Towards Formal-based Semantic Interoperability in Multi-Clouds

Stéphanie Challita, Fawaz Paraiso, Philippe Merle

Towards Formal-based Semantic Interoperability in Multi-Clouds

The FCLOUDS Framework

Stéphanie Challita, Fawaz Paraiso and Philippe Merle
Inria Lille - Nord Europe / University of Lille / CRIStAL UMR CNRS 9189, France
Email: firstname.lastname@inria.fr

***Abstract*—Multi-cloud computing has been proposed as a way to reduce vendor lock-in, to improve resiliency during outages and geo-presence, to boost performance and to lower costs. However, semantic differences between cloud providers, as well as their heterogeneous management interfaces, make changing from one provider to another very complex and costly. This is quite challenging for the implementation of multi-cloud systems. In this paper, we aim to take advantage of formal methods to define a precise semantics for multi-clouds. We propose FCLOUDS, a formal-based framework for semantic interoperability in multi-clouds. This framework contains a catalogue of formal models that mathematically describe cloud APIs and reason over them. A precise alignment can be described between their concepts, which promotes semantic interoperability.**

***Index Terms*—Multi-Clouds; Interoperability; Formal Language; Formal Verification**

I. Introduction

Multi-cloud computing describes a paradigm where the application provisions multiple cloud heterogeneous services simultaneously. This strategy has been adopted in the cloud computing industry since a while in order to improve disaster recovery and geo-presence, to use unique cloud services from different providers as they are needed, and to ensure unlimited scalability of cloud applications [1]. Several cloud outages have taken place in the past [2], which prove that the sentence "don't place all your eggs in one basket" is equally applicable to the cloud ecosystem. Today, there is a variety of cloud providers like Amazon, Google, Microsoft, etc. who offer functionalities as services at different levels of abstraction such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Even at the same abstraction level, providers employ varying terminology and features, which usually do not map directly those of competing providers [3]. These *semantic differences* are critical in cloud computing as they make migrating an application across providers a very complicated and costly task. In addition, the management of a potentially large number of cloud services with *heterogeneous interfaces* is a challenge, because of incompatibility between the different interfaces. Worse still, the semantics of these interfaces is informally described in their documentation available at provider websites. It is then impossible to understand the behaviour of a cloud when the developer requests a virtual machine for example. For the above concerns, cloud interoperability is prevented and vendor lock-in is promoted, which hinders the implementation of a multi-cloud system.

A considerable number of programming libraries and model-driven solutions were developed for interoperability in multi-cloud systems [4]. However, as detailed in Section II, their abstraction level remains insufficient to overcome semantic differences, and their small formalisation level induces ambiguities and worsens the problem of heterogeneity. For complex systems, such as multi-clouds, a more precise and formal approach is required.

To tackle this issue, we propose FCLOUDS, a formal-based framework for semantic interoperability in multi-clouds. FCLOUDS relies on a catalogue of cloud formal models, i.e., models that mathematically describe cloud concepts and operations and reason over them by proving several properties. Interoperability rules can be formally defined in order to compare different cloud offers and avoid any kind of confusion between them. Our semantic framework represents the main pillar to build a consistent bridge, with a unified API implementing mathematical transformations, to promote real-world interoperability between cloud providers.

The remainder of this paper is structured as follows. In Section II we outline multi-cloud interoperability and we explain the motivation behind our contribution. In Section III we present our framework FCLOUDS, its components and a usage scenario. Finally, we conclude in Section IV.

II. Multi-cloud Interoperability

The cloud market counts numerous cloud solutions at different levels of abstraction, traditionally called service layers. From our point of view, these solutions can be classified into four spaces, as shown in Fig. 1. We identify *Provider Space*, *Programming Space*, *Modelling Space* and *Semantic Space*. In the following, we present these spaces and highlight the heterogeneity problem at each space of the cloud stack.

