Proving Correctness of Reconfigurable Systems

Radu IOSIF (VERIMAG, Grenoble)
mailto:Radu.Iosif@univ-grenoble-alpes.fr

1 Context and Motivation

Distributed computing has become vital for the sustainability of modern life, both socially (social networking) and economically (banking, commerce and finance). For example, cloud applications are used to store and share information among worldwide populations, whereas blockchain technology is the basis of cryptocurrencies and becomes more and more used in finance. Because of the widespread use of mobile devices, our understanding of a computing system has also changed, from a single-threaded or multi-threaded program running on a single (possibly multi-core) computer, to a distributed system in which processes running on different network nodes (computers, mobile phones or other intelligent devices) communicate by sending each other messages over the network. Due to the huge complexity of behavior scenarios, building distributed systems that behave correctly is particularly challenging. Moreover, since such systems control many aspects of human life, it is important to guarantee their correctness by using computer-driven *verification* methods from the earliest stages of design. For instance, a security breach in a cross-chain cryptocurrency exchange system recently caused over $40 million to be lost to hackers[1].

Just as a natural system, a distributed computing system is in continuous evolution, by replacing obsolete hardware and software components with newer versions or even changing the topology of the network (adding new nodes, removing defective nodes or electing a leader and moving from a ring to a star configuration, in which the leader coordinates all the other components). Due to their highly modular nature, distributed systems are easily *reconfigurable*. However, this flexibility typically complicates their design, because the designer must predict and avoid possible faults due to incomplete or incorrect reconfiguration.

2 Challenges and Goal

Due to inherent theoretical limitations, such as the undecidability or intractability of logical decision problems, formal verification is not feasible, unless the reduction of the verification problem to a logical decision problem takes into account a small part of the system, while framing out the rest, not relevant for the property that needs to be checked. This principle, known as *local reasoning* [2] is supported by Separation Logic (SL) [5], a variant of the logic of Bunched Implications (BI) [4], tailored to reasoning about resource ownership in a sequential (interprocedural) or (shared-memory) multithreading setting.

A first challenge of this project is steering aways of the traditional view of concurrent processes that own and share passive resources. In our view, processes *are* resources themselves. This hypothesis is justified by the following aspects of distributed computing systems:

- *Lack of shared memory*: in a distributed system all communication is done via messages sent over the network or, in a more abstract view, via jointly executed atomic actions, triggering a local state change each. Consequently, the system can be viewed as a structure whose nodes are the components (processes) and whose relations represent interactions between components. Such systems can be reasoned about *compositionally* (inferring properties of disjoint substructures separately) and *hierarchically*, i.e. encapsulating substructures within meta-components.
- *Reconfigurability*: in reconfigurable distributed systems, the changes in the coordinating architecture are caused by special actions, that mutate components and interactions. Reasoning about such mutations can be done locally, by describing only the components that are directly involved in a reconfiguration, while framing out the rest of the system. Again, considering components as active resources plays an important role in having *local specifications* of reconfiguration actions that can be combined using *frame rules*.

The work in this project will be a continuation of the Behavior-Interaction-Priorities (BIP) framework [1] and its Dynamic Reconfigurable BIP (DR-BIP) extension [3].

[1] https://www.wired.com/story/hack-binance-cryptocurrency-exchange/

A second challenge is related to the size and the complexity of nowadays distributed systems, consisting of myriads of interconnected software components, that are instances (replicas) of a relatively small number of *component types*, usually built by different developpers. Since the number of instances of each component type is, in general, quite large, we consider it to be *arbitrarily* large. This assumes that a distributed systems is *parameterized* by the:

- *Architectural pattern* that coordinates the communication between components (e.g. a clique, pipeline, ring, star or tree, etc.), and
- *Number of components* of each type, (e.g. m readers and n writers, where m and n are not known).

The goal of this PhD project is the development of a rigourous design framework for distributed systems, that uses verification from the early stages of model building. Formal verification is the process of transforming a query about the correctness of a system (e.g. is every store request to a cloud treated by storing the data in such a way that any future retrieval request will eventually find it?) into a logical validity query (is a given formula true in general?) that can be handled by automated reasoning techniques (theorem proving). Hence, we recognize *logic*, *automated reasoning* and *model checking* as being the main ingredients of formal verification, and also central research topics in our project.

The successful candidate is expected to work on logics for high-level reasoning about parameterized architectures of distributed systems. The decision problems of these logics will be studied from a theoretical point of view, such as existence of proof systems and the complexity of related algorithms. The architecture description logics will be used to verify systems by parameterized model checking techniques based on e.g. invariant synthesis and automata learning. We plan on implementing proof-of-concept prototype tools and perform experiments on models of real-life systems (inter-blockchain communication, mobile applications, etc.)

3 Hosting Laboratory

VERIMAG is an academic laboratory focusing on theoretical and practical aspects of formal methods for embedded system development. Since its creation, in 1993, VERIMAG has a proven record in both basic theoretical research and in development of tools for system verification. In the recent years, VERIMAG became deeply involved in the area of embedded and cyberphysical systems. VERIMAG hosts 20 professors, 7 full-time researchers and over 15 PhD students.

4 How to Apply

Send your CV and a transcript of your university grades to:

mailto:Radu.Iosif@univ-grenoble-alpes.fr

References

[1] A. Basu, S. Bensalem, M. Bozga, J. Combaz, M. Jaber, T. Nguyen, and J. Sifakis. Rigorous component-based system design using the BIP framework. *IEEE Software*, 28(3):41–48, 2011.

[2] C. Calcagno, P. W. O'Hearn, and H. Yang. Local action and abstract separation logic. In *22nd Annual IEEE Symposium on Logic in Computer Science (LICS 2007)*, pages 366–378, July 2007.

[3] R. El Ballouli, S. Bensalem, M. Bozga, and J. Sifakis. Programming dynamic reconfigurable systems. In *FACS'18*, pages 118–136, 2018.

[4] P. W. O'Hearn and D. J. Pym. The logic of bunched implications. *BSL*, 5(2):215–244, 06 1999.

[5] J. Reynolds. Separation logic: A logic for shared mutable data structures. In *LICS'02*, pages 55–74, 2002.