Verifying Cloud Systems using a Bigraphical Maude-based Model Checker

Zakaria Benzadri, Chafia Bouanaka and Faiza Belala

LIRE Laboratory, Department of Software and Information Systems Technology,
University of Constantine II, Constantine, Algeria

Abstract. Formal methods are system design techniques based on rigorous mathematical models to build software systems. Our aim in this paper is to define formal specifications of cloud systems and offer analysis support to formally model check their inherent properties. Hence, we define a formal semantic framework, based on Bigraphical Reactive Systems (BRS), for specifying cloud architectures and their shape shifting to ensure service availability and quick scalability. Then, we propose a Bigraphical Maude-based Model Checker to formally verify some properties.

1 Introduction

Complexity is in the core of today's Cloud systems, since large organizations have to manage increasingly heterogeneous cloud systems at different level: hundreds of services, thousands of servers, hundreds of thousands of transactions, more and more numerous and spread over the world users, whether customers or suppliers. The inexorable increase in the scope of cloud computing is accompanied by a concentration of risks and brought issues.

The evolutionary pressure imposed upon organizations, and the ever-increasing integration of dematerialization only accelerate this phenomenon and give a critical role to cloud systems. Adverse effects of such complexity are numerous, giving rise to many obstacles [1], such as service availability, performance unpredictability, quick scalability, bugs in large-scale distributed systems, etc., that handicap cloud computing adoption and growth. In this context, formal methods are an effective approach to ensure cloud systems reliability. Since, formal methods allow developing complex software under a firm mathematical foundation resulting in high quality, more correct software compared to conventional design methods. Nevertheless, some attempts have been proposed but do not deal with all fundamental concepts of cloud computing. Particularly, they focus on cloud computing technological concepts;

– Hanfei, et al. [2] and Grandison, et al. [3], gave some discussion and exploration on establishing relationships between virtualization and cloud computing. Throughout their work, they attempt to give out a formal definition of cloud computing from a virtualization viewpoint using its theoretical basic concepts.

Benzadri Z., Bouanaka C. and Belala F..
Verifying Cloud Systems using a Bigraphical Maude-based Model Checker.
DOI: 10.5220/0004978800030012
In *Proceedings of the International Workshop on Emerging Software as a Service and Analytics* (ESaaSA-2014), pages 3-12
ISBN: 978-989-758-026-0

- Shi-Xin, et al. [4], propose an access control model to achieve a fine-grained, data confidentiality and scalability via a formal definition of the HABAC model (Hierarchy Attribute-Based Access Control).
- Adamov and Hahanov [5], define a security model for individual cyberspace (ICS) protection as a means to ensure a secured user's virtual environment, they establish an analysis of security issues related to ICS and propose a conceptual model for modern security environments.
- Freitas and Paul [6], present an abstract formalisation of federated cloud workflows using the Z notation. They define various properties using rules restricting valid options in two categories: security and cost.
- Binz, et al. [7], propose Enterprise Topology Graphs (ETG) as formal model to describe an enterprise topology. Based on the established graph theory, ETG bring formalization and provability to the cloud. Also, authors show how ETG can improve the environmental impact of the enterprise IT.

Consequently, cloud computing paradigm lacks a formal and verifiable model of its basic concepts: service delivery and deployment models, only some technological attempts are realized; for virtualization as it has been done in [2] and [3], for security as in [4], [5] and [6], or for enterprise IT as in [7].

In this work, we adopt, Bigraphical Reactive Systems (BRS), proposed by Milner [8] , as a formalism to specify fundamental aspects of cloud computing. In particular, we propose a theoretical framework based on BRS aiming to formalize relationships between service providers and customers in different delivery and deployment models of cloud computing. Based on a judicious coupling between bigraphs theory as a semantic framework and Maude language[9] as a specification language, we propose the Bigraphical Maude-Based Model Checker (BMMC) that combines logical reflection and hierarchical structuring of the underlying adopted theoretical framework to simply execute and verify complex cloud systems. The proposed approach is validated through a concrete verification of two cloud computing inherent properties: service availability and quick scalability.

This paper is arranged as follows. In the next section we define the bigraphical model for cloud computing. Section 3 presents a mapping of the defined model to Maude-based specification and exploits it to formally verify cloud systems. Finally, conclusion and future work are addressed in section 4.

