Modular extraction of flow-graphs from Java bytecode

Our research group in TCS develops and implements a novel compositional technique for the formal verification of control-flow properties of software. One of its main tasks is the extraction of control-flow graphs[1] from software, which are special graphs that represent all possible paths that a program can traverse during its execution. We implemented the technique in a tool set called CVPP[2], which formally verifies Java software for control-flow safety properties. For example, check if an undesired sequence of method invocations can happen. Those tools are encapsulated by ProMoVer[3], which automates a particular verification scenario.

We have an initial version of the control-flow graph extractor from Java bytecode[4] implemented in the OCAML[5] language. It uses SAWJA[6], a framework for the static analysis of Java software. We plan to extend it in two ways. The first is to implement the extraction modularly. It means that the extractor should be able to produce flow-graphs even when we don't have the full implementation of the software by exploiting the information from interfaces. The second task is to add the extraction of exceptional flow executions. Exceptions are part of the semantics of Java language and can change the results of verification process.

Proposed Content:

- Generation of software models for formal verification.
- Research of software semantics.
- Novel technique for modular flow-graph extraction.

Expected Results

- A modular flow-graph extractor tool from Java bytecode based on SAWJA that supports exception handling.
- Quantitative comparison of the impact of the flow-graphs when exception handling is presented. Metrics would be graphs size and extraction time.
- Proposal of possible refinements of the extracted flow-graphs.

Required Competence

- Computer Science/ Engineering, with specialization in Formal Methods
- Knowledge of Java language, JVM and Java bytecode.
- Programming skills. Preferentially experience with functional languages.
- Prior knowledge of OCAML or any other ML-derived language is a plus.

Contact: Dilian Gurov (dilian@csc.kth.se)

1 http://en.wikipedia.org/wiki/Control_flow_graph
2 http://www.csc.kth.se/~dilian/Projects/CVPP/
3 http://www.nada.kth.se/~siavashs/ProMoVer/
4 http://en.wikipedia.org/wiki/Java_bytecode
5 http://caml.inria.fr/
6 http://sawja.inria.fr/