Formal Specification and Verification of Java Refactorings

Alejandra Garrido and José Meseguer
University of Illinois at Urbana-Champaign

SCAM'06
September 2006

Context

- Refactoring of Java programs and its formalization.

- Refactorings are small, usually interactive transformations that preserve program behavior. Refactorings improve code readability and extensibility, making it easier to reuse.

Motivation

1. Lack of formal specifications of refactorings, in terms of preconditions and transformation.

2. Lack of verification of refactoring correctness, i.e.,, there is generally no proof that a refactoring preserves behavior.

Maude provides us with:

- A framework for rewriting logic, which has been proved particularly suitable for source code transformations.
- A powerful way for specifying the syntax and semantics of programming languages.
- An interpreter that allows the direct execution of semantic specifications with high efficiency.

The Semantics of Java

- A rewrite theory (Σ_{Java}, E_{Java}, R_{Java}), though we restrict ourselves to the equational theory (Σ_{Java}, E_{Java})

- The syntax and semantics of Java are specified as a series of modules, each module concentrating on a particular language feature.

Using Maude for Refactoring

- We specify the refactorings with transformation rules in the form of Maude equations, based on the Maude specification of the *syntax* of Java.
- We prove the refactorings correct by reasoning about the Maude specification of the *semantics* of the Java entity being transformed.
- We get a correct refactoring engine for free, because of the executability of Maude specifications.

Examples

- Push Down / Pull Up Method
- Push Down / Pull Up Field
- Rename Temporary

Conclusions

This work shows how three important goals can be simultaneously achieved within the same framework:

1. Formally specifying refactorings for a language;
2. Proving them correct with respect to the language semantics;
3. Deriving a provably correct refactoring tool from the formal refactoring specifications.

Future Work

- Extend the library of basic refactorings for Java, together with their proofs, trying to mechanize those proofs.
- Generalize the approach by using Maude strategies in the definition of refactorings.
- Develop a generic library of provably correct refactorings, based on modular semantic definitions of language features.