Untyped Lambda Calculus

Structure and Evaluation, Currying, Church Encodings

Sven Tennie

July 29, 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 proove

Why should I care?

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

$t ::=$	Terms:
x	Variable
$\lambda x.t$	Abstraction
$t\ t$	Application

Untyped Lambda Calculus

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

Example - Identity

Lambda Calculus

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

$$
\begin{array}{llr}
t ::= & & \text{Terms:} \\
 & x & \text{Variable} \\
 & \lambda x.t & \text{Abstraction} \\
 & t\ t & \text{Application}
\end{array}
$$

Example - Identity

Lambda Calculus

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

Javascript

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

Evaluation / Reduction

$$\underbrace{\underbrace{(\underbrace{\lambda x.\, \underbrace{\lambda y.\, \underbrace{x\ y}_{\textit{Application}}}_{\textit{Abstraction}}}_{\textit{Abstraction}})\ \underbrace{a}_{\textit{Variable}}}_{\textit{Application}}\ \underbrace{b}_{\textit{Variable}}}_{\textit{Application}}$$

Evaluation / Reduction

$$\underbrace{\underbrace{(\underbrace{\lambda x.\,\underbrace{\lambda y.\,\underbrace{x\ y}_{\textit{Application}}}_{\textit{Abstraction}}}_{\textit{Abstraction}})\ \underbrace{a}_{\textit{Variable}}}_{\textit{Application}}\ \underbrace{b}_{\textit{Variable}}}_{\textit{Application}}$$

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

Parentheses are not part of the grammer? See next slide. . . :)

Evaluation / Reduction

$$\underbrace{\underbrace{\underbrace{(\lambda x.\underbrace{\lambda y.\underbrace{x\ y}_{\textit{Application}}}_{\textit{Abstraction}})}_{\textit{Abstraction}}\ \underbrace{a}_{\textit{Variable}}}_{\textit{Application}}\ \underbrace{b}_{\textit{Variable}}}_{\textit{Application}}$$

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

$$\rightarrow(\lambda \mathrm{y}.a\ \mathrm{y})\ \mathrm{b}$$

Parentheses are not part of the grammer? See next slide... :)

Evaluation / Reduction

$$\underbrace{\underbrace{(\underbrace{\lambda x.\underbrace{\lambda y.\underbrace{x\ y}_{\textit{Application}}}_{\textit{Abstraction}}}_{\textit{Abstraction}})\ \underbrace{a}_{\textit{Variable}}}_{\textit{Application}}\ \underbrace{b}_{\textit{Variable}}}_{\textit{Application}}$$

$$
\begin{aligned}
&(\lambda \mathrm{x}.\lambda y.\mathrm{x}\ \ y)\ \mathrm{a}\ b \\
\rightarrow&(\lambda \mathrm{y}.a\ \ \mathrm{y})\ \mathrm{b} \\
\rightarrow&a\ \ b
\end{aligned}
$$

Parentheses are not part of the grammer? See next slide. . . :)

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 *binder*

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 *binder*

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"

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

Currying

Idea

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

Currying

Idea

- Take a funktion with n arguments
- Create a funktion 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 funktion with n arguments
- Create a funktion 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"

Reductions and Conversions

Alpha conversion

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

Reductions and Conversions

Alpha conversion

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

Beta reduction

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

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, no η conversion possible:

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

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

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

Full Beta-Reduction

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

$$\underbrace{(\lambda x.x)\ \underbrace{((\lambda x.x)\ (\lambda z.\,\underbrace{(\lambda x.x)\ z}_{\mathit{RedEx}}))}_{\mathit{RedEx}}}_{\mathit{RedEx}}$$

Full Beta-Reduction

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

Normal Order Reduction

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

Normal Order Reduction

- Left-most, Outer-most RedEx

Normal Order Reduction

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

Normal Order Reduction

- Left-most, Outer-most RedEx

Normal Order Reduction

$$
\begin{aligned}
&(\lambda x.x)\ ((\lambda x.x)\ (\lambda z.(\lambda x.x)\ z)) \\
\rightarrow &(\lambda x.x)\ (\lambda z.(\lambda x.x)\ z)
\end{aligned}
$$

