

Lambda Calculus

Now you can bring a computer to your tests!

Vincent Macri

William Lyon Mackenzie C.I. Math Club

© Caroline Liu, Vincent Macri, and Samantha Unger, 2018



Table of Contents

1 Introduction

2 One Argument

3 Booleans



What is lambda calculus?

Introduction

- Created by Alonzo Church



What is lambda calculus?

Introduction

- Created by Alonzo Church
- A way of representing **pure** mathematical functions



What is lambda calculus?

Introduction

- Created by Alonzo Church
- A way of representing **pure** mathematical functions
- Can represent any computer program



What is lambda calculus?

Introduction

- Created by Alonzo Church
- A way of representing **pure** mathematical functions
- Can represent any computer program
- Equivalent to Turing machines



$x + 1$

Introduction

In math class, we would define a function that accepts an argument x and outputs $x + 1$ as so:

$$f(x)$$



$x + 1$

Introduction

In math class, we would define a function that accepts an argument x and outputs $x + 1$ as so:

$$f(x) = x + 1$$



$x + 1$

Introduction

In math class, we would define a function that accepts an argument x and outputs $x + 1$ as so:

$$f(x) = x + 1$$

In lambda calculus, we do it like this:

$$\lambda$$


$x + 1$

Introduction

In math class, we would define a function that accepts an argument x and outputs $x + 1$ as so:

$$f(x) = x + 1$$

In lambda calculus, we do it like this:

$$\lambda x$$



In math class, we would define a function that accepts an argument x and outputs $x + 1$ as so:

$$f(x) = x + 1$$

In lambda calculus, we do it like this:

$$\lambda x.$$



$x + 1$

Introduction

In math class, we would define a function that accepts an argument x and outputs $x + 1$ as so:

$$f(x) = x + 1$$

In lambda calculus, we do it like this:

$$\lambda x. x + 1$$



$x + 1$

Introduction

In math class, we would define a function that accepts an argument x and outputs $x + 1$ as so:

$$f(x) = x + 1$$

In lambda calculus, we do it like this:

$$\lambda x.x + 1$$

If we wanted to find $4 + 1$, we could do this:

$$f(4) = 4 + 1 = 5$$



$x + 1$

Introduction

In math class, we would define a function that accepts an argument x and outputs $x + 1$ as so:

$$f(x) = x + 1$$

In lambda calculus, we do it like this:

$$\lambda x.x + 1$$

If we wanted to find $4 + 1$, we could do this:

$$f(4) = 4 + 1 = 5$$

In lambda calculus, we **apply** a value to a function like this:

$$(\lambda x.x + 1)4 = 4 + 1 = 5$$



$x + 1$

Introduction

In math class, we would define a function that accepts an argument x and outputs $x + 1$ as so:

$$f(x) = x + 1$$

In lambda calculus, we do it like this:

$$\lambda x.x + 1$$

If we wanted to find $4 + 1$, we could do this:

$$f(4) = 4 + 1 = 5$$

In lambda calculus, we **apply** a value to a function like this:

$$(\lambda x.x + 1)4 = 4 + 1 = 5$$

You can think of λ as f , and $.$ as $=$.



Table of Contents

1 Introduction

2 One Argument

3 Booleans



Currying

One Argument

How would we define a function that outputs $x + y$ in math class?



How would we define a function that outputs $x + y$ in math class?

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



Currying

One Argument

How would we define a function that outputs $x + y$ in math class?

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

In lambda calculus, functions are only allowed to have **one** parameter.



Currying

One Argument

How would we define a function that outputs $x + y$ in math class?

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

In lambda calculus, functions are only allowed to have **one** parameter.

So, to add two numbers, we have a function output another function, like this:

$$\lambda x. \lambda y. x + y$$



Currying

One Argument

How would we define a function that outputs $x + y$ in math class?

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

In lambda calculus, functions are only allowed to have **one** parameter.

So, to add two numbers, we have a function output another function, like this:

$$\lambda x. \lambda y. x + y$$

And we use it like this:

$$(\lambda x. \lambda y. x + y)(2 \ 3)$$



How would we define a function that outputs $x + y$ in math class?

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

In lambda calculus, functions are only allowed to have **one** parameter.

So, to add two numbers, we have a function output another function, like this:

$$\lambda x. \lambda y. x + y$$

And we use it like this:

$$(\lambda x. \lambda y. x + y)(2 \ 3) = (\lambda y. 2 + y)3$$



How would we define a function that outputs $x + y$ in math class?

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

In lambda calculus, functions are only allowed to have **one** parameter.

So, to add two numbers, we have a function output another function, like this:

$$\lambda x. \lambda y. x + y$$

And we use it like this:

$$(\lambda x. \lambda y. x + y)(2 \ 3) = (\lambda y. 2 + y)3 = 2 + 3 = 5$$



Why one argument?

One Argument

- Very simple



Why one argument?

One Argument

- Very simple
- Very powerful



Why one argument?

One Argument

- Very simple
- Very powerful
- Functions can only have one variable



But I'm lazy

One Argument

Me too!



But I'm lazy

One Argument

Me too!

We have some shortcuts to help us write down lambda calculus expressions, but it's important to remember what they represent, without the shortcuts.



