Algebraic Effects

Algebraic effects, monad transformers, category theory, catamorphisms, zygomorphisms. What do all of these words have in common? They are also scary-fancy words made up by mathematicians to describe some really cool concepts in functional programming.

If scooby-doo has taught me anything, it is that when you take that scary mask off the ghost, you see a normal person who does normal things. Except for the wearing scary mask part, that is a bit weird.

So, let's take the mask off this bad guy. Let's start by understanding what a being effectful really means.

What is an effect?

An effect is any operation in your program that is interacting with something outside your program.

Problem with side effects

Testability

The haskell way of doing it

Monads and composition (maybe?)

Monad transformers (maybe?)

Algebraic effects

Algebraic effects as a concept is very new and has only been implemented in a few languages. It is still a topic of research in the functional programming world. OCaml is one of the languages that are working on pushing algebraic effects in production. Languages like eff, koka have effects as their first-class citizens which makes their design choices very interesting. The programs written in those languages are extremely testable.

What's so algebraic about it?

React hooks and redux saga

AE library

How to test ae programs

The normal way

The sexy way

Conclusion