A Resourceful Monad for IO

Luke Lau

April 29, 2020

Monads can be used to model side effects such as IO in functional programming languages such as Haskell. However monads enforce computation to be sequenced, so many libraries provide constructs for concurrency. But now we end up with traditional concurrency issues such as race conditions and simultaneous resource access.

One idea to help prevent such concurrency bugs is to keep track of the resources being used by the programs. Separation logic has been used for this purpose, albeit by making assertions about imperative programs. This dissertation creates a type system to make the same guarantees directly at the type level, and to help the programmer reason about what resources are in use. A proof of type soundness is given, which is also formalised within Agda. The final system is simple, pragmatic and integrates well with existing functional programming languages, with lots of interesting further directions to explore in.