Adding Haskell-stlye Overloading to Curry

Wolfgang Lux

University of Münster
`wlux@uni-muenster.de`

Abstract. The integrated declarative language Curry [Han06] aims at providing a common foundation for research and education in the domain of functional logic languages. Its syntax and semantics are very closely related to the lazy functional language Haskell [Pey03], but currently it lacks some of Haskell's more advanced features. The most important of these is Haskell's systematic approach to overloading with type classes. In this talk, we argue that type classes are a too important feature to be omitted. Type classes have proven very useful in Haskell over more than a decade now and are one of Haskell's well recognized features, increasing the user's ability to write generic and more concise code. This feature has been missed more than once by users of Curry.

Our own experience with adding type classes to the Münster Curry compiler shows that it is mostly straightforward, since the theory behind and the implementation of type classes are well studied [Aug93,PJ93,Jon95]. However, there are some problematic areas, including overloading of numeric literals, the implementation of unification and interaction with the user, the treatment of ambiguous types, and the soundness problems of rank-2 types in a functional logic language. We present design options for these issues and motivate the design decisions taken for the Münster Curry compiler.

References

[Aug93] Lennart Augustsson. Implementing Haskell overloading. In *Proc. FPCA '93*, pages 65–73. ACM Press, 1993.

[Han06] Michael Hanus (ed.). Curry: An integrated functional logic language. (version 0.8.2). `http://www.informatik.uni-kiel.de/~mh/curry/report.html`, 2006.

[Jon95] Mark P. Jones. Dictionary-free overloading by partial evaluation. *Lisp and Symbolic Computation*, 8(3):229–248, 1995.

[Pey03] Simon L. Peyton Jones, editor. *Haskell 98 Language and Libraries The Revised Report.* Cambridge University Press, 2003.

[PJ93] John Peterson and Mark P. Jones. Implementing type classes. In *Proc. PLDI'93*, SIGPLAN Notices 28(6), pages 227–236. ACM, 1993.