Modeling Self-Adaptive Fog Systems Using Bigraphs

Hamza Sahli, Thomas Ledoux, Éric Rutten

Modeling Self-Adaptive Fog Systems Using Bigraphs

Hamza Sahli[1], Thomas Ledoux[2], and Éric Rutten[3]

[1] Stack team, Inria Rennes - Bretagne Atlantique, Nantes, France
[2] IMT Atlantique, Stack, Inria, LS2N, Nantes, France
[3] Univ. Grenoble Alpes, Inria, CNRS, LIG, F-38000 Grenoble, France
{hamza.sahli, eric.rutten, thomas.ledoux}@inria.fr

Abstract. Fog systems are a recent trend of distributed computing having vastly ubiquitous architectures and distinct requirements making their design difficult and complex. Fog computing is based on an idea that consists of leveraging both resource-scarce computing nodes around the Edge to perform latency and delay sensitive tasks and Cloud servers for the more intensive computation. A convenient way to address the challenge of designing Fog systems is through the use of formal methods, which provide the needed precision and high-level assurance for their specification through formal verification. In this paper, we present a novel formal model defining spatial and structural aspects of Fog-based systems using Bigraphical Reactive Systems, a fully graphical process algebraic formalism. The model is extended with reaction rules to represent the dynamic behavior of Fog systems in terms of self-adaptation. The notion of bigraph patterns is used in conjunction with boolean and temporal operators to encode spatio-temporal properties inherent to Fog systems and applications. The feasibility of the modelling approach is demonstrated via a motivating case study and various self-adaptation scenarios.

Keywords: Fog systems · Self-adaptation · Formal methods · Modeling · Bigraphical reactive systems.

1 Introduction

Fog computing [7,14] is an emerging paradigm that aims to decentralize Cloud systems with distributed micro data-centers in the core network and even more resource-scarce devices at the Edge of the network. The main idea behind the Fog is to leverage resources around the Edge to perform latency and delay sensitive tasks closer to end-users and thus avoid network bottlenecks. Cloud servers are exploited only for the more intensive and latency insensitive functions. By introducing location-based awareness, the Fog paradigm grants enhanced performance and more assurance about computation and data placement.

A typical Fog system architecture is considered as a set of heterogeneous computing nodes scattered across separate layers. The bottom one is the Edge

or the *Fog layer* organized as multiple clusters acting as small to medium sized data-centers close to data producers. Each of these clusters is composed of several computing devices of different kinds, referred to usually as Fog/Edge nodes. The latter can range from networking devices equipped with low computing capacities (e.g. routers) to more powerful computing entities (e.g. Raspberry Pi). Fog nodes are ubiquitous by nature, sometimes mobile and mostly possess limited processing power, storage and energy abilities. The top layer of a Fog system is the *Cloud layer* which contains much powerful servers performing computation-intensive tasks. Deployed on resource-scarce computing nodes, a Fog application should be designed as a set of loosely coupled and fine-grained components. To this end, the concept of micro-services provides a new architectural style where Fog-based applications can be quickly deployed and smoothly scaled. In such highly dynamic and large-scale Fog systems, aspects such as locality of nodes and their characteristics, cluster formations and neighboring relation, services distribution, as well as Fog systems temporal evolution are key concerns making the task of designing Fog systems challenging. The main questions we address is how to design a Fog system accurately and the proper means to express faithfully properties providing assurances about respecting system and application constraints/requirements while continuously evolving over time. Formal methods offer the appropriate answers to tackle these open issues. They present mathematical concepts which provide the needed rigour and high-level assurance for qualitative specification of Fog systems infrastructures and applications, in addition to their dynamic behaviors and spatio-temporal properties.

In this paper, we propose a preliminary modelling approach for self-adaptive Fog systems based on a formalism introduced by R. Milner [6]. Bigraphical reactive systems (BRS for short) is adopted as a formal foundation for its ability to represent the hierarchical locations of Fog-based systems entities and applications, as well as their characteristics and the complex interactions between the various elements. The notion of bigraphical reaction rules is used to capture the temporal evolution of Fog systems in terms of self-adaption in a natural way. Besides, bigraph patterns can be combined with standard boolean operators or more expressive temporal logic operators to define spatio-temporal properties related to such systems behavior and Fog applications requirements, serving as a basis to perform formal analysis. Moreover, BRS is characterized by a human-oriented graphical aspect supporting equivalent representation to algebraic ones. This feature provides an intuitive representation of Fog-based systems making them more convenient to comprehend by DevOps engineers. All these distinctive features among others omitted here (e.g. the concept of regions very close to distributed layers) make BRS formalism a fitting candidate to formalize Fog. Although other formalisms may present more or less suitability in some modelling areas to some extent, we advocate that BRS is a more fitting as it covers ideally most modelling requirements of Fog systems and allows to express and reason about their behavioral properties via formal verification.

Bigraphical reactive systems were applied on a wide range of distributed and ubiquitous system such as multi-agent systems [5], wireless networks [2], IoT in-

frastructures [11], Cloud-based elastic system [9,10]. Among existing bigraphical approaches, the most related ones in principal to ours are [11] where a BRS-based framework for modelling large-scale sensor network infrastructures and verification of temporal logic properties specifying application requirements is proposed. Authors in [12] presented a similar bigraphical approach for the modelling of evolving Cyber-Physical Spaces and reasoning about spatio-temporal properties. To the best of our knowledge, the approach we propose here is the first to adopt bigraphical reactive systems for modelling structural and behavioural aspects of Fog systems and formalizing relevant constraints and properties. As a matter of fact, we believe our model is the first of its kind to apply a formal method in order to address these kinds of aspects inherent to Fog systems.

The remainder of this paper is organized as follows. Section 2 summarizes the main elements of BRS. In Section 3, we propose a formalization of Fog systems, then we address the representation of structural aspect using sorted bigraphs and finally we extend the model to include dynamic behaviour in terms of self-adaptation. Section 4 is dedicated to presenting a motivating case study, several adaptation scenarios and properties demonstrating the feasibility of our approach. Finally, Section 5 concludes and discusses our perspectives.

2 Bigraphical Reactive Systems

Bigraphical Reactive Systems (BRS) is a universal meta-modeling formalism originally defined by R. Milner in [6] to provide a fully graphical formal model that emphasizes the orthogonal notions of locality and connectivity. The formalism comes as well with a mathematical algebraic language capable of describing the structural and behavioral aspects of ubiquitous and interacting systems that evolve in both time and space. A BRS $BG(\Sigma, \mathcal{R})$ consists of a category of bigraphs defined over a sorting *discipline* Σ to represent the state of a given system and a set of parametric *reaction rules* $\mathcal{R}$ defining the dynamic evolution of the system by modeling the temporal self-adaption of the set of bigraphs [6]. A bigraph consists of a combination of a *place graph*, i.e. entities in terms of spatial distribution and nesting, and a *link graph* specifying the interconnections between these entities.

