Type-Based Linearization of the Lambda Calculus

Mário Florido
Universidade do Porto, DCC-FCUP

In this work we show that linear λ-terms are enough to fully characterize terms typable by Intersection Types. Intersection types originate in the work of Coppo and Dezani [CD80] and give a characterization of strongly normalizable terms. These systems type more terms than the Simple Type System for the λ-calculus, but they are rather complex systems whose definitions can be quite difficult to understand.

Here we address the following problem: to which extent can we approximate a typed term in the intersection type system by a linear term?

We present a notion of *term expansion*. Under that notion we show that one can define the *linear version* of a term if and only if the term is typable in an intersection type system (which types exactly the strongly normalizable terms). We then show that every term typable in an intersection type system has an expansion which is a linear term typable in the Curry type system (which is the simplest type system for the λ-calculus). Finally we prove that linear versions are preserved by weak head reduction, a notion of reduction considered by functional languages. This shows that expansion preserves computational behaviors of programs. These results show that it is possible to fully characterize a large class of functions (the functions definable by the strongly normalizable terms) with a rather simple class: the linear λ-terms.

This work is mainly based on results previoulsy presented in [FD04] and in ideas originated by work done in [AF05].

References

[AF05] Sandra Alves and Mário Florido. Weak linearization of the lambda calculus. *Theoretical Computer Science*, 342(1):79–103, 2005.

[CD80] M. Coppo and M. Dezani-Ciancaglini. An extension of the basic functionality theory for the λ-calculus. *Notre Dame Journal of Formal Logic*, 21(4):685–693, 1980.

[FD04] Mário Florido and Luís Damas. Linearization of the lambda-calculus and its relation with intersection type systems. *Journal of Functional Programming*, 14(5):519–546, 2004.