formal approaches proposed in context-aware systems using bigraphs, we cite (Cherfia et al., 2016), (Sahnoun et al., 2017). In these approaches, concepts like hierarchy and modularity are the keys concepts used to reduce the design complexity. The work in (Sahnoun et al., 2017) represents the most recent approach combining multi-agent systems and BRS for the formal modelling of context-aware systems. In this work, a CAS comprises two bigraphs representing the contextual part and the operational part of the system. The structure and the behaviour of each part are defined independently, and their composition is modelled as a BRS. To validate the soundness of the approach, the authors used a Bigraphical Model Checker (BigMC) (Perrone et al., 2012).

To summarise, our work adds to the current works a generic and clear methodology for modelling context-aware systems. On the other hand, approaches as (Mahfooz Ul Haque et al., 2021) and (van Engelenburg et al., 2019) are dedicated to a specific application domain. Further, in (Rakib and Faruqui, 2021) and (Hsieh, 2021), the environment and hardware parts and their relation with the system are not addressed, and no details are provided.

The proposed approach introduced in this work represents an extension of our previous work (Sahnoun et al., 2017). We precisely focus on the structural aspect by proposing multi-layered modelling (the environment, interaction, application, and hardware layers). We note the integration of the global context (the environment layer) and local context as an essential modelling dimension by explicitly specifying the relationship between the environment and the system reaction, in contrast with the existing works. This is done using sorted bigraphs. Sorts are utilized to recognize node types for structural purposes and constraints, whereas controls distinguish states and parameters a node can have. The proposed specifications allow the designer to specify the adequate CAS in conformity with its application domain. Further, the dynamic behaviour and the reconfiguration of the CAS are modelled by a set of defined reaction rules. We strictly seek to meet three conditions: (i) Reduce complexity by providing solid concept layers, (ii) ensure proactivity and adaptability, and (iii) safety. To that, we have chosen bigraphs as an underlying theory in our approach. It fits well with the conditions mentioned above. Bigraphs provides rigor through mathematical equations to ensure robustness and

correctness. So, we can easily model the intrinsic hierarchical nature of the system. Besides, CAS requires intelligence. This naturally leads us to apply intelligent agent-based systems. It also provides a way to validate the designed CAS before the implementation using the model checking tool.
Table 6 summarises a small comparative study, organised according to the following features (i) the used formalism or formal model, (ii) the provided modelling features in terms of environment, hard and soft system, relationships and dependencies of context and context-aware system, dynamicity and (iii) the provided verification and evaluation of the modelling approach.

Table 6. Comparison of context-aware systems modelling approaches

Approach	**Formalism/formal model**	**CAS modelling features**				**Verification**
		Environment	Hard /soft system	Dependencies	Dynamicity	
(Rakib and Faruqui, 2021)	Ontology	-	Soft system	-	-	Covered
(Hsieh, 2021)	Discrete timed Petri nets	-	Hard /soft system	-	Partially Covered	Not covered
(Mahfooz Ul Haque et al., 2021)	Resource-bounded logic	-	Hard /soft system	-	Partially Covered	Covered
(van Engelenburg et al., 2019)	First-order logic	-	Soft system	-	Partially Covered	Not covered
(Cherfia et al., 2016)	**Bigraphs**	**Not covered**	**Hard system**	**Partially Covered**	**Partially Covered**	**Covered**
(Sahnoun et al., 2017)	Bigraphs	Not covered	Hard /soft system	Partially Covered	Partially Covered	Covered
Our approach	Ontology + sorted bigraphs +BRS	Covered	Hard /soft system	Covered	Covered	Covered

Conclusion

In this study, we provided a view of context-aware systems, including all modelling features (Environment, interaction, **software and hardware**) involved in the development of such systems. Structural and behavioural aspects of context-aware systems have been modelled using the BRS formalism. Precisely, **we used** bigraphs (sorted bigraphs) to explicitly define the different layers of our CAS. Each layer **has been** considered as a subsystem of our CAS. It **has been** specified by a bigraph along with its sorting discipline (sorts, signature, and formation rules).
Further, bigraphical reaction rules are used to model the behaviour and reconfiguration of the different layers of our context-aware system. Furthermore, the bigraph specifications **have been** coded in the Maude language to allow their execution. We also check the correctness of the resulted behaviours utilizing a state-based model-checker, relying on LTL. In this present work, **we took** a first step towards the modelling of the context-aware system by proposing:

- A Multi-layered modelling, identifying what generates, separates, and links hierarchical levels
- A solid formal foundation that, ensures the robustness and the correctness of the system and its evolution using execution and verification tools.

