Introduction to the Lambda Calculus

Jared Corduan

November 5, 2019

These slides are available at:
https://github.com/JaredCorduan/lambda-calc-cofc

models of computation

What is a computation?

- λ-calculus (1935, Church)
- μ-recursive functions (1935, Gödel)
- Post machines (1936, Post)
- Turing machines (1936, Turing)
- flow charts (1947, Goldstine and Von Neumann)
- register machines (1963, Shepherdson and Sturgis)

formalization

- Babylonian division algorithms date 2500 BC
- strong intuition
- why the 1930's?

Explosion of Math in the 1800's

- explosion of mathematics in the nineteenth century
- more abstract, less attached to the sciences
- nonconstructive proofs
- rise of first order logic via Frege and Peirce.

What is a good foundation for mathematics?

- potential infinity vs actual infinity
- sets or functions?
- types?
- What constitutes a proof?
- What is an algorithm?

Pricipia

- Pricipia Mathematica, by Alfred North Whitehead and Bertrand Russell in 1910
- Church introduces the lambda calculus
- entscheidungsproblem in 1935.

Influence on programming languages

- Lisp
- ALGOL 60
- ML
- Haskell

Lambda Calculus

In math class, you might see

$$f(x, y) = x^2 + y$$

$$f(1, 2) = 1^2 + 2$$

Lambda Calculus

In math class, you might see

$$f(x, y) = x^2 + y$$

$$f(1, 2) = 1^2 + 2$$

Or

$$x, y \mapsto x^2 + y$$

$$1, 2 \mapsto 1^2 + 2$$

$$1 \mapsto 1^2 + y$$

In math class, you might see

$$f(x, y) = x^2 + y$$

$$f(1, 2) = 1^2 + 2$$

Or

$$x, y \mapsto x^2 + y$$

$$1, 2 \mapsto 1^2 + 2$$

$$1 \mapsto 1^2 + y$$

Consider the lambda notation:

$$\lambda x.\lambda y.x^2 + y$$

$$(\lambda x.\lambda y.x^2 + y)(1) = \lambda y.1^2 + y$$

$$(\lambda y.1^2 + y)(2) = 1^2 + 2$$

Lambda Terms

Lambda Terms build up from:

- variables: x, y, f, ☕, etc
- abstraction: $\lambda x.M$, for a term M.
- application MN, for terms M and N.

Lambda Terms

Lambda Terms build up from:

- variables: x, y, f, ☕, etc
- abstraction: $\lambda x.M$, for a term M.
- application MN, for terms M and N.

Examples:

$$
\begin{array}{rcl}
I & = & \lambda x.x \\
I & = & \lambda \text{☕}.\text{☕} \\
K & = & \lambda x.\lambda y.x \\
S & = & \lambda x.\lambda y.\lambda z.xz(yz) \\
\Omega & = & (\lambda x.xx)(\lambda x.xx) \\
Y & = & \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))
\end{array}
$$

Substitution:

Substitute N for x in M:

$$M[x := N]$$

Substitution:

Substitute N for x in M:

$$M[x := N]$$

x🯅$x[x :=$ ☕$]$ $=$

Substitution:

Substitute N for x in M:

$$M[x := N]$$

$$x🯅x[x := ☕] \;=\; ☕🯅☕$$

Substitution:

Substitute N for x in M:

$$M[x := N]$$

Caveat emptor: care is needed to define capture-avoiding substitution, to avoid things like

$$(\lambda y.x)[y := x] = \lambda x.x$$

Reducible Expressions

A "reducible expression" or "redex" is a term of the form:

$$(\lambda x.M)N$$

Reducible Expressions

A “reducible expression” or “redex” is a term of the form:

$$(\lambda x.M)N$$

$\beta-$reduction:

$$(\lambda x.M)N \rightarrow_\beta M[x := N]$$

A “reducible expression” or “redex” is a term of the form:

$$(\lambda x.M)N$$

β−reduction:

$$(\lambda x.M)N \rightarrow_\beta M[x := N]$$

$(\lambda x.x$  $x)$  $\rightarrow_\beta$

Reducible Expressions

A “reducible expression” or “redex” is a term of the form:

$$(\lambda x.M)N$$

$\beta-$reduction:

$$(\lambda x.M)N \to_\beta M[x := N]$$

$$(\lambda x.x\,\text{🯅}\,x)\ \text{☕} \quad \to_\beta \quad \text{☕}\,\text{🯅}\,\text{☕}$$