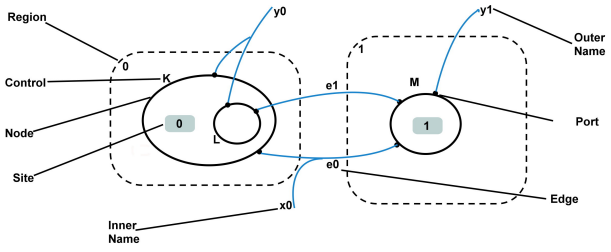

Fig. 1. Anatomy of a bigraph.

Graphical and Algebraic Forms. Bigraphs can be described in two ways, in a diagrammatic depiction or with an equivalent algebraic representation. Figure 1 depicts the anatomy of a simple bigraph. Graphically, a bigraph as an ordinary graph is composed of several nodes linked by edges. Nodes encode the modeled system entities which can be real or virtual, while edges or links represent the various interactions and relationships between the entities. Nodes are always nested within regions, i.e. roots, depicted by dashed rectangles; they may also be nested within each other. This defines a parenting relationship between them in terms of nodes containment, or in other words the place graph of the bigraph. As a rule, nodes in the theory of BRS are permitted to take any kind of shape or color in some cases. A node can be dotted with zero, one or many ports used to link it with other nodes by means of edges, forming a link graph of a bigraph. A control is associated with each node consisting of a node type identifier. Controls denote the arity of nodes i.e. ports number, which of these nodes are atomic i.e. empty nodes, and which of the non-atomic nodes are active, i.e. subject to reactions, or passive. Sites modeled by grey squares are considered as an abstraction indicating the presence of other nodes which represents hidden parts of the model. The inner/outer names of a bigraph are links or potential links to other bigraphs. By convention, inner/outer names are drawn above or below the bigraph (e.g. x0 and y0 in Figure 1).

In addition to their rich graphical representation, a term algebraic language resembling traditional process calculi is defined to encode bigraphs. The language primary elements and operations for composing elementary bigraphs are listed in Table 1. For instance, the parallel product $U||V$ term may be used to compose bigraphs by juxtaposing their roots and merging their common names. The merge product term $U|V$ allows juxtaposing two bigraphs and merging their regions into a single one. As an additional example is the nesting operation $U.V$ which denotes a bigraphical term placed inside another one.

Table 1. Terms language for bigraphs.

Term	Signification
$U\|\|V$	Parallel product
$U\|V$	Merge product
$U.V$	Nesting (U contains V)
K_x	Node of type (control) K having a name or link x
id	Identity or elementary bigraph
d_i	Site numbered i
1	The barren (empty) region

Bigraph Definition. A bigraph is formally defined by $G = (V, E, ctrl, G^P, G^L) : I \to J$, where V and E are respectively finite sets of nodes and edges. $ctrl : V \to K$ is mapping function associating a control to each node in the bigraph G. K referred to as the bigraph signature is its set of controls. $G^P = (V, ctrl, prnt) : m \to n$ is the place graph of G. m and n are respectively the number of sites and regions, $prnt : m \uplus V \to V \uplus n$ is a mapping function that identifies the parent of each node. $G^L = (V, E, ctrl, link) : X \to Y$ represents the link graph of the bigraph G, where $link : X \uplus P \to E \uplus Yi$ is a link map, X and Y are respectively

inner and outer names. P is a set of ports of the bigraph G. $I = \langle m, X \rangle$ and $J = \langle n, Y \rangle$ are respectively inner and outer interfaces of the bigraph encoding the potentiality of G to interact its external environments.

Sorting Discipline. A bigraph can be further associated with a sorting discipline allowing to classify the different bigraphical terms specified in the bigraph such as nodes, links and sites. This sorting logic helps as well as defining the necessary constraints on the designed bigraphs. A sorting discipline Σ is a triple $\Sigma = \{\Theta, K, \Phi\}$, where Θ is a non empty set of sorts. K is the bigraph signature, and Φ is a set of formation rules associated with a bigraph. The latter consists of a set of structural constraints a bigraph has to satisfy. Note that disjunctive sorts are written as $x\hat{y}$ which means a node can either be of sort x or y.

Bigraphical Reaction Rules. Dynamic behavior in terms of system evolution is defined in BRS via rewrite rules, called bigraphical reaction rules. A reaction rule $\mathcal{R}(p) = (R, R')$ consists of a pair of bigraphs referred to as *redex* and *reactum* and can be written by $R \rightarrow R'$. The redex on the left-hand side represents the parts of a bigraph to be altered, while the reactum on right-hand side specifies how those parts are altered. The transition is triggered on a bigraph G when R has an occurrence or a match in G. The result of the rule application is a new bigraph G' obtained by inserting R in G to replace R'.

3 Bigraphical Model of Fog Systems

In this section, we formally define bigraphs used to specify a Fog system in our model. We describe the way sorted bigraphs are used to capture the semantics of such systems. The model is extended via parametric reaction rules explaining the evolution of a Fog system over time and modeling self-adaptation.

3.1 Formalizing Fog Systems

A Fog-based system is modeled in our approach by a bigraph FS composed of the parallel product of two elementary bigraphs representing the Fog and Cloud layers. The parallel product term usage characterizes our model by a genericness and extendability, making it easily scalable and modifiable. This bigraphical term offers the possibility to include additional bigraphs representing the external environment as other Cloud or Fog service providers and thus an ability to represent different systems architectures (e.g. federation of Fogs).

The proposed formalization is achieved by associating a precise semantics in the theory of BRS to each Fog architectural element. A Fog system and its two Cloud and Fog layers are formalized via bigraphs. The different Cloud instances and Fog nodes are expressed with bigraphical nodes and the various kinds of interactions are modeled by edges. Sites represent parts of a Fog system that we wish to hide or abstract away. The proposed formalization helps in exemplifying the different parts of a Fog-based system, as well as exposing the conceptual entities involved in such systems' typical architecture[4].

[4] In order to keep the definition succinct, we omit the complete formal definitions of Fog and Cloud bigraphs, as they can be redundant.