Normal Order Reduction

- Left-most, Outer-most RedEx

Normal Order Reduction

$$
\begin{aligned}
&(\lambda x.x)\ ((\lambda x.x)\ (\lambda z.(\lambda x.x)\ z)) \\
\rightarrow &(\lambda x.x)\ (\lambda z.(\lambda x.x)\ z)
\end{aligned}
$$

Normal Order Reduction

- Left-most, Outer-most RedEx

Normal Order Reduction

$$
\begin{aligned}
&(\lambda x.x)\ ((\lambda x.x)\ (\lambda z.(\lambda x.x)\ z)) \\
\rightarrow &(\lambda x.x)\ (\lambda z.(\lambda x.x)\ z) \\
\rightarrow &\lambda z.(\lambda x.x)\ z
\end{aligned}
$$

Normal Order Reduction

- Left-most, Outer-most RedEx

Normal Order Reduction

$$
\begin{aligned}
&(\lambda x.x)\ ((\lambda x.x)\ (\lambda z.(\lambda x.x)\ z)) \\
\rightarrow &(\lambda x.x)\ (\lambda z.(\lambda x.x)\ z) \\
\rightarrow &\lambda z.(\lambda x.x)\ z
\end{aligned}
$$

Normal Order Reduction

- Left-most, Outer-most RedEx

Normal Order Reduction

$$
\begin{aligned}
&(\lambda x.x)\ ((\lambda x.x)\ (\lambda z.(\lambda x.x)\ z)) \\
\rightarrow &(\lambda x.x)\ (\lambda z.(\lambda x.x)\ z) \\
\rightarrow &\lambda z.(\lambda x.x)\ z \\
\rightarrow &\lambda z.z
\end{aligned}
$$

Normal Order Reduction

- Left-most, Outer-most RedEx

Call-by-Name

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

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 result $\rightarrow$ *Call-by-Need*

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

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 result $\rightarrow$ *Call-by-Need*

Call-by-Name

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

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 result $\rightarrow$ *Call-by-Need*

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

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 result → *Call-by-Need*

Call-by-Name

$$
\begin{aligned}
&(\lambda x.x)\ ((\lambda x.x)\ (\lambda z.(\lambda x.x)\ z)) \\
\to&(\lambda x.x)\ (\lambda z.(\lambda x.x)\ z) \\
\to&\lambda z.(\lambda x.x)\ z
\end{aligned}
$$

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 result $\to$ *Call-by-Need*

Call-by-Name

$$
\begin{aligned}
&(\lambda x.x)\ ((\lambda x.x)\ (\lambda z.(\lambda x.x)\ z)) \\
\rightarrow&(\lambda x.x)\ (\lambda z.(\lambda x.x)\ z) \\
\rightarrow&\lambda z.(\lambda x.x)\ z \\
\not\rightarrow&
\end{aligned}
$$

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 result $\rightarrow$ *Call-by-Need*

Call-by-Value

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

Call-by-Value

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

Call-by-Value

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

Call-by-Value

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

Call-by-Value

$$
\begin{aligned}
&(\lambda x.x)\ ((\lambda x.x)\ (\lambda z.(\lambda x.x)\ z)) \\
\rightarrow &(\lambda x.x)\ (\lambda z.(\lambda x.x)\ z)
\end{aligned}
$$

Call-by-Value

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

Call-by-Value

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

Call-by-Value

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

Call-by-Value

$$
\begin{aligned}
&(\lambda x.x)\ ((\lambda x.x)\ (\lambda z.(\lambda x.x)\ z)) \\
\to&(\lambda x.x)\ (\lambda z.(\lambda x.x)\ z) \\
\to&\lambda z.(\lambda x.x)\ z
\end{aligned}
$$

Call-by-Value

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

Call-by-Value

$$
\begin{aligned}
&(\lambda x.x)\ ((\lambda x.x)\ (\lambda z.(\lambda x.x)\ z)) \\
\rightarrow&(\lambda x.x)\ (\lambda z.(\lambda x.x)\ z) \\
\rightarrow&\lambda z.(\lambda x.x)\ z \\
\not\rightarrow&
\end{aligned}
$$

