Functional Mottoes

and their enumeration

A Pattern Emerges

In Bewl, I kept having to write functions that look like this:

```
def rm[X, S]: (S => S => X) => S => X =
  ssx => s => ssx(s)(s)
```

These are so weird that they spawned a subproject of their own

Functional Mottoes

The simplest examples:

```
def identity[X]: X => X =
  x => x

def composition[X, Y, Z]:
  (X => Y) => (Y => Z) =>
(X => Z) =
  xy => yz => x =>
yz(xy(x))
```

Beautiful. They almost write themselves!

More Little Gems

```
def mu[X, H]: ((((X => H) => H) => H) => H) => ((X =>
H) => H) =
  xhhhh => xh => xhhhh(_(xh))
```

```
def strength[X, Y, H]: (
  X => Y
  ) => (
  ((X => H) => H) => ((Y => H) => H)
  ) =
  xy => xhh => yh => xhh(x => yh(xy(x)))
```

It goes on, and on

```
def em[X, H, A]: (
  ((X => H) => H) => X
  ) => (
  (((A => X) => H) => H) => (A => X)
) =
xhhx => axhh => a =>
  xhhx(xh =>
    axhh(ax =>
      xh(ax(a))
    )
  )
```

Mottoes Defined

- A *functional motto* is a function in Scala whose arguments are types, and which "almost writes itself" in the sense that there is only one legal way to construct something matching the type signature (other than returning **Null** or **???**)

What are they?

- As written here, legal Scala
- "Solving" them is fun, like doing crossword puzzles
- If you stare at them for long enough, you start to crack the code: each one encodes some meaningful axiom or construction of category theory
- Conjecture: There aren't that many (of any given complexity / length). And maybe they can all be derived from some finite subset.

Mottoes in Bewl

```
override def tensorialStrength[
  Y <: ~
](
  daa: DOT[Y]
) = (
  (dash x daa) > dot > dot
  ).transpose {
  dash x (daa > dot > dot)
} {
  case (
    (x, yss),
    xys
    ) =>
    yss(
      (dot > daa(dash x daa) {
          y => dash x daa pair(x, y)
        }) (
          xys
        )
    )
}
```

This is entirely mechanical, but took ages to figure out. Can it be automated?

Mottoes: The Mission

- Write a meta-program to enumerate *all* functional mottoes (up to equivalence)
- List all "expression types", then calculate their mottoes
- This has involved learning about lazy streams in Scala, to do the enumeration…
- …and parsing / recursing over expression trees
- See github.com/fdilke/scala-exps
  package `com.fdilke.mottoes`

What is the punchline?

- I'll either discover new categorical axioms/ constructions, or demonstrate systematically that the existing ones are enough
- The Bewl equivalents to these mottoes, dealing with values in an arbitrary topos, are not so pretty. And they still "write themselves", but it takes all afternoon
- But with an enhanced DSL, that could be fixed!

Thank You