λet's talk about Y

Harry Schwartz

thoughtbot

February 17, 2016

Combinators

A combinator is a λ-term with no free variables.

- So $\lambda xy.xy(xy)$ is a combinator, since both x and y are bound, but
- $\lambda x.xy$ isn't, because y is free.

Especially interesting combinators have names

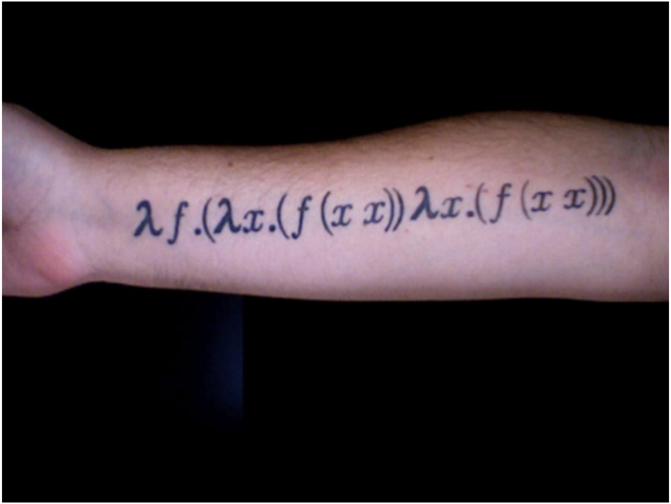
B	$\lambda xyz.x(yz)$
C	$\lambda xyz.xzy$
I	$\lambda x.x$
K	$\lambda xy.x$
S	$\lambda xyz.xz(yz)$
W	$\lambda xy.xyy$

SKI calculus, B,C,K,W system, etc...

The Y combinator

$$\lambda f.(\lambda x.f(x\ x))(\lambda x.f(x\ x))$$

People feel strongly about Y



Why Y?

The λ-calculus is Turing-complete, but it doesn't have an obvious mechanism for recursion/looping.

```
def factorial(n)
  if n == 0
    1
  else
    n * factorial(n - 1)
  end
end
```

The problem: How do we reference "the function we're in?"

How to achieve self-reference?

If we permitted self-reference, we could say:

$$f := \lambda x.(x == 0 \ ? \ 1 \ : \ x * f(x-1))$$

How to achieve self-reference?

If we permitted self-reference, we could say:

$$f := \lambda x.(x == 0 \; ? \; 1 \;\; : \;\; x * f(x-1))$$

Instead, we can pass f in as an argument:

$$F := \lambda f.\lambda x.(x == 0 \; ? \; 1 \;\; : \;\; x * f(x-1))$$

How to achieve self-reference?

If we permitted self-reference, we could say:

$$f := \lambda x.(x == 0 \ ? \ 1 \ : \ x * f(x-1))$$

Instead, we can pass f in as an argument:

$$F := \lambda f.\lambda x.(x == 0 \ ? \ 1 \ : \ x * f(x-1))$$

We need to find a p that satisfies $Fp = p$. This is the Big Idea.

Fixed-points

x is a *fixed-point* of f iff $f(x) = x$.

So if $Fp = p$, then p is a *fixed-point* of F.

Fixed-points in λ-calculus

For any f, $(\lambda x.f(x\ x))(\lambda x.f(x\ x))$ is a fixed-point of f.

Proof:

$$\begin{aligned} X &= (\lambda x.f(x\ x))(\lambda x.f(x\ x)) \\ &= \lambda[x := (\lambda x.f(x\ x))].f(x\ x) \\ &= f((\lambda x.f(x\ x))(\lambda x.f(x\ x))) \\ &= fX \end{aligned}$$

And here's the Y combinator again

So if we wanted a function that'd return a fixed-point of another function:

$$Y := \lambda f.(\lambda x.f(x\ x))(\lambda x.f(x\ x))$$

And here's the Y combinator again

So if we wanted a function that'd return a fixed-point of another function:

$$Y := \lambda f.(\lambda x.f(x\ x))(\lambda x.f(x\ x))$$

Equivalently:

$$YF = F(YF)$$

And here's the Y combinator again

So if we wanted a function that'd return a fixed-point of another function:

$$Y := \lambda f.(\lambda x.f(x\ x))(\lambda x.f(x\ x))$$

Equivalently:

$$YF = F(YF)$$

Y is an example of a *fixed-point combinator*.

Let's crank this out

$$
\begin{aligned}
3! &= YF\ 3 \\
&= F(YF)\ 3 \\
&= \lambda f.\lambda x.(x == 0\ ?\ 1\ :\ x * f(x-1))\ (YF)\ 3 \\
&= \lambda x.(x == 0\ ?\ 1\ :\ x * (YF)(x-1))\ 3 \\
&= 3 == 0\ ?\ 1\ :\ 3 * (YF)(3-1) \\
&= 3 * (YF)\ 2 \\
&= 3 * F(YF)\ 2 \\
&= 3 * (\lambda f.\lambda x.(x == 0\ ?\ 1\ :\ x * f(x-1))(YF)\ 2) \\
&= 3 * (\lambda x.(x == 0\ ?\ 1\ :\ x * (YF)(x-1))\ 2) \\
&= 3 * (2 == 0\ ?\ 1\ :\ 2 * (YF)(2-1)) \\
&= 3 * (2 * (YF)\ 1)
\end{aligned}
$$

Let's crank this out (cont.)

$$
\begin{aligned}
3! &= \cdots \\
&= 3 * (2 * (YF)\ 1) \\
&= 6 * (YF)\ 1 \\
&= 6 * F(YF)\ 1 \\
&= 6 * (\lambda f.\lambda x.(x == 0\ ?\ 1\ :\ x * f(x-1))(YF)\ 1) \\
&= 6 * (\lambda x.(x == 0\ ?\ 1\ :\ x * (YF)(x-1))1) \\
&= 6 * (1 == 0\ ?\ 1\ :\ 1 * (YF)(1-1)) \\
&= 6 * (YF)\ 0 \\
&= 6 * F(YF)\ 0 \\
&= 6 * (\lambda f.\lambda x.(x == 0\ ?\ 1\ :\ x * f(x-1))(YF)\ 0) \\
&= 6 * (0 == 0\ ?\ 1\ :\ 0 * (YF)(0-1)) \\
&= 6 * 1 \\
&= 6
\end{aligned}
$$

More stuff to read

- The article I cribbed this presentation from
- Wikipedia: Fixed-point combinator
- Wikipedia: SKI combinator calculus
- λ-calculus in the Haskell wiki
- Raymond Smullyan, *To Mock a Mockingbird*
- Douglas Hofstadter, *Gödel, Escher, Bach*