In the next step, we plan to enlarge our specifications to provide more adaptation capabilities. Finally, our objective is to provide a complete automated executable and verifiable environment for developing context-aware systems. Going even further, we are currently building prototypes from the resulting specifications.

References

Benzadri, Z., Bouheroum, A., Belala, F., 2021. A Formal Framework for Secure Fog Architectures. International Journal of Organizational and Collective Intelligence 11. https://doi.org/10.4018/IJOCI.2021040103

Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D., 2010. A survey of context modelling and reasoning techniques. Pervasive and Mobile Computing 6. https://doi.org/10.1016/j.pmcj.2009.06.002

Boucebsi, R., Belala, F., 2020. A Bigraphical Reactive Systems with Sharing for modeling Wireless Mesh Networks. Journal of King Saud University - Computer and Information Sciences 32. https://doi.org/10.1016/j.jksuci.2018.10.016

Cherfia, T.A., Belala, F., Barkaoui, K., 2016. A bigraph-based framework for specification and analysis of context-aware systems. International Journal of Critical Computer-Based Systems 6. https://doi.org/10.1504/IJCCBS.2016.081808

Common Criteria, 2021. https://commoncriteriaportal.org/cc/ [WWW Document].

Dey, A.K., 2001. Understanding and Using Context. Personal and Ubiquitous Computing 5. https://doi.org/10.1007/s007790170019

Dib, A.T.E., Barkaoui, K., Sahnoun, Z., 2016. Specification and verification of reconfigurable multi-agent system architectures. Multiagent and Grid Systems 12. https://doi.org/10.3233/MGS-160246

Hsieh, F.-S., 2021. A Dynamic Context-Aware Workflow Management Scheme for Cyber-Physical Systems Based on Multi-Agent System Architecture. Applied Sciences 11. https://doi.org/10.3390/app11052030

Lüddecke, D., Bergmann, N., Schaefer, I., 2014. Ontology-Based Modeling of Context-Aware Systems. https://doi.org/10.1007/978-3-319-11653-2_30

Mahfooz Ul Haque, H., Zulfiqar, H., Ahmed, A., Ali, Y., 2021. A context-aware framework for modelling and verification of smart parking systems in urban cities. Concurrency and Computation: Practice and Experience 33. https://doi.org/10.1002/cpe.5401

Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, Carolyn Talcott, 2007. All About Maude - A High-Performance Logical Framework. Springer Berlin Heidelberg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-71999-1

Milner, R., 2008. Bigraphs and Their Algebra. Electronic Notes in Theoretical Computer Science 209. https://doi.org/10.1016/j.entcs.2008.04.002

Milner, R., 2004. Bigraphs for Petri Nets. https://doi.org/10.1007/978-3-540-27755-2_19

Perrone, G., Debois, S., Hildebrandt, T.T., 2012. A model checker for Bigraphs, in: Proceedings of the 27th Annual ACM Symposium on Applied Computing - SAC '12. ACM Press, New York, New York, USA. https://doi.org/10.1145/2245276.2231985

Preuveneers, D., van den Bergh, J., Wagelaar, D., Georges, A., Rigole, P., Clerckx, T., Berbers, Y., Coninx, K., Jonckers, V., de Bosschere, K., 2004. Towards an Extensible Context Ontology for Ambient Intelligence. https://doi.org/10.1007/978-3-540-30473-9_15

Qin, W., Shi, Y., Suo, Y., 2007. Ontology-based context-aware middleware for smart spaces. Tsinghua Science and Technology 12. https://doi.org/10.1016/S1007-0214(07)70179-7

Rakib, A., Faruqui, R.U., 2021. Model checking ontology-driven reasoning agents using strategy and abstraction. Concurrency and Computation: Practice and Experience 33. https://doi.org/10.1002/cpe.5205

Sahnoun, A., Dib, A.T.E., Maamri, R., 2017. A multi-agent based approach for modelling context-aware systems. Multiagent and Grid Systems 13. https://doi.org/10.3233/MGS-170276

van Engelenburg, S., Janssen, M., Klievink, B., 2019. Designing context-aware systems: A method for understanding and analysing context in practice. Journal of Logical and Algebraic Methods in Programming 103. https://doi.org/10.1016/j.jlamp.2018.11.003

Xu, N., Zhang, W.S., Yang, H.D., Zhang, X.G., Xing, X., 2013. CACOnt: A Ontology-Based Model for Context Modeling and Reasoning. Applied Mechanics and Materials 347–350. https://doi.org/10.4028/www.scientific.net/AMM.347-350.2304