

# ARITHMETIC ON CHURCH NUMERALS USING A NOTATIONAL TRICK

Jules Jacobs

September 25, 2021

## Abstract

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

## 1 Addition, multiplication, and exponentiation

Church represents 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 numeral  $\mathbf{n}$  back to an ordinary nat by applying it to the ordinary successor function  $S : \mathbb{N} \rightarrow \mathbb{N}$  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 numerals 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 numerals will view them in that way: as a function that takes *two* arguments  $f$  and  $z$  that computes  $f(f(\dots(f z)\dots))$ , but this point of view gets incredibly confusing when you try to define arithmetic on them, particularly multiplication and exponentiation. So think about  $\mathbf{n} f = f^n$  as performing  $n$ -fold function composition.

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

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

This may seem strange, but using this notation we can *define* the first few Church numerals as:

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

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

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

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

Exponentiation of Church numerals is even better: our notation already makes  $\mathbf{n}^{\mathbf{m}}$  do the right thing:

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

The proofs that this does arithmetic correctly look like a triviality when using our notation: if  $[n]$  is the Church numeral corresponding to an ordinary natural number  $n \in \mathbb{N}$  (i.e., satisfying  $f^{[n]} = f^n$ , 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), then

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

The proofs for multiplication and exponentiation are similar.

## 2 Predecessor

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

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

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

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

$$(0, x) \rightarrow (1, 0) \rightarrow (2, 1) \rightarrow (3, 2) \rightarrow (4, 3) \rightarrow \dots$$

So

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

So we can define the predecessor function:

$$p = \lambda \mathbf{n}. f^n(0, 0)$$

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

### 2.1 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  $f$ :

$$\begin{aligned} \text{fst} &= \lambda x. x \ (\lambda a. \lambda b. a) \\ \text{snd} &= \lambda x. x \ (\lambda a. \lambda b. b) \end{aligned}$$

## 2.2 Disjoint union

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

$$\text{inl}(a) = \lambda f. \lambda g. f a$$

$$\text{inr}(a) = \lambda f. \lambda g. g a$$

Then we can define:

$$f(\text{inl}(a)) = \text{inr}(a)$$

$$f(\text{inr}(a)) = \text{inr}(s(a))$$

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

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

And we can define:

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

## 3 Fast growing functions

Given any function  $g : N \rightarrow N$  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 numerals:

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

If we take  $g = S$  the successor function, 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  $h_1$ :

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

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 numerals:

$$g = \lambda a. a^a$$

$$A = \lambda k. (\lambda f. \lambda n. f^n n)^k g k$$

$$h = \lambda k. (\lambda f. \lambda n. f^n n)^k A k$$

$$\mathbf{3} = \lambda f. \lambda z. f(f f z)$$

$$X = h \mathbf{3}$$

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

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

Which can be generalised as:

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

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

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

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 numeral you can write down in less than 30 symbols?

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

— Jules