2 BRS for Cloud Computing Specification

A BRS (Bigraphical Reactive Systems) [8] consists of a category of bigraphs and a set of reaction rules that may be applied to transform these bigraphs. The bigraph as an ordinary graph is composed of nodes and edges, unlike nodes in a bigraph can be nested giving rise to hierarchical and larger bigraphs. Additionally, it is the resulting graph of composing a link graph; representing interconnection between nodes, and a place graph; expressing the physical locations of theses nodes; hence the prefix "bi" in bigraph.

In this work, we adopt BRS for two reasons. On the one hand, our model emphasizes on both locality and connectivity that can be used to specify cloud elements location and interconnection (structural aspects)[10]. And on the other hand, bigraphical reaction rules are very useful to formalize cloud services elasticity providing them the ability to reconfigure themselves (dynamic aspects).

2.1 Structural Aspects

Main elements of a cloud architecture can be divided into two essential parts: front-end and back-end elements; that are usually connected via internet. The Front-end encloses customer's computer and necessary interfaces to access the cloud and the back-end contains cloud services. So, BRS modelling of cloud elements is realized as follows:

Cloud Services. A Cloud Service represents an abstraction of three different service delivery models that collaborate to ensure front end requests [11] Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Albeit, all available cloud services will be modelled by nodes, controls (kinds of node[8]) attached to them allow us to distinguish between the three service delivery models. We suggest that nodes of control IaaS and PaaS are active (node permitting reactions inside), and nodes of control SaaS are atomic (empty node). To offer several access modes to cloud services, we propose that each node or cloud service contains three different ports: Public, Private, and Community.

Cloud Customers. A Cloud Customer represents any entity that requests a cloud service. Two types of cloud customers are identified: End users accessing only SaaS, and ISV (Independent Software Vendor) accessing IaaS and PaaS. Thus, we model Cloud Customers as nodes equipped with specific controls to be able to distinguish between both types of Cloud Customers; End user and ISV. Both are atomic nodes and have one port to send their requests.

Cloud Architecture. Interconnecting cloud computing elements is modelled via two distinct relations: hierarchy for cloud services imbrication (IaaS, PaaS and SaaS) and interaction for service request. Formally, a BRS offers two types of independent graphs: Place graph and Link graph, that will be exploited as follows:

Place Graph. formally expresses cloud services and customers location. Figure 1 represents the hierarchy of cloud services within a place graph, where a node of control IaaS (infrastructure) can only contain nodes of control PaaS (platform), also, a node of control PaaS can only contain nodes of control SaaS (software), finally, a node of control SaaS do not contain any nodes.

Link Graph. is independent of locality and expresses service interactions, in terms of service demand/response relationship between cloud services and customers. Link graph of Figure 1 contains five customers: C1, C2, C3, C4, C5 and four cloud services: S1, S2, S3, S4 interconnected as follows: cloud customer C2 is relied to

cloud service S1 via its community port (green color), and cloud customers C4 and C5 are relied to cloud service S4, the first one with a public port (blue color) and the second one with a private port (red color).

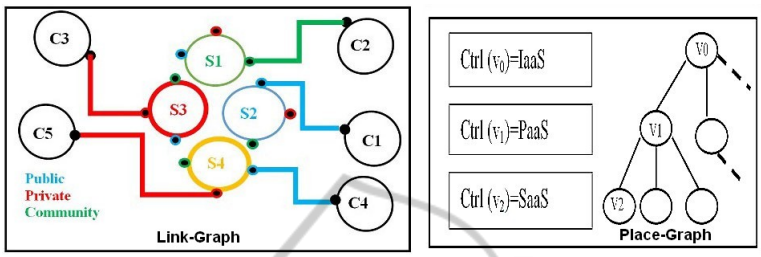

Fig. 1. Place and Link Graphs of a Cloud Architecture.

Composing link and place graphs gives rise to the suited bigraph as defined in what follows:

Definition 1. Cloud Architecture Bigraph takes the form:

$$(Vc, Vs, E, Ctrlc, Ctrls, GP, GL)$$

where: Vc represents the set of cloud customer nodes, and Vs represents all cloud service nodes. E is a finite set of edges. Each node has a control(kinds of node), which is an identifier belonging to a set that is called a signature. Ctrlc: Vc – Sc is a control map that assigns a control to each cloud customer where the signature Sc is defined by $Sc = [EU : (1, atomic), ISV : (1, atomic)]$, and Ctrls: Vs – Ss is the control map that assigns a control to each cloud service where the signature Ss is defined by $Ss = [IaaS : (3, active), PaaS : (3, active), SaaS : (3, atomic)]$. GP represents the place graph and GL represents the link graph.

