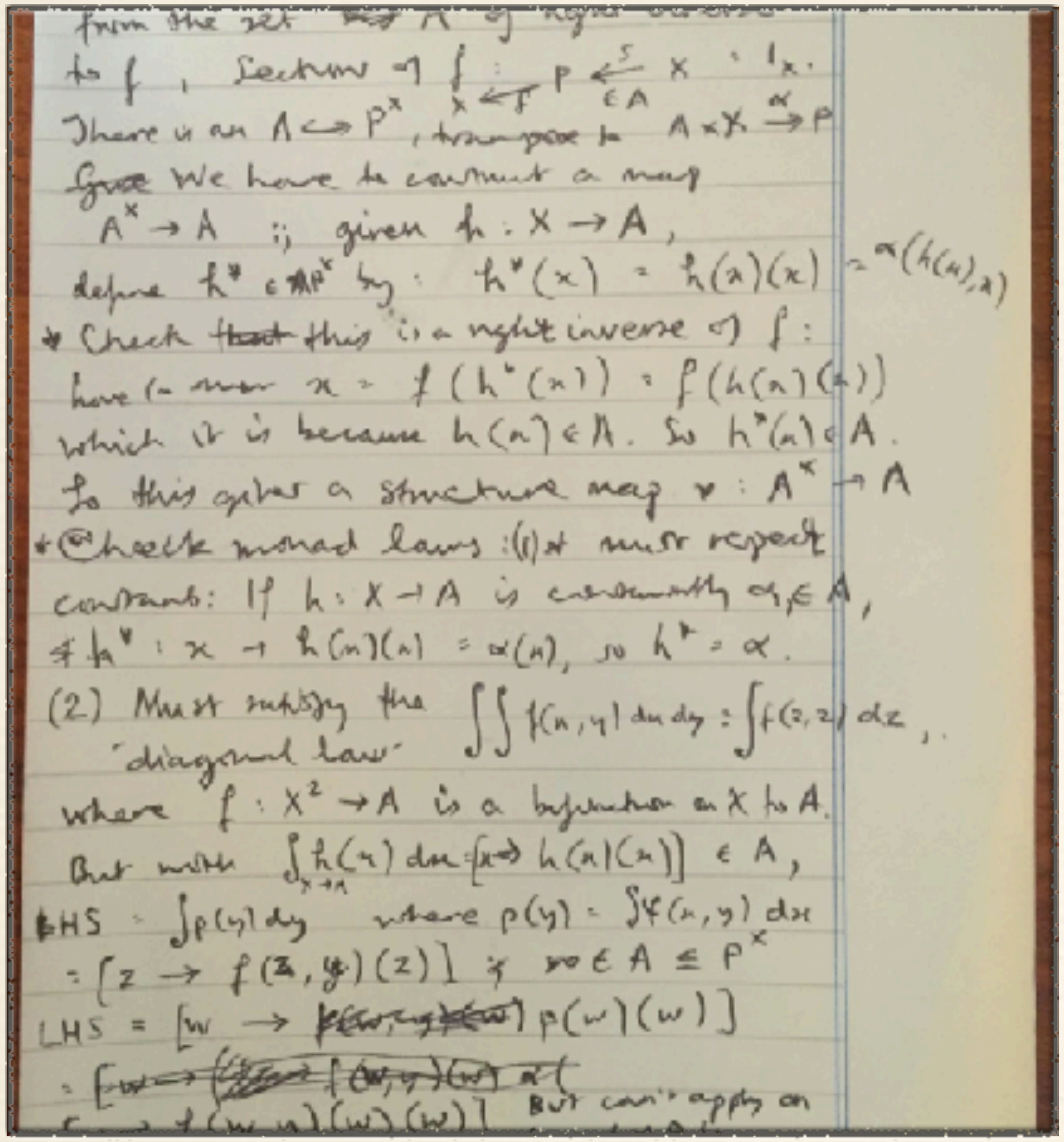
Strong Monads

github.com/fdilke/bewl



There is an $A \hookrightarrow P^X$, transpose to $A \times X \xrightarrow{\alpha} P$

We have to construct a map

$A^X \to A$;; given $h : X \to A$,

define $h^* \in P^X$ by : $h^*(x) = h(x)(x) = \alpha(h(x), x)$

Check this is a right inverse of f :

which it is because $h(x) \in A$. So $h^*(x) \in A$.

So this gives a structure map $\alpha : A^X \to A$

Check monad laws : (1) it must respect

constants: If $h : X \to A$ is constantly $a \in A$,

(2) Must satisfy the $\iint f(x,y)\,dx\,dy = \int f(z,z)\,dz$

"diagonal law"

where $f : X^2 \to A$ is a function on X to A.

