Untyped Lambda Calculus

Structure and Evaluation, Currying, Church Encodings

Sven Tennie

November 18, 2018

Dream IT
https://dreamit.de

Introduction

Lambda Calculus

- Invented by Alonzo Church (1920s)
- Equally expressive to the Turing Machine(s)
- Formal Language
- Computational Model
    - Lisp (1950s)
    - ML
    - Haskell
- "Lambda Expressions" in almost every modern programming language

Why should I care?

- Simple Computational Model
    - to describe structure and behaviour (E.g. Operational Semantics, Type Systems)
    - to reason and prove

Why should I care?

- Simple Computational Model
  - to describe structure and behaviour (E.g. Operational Semantics, Type Systems)
  - to reason and prove

- Explains why things in FP are like they are
  - Pure Functions
  - Higher-Order Functions
  - Currying
  - Lazy Evaluation

Why should I care?

- Simple Computational Model
  - to describe structure and behaviour (E.g. Operational Semantics, Type Systems)
  - to reason and prove

- Explains why things in FP are like they are
  - Pure Functions
  - Higher-Order Functions
  - Currying
  - Lazy Evaluation

- Understand FP Compilers
  - Introduce FP stuff into other languages
  - Write your own compiler
  - GHC uses an enriched Lambda Calculus internally

Basics

Untyped Lambda Calculus

$$t ::= \qquad\qquad\qquad\qquad \text{Terms:}$$

x	Variable
$\lambda x.t$	Abstraction
$t\ t$	Application

Untyped Lambda Calculus

$$t ::= \qquad\qquad\qquad\qquad \text{Terms:}$$
$$x \qquad\qquad\qquad\qquad \text{Variable}$$
$$\lambda x.t \qquad\qquad\qquad \text{Abstraction}$$
$$t\ t \qquad\qquad\qquad \text{Application}$$

Example - Identity

Lambda Calculus

$$\underbrace{\underbrace{\lambda x.x}_{\text{Abstraction}} \quad \underbrace{y}_{\text{Variable}}}_{\text{Application}} \rightarrow y$$

Untyped Lambda Calculus

$$t ::= \quad\quad\quad\quad\quad\quad \text{Terms:}$$
$$x \quad\quad\quad\quad\quad\quad \text{Variable}$$
$$\lambda x.t \quad\quad\quad\quad\quad\quad \text{Abstraction}$$
$$t\ t \quad\quad\quad\quad\quad\quad \text{Application}$$

Example - Identity

Lambda Calculus

$$\underbrace{\underbrace{\lambda x.x}_{\text{Abstraction}}\ \underbrace{y}_{\text{Variable}}}_{\text{Application}} \rightarrow y$$

Javascript

$$\underbrace{(\underbrace{function\ (x)\{return\ x;\}}_{\text{Abstraction}})\ (\ \underbrace{y}_{\text{Variable}}\ )}_{\text{Application}}$$

Abstractions
Think: Function Definitions

$(\underline{\lambda x.\lambda y.x\ y})\ a\ b$

Abstractions
Think: Function Definitions

$(\underline{\lambda x.\lambda y.x\ y})\ a\ b$

Variables
Think: Parameters

$(\lambda x.\lambda y.\underline{x}\ \underline{y})\ \underline{a}\ \underline{b}$

Example - $(\lambda x.\lambda y.x\ y)\ a\ b$

Abstractions
Think: Function Definitions

$(\lambda x.\underline{\lambda y.x\ y})\ a\ b$

Variables
Think: Parameters

$(\lambda x.\lambda y.\underline{x}\ \underline{y})\ \underline{a}\ \underline{b}$

Applications
Think: Function Calls

$\underline{(\lambda x.\lambda y.x\ y)\ \underline{a\ b}}$

$(\lambda x. \quad \lambda y.x \quad y) \quad a \quad b$

Example - $(\lambda x.\lambda y.x\ y)\ a\ b$

$(\lambda x.\qquad \lambda y.x \qquad y)\qquad a\qquad b\qquad$ Substitute $x \mapsto a$

$(\lambda x. \qquad \lambda y.x \qquad y) \qquad a \qquad b \qquad$ Substitute $x \mapsto a$

$$
\begin{array}{lllll}
(\lambda x. & \lambda y.x & y) & a & b & \text{Substitute } x \mapsto a \\
\rightarrow & (\lambda y.a & y) & & b &
\end{array}
$$

Example - $(\lambda x. \lambda y. x \; y) \; a \; b$

$(\lambda x.$	$\lambda y.x$	$y)$	a	b	Substitute $x \mapsto a$
$\rightarrow$	$(\lambda y.a$	$y)$		b	Substitute $y \mapsto b$

Example - $(\lambda x.\lambda y.x\ y)\ a\ b$

$(\lambda x.$	$\lambda y.x$	$y)$	a	b	Substitute $x \mapsto a$
$\rightarrow$	$(\lambda y.a$	$y)$		b	Substitute $y \mapsto b$

$(\lambda x.$	$\lambda y.x$	$y)$	a	b	Substitute $x \mapsto a$
$\rightarrow$	$(\lambda y.a$	$y)$		b	Substitute $y \mapsto b$
$\rightarrow$	a	b			

Notational Conventions

- We use parentheses to clearify what's meant
- Applications associate to the left

$$s\ t\ u \equiv (s\ t)\ u$$

- Abstractions expand as much to the right as possible

$$\lambda x.\lambda y.x\ y\ x \equiv \lambda x.(\lambda y.(x\ y\ x))$$

Scope

$$\lambda x.\lambda y.x\ y\ z$$

Bound and **Free**

λy y is *bound*, x and z are *free*

λx x and y are *bound*, z is *free*

λx, λy *binders*

Scope

$$\lambda x . \lambda y . x \ y \ z$$

Bound and Free

> λy y is *bound*, x and z are *free*
>
> λx x and y are *bound*, z is *free*
>
> λx, λy *binders*

A term with no free variables is *closed*

- A *combinator*
- $id \equiv \lambda x.x$
- Y, S, K, I …

Higher Order Functions

- Functions that take or return functions
    - Are there "by definition"

$$\underbrace{\underbrace{\lambda x.x}_{\text{Abstraction}} \quad \underbrace{\lambda y.y}_{\text{Abstraction}}}_{\text{Application}} \quad \rightarrow \quad \underbrace{\lambda y.y}_{\text{Abstraction}}$$

Currying

Idea

- Take a function with n arguments
- Create a function that takes one argument and returns a function with $n - 1$ arguments

Currying

Idea

- Take a function with n arguments
- Create a function that takes one argument and returns a function with $n - 1$ arguments

Example

- (+1) Section in Haskell
- $(\lambda x.\lambda y. + x\ y)\ 1 \rightarrow \lambda y. +\ 1\ y$

Currying

Idea

- Take a function with n arguments
- Create a function that takes one argument and returns a function with $n-1$ arguments

Example

- (+1) Section in Haskell
- $(\lambda x.\lambda y. + x\ y)\ 1 \rightarrow \lambda y. +\ 1\ y$

- Partial Function Application is there "by definition"
  - You can use this stunt to "curry" in every language that supports "Lambda Expressions"

Alpha Conversion

$$\lambda x.x \rightarrow_\alpha \lambda y.y$$

Reductions and Conversions

Alpha Conversion

$$\lambda x.x \to_\alpha \lambda y.y$$

Beta Reduction

$$(\lambda x.x)\ y \to_\beta y$$

Reductions and Conversions

Alpha Conversion

$$\lambda x.x \to_\alpha \lambda y.y$$

Beta Reduction

$$(\lambda x.x)\ y \to_\beta y$$

Eta Conversion
Iff (if and only if) x is not free in f:

$$\lambda x.f\ x \to_\eta f$$

$$(\lambda x.\underbrace{(\lambda y.y)}_{f}\ x)\ a \to_\eta \underbrace{(\lambda y.y)}_{f}\ a$$

