Featherweight Java – exercise

Piotr Krzemiński

9th April 2014

There are two variants of the exercise: theoretical and practical. You have to pick at least one of them, solve it, and deliver the solution to pio.krzeminski@gmail.com. Deadline is 6th May 2014.

1 Theoretical Variant

We have convenient properties of FJ reduction relation stating about *cast-safety* of expressions:

- reduction preserves cast-safety
- progress of cast-safety programs
- no typecast errors in cast-safe programs

Which of them hold also for FGJ reduction?

Define *cast-safety* property for FGJ expressions and formulate similar theorems for FGJ programs. Prove them formally or negate by showing a counterexample.

2 Practical Variant

In this variant your task is to provide implementation of:

- typechecker for FJ programs
- call-by-value[1] evaluator for FJ expressions
- type erasure (translation of FGJ programs to FJ ones)

Beside the implementation, you have to provide formal semantic description for call-by-value FJ reduction in any format you like (big-step, s.o.s, denotational, CPS, etc.). It may be pretty PDF, but it is not obligatory. It might also be clear and readable ASCII text file.

Provide some interesting examples as both FJ and FGJ programs. Suggestions:

- examples from original paper (those with `Pair` classes)
- encoding of natural numbers similar to Church's numerals using FJ class hierarchy + demonstration of recursion through `this` by defining operations on them (addition, multiplication, etc.)
- implementation of some well-known generic data structure (list, stack, binary tree, etc.)

The implementation has to be done in functional language (Haskell, OCaml, Scala, F#, etc.). Imperative code is forbidden.

[1]I know there are sources where CBV semantics is already formulated for Featherweight Java, but try to do it by yourself, not using them.