A. Provider Space

Amazon Web Services (AWS), Google Cloud Platform (GCP), OpenStack and European Grid Infrastructure Federated Cloud (EGI FC) are public, private and hybrid cloud providers respectively. They have emerged during the last ten years and offer various infrastructure, platform or software services. Even when they offer the same service, the latter may have

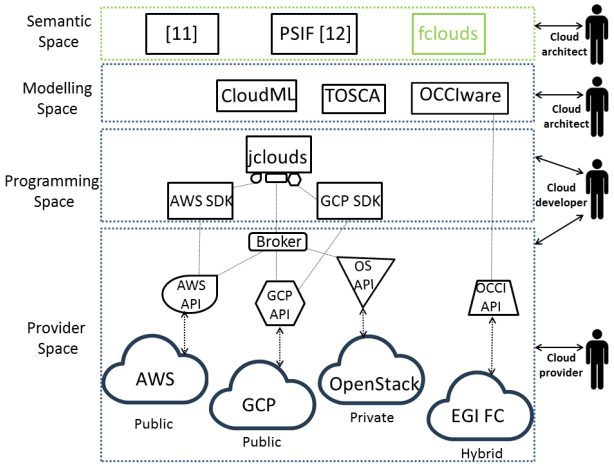

Fig. 1: Multi-Cloud Spaces.

different names [5], characteristics and functionalities. In addition, cloud providers rely on APIs, i.e., management interfaces that provide programmatic remote access to their resources. Each API is based on an architecture and/or a technology that does not necessarily facilitate their interoperability. For example, AWS is accessible via a SOAP API, whereas GCP is based on a REST API, which leads to an incompatibility between two different APIs. As for semantics, cloud APIs are defined only informally, i.e., described in natural language in website pages. The developer would not understand the exact behaviour of the provider when he/she asks for a VM for example. This usually leads to vendor lock-in. To address this problem, the solution in this space would be a broker such as Rightscale, Kaavo, etc. The broker offers a unique interface to handle the heterogeneity of different APIs, i.e., to only mask the problem and not semantically resolve it.

B. Programming Space

In order to allow developers to provision cloud services, each cloud offers one or several language-specifc Software Development Kits (SDKs) to hide technical details of APIs. However, these SDKs are heterogeneous. Therefore, many libraries like jclouds, libcloud and Fog have emerged to allow developers to add abstraction between the cloud SDKs and enable multi-clouds. The multi-cloud libraries are tightly coupled to their programming languages like Java, Python and Ruby, so the language compiler is able to check the correctness of the developer code but does not know how to perform a verification related to the cloud computing field. The developer needs to ignore implementation details and focus on general properties and characteristics. This will help him to avoid premature commitment to implementation choices.

C. Modelling Space

Meanwhile, there is a need for cloud architects to design their applications for multi-clouds. For this, model-based solutions are becoming increasingly popular in cloud computing as they provide domain-specific modeling languages and frameworks that enable architects to describe/select/adapt multi-cloud environments. This strategy is summarised as *"Model once, generate anywhere"*. We identify some of the notable model-based solutions for multi-clouds, namely CloudML [4], TOSCA [6] and OCCIware [7]. Unlike programming libraries, they work at a high level of abstraction by focusing on cloud concerns rather than implementation details. We believe that model-driven engineering brings many benefits for multi-clouds [8]. However, we notice that existing model-based multi-cloud solutions lack for a precise semantics in their specifications. For example, the specifications of CloudML and TOSCA are informal documents written in English prose and their behavioural semantics is hidden in their model interpreter. As for the OCCIware model, despite using OCL constraints for defining *static semantics*, OCL syntax is not formal by itself [9]. In addition, OCCIware model employs only ambiguous natural language to express *dynamic semantics*, i.e., the system behaviour. The lack of formalisation hinders the understanding of the models, thus complicates the interoperability in multi-clouds.

D. Semantic Space

A set of common principles that all interoperability solutions adhere to must be agreed on. Accordingly, we argue in the following for the need to explore the *Semantic space* through FCLOUDS framework:

- to provide a programming-language-independent, unambiguous mathematical specification of the multi-cloud solutions. It encourages focusing on what a system should do rather than how to accomplish it, and has only one interpretation, unlike natural language statements;
- to allow a formal validation of cloud structural and behavioural models. This is quite advantageous because the earlier a defect is removed the cheaper it will be to correct it;
- to reason about the cloud formal models and prove cloud properties, which are constraints denoting characteristics of cloud configurations and/or operations. They guarantee the accuracy and correctness of the multi-cloud solutions.