Definition 1. Let FL be a bigraph modeling the Fog layer of a Fog system and CL be a bigraph modeling its Cloud layer, a Fog system bigraph FS is composed by applying the parallel product term $\|$ on FL and CL (see Equation 1).

$$FS \stackrel{\text{def}}{=} CL \| FL \tag{1}$$

Definition 2. A Fog system bigraph is defined formally in our approach by

$$FS = (V_{FS}, E_{FS}, ctrl_{FS}, FS^P, FS^L) : I_{FS} \rightarrow J_{FS} \tag{2}$$

- $V_{FS} = V_{FL} \uplus V_{CL}$ is a finite set of nodes of a Fog system FS obtained from the disjoint union of the sets of nodes of the Fog FL and Cloud layers CL.
- $E_{FS} = E_{FL} \uplus E_{CL}$ is a finite set of edges given by the disjoint union of sets of edges of the Fog FL and Cloud CL layers.
- $ctrl_{FS} = ctrl_{FL} \uplus ctrl_{CL} : V_{FS} \rightarrow \mathcal{K}_{FS}$ is a control mapping function assigning to each node $v \in V_{FS}$ of the bigraph FS representing a Fog system, a control $k \in \mathcal{K}_{FS}$ defining its type. A signature $\mathcal{K}_{FS} = \mathcal{K}_{FL} \uplus \mathcal{K}_{CL}$ is the set of controls defined for FS.
- $FS^P = FL^P || CL^P$ is the place graph of FS obtained by the parallel product of place graphs of the Fog layer FL and Cloud layer CL.
- $FS^L = FL^L || CL^L$ link graph of FS consisting of the parallel product of the link graphs of FL and CL.
- $I_{FS} = I_{FL} || I_{CL} = \langle m_{FS}, X_{FS} \rangle, m_{FS} = m_{FL} + m_{CL}$ and $X_{FS} = X_{FL} \cup X_{CL}$ is the inner interface of the bigraph FS. m_{FS} is the number of sites and X_{FS} is a set of inner names of FS.
- $J_{FS} = J_{FL} || J_{CL} = \langle n_{FS}, Y_{FS} \rangle, n_{FS} = n_{FL} + n_{CL}$ and $Y_{FS} = Y_{FL} \cup Y_{CL}$, is the outer interface of FS, where n_{FS} and Y_{FS} are respectively the number of regions and the set of outer names of FS.

3.2 Modeling Fog Systems with Sorted Bigraphs

We now outline how the structural and conceptual aspects of a given Fog system are encoded through sorted bigraphs. Accordingly, we define a sorting discipline for the class of bigraphs used to describe a Fog system architecture in our model. A sorting discipline provides the proper semantics to represent faithfully a Fog architecture. Besides, it imposes the necessary structural constraints via formation rules to preserve architectural integrity and ensure bigraphs are designed in a meaningful way.

The sorting discipline for the bigraph FS is defined by $\Sigma_{FS} = \{\Theta_{FS}, K_{FS}, \Phi_{FS}\}$, where $\Theta_{FS} = \Theta_{FL} \uplus \Theta_{CL}$, $K_{FS} = K_{FL} \uplus K_{CL}$ and $\Phi_{FS} = \Phi_{FL} \uplus \Phi_{CL}$. The controls of the set K_{FS} we use to represent a Fog-based system listed in Table 2 are organized into different sorts of the set Φ_{FS}. In greater details, sorts $l = \{L\}$ and $p = \{P\}$ contains controls used to model node clusters L and the connection points between them, representing their neighboring relationship. The controls of sort $n = \{N, N_L, N_U, N_U\}$ encode a Fog node in its different possible states. Following the same principle, sorts $s = \{SE, SE_L, SE_U, SE_F\}$, $v = \{VM, VM_L, VM_U\}$ and $t = \{CN, CN_L, CN_U\}$ represent respectively different states of a server, virtual machine and container. A service or micro-service

is modeled with nodes of sort $m = \{s\}$. The various entities computing capacities such as memory, CPU, battery, and storage as well as services requirements in terms of computing resources are encoded respectively by nodes of sort $a = \{M\langle v\rangle, C\langle v\rangle, B\langle v\rangle, S\langle v\rangle\}$ and $q = \{M_r\langle v\rangle, C_r\langle v\rangle, B_r\langle v\rangle, S_r\langle v\rangle\}$ where v is parameter recording the value of each kind of node.

Nodes of sort a are always placed in a node having a sort $c = \{C\}$. Such nesting condition and other kinds constraints are formalized in the set of formation rules Φ_{FS} with conditions Φ_i, $1 \leq i \leq 11$, listed in Table 3. These formation rules are defined to ensure only bigraphs with meaningful structure can be constructed and thus maintain architectural integrity even after self-adaptations. More precisely, conditions Φ_1-Φ_2 are constraints on nodes activity. For instance, Φ_2 states that all nodes having a sort $\widehat{lnsvtmc}$ are active, which means they are subject to reactions. Formation rules Φ_3-Φ_9 are conditions on nodes placing. An example of this kind of nesting rule has been given above. Finally, conditions Φ_{10}-Φ_{11} are linking formation rules. As an example, rule Φ_{11} imposes on a node having an m-sort to be connected to nodes of its own sort or to $\widehat{io}$-names, encoding the bigraph FS ability to interact with the external environment.

Table 2. Controls and sorts used in a bigraph FS.

Meaning	Control	Arity	Sort	Bigraph	Notation
Cluster	L	0	l	FL	Large box
Connection points	P	1	p	FL	Small square
Stable node	N	0	n	FL	Rounded box
Loaded node	N_L	0	n	FL	Rounded box
Underused node	N_U	0	n	FL	Rounded box
Failed node	N_F	0	n	FL	Rounded box
Stable server	SE	0	s	CL	Rounded box
Loaded server	SE_L	0	s	CL	Rounded box
Underused server	SE_U	0	s	CL	Rounded box
Failed server	SE_F	0	s	CL	Rounded box
VM stable	VM	0	v	FL,CL	Rounded box
VM loaded	VM_L	0	v	FL,CL	Rounded box
VM underused	VM_U	0	v	FL,CL	Rounded box
Stable container	CN	0	t	FL, CL	Rounded box
Loaded container	CN_L	0	t	FL, CL	Rounded box
Underused container	CN_U	0	t	FL, CL	Rounded box
Service or micro-service	S	2	m	FL, CL	Circle
Entity capacity	C	0	c	FL, CL	Medium Box
Capacity values	$M\langle v\rangle$, $C\langle v\rangle$,...	0	a	FL, CL	Small circle
Requirement values	$M_r\langle v\rangle$, $C_r\langle v\rangle$,...	0	r	FL, CL	Small circle

An example of a graphical form of a bigraph FS encoding a Fog-based system is given in Figure 2. Note that for the sake of simplicity, controls of sorts p, a and r are purposely omitted in the graphical form. The bigraph FS is obtained by composing FL and CL bigraphs representing Fog and Cloud layers using the parallel product operation. In Figure 2 the Fog and Cloud layers are modeled respectively by regions 0 and 1, depicted with dashed rectangles. Region 0 contains several clusters modeled by nodes having controls $L1 - L4$. A cluster node may be linked to an arbitrary number of other cluster nodes through connection points represented by nodes of control P. These connection points are used to express the neighboring relationship between Fog clusters, such that when two

clusters are directly linked to each other, they are considered as direct or one-hop neighbors. For instance, clusters $L1$ and $L2$ are considered in our model as direct neighbors, while clusters $L2$ and $L4$ are considered as two-hop neighbors. The grey shapes in Figure 2 are sites, used as abstraction for hidden away entities.

