Applicative Parsing

Julie Moronuki

LambdaConf Winter Retreat

% January 10, 2017

Hour 1: Applicative

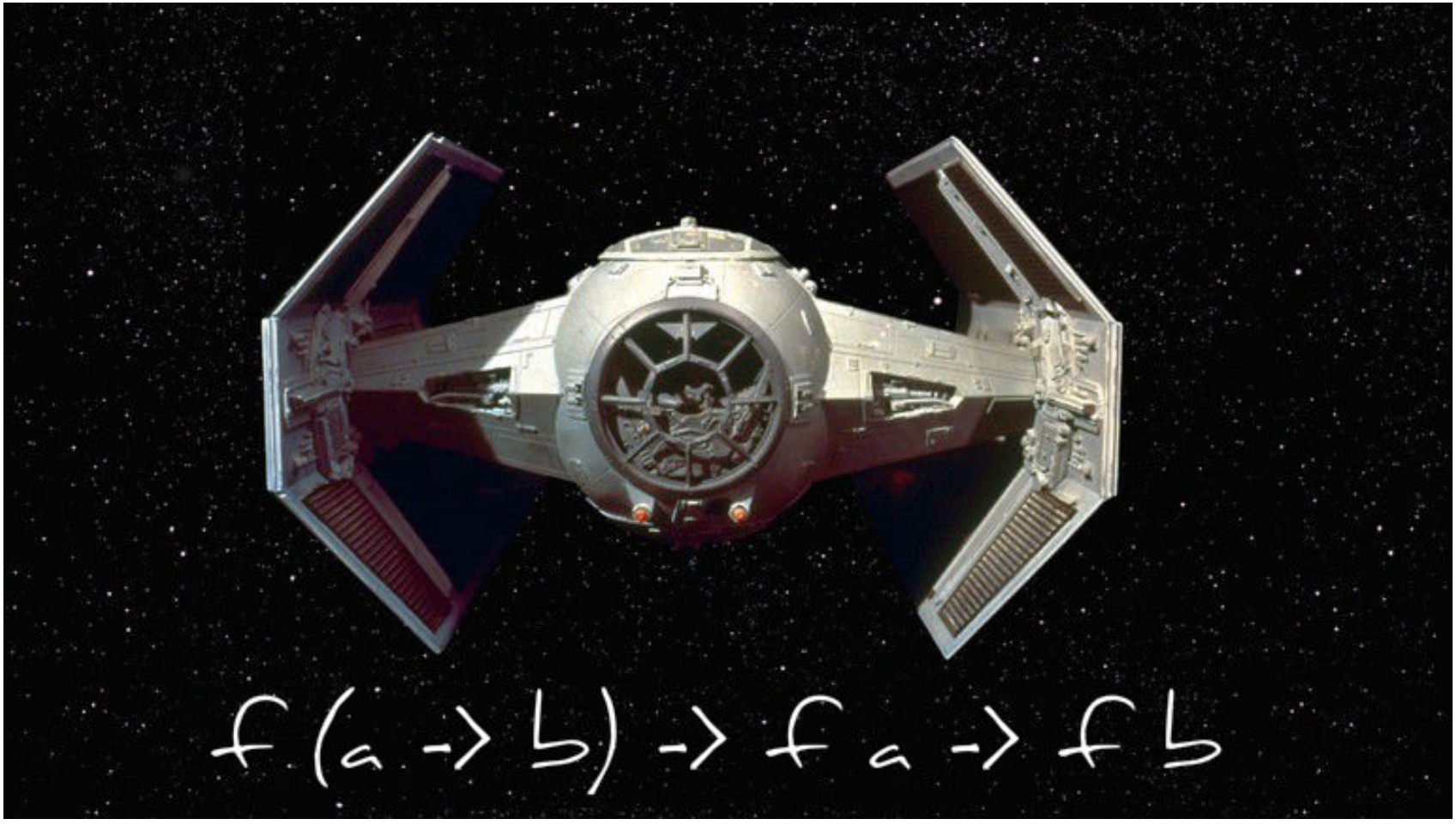

Figure 1: Tie Fighter of Doom

Functor

Monad

partial application

- see here: https://www.schoolofhaskell.com/school/advanced-haskell/functors-applicative-functors-and-monads#partial-application
- but use example from whatever code you've been working on

Applicative

Figure 2: I have altered the Functor.

Applicative

- the two functions must be independent, not relying on each other for outcome
- if you are gonna use the palindrome thing, then you'll need two parameters, not just one input string
- oh maybe do an anagram checker?

Applicatives vs Monads

- context
- composability (applicatives compose; monads need transformers)

Examples of monadic code and applicative code

- Validation
- gonna need good examples

Applicative Do

Parsing

Monadic parsing

- Parsec?

Alternative

Applicative parsing

- usually context free due to the independent outcomes quality
- can also be used to parse context-sensitive grammars tho!
- do not address this - reference to Yorgey's post about it

Examples of monadic and applicative parsing

- context free and context sensitive

Hour 2: Electric Boogaloo

In this hour, we'll be working on a small project with the optparse-applicative library. # Example
– optex – stack exec optex

Options.Applicative.Builder

Here are some basic argument types we can use: commands and flags.

```
command :: String -> ParserInfo a -> Mod CommandFields a
```

Add a command to a subparser option.

```
flag :: a  -> a -> Mod FlagFields a -> Parser a
--     [1]   [2]         [3]               [4]
```

1. default value
2. active value
3. option modifier
4. Builder for a flag parser.

A flag that switches from a "default value" to an "active value"

Start working on address