Church Numerals

202303150503

Type : #Note
Tags : Lambda Calculus

Church Numerals

Numerals

The most common way of defining numerals in lambda calculus is the definition given by *Alonzo Church*, which is also called the **Church Numerals**, which are defined as repeated application of a function of a value.

The Church Numerals are defined as

$$\begin{aligned} &0 = \lambda fx.\, x \\ &1 = \lambda fx.\, f\ x \\ &2 = \lambda fx.\, f\ (f\ x) \\ &3 = \lambda fx.\, f\ (f\ (f\ x)) \\ &\ \vdots \end{aligned}$$

Successor Function

We can define a SUCC function which finds the successor of a given church numeral by

$$\mathrm{SUCC} := \lambda\, n.\, (\lambda\ fx.\ \ f(n\ f\ x))$$

which add one f behind the chain of applied fs

> **Infix Notation**
>
> we can write $SUCC(n)$ as $n++$

Addition

Adding a number by n can be thought of repeatedly applying the successor function to it n times. Hence an addition function can be written as

$$\text{ADD} := \lambda\, mn.\; m\;\text{SUCC}\;n$$

which captures the above procedure.

Given the way that the church numerals are define, we can directly append the parameter f a certain number of times before the defintion, so another way to write the add function would be

$$ADD := \lambda\, mnfx.\; \underbrace{m\;f}_{(i)}\;\underbrace{(n\;f\;x)}_{(ii)}$$

(i) applying f m many times to
(ii) f applied to x n times.

> **Infix Notation**
>
> we can write $ADD(m, n)$ as $m + n$

Multiplication

We can define Multiplication as repeated addition as

$$\text{MULT} := \lambda\, mn.\; m\;(\text{PLUS}\;n)\;0$$

A smaller version to do would be

$$MULT := \lambda\, mn. (\lambda fx.\; \underbrace{n\;(\lambda x. mfx)}_{(i)}\;x)$$

(i) Here the term inside the bracket is a function which applies f m times on an input, and that is applied on x $n-$many times, which is the same as applied the function f on the input x $n * m$ times.
Applying eta reduction on the above function to make a bit cleanere gives us

$$MULT := \lambda\, mn. (\lambda f. n(mf))$$

Infix Notation

we can write $MULT(m, n)$ as $m \times n$

Exponentiation

Exponentiation can be defined as repeated multiplication, and that definition can be written as

$$EXP := \lambda mn.\, n\ (MULT\ m)\ 1$$

The shorter way to write exponentiation would be the following.
The procedure would be to apply f repeadly m times, then apply that m times, and that m times and so on again and again, repeating this process n many times. A numeral can be thought of taking a single function as an input and returning that applied repeatedly as an output.

$$EXP := \lambda mnfx.\ (n\ \underbrace{m(\lambda x.\, fx)}_{(i)})\ x$$

The part representing *doing f m many times* is represented by (i), and repeating that process n times can be thought of as the entire expression

and applying eta reduction would simply give

$$\text{EXP} := \lambda mn.\ n\ m$$

Infix Notation

we can write $EXP(m, n)$ as m^n

Predecessor

The predecessor function returns the precessor of the number in the normal sense except at 0 where it returns 0.
The definition of the predecessor uses pairs of consecutive numbers.
The procedure takes a number n as an input and works with the following sequence of pair

$$
\begin{gathered}
(0,0) \\
(0,1) \\
(1,2) \\
\vdots \\
(i, i+1) \\
(i+1, i+2) \\
\vdots \\
(n-2, n-1)
\end{gathered}
$$

at which point the second elment of the tuple is returned
we know when to stop because there are exactly n steps taken, so we can write the precessor function as

$$PRED := \lambda n. \left(n \underbrace{\lambda p. \left(p.2, p.2++\right)}_{\text{next step function}} (0,0)\right).2$$

Infix Notation

we can write $PRED(n)$ as $n--$

Subtraction

Subtraction can be defined as repeatedly applying the precessor function.

$$SUB := \lambda\, mn.\ n\, \text{PRED}\, m$$

Infix notation

we can write $SUB(m,n)$ as $m-n$

References

Church Booleans