Call-by-Value

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

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

Definitions

$$\begin{aligned} \mathit{true} &\equiv \lambda t.\lambda f.t \\ \mathit{false} &\equiv \lambda t.\lambda f.f \end{aligned}$$

Definitions

$$
\begin{aligned}
\mathit{true} &\equiv \lambda t.\lambda f.t \\
\mathit{false} &\equiv \lambda t.\lambda f.f \\
\\
\mathit{test} &\equiv \lambda c.\lambda t.\lambda f.c\ t\ f
\end{aligned}
$$

Example

Example

Example

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

Example

$\textit{test true a b}$

Example

$$\begin{aligned} & \mathit{test}\ \mathit{true}\ a\ b \\ \equiv\ & (\lambda c.\lambda t.\lambda f.c\ t\ f)\ \mathit{true}\ a\ b \end{aligned}$$

Example

$$\begin{aligned} & \mathit{test}\ \mathit{true}\ a\ b \\ \equiv\ & (\lambda c.\lambda t.\lambda f.c\ t\ f)\ \mathit{true}\ a\ b \end{aligned}$$

Example

$$
\begin{aligned}
& \mathit{test}\ \mathit{true}\ a\ b \\
\equiv\ & (\lambda c.\lambda t.\lambda f.c\ t\ f)\ \mathit{true}\ a\ b \\
\rightarrow\ & (\lambda t.\lambda f.\mathit{true}\ t\ f)\ a\ b
\end{aligned}
$$

Example

$$
\begin{aligned}
& \mathit{test}\ \mathit{true}\ a\ b \\
\equiv\ & (\lambda c.\lambda t.\lambda f.c\ t\ f)\ \mathit{true}\ a\ b \\
\rightarrow\ & (\lambda t.\lambda f.\mathit{true}\ t\ f)\ a\ b
\end{aligned}
$$

Example

$$
\begin{aligned}
& \mathit{test}\ \mathit{true}\ a\ b \\
\equiv\ & (\lambda c.\lambda t.\lambda f.c\ t\ f)\ \mathit{true}\ a\ b \\
\to\ & (\lambda t.\lambda f.\mathit{true}\ t\ f)\ a\ b \\
\to\ & (\lambda f.\mathit{true}\ a\ f)\ b
\end{aligned}
$$

Example

$$
\begin{aligned}
& \mathit{test}\ \mathit{true}\ a\ b \\
\equiv\ & (\lambda c.\lambda t.\lambda f.c\ t\ f)\ \mathit{true}\ a\ b \\
\rightarrow\ & (\lambda t.\lambda f.\mathit{true}\ t\ f)\ a\ b \\
\rightarrow\ & (\lambda f.\mathit{true}\ a\ f)\ b
\end{aligned}
$$

Example

$$
\begin{aligned}
& test\ true\ a\ b \\
\equiv\ & (\lambda c.\lambda t.\lambda f.c\ t\ f)\ true\ a\ b \\
\rightarrow\ & (\lambda t.\lambda f.true\ t\ f)\ a\ b \\
\rightarrow\ & (\lambda f.true\ a\ f)\ b \\
\rightarrow\ & true\ a\ b
\end{aligned}
$$

Example

$$
\begin{aligned}
& test\ true\ a\ b \\
\equiv\ & (\lambda c.\lambda t.\lambda f.c\ t\ f)\ true\ a\ b \\
\rightarrow\ & (\lambda t.\lambda f.true\ t\ f)\ a\ b \\
\rightarrow\ & (\lambda f.true\ a\ f)\ b \\
\rightarrow & true\ a\ b
\end{aligned}
$$