Reductions and Conversions

Alpha Conversion

$$\lambda x.x \rightarrow_\alpha \lambda y.y$$

Beta Reduction

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

Eta Conversion
Iff (if and only if) x is not free in f:

$$\lambda x.f\ x \rightarrow_\eta f$$

$$(\lambda x.\underbrace{(\lambda y.y)}_{f}\ x)\ a \rightarrow_\eta \underbrace{(\lambda y.y)}_{f}\ a$$

If x is free in f, η conversion not possible:

$$\lambda x.\underbrace{(\lambda y.y\ \overset{\overset{\text{Bound}}{\downarrow}}{x}\ )}_{f}\ x \not\rightarrow_\eta (\lambda y.y\ \overset{\overset{\text{Free?!}}{\downarrow}}{x}\ )$$

Remarks

- Everything (Term) is an Expression
    - No statements
- No "destructive" Assignments
    - The reason why FP Languages promote pure functions
    - But you could invent a built-in function to manipulate "state"…

Evaluation

Operational Semantics

- We learned how to write down and talk about Lambda Calculus Terms
- How to evaluate them?
- Different Strategies
  - Interesting outcomes

Full Beta-Reduction

- RedEx
  - **Red**ucible **Ex**pression
  - Always an Application

$$(\lambda x.x) \; ((\lambda y.y) \; (\lambda z. \underbrace{(\lambda a.a) \; z}_{RedEx}))$$

$$\underbrace{}_{RedEx}$$

$$\underbrace{}_{RedEx}$$

Full Beta-Reduction

- RedEx
    - **Red**ucible **Ex**pression
    - Always an Application

$$\underbrace{(\lambda x.x)\ \underbrace{((\lambda y.y)\ (\lambda z.\underbrace{(\lambda a.a)\ z}_{RedEx}))}_{RedEx}}_{RedEx}$$

Full Beta-Reduction

- Any RedEx, Any Time
- Like in Arithmetics
- Too vague for programming...

$$(\lambda x.x)\ ((\lambda y.y)\ (\lambda z.(\lambda a.a)\ z))$$

Normal Order Reduction

- Left-most, Outer-most RedEx

$$(\lambda x.x)\ ((\lambda y.y)\ (\lambda z.(\lambda a.a)\ z))$$

Normal Order Reduction

- Left-most, Outer-most RedEx

$$(\lambda x.x)\ ((\lambda y.y)\ (\lambda z.(\lambda a.a)\ z))$$
$$\rightarrow (\lambda y.y)\ (\lambda z.(\lambda a.a)\ z)$$

Normal Order Reduction

- Left-most, Outer-most RedEx

Normal Order Reduction

$$(\lambda x.x)\ ((\lambda y.y)\ (\lambda z.(\lambda a.a)\ z))$$
$$\rightarrow (\lambda y.y)\ (\lambda z.(\lambda a.a)\ z)$$

Normal Order Reduction

- Left-most, Outer-most RedEx

$$(\lambda x.x)\ ((\lambda y.y)\ (\lambda z.(\lambda a.a)\ z))$$
$$\rightarrow (\lambda y.y)\ (\lambda z.(\lambda a.a)\ z)$$
$$\rightarrow \lambda z.(\lambda a.a)\ z$$

Normal Order Reduction

- Left-most, Outer-most RedEx

$$(\lambda x.x)\ ((\lambda y.y)\ (\lambda z.(\lambda a.a)\ z))$$
$$\rightarrow (\lambda y.y)\ (\lambda z.(\lambda a.a)\ z)$$
$$\rightarrow \lambda z.(\lambda a.a)\ z$$

Normal Order Reduction

- Left-most, Outer-most RedEx

$$(\lambda x.x)\ ((\lambda y.y)\ (\lambda z.(\lambda a.a)\ z))$$
$$\rightarrow (\lambda y.y)\ (\lambda z.(\lambda a.a)\ z)$$
$$\rightarrow \lambda z.(\lambda a.a)\ z$$
$$\rightarrow \lambda z.z$$

Normal Order Reduction

- Left-most, Outer-most RedEx

Call-by-Name

$$(\lambda x.x)\ ((\lambda y.y)\ (\lambda z.(\lambda a.a)\ z))$$

Call-by-Name

- like Normal Order Reduction, but no reductions inside Abstractions
    - Abstractions are values
- lazy, non-strict
    - Parameters are not evaluated before they are used
- Optimization: Save results → *Call-by-Need*

$$(\lambda x.x)\ ((\lambda y.y)\ (\lambda z.(\lambda a.a)\ z))$$

Call-by-Name

- like Normal Order Reduction, but no reductions inside Abstractions
    - Abstractions are values
- lazy, non-strict
    - Parameters are not evaluated before they are used
- Optimization: Save results → *Call-by-Need*

$$(\lambda x.x)\ ((\lambda y.y)\ (\lambda z.(\lambda a.a)\ z))$$
$$\rightarrow (\lambda y.y)\ (\lambda z.(\lambda a.a)\ z)$$

Call-by-Name

- like Normal Order Reduction, but no reductions inside Abstractions
    - Abstractions are values
- lazy, non-strict
    - Parameters are not evaluated before they are used
- Optimization: Save results $\rightarrow$ *Call-by-Need*

15

$$(\lambda x.x)\ ((\lambda y.y)\ (\lambda z.(\lambda a.a)\ z))$$
$$\rightarrow (\lambda y.y)\ (\lambda z.(\lambda a.a)\ z)$$

Call-by-Name

- like Normal Order Reduction, but no reductions inside Abstractions
  - Abstractions are values
- lazy, non-strict
  - Parameters are not evaluated before they are used
- Optimization: Save results → *Call-by-Need*

$$(\lambda x.x)\ ((\lambda y.y)\ (\lambda z.(\lambda a.a)\ z))$$
$$\rightarrow (\lambda y.y)\ (\lambda z.(\lambda a.a)\ z)$$
$$\rightarrow \lambda z.(\lambda a.a)\ z$$

Call-by-Name

- like Normal Order Reduction, but no reductions inside Abstractions
    - Abstractions are values
- lazy, non-strict
    - Parameters are not evaluated before they are used
- Optimization: Save results $\rightarrow$ *Call-by-Need*

$$(\lambda x.x)\ ((\lambda y.y)\ (\lambda z.(\lambda a.a)\ z))$$
$$\rightarrow (\lambda y.y)\ (\lambda z.(\lambda a.a)\ z)$$
$$\rightarrow \lambda z.(\lambda a.a)\ z$$
$$\nrightarrow$$

Call-by-Name

- like Normal Order Reduction, but no reductions inside Abstractions
    - Abstractions are values
- lazy, non-strict
    - Parameters are not evaluated before they are used
- Optimization: Save results $\rightarrow$ *Call-by-Need*

Call-by-Value

$$(\lambda x.x)\ ((\lambda y.y)\ (\lambda z.(\lambda a.a)\ z))$$

Call-by-Value

- Outer-most, only if right-hand side was reduced to a value
- No reductions inside Abstractions
  - Abstractions are values
- eager, strict
  - Parameters are evaluated before they are used

