Well-typings for Java_λ
– Abstract –

Martin Plümicke

April 13, 2011

In several steps the Java type system is extended by features, which we know from functional programming languages. In Java 5.0 [GJSB05] generic types are introduced. Furthermore a reduced form of existential types (bounded wildcards) is introduced. For Java 8 it is announced, that closures (λ–expressions) should be introduced. In october 2010 [Goe] it was planed, against previous announcements, to omit function types. The reason was, that function types could confuse the programmer in several ways. Instead SAM types should be used. A SAM type is either an interface which declare just one method or an abstract class which declare just one abstract method.
To omit function types is a strong restriction.
To solve the problem, on the one hand to allow functions types and on the other hand to simplify the language, we introduce similar as in functional programming languages a type inference system. The corresponding algorithm is an adoption of Fuh and Mishra's type inference algorithm [FM88], which we adopted to the Java_λ type system.
The inferred types are well-typings. A well-typing is a conditioned type for an expression, where the conditions are given by a set of consistent coercions.

References

[FM88] You-Chin Fuh and Prateek Mishra. Type inference with subtypes. *Proceedings 2nd European Symposium on Programming (ESOP '88)*, pages 94–114, 1988.

[GJSB05] James Gosling, Bill Joy, Guy Steele, and Gilad Bracha. *The Java^{TM} Language Specification*. The Java series. Addison-Wesley, 3rd edition, 2005.

[Goe] Brian Goetz. State of the lambda. `http://cr.openjdk.java.net/~Briangoetz/lambda/lambda-state-3.html`.