Examples

$$SKK = (\lambda x.\lambda y.\lambda z.xz(yz))(\lambda a.\lambda b.a)(\lambda c.\lambda d.c)$$

Examples

$$SKK = (\lambda x.\lambda y.\lambda z.xz(yz))(\lambda a.\lambda b.a)(\lambda c.\lambda d.c)$$

Examples

$$\begin{aligned} SKK &= (\lambda x.\lambda y.\lambda z.xz(yz))(\lambda a.\lambda b.a)(\lambda c.\lambda d.c) \\ &\to_\beta \lambda y.\lambda z.(\lambda a.\lambda b.a)z(yz)(\lambda c.\lambda d.c) \end{aligned}$$

Examples

$$
\begin{aligned}
SKK &= (\lambda x.\lambda y.\lambda z.xz(yz))(\lambda a.\lambda b.a)(\lambda c.\lambda d.c) \\
&\to_\beta \lambda y.\lambda z.(\lambda a.\lambda b.a)z(yz)(\lambda c.\lambda d.c)
\end{aligned}
$$

Examples

$$
\begin{aligned}
SKK &= (\lambda x.\lambda y.\lambda z.xz(yz))(\lambda a.\lambda b.a)(\lambda c.\lambda d.c) \\
&\rightarrow_\beta \lambda y.\lambda z.(\lambda a.\lambda b.a)z(yz)(\lambda c.\lambda d.c)
\end{aligned}
$$

Examples

$$
\begin{aligned}
SKK &= (\lambda x.\lambda y.\lambda z.xz(yz))(\lambda a.\lambda b.a)(\lambda c.\lambda d.c) \\
&\to_\beta \lambda y.\lambda z.(\lambda a.\lambda b.a)z(yz)(\lambda c.\lambda d.c)
\end{aligned}
$$

Examples

$$
\begin{aligned}
SKK &= (\lambda x.\lambda y.\lambda z.xz(yz))(\lambda a.\lambda b.a)(\lambda c.\lambda d.c) \\
&\to_\beta \lambda y.\lambda z.(\lambda a.\lambda b.a)z(yz)(\lambda c.\lambda d.c) \\
&\to_\beta \lambda z.(\lambda a.\lambda b.a)z((\lambda c.\lambda d.c)z)
\end{aligned}
$$

Examples

$$\begin{aligned} SKK &= (\lambda x.\lambda y.\lambda z.xz(yz))(\lambda a.\lambda b.a)(\lambda c.\lambda d.c) \\ &\rightarrow_\beta \lambda y.\lambda z.(\lambda a.\lambda b.a)z(yz)(\lambda c.\lambda d.c) \\ &\rightarrow_\beta \lambda z.(\lambda a.\lambda b.a)z((\lambda c.\lambda d.c)z) \end{aligned}$$

Examples

$$
\begin{aligned}
SKK &= (\lambda x.\lambda y.\lambda z.xz(yz))(\lambda a.\lambda b.a)(\lambda c.\lambda d.c) \\
&\to_\beta \lambda y.\lambda z.(\lambda a.\lambda b.a)z(yz)(\lambda c.\lambda d.c) \\
&\to_\beta \lambda z.(\lambda a.\lambda b.a)z((\lambda c.\lambda d.c)z) \\
&\to_\beta \lambda z.(\lambda b.z)((\lambda c.\lambda d.c)z)
\end{aligned}
$$

Examples

$$
\begin{aligned}
SKK &= (\lambda x.\lambda y.\lambda z.xz(yz))(\lambda a.\lambda b.a)(\lambda c.\lambda d.c) \\
&\to_\beta \lambda y.\lambda z.(\lambda a.\lambda b.a)z(yz)(\lambda c.\lambda d.c) \\
&\to_\beta \lambda z.(\lambda a.\lambda b.a)z((\lambda c.\lambda d.c)z) \\
&\to_\beta \lambda z.(\lambda b.z)((\lambda c.\lambda d.c)z) \\
&\to_\beta \lambda z.(\lambda b.z)(\lambda d.z)
\end{aligned}
$$

