When Compilers Are Mirrors

Martin Odersky

EPFL
martin.odersky@epfl.ch
http://lampwww.epfl.ch/~odersky

Abstract. When compilers are reflective mirrors, interesting things happen. Reflection and compilers do tantalizing similar things. Yet, in mainstream, statically typed languages the two have been only loosely coupled, and generally share very little code. In this talk I explore what happens if one sets out to overcome their separation.

The first half of the talk addresses the challenge how reflection libraries can share core data structures and algorithms with the language's compiler without having compiler internals leaking into the standard library API. It turns out that a component system based on abstract types and path-dependent types is a good tool to solve this challenge. I'll explain how the "multiple cake pattern" can be fruitfully applied to expose the right kind of information.

The second half of the talk explores what one can do when strong, mirror-based reflection is a standard tool. In particular, the compiler itself can use reflection, leading to a particular system of low-level macros that rewrite syntax trees. One core property of these macros is that they can express staging, by rewriting a tree at one stage to code that produces the same tree at the next stage. Staging lets us implement type and abstract syntax tree reification. What's more, staging can also be applied to the macro system itself, with the consequence that a simple low-level macro system can produce a high-level hygienic one, without any extra effort from the language or compiler.