Figure 3 is another simple example of a Fog layer bigraph FL modeled with region 0, which shows in depth the structure of a Fog cluster in our model. Sites are used here to abstract away the elements of the bigraphical model that we do not reason about. A Fog cluster L hosts a set a N nodes modelling Fog computing nodes. In the example, nodes of control $N1$, $N2_L$ and $N3_U$ encode respectively stable, loaded and underused Fog nodes. Each node may host a set of application containers modelled with nodes having control CN. The latter may host themselves nodes of control S, modelling application services. The computing capacity of Fog nodes and containers is encoded with a node of control C, grouping nodes representing their respective memory, CPU, battery level, etc. For instance, the computing capacity of Fog node $N1$ represented by a node C within $N1$ can be represented by the following algebraic notation: $\mathtt{C} = \mathtt{C}.(\mathtt{M}\langle 512\rangle|\mathtt{C}\langle 1.1\rangle|\mathtt{B}\langle 10\rangle)|\mathtt{d})$ to express the computing capability of $N1$. Finally, service requirements in terms of processing power are encoded in a similar way by nodes taking the shape of small circles within the node of S control. In our example, the service S requirements are expressed algebraically by: $\mathtt{S} = \mathtt{S}_{\mathtt{fi}}.(\mathtt{M}_{\mathtt{r}}\langle 32\rangle|\mathtt{C}_{\mathtt{r}}\langle 0.1\rangle|\mathtt{B}_{\mathtt{r}}\langle 0.2\rangle)$.

Table 3. Formation rules Φ_{CS} for the bigraph FS

	Rule description
Φ_1	All $\widehat{lnsvtmc}$-nodes are active
Φ_2	All $\widehat{par}$-nodes are atomic
Φ_3	All children of a l-node have sort $\widehat{pm}$
Φ_4	All children of a n-node have sort $\widehat{vtc}$
Φ_5	All children of a s-node have sort $\widehat{vtc}$
Φ_6	All children of a v-node have sort $\widehat{mc}$
Φ_7	All children of a t-node have sort $\widehat{mc}$
Φ_8	All children of a c-node have sort a
Φ_9	All children of a m-node have sort r
Φ_{10}	A p-node is always linked only to one p-node
Φ_{11}	An m-node has two ports which may be linked to m-nodes or $\widehat{io}$-names

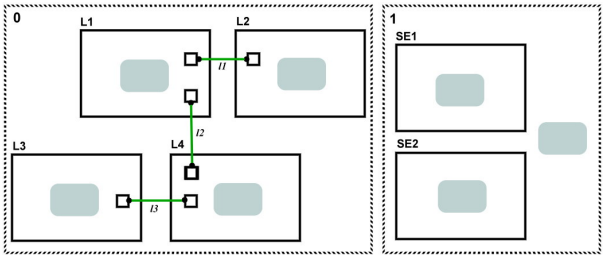

Fig. 2. A simple example of a bigraph FS.

3.3 Modeling Fog Systems Self-Adaptation

A bigraphical reactive system consists of a class of bigraphs modelling conceptual aspects of a system and a set of reaction rules encoding their dynamic behavior. The behavioral model consists of a set of parametric reaction rules giving the Fog system model the ability to self-adapt. Each reaction rule correspond to a reconfiguration action triggering a change that rewrites parts of the bigraph FS following the occurrence of a respective self-adaption event of the Fog system. In the following, we present a set of reaction rules modeling self-adaptation actions carried out in a Fog-based system while preserving the defined sorting discipline Σ_{FS}. Due to the space limitation, here we do not give all the reaction rules in our behavioral model, instead, we present some relevant examples. The exhaustive behavioral model includes additional reaction rules such as virtual machine/container migration and replication, node mobility (relocation), turning on/off a node, service deployment, battery draining, tagging an underused or loaded container or virtual machine, etc. Examples of reaction rules defined in the algebraic terms language dedicated to BRS are summarized in Table 4.

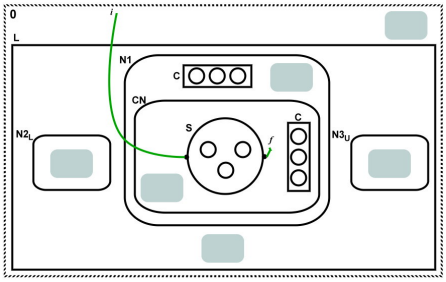

Fig. 3. Example of bigraphical representation for a Fog cluster.

The first example is a reaction rule which models the establishment of a neighboring relationship between two clusters. On the left-hand side of the rule (redex), the two nodes L and L' modelling two Fog clusters are independent of each other and not linked together, meaning that currently, they do not recognize one another as neighbors. On the right-hand side (reactum), two connection points modeled with nodes of control P are created within each cluster node and a communication link l is established between them to represent the new neighboring relationship (see Figure 4.a). Another example of reaction rule for resizing the memory allocated to a loaded or underused computing instance such a loaded container represented by control CN_L is presented in Figure 4.b. The rule is employed to modify the memory parameter of the memory node M within a capacity node C from value x to x'. We consider the memory increased if parameters $x < x'$ and decreased if $x' > x$. Finally, the last example is a reaction rule graphical representation for tagging a node after it fails (see Figure 4.c). The reaction rule principle consists of replacing the node control N on rule's redex by another control N_F on the reactum.

4 Motivating Case Study

4.1 A Fog System Architecture

Assume a Fog-based system distributed over a part of a college campus, specifically, a department and a dormitory. The Fog layer contains a number of small size Fog nodes such as cameras, routers and boxes, as well as larger nodes such as Raspberry Pis and small servers. The infrastructure includes, in addition, a more powerful server in the Cloud layer.