Example

$$
\begin{aligned}
& \mathit{test}\ \mathit{true}\ a\ b \\
\equiv\ & (\lambda c.\lambda t.\lambda f.c\ t\ f)\ \mathit{true}\ a\ b \\
\rightarrow\ & (\lambda t.\lambda f.\mathit{true}\ t\ f)\ a\ b \\
\rightarrow\ & (\lambda f.\mathit{true}\ a\ f)\ b \\
\rightarrow\ & \mathit{true}\ a\ b \\
\equiv\ & (\lambda t.\lambda f.t)\ a\ b
\end{aligned}
$$

Example

$$
\begin{aligned}
& \mathit{test}\ \mathit{true}\ a\ b \\
\equiv\ & (\lambda c.\lambda t.\lambda f.c\ t\ f)\ \mathit{true}\ a\ b \\
\rightarrow\ & (\lambda t.\lambda f.\mathit{true}\ t\ f)\ a\ b \\
\rightarrow\ & (\lambda f.\mathit{true}\ a\ f)\ b \\
\rightarrow & \mathit{true}\ a\ b \\
\equiv & (\lambda t.\lambda f.t)\ a\ b
\end{aligned}
$$

Example

$$
\begin{aligned}
& \mathit{test}\ \mathit{true}\ a\ b \\
\equiv\ & (\lambda c.\lambda t.\lambda f.c\ t\ f)\ \mathit{true}\ a\ b \\
\rightarrow\ & (\lambda t.\lambda f.\mathit{true}\ t\ f)\ a\ b \\
\rightarrow\ & (\lambda f.\mathit{true}\ a\ f)\ b \\
\rightarrow\ & \mathit{true}\ a\ b \\
\equiv\ & (\lambda t.\lambda f.t)\ a\ b \\
\rightarrow\ & (\lambda f.a)\ b
\end{aligned}
$$

Example

$$
\begin{aligned}
& test\ true\ a\ b \\
\equiv\ & (\lambda c.\lambda t.\lambda f.c\ t\ f)\ true\ a\ b \\
\rightarrow\ & (\lambda t.\lambda f.true\ t\ f)\ a\ b \\
\rightarrow\ & (\lambda f.true\ a\ f)\ b \\
\rightarrow\ & true\ a\ b \\
\equiv\ & (\lambda t.\lambda f.t)\ a\ b \\
\rightarrow\ & (\lambda f.a)\ b
\end{aligned}
$$

Example

$$
\begin{aligned}
& \mathit{test}\ \mathit{true}\ a\ b \\
\equiv\ & (\lambda c.\lambda t.\lambda f.c\ t\ f)\ \mathit{true}\ a\ b \\
\to\ & (\lambda t.\lambda f.\mathit{true}\ t\ f)\ a\ b \\
\to\ & (\lambda f.\mathit{true}\ a\ f)\ b \\
\to & \mathit{true}\ a\ b \\
\equiv & (\lambda t.\lambda f.t)\ a\ b \\
\to & (\lambda f.a)\ b \\
\to & a
\end{aligned}
$$

And

Definitions

$$\begin{aligned} \mathit{true} &\equiv \lambda t.\lambda f.t \\ \mathit{false} &\equiv \lambda t.\lambda f.f \end{aligned}$$

Definitions

$$\begin{aligned} \mathit{true} &\equiv \lambda t.\lambda f.t \\ \mathit{false} &\equiv \lambda t.\lambda f.f \\[1em] \mathit{and} &\equiv \lambda p.\lambda q.p\ q\ p \end{aligned}$$

Example

Example

Example

$$and\ \ true\ \ false$$

Example

and true false

Example

$$
\begin{aligned}
&\mathit{and}\ \mathit{true}\ \mathit{false} \\
\equiv&(\lambda p.\lambda q.p\ q\ p)\ \mathit{true}\ \mathit{false}
\end{aligned}
$$

Example

$$
\begin{aligned}
& \mathit{and}\ \mathit{true}\ \mathit{false} \\
\equiv & (\lambda p.\lambda q.p\ q\ p)\ \mathit{true}\ \mathit{false}
\end{aligned}
$$

