Chair of Programming Methodology

Master's Project

Embedding JML-annotated Programs in the Coq Proof System

Andreas Kägi

September 2008

Introduction. JML is a well-known specification language for Java programs. While using runtime assertions to ensure that a particular program instance meets its JML specification is already useful, it is highly desirable to statically prove this fact for all program instances. One possibility to achieve this goal is to embed a JML-annotated Java program within a theorem prover. Using previously defined syntax and semantics for Java and JML within the theorem prover, one can then prove the propositions resulting from the JML specification to be valid.

The goal of this master's project is to develop a frontend that translates a JML-annotated Java program into a set of input files for the Coq theorem prover. These generated files, specific to every program, complement the syntax and semantics common to all programs with the concrete program specification and implementation, as well as additional type and declaration information. The Java and JML source is first translated into a Coq embedding that supports the full syntax of JML. This embedding is then reduced into a form using only basic JML syntax, where all rewriteable constructs have been replaced.

The minimum requirements for successful (passing grade) completion of the project are:

1. Implementation of the Java and basic JML syntax within Coq providing a concrete representation of all existing abstract data types.
2. Definition and implementation of abstract data types within Coq for all additional JML constructs (full syntax) except for floating point arithmetic constructs.
3. Development of a translation frontend of JML/Java input files into a set of Coq output files supporting the full JML syntax. The development is done in Java and builds on top of the ESC/Java2 frontend.
4. Definition of a translation of the full JML syntax into the basic syntax within Coq.
5. Realisation of a case study providing a small library of examples demonstrating a correct translation of all formalised constructs.
6. Quality assurance:
    - Thorough documentation of all implementations
    - Test suite for the translator frontend covering all non-trivial methods (JUnit)
    - Specifications for all non-trivial methods of the translator (JML)
7. Three presentations and a project report

Possible extensions:

- Derivation of proofs that the implementation of the abstract data types behaves as specified.
- Verification of the Java programs from the case study with respect to their JML specification, using the previously realised Coq embedding.

Supervisors: Hermann Lehner, Prof. Dr. Peter Müller