For the above reasons, we propose in our current work to head towards using formal models and verification for cloud computing. Previously, one existing work [10] proposed a formal model for cloud computing using Bigraphical Reactive Systems (BRS). Besides using different techniques to reason over the cloud, [10] differs in its objective too. It aims to reason over cloud concepts for deployment and adaptation purpose. It does not deal with behavioural semantics and does not focus on the interoperability concern. For the field of semantic interoperability, the authors in [11] proposed an ontology-based framework for semantic interoperability in multi-clouds. They define translations between IaaS concepts of three standards. However, they do not consider any alignment between API operations. PaaS Semantic Interoperability Framework (PSIF) [12] was implemented in the context of Cloud4SOA project. PSIF proposes common PaaS models

that describes structural, functional and behavioural semantics. FCLOUDS goes beyond PSIF and aims to verify properties of the models thanks to the usage of formal methods.

III. The FCLOUDS Framework

In this section, we present **FCLOUDS**, which is a conceptual formal-based framework for semantic interoperability in multi-clouds. We begin by giving a scenario that motivates our approach and then we describe how we model FCLOUDS structure and behaviour. We also explain the reasoning process and identify three properties to be held in FCLOUDS models.

A. Usage Scenario

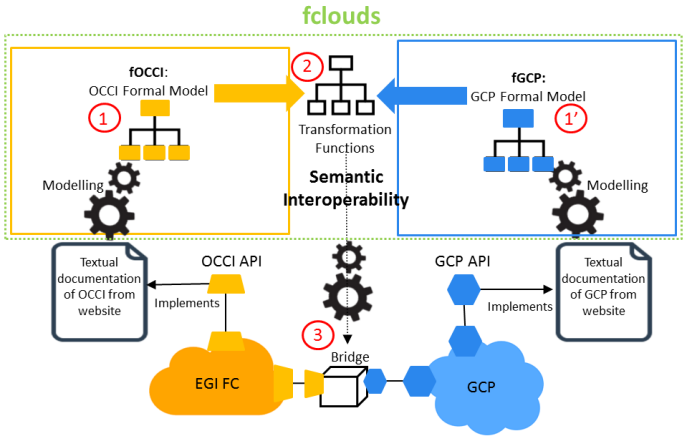

Fig. 2: FCLOUDS Usage Scenario.

We assume that a developer would like to build a multi-cloud system spread over two clouds, the private EGI FC and the public GCP. EGI is based on Open Cloud Computing Interface (OCCI) REST API and GCP on its own REST API, so the developer is faced to two heterogeneous APIs implementing different concepts and paradigms. To provision a virtual machine, the HTTP request has a different format for each API, which is quite frustrating. The developer would like a single API for both clouds to seamlessly access their resources. However, the GCP API will not work on OCCI and if OCCI wanted to provide equivalent services to GCP, it should not only adopt the same type of API but also the same concepts with the risk of misunderstandings, inconsistencies and incompleteness. Conflicts and misunderstandings about the semantics of cloud providers can be solved, or at least identified at an earlier stage, if aspects of structure and operations are conveyed through the use of formal models.

As depicted in Fig. 2, *the first stage* of FCLOUDS requires extracting from websites, in a manual or automated way, knowledge regarding the services offered by cloud providers. Then, we proceed by a precise modelling to understand and validate the behaviour of cloud APIs, in order to overcome their semantic heterogeneity. *In the second stage* of FCLOUDS, we proceed with transformation models, which explain how knowledge collected against one cloud provider can be transformed to fit another. This stage allows a rigorous comparison and semantic connections between cloud providers. Finally, *in the third stage*, our formal framework is incorporated in the development and maintenance of a bridge with a unified API, to promote real-world interoperability, while formal semantics is properly reflected in its behaviour.