Example

$$
\begin{aligned}
& \mathit{and}\ \mathit{true}\ \mathit{false} \\
\equiv & (\lambda p.\lambda q.p\ q\ p)\ \mathit{true}\ \mathit{false} \\
\rightarrow & (\lambda q.\mathit{true}\ q\ \mathit{true})\ \mathit{false}
\end{aligned}
$$

Example

$$
\begin{aligned}
& \mathit{and}\ \mathit{true}\ \mathit{false} \\
\equiv & (\lambda p.\lambda q.p\ q\ p)\ \mathit{true}\ \mathit{false} \\
\rightarrow & (\lambda q.\mathit{true}\ q\ \mathit{true})\ \mathit{false}
\end{aligned}
$$

Example

$$\begin{aligned} & \mathit{and}\ \mathit{true}\ \mathit{false} \\ \equiv & (\lambda p.\lambda q.p\ q\ p)\ \mathit{true}\ \mathit{false} \\ \rightarrow & (\lambda q.\mathit{true}\ q\ \mathit{true})\ \mathit{false} \\ \rightarrow & \mathit{true}\ \mathit{false}\ \mathit{true} \end{aligned}$$

Example

$$
\begin{aligned}
& and\ true\ false \\
\equiv & (\lambda p.\lambda q.p\ q\ p)\ true\ false \\
\rightarrow & (\lambda q.true\ q\ true)\ false \\
\rightarrow & true\ false\ true
\end{aligned}
$$

Example

$$
\begin{aligned}
& \mathit{and}\ \mathit{true}\ \mathit{false} \\
\equiv & (\lambda p.\lambda q.p\ q\ p)\ \mathit{true}\ \mathit{false} \\
\rightarrow & (\lambda q.\mathit{true}\ q\ \mathit{true})\ \mathit{false} \\
\rightarrow & \mathit{true}\ \mathit{false}\ \mathit{true} \\
\equiv & (\lambda t.\lambda f.t)\ \mathit{false}\ \mathit{true}
\end{aligned}
$$

Example

$$
\begin{aligned}
&\mathit{and}\ \mathit{true}\ \mathit{false} \\
\equiv&(\lambda p.\lambda q.p\ q\ p)\ \mathit{true}\ \mathit{false} \\
\rightarrow&(\lambda q.\mathit{true}\ q\ \mathit{true})\ \mathit{false} \\
\rightarrow&\mathit{true}\ \mathit{false}\ \mathit{true} \\
\equiv&(\lambda t.\lambda f.t)\ \mathit{false}\ \mathit{true}
\end{aligned}
$$

Example

$$
\begin{aligned}
& \mathit{and}\ \mathit{true}\ \mathit{false} \\
\equiv & (\lambda p.\lambda q.p\ q\ p)\ \mathit{true}\ \mathit{false} \\
\rightarrow & (\lambda q.\mathit{true}\ q\ \mathit{true})\ \mathit{false} \\
\rightarrow & \mathit{true}\ \mathit{false}\ \mathit{true} \\
\equiv & (\lambda t.\lambda f.t)\ \mathit{false}\ \mathit{true} \\
\rightarrow & (\lambda f.\mathit{false})\ \mathit{true}
\end{aligned}
$$

Example

$$
\begin{aligned}
& \mathit{and}\ \mathit{true}\ \mathit{false} \\
\equiv & (\lambda p.\lambda q.p\ q\ p)\ \mathit{true}\ \mathit{false} \\
\rightarrow & (\lambda q.\mathit{true}\ q\ \mathit{true})\ \mathit{false} \\
\rightarrow & \mathit{true}\ \mathit{false}\ \mathit{true} \\
\equiv & (\lambda t.\lambda f.t)\ \mathit{false}\ \mathit{true} \\
\rightarrow & (\lambda f.\mathit{false})\ \mathit{true}
\end{aligned}
$$

Example

$$
\begin{aligned}
& \mathit{and}\ \mathit{true}\ \mathit{false} \\
\equiv & (\lambda p.\lambda q.p\ q\ p)\ \mathit{true}\ \mathit{false} \\
\rightarrow & (\lambda q.\mathit{true}\ q\ \mathit{true})\ \mathit{false} \\
\rightarrow & \mathit{true}\ \mathit{false}\ \mathit{true} \\
\equiv & (\lambda t.\lambda f.t)\ \mathit{false}\ \mathit{true} \\
\rightarrow & (\lambda f.\mathit{false})\ \mathit{true} \\
\rightarrow & \mathit{false}
\end{aligned}
$$

