Minimally Strict Polymorphic Functions

Jan Christiansen
Christian-Albrechts-Universität Kiel
jac@informatik.uni-kiel.de

Abstract

In a non-strict functional programming language like Haskell functions that yield the same results for total arguments can still differ for partial arguments. We can relate functions that agree for total arguments by a less-strict ordering. Naturally the question arises whether one can identify if a function is as non-strict as possible. A tool, called Sloth, assists programmers in checking whether a function is minimally strict.

To test a polymorphic function we have to choose a monomorphic instance of the function. By employing free theorems we show that a polymorphic function is indeed minimally strict if and only if its monomorphic Boolean instance is minimally strict. In fact, we only prove this statement for the polymorphic function type $\forall\alpha.[\alpha] \to [\alpha]$. But we can generalize the statement by employing the type classes *Foldable* and *Traversable*.