Table 4. Reaction rule examples for the bigraph FS

Reaction rule	Algebraic form
$\texttt{neighbor} - \texttt{link(i,j)}$	$\stackrel{\text{def}}{=} \mathtt{L_i.(d)}\vert\mathtt{L_j.(d)}\vert\mathtt{id} \rightarrow \mathtt{L_i.(P_1}\vert\mathtt{d)}\vert\mathtt{L_j.(P_1}\vert\mathtt{d)}\vert\mathtt{id}$
$\texttt{add} - \texttt{node(i)}$	$\stackrel{\text{def}}{=} \mathtt{L.(d)}\vert\mathtt{N.(d)}\vert\mathtt{id} \rightarrow \mathtt{L.(N_i.(d)}\vert\mathtt{d)}\vert\mathtt{id}$
$\texttt{replicate} - \texttt{vm(i)}$	$\stackrel{\text{def}}{=} \mathtt{VM_i.(d)}\vert\mathtt{id} \rightarrow \mathtt{VM_i.(d)}\vert\mathtt{VM_j.(d)}\vert\mathtt{id}$
$\texttt{consolidate} - \texttt{vm(j)}$	$\stackrel{\text{def}}{=} \mathtt{VM_i.(d)}\vert\mathtt{VM_j.(d)}\vert\mathtt{id} \rightarrow \mathtt{VM_i.(d)}\vert\mathtt{id}$
$\texttt{resize} - \texttt{memory(x,x',i)}$	$\stackrel{\text{def}}{=} \mathtt{CN_{Li}.(C.(M\langle x\rangle}\vert\mathtt{d)}\vert\mathtt{d)}\vert\mathtt{id} \rightarrow \mathtt{CN_{Li}.(C.(M\langle x'\rangle}\vert\mathtt{d)}\vert\mathtt{d)}\vert\mathtt{id}$
$\texttt{resize} - \texttt{cpu(x,x',i)}$	$\stackrel{\text{def}}{=} \mathtt{CN_{Li}.(C.(C\langle x\rangle}\vert\mathtt{d)}\vert\mathtt{d)}\vert\mathtt{id} \rightarrow \mathtt{CN_{Li}.(C.(C\langle x'\rangle}\vert\mathtt{d)}\vert\mathtt{d)}\vert\mathtt{id}$
$\texttt{failed} - \texttt{node(i)}$	$\stackrel{\text{def}}{=} \mathtt{N_i.(d)}\vert\mathtt{id} \rightarrow \mathtt{N_{Fi}.(d)}\vert\mathtt{id}$

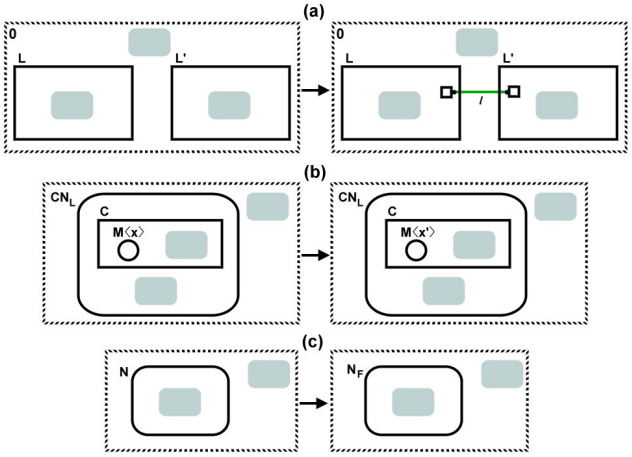

Fig. 4. Reaction rules for (a) Establishing neighboring relationship (b) Resizing memory (c) Tagging a failed node.

Consider two smart city Fog applications, designed as a set of micro-services distributed across multiple Fog cluster on different nodes. The first application is a *smart surveillance* application [8], proposing a near site Edge/Fog-based solution for video analysis. Micro-service architecture is adopted to decouple the complicated video analysis tasks such as facial features extraction, car recognition and real-time behavior analysis. Five services are dedicated to such tasks, divided into low-level tasks deployed in the smaller nodes of the Edge infrastructure and computing intensive or high-level tasks outsourced to the bigger

Fog nodes or to the Cloud. The *smart surveillance* application is composed of five micro-service, low-level ones: (S1) Feature extraction, (S2) Audio analysis, (S3) Licence plate recognition. High-level services include (S4) Gesture and facial recognition and (S5) Suspicious behavior analysis. The second application in our example is a *smart bell* application [13], which notifies the college dormitory inhabitants when they receive visitor(s). The micro-services composing the application are classified similarly into low-level and high-level services. The low-level services are (S1) Feature extraction, (S6) Bell ringing decision, and the high-level services are (S4) Gesture and facial recognition, (S7) Visitor habits analysis. Note that some micro-services in the two applications are identical.

The presented Fog-based infrastructure deploying the two Fog applications can be encoded in our model by the formulae below. Due to the space limitation, we give only an abbreviated algebraic form of the example bigraph FS where we abstract away some nodes using sites. The bigraph model encodes essentially three clusters L^1, L^2 and L^3 containing 12 nodes (N^1-N^{12}) used to host mostly low-level services S^1, S^2, S^3, S^6 and a high-level service S^4 at the Fog layer (FL). A neighboring relationships exists between clusters L^1, L^2 and L^2, L^3. A Cloud server SE within a CL-bigraph is hosting two of the high-level services S^5, S^7. Note that a micro-service can have several instances representing its workers (e.g. $\mathtt{S}^1, \mathtt{S}^{1-1}, \mathtt{etc.}$) deployed in containers $CN^1 - CN^{21}$ and virtual machine VM^1, VM^2. In the algebraic representation we occasionally omit some workers and links between services for presentation reasons.