Or

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

Pairs

Definitions

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

Definitions

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

Example

Example

Example

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

Example

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

Example

$$
\begin{aligned}
pair_{AB} &\equiv pair\ a\ b \\
&\equiv (\lambda x.\lambda y.\lambda z.z\ x\ y)\ a\ b
\end{aligned}
$$

Example

$$
\begin{aligned}
pair_{AB} &\equiv pair\ a\ b \\
&\equiv (\lambda x.\lambda y.\lambda z.z\ x\ y)\ a\ b
\end{aligned}
$$

Example

$$
\begin{aligned}
pair_{AB} &\equiv & pair\ a\ b \\
&\equiv & (\lambda x.\lambda y.\lambda z.z\ x\ y)\ a\ b \\
&\rightarrow & (\lambda y.\lambda z.z\ a\ y) b
\end{aligned}
$$

Example

$$
\begin{aligned}
pair_{AB} &\equiv pair\ a\ b \\
&\equiv (\lambda x.\lambda y.\lambda z.z\ x\ y)\ a\ b \\
&\rightarrow (\lambda y.\lambda z.z\ a\ y)b
\end{aligned}
$$

Example

$$
\begin{aligned}
pair_{AB} &\equiv \quad pair\ a\ b \\
&\equiv \quad (\lambda x.\lambda y.\lambda z.z\ x\ y)\ a\ b \\
&\to \quad (\lambda y.\lambda z.z\ a\ y) b \\
&\to \quad \lambda z.z\ a\ b
\end{aligned}
$$

Example

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

Pairs (continued)

Definitions

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

Example

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Numerals

Peano axioms

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

$$
\begin{aligned}
0 &\equiv \lambda f.\lambda x.x \\
1 &\equiv \lambda f.\lambda x.f\ x \\
2 &\equiv \lambda f.\lambda x.f\ (f\ x) \\
3 &\equiv \lambda f.\lambda x.f\ (f\ (f\ x))
\end{aligned}
$$

Meaning

0 f is evaluated 0 times

1 f is evaluated once

x can be every lambda term

Numerals Example - Successor

Definitions

$$
\begin{aligned}
0 &\equiv \lambda f.\lambda x.x \\
1 &\equiv \lambda f.\lambda x.f\ x
\end{aligned}
$$

Definitions

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

Example

Example

Example

successor 1

Example

successor 1

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Example

$$
\begin{aligned}
& \textit{successor}\ 1 \\
\equiv\ & (\lambda n.\lambda f.\lambda x.f\ (n\ f\ x))\ 1 \\
\rightarrow\ & \lambda f.\lambda x.f\ (1\ f\ x) \\
\equiv\ & \lambda f.\lambda x.f\ ((\lambda f.\lambda x.f\ x)\ f\ x) \\
\rightarrow\ & \lambda f.\lambda x.f\ ((\lambda x.f\ x)\ x) \\
\rightarrow\ & \lambda f.\lambda x.f\ (f\ x) \\
\equiv\ & 2
\end{aligned}
$$

Note

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

Numerals Example - 0 + 0

Definitions

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

Definitions

$$
\begin{aligned}
0 &\equiv \lambda f.\lambda x.x \\
plus &\equiv \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x)
\end{aligned}
$$

Definitions

$$\textit{plus}\ 0\ 0$$

Definitions

$$plus\ 0\ 0$$

Definitions

$$
\begin{aligned}
& \textit{plus}\ 0\ 0 \\
\equiv\quad & (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0
\end{aligned}
$$

Definitions

$$
\begin{aligned}
& \mathit{plus}\ 0\ 0 \\
\equiv\ & (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0
\end{aligned}
$$

Definitions

$$
\begin{aligned}
& \mathit{plus}\ 0\ 0 \\
\equiv\ & (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0 \\
\rightarrow\ & (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0
\end{aligned}
$$