2.2 Dynamic Aspects

Additionallly to cloud architecture structural aspects formalization, BRS is expressive enough to be adopted for representing cloud dynamics. In this section, we define two categories of reaction rules: Service Allocation Reaction Rules (SARR) and SCalability Reaction Rules (SCRR).

Service Allocation Reaction Rules. We propose two reaction rules, defining the dynamics of bigraphs in terms of service allocation. This category affects bigraph linkage only, where the redex a left hand pattern (R) of a reaction rule, can match any cloud service (IaaS, PaaS or SaaS) and customer (EU or ISV) controls. Figure 2 shows a cloud customer (C1) allocating a cloud service (S1) representing the first reaction rule. Also, it shows a cloud customer (C2) liberating a cloud service (S2) representing the second reaction rule.

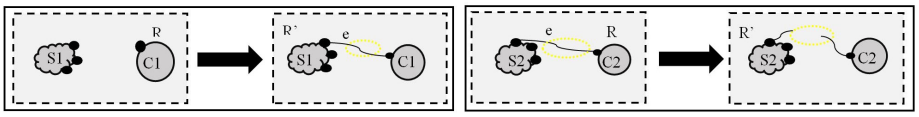

Fig. 2. Service Allocation/deallocation Reaction Rules.

SCalability Reaction Rules. The need of cloud services scalability is justified by system surcharge with customer requests rendering system services unavailable. One solution of cloud services unavailability is a scaling-up of the cloud system, by creating new instances of the concerned cloud service (loaded in memory). This instance will be responsible of taking in charge new customer demands (by doubling cloud service initial bandwidth). Also, a garbage collector can be automatically triggered when cloud customers liberate a cloud service. It models a scaling-down of a cloud system. Thus, we define two reaction rules; representing cloud system scaling up and down respectively (see Figures 3).

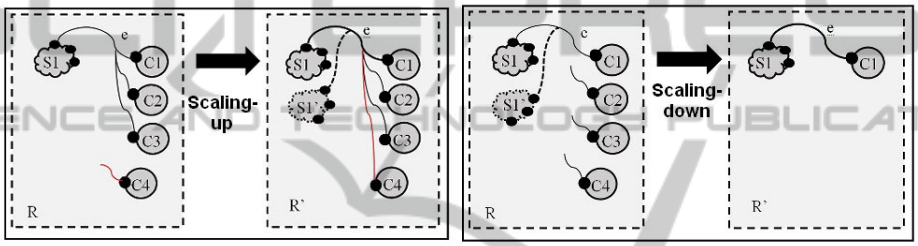

Fig. 3. Scaling-up/down Reaction Rules.

Table 1 bellow summarizes the correspondence between Cloud Computing elements and BRS concepts:

Table 1. Mapping cloud computing concepts to BRS.

Cloud Computing	**Bigraphical Reactive Systems (BRS)**
Service	Node.
Customer	Node.
Service models	Controls.
Customer types	Controls.
Cloud architecture	Bigraph.
Cloud dynamics	Reaction rules.

3 Verifying Cloud Systems

A set of tools is proposed for BRS edition as BigMC[12], BPLTool [13], DBtk [14], etc. However, during their exploration, we have been confronted with several limitations to verify a specific bigraph with new types of nodes (controls) adapted to the context of Cloud Computing. Therefore, we propose the Bigraphical Maude-Based Model Checker (BMMC) in which a cloud architecture can be easily executed and verified. It confronts system model to system invariants to produce two possible results. A positive

result is returned if the bigraphical model execution satisfies the desired properties or a negative result otherwise.

The use of rewriting logic via its Maude language[9], offers an executable and analysable specification that takes advantage of tools around Maude environment, as the model-checker for linear temporal properties verification[9].

3.1 BMMC Modules Description

A set of modules has been defined to map cloud bigraphical concepts to Maude. The following descriptions of modules show how the previous specification is translated to Maude.

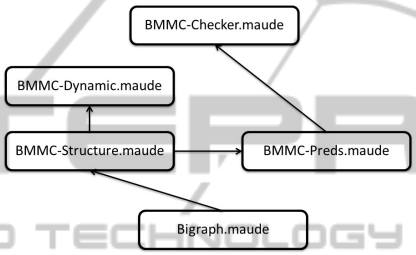

Fig. 4. Overview of BMMC modules.

Figure 4 gives a basic overview of the BMMC modules and their sub-module dependencies.