Call-by-Value

$$(\lambda x.x) \; ((\lambda y.y) \; (\lambda z.(\lambda a.a) \; z))$$

Call-by-Value

- Outer-most, only if right-hand side was reduced to a value
- No reductions inside Abstractions
    - Abstractions are values
- eager, strict
    - Parameters are evaluated before they are used

$$(\lambda x.x) \; ((\lambda y.y) \; (\lambda z.(\lambda a.a) \; z))$$
$$\rightarrow (\lambda x.x) \; (\lambda z.(\lambda a.a) \; z)$$

Call-by-Value

- Outer-most, only if right-hand side was reduced to a value
- No reductions inside Abstractions
  - Abstractions are values
- eager, strict
  - Parameters are evaluated before they are used

$$(\lambda x.x)\ ((\lambda y.y)\ (\lambda z.(\lambda a.a)\ z))$$
$$\rightarrow (\lambda x.x)\ (\lambda z.(\lambda a.a)\ z)$$

Call-by-Value

- Outer-most, only if right-hand side was reduced to a value
- No reductions inside Abstractions
  - Abstractions are values
- eager, strict
  - Parameters are evaluated before they are used

Call-by-Value

$$(\lambda x.x)\ ((\lambda y.y)\ (\lambda z.(\lambda a.a)\ z))$$
$$\rightarrow (\lambda x.x)\ (\lambda z.(\lambda a.a)\ z)$$
$$\rightarrow \lambda z.(\lambda a.a)\ z$$

Call-by-Value

- Outer-most, only if right-hand side was reduced to a value
- No reductions inside Abstractions
  - Abstractions are values
- eager, strict
  - Parameters are evaluated before they are used

$$(\lambda x.x)\ ((\lambda y.y)\ (\lambda z.(\lambda a.a)\ z))$$
$$\rightarrow (\lambda x.x)\ (\lambda z.(\lambda a.a)\ z)$$
$$\rightarrow \lambda z.(\lambda a.a)\ z$$
$$\not\rightarrow$$

Call-by-Value

- Outer-most, only if right-hand side was reduced to a value
- No reductions inside Abstractions
    - Abstractions are values
- eager, strict
    - Parameters are evaluated before they are used

Church Encodings

Church Encodings

- Encode Data into the Lambda Calculus
- To simplify our formulas, let's say that we have declarations

$$id \equiv \lambda x.x$$
$$id\ y \rightarrow y$$

Booleans

$$true \equiv \qquad \lambda t.\lambda f.t$$
$$false \equiv \qquad \lambda t.\lambda f.f$$

Booleans

$$true \equiv \lambda t.\lambda f.t$$
$$false \equiv \lambda t.\lambda f.f$$

$$test \equiv \lambda c.\lambda t.\lambda f.c\ t\ f$$

Booleans

$test\ true\ a\ b$

$$true \equiv \quad \lambda t.\lambda f.t$$

$$false \equiv \quad \lambda t.\lambda f.f$$

$$test \equiv \quad \lambda c.\lambda t.\lambda f.c\ t\ f$$

Booleans

$$test\ true\ a\ b$$

$$true \equiv \qquad \lambda t.\lambda f.t$$
$$false \equiv \qquad \lambda t.\lambda f.f$$

$$test \equiv \quad \lambda c.\lambda t.\lambda f.c\ t\ f$$

Booleans

$$test\ true\ a\ b$$
$$\equiv (\lambda c.\lambda t.\lambda f.c\ t\ f)\ true\ a\ b$$

$$true \equiv \qquad \lambda t.\lambda f.t$$
$$false \equiv \qquad \lambda t.\lambda f.f$$

