Semester Project

Specification Model Library for the Interactive Program Prover Jive

In a co-operative effort, the Software Component Technology Group together with the Softwaretechnik Group at TU Kaiserslautern works on the development of an interactive program prover, Jive (Java Interactive Verification Environment). The tool enables its user to formally prove that a given Java program behaves as expected relative to its interface specifications.

Currently there is an ongoing project which aims at changing the specification language of Jive from first-order logic to a specification language that is specifically tailored to Java programs. This change will ease the task of writing interface specifications for programmers who do not have a solid background in mathematics or logic.

The specification language, called Java Modeling Language (JML), uses enriched Java expressions to specify properties of programs via pre- and post-conditions for methods and invariants for classes. Due to many similarities with Java the overhead of learning JML is negligible for programmers compared to other specification languages based on first-order logic. Specifications have to be precise and abstract. Therefore, so-called specification-only model classes are used to provide a mathematical model for data types such as sequences, sets, etc.

Instead of such model classes, the Jive tool works with abstract data types specified in predicate logic. To be able to use JML specifications in Jive, there has to be a corresponding abstract data type for each model class in the JML library. The goal of this project is to develop these abstract data type definitions in Isabelle, the theorem-prover of the Jive tool.

To achieve this goal, the student should define abstract data types for at least the three model classes JMLObjectSequence, JMLObjectToObjectRelation and JMLMath, where existing Isabelle theories should be reused as much as possible. The abstract data types have to be complemented by additional functions and lemmas to reflect all aspects specified in the corresponding model classes. If necessary, extra axioms and lemmas have to be defined to simplify reasoning about the abstract data types.