Definitions

$$
\begin{aligned}
& & plus\ 0\ 0 \\
&\equiv & (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0 \\
&\to & (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0
\end{aligned}
$$

Definitions

$$
\begin{aligned}
& & \mathit{plus}\ 0\ 0 \\
&\equiv & (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0 \\
&\rightarrow & (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0 \\
&\rightarrow & \lambda f.\lambda x.0\ f\ (0\ f\ x)
\end{aligned}
$$

Definitions

$$
\begin{aligned}
& & \mathit{plus}\ 0\ 0 \\
&\equiv & (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0 \\
&\rightarrow & (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0 \\
&\rightarrow & \lambda f.\lambda x.0\ f\ (0\ f\ x)
\end{aligned}
$$

Definitions

$$\begin{aligned}
& \mathit{plus}\ 0\ 0 \\
\equiv\quad & (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0 \\
\rightarrow\quad & (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0 \\
\rightarrow\quad & \lambda f.\lambda x.0\ f\ (0\ f\ x) \\
\equiv\quad & \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x)
\end{aligned}$$

Definitions

$$
\begin{aligned}
& \mathit{plus}\ 0\ 0 \\
\equiv\ & (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0 \\
\rightarrow\ & (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0 \\
\rightarrow\ & \lambda f.\lambda x.0\ f\ (0\ f\ x) \\
\equiv\ & \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x)
\end{aligned}
$$

Definitions

$$
\begin{aligned}
& & \mathit{plus}\ 0\ 0 \\
&\equiv & (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0 \\
&\rightarrow & (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0 \\
&\rightarrow & \lambda f.\lambda x.0\ f\ (0\ f\ x) \\
&\equiv & \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x) \\
&\rightarrow & \lambda f.\lambda x.(\lambda x.x)\ (0\ f\ x)
\end{aligned}
$$

Definitions

$$
\begin{aligned}
& & plus\ 0\ 0 \\
&\equiv & (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0 \\
&\rightarrow & (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0 \\
&\rightarrow & \lambda f.\lambda x.0\ f\ (0\ f\ x) \\
&\equiv & \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x) \\
&\rightarrow & \lambda f.\lambda x.(\lambda x.x)\ (0\ f\ x)
\end{aligned}
$$

Definitions

$$
\begin{aligned}
& & \mathit{plus}\ 0\ 0 \\
\equiv & & (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0 \\
\to & & (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0 \\
\to & & \lambda f.\lambda x.0\ f\ (0\ f\ x) \\
\equiv & & \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x) \\
\to & & \lambda f.\lambda x.(\lambda x.x)\ (0\ f\ x) \\
\to & & \lambda f.\lambda x.0\ f\ x
\end{aligned}
$$

Definitions

$$
\begin{array}{rr}
 & \mathit{plus}\ 0\ 0 \\
\equiv & (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0 \\
\to & (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0 \\
\to & \lambda f.\lambda x.0\ f\ (0\ f\ x) \\
\equiv & \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x) \\
\to & \lambda f.\lambda x.(\lambda x.x)\ (0\ f\ x) \\
\to & \lambda f.\lambda x.0\ f\ x
\end{array}
$$

Definitions

$$
\begin{aligned}
& \mathit{plus}\ 0\ 0 \\
\equiv\ & (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0 \\
\rightarrow\ & (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0 \\
\rightarrow\ & \lambda f.\lambda x.0\ f\ (0\ f\ x) \\
\equiv\ & \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x) \\
\rightarrow\ & \lambda f.\lambda x.(\lambda x.x)\ (0\ f\ x) \\
\rightarrow\ & \lambda f.\lambda x.0\ f\ x \\
\equiv\ & \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ x
\end{aligned}
$$