– The Bigraph module includes sort and operator declarations for Bigraph theory:

```
mod Bigraph is
...
sorts Node Bigraph Edge Port Control .
...
op _|_ :-> Bigraph Bigraph [ctor comm assoc id: null] .
op edge_: Nat-> Edge [ctor] .
...
```

Prime product operator (|) is a juxtaposition operator of nodes or bigraphs, since it takes as arguments two bigraphs with nodes being a sub-sort of bigraph.

– The syntax and semantics of cloud elements are defined in the BMMC Structure module:

```
mod BMMC-Structure is
...
op IaaS :-> Ss [ctor] .
op PaaS :-> Ss [ctor] .
op SaaS :-> Ss [ctor] .
```

```
op EU :-> Sc [ctor] .
op ISV :-> Sc [ctor] .
...
op service<_;_;_>[_].{ _ } : Qid Ss S_State Port
                                        Bigraph -> Vs [ctor] .
op customer<_;_;_;_>[_] : Qid Sc Qid Port
                                          Edge -> Vc [ctor] .
endm
```

Service operator serves to specify a cloud service, it takes as arguments:

- a Quoted Identifier to specify its name;
- a Control specifying its service delivery model;
- an Attribute specifying a service state (available, unavailable, and cloned).
- a set of Edges to be connected to its predefined ports;
- a set of child nodes being contained within the considered service.

Customer operator allows specifying a cloud customer, it takes as arguments:

- a Quoted Identifier to specify its name;
- a Control specifying its customer type;
- a Quoted Identifier to specify the requested service;
- a Port on which the requested service will be delivered;
- an Edge connected to its port.

– Cloud systems dynamics has been formalized with BMMC Dynamic module, containing a set of rewrite rules corresponding to BRS reaction rules. We assume that the term [Req] in customer operator characterizes a service requester, while the term [Lib] represents a customer who liberates a requested service.

```
mod BMMC-Dynamic is
...
rl [Service-Allocation] :
  service< i ; cs:Ss ; available >[ e1 , e2 , e3 ].{ b2 } |
  customer< i1 ; cc1:Sc ; i ; p:Port >[ Req ] | b1
 =>
  service< i ; cs:Ss ; available >[ e1 , e2 , e3 ].{ b2 } |
  customer< i1 ; cc1:Sc ; i ; p:Port >[ e1 ] | b1 .
rl [Service-Liberation] :
customer< i1 ; cc:Sc ; i ; p:Port >[ edge na ] |  b1
=>
customer< i1 ; cc:Sc ; i ; p:Port >[ Lib ] |  b1 .
...
```

Reaction rules (defined in Section 2.2) will be transformed into rewrite rules. Thus, several rewrite rules are defined in terms of service allocation and system scalability.

– The BMMC Preds module represents a system specification property expressed in linear temporal logic. The relevant state predicates are typically part of the property specification. So, we propose two state predicates to identify whether a service is cloned or a customer is a service requester. These predicates can be parametrized with service and customer operators; their semantics is defined as follows:

```
mod BMMC-PREDS is
...
    subsort Bigraph < State .
    op requester : -> Prop .
    op cloned : -> Prop .
...
eq customer< i1:Qid ; c:Sc ; i2:Qid ; p:Port >[ Req ] | b
|=
requester = true .
eq service< i1:Qid ; c:Ss ; cloned >[e1, e2, e3].{ b1 } | b
|=
cloned = true .
...
```


– Model checking LTL formula is realized using a given initial state. For this reason, BMMC-Checker module, defines an initial state containing a service (s2) of control IaaS that is unavailable and a customer (c4) of control ISV requesting service (s2) via its public port. In this case and during execution, the system will automatically (by running the scaling-up rule) add an instance (”cloned”) of service (s2) to treat customer (c4) request.

```
mod BMMC-CHECKER is
        ...
    op initial1 : -> Bigraph .
    eq initial1 =
service<'s2; IaaS; unavailable>[edge 1, edge 2, edge 3].{null}
| customer< 'c4 ; ISV ; 's2 ; PbP >[ Req ] .
endm
```

3.2 Services Availability and Quick Scalability Properties

The BMMC, implemented in Maude as a system module, provides an executable specification of a cloud system. It can be exploited to simulate the cloud system so specified. But we can do more. Under appropriate conditions we can check that our bigraphical model satisfies some important properties. We deal here with simpler, yet very useful, properties such as service availability and quick scalability, while considering initial state specified in ”BMMC-Checker” module in this example case. Figure 5 bellow shows model checking results of service availability and quick scalability properties that are both verified.

