Exceptionally Monadic Error Handling

Looking at *bind* and squinting really hard

Jan Malakhovski[*]

IRIT, University of Toulouse-3 and ITMO University

February 2014 - October 2018

Abstract

We notice that the type of `catch :: c a -> (e -> c a) -> c a` operator is a special case of monadic `bind` operator `(>>=) :: m a -> (a -> m b) -> m b`, the semantics (surprisingly) matches, and this observation has many interesting consequences.

For instance, the reader is probably aware that the monadic essence of the `(>>=)` operator of the error monad $\lambda A.E \vee A$ is to behave like identity monad for "normal" values and to stop on "errors". The unappreciated fact is that handling of said "errors" with a `catch` operator of the "flipped" "conjoined" error monad $\lambda E.E \vee A$ is, too, a monadic computation that treats still unhandled "errors" as "normal" values and stops when an "error" is finally handled.

We show that for an appropriately indexed type of computations such a "conjoined" structure naturally follows from the conventional operational semantics of `throw` and `catch` operators. Consequently, we show that this structure uniformly generalizes *all* conventional monadic error handling mechanisms we are aware of. We also demonstrate several more interesting instances of this structure of which at least bi-indexed monadic parser combinators and conventional exceptions implemented via continuations have immediate practical applications. Finally, we notice that these observations provide surprising perspectives on error handling in general and point to a largely unexplored trail in programming language design space.

Contents

[*] papers@oxij.org; preferably with paper title in the subject line