Examples

$$
\begin{aligned}
SKK &= (\lambda x.\lambda y.\lambda z.xz(yz))(\lambda a.\lambda b.a)(\lambda c.\lambda d.c) \\
&\rightarrow_\beta \lambda y.\lambda z.(\lambda a.\lambda b.a)z(yz)(\lambda c.\lambda d.c) \\
&\rightarrow_\beta \lambda z.(\lambda a.\lambda b.a)z((\lambda c.\lambda d.c)z) \\
&\rightarrow_\beta \lambda z.(\lambda b.z)((\lambda c.\lambda d.c)z) \\
&\rightarrow_\beta \lambda z.(\lambda b.z)(\lambda d.z) \\
&\rightarrow_\beta \lambda z.z
\end{aligned}
$$

Examples

$$
\begin{aligned}
SKK &= (\lambda x.\lambda y.\lambda z.xz(yz))(\lambda a.\lambda b.a)(\lambda c.\lambda d.c) \\
&\rightarrow_\beta \lambda y.\lambda z.(\lambda a.\lambda b.a)z(yz)(\lambda c.\lambda d.c) \\
&\rightarrow_\beta \lambda z.(\lambda a.\lambda b.a)z((\lambda c.\lambda d.c)z) \\
&\rightarrow_\beta \lambda z.(\lambda b.z)((\lambda c.\lambda d.c)z) \\
&\rightarrow_\beta \lambda z.(\lambda b.z)(\lambda d.z) \\
&\rightarrow_\beta \lambda z.z \\
&= I
\end{aligned}
$$

Examples

$$\Omega \quad = \quad (\lambda x.xx)(\lambda x.xx)$$

Examples

$$\begin{aligned} \Omega &= (\lambda x.xx)(\lambda x.xx) \\ &\to_\beta (\lambda x.xx)(\lambda x.xx) \end{aligned}$$

Examples

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

Examples

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

$$\begin{aligned} YF &= (\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))F \\ &\to_\beta (\lambda x.F(xx))(\lambda x.F(xx)) \\ &\to_\beta F((\lambda x.F(xx))(\lambda x.F(xx))) \end{aligned}$$

Examples

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

$$\begin{aligned} YF &= (\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))F \\ &\to_\beta (\lambda x.F(xx))(\lambda x.F(xx)) \\ &\to_\beta F((\lambda x.F(xx))(\lambda x.F(xx))) \end{aligned}$$

$$\begin{aligned} F(YF) &= (\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))F \\ &\to_\beta F((\lambda x.F(xx))(\lambda x.F(xx))) \end{aligned}$$

Examples

$$
\begin{aligned}
Y &= \lambda f.(\lambda x.f(xx))(\lambda x.f(xx)) \\[1em]
YF &= (\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))F \\
&\rightarrow_\beta (\lambda x.F(xx))(\lambda x.F(xx)) \\
&\rightarrow_\beta F((\lambda x.F(xx))(\lambda x.F(xx))) \\[1em]
F(YF) &= (\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))F \\
&\rightarrow_\beta F((\lambda x.F(xx))(\lambda x.F(xx))) \\[1em]
YF &\sim_\beta F(YF)
\end{aligned}
$$

Arithmetic in the Lambda Calculus

- $0 := \lambda f.\lambda x.x$
- $1 := \lambda f.\lambda x.fx$
- $2 := \lambda f.\lambda x.f(fx)$

- $\text{SUCC} := \lambda n.\lambda f.\lambda x.f(nfx)$
- $\text{PLUS} := \lambda m.\lambda n.\lambda f.\lambda x.mf(nfx)$

One plus one is two!

$$\text{PLUS } 1\ 1 \quad = \quad (\lambda m.\lambda n.\lambda f.\lambda x.mf(nfx))(\lambda g.\lambda y.gy)(\lambda h.\lambda z.hz)$$

One plus one is two!

$$\begin{aligned}\text{PLUS } 1\ 1 &= (\lambda m.\lambda n.\lambda f.\lambda x.mf(nfx))(\lambda g.\lambda y.gy)(\lambda h.\lambda z.hz) \\ &\to_\beta (\lambda n.\lambda f.\lambda x.(\lambda g.\lambda y.gy)f(nfx))(\lambda h.\lambda z.hz)\end{aligned}$$

One plus one is two!

