SW project NeVer

NetBeans Verification Platform

Motivation

As well as Eclipse [4], NetBeans [7] is a widely used IDE primarily designed for Java. Both Eclipse and NetBeans feature modular design ensuring easy extensibility. In fact, they have been extended to support other programming languages (e.g. C/C++), features, and technologies (e.g. versioning via CVS/SVN, JUnit testing, profiling …). Since Eclipse and NetBeans target the same end-user, they compete and strive to add any feature the other one does. Under these circumstances, it is rather surprising that, although there are projects targeting code verification for Eclipse (as e.g. [1], [3], [6], and [8]), there is no such effort to support code verification in NetBeans.

On the other hand, Java PathFinder (JPF) [5], which is a state-of-the-art open-source model checker for Java programs, misses a GUI. The proper GUI and integration into a widely used IDE could make JPF accessible to wide programmers' community.

The goals of the NeVer project are twofold. (i) NeVer aims at creation of a NetBeans module, which would enhance NetBeans with general GUI support for configuring, running, and visualizing results of Java program verification. The NeVer project will provide a clear API to be used by back-ends specific to concrete model checkers. (ii) As a proof-of-concept, a JPF-specific back-end will be created. Put together, NeVer will allow for model checking Java programs using JPF, while providing an API general enough for incorporation of other/future Java model checkers (e.g. Bogor [2]) as well.

Goals:

- **NeVer platform**
    - Support for unit-checking (= unit-testing using a model checker)
        - Generation of checking harnesses
        - Results summarization

      Expected approach:
      Either via extending JUnit support in NetBeans or creation of a new GUI

    - Checking/simulation GUI
        - Common GUI for configuration of a model checker
        - Common GUI for running model checkers
        - UI for analysis of error traces – error trace walkthrough (watches, stack, …)
        - UI for simulation mode – user-guided simulation ("random" choices)

      Expected approach:
      Using the NetBeans debugger API

- **JPF NeVer plugin**
  - Checking/simulation GUI bindings
  - Unit-checking bindings
    - Environment generation

Optional goals:

- General support for data and predicate abstraction
  - CEGAR loop support

Supervisor

Ondřej Šerý

Team

3-4 students

References

[1] Blast homepage: http://mtc.epfl.ch/software-tools/blast/.

[2] Bogor homepage: http://bogor.projects.cis.ksu.edu/.

[3] CBMC homepage: http://www.cs.cmu.edu/~modelcheck/cbmc/.

[4] Eclipse homepage: http://www.eclipse.org/.

[5] Java PathFinder homepage: http://javapathfinder.sourceforge.net/.

[6] JMLEclipse homepage: http://jmleclipse.projects.cis.ksu.edu/.

[7] NetBeans homepage: http://www.netbeans.org/.

[8] SATABS homepage: http://www.verify.ethz.ch/satabs/.