Candidature, bourse de thèse Labex
OASIS – FOCUS

June 21, 2013

Title: Deadlock analysis for concurrent and distributed programming

Thesis advisors

Ludovic Henrio (HDR) - CNRS / I3S
Team: OASIS (INRIA/CNRS/I3S/Université de Nice Sophia-Antipolis)

Cosimo Laneve (Professor) - Università di Bologna
Team: FOCUS - INRIA

Subject

This thesis is in the field of programming languages, especially languages for programming multicore and distributed applications. The overall objective of this thesis is to study and improve the state or the art of programming languages dedicated to these architectures. We aim at improvements both in terms of ease of programming and in terms of efficient and safe execution of programs.

Background: Programming distributed systems is a challenging task, and providing a tool suite for helping the programmer to write correct programs over concurrent and distributed systems is crucial. For this, we believe that the approach advocated by actors and active objects is valuable. Indeed, these models focus on the asynchrony between remote entities and by nature decouple the execution flow in each distributed entity as much as possible. This approach is particularly relevant for programming large scale distributed systems and/or service-oriented applications. An active object is an object associated with a single thread of execution. Unfortunately, active-objects are not as efficient as they could be when it comes to execution on multicore machines. To fully use the computing power of multicore architectures, different threads should share access to the same memory, which is contradictory with the principles of active objects.

The OASIS team is working on the design of programming paradigms, languages, and middleware for distributed systems. Recently, the team proposed a new programming model called "multi-active objects". The multi-active objects benefit from the ease of programming provided by the active objects and actors, while overcoming their limitations in terms of expressiveness and efficiency on multicore machines. This model is among the first high-level programming paradigms tackling both distributed and parallel computing and it is extremely promising in terms of ease of programming and efficiency. Multi-active objects offer a new compromise between the model of strict and secure programming proposed by active objects, and classical multi-threaded approach, very tolerant but very prone to errors. However, the programming model for multi-active objects also requires new advances in the fundamental study of distributed programming languages to prove the properties and to formalize this new programming model.

The FOCUS team has also a solid experience in distributed programming languages and their verification. In the past they have prototyped a distributed pi-calculus language with XML

datatypes, called PiDuce, and studied its relation with web services. More recently, the FOCUS team has been involved in the HATS European project, where the distributed actor-based language ABS has been developed. This language has a model similar to the one of active objects. The major contribution of the FOCUS team has been the development of a deadlock analysis framework for ABS programs.

Locks, Impact and Challenges: The expected impact of this thesis is to contribute to the formal design and the practical implementation of multi-active objects. Mostly, we want to benefit from this thesis to show that our programming model is easy to use, can be formally specified, and is efficient in practice. This thesis will particularly focus on the formal specification and on the support for writing programs that run safely. One of the major source of bugs in concurrent and distributed systems is deadlocks. Deadlocks come from the fact that most programs rely on synchronization points, where one task is waiting for the completion of another task before being able to finish. If the graph of synchronization dependencies form a cycle, then there is a deadlock. Finding deadlocks in distributed programs requires a strong expertise. Alleviating the programmer from the difficult task by providing him/her with automatic tools able to detect automatically those deadlocks will be a valuable contribution. Indeed, despite the ease of programming featured by active objects and the fact that multi-threading allows us to avoid several misbehaviours, this programming model is far from being deadlock free.

We want the relevant information for detecting deadlock to be automatically inferred from the program, namely the information needed to statically build the graph of synchronization dependencies. Such information in our experience takes the form of (behavioral) types, which are the input of a deadlock analyzer. This is a more complex task than a mere adaptation of the tool already developed for ABS, for several reasons. First, even if ABS and multi-active objects have similar concurrency models, the impact of their differences must be fully understood. For instance, ABS provides an explicit blocking operator for retrieving future values, while in the multi-active objects paradigm this is an implicit operation performed at the first access. Second, the multi-active object paradigm provides powerful user-defined annotations and scheduling policies, which should be carefully managed by the inference engine in order to properly recognize dependencies between threads. Third, once the inference system has been developed, then it could represent a basis for inferring other kind of information from the code, such as tracking resource access information. Therefore, the system might be able to verify other properties related to the interplay of concurrent threads – e.g. noninterference of methods (in order to maximize the parallelism).

Objectives and expected results: This thesis proposal contribute to the formal model of multi-active objects and to their practical implementation. In particular, we will provide tools for the static analysis of deadlocks. As regards this last point, while several works already focus on the safe design of distributed applications, those works are not massively adopted in practice. The main originality of our approach is twofold. First we rely on a programming model that enforces a strong separation between remote entities. This helps us for automatically inferring behavioral types for those entities and will make our analysis more compositional, more scalable, and thus better adapted to real applications. Second, the design of our language will be mainly driven by the efficiency of program execution, by the expressiveness of the model, and by the easiness of writing distributed applications.

The main objectives of this thesis are the following ones:

- Contribution to the programming model. The precise definition of the semantics of multi-active objects, a comparison with existing active-object models, and the analysis of the potential deadlocks. At this points, several variants of the programming language will be studied, some of them might simplify the deadlock analysis or make it more powerful. Being able to study such languages restrictions will guarantee the success of this thesis and will allow us to study different categories of applications.

- Definition of a deadlock analysis algorithm for multi-active objects. This objective should be divided in the following steps:

    1. Identify which are the synchronization points in the multi-active objects as opposed to the one already identified for ABS, and define an inference system. A first step in this direction would be the definition of the inference system for a variant of ABS without explicit synchronization (get) operation. This step will bridge the semantic gap between the two languages.
    2. Extend the preliminary inference system to the full multi-active object model.
    3. Possibly adapt/extend/empower the deadlock analyzer developed for ABS.

Complementarity of the two teams: We will definitely benefit from the complementarity of the two research teams that are involved in this proposal. The OASIS team has strong experience in the practical implementation of middleware and languages for large-scale distributed computing. The FOCUS team has strong experience on program analysis and semantics of distributed programming languages. We are therefore convinced that the interaction between the two teams will allow us to address the above objectives and will allow the PhD student to achieve an original and meaningful contribution.