Software Verification for Hardening Eclipse Golo in the context of Connected Objects

Supervisors

- Olivier Simonin (olivier.simonin@insa-lyon.fr)
- Nicolas Stouls (nicolas.stouls@insa-lyon.fr)
- Julien Ponge (julien.ponge@insa-lyon.fr)

Abstract

In a world of connected things, objects have mobility and hardware heterogeneity. This last thing is the reason why most used languages are based on virtual machine, allowing to compile one and to run everywhere. In this context, Eclipse Golo is a language and a development toolkit providing lightweight language to design JVM based softwares, supporting natively code dynamicity. Golo exploits the invokeDynamic primitive of Java Virtual Machine to give more flexibility in the associated programming language. The problem addressed in this PhD thesis is the one of the formal verification of such softwares. How to statically gain confidence into software where pieces of code could be load at runtime ? Another approach consists in embedding a part of the verification as code, injected into software. Which tradeoff can we found between static analyze and embedded verification code in order to reduce the execution cost ?

Context

Golo [1] is a programming language, as well as a development toolkit, dedicated to the development of dynamic softwares. Recently accepted as an official Eclipse project [11], Golo is based on Java Virtual Machine and its invokedynamic primitive. The code produced by Golo is highly optimized for dynamic softwares (hot load or unload of modules). By the way, Golo offers an interesting alternative for connected things, with mobility and hardware heterogeneity.

However, code dynamicity and high level language imply complexity to verify the code. Many approaches already exist to statically or dynamically verify some good properties on object oriented source-code. For instance, Krakatoa [2] is a tool dedicated to proving correctness of Java code against some invariant properties, while some other approaches are oriented on some specific vulnerabilities [7], like hidden channel [5]. Such static approaches are interesting because they don't need to execute the system. However, if properties are too complex, a static analyze couldn't conclude.

In another way, some approaches are based on the automatic instrumentation of the produced code [3] or of the intermediate code [4,9]. Such approaches are interesting thanks to the expressiveness of checkable properties. However, they sometimes need to execute the code.

Finally, the limit of these works is reach while considering mobile code, where another kind of specification has to be verify dynamically, where specification primitives has to deal with hot load or unload of code [10]. Indeed, Golo hot load and class augmentation could be compare to the problems of model extraction from Ruby, Python or javascript source codes.

Objectives

The problem addressed in this PhD thesis is the one of the verification of such softwares. How to gain confidence into software where pieces of code could be load at runtime ? In a first step, the PhD candidate will propose a verification solution embedded into the golo compiler where some static a simple properties will be verified. In a second step, the contribution will deport at runtime
some part of the verification that could be check statically. Finally, the PhD candidate will propose some enhancements considering some compilation optimization to not include verification nodes in all instantiated sites.

References