Lifting Curry's Monomorphism Restriction

Wolfgang Lux

University of Münster
wlux@uni-muenster.de

Abstract. The multi-paradigm declarative language Curry [Han03] combines features from modern functional, logic, and functional-logic programming languages. Curry's syntax is similar to that of the functional language Haskell [Pey03] and, also like Haskell, Curry's type system is based on the Hindley-Milner typing discipline [Hin69], which allows automatic type inference without declaring types explicitly. However, Curry's type system suffers from a monomorphism restriction that requires the types of all local variables to be monomorphic, whereas in the Hindley-Milner type system, the types of let-bound variables can be generalized polymorphically.

Curry's monomorphism restriction is a consequence of the presence of unbound logical variables whose type cannot be generalized. Yet, restricting all let-bound variables to monomorphic types unnecessarily rejects some perfectly sound programs and also is an obstacle to compiling Haskell programs with a Curry compiler. This paper shows how Curry's monomorphism restriction can be lifted for a broad class of definitions by means of a purely syntactic analysis. This analysis is related to the value restriction employed by ML-like languages in order to ensure type soundness of programs in the presence of mutable variables [WF94].

References

[Han03] Michael Hanus (ed.). Curry: An integrated functional logic language. (version 0.8).
http://www.informatik.uni-kiel.de/~mh/curry/report.html, 2003.

[Hin69] Roger Hindley. The principal type-scheme of an object in combinatory logic. *Transactions of the American Mathematical Society*, 146:29–60, 1969.

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

[WF94] Andrew K. Wright and Matthias Felleisen. A syntactic approach to type soundness. *Information and Computation*, 115(1):38–94, 1994.