Research internships (Bachelor, Master 1, Master 2)
Towards formally verified configuration languages
The ANR project For-CoaLa

Hélène Coullon
IMT Atlantique, LS2N, Inria, France

Frédéric Loulergue
Université d'Orléans, LIFO, France

2024-2025

Keywords *:* Infrastructure-as-Code, configuration, DevOps, dynamic reconfiguration, verification, formal methods, interactive theorem provers, Coq.

1 Context

Large distributed software systems (applications or infrastructures) are now ubiquitous, with component-based systems (e.g., service-oriented architectures or microservices) offering a convenient way to structure large systems, particularly distributed systems deployed in the cloud, in the core or at the edge of the network. Isolating functionalities in components and building systems through composition greatly enhances the adaptability and scalability of systems, two important requirements for many organizations.

However, the advantages of distributed architectures come at the price of increased complexity and technical challenges related to observability, coordination, maintenance, etc. A set of operations denoted as DevOps, i.e., operations handled somewhere between the developer and the machine administrator, includes notably the system configurations and reconfigurations required to achieve adaptability. These operations include the initial and dynamic changes that can occur in a distributed service-oriented software architecture : adding or removing services, connecting or disconnecting services, and changing some parameters or the internal behavior of services [3]. They may be required to handle various dynamic scenarios such as fault tolerance, scalability, software updates, various optimizations, etc.

Such changes may lead to faults. For example, recent outages have been traced back to problems within maintenance codes [1] [2]. A study of 597 unplanned outages that affected popular cloud services between 2009 and 2015 found that 16% of them were caused by a software or hardware upgrade [**?**]. The study concludes that "the complexity of the cloud hardware and software ecosystem has outpaced existing testing, debugging, and verification tools". In fact, testing and debugging methods are largely inadequate in the context of distributed systems, while the adoption of more suitable formal methods remains marginal in industry. The latter can be attributed to the difficulty in using formal methods and tools. However, formal methods can lighten the burden of program developers, DevOps engineers, and system administrators instead of adding to it.

On the one hand, many configuration tools and languages exist in the DevOps community, some of them being specific to the provisioning of resources in Cloud providers, packaging problems, containerized deployments, configuration management of applications or infrastructures, etc. Such languages are often referred to as *Infrastructure-as-Code* (IaC for short). IaC languages offer easier constructs than

1. `https://blog.cloudflare.com/cloudflare-outage-on-july-17-2020/`
2. `https://engineering.fb.com/2021/10/05/networking-traffic/outage-details/`

low-level scripts to facilitate software engineering properties such as composition and reuse. The main advantage of these tools is their full integration and adoption in the DevOps community. Their disadvantage is that they lack both formal and textual specifications. Moreover, their contours are blurred. On the other hand, many initiatives have been studied in academia to contribute to the deployment, configuration, and reconfiguration of distributed software [3], bringing improvements such as expressivity, speed, safety, etc. Many come with precise and sometimes formal definitions. However, they lack the breadth of the mainstream DevOps tools.

2 Work

The goal of FOR-COALA is twofold : (1) understand and bridge the gap between a popular tool from the DevOps community and a tool from academia ; (2) improve the understanding of these languages based on mechanized formal semantics and develop verified semantic-preserving cross-language transformations. The outcomes of the project will be two open source formally certified configuration languages, their execution engine, and a verified translation tool.

— From academia we choose CONCERTO

— From the industry tool perspective, we chose ANSIBLE [3] [5, 4].

CONCERTO is currently the tool that offers the highest level of parallelism when running configuration programs, therefore offering lower configuration and reconfiguration durations [1]. For this reason, it is currently attracting attention from industrial partners, in particular OVH and Orange. The execution semantics of CONCERTO has been manually formalized [2]. Third, a tool on top of CONCERTO adds declarative features to CONCERTO by automatically inferring (using the SMT solver Z3) the configuration program from a set of higher-order DevOps goals or objectives [6]. In other words, CONCERTO offers recent and active contributions with a strong basis for mechanized formalization. From an academic tool perspective, the research challenge of FOR-COALA is to cope with additional complex language characteristics compared to industrial tools, particularly the high level of parallelism (non-determinism) offered by CONCERTO.