B. Overview of FCLOUDS

FCLOUDS framework is based on several cloud formal models that can be composable. It consists of a formal model for OCCI (FOCCI), a formal model for GCP (FGCP), a formal model for AWS (FAWS), etc. (see Fig. 3). Later on, we will define *Transformations* that find equivalence and specialization relationships between them, thus we can seamlessly achieve semantic interoperability.

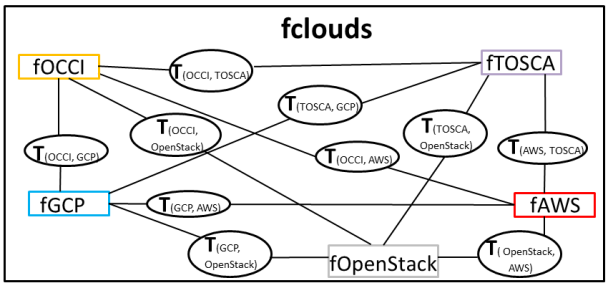

Fig. 3: FCLOUDS Framework Overview.

In the following, we detail the process of formalising cloud APIs, which is at the basis of FCLOUDS framework. As shown in Fig. 4, it is developed in two main steps: **Modelling** and **Reasoning**.

C. Modelling

Modelling using formal methods is the process of providing a precise specification of a cloud model, i.e., defining and validating:

- **Cloud structure and constraints**, as represented in Frame (1) in Fig. 4, to denote the types of cloud concepts such as virtual machines, containers, storage, operating systems, servers, applications, etc. and describe configurations of these types. FCLOUDS models support cloud computing concepts specification while simplifying irrelevant details to focus on the most important characteristics.
- **Cloud API operations**, as represented in Frame (2) in Fig. 4, to denote the operations that the developer uses to provision, manage or release cloud services through the cloud API.

In our work, we aim to choose the formal language Alloy that provides a concise specification of FCLOUDS models, with both a graphical output and a textual output, so it can be easy to analyse and reason over them.

D. Reasoning

Using formal models has the advantage of allowing reasoning over concepts and operations for a better understanding of

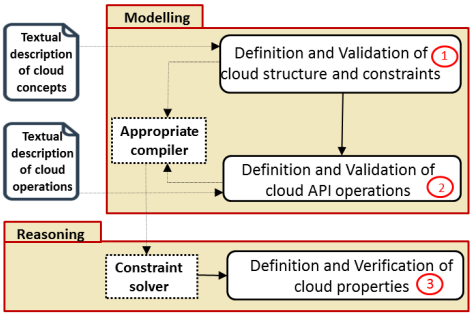

Fig. 4: Formalisation Process.

their semantics and how they work. Therefore, once FCLOUDS models have been specified, we proceed by the **definition and verification of cloud structural and behavioural properties** as shown in Frame (3) in Fig. 4. This step allows to ensure the correctness of all cloud formal models in the FCLOUDS framework, so we can draw later inferences between them.

At first, we verify using a constraint solver whether some general properties hold in our formal model. Among these properties, we are mainly interested in checking:

a) Consistency: to verify that there are no contradictory constraints, so the concepts are instantiable and each cloud API operation is executable. We can also analyze what could not be instantiated, thus can't be deployed in real-world. In these cases, our formal model might be overconstraining so we deem necessary to release some constraints.

The notion of consistency is basic and does not suffice in order to validate our model. There are other examples of reliable verification and validation tasks that can be performed on pairs of operations such as sequentiality and/or reversibility of operations:

b) Sequentiality: to check that sometimes a cloud API operation cannot happen if another operation did not happen at the time before. For example, the developer can adapt the performance of a virtual machine only if it was created before.

c) Reversibility: to check that some cloud API operations are reversible because they contain contradictory mathematical logic. For example, de-provisioning a virtual machine reverses the operation of provisioning it.

