Free Theorems Involving Type Constructor Classes

Janis Voigtländer

Institut für Theoretische Informatik
Technische Universität Dresden
01062 Dresden, Germany

voigt@tcs.inf.tu-dresden.de

Abstract

One of the strengths of functional languages like Haskell is an expressive type system. For example, free theorems [3] allow the derivation of useful statements about programs from their (polymorphic) types alone. And yet, some of the benefits this should hold for reasoning about programs seem not to be realised to full extent. For example, Haskell uses monads [1] to structure programs by separating concerns [4] and to safely mingle pure and impure computations [2]. A lot of code can be kept independent of a concrete choice of monad. This pertains to functions from the Prelude (Haskell's standard library) like

$$\mathit{sequence} :: \mathsf{Monad}\ \mu \Rightarrow [\mu\ \alpha] \to \mu\ [\alpha]\,,$$

but also to many user-defined functions. This is certainly a boon for modularity of programs. But what about reasoning?

Type signatures like above signify polymorphism not only over ordinary types (like α), but also over type *constructors* (like μ) restricted by *class constraints* (like Monad). We show how to obtain free theorems in such situations, and discuss interesting applications.

References

1. E. Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, 1991.
2. S.L. Peyton Jones and P. Wadler. Imperative functional programming. In *Principles of Programming Languages, Proceedings*, pages 71–84. ACM Press, 1993.
3. P. Wadler. Theorems for free! In *Functional Programming Languages and Computer Architecture, Proceedings*, pages 347–359. ACM Press, 1989.
4. P. Wadler. The essence of functional programming (Invited talk). In *Principles of Programming Languages, Proceedings*, pages 1–14. ACM Press, 1992.