$$\begin{aligned}
\text{PLUS } 1\ 1 &= (\lambda m.\lambda n.\lambda f.\lambda x.mf(nfx))(\lambda g.\lambda y.gy)(\lambda h.\lambda z.hz) \\
&\rightarrow_\beta (\lambda n.\lambda f.\lambda x.(\lambda g.\lambda y.gy)f(nfx))(\lambda h.\lambda z.hz) \\
&\rightarrow_\beta \lambda f.\lambda x.(\lambda g.\lambda y.gy)f((\lambda h.\lambda z.hz)fx)
\end{aligned}$$

One plus one is two!

$$\begin{aligned}
\text{PLUS } 1\ 1 &= (\lambda m.\lambda n.\lambda f.\lambda x.mf(nfx))(\lambda g.\lambda y.gy)(\lambda h.\lambda z.hz) \\
&\rightarrow_\beta (\lambda n.\lambda f.\lambda x.(\lambda g.\lambda y.gy)f(nfx))(\lambda h.\lambda z.hz) \\
&\rightarrow_\beta \lambda f.\lambda x.(\lambda g.\lambda y.gy)f((\lambda h.\lambda z.hz)fx) \\
&\rightarrow_\beta \lambda f.\lambda x.(\lambda y.fy)((\lambda h.\lambda z.hz)fx)
\end{aligned}$$

One plus one is two!

$$
\begin{aligned}
\text{PLUS 1 1} &= (\lambda m.\lambda n.\lambda f.\lambda x.mf(nfx))(\lambda g.\lambda y.gy)(\lambda h.\lambda z.hz) \\
&\to_\beta (\lambda n.\lambda f.\lambda x.(\lambda g.\lambda y.gy)f(nfx))(\lambda h.\lambda z.hz) \\
&\to_\beta \lambda f.\lambda x.(\lambda g.\lambda y.gy)f((\lambda h.\lambda z.hz)fx) \\
&\to_\beta \lambda f.\lambda x.(\lambda y.fy)((\lambda h.\lambda z.hz)fx) \\
&\to_\beta \lambda f.\lambda x.f((\lambda h.\lambda z.hz)fx)
\end{aligned}
$$

One plus one is two!

$$
\begin{aligned}
\text{PLUS } 1\ 1 &= (\lambda m.\lambda n.\lambda f.\lambda x.mf(nfx))(\lambda g.\lambda y.gy)(\lambda h.\lambda z.hz) \\
&\to_\beta (\lambda n.\lambda f.\lambda x.(\lambda g.\lambda y.gy)f(nfx))(\lambda h.\lambda z.hz) \\
&\to_\beta \lambda f.\lambda x.(\lambda g.\lambda y.gy)f((\lambda h.\lambda z.hz)fx) \\
&\to_\beta \lambda f.\lambda x.(\lambda y.fy)((\lambda h.\lambda z.hz)fx) \\
&\to_\beta \lambda f.\lambda x.f((\lambda h.\lambda z.hz)fx) \\
&\to_\beta \lambda f.\lambda x.f((\lambda z.fz)x)
\end{aligned}
$$

One plus one is two!

$$
\begin{aligned}
\text{PLUS } 1\ 1 &= (\lambda m.\lambda n.\lambda f.\lambda x.mf(nfx))(\lambda g.\lambda y.gy)(\lambda h.\lambda z.hz) \\
&\to_\beta (\lambda n.\lambda f.\lambda x.(\lambda g.\lambda y.gy)f(nfx))(\lambda h.\lambda z.hz) \\
&\to_\beta \lambda f.\lambda x.(\lambda g.\lambda y.gy)f((\lambda h.\lambda z.hz)fx) \\
&\to_\beta \lambda f.\lambda x.(\lambda y.fy)((\lambda h.\lambda z.hz)fx) \\
&\to_\beta \lambda f.\lambda x.f((\lambda h.\lambda z.hz)fx) \\
&\to_\beta \lambda f.\lambda x.f((\lambda z.fz)x) \\
&\to_\beta \lambda f.\lambda x.f(fx)
\end{aligned}
$$

One plus one is two!