$$test \equiv \quad \lambda c.\lambda t.\lambda f.c\ t\ f$$

$$true \equiv \qquad \lambda t.\lambda f.t$$

$$false \equiv \qquad \lambda t.\lambda f.f$$

$$test \equiv \quad \lambda c.\lambda t.\lambda f.c\ t\ f$$

$$test\ true\ a\ b$$

$$\equiv (\lambda c.\lambda t.\lambda f.c\ t\ f)\ true\ a\ b$$

$$test\ true\ a\ b$$

$$\equiv (\lambda c.\lambda t.\lambda f.c\ t\ f)\ true\ a\ b$$

$$\rightarrow (\lambda t.\lambda f.true\ t\ f)\ a\ b$$

$$true \equiv \qquad \lambda t.\lambda f.t$$

$$false \equiv \qquad \lambda t.\lambda f.f$$

$$test \equiv \quad \lambda c.\lambda t.\lambda f.c\ t\ f$$

Booleans

$$true \equiv \qquad \lambda t.\lambda f.t$$

$$false \equiv \qquad \lambda t.\lambda f.f$$

$$test \equiv \quad \lambda c.\lambda t.\lambda f.c\ t\ f$$

$$test\ true\ a\ b$$

$$\equiv (\lambda c.\lambda t.\lambda f.c\ t\ f)\ true\ a\ b$$

$$\rightarrow (\lambda t.\lambda f.true\ t\ f)\ a\ b$$

$$test\ true\ a\ b$$

$$\equiv (\lambda c.\lambda t.\lambda f.c\ t\ f)\ true\ a\ b$$

$$true \equiv \qquad \lambda t.\lambda f.t$$

$$\rightarrow (\lambda t.\lambda f.true\ t\ f)\ a\ b$$

$$false \equiv \qquad \lambda t.\lambda f.f$$

$$\rightarrow (\lambda f.true\ a\ f)\ b$$

$$test \equiv \quad \lambda c.\lambda t.\lambda f.c\ t\ f$$

$$test\ true\ a\ b$$

$$\equiv (\lambda c.\lambda t.\lambda f.c\ t\ f)\ true\ a\ b$$

$$\rightarrow (\lambda t.\lambda f.true\ t\ f)\ a\ b$$

$$\rightarrow (\lambda f.true\ a\ f)\ b$$

$$true \equiv \lambda t.\lambda f.t$$

$$false \equiv \lambda t.\lambda f.f$$

$$test \equiv \lambda c.\lambda t.\lambda f.c\ t\ f$$

$$true \equiv \qquad \lambda t.\lambda f.t$$

$$false \equiv \qquad \lambda t.\lambda f.f$$

$$test \equiv \quad \lambda c.\lambda t.\lambda f.c\ t\ f$$

$$test\ true\ a\ b$$

$$\equiv (\lambda c.\lambda t.\lambda f.c\ t\ f)\ true\ a\ b$$

$$\rightarrow (\lambda t.\lambda f.true\ t\ f)\ a\ b$$

$$\rightarrow (\lambda f.true\ a\ f)\ b$$

$$\rightarrow true\ a\ b$$

$$true \equiv \qquad \lambda t.\lambda f.t$$

$$false \equiv \qquad \lambda t.\lambda f.f$$

$$test \equiv \quad \lambda c.\lambda t.\lambda f.c\ t\ f$$

$test\ true\ a\ b$

$\equiv (\lambda c.\lambda t.\lambda f.c\ t\ f)\ true\ a\ b$

$\rightarrow (\lambda t.\lambda f.true\ t\ f)\ a\ b$

$\rightarrow (\lambda f.true\ a\ f)\ b$

$\rightarrow true\ a\ b$

$$test\ true\ a\ b$$

$$\equiv (\lambda c.\lambda t.\lambda f.c\ t\ f)\ true\ a\ b$$

$$\rightarrow (\lambda t.\lambda f.true\ t\ f)\ a\ b$$

$$\rightarrow (\lambda f.true\ a\ f)\ b$$

$$\rightarrow true\ a\ b$$

$$\equiv (\lambda t.\lambda f.t)\ a\ b$$

$$true \equiv \qquad \lambda t.\lambda f.t$$

$$false \equiv \qquad \lambda t.\lambda f.f$$

$$test \equiv \quad \lambda c.\lambda t.\lambda f.c\ t\ f$$

Booleans

$$true \equiv \lambda t.\lambda f.t$$

$$false \equiv \lambda t.\lambda f.f$$

$$test \equiv \lambda c.\lambda t.\lambda f.c\ t\ f$$

$test\ true\ a\ b$

$\equiv (\lambda c.\lambda t.\lambda f.c\ t\ f)\ true\ a\ b$

$\rightarrow (\lambda t.\lambda f.true\ t\ f)\ a\ b$

$\rightarrow (\lambda f.true\ a\ f)\ b$

$\rightarrow true\ a\ b$

$\equiv (\lambda t.\lambda f.t)\ a\ b$

$$true \equiv \quad \lambda t.\lambda f.t$$

$$false \equiv \quad \lambda t.\lambda f.f$$

$$test \equiv \quad \lambda c.\lambda t.\lambda f.c \; t \; f$$

$test \; true \; a \; b$

$\equiv (\lambda c.\lambda t.\lambda f.c \; t \; f) \; true \; a \; b$

$\rightarrow (\lambda t.\lambda f.true \; t \; f) \; a \; b$

$\rightarrow (\lambda f.true \; a \; f) \; b$

$\rightarrow true \; a \; b$

$\equiv (\lambda t.\lambda f.t) \; a \; b$

$\rightarrow (\lambda f.a) \; b$

Booleans

$$true \equiv \qquad \lambda t.\lambda f.t$$

$$false \equiv \qquad \lambda t.\lambda f.f$$

$$test \equiv \quad \lambda c.\lambda t.\lambda f.c\ t\ f$$

$test\ true\ a\ b$

$\equiv (\lambda c.\lambda t.\lambda f.c\ t\ f)\ true\ a\ b$

$\rightarrow (\lambda t.\lambda f.true\ t\ f)\ a\ b$

$\rightarrow (\lambda f.true\ a\ f)\ b$

$\rightarrow true\ a\ b$

$\equiv (\lambda t.\lambda f.t)\ a\ b$

$\rightarrow (\lambda f.a)\ b$

$$true \equiv \quad \lambda t.\lambda f.t$$

$$false \equiv \quad \lambda t.\lambda f.f$$

$$test \equiv \quad \lambda c.\lambda t.\lambda f.c \; t \; f$$

$test \; true \; a \; b$

$\equiv (\lambda c.\lambda t.\lambda f.c \; t \; f) \; true \; a \; b$

$\rightarrow (\lambda t.\lambda f.true \; t \; f) \; a \; b$

$\rightarrow (\lambda f.true \; a \; f) \; b$

$\rightarrow true \; a \; b$

$\equiv (\lambda t.\lambda f.t) \; a \; b$

$\rightarrow (\lambda f.a) \; b$

$\rightarrow a$

And

$$true \equiv \quad \lambda t.\lambda f.t$$
$$false \equiv \quad \lambda t.\lambda f.f$$

And

$$true \equiv \lambda t.\lambda f.t$$
$$false \equiv \lambda t.\lambda f.f$$

$$and \equiv \lambda p.\lambda q.p\ q\ p$$

And

$$\textit{and true false}$$

$$\textit{true} \equiv \qquad \lambda t.\lambda f.t$$

$$\textit{false} \equiv \qquad \lambda t.\lambda f.f$$

$$\textit{and} \equiv \quad \lambda p.\lambda q.p\ q\ p$$

And

$$and\ true\ false$$

$$true \equiv \qquad \lambda t.\lambda f.t$$
$$false \equiv \qquad \lambda t.\lambda f.f$$

$$and \equiv \quad \lambda p.\lambda q.p\ q\ p$$

$$true \equiv \qquad \lambda t.\lambda f.t$$
$$false \equiv \qquad \lambda t.\lambda f.f$$

