Arithmetic on Church numerals using a notational trick

Jules Jacobs

September 23, 2021

Abstract

Church naturals allow us to represent numbers in pure lambda calculus. In this short note we'll see how to define addition, multiplication, and power on Church nats using a cute notational trick. As a bonus, we'll see how to define predecessor and fast growing functions.

Church repesents a natural number n as a higher order function, which I'll denote $\mathbf{n}$. The function $\mathbf{n}$ takes another function f and composes f with itself n times:

$$\mathbf{n}\, f = \underbrace{f \circ f \cdots \circ f}_{n \text{ times}} = f^n$$

We can convert a Church nat $\mathbf{n}$ back to an ordinary nat by applying it to the ordinary successor function $S : \mathbb{N} \to \mathbb{N}$ given by $S\ n = n+1$: then $\mathbf{n}\ S\ 0$ gives us back an ordinary natural number n because $\mathbf{n}\ S\ 0$ is the n-fold application of the successor function to the number 0, which just increments it n times.

The first few Church natural numbers are:

$$\begin{aligned}
\mathbf{0} &\triangleq \lambda f.\lambda z.z \\
\mathbf{1} &\triangleq \lambda f.\lambda z.f z \\
\mathbf{2} &\triangleq \lambda f.\lambda z.f (f z) \\
\mathbf{3} &\triangleq \lambda f.\lambda z.f (f (f z))
\end{aligned}$$

Many descriptions of Church nats will view them in that way: as a function that takes *two* arguments f and z that computes $f(f(\ldots(f z)\ldots))$, but this point of view gets incredibly confusing when you try to define arithmetic on them, particularly multiplication and power. So think about $\mathbf{n} f = f^n$ as performing n-fold function composition.

If will be helpful to introduce an alternative notation for function application:

$$x^f \equiv f(x)$$

This may seem strange at first, but consider that using this notation we can *define* the first few Church natural numbers as:

$$\begin{aligned}
f^{\mathbf{0}} &\triangleq \mathsf{id} \\
f^{\mathbf{1}} &\triangleq f \\
f^{\mathbf{2}} &\triangleq f \circ f \\
f^{\mathbf{3}} &\triangleq f \circ f \circ f
\end{aligned}$$

Note that on the left hand side, we are really defining $\mathbf{3}$ as $\mathbf{3}(f) \triangleq f \circ f \circ f$.

The advantage of this notation is apparent when defining addition and multiplication on Church nats:

$$f^{\mathbf{n}+\mathbf{m}} \triangleq f^{\mathbf{n}} \circ f^{\mathbf{m}} \qquad\qquad f^{\mathbf{n}\cdot\mathbf{m}} \triangleq (f^{\mathbf{n}})^{\mathbf{m}}$$

Raising Church nats to a power is even better: our notation already makes $\mathbf{n^m}$ do the right thing:

$$\mathbf{n^m} \equiv \mathbf{m}(\mathbf{n}) \qquad \text{(already does the right thing!)}$$

The notation makes the proofs that this does arithmetic correctly a triviality: if $[n]$ is the Church nat corresponding to an ordinary natural number $n \in \mathbb{N}$, then

$$f^{[n]+[m]} = f^{[n]} \circ f^{[m]} = f^n \circ f^m = f^{n+m} = f^{[n+m]}$$

where $f^{[m]} \equiv [m](f)$ according to our notation, and f^n for ordinary natural number $n \in \mathbb{N}$ is n-fold function composition. The proofs for multiplication and power are analogous.

Predecessor

Surprisingly, defining the predecessor on Church nats is the most difficult. I think this is due to Curry.

We define the function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N} \times \mathbb{N}$:

$$f((a,b)) = (s(a), a)$$

If we start with $(0,x)$ and keep applying f we get the following sequence:

$$(0,x) \to (1,0) \to (2,1) \to (3,2) \to (4,3) \to \cdots$$

So

$$f^n((0,x))_1 = n$$
$$f^n((0,x))_2 = \begin{cases} x & \text{if } n = 0 \\ n-1 & \text{if } n > 0 \end{cases}$$

So we can define the predecessor function:

$$p = \lambda n.n\ f\ (0,0)$$

So that $p(0) = 0$ and $p(n) = n-1$ for $n > 0$.