Definitions

$$
\begin{aligned}
& plus\ 0\ 0 \\
\equiv\quad & (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0 \\
\rightarrow\quad & (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0 \\
\rightarrow\quad & \lambda f.\lambda x.0\ f\ (0\ f\ x) \\
\equiv\quad & \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x) \\
\rightarrow\quad & \lambda f.\lambda x.(\lambda x.x)\ (0\ f\ x) \\
\rightarrow\quad & \lambda f.\lambda x.0\ f\ x \\
\equiv\quad & \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ x
\end{aligned}
$$

Definitions

$$
\begin{aligned}
& & \mathit{plus}\ 0\ 0 \\
&\equiv & (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0 \\
&\rightarrow & (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0 \\
&\rightarrow & \lambda f.\lambda x.0\ f\ (0\ f\ x) \\
&\equiv & \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x) \\
&\rightarrow & \lambda f.\lambda x.(\lambda x.x)\ (0\ f\ x) \\
&\rightarrow & \lambda f.\lambda x.0\ f\ x \\
&\equiv & \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ x \\
&\rightarrow & \lambda f.\lambda x.(\lambda x.x)\ x
\end{aligned}
$$

Definitions

$$
\begin{aligned}
& \mathit{plus}\ 0\ 0 \\
\equiv\ & (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0 \\
\rightarrow\ & (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0 \\
\rightarrow\ & \lambda f.\lambda x.0\ f\ (0\ f\ x) \\
\equiv\ & \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x) \\
\rightarrow\ & \lambda f.\lambda x.(\lambda x.x)\ (0\ f\ x) \\
\rightarrow\ & \lambda f.\lambda x.0\ f\ x \\
\equiv\ & \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ x \\
\rightarrow\ & \lambda f.\lambda x.(\lambda x.x)\ x
\end{aligned}
$$

Definitions

$$
\begin{aligned}
& \mathit{plus}\ 0\ 0 \\
\equiv\ & (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0 \\
\to\ & (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0 \\
\to\ & \lambda f.\lambda x.0\ f\ (0\ f\ x) \\
\equiv\ & \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x) \\
\to\ & \lambda f.\lambda x.(\lambda x.x)\ (0\ f\ x) \\
\to\ & \lambda f.\lambda x.0\ f\ x \\
\equiv\ & \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ x \\
\to\ & \lambda f.\lambda x.(\lambda x.x)\ x \\
\to\ & \lambda f.\lambda x.x
\end{aligned}
$$

Definitions

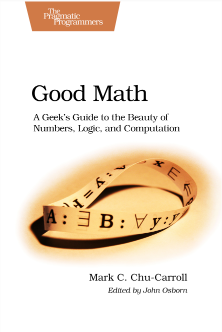
$$
\begin{aligned}
& \mathit{plus}\ 0\ 0 \\
\equiv\ & (\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x))\ 0\ 0 \\
\rightarrow\ & (\lambda n.\lambda f.\lambda x.0\ f\ (n\ f\ x))\ 0 \\
\rightarrow\ & \lambda f.\lambda x.0\ f\ (0\ f\ x) \\
\equiv\ & \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ (0\ f\ x) \\
\rightarrow\ & \lambda f.\lambda x.(\lambda x.x)\ (0\ f\ x) \\
\rightarrow\ & \lambda f.\lambda x.0\ f\ x \\
\equiv\ & \lambda f.\lambda x.(\lambda f.\lambda x.x)\ f\ x \\
\rightarrow\ & \lambda f.\lambda x.(\lambda x.x)\ x \\
\rightarrow\ & \lambda f.\lambda x.x \\
\equiv\ & 0
\end{aligned}
$$

End

Thanks

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

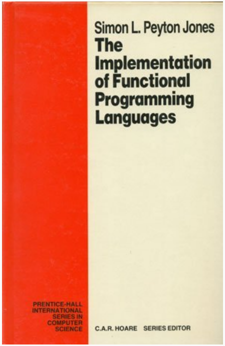
Good Math



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

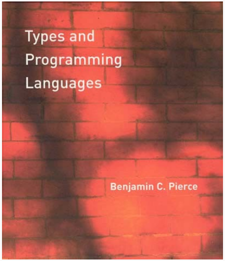
- Easy to understand

The Implementation of Functional Programming Languages



- 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 and Programming Languages



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