$\mathtt{FS} \stackrel{\text{def}}{=} \mathtt{/l1/l2/f1/f2/f3/f4\ FL||CL}$

$\mathtt{FL} \stackrel{\text{def}}{=} (\mathtt{L}^1.(\mathtt{P}_{11}|\mathtt{d})|\mathtt{L}^2.(\mathtt{P}_{11}|\mathtt{P}_{12}|\mathtt{d})|\mathtt{L}^3.(\mathtt{P}_{12}|\mathtt{d})$

$\mathtt{L}^1 \stackrel{\text{def}}{=} (\mathtt{N}^1.(\mathtt{CN}^1.(\mathtt{S}^2.(\mathtt{d})|\mathtt{d})|\mathtt{CN}^2.(\mathtt{S}^{2-1}.(\mathtt{d})|\mathtt{d})|\mathtt{d})|\mathtt{N}^2.(\mathtt{CN}^3.(\mathtt{S}^{2-2}.(\mathtt{d})|\mathtt{d})|\mathtt{CN}^4.(\mathtt{S}^{2-3}.(\mathtt{d})|\mathtt{d})|$
$\mathtt{CN}^5.(\mathtt{S}^3.(\mathtt{d})|\mathtt{d})|\mathtt{d})|\mathtt{N}^3.(\mathtt{CN}^6.(\mathtt{S}^{3-1}.(\mathtt{d})|\mathtt{d})|\mathtt{CN}^7.(\mathtt{S}^{3-2}.(\mathtt{d})|\mathtt{d})|\mathtt{d})|\mathtt{N}^4.(\mathtt{CN}^8.(\mathtt{S}^{3-3}.(\mathtt{d})|\mathtt{d})|\mathtt{d})|\mathtt{d})$

$\mathtt{L}^2 \stackrel{\text{def}}{=} (\mathtt{N}^5.(\mathtt{CN}^9.(\mathtt{S}^1.(\mathtt{d})|\mathtt{d})|\mathtt{CN}^{10}.(\mathtt{S}^{1-1}.(\mathtt{d})|\mathtt{d})|\mathtt{CN}^{11}.(\mathtt{S}^{1-2}.(\mathtt{d})|\mathtt{d})|\mathtt{d})|\mathtt{N}^6.(\mathtt{CN}^{12}.(\mathtt{S}^{1-3}.(\mathtt{d})|\mathtt{d})|\mathtt{CN}^{13}.$
$(\mathtt{S}^{1-4}.(\mathtt{d})|\mathtt{d})|\mathtt{d})|\mathtt{N}^7.(\mathtt{CN}^{14}.(\mathtt{S}^4.(\mathtt{d})|\mathtt{d})|\mathtt{CN}^{15}.(\mathtt{S}^{4-1}.(\mathtt{d})|\mathtt{d})|\mathtt{d})|\mathtt{d})$

$\mathtt{L}^3 \stackrel{\text{def}}{=} (\mathtt{N}^8.(\mathtt{CN}^{16}.(\mathtt{S}^{1-5}.(\mathtt{d})|\mathtt{d})|\mathtt{d})|\mathtt{N}^9.(\mathtt{CN}^{17}.(\mathtt{S}^{1-6}.(\mathtt{d})|\mathtt{d})|\mathtt{d})|\mathtt{N}^{10}.(\mathtt{CN}^{18}.(\mathtt{S}^{1-7}.(\mathtt{d})|\mathtt{d})|\mathtt{CN}^{19}.(\mathtt{S}^6.$
$(\mathtt{d})|\mathtt{d})|\mathtt{d})|\mathtt{N}^{11}.(\mathtt{CN}^{20}.(\mathtt{S}^{6-1}.(\mathtt{d})|\mathtt{d})|\mathtt{d})|\mathtt{N}^{12}.(\mathtt{CN}^{21}.(\mathtt{S}^{6-2}.(\mathtt{d})|\mathtt{d})|\mathtt{d})|\mathtt{d})$

$\mathtt{CL} = \mathtt{SE1}.(\mathtt{VM1}.(\mathtt{S}^5_{\mathtt{f2}}.(\mathtt{d})|\mathtt{S}^{5-1}_{\mathtt{f2}}.(\mathtt{d})|\mathtt{S}^{5-2}_{\mathtt{f2}}.(\mathtt{d})|\mathtt{d})|\mathtt{VM2}.(\mathtt{S}^7_{\mathtt{f4}}.(\mathtt{d})|\mathtt{S}^{7-1}_{\mathtt{f4}}.(\mathtt{d})|\mathtt{d})|\mathtt{d})$

4.2 Self-Adaptation Strategies

Concentration of students and various persons such as visitors present at a college campus differ around the year. For instance, during the first and second semesters, more people are expected to be on the campus, while at the holidays very few are there. The estimate of students may as well fluctuate between the semesters since a portion of students usually does an internship outside of the college in the second semester. Besides, the number of people outside of dormitory varies around the day: more people are outdoors in the morning and more are indoors as the day goes by. Considering these different situations, it is obvious that the two Fog applications will have fluctuated workload very often. As computing nodes at the Fog are mostly resources-scarce, occasionally energy-constrained and unreliable, a Fog-based system must possess an ability to self-adapt at run-time in order to better optimize resources consumption,

according to the context variation and depending on the workload and nodes availability. Here, we demonstrate the use of our defined bigraphical reaction rules to model self-adaptation strategies, which can be used in such systems in order to get around resources deficiency distinguishing computing nodes at the Fog. These reaction rules can be used to describe all possible evolution scenarios and reachable configurations of a Fog system in terms of self-adaptation in two orthogonal ways to rewrite an FS bigraph: iterative application of each reaction rule at a time and as a sequence of reaction modeling self-adaptation strategies.

Assume two self-adaptation strategies controlling the Fog system and applications of our case study. The first strategy operates at the application level to scale out/in the two Fog applications horizontally. It scales out by creating additional workers for their respective micro-services when the demand average peaks and scales in by destroying workers when it drops. Two action plans modeling this adaptation strategy as sequences of parametric reaction rule that updates the bigraph model from a state S_n to S_{n+1} are encoded by formulae (a) and (b). The first action plan AP_a contains the reaction for creating additional workers for micro-services S1, S2, S3, S4 and S6, which we consider as the most subject to be affected by the workload augmentation. The action plan AP_b contains rules for consolidation the unneeded workers of the aforementioned micro-services when they are unneeded. We use $R*$ to indicate an arbitrary number of applications of a reaction (zero or more). $\longrightarrow\!\!\!\!\triangleright$ notation indicates a transition between reaction rules. A rule parameter i is used to model the index of the respective service e.g. add-worker(i), such that in our example $1 \leq i \leq 7$.

$$\mathtt{AP_a} \stackrel{\text{def}}{=} \mathtt{add-worker(1)}^* \longrightarrow\!\!\!\!\triangleright\ \mathtt{add-worker(2)}^* \longrightarrow\!\!\!\!\triangleright\ \mathtt{add-worker(3)}^* \longrightarrow\!\!\!\!\triangleright\ \mathtt{add-worker(4)}^* \longrightarrow\!\!\!\!\triangleright\ \mathtt{add-worker(6)}^* \quad \text{(a)}$$

$$\mathtt{AP_b} \stackrel{\text{def}}{=} \mathtt{release-worker(1)}^* \longrightarrow\!\!\!\!\triangleright\ \mathtt{release-worker(2)}^* \longrightarrow\!\!\!\!\triangleright\ \mathtt{release-worker(3)}^* \longrightarrow\!\!\!\!\triangleright\ \mathtt{release-worker(4)}^* \longrightarrow\!\!\!\!\triangleright\ \mathtt{release-worker(6)}^* \quad \text{(b)}$$