Pairs

We made use of pairs to define the predecessor, so to use pure lambda calculus we need to define pairs in terms of lambda. We represent a pair (a,b) as:

$$(a,b) = \lambda f.f\ a\ b$$

We can extract the components by passing in the function f:

$$\mathsf{fst} = \lambda x.x\ (\lambda a.\lambda b.a)$$
$$\mathsf{snd} = \lambda x.x\ (\lambda a.\lambda b.b)$$

Disjoint union

Another way to define the predecessor is with disjoint unions. We take:

$$\mathsf{inl}(a) = \lambda f.\lambda g.f\,a$$
$$\mathsf{inr}(a) = \lambda f.\lambda g.g a$$

Then we can define:

$$f(\mathsf{inl}(a)) = \mathsf{inr}(a)$$
$$f(\mathsf{inr}(a)) = \mathsf{inr}(s(a))$$

We can do this pattern match on an inl/inr by calling it with the two branches as arguments:

$$f(x) = x\ (\lambda a.\mathsf{inr}(a))\ (\lambda a.\mathsf{inr}(s(a)))$$

And we can define:

$$p(n) = (n\ f\ \ \mathsf{inl}(0))\ (\lambda x.x)\ (\lambda x.x)$$

Fast growing functions

Given any function $g : N \to N$ we can define a series of ever faster growing functions as follows:

$$f_0(n) = g(n)$$
$$f_{k+1}(n) = f_k^n(n)$$

We can define this function using Church naturals:

$$f_k = k\ (\lambda f.\lambda n.nf\,n)\ g$$

If we take $g = s$ then,

$$f_0(n) = n + 1$$
$$f_1(n) = 2n$$
$$f_2(n) = 2^n \cdot n$$

The function $A(n) = f_n(n)$ grows pretty quickly. We can play the same game again, by putting $g = A$, obtaining a sequence:

$$h_0(n) = A(n)$$
$$h_{k+1}(n) = h_k^n(n)$$

To get a feeling for how fast this grows, consider h_1:

$$\begin{aligned} h_1(n) &= h_0^n(n) \\ &= A(A(A(\ldots A(A(n))))) \\ &= A(A(A(\ldots A(f_n(n))))) \\ &= A(A(A(\ldots f_{f_n(n)}(f_n(n))))) \end{aligned}$$

An expression like $h_3(3)$ gives us a relatively short lambda term that will normalise to a huge term. We might as well start with $g(n) = n^n$ since that's even easier to write using Church naturals:

$$\begin{aligned}
g &= \lambda a.a\ a \\
A &= \lambda k.k\ (\lambda f.\lambda n.n f\, n)\ g\ k \\
h &= \lambda k.k\ (\lambda f.\lambda n.n f\, n)\, A\, k \\
3 &= \lambda f.\lambda z.f\,(f\,(f\ z)) \\
X &= h\ 3
\end{aligned}$$

You can't write down anything close to the number X even if you were to write a hundred pages of towers of exponentials. Of course, we can continue this game, and define a sequence

$$\begin{aligned}
g_0 &= \lambda a.a\ a \\
g_1 &= \lambda k.k\ (\lambda f.\lambda n.n f\, n)\ g_0\ k \\
g_2 &= \lambda k.k\ (\lambda f.\lambda n.n f\, n)\ g_1\ k \\
&\dots
\end{aligned}$$

Which can be generalised as:

$$\begin{aligned}
f(g) &= \lambda k.k\ (\lambda f.\lambda n.n f\, n)\ g\ k \\
g_n &= f^n(g_0)
\end{aligned}$$

So we get an even more compact, yet much larger number with:

$$\begin{aligned}
f &= \lambda g.\lambda k.k\ (\lambda f.\lambda n.n f\, n)\ g\ k \\
Y &= (3\ f)\ (\lambda a.aa)\ 3
\end{aligned}$$

Of course, you can easily define much faster growing functions. But here's a challenge: what's the shortest lambda term that normalises, but takes more than the age of the universe to normalise? Or: what's the largest Church natural you can write down in less than 30 symbols?

Please let me know of any mistakes. I haven't checked for mistakes at all :)