ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Software Component
Technology Group

Master Thesis

03/05/2007 – 09/04/2007

Automated Program Verifier for Java Bytecode

Samuel Willimann

Introduction

The Java Modeling Language (JML) follows the "design by contract" paradigm to remove potential ambiguity with regard to the software designer's intentions. It allows programmers to insert program specifications (such as preconditions, postconditions or invariants) directly into the Java source code, which can then be used by various tools to check whether the program fulfills the given specifications.

The Bytecode Modeling Language (BML) on the other hand is similar to JML in the sense that it can also be used to write program specifications directly in the program code. However, BML annotations are embedded directly into the Java Bytecode. This technique enables programmers to include their specifications directly in the compiled programs.

A Java program annotated with JML or BML has to be transformed into verification conditions (VCs) in order to be verified by a theorem prover. We use a tool called Boogie to generate those VCs, which is a modular, static program verifier and part of the Spec# programming system. Its main purpose is to verify programs written in Spec# (an extension of C# offering support for specification constructs). In the verification process, the program is first translated into an intermediate imperative language called BoogiePL, from which VCs are generated and passed to an automatic theorem prover. This allows us to use Boogie for verifying programs written in programming languages other than C# (annotated Java Bytecode in this particular case) by implementing a transformer from the chosen programming language to BoogiePL.

Task Description

A translator for converting annotated Java Bytecode to BoogiePL was implemented by Ovidio Mallo in his Master Thesis in Winter 2006/07. Partially based on this translator, the goal of the present thesis is to tackle some aspects that have not yet been covered, and to integrate the translator seamlessly into the whole software verification process so that the correctness of a BML annotated Java Bytecode program can be proven automatically.

Main Goals

- Optimize method calls by using BoogiePLs 'call' statement.
- Improve invariant checks (i.e. determine which invariants have to be checked and whether the invariant is admissible). The treatment of invariant checking has to be sound and modular.

Possible Extensions

- Introduce triggers to the BoogiePL axioms and test improvement of performance with case studies.
- Extend supported language of the BML-to-BoogiePL translator (if necessary).
- Integrate the UMBRA frontend into the transformer.
- Annotate source code of the translator with JML at critical points.

Supervisors: Prof. Peter Müller and Hermann Lehner