The specification of the considered properties:

– *”Services Availability is fulfilled if system model (in its canonical final states) do not contain an unsatisfied customer request”*.

– *"Quick scalability is verified if all states, in which no more rewrites are possible, contain at least one instance of a cloned cloud service"*.

These specifications are expressed by the following LTL formulas:

– *O [] not (requester)* .
– *O [] (cloned)*.

```
Maude>

Maude> reduce in BMMC_CHECK : modelCheck(initial1, O [] ~requester) .
rewrites: 11 in 14099565028ms cpu (1ms real) (0 rewrites/second)
result Bool: true

Maude> reduce in BMMC_CHECK : modelCheck(initial1, O [] cloned) .
rewrites: 10 in 14099565028ms cpu (1ms real) (0 rewrites/second)
result Bool: true
```

Fig. 5. Verification of Services Availability and Quick Scalability properties.

4 Conclusion

Bigraphical Reactive Systems (BRS) have been adopted as a semantic framework for cloud computing specification. BRS seem adequate for two reasons. Firstly, the model emphasizes on both locality and connectivity that can be used to specify location and interconnection of cloud systems. Secondly, a set of reaction rules, providing to bigraphs the ability to reconfigure themselves, are very useful to formalize cloud system dynamics, especially cloud services elasticity. The defined bigraphical model allows developers to correctly reason about most cloud computing features as modelling, composition and reconfiguration. Particularly, we have shown how this model provides a flexible theoretical framework where cloud service delivery and deployment models can be naturally defined. A nice consequence of this axiomatization is that relationships between cloud services and cloud customers have been exploited to formally verify some properties. This would not have been possible without the proposal of the BMMC, which is the mapping of bigraphical model to a Maude-based formal executable specification. Compared to existing works, the proposed approach achieves a balance between understandability of representation and formal reasoning techniques.

As future work, we plan to exploit meta-modelling and model transformation tools to realize a graphical interface and a BMMC specifications generator in order to facilitate the exploitation of our proposed approach, thereby permitting graphical edition and automatic generation of the corresponding specifications.

References

1. Michael, A., Armando, F., et al.: Above the clouds: A berkeley view of cloud computing, EECS Department, University of California, Berkeley (2009)
2. Dong, H., Hao, Q., Zhang, T., Zhang, B.: Formal discussion on relationship between virtualization and cloud computing. In: PDCAT. (2010) 448–453
3. Grandison, T., Maximilien, E., Thorpe, S., Alba, A.: Towards a formal definition of a computing cloud. In: SERVICES, 6th World Congress on. (2010) 191–192

4. Luo, S. X., Liu, F. M., Ren, C. L.: A hierarchy attribute-based access control model for cloud storage. In: ICMLC. Volume 3. (2011) 1146–1150
5. Adamov, A., Hahanov, V.: A security model of individual cyberspace. In: EWDTS. (2011) 169–172
6. Freitas, L., Watson, P.: Formalising workflows partitioning over federated clouds. In: SERVICES, IEEE Eighth World Congress on. (2012) 219–226
7. Binz, T., Fehling, C., Leymann, F., Nowak, A., Schumm, D.: Formalizing the cloud through enterprise topology graphs. In: Cloud Computing (CLOUD). (2012) 742–749
8. Milner, R.: The Space and Motion of Communicating Agents. Cambridge University Press (2009)
9. Clavel, M., Durn, F., Eker, S., et al.: All about maude - a high-performance logical framework. Lecture Notes in Computer Science, Springer (2007)
10. Benzadri, Z., Belala, F., Bouanaka, C.: Towards a formal model for cloud computing. In: ICSOC 2013 Workshops. (2013)
11. Mell, P., Grance, T.: The nist definition of cloud computing, National Institute of Standards and Technology (NIST) (2011)
12. Perrone, G., Debois, S., Hildebrandt, T. T.: A model checker for bigraphs. In Ossowski, S., Lecca, P., eds.: SAC, ACM (2012) 1320–1325
13. Glenstrup, A. J., Damgaard, T. C., Birkedal, L., Hjsgaard, E.: An implementation of bigraph matching (2008)
14. Bacci, G., Grohmann, D., Miculan, M.: Dbtk: A toolkit for directed bigraphs. In: CALCO. Volume 5728., Springer (2009) 413–422