Efficient Unification of Recursive Second Order Types

Gauthier Nadji and Pottier François

February 10, 2004

Abstract

Although equirecursive types [2] and types with binders (e.g. universal or existential quantifiers) are widely understood, the combination of both has attracted attention only recently [3, 1]. The unification problem with those recursive second order types(RSOT) is at the core of a type deduction algorithm for example. Colazzo and Ghelli solved the problem of subtyping recursive types in kernel fun, and gave a non-intuitive algorithm whose complexity was only conjectured to be exponential. Glew formalized a theory for RSOT and gave a semantics of a type in terms of an infinite tree with binders and De Bruijn indices for bound variables. He also presented a quadratic algorithm for deciding the equality of two types, using a special class of automata, which gave insights into Colazzo and Ghelli's.

We slightly generalize the problem of equality to the one of unification, and give a decision procedure for it with time complexity $O(n\ log\ n)$. This is the major contribution of this work. The key remark leading to our algorithm is that unification of recursive types modulo α-equivalence can be reduced to unification without α-equivalence, by properly and efficiently choosing the names of the binding variables.

Besides we believe the framework we use (syntactic model with α-equivalence and equality by coinduction) and the algorithm are simpler and more intuitive than Glew's.

Finally we conjecture that the formulation of the algorithm allows to solve implication constraints as efficiently.

This is a work in progress, we are currently writing an article that explains in details the algorithm and investigate a little more around the central problem.

References

[1] Dario Colazzo and Giorgio Ghelli. Subtyping recursive types in kernel fun. In *Logic in Computer Science*, pages 137–146, 1999.

[2] Vladimir Gapeyev, Michael Levin, and Benjamin Pierce. Recursive subtyping revealed. In *International Conference on Functional Programming (ICFP), Montreal, Canada*, 2000.

[3] Neal Glew. A theory of second order trees. In *European Symposium on Programming 2002*, pages 147–161 of LNCS 2305, 2002. URL: `http://glew.org/nglew/papers/tsot-esop.ps`.