Invited Talk: The Sufficiently Smart Compiler can Prove Theorems!

Joachim Breitner
University of Pennsylvania
Philadelphia, Pennsylvania, USA
joachim@cis.upenn.edu

After decades of work on functional programming and on interactive theorem proving, a Haskell programmer who wants to include simple equational proofs in his code, e.g. that some Monad laws hold, is still most likely to simply do the proof by hand, as all the advanced powerful proving tools are inconvenient.

But one powerful tool capable of doing (some of) these proofs is hidden in plain sight: GHC, the Haskell compiler! Its optimization machinery, in particular the simplifier, can prove many simple equations all by himself, simply by compiling both sides and noting that the result is the same. Furthermore, and crucially to make this approach applicable to more complicated equations, the compiler can be instructed to do almost arbitrary symbolic term rewritings by using Rewrite Rules.

In this hands-on talk I will show how, with a very small amount of plumbing, I can conveniently embed the proof obligations for the monad laws for a non-trivial functor in the code, and have GHC prove them to me. I am looking forward to a discussion of the merits, limits and capabilities of this approach.

Submitted to:
WPTE17