The second adaptation strategy concerns vertical resizing of containers hosting instances of the different micro-services. Based on resources consumption level of the hosted instances, the strategy adjusts the size of a given container by allocating to it additional computing resources. In our example, we assume a scenario where the size of a given containers (parameter i is used to determine the concern containers) needs to be adjusted to better accommodate the requirements of their hosted workers in terms of computing power. The reaction rules sequence modelling such strategy which resizes certain containers up or down is given by action plans AP_c and AP_d of formulae (c) and (d) respectively. The AP_c reaction rules sequence allocates additional memory, CPU and storage space to the loaded containers, while AP_d scales down the underused ones.

$$\mathtt{AP_c} \stackrel{\text{def}}{=} \mathtt{loaded-container(i)}^* \longrightarrow\!\!\!\!\triangleright\ \mathtt{resize-memory(x,x',i)}^* \longrightarrow\!\!\!\!\triangleright\ \mathtt{resize-cpu(x,x',i)}^* \longrightarrow\!\!\!\!\triangleright\ \mathtt{resize-storage(x,x',i)}^* \quad \text{(c)}$$

$$\mathtt{AP_d} \stackrel{\text{def}}{=} \mathtt{underused-container(i)}^* \longrightarrow\!\!\!\!\triangleright\ \mathtt{resize-memory(x,x',i)}^* \longrightarrow\!\!\!\!\triangleright\ \mathtt{resize-cpu(x,x',i)}^* \longrightarrow\!\!\!\!\triangleright\ \mathtt{resize-storage(x,x',i)}^* \quad \text{(d)}$$

In a Fog/Edge infrastructure, node failure is a very common phenomenon that may be caused by environmental conditions. In the example of our case study, nodes deployed in open air exposed to rough weather or nodes equipped

with intermittent energy sources (e.g. solar panels) such as cameras are typically subject to failure [3]. In nodes failure scenario, a strategy may be defined to self-adapt a Fog system by replacing the failed nodes with backup nodes or newly installed ones. Node failure event may be modelled with the reaction rules sequence of formulae (e), nodes that get tagged failed in any given cluster would be replaced with backup nodes (if possible) to ensure the two Fog applications continue to be available, especially the safety-critical application of *smart surveillance*.

$\mathtt{AP_d} \stackrel{\text{def}}{=} \mathtt{failed} - \mathtt{node(i)}^* \multimap \mathtt{add} - \mathtt{node(i)}^*$ (e)

4.3 Constraints and Properties

Now we present how we can employ the notion of bigraphical patterns introduced in [1], to express constraints and properties characterizing Fog application requirements and relevant behaviors of self-adaptive Fog systems.

Application constraints. First, we present the formalization of Fog applications specific constraints in terms of state properties. The latter are expressed as predicated via logical formulae obtained by combining bigraph patterns with the basic boolean operators. The different predicates may define constraints on resource requirements, interactions between applications, micro-services placement, etc. These state properties are expected to be satisfied after every Fog system (re)configuration such that if a bigraphical pattern has a match in a given bigraph FS modelling a Fog system, it is considered true. Next, we give two examples of such constraints.

Battery constraint. A Fog application may have a requirement specifying that the battery level of nodes on which its micro-services are deployed is not supposed to fall below a minimum threshold. For instance, critical service $S4$ of both applications in our example, may required a minimum number of hosting nodes to be always available in order to operate correctly. A strategy such as AP_e can serve to satisfy this requirement continuously. This constraint is expressed by the predicate Ψ_1. The constraint is satisfied if the battery level modeled by a control $B_i\langle v\rangle$ of each node is always above a minimum threshold. $N_{app} \subseteq V_{FS}$ is a set of nodes hosting certain application services.

$\Psi_1(\mathtt{v}, \mathtt{min}, \mathtt{i}) \stackrel{\text{def}}{=} \neg \mathtt{B_i}\langle \mathtt{v}\rangle$, such that $v < min$ and $i \in N_{app}$

Application Placement. To ensure secure placement of different application, a Fog system may have a constraint dictating the placement of certain micro-services belonging to different applications in different nodes, servers or virtual machines. For instance, a predicate is given below, formalizing a mutual exclusion between micro-services $S5$ and $S7$ of the *smart surveillance* and *smart bell* applications and their workers to ensure they do not co-existed on the same VM.

$\Psi_2(\mathtt{x}, \mathtt{i}, \mathtt{j}) \stackrel{\text{def}}{=} (\mathtt{VM_x}.(\mathtt{S_5^i}) \wedge \neg \mathtt{VM_x}.(\mathtt{S_7^i})) \vee (\neg \mathtt{VM_x}.(\mathtt{S_5^i}) \wedge \mathtt{VM_x}.(\mathtt{S_7^i}))$

Behavioral Properties. To express temporal properties over the run-time behavior of Fog systems, it is possible to combine bigraph patterns not just with simple boolean operators, but with more expressive temporal logics such as Linear Temporal Logic (LTL in short) [4]. Here we formalize relevant behavioral properties of self-adaptive Fog system by integrating bigraph patterns defined

over our model as atomic propositions with temporal operators of LTL. For a given atomic proposition ϕ, we write $\circ\phi$, $\Box\phi$, $\Diamond\phi$, $\phi_1 \cup \phi_2$ to denote respectively the temporal operators "next", "always", "eventually" and "until". A temporal property should be true for all possible future evolutions of a Fog system model after the application of bigraphical reaction rules. We present two examples of such temporal properties expressed as LTL formulae.

Node replacement. A habitual scenario in a Fog or Edge infrastructure is node failure due to environmental conditions or energy constraints. Thus, to ensure the Fog system stays operational, a failed node should be replaced by a backup or newly installed node. The property specifies the fact that if a node tagged with control N_F in our model exists, it will be replaced or repaired. The property can be used to evaluated a node replacing strategy such as AP_e.

$\Psi_3(\mathtt{x},\mathtt{y}) \stackrel{\text{def}}{=} \Box(\mathtt{L_x}.(\mathtt{N_{yF}}|\mathtt{d}) \rightarrow \Diamond(\mathtt{L_x}.(\mathtt{N_y}|\mathtt{d})))$

Resource optimization. Because of nodes resource scarcity, a Fog system needs to control resource consumption and better optimize their utilization. For instance, an underused VM should be either used more, consolidated or scaled down in order to better utilize computing capabilities of the hosting server. Such property can validate the efficiency of a horizontal consolidation strategy such as AP_b. It is encoded with the LTL formulae below. Informally, it states that it should be always the case if an underused VM exists in the Fog system, i.e. a node tagged with control VM_U in our model, eventually it will be in a stable state, consolidated or resized.