$$
\begin{aligned}
\text{PLUS } 1\ 1 &= (\lambda m.\lambda n.\lambda f.\lambda x.mf(nfx))(\lambda g.\lambda y.gy)(\lambda h.\lambda z.hz) \\
&\rightarrow_\beta (\lambda n.\lambda f.\lambda x.(\lambda g.\lambda y.gy)f(nfx))(\lambda h.\lambda z.hz) \\
&\rightarrow_\beta \lambda f.\lambda x.(\lambda g.\lambda y.gy)f((\lambda h.\lambda z.hz)fx) \\
&\rightarrow_\beta \lambda f.\lambda x.(\lambda y.fy)((\lambda h.\lambda z.hz)fx) \\
&\rightarrow_\beta \lambda f.\lambda x.f((\lambda h.\lambda z.hz)fx) \\
&\rightarrow_\beta \lambda f.\lambda x.f((\lambda z.fz)x) \\
&\rightarrow_\beta \lambda f.\lambda x.f(fx) \\
&= 2
\end{aligned}
$$

Factorial!

$$7! \quad = \quad 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$$

Factorial!

$$7! = 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$$

$$\text{fact } n = \begin{cases} 1 & n = 0 \\ n \cdot \text{fact}(n-1) & \text{otherwise} \end{cases}$$

Factorial!

$$7! = 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$$

$$\text{fact } n = \begin{cases} 1 & n = 0 \\ n \cdot \text{fact}(n-1) & \text{otherwise} \end{cases}$$

$$\text{F } f \ n = \text{ite } (\text{isZero } n) \text{ one } (\text{mul } n \ (f(\text{pred } n)))$$

Factorial!

$$7! = 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$$

$$\text{fact } n = \begin{cases} 1 & n = 0 \\ n \cdot \text{fact}(n-1) & \text{otherwise} \end{cases}$$

$$\begin{aligned} \text{F } f \; n &= \text{ite (isZero } n\text{) one (mul } n \; (f(\text{pred } n))) \\ \text{fact} &= YF \end{aligned}$$

Factorial!

$$7! = 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$$

$$\text{fact } n = \begin{cases} 1 & n = 0 \\ n \cdot \text{fact}(n-1) & \text{otherwise} \end{cases}$$

$$\begin{aligned} \text{F } f \ n &= \text{ite } (\text{isZero } n) \text{ one } (\text{mul } n \ (f (\text{pred } n))) \\ \text{fact} &= YF \\ \text{fact } n &= YFn \end{aligned}$$

Factorial!

$$7! = 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$$

$$\text{fact } n = \begin{cases} 1 & n = 0 \\ n \cdot \text{fact}(n-1) & \text{otherwise} \end{cases}$$

$$\begin{aligned} \text{F } f \ n &= \text{ite (isZero } n\text{) one (mul } n\ (f(\text{pred } n))) \\ \text{fact} &= YF \\ \text{fact } n &= YFn \\ &\sim_\beta F(YF)n \end{aligned}$$

Factorial!

$$7! = 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$$

$$\text{fact } n = \begin{cases} 1 & n = 0 \\ n \cdot \text{fact}(n-1) & \text{otherwise} \end{cases}$$

$$\begin{aligned}
\text{F } f \ n &= \text{ite (isZero } n\text{) one (mul } n \ (f(\text{pred } n))) \\
\text{fact} &= YF \\
\text{fact } n &= YFn \\
&\sim_\beta F(YF)n \\
&= \text{ite (isZero } n\text{) one (mul } n \ (YF(\text{pred } n)))
\end{aligned}$$

Factorial!

$$7! = 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$$

$$\text{fact } n = \begin{cases} 1 & n = 0 \\ n \cdot \text{fact}(n-1) & \text{otherwise} \end{cases}$$

$$\begin{aligned}
\text{F } f \ n &= \text{ite (isZero } n\text{) one (mul } n \ (f(\text{pred } n))) \\
\text{fact} &= YF \\
\text{fact } n &= YFn \\
&\sim_\beta F(YF)n \\
&= \text{ite (isZero } n\text{) one (mul } n \ (YF(\text{pred } n))) \\
&= \text{ite (isZero } n\text{) one (mul } n \ (\mathit{fact}(\text{pred } n)))
\end{aligned}$$

Logic in the Lambda Calculus

- true $:= \lambda x.\lambda y.x$
- false $:= \lambda x.\lambda y.y$
- and $:= \lambda p.\lambda q.pqp$
- or $:= \lambda p.\lambda q.ppq$
- not $:= \lambda p.p$ false true
- ite $:= \lambda p.\lambda a.\lambda b.pab$

