Monads

Dhruv Rajan

May 29, 2017

1 Background

Functors are things that may be mapped over, in order to perform functions on values within certain contexts. In general, operate on a datatype `f a` with a function `a -> b` to produce a datatype `f b`. This abstraction is encoded in the `fmap` function: `(Functor f) => (a -> b) -> f a -> f b`

Applicatives are an improvement upon Functors, which allow the mapping function to itself be wrapped inside a context. Thus, the function `<*>` operates in the same way as `fmap`, besides this detail, and has type `(Applicative f) => f (a -> b) -> f a -> f b`.

Monads can be considered an extension of Applicative Functors. While applicatives are concerned with applying wrapped functions to wrapped values, Monads are concerned with applying a function which /takes/ an unwrapped value and returns a wrapped value to a wrapped value. That is, the ability to apply a function `>>= :: (Monad m) => m a -> (a -> m b) -> m b`