$\Psi_4(\mathtt{x},\mathtt{y},\mathtt{v},\mathtt{v}',\mathtt{w},\mathtt{w}') \stackrel{\text{def}}{=} \Box(\mathtt{SE_x}.(\mathtt{VM_{yU}}.(\mathtt{M}\langle\mathtt{v}\rangle|\mathtt{C}\langle\mathtt{w}\rangle|\mathtt{d})|\mathtt{d}) \rightarrow \Diamond((\mathtt{SE_x}.(\mathtt{VM_y}.(\mathtt{d})))\vee$
$(\mathtt{SE_x}.(\mathtt{VM_y}.(\mathtt{M}\langle\mathtt{v}'\rangle|\mathtt{C}\langle\mathtt{w}'\rangle|\mathtt{d})|\mathtt{d})) \vee (\mathtt{SE_x}.(\mathtt{d}))$

5 Conclusion and Perspectives

Modeling a Fog system spatial distribution and temporal evolution in terms of self-adaption requires an adequate formal foundation that captures such aspects and allows automated computation and analysis of properties. In this paper, we have demonstrated how BRS can present a suitable formalism to structurally and behaviorally model Fog-based systems. Our primary motivation is to concentrate on proposing an extendable formal model capable of representing faithfully self-adaptive Fog systems and related properties. Beyond this scope, formal verification of the presented Fog systems spatio-temporal properties is an important perspective of ours, which we intend to address with the appropriate attention. Our aim is to employ the proposed model as a basis for reasoning about various properties including the ones given above using high-level automated formal analysis techniques. The proposed reaction rules can be applied to generate a sequence of bigraphical models representing past and current states of the Fog system. After every each adaptation, the resulting models can be checked whether certain properties are maintained or violated in the current configuration.

We are considering two kinds of formal analysis approaches: (i) real-time verification of state properties expressed via bigraph patterns such as Fog applications requirements and constraints, (ii) offline model-checking of temporal properties over the behavior of a Fog system. The idea is to construct a framework allowing not only to evaluate self-adaptation strategies by detecting prop-

erties violation rate in each strategy but also to identify conflicting ones and report details. In the real-time verification approach, it is possible to use a real Fog system (or a simulator) generating self-adaption events that trigger the application of reaction rule sequences. When applied, the latter would rewrite the current bigraphical model to yield a new state that reflects the actual configuration of the system. State properties can be checked after each self-adaption over the most recent model to verify their satisfaction or violation. The offline verification approach can be employed at the early stages of a Fog system design or at the application deployment time. The objective is to obtain formal assurance about the system through model-checking of temporal properties to verify the absence of unwarranted system behaviors. The model-checking technique is based on a temporal logic which allows reasoning over different computation paths to explore all possible states of the system. This feature provides better coverage but is more computationally expensive, which makes it more suitable for the offline approach rather than for the real-time one (i.e. high analysis overhead).

References

1. Benford, S., Calder, M., Rodden, T., Sevegnani, M.: On lions, impala, and bigraphs: Modelling interactions in physical/virtual spaces. ACM Trans. Comput.-Hum. Interact. **23**, 9:1–9:56 (2016)
2. Calder, M., Koliousis, A., Sevegnani, M., Sventek, J.: Real-time verification of wireless home networks using bigraphs with sharing. Science of Computer Programming **80**, 288 – 310 (2014)
3. Fang, X., Bate, I.: Issues of using wireless sensor network to monitor urban air quality. In: Proceedings of the 1st ACM Int. Workshop on the Engineering of Reliable, Robust, and Secure Embedded Wireless Sensing Systems. pp. 32–39. FAILSAFE'17, ACM, New York, NY, USA (2017)
4. Huth, M., Ryan, M.: Logic in Computer Science: Modelling and Reasoning About Systems. Cambridge University Press, New York, NY, USA (2004)
5. Mansutti, A., Miculan, M., Peressotti, M.: Multi-agent systems design and prototyping with bigraphical reactive systems. In: Distributed Applications and Interoperable Systems. pp. 201–208. Springer Berlin Heidelberg (2014)
6. Milner, R.: The Space and Motion of Communicating Agents. Cambridge University Press, New York, NY, USA, 1st edn. (2009)
7. Mouradian, C., Naboulsi, D., Yangui, S., Glitho, R.H., Morrow, M.J., Polakos, P.A.: A comprehensive survey on fog computing: State-of-the-art and research challenges. IEEE Communications Surveys Tutorials **20**(1), 416–464 (2018)
8. Nagothu, D., Xu, R., Nikouei, S.Y., Chen, Y.L.: A microservice-enabled architecture for smart surveillance using blockchain technology. 2018 IEEE Int. Smart Cities Conference (ISC2) pp. 1–4 (2018)
9. Sahli, H., Belala, F., Bouanaka, C.: A brs-based approach to model and verify cloud systems elasticity. Procedia Computer Science **68**, 29 – 41 (2015), 1st Int. Conference on Cloud Forward: From Distributed to Complete Computing
10. Sahli, H., Hameurlain, N., Belala, F.: A bigraphical model for specifying cloud-based elastic systems and their behaviour. Int. Journal of Parallel, Emergent and Distributed Systems **32**(6), 593–616 (2017)

11. Sevegnani, M., Kabac, M., Calder, M., McCann, J.: Modelling and verification of large-scale sensor network infrastructures. In: 2018 23rd Int. Conference on Engineering of Complex Computer Systems (ICECCS). pp. 71–81 (Dec 2018)
12. Tsigkanos, C., Kehrer, T., Ghezzi, C.: Modeling and verification of evolving cyber-physical spaces. In: Proceedings of the 11th Joint Meeting on Foundations of Software Engineering. pp. 38–48. ESEC/FSE 2017, ACM, NY, USA (2017)
13. Xia, Y., Etchevers, X., Letondeur, L., Coupaye, T., Desprez, F.: Combining hardware nodes and software components ordering-based heuristics for optimizing the placement of distributed iot applications in the fog. In: 33rd Annual ACM Symposium on Applied Computing. pp. 751–760. SAC '18, ACM, NY, USA (2018)
14. Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J., Jue, J.P.: All one needs to know about fog computing and related edge computing paradigms: A complete survey. Journal of Systems Architecture (2019)