IV. Conclusion

The wide number of available cloud providers, their high heterogeneity and semantic differences make it complicated to exploit multi-cloud assets. In this paper, we provide a classification of multi-cloud solutions, and explain the need for formal languages and techniques. We propose FCLOUDS, a formal-based framework for semantic interoperability in multi-clouds. It carries cloud mathematical models, that allow formal description of cloud APIs core concepts and operations, for a better understanding of their behaviour. To our knowledge, FCLOUDS is the first framework that promotes mathematical reasoning and verification over cloud models. Having rigorously specified the structure and behaviour semantics of each cloud, we can define transformation functions, thus ensure semantic interoperability between them.

For future work, we will continuously complete the FCLOUDS framework with a wide catalogue of formal cloud models that describe mathematically concepts, capabilities and constraints of cloud providers and verify cloud specific properties. We will define and verify other properties such as *Reachability*, i.e., when executing operations on cloud resources through APIs, there is always a transition from a resource state to another. Finally, we will define semantic links between FCLOUDS models, which will give rise to transformation functions also verifying these properties. This goes into the direction of our long-term goal, which is allowing FCLOUDS framework to be executable inside the first formal-based real-world interoperability bridge.

Acknowledgment

This work is supported by both the OCCIware research and development project funded by French Programme d'Investissements d'Avenir (PIA) and Hauts-de-France Regional Council.

References

[1] D. Petcu, "Multi-Cloud: Expectations and Current Approaches," in *Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds*. ACM, 2013, pp. 1–6.

[2] S. R. Ko, S. Lee, and V. Rajan, "Cloud Computing Vulnerability Incidents: A Statistical Overview," *Cloud Security Alliance*, 2013.

[3] G. Sousa, W. Rudametkin, and L. Duchien, "Automated Setup of Multi-Cloud Environments for Microservices-Based Applications," in *9th IEEE International Conference on Cloud Computing*. IEEE, 2016, pp. 327–334.

[4] N. Ferry, A. Rossini, F. Chauvel, B. Morin, and A. Solberg, "Towards Model-Driven Provisioning, Deployment, Monitoring, and Adaptation of Multi-Cloud Systems," in *2013 IEEE Sixth International Conference on Cloud Computing (CLOUD)*. IEEE, 2013, pp. 887–894.

[5] F. Petrillo, P. Merle, N. Moha, and Y.-G. Guéhéneuc, "Towards a REST Cloud Computing Lexicon," in *7th International Conference on Cloud Computing and Services Science, CLOSER 2017*. IEEE, 2017, pp. 348–355.

[6] T. Binz, G. Breiter, F. Leyman, and T. Spatzier, "Portable Cloud Services Using TOSCA," *IEEE Internet Computing*, no. 3, pp. 80–85, 2012.

[7] J. Parpaillon, P. Merle, O. Barais, M. Dutoo, and F. Paraiso, "OCCIware - A Formal and Tooled Framework for Managing Everything as a Service," in *Projects Showcase@ STAF'15*, vol. 1400, 2015, pp. 18–25.

[8] H. Bruneliere, J. Cabot, and F. Jouault, "Combining Model-Driven Engineering and Cloud Computing," in *Modeling, Design, and Analysis for the Service Cloud-MDA4ServiceCloud'10: Workshop's 4th edition (co-located with the 6th European Conference on Modelling Foundations and Applications-ECMFA 2010)*, 2010.

[9] M. Richters and M. Gogolla, "On Formalizing the UML Object Constraint Language OCL," in *International Conference on Conceptual Modeling*. Springer, 1998, pp. 449–464.

[10] Z. Benzadri, F. Belala, and C. Bouanaka, "Towards a Formal Model for Cloud Computing," in *International Conference on Service-Oriented Computing*. Springer, 2013, pp. 381–393.

[11] K. Yongsiriwit, M. Sellami, and W. Gaaloul, "A Semantic Framework Supporting Cloud Resource Descriptions Interoperability," in *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*. IEEE, 2016, pp. 585–592.

[12] N. Loutas, E. Kamateri, and K. Tarabanis, "A Semantic Interoperability Framework for Cloud Platform as a Service," in *2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2011, pp. 280–287.