Local Reasoning about Reconfigurable Component-based Systems

Radu IOSIF (VERIMAG, Grenoble)

http://nts.imag.fr/index.php/Radu_Iosif

1 Background

Nowadays computer systems are highly complex. Most of them are intrinsically distributed and dynamically reconfigurable (cloud-based services, Internet of Things), concurrent at both the physical level of CPU cores and the logical level of threads, engaging in complex interactions not only between their own hardware and software parts, but also with their networking environment. Designing and understanding such complex systems is only possible due to their modularity: a system is hierarchically organized as an architecture of components, whose internal details are encapsulated within simple well-defined interfaces. The modularity of a component-based computer system is instrumental in performing updates (replacing one or more components with newer versions having the same interfaces) and reconfigurations (changing the coordinating architecture by adding new components, removing obsolete versions or even changing the topology of interactions, e.g. from a token ring to a star).

Because such complex systems control many aspects of human life (airline traffic, power grids, social networks), it is important to ensure their correctness a priori, by design, and also a posteriori, by verification. Designing and verifying such systems rely on two interdependent ingredients: (i) locality, which is the ability to describe the effect of an update only from the parts involved while ignoring the ones unchanged, and (ii) compositionality, which is the ability to join the results of local analyses into a global condition capturing the correctness requirement for the entire system.

Logic has been recognized as the main ingredient of formal design and verification techniques. The last two decades have seen the advent of resource logics, such as Bunched Implications (BI) [2] and Separation Logic (SL) [3], that offer means to reason about resource composition. These logics now constitute the foundation of industrial-scale program verifiers such as Infer (Facebook) or SLAyer (Microsoft) that can prove correctness of programs with respect to memory errors (non-allocated pointer accesses, memory leaks).

2 Challenges and Goal

The components of a modular system can be viewed, at a higher level of abstraction, as resources, that can be either static or dynamic (a piece of data or a process), physical or logical (CPUs or threads), having atomic or more elaborated structure (a memory cell or a linked list). The composition operators can be aggregative, i.e. unions of disjoint sets, or non-aggregative (the decomposition of a resource is not solely characterized by a given partition of the resource domain). Separation Logic (SL) [3] is a well-studied logic for describing aggregative composition of dynamic memory data structures (lists, trees, etc.). However non-aggregative composition, such as parallel composition of processes, has received relatively little attention from logicians, which suggests that this latter type of composition is harder to axiomatise and reason about.

Recent work by members of VERIMAG focuses on the development of a resource logic used to model and reason about the architectures of distributed component-based systems [1]. Intuitively, by architecture we mean the shape (ring, star, clique, etc.) of the interaction network in a distributed system, where processes run on physical nodes, organized in a network. On a higher level of abstraction, an architecture is an algebraic operator on the behaviors of processes (described as finite-state machines) that generalizes the well-known parallel composition operators, such as synchronous (rendez-vous) or asynchronous (message-passing) schemes. The Separation Logic of Architectures (SLA) describes architectures that are not necessarily fixed a priori, but can be modified (reconfigured) during execution. Examples include: migrating/mobile applications, adaptive systems such as autonomous vehicles, etc.

The goal of this workshop is the study of several aspects of SLA: (1) decision procedures for various fragments and extensions of the logic, (2) axiomatisation of the execution semantics of distributed systems based on the principle of locality, and (3) compositional verification of correctness properties for systems
whose architectures are described using SLA. The internship comprises theoretical as well as implementation work.

References