$$and \equiv \quad \lambda p.\lambda q.p\ q\ p$$

and *true* *false*

$$\equiv (\lambda p.\lambda q.p\ q\ p)\ true\ false$$

$$true \equiv \qquad \lambda t.\lambda f.t$$

$$false \equiv \qquad \lambda t.\lambda f.f$$

$$and \equiv \quad \lambda p.\lambda q.p \; q \; p$$

and true false

$$\equiv (\lambda p.\lambda q.p \; q \; p) \; true \; false$$

$true \equiv \qquad \lambda t.\lambda f.t$

$false \equiv \qquad \lambda t.\lambda f.f$

$and \equiv \quad \lambda p.\lambda q.p\ q\ p$

$and\ true\ false$

$\equiv (\lambda p.\lambda q.p\ q\ p)\ true\ false$

$\rightarrow (\lambda q.true\ q\ true)\ false$

And

$$true \equiv \qquad \lambda t.\lambda f.t$$

$$false \equiv \qquad \lambda t.\lambda f.f$$

$$and \equiv \quad \lambda p.\lambda q.p\ q\ p$$

$$and\ true\ false$$

$$\equiv (\lambda p.\lambda q.p\ q\ p)\ true\ false$$

$$\rightarrow (\lambda q.true\ q\ true)\ false$$

$true \equiv$ $\lambda t.\lambda f.t$

$false \equiv$ $\lambda t.\lambda f.f$

$and \equiv$ $\lambda p.\lambda q.p\ q\ p$

$and\ true\ false$

$\equiv (\lambda p.\lambda q.p\ q\ p)\ true\ false$

$\rightarrow (\lambda q.true\ q\ true)\ false$

$\rightarrow true\ false\ true$

And

$$true \equiv \quad \lambda t.\lambda f.t$$

$$false \equiv \quad \lambda t.\lambda f.f$$

$$and \equiv \quad \lambda p.\lambda q.p\ q\ p$$

and true false

$\equiv (\lambda p.\lambda q.p\ q\ p)$ *true false*

$\rightarrow (\lambda q.true\ q\ true)$ *false*

$\rightarrow true$ *false true*

$$true \equiv \qquad \lambda t.\lambda f.t$$

$$false \equiv \qquad \lambda t.\lambda f.f$$

$$and \equiv \quad \lambda p.\lambda q.p\ q\ p$$

$$and\ true\ false$$

$$\equiv (\lambda p.\lambda q.p\ q\ p)\ true\ false$$

$$\rightarrow (\lambda q.true\ q\ true)\ false$$

$$\rightarrow true\ false\ true$$

$$\equiv (\lambda t.\lambda f.t)\ false\ true$$

$true \equiv \qquad \lambda t.\lambda f.t$

$false \equiv \qquad \lambda t.\lambda f.f$

$and \equiv \quad \lambda p.\lambda q.p\ q\ p$

$and\ true\ false$

$\equiv (\lambda p.\lambda q.p\ q\ p)\ true\ false$

$\rightarrow (\lambda q.true\ q\ true)\ false$

$\rightarrow true\ false\ true$

$\equiv (\lambda t.\lambda f.t)\ false\ true$

$true \equiv \qquad \lambda t.\lambda f.t$

$false \equiv \qquad \lambda t.\lambda f.f$

$and \equiv \quad \lambda p.\lambda q.p\ q\ p$

$and\ true\ false$

$\equiv (\lambda p.\lambda q.p\ q\ p)\ true\ false$

$\rightarrow (\lambda q.true\ q\ true)\ false$

$\rightarrow true\ false\ true$

$\equiv (\lambda t.\lambda f.t)\ false\ true$

$\rightarrow (\lambda f.false)\ true$

$true \equiv$ $\lambda t.\lambda f.t$

$false \equiv$ $\lambda t.\lambda f.f$

$and \equiv$ $\lambda p.\lambda q.p\ q\ p$

$and\ true\ false$

$\equiv (\lambda p.\lambda q.p\ q\ p)\ true\ false$

$\rightarrow (\lambda q.true\ q\ true)\ false$

$\rightarrow true\ false\ true$

$\equiv (\lambda t.\lambda f.t)\ false\ true$

$\rightarrow (\lambda f.false)\ true$

And

$$true \equiv \qquad \lambda t.\lambda f.t$$

$$false \equiv \qquad \lambda t.\lambda f.f$$

$$and \equiv \quad \lambda p.\lambda q.p\ q\ p$$

and true false

$\equiv(\lambda p.\lambda q.p\ q\ p)$ true false

$\rightarrow(\lambda q.true\ q\ true)$ false

$\rightarrow true$ false true

$\equiv(\lambda t.\lambda f.t)$ false true

$\rightarrow(\lambda f.false)$ true

$\rightarrow false$

$$\lambda p. \lambda q. p \; p \; q$$

Pairs

$pair \equiv \quad \lambda x.\lambda y.\lambda z.z\ x\ y$

Pairs

$$pair \equiv \quad \lambda x.\lambda y.\lambda z.z\ x\ y$$
$$first \equiv \quad (\lambda p.p)\ \lambda x.\lambda y.x$$
$$second \equiv \quad (\lambda p.p)\ \lambda x.\lambda y.y$$

Pairs

$$pair_{AB} \equiv \qquad\qquad pair\ a\ b$$

$$
\begin{aligned}
pair &\equiv \lambda x.\lambda y.\lambda z.z\ x\ y \\
first &\equiv (\lambda p.p)\ \lambda x.\lambda y.x \\
second &\equiv (\lambda p.p)\ \lambda x.\lambda y.y
\end{aligned}
$$

Pairs

$$pair_{AB} \equiv \qquad\qquad pair\ a\ b$$

$$pair \equiv \quad \lambda x.\lambda y.\lambda z.z\ x\ y$$
$$first \equiv \quad (\lambda p.p)\ \lambda x.\lambda y.x$$
$$second \equiv \quad (\lambda p.p)\ \lambda x.\lambda y.y$$

Pairs

$$pair_{AB} \equiv \quad pair\ a\ b$$
$$\equiv \quad (\lambda x.\lambda y.\lambda z.z\ x\ y)\ a\ b$$

$$pair \equiv \quad \lambda x.\lambda y.\lambda z.z\ x\ y$$
$$first \equiv \quad (\lambda p.p)\ \lambda x.\lambda y.x$$
$$second \equiv \quad (\lambda p.p)\ \lambda x.\lambda y.y$$

Pairs

$$pair_{AB} \equiv \quad\quad\quad\quad pair\ a\ b$$
$$\equiv \quad (\lambda x.\lambda y.\lambda z.z\ x\ y)\ a\ b$$

$$pair \equiv \quad \lambda x.\lambda y.\lambda z.z\ x\ y$$
$$first \equiv \quad (\lambda p.p)\ \lambda x.\lambda y.x$$
$$second \equiv \quad (\lambda p.p)\ \lambda x.\lambda y.y$$

Pairs

$pair \equiv \lambda x.\lambda y.\lambda z.z\ x\ y$

$first \equiv (\lambda p.p)\ \lambda x.\lambda y.x$

$second \equiv (\lambda p.p)\ \lambda x.\lambda y.y$

$pair_{AB} \equiv pair\ a\ b$

$\equiv (\lambda x.\lambda y.\lambda z.z\ x\ y)\ a\ b$

$\rightarrow (\lambda y.\lambda z.z\ a\ y)\ b$

Pairs

$$pair \equiv \quad \lambda x.\lambda y.\lambda z.z\ x\ y$$

$$first \equiv \quad (\lambda p.p)\ \lambda x.\lambda y.x$$

$$second \equiv \quad (\lambda p.p)\ \lambda x.\lambda y.y$$

$$pair_{AB} \equiv \qquad\qquad pair\ a\ b$$

$$\equiv \quad (\lambda x.\lambda y.\lambda z.z\ x\ y)\ a\ b$$

$$\rightarrow \qquad (\lambda y.\lambda z.z\ a\ y)\ b$$

Pairs

$$pair \equiv \lambda x.\lambda y.\lambda z.z\ x\ y$$

$$first \equiv (\lambda p.p)\ \lambda x.\lambda y.x$$

$$second \equiv (\lambda p.p)\ \lambda x.\lambda y.y$$

$$pair_{AB} \equiv \quad pair\ a\ b$$

$$\equiv \quad (\lambda x.\lambda y.\lambda z.z\ x\ y)\ a\ b$$

$$\rightarrow \quad (\lambda y.\lambda z.z\ a\ y)\ b$$

$$\rightarrow \quad \lambda z.z\ a\ b$$

Pairs

$$pair \equiv \lambda x.\lambda y.\lambda z.z\ x\ y$$

$$first \equiv (\lambda p.p)\ \lambda x.\lambda y.x$$

$$second \equiv (\lambda p.p)\ \lambda x.\lambda y.y$$

$$pair_{AB} \equiv pair\ a\ b$$

$$\equiv (\lambda x.\lambda y.\lambda z.z\ x\ y)\ a\ b$$

$$\rightarrow (\lambda y.\lambda z.z\ a\ y)\ b$$

$$\rightarrow \lambda z.z\ a\ b$$

$$\equiv pair'_{ab}$$

Pairs (continued)

$$pair \equiv \quad \lambda x.\lambda y.\lambda z.z\ x\ y$$
$$first \equiv \quad (\lambda p.p)\ \lambda x.\lambda y.x$$
$$pair'_{ab} \equiv \qquad \lambda z.z\ a\ b$$

$$first\ pair'_{ab}$$

$$
\begin{aligned}
pair &\equiv \quad \lambda x.\lambda y.\lambda z.z\ x\ y \\
first &\equiv \quad (\lambda p.p)\ \lambda x.\lambda y.x \\
pair'_{ab} &\equiv \qquad \lambda z.z\ a\ b
\end{aligned}
$$

$$first\ pair'_{ab}$$

$$
\begin{aligned}
pair &\equiv \lambda x.\lambda y.\lambda z.z\ x\ y \\
first &\equiv (\lambda p.p)\ \lambda x.\lambda y.x \\
pair'_{ab} &\equiv \lambda z.z\ a\ b
\end{aligned}
$$

$$
\begin{aligned}
pair &\equiv \lambda x.\lambda y.\lambda z.z\ x\ y \\
first &\equiv (\lambda p.p)\ \lambda x.\lambda y.x \\
pair'_{ab} &\equiv \lambda z.z\ a\ b
\end{aligned}
$$

$$
\begin{aligned}
& \quad first\ pair'_{ab} \\
\equiv &\ (\lambda p.p)\ (\lambda x.\lambda y.x)\ pair'_{ab}
\end{aligned}
$$

$$pair \equiv \lambda x.\lambda y.\lambda z.z\ x\ y$$

$$first \equiv (\lambda p.p)\ \lambda x.\lambda y.x$$

$$pair'_{ab} \equiv \lambda z.z\ a\ b$$

$$first\ pair'_{ab}$$

$$\equiv (\lambda p.p)\ (\lambda x.\lambda y.x)\ pair'_{ab}$$

$$pair \equiv \lambda x.\lambda y.\lambda z.z\ x\ y$$
$$first \equiv (\lambda p.p)\ \lambda x.\lambda y.x$$
$$pair'_{ab} \equiv \lambda z.z\ a\ b$$

$$first\ pair'_{ab}$$
$$\equiv (\lambda p.p)\ (\lambda x.\lambda y.x)\ pair'_{ab}$$
$$\rightarrow pair'_{ab}\ (\lambda x.\lambda y.x)$$