Logic in the Lambda Calculus

- true := $\lambda x.\lambda y.x$
- false := $\lambda x.\lambda y.y$
- and := $\lambda p.\lambda q.pqp$
- or := $\lambda p.\lambda q.ppq$
- not := $\lambda p.p$ false true
- ite := $\lambda p.\lambda a.\lambda b.pab$

Example:

$$\text{and true false} \quad = \quad (\lambda p.\lambda q.pqp)(\lambda x.\lambda y.x)(\lambda w.\lambda z.z)$$

Logic in the Lambda Calculus

- true $:= \lambda x.\lambda y.x$
- false $:= \lambda x.\lambda y.y$
- and $:= \lambda p.\lambda q.pqp$
- or $:= \lambda p.\lambda q.ppq$
- not $:= \lambda p.p$ false true
- ite $:= \lambda p.\lambda a.\lambda b.pab$

Example:

$$\begin{array}{rcl} \text{and true false} & = & (\lambda p.\lambda q.pqp)(\lambda x.\lambda y.x)(\lambda w.\lambda z.z) \\ & \rightarrow_\beta & (\lambda q.(\lambda x.\lambda y.x)q(\lambda x.\lambda y.x))(\lambda w.\lambda z.z) \end{array}$$

Logic in the Lambda Calculus

- true := $\lambda x.\lambda y.x$
- false := $\lambda x.\lambda y.y$
- and := $\lambda p.\lambda q.pqp$
- or := $\lambda p.\lambda q.ppq$
- not := $\lambda p.p$ false true
- ite := $\lambda p.\lambda a.\lambda b.pab$

Example:

$$\begin{aligned}\text{and true false} &= (\lambda p.\lambda q.pqp)(\lambda x.\lambda y.x)(\lambda w.\lambda z.z)\\ &\rightarrow_\beta (\lambda q.(\lambda x.\lambda y.x)q(\lambda x.\lambda y.x))(\lambda w.\lambda z.z)\\ &\rightarrow_\beta (\lambda x.\lambda y.x)(\lambda w.\lambda z.z)(\lambda x.\lambda y.x)\end{aligned}$$

Logic in the Lambda Calculus

- true $:= \lambda x.\lambda y.x$
- false $:= \lambda x.\lambda y.y$
- and $:= \lambda p.\lambda q.pqp$
- or $:= \lambda p.\lambda q.ppq$
- not $:= \lambda p.p$ false true
- ite $:= \lambda p.\lambda a.\lambda b.pab$

Example:

$$\begin{aligned}
\text{and true false} &= (\lambda p.\lambda q.pqp)(\lambda x.\lambda y.x)(\lambda w.\lambda z.z) \\
&\to_\beta (\lambda q.(\lambda x.\lambda y.x)q(\lambda x.\lambda y.x))(\lambda w.\lambda z.z) \\
&\to_\beta (\lambda x.\lambda y.x)(\lambda w.\lambda z.z)(\lambda x.\lambda y.x) \\
&\to_\beta (\lambda y.\lambda w.\lambda z.z)(\lambda x.\lambda y.x)
\end{aligned}$$

Logic in the Lambda Calculus

- true $:= \lambda x.\lambda y.x$
- false $:= \lambda x.\lambda y.y$
- and $:= \lambda p.\lambda q.pqp$
- or $:= \lambda p.\lambda q.ppq$
- not $:= \lambda p.p$ false true
- ite $:= \lambda p.\lambda a.\lambda b.pab$

Example:

$$\begin{array}{rcl} \text{and true false} & = & (\lambda p.\lambda q.pqp)(\lambda x.\lambda y.x)(\lambda w.\lambda z.z) \\ & \rightarrow_\beta & (\lambda q.(\lambda x.\lambda y.x)q(\lambda x.\lambda y.x))(\lambda w.\lambda z.z) \\ & \rightarrow_\beta & (\lambda x.\lambda y.x)(\lambda w.\lambda z.z)(\lambda x.\lambda y.x) \\ & \rightarrow_\beta & (\lambda y.\lambda w.\lambda z.z)(\lambda x.\lambda y.x) \\ & \rightarrow_\beta & \lambda w.\lambda z.z \end{array}$$