ANSIBLE is currently one of the most widely used tools, probably due to its very wide spectrum (from low-level scripting to high-level modular and compositional features) and low installation requirements (it is agent-free unlike Puppet, another popular tool). Second, an extensive set of ANSIBLE playbooks (i.e. ANSIBLE programs) and roles (i.e., a set of programs to configure a given application or system) are publicly available online and can be leveraged in the project. Third, CONCERTO and ANSIBLE have already been compared and share common aspects that will facilitate a cross-language transformation.

There are several topics to which the interns could contribute :

1. statistics and data mining in ANSIBLE repositories to identify the most central language constructs and commands to certify,
2. case studies of Concerto on which the FOR-COALA tools will be evaluated,
3. the formalization and verification of some pieces of Ansible and Concerto in COQ.

As an intern of the project, the selected candidates will :

— contribute to one or several of the topics above (note that most of the need is on topic 3),

— contribute to the writing of research reports and papers, and to popularization articles in a blog.

3. `https://www.ansible.com`

3 Expected skills

The following skills are expected from the successful candidates :

- a student either in Bachelor's or in the first or last year of a Master's degree in Computer Science ;
- knowledge and interest in formal methods, in particular, interactive theorem provers (Coq or others) ;
- knowledge in distributed software systems ;
- knowledge in DevOps approaches such as Infrastructure-as-Code, containers, orchestration etc. ;
- good knowledge of the Python programming language ;
- a good level of English to contribute to the writing of reports and papers ;
- an ability to collaborate and communicate ;
- curiosity and an appetite for learning new things.

4 Additional information

Advisors

— Hélène Coullon, IMT Atlantique & Inria team STACK & LS2N, helene.coullon@imt-atlantique.fr

— Frédéric Loulergue, Université d'Orléans& LIFO, frederic.loulergue@univ-orleans.fr

Duration from 3 to 8 months

Salary legal amount of 3,90€ / hour, full time

Locations

— IMT Atlantique, équipe Inria Stack, laboratoire LS2N à Nantes

— équipe LMV, laboratoire LIFO à Orléans

Application (email to the advisors)

— Curriculum Vitæ

— Topics of For-CoaLa you are interested in

— Recent and past grades

— Preferred location (Nantes or Orléans)

Références

[1] Maverick Chardet, Hélène Coullon, and Christian Pérez. Predictable Efficiency for Reconfiguration of Service-Oriented Systems with Concerto. In *CCGrid*. IEEE, 2020. doi:10.1109/CCGrid49817.2020.00-59.

[2] Maverick Chardet, Hélène Coullon, and Simon Robillard. Toward safe and efficient reconfiguration with concerto. *Science of Computer Programming*, 2021. doi:10.1016/j.scico.2020.102582. hal: hal-03103714.

[3] Hélène Coullon, Ludovic Henrio, Frédéric Loulergue, and Simon Robillard. Component-based distributed software reconfiguration : A verification-oriented survey. 2023. doi:10.1145/3595376.

[4] Ruben Opdebeeck, Ahmed Zerouali, and Coen De Roover. Andromeda : A dataset of ansible galaxy roles and their evolution. In *IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, 2021. doi:10.1109/MSR52588.2021.00078.

[5] Ruben Opdebeeck, Ahmed Zerouali, and Coen De Roover. Smelly variables in ansible infrastructure code : Detection, prevalence, and lifetime. In *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*, 2022. doi:10.1145/3524842.3527964.

[6] Simon Robillard and Hélène Coullon. SMT-Based Planning Synthesis for Distributed System Reconfigurations. In *Fundamental Approaches to Software Engineering (FASE)*, 2022. doi:10.1007/978-3-030-99429-7_15.