Pairs (continued)

$$pair \equiv \lambda x.\lambda y.\lambda z.z\ x\ y$$

$$first \equiv (\lambda p.p)\ \lambda x.\lambda y.x$$

$$pair'_{ab} \equiv \lambda z.z\ a\ b$$

$$first\ pair'_{ab}$$
$$\equiv\quad (\lambda p.p)\ (\lambda x.\lambda y.x)\ pair'_{ab}$$
$$\rightarrow\quad pair'_{ab}\ (\lambda x.\lambda y.x)$$

$$pair \equiv \quad \lambda x.\lambda y.\lambda z.z\ x\ y$$

$$first \equiv \quad (\lambda p.p)\ \lambda x.\lambda y.x$$

$$pair'_{ab} \equiv \quad\quad\quad \lambda z.z\ a\ b$$

$$first\ pair'_{ab}$$

$$\equiv \quad (\lambda p.p)\ (\lambda x.\lambda y.x)\ pair'_{ab}$$

$$\rightarrow \quad\quad pair'_{ab}\ (\lambda x.\lambda y.x)$$

$$\equiv \quad\quad (\lambda z.z\ a\ b)\ (\lambda x.\lambda y.x)$$

Pairs (continued)

$$pair \equiv \lambda x.\lambda y.\lambda z.z\ x\ y$$
$$first \equiv (\lambda p.p)\ \lambda x.\lambda y.x$$
$$pair'_{ab} \equiv \lambda z.z\ a\ b$$

$$first\ pair'_{ab}$$
$$\equiv (\lambda p.p)\ (\lambda x.\lambda y.x)\ pair'_{ab}$$
$$\rightarrow pair'_{ab}\ (\lambda x.\lambda y.x)$$
$$\equiv (\lambda z.z\ a\ b)\ (\lambda x.\lambda y.x)$$

Pairs (continued)

$$pair \equiv \lambda x.\lambda y.\lambda z.z\ x\ y$$

$$first \equiv (\lambda p.p)\ \lambda x.\lambda y.x$$

$$pair'_{ab} \equiv \lambda z.z\ a\ b$$

$$first\ pair'_{ab}$$
$$\equiv (\lambda p.p)\ (\lambda x.\lambda y.x)\ pair'_{ab}$$
$$\rightarrow pair'_{ab}\ (\lambda x.\lambda y.x)$$
$$\equiv (\lambda z.z\ a\ b)\ (\lambda x.\lambda y.x)$$
$$\rightarrow (\lambda x.\lambda y.x)\ a\ b$$

Pairs (continued)

$$pair \equiv \quad \lambda x.\lambda y.\lambda z.z \; x \; y$$

$$first \equiv \quad (\lambda p.p) \; \lambda x.\lambda y.x$$

$$pair'_{ab} \equiv \quad \lambda z.z \; a \; b$$

$$first \; pair'_{ab}$$

$$\equiv \quad (\lambda p.p) \; (\lambda x.\lambda y.x) \; pair'_{ab}$$

$$\rightarrow \quad pair'_{ab} \; (\lambda x.\lambda y.x)$$

$$\equiv \quad (\lambda z.z \; a \; b) \; (\lambda x.\lambda y.x)$$

$$\rightarrow \quad (\lambda x.\lambda y.x) \; a \; b$$

$$pair \equiv \lambda x.\lambda y.\lambda z.z\ x\ y$$

$$first \equiv (\lambda p.p)\ \lambda x.\lambda y.x$$

$$pair'_{ab} \equiv \lambda z.z\ a\ b$$

$$first\ pair'_{ab}$$
$$\equiv\ (\lambda p.p)\ (\lambda x.\lambda y.x)\ pair'_{ab}$$
$$\rightarrow\ pair'_{ab}\ (\lambda x.\lambda y.x)$$
$$\equiv\ (\lambda z.z\ a\ b)\ (\lambda x.\lambda y.x)$$
$$\rightarrow\ (\lambda x.\lambda y.x)\ a\ b$$
$$\rightarrow\ (\lambda y.a)\ b$$

$$pair \equiv \lambda x.\lambda y.\lambda z.z\ x\ y$$

$$first \equiv (\lambda p.p)\ \lambda x.\lambda y.x$$

$$pair'_{ab} \equiv \lambda z.z\ a\ b$$

$$first\ pair'_{ab}$$

$$\equiv (\lambda p.p)\ (\lambda x.\lambda y.x)\ pair'_{ab}$$

$$\rightarrow pair'_{ab}\ (\lambda x.\lambda y.x)$$

$$\equiv (\lambda z.z\ a\ b)\ (\lambda x.\lambda y.x)$$

$$\rightarrow (\lambda x.\lambda y.x)\ a\ b$$

$$\rightarrow (\lambda y.a)\ b$$

Pairs (continued)

$$pair \equiv \lambda x.\lambda y.\lambda z.z\ x\ y$$

$$first \equiv (\lambda p.p)\ \lambda x.\lambda y.x$$

$$pair'_{ab} \equiv \lambda z.z\ a\ b$$

$$
\begin{aligned}
& first\ pair'_{ab} \\
\equiv\ & (\lambda p.p)\ (\lambda x.\lambda y.x)\ pair'_{ab} \\
\rightarrow\ & pair'_{ab}\ (\lambda x.\lambda y.x) \\
\equiv\ & (\lambda z.z\ a\ b)\ (\lambda x.\lambda y.x) \\
\rightarrow\ & (\lambda x.\lambda y.x)\ a\ b \\
\rightarrow\ & (\lambda y.a)\ b \\
\rightarrow\ & a
\end{aligned}
$$

Numerals

Peano Axioms
Every natural number can be defined with 0 and a successor function

$$0 \equiv \qquad \lambda f.\lambda x.x$$
$$1 \equiv \qquad \lambda f.\lambda x.f\, x$$
$$2 \equiv \qquad \lambda f.\lambda x.f\,(f\, x)$$
$$3 \equiv \quad \lambda f.\lambda x.f\,(f\,(f\, x))$$

Meaning

- 0 f is evaluated 0 times
- 1 f is evaluated once
- x can be every lambda term

$$0 \equiv \qquad \lambda f.\lambda x.x$$
$$1 \equiv \qquad \lambda f.\lambda x.f\, x$$

Numerals Example - Successor

$$0 \equiv \lambda f.\lambda x.x$$
$$1 \equiv \lambda f.\lambda x.f\,x$$

$$successor \equiv \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)$$

Numerals Example - Successor

successor 1

$$0 \equiv \lambda f.\lambda x.x$$
$$1 \equiv \lambda f.\lambda x.f\ x$$

$$successor \equiv \lambda n.\lambda f.\lambda x.f\ (n\ f\ x)$$

successor 1

$$0 \equiv \qquad \lambda f.\lambda x.x$$
$$1 \equiv \qquad \lambda f.\lambda x.f\, x$$

$$successor \equiv \quad \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)$$

