Taming Selective Strictness

Daniel Seidel* and Janis Voigtländer
Technische Universität Dresden, 01062 Dresden, Germany
{seideld,voigt}@tcs.inf.tu-dresden.de

Abstract

Free theorems [Wad89] establish interesting properties of parametrically polymorphic functions, solely from their types, and serve as a nice proof tool. For pure and lazy functional programming languages, they can be used with very few preconditions. Unfortunately, in the presence of selective strictness, as provided in languages like Haskell, their original strength is reduced [JV04]. We present an approach for restrengthening them. By a refined type system which tracks the use of strict evaluation, we rule out unnecessary restrictions that otherwise emerge from the general suspicion that strict evaluation may be used at any point. Additionally, we provide an algorithm determining all refined types for a given term. The algorithm has been implemented, and a web interface to it is available.[1]

For motivation, consider the well-known Haskell Prelude function *foldl* and its strict variant *foldl′* in the Haskell standard library `Data.List`. Both are of type $\forall\alpha.\forall\beta.(\alpha \to \beta \to \alpha) \to \alpha \to [\beta] \to \alpha$, and the corresponding free theorem, ignoring potential strict evaluation, states that $f\ (foldl\ c\ n\ xs) = foldl\ c'\ (f\ n)\ (map\ g\ xs)$, and analogously for *foldl′*, for appropriately typed c, c', n, xs and strict f, g such that $f\ (c\ x\ y) = c'\ (f\ x)\ (g\ y)$ for all x and y. In the untamed presence of selective strictness, certain additional restrictions would be required, namely that f and g are total ($f\ x \neq \bot$ and $g\ x \neq \bot$ for every $x \neq \bot$), $c = \bot$ iff $c' = \bot$, and for every x, $c\ x = \bot$ iff $c'\ (f\ x) = \bot$. But with our techniques we obtain that for *foldl* the equation holds without any of these additional restrictions, and that for *foldl′* it holds with only the single restriction that f is total.

References

[JV04] P. Johann and J. Voigtländer. Free Theorems in the Presence of seq. In *Principles of Programming Languages, Proceedings*, pages 99–110. ACM Press, 2004.

[Wad89] P. Wadler. Theorems for free! In *Functional Programming Languages and Computer Architecture, Proceedings*, pages 347–359. ACM Press, 1989.

*This author was supported by the DFG under grant VO 1512/1-1.

[1] `http://linux.tcs.inf.tu-dresden.de/~seideld/cgi-bin/polyseq.cgi`