The usual disclaimer

It's hard core math, dumbed down into 10 slides

Monads and strong monads

- Why it took me a while to figure out that strength is even necessary
- Recap on monads
- Why this isn't enough - they need an extra structure
- What is the extra structure?
- How do you express this in code?

It took a while to realise I needed strong monads, because there are many misconceptions

- When computer scientists talk about monads, they usually mean “strong monads” in the sense of category theory
- In a category with exponentials, a strong monad is a monad that has extra structure respecting the exponential
- So it’s an abuse of language (which I’ll use anyway) to ask if a given monad is strong. The question should really be “does it come with a naturally occurring strength”?
- The monads occurring in CS are all strong
- All monads on the category of sets are automatically strong
- In fact you have to go to some trouble to find a monad that is NOT strong (i.e. can’t be the monad part of a strong monad)
- Actually, the subset of functionality of a strong monad that makes them work with for comprehensions is the applicative functor, of which more anon

Monads explained in one slide

- In Scala, classes like **List**, **Option** and **Future** can be used in *for comprehensions*
- This is because they have certain features in common
- For example, given an instance of X, you can create an instance of a **List[X]**
- and you can collapse a **List[List[X]]** into a **List[X]**
- Same with the others. They are all *monads*

But… We want more

- Consider the set R of real numbers (usually viewed as a line)
- It has a *ring* structure - the operators 0,1,+,-,* are defined on R
- Given a bunch of functions with real values, adding and multiplying them is the most natural thing in the world. So…
- Given some other object X, the mappings from X to R inherit the ring structure. You can do arithmetic with them!
- Now generalise out of sight

Why monads need to be strong

- If R is a ring, we can make R^X into a ring
- More generally, if R is an object in a category with exponentials, and R has been given a structure, R^X should have that structure too
- A lot of algebraic structures can be encoded as monads. For example, there is a monad whose algebras are precisely rings
- For algebras over a monad to have this nice $R \to R^X$ property, we need the monad to be equipped with a *strength*

What is the exact definition?

Strong monad

From Wikipedia, the free encyclopedia

In category theory, a **strong monad** over a monoidal category $(C, \otimes, I)$ is a monad (T, η, μ) together with a natural transformation $t_{A,B} : A \otimes TB \to T(A \otimes B)$, called (*tensorial*) *strength*, such that the diagrams

$$\begin{tikzcd}
I \otimes TA \arrow[r, "t_{I,A}"] \arrow[dr, "\lambda_{TA}"'] & T(I \otimes A) \arrow[d, "T(\lambda_A)"] \\
& TA
\end{tikzcd},
\quad
\begin{tikzcd}
(A \otimes B) \otimes TC \arrow[rr, "t_{A \otimes B,C}"] \arrow[d, "\alpha_{A,B,TC}"'] & & T((A \otimes B) \otimes C) \arrow[d, "T(\alpha_{A,B,C})"] \\
A \otimes (B \otimes TC) \arrow[r, "1_A \otimes t_{B,C}"] & A \otimes T(B \otimes C) \arrow[r, "t_{A,B \otimes C}"] & T(A \otimes (B \otimes C))
\end{tikzcd},$$

$$\begin{tikzcd}
A \otimes B \arrow[r, "1_A \otimes \eta_B"] \arrow[dr, "\eta_{A \otimes B}"'] & A \otimes TB \arrow[d, "t_{A,B}"] \\
& T(A \otimes B)
\end{tikzcd}, \text{ and }
\begin{tikzcd}
A \otimes T^2B \arrow[r, "t_{A,TB}"] \arrow[d, "1_A \otimes \mu_B"'] & T(A \otimes TB) \arrow[r, "T(t_{A,B})"] & T^2(A \otimes B) \arrow[d, "\mu_{A \otimes B}"] \\
A \otimes TB \arrow[rr, "t_{A,B}"] & & T(A \otimes B)
\end{tikzcd}$$

These 4 rather horrific diagrams have to commute

Luckily, all this is easy to code up in Bewl

```scala
object StrongMonad {
  trait At[
    M[X <: ~] <: ~,
    X <: ~
  ] extends Monad.At[M, X] {
    def tensorialStrength[
      Y <: ~
    ](
      dash: DOT[Y]
    ):
      X x M[Y] > M[X x Y]
  }
}
```

```scala
def sanityTest4[
  A <: ~,
  B <: ~
](
  a: DOT[A],
  b: DOT[B]
) =
  tensorialStrength(a, b) o (
    (a *- b) x (
      η(b) o (a -* b)
    )
  ) shouldBe
    η(a x b)
```

…and we can test-drive the
construction of strong monads.
Next time, I will explain the point of all this

"If people do not believe that mathematics is simple, it is only because they do not realize how complicated life is."

–John von Neumann

THANK YOU

http://github.com/fdilke/bewl