Numerals Example - Successor

$$0 \equiv \lambda f.\lambda x.x$$

$$1 \equiv \lambda f.\lambda x.f\, x$$

$$successor \equiv \lambda n.\lambda f.\lambda x.f\, (n\ f\ x)$$

$$
\begin{aligned}
&successor\ 1\\
\equiv\ &(\lambda n.\lambda f.\lambda x.f\, (n\ f\ x))\ 1
\end{aligned}
$$

$$0 \equiv \lambda f.\lambda x.x$$

$$1 \equiv \lambda f.\lambda x.f\ x$$

$$successor \equiv \lambda n.\lambda f.\lambda x.f\ (n\ f\ x)$$

$$successor\ 1$$
$$\equiv (\lambda n.\lambda f.\lambda x.f\ (n\ f\ x))\ 1$$

$$0 \equiv \qquad \lambda f.\lambda x.x$$

$$1 \equiv \qquad \lambda f.\lambda x.f\, x$$

$$successor \equiv \quad \lambda n.\lambda f.\lambda x.f\,(n\; f\; x)$$

$$successor\; 1$$

$$\equiv \quad (\lambda n.\lambda f.\lambda x.f\,(n\; f\; x))\; 1$$

$$\rightarrow \qquad \lambda f.\lambda x.f\,(1\; f\; x)$$

$$successor\ 1$$

$$\equiv\quad (\lambda n.\lambda f.\lambda x.f\,(n\ f\ x))\ 1$$

$$0 \equiv \qquad \lambda f.\lambda x.x$$

$$\rightarrow\quad \lambda f.\lambda x.f\,(1\ f\ x)$$

$$1 \equiv \qquad \lambda f.\lambda x.f\,x$$

$$successor \equiv\ \ \lambda n.\lambda f.\lambda x.f\,(n\ f\ x)$$

Note
We use *Normal Order Reduction* to reduce inside abstractions!

$$successor\ 1$$

$$\equiv \quad (\lambda n.\lambda f.\lambda x.f\,(n\ f\ x))\ 1$$

$$0 \equiv \qquad \lambda f.\lambda x.x$$

$$\rightarrow \qquad \lambda f.\lambda x.f\,(1\ f\ x)$$

$$1 \equiv \qquad \lambda f.\lambda x.f\ x$$

$$\equiv \quad \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\ x)\ f\ x)$$

$$successor \equiv \quad \lambda n.\lambda f.\lambda x.f\,(n\ f\ x)$$

Note
We use *Normal Order Reduction* to reduce inside abstractions!

Numerals Example - Successor

$$successor\ 1$$

$$0 \equiv \lambda f.\lambda x.x$$

$$1 \equiv \lambda f.\lambda x.f\ x$$

$$successor \equiv \lambda n.\lambda f.\lambda x.f\ (n\ f\ x)$$

$$\equiv (\lambda n.\lambda f.\lambda x.f\ (n\ f\ x))\ 1$$

$$\rightarrow \lambda f.\lambda x.f\ (1\ f\ x)$$

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

Note
We use *Normal Order Reduction* to reduce inside abstractions!

$$successor\ 1$$

$$\equiv\quad (\lambda n.\lambda f.\lambda x.f\,(n\ f\ x))\ 1$$

$$0 \equiv \qquad \lambda f.\lambda x.x$$

$$\rightarrow\quad \lambda f.\lambda x.f\,(1\ f\ x)$$

$$1 \equiv \qquad \lambda f.\lambda x.f\ x$$

$$\equiv\quad \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\ x)\ f\ x)$$

$$\rightarrow\quad \lambda f.\lambda x.f\,((\lambda x.f\ x)\ x)$$

$$successor \equiv\quad \lambda n.\lambda f.\lambda x.f\,(n\ f\ x)$$

Note
We use *Normal Order Reduction* to reduce inside abstractions!

Numerals Example - Successor

$$0 \equiv \lambda f. \lambda x. x$$

$$1 \equiv \lambda f. \lambda x. f\, x$$

$$successor \equiv \lambda n. \lambda f. \lambda x. f\, (n\, f\, x)$$

$$successor\ 1$$

$$\equiv (\lambda n. \lambda f. \lambda x. f\, (n\, f\, x))\ 1$$

$$\rightarrow \lambda f. \lambda x. f\, (1\, f\, x)$$

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

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

Note
We use *Normal Order Reduction* to reduce inside abstractions!

$$successor\ 1$$

$$\equiv (\lambda n.\lambda f.\lambda x.f\ (n\ f\ x))\ 1$$

$$0 \equiv \lambda f.\lambda x.x$$

$$\rightarrow \lambda f.\lambda x.f\ (1\ f\ x)$$

$$1 \equiv \lambda f.\lambda x.f\ x$$

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

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

$$successor \equiv \lambda n.\lambda f.\lambda x.f\ (n\ f\ x)$$

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

Note
We use *Normal Order Reduction* to reduce inside abstractions!

$$successor\ 1$$

$$\equiv\ (\lambda n.\lambda f.\lambda x.f\,(n\ f\ x))\ 1$$

$$0 \equiv \lambda f.\lambda x.x$$

$$\rightarrow\ \lambda f.\lambda x.f\,(1\ f\ x)$$

$$1 \equiv \lambda f.\lambda x.f\ x$$

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

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

$$successor \equiv \lambda n.\lambda f.\lambda x.f\,(n\ f\ x)$$

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

$$\equiv\ 2$$

Note
We use *Normal Order Reduction* to reduce inside abstractions!

$$0 \equiv \qquad \lambda f.\lambda x.x$$

$$0 \equiv \qquad \lambda f.\lambda x.x$$

$$plus \equiv \quad \lambda m.\lambda n.\lambda f.\lambda x.m \; f \; (n \; f \; x)$$

$$plus\ 0\ 0$$

$$0 \equiv \lambda f.\lambda x.x$$

$$plus \equiv \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x)$$

plus 0 0

$$0 \equiv \qquad\qquad \lambda f.\lambda x.x$$

$$plus \equiv \quad \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x)$$

$$plus\ 0\ 0$$

$$\equiv\quad (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0$$

$$0 \equiv \qquad\qquad \lambda f.\lambda x.x$$

$$plus \equiv \quad \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x)$$

$$plus\ 0\ 0$$
$$\equiv\ (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0$$

$$0 \equiv \lambda f.\lambda x.x$$

$$plus \equiv \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x)$$

25

$$plus \; 0 \; 0$$
$$\equiv \quad (\lambda m.\lambda n.\lambda f.\lambda x.m \; f \; (n \; f \; x)) \; 0 \; 0$$
$$\rightarrow \quad (\lambda n.\lambda f.\lambda x.0 \; f \; (n \; f \; x)) \; 0$$

$$0 \equiv \qquad \lambda f.\lambda x.x$$

$$plus \equiv \quad \lambda m.\lambda n.\lambda f.\lambda x.m \; f \; (n \; f \; x)$$

$$plus\ 0\ 0$$

$$\equiv\quad (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0$$

$$\rightarrow\quad (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0$$

$$0 \equiv \lambda f.\lambda x.x$$

$$plus \equiv \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x)$$

$$plus\ 0\ 0$$

$$\equiv\quad (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0$$

$$\rightarrow\quad (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0$$

$$\rightarrow\quad \lambda f.\lambda x.0\ f\ (0\ f\ x)$$

$$0 \equiv \qquad\qquad \lambda f.\lambda x.x$$

$$plus \equiv \quad \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x)$$

$$plus\ 0\ 0$$

$$\equiv\quad (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0$$

$$\rightarrow\quad (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0$$

$$\rightarrow\quad \lambda f.\lambda x.0\ f\ (0\ f\ x)$$

$$0 \equiv\quad \lambda f.\lambda x.x$$

$$plus \equiv\quad \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x)$$

$$plus \ 0 \ 0$$

$$\equiv \quad (\lambda m.\lambda n.\lambda f.\lambda x.m \ f \ (n \ f \ x)) \ 0 \ 0$$