- true $:= \lambda x.\lambda y.x$
- false $:= \lambda x.\lambda y.y$
- and $:= \lambda p.\lambda q.pqp$
- or $:= \lambda p.\lambda q.ppq$
- not $:= \lambda p.p$ false true
- ite $:= \lambda p.\lambda a.\lambda b.pab$

Example:

$$
\begin{array}{rcl}
\text{and true false} & = & (\lambda p.\lambda q.pqp)(\lambda x.\lambda y.x)(\lambda w.\lambda z.z) \\
& \to_\beta & (\lambda q.(\lambda x.\lambda y.x)q(\lambda x.\lambda y.x))(\lambda w.\lambda z.z) \\
& \to_\beta & (\lambda x.\lambda y.x)(\lambda w.\lambda z.z)(\lambda x.\lambda y.x) \\
& \to_\beta & (\lambda y.\lambda w.\lambda z.z)(\lambda x.\lambda y.x) \\
& \to_\beta & \lambda w.\lambda z.z \\
& = & \text{false}
\end{array}
$$

Normal Form

An lambda expression without a redex is called a **normal form**.

- We know they do not always exist (such as with Ω).
- Are they unique?
- Does it matter how you chose each redex?

Church-Rosser

Theorem

Given terms X, Y_1, and Y_2 such that:

$$\begin{array}{ccc} X & \twoheadrightarrow & Y_1 \\ \big\downarrow\!\!\downarrow & & \\ Y_2 & & \end{array}$$

Church-Rosser

Theorem

Given terms X, Y_1, and Y_2 such that:

$$\begin{array}{ccc} X & \twoheadrightarrow & Y_1 \\ \downarrow\!\!\downarrow & & \downarrow\!\!\downarrow \\ Y_2 & \twoheadrightarrow & Z \end{array}$$

There exists a Z as above.

Church-Rosser

Theorem

Given terms X, Y_1, and Y_2 such that:

$$\begin{array}{ccc} X & \twoheadrightarrow & Y_1 \\ \downarrow\!\!\downarrow & & \downarrow\!\!\downarrow \\ Y_2 & \twoheadrightarrow & Z \end{array}$$

There exists a Z as above.

Corollary

Normal forms are unique when they exist.

Reduction Strategies

- Call by Value - reduce the leftmost innermost redex first.
- Call by Name - reduce the leftmost outermost redex first.
- Call by Need - optimization of call by name.

Reduction Strategies

- Call by Value - reduce the leftmost innermost redex first.
- Call by Name - reduce the leftmost outermost redex first.
- Call by Need - optimization of call by name.

Theorem

Call by Name will always find the normal form if it exists.

Python Examples

https://github.com/JaredCorduan/lambda-calc-cofc/blob/master/lambda.py

Final Takeaway

The lambda calculus explains computer science in three steps:

- variables
- abstraction
- application

Final Takeaway

The lambda calculus explains computer science in three steps:

- variables
- abstraction
- application

It's the ultimate Occam's razor for computation.

Final Takeaway

The lambda calculus explains computer science in three steps:

- variables
- abstraction
- application

It's the ultimate Occam's razor for computation.

Next Steps:

- simply typed lambda calculus
- propositions as types

thank you for listening!

Primitive Recursion in the Lambda Calculus

Let $f : \mathbb{N}^{k+1} \to \mathbb{N}$ be defined by:

$$\begin{aligned} f(0, n_1, \ldots, n_k) &:= g(n_1, \ldots, n_k) \\ f(n+1, n_1, \ldots, n_k) &:= h(f(n, n_1, \ldots, n_k), n, n_1, \ldots, n_k) \end{aligned}$$

Define:

$$\begin{aligned} \langle M, N \rangle &:= \lambda x.xMN \\ \pi_1 &:= \lambda p.p(\lambda x.\lambda y.x) \\ \pi_2 &:= \lambda p.p(\lambda x.\lambda y.y) \\ \text{Init} &:= \langle 0, Gx_1 \ldots x_k \rangle \\ \text{Step} &:= \lambda p.\langle \text{SUCC}(\pi_1 p), H(\pi_2 p)(\pi_1 p)x_1 \ldots x_k \rangle \\ F &:= \lambda x.\lambda x_1. \ldots \lambda x_k.\pi_2(x \text{ Step Init}) \end{aligned}$$