$$\rightarrow \quad (\lambda n.\lambda f.\lambda x.0 \ f \ (n \ f \ x)) \ 0$$

$$\rightarrow \quad \lambda f.\lambda x.0 \ f \ (0 \ f \ x)$$

$$0 \equiv \qquad \lambda f.\lambda x.x \qquad \equiv \quad \lambda f.\lambda x.(\lambda f.\lambda x.x) \ f \ (0 \ f \ x)$$

$$plus \equiv \quad \lambda m.\lambda n.\lambda f.\lambda x.m \ f \ (n \ f \ x)$$

$$plus\ 0\ 0$$

$$\equiv\quad (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0$$

$$\rightarrow\quad (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0$$

$$\rightarrow\quad \lambda f.\lambda x.0\ f\ (0\ f\ x)$$

$$0 \equiv \qquad\qquad \lambda f.\lambda x.x \quad \equiv\quad \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x)$$

$$plus \equiv\quad \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x)$$

$$plus\ 0\ 0$$

$$\equiv \quad (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0$$

$$\rightarrow \quad (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0$$

$$\rightarrow \quad \lambda f.\lambda x.0\ f\ (0\ f\ x)$$

$$0 \equiv \qquad\qquad \lambda f.\lambda x.x \quad \equiv \quad \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x)$$

$$\rightarrow \quad \lambda f.\lambda x.(\lambda x.x)\ (0\ f\ x)$$

$$plus \equiv \quad \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x)$$

$$plus\ 0\ 0$$

$$\equiv\ (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0$$

$$\rightarrow\ (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0$$

$$\rightarrow\ \lambda f.\lambda x.0\ f\ (0\ f\ x)$$

$$0 \equiv \qquad\qquad \lambda f.\lambda x.x \qquad \equiv\ \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x)$$

$$\rightarrow\ \lambda f.\lambda x.(\lambda x.x)\ (0\ f\ x)$$

$$plus \equiv\ \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x)$$

$$plus\ 0\ 0$$

$$\equiv\ (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0$$

$$\rightarrow\ (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0$$

$$\rightarrow\ \lambda f.\lambda x.0\ f\ (0\ f\ x)$$

$$0 \equiv\ \lambda f.\lambda x.x \qquad \equiv\ \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x)$$

$$\rightarrow\ \lambda f.\lambda x.(\lambda x.x)\ (0\ f\ x)$$

$$plus \equiv\ \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x) \qquad \rightarrow\ \lambda f.\lambda x.0\ f\ x$$

$$plus\ 0\ 0$$

$$\equiv\ (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0$$

$$\rightarrow\ (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0$$

$$\rightarrow\ \lambda f.\lambda x.0\ f\ (0\ f\ x)$$

$$0 \equiv \qquad \lambda f.\lambda x.x \qquad \equiv \qquad \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x)$$

$$\rightarrow\ \lambda f.\lambda x.(\lambda x.x)\ (0\ f\ x)$$

$$plus \equiv \quad \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x) \qquad \rightarrow \qquad \lambda f.\lambda x.0\ f\ x$$

$$plus\ 0\ 0$$

$$\equiv\ (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0$$

$$\rightarrow\ (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0$$

$$\rightarrow\ \lambda f.\lambda x.0\ f\ (0\ f\ x)$$

$$0 \equiv \qquad\qquad \lambda f.\lambda x.x \qquad\equiv\ \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x)$$

$$\rightarrow\ \lambda f.\lambda x.(\lambda x.x)\ (0\ f\ x)$$

$$plus \equiv \quad \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x) \qquad \rightarrow\ \lambda f.\lambda x.0\ f\ x$$

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

25

$$plus\ 0\ 0$$

$$\equiv (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0$$

$$\rightarrow (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0$$

$$\rightarrow \lambda f.\lambda x.0\ f\ (0\ f\ x)$$

$$0 \equiv \qquad \lambda f.\lambda x.x \qquad \equiv \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x)$$

$$\rightarrow \lambda f.\lambda x.(\lambda x.x)\ (0\ f\ x)$$

$$plus \equiv \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x) \qquad \rightarrow \lambda f.\lambda x.0\ f\ x$$

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

$$plus\ 0\ 0$$

$$\equiv\ (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0$$

$$\rightarrow\ (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0$$

$$\rightarrow\ \lambda f.\lambda x.0\ f\ (0\ f\ x)$$

$$0 \equiv \qquad\qquad \lambda f.\lambda x.x \qquad \equiv\ \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x)$$

$$\rightarrow\ \lambda f.\lambda x.(\lambda x.x)\ (0\ f\ x)$$

$$plus \equiv\ \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x) \qquad \rightarrow\ \lambda f.\lambda x.0\ f\ x$$

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

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

$plus\ 0\ 0$

$\equiv\quad (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0$

$\rightarrow\quad (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0$

$\rightarrow\quad \lambda f.\lambda x.0\ f\ (0\ f\ x)$

$0 \equiv \qquad\qquad \lambda f.\lambda x.x \qquad \equiv\quad \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x)$

$\rightarrow\quad \lambda f.\lambda x.(\lambda x.x)\ (0\ f\ x)$

$plus \equiv \quad \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x) \qquad \rightarrow\quad \lambda f.\lambda x.0\ f\ x$

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

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

$$plus\ 0\ 0$$

$$\equiv (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0$$

$$\rightarrow (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0$$

$$\rightarrow \lambda f.\lambda x.0\ f\ (0\ f\ x)$$

$$0 \equiv \lambda f.\lambda x.x \qquad \equiv \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x)$$

$$\rightarrow \lambda f.\lambda x.(\lambda x.x)\ (0\ f\ x)$$

$$plus \equiv \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x) \qquad \rightarrow \lambda f.\lambda x.0\ f\ x$$

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

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

$$\rightarrow \lambda f.\lambda x.x$$

$$plus\ 0\ 0$$

$$\equiv\quad (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0$$

$$\rightarrow\quad (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0$$

$$\rightarrow\quad \lambda f.\lambda x.0\ f\ (0\ f\ x)$$

$$0 \equiv \qquad\qquad \lambda f.\lambda x.x \qquad\equiv\quad \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x)$$

$$\rightarrow\quad \lambda f.\lambda x.(\lambda x.x)\ (0\ f\ x)$$

$$plus \equiv\ \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x) \qquad\rightarrow\quad \lambda f.\lambda x.0\ f\ x$$

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

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

$$\rightarrow\quad \lambda f.\lambda x.x$$

$$\equiv\quad 0$$

End

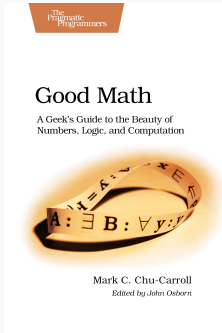## Thanks

- Hope you enjoyed this talk and learned something new.
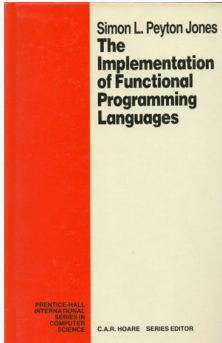- Hope it wasn't too much math and dusty formulas … :)

/"A Geek's Guide to the Beauty of Numbers, Logic, and Computation"/

- Easy to understand
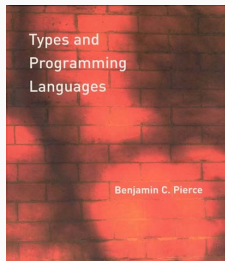
- How to compile to the Lambda Calculus?
- Out-of-print, but freely available
  - https://www.microsoft.com/en-us/research/publication/the-implementation-of-functional-programming-languages/

- Types systems explained by building interpreters and proving properties
- Very "mathematical", but very complete and self-contained