But I'm lazy

One Argument

Me too!

We have some shortcuts to help us write down lambda calculus expressions, but it's important to remember what they represent, without the shortcuts.

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

Can be abbreviated as:

$$\lambda xyz. A$$



But I'm lazy

One Argument

Me too!

We have some shortcuts to help us write down lambda calculus expressions, but it's important to remember what they represent, without the shortcuts.

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

Can be abbreviated as:

$$\lambda xyz. A$$

Also, we assume that we evaluate a function with “multiple” arguments starting with the leftmost parameter.



This is stupid. It just makes everything harder.

One Argument

you're right!



This is stupid. It just makes everything harder.

One Argument

For those examples, you're right!



This is stupid. It just makes everything harder.

One Argument

For those examples, you're right!
Let's get to the fun stuff now!



Table of Contents

1 Introduction

2 One Argument

3 Booleans



Boolean logic

Booleans

Quote

“Any program can be written in lambda calculus.”



Boolean logic

Booleans

Quote

“Any program can be written in lambda calculus.”

— Me, 5 minutes ago



Boolean logic

Booleans

Quote

“Any program can be written in lambda calculus.”

— Me, 5 minutes ago

So, let's bring on the Booleans!



TRUE and FALSE

Booleans

Definition (TRUE)

$$\text{TRUE} = \lambda xy.x$$



TRUE and FALSE

Booleans

Definition (TRUE)

$$\text{TRUE} = \lambda xy.x$$

Definition (FALSE)

$$\text{FALSE} = \lambda xy.y$$



TRUE and FALSE

Booleans

Definition (TRUE)

$$\text{TRUE} = \lambda xy.x$$

Definition (FALSE)

$$\text{FALSE} = \lambda xy.y$$

So TRUE returns the first value, and FALSE returns the second.



TRUE and FALSE

Booleans

Definition (TRUE)

$$\text{TRUE} = \lambda xy.x$$

Definition (FALSE)

$$\text{FALSE} = \lambda xy.y$$

So TRUE returns the first value, and FALSE returns the second.
We will use TRUE and FALSE as shorthand for these definitions.



NOT

Booleans

Definition (NOT)

$$\text{NOT} = \lambda b.b(\text{FALSE } \text{TRUE})$$



NOT

Booleans

Definition (NOT)

$$\text{NOT} = \lambda b.b(\text{FALSE } \text{TRUE})$$

NOT TRUE

$$(\lambda b.b(\text{FALSE } \text{TRUE})) \text{TRUE} =$$



NOT

Booleans

Definition (NOT)

$$\text{NOT} = \lambda b.b(\text{FALSE } \text{TRUE})$$

NOT TRUE

$$(\lambda b.b(\text{FALSE } \text{TRUE})) \text{TRUE} = \text{TRUE}(\text{FALSE } \text{TRUE})$$



NOT

Booleans

Definition (NOT)

$$\text{NOT} = \lambda b.b(\text{FALSE } \text{TRUE})$$

NOT TRUE

$$\begin{aligned} (\lambda b.b(\text{FALSE } \text{TRUE})) \text{TRUE} &= \text{TRUE}(\text{FALSE } \text{TRUE}) \\ &= \lambda xy.x(\text{FALSE } \text{TRUE}) \end{aligned}$$



NOT

Booleans

Definition (NOT)

$$\text{NOT} = \lambda b.b(\text{FALSE } \text{TRUE})$$

NOT TRUE

$$\begin{aligned}(\lambda b.b(\text{FALSE } \text{TRUE})) \text{ TRUE} &= \text{TRUE}(\text{FALSE } \text{TRUE}) \\&= \lambda xy.x(\text{FALSE } \text{TRUE}) \\&= \text{FALSE}\end{aligned}$$



AND

Booleans

Definition (AND)

$$\text{AND} = (\lambda pq.p)(q \text{ } p)$$



AND

Booleans

Definition (AND)

$$\text{AND} = (\lambda pq.p)(q \text{ } p)$$

AND(TRUE FALSE)

$$((\lambda pq.p)(q \text{ } p)) (\text{TRUE FALSE})$$



Definition (AND)

$$\text{AND} = (\lambda pq.p)(q \text{ } p)$$

AND(TRUE FALSE)

$$\begin{aligned} & ((\lambda pq.p)(q \text{ } p)) (\text{TRUE FALSE}) \\ &= \text{TRUE}(\text{FALSE TRUE}) \end{aligned}$$



Definition (AND)

$$\text{AND} = (\lambda pq.p)(q\ p)$$

AND(TRUE FALSE)

$$\begin{aligned} & ((\lambda pq.p)(q\ p))\ (\text{TRUE}\ \text{FALSE}) \\ &= \text{TRUE}(\text{FALSE}\ \text{TRUE}) \\ &= \text{FALSE} \end{aligned}$$



IFTHENELSE

Booleans

Definition (AND)

$$\text{IFTHENELSE} = (\lambda b t f . b)(t \ f)$$

Where b is a Boolean, t is the value to return if $b = \text{TRUE}$, and f is the value to return if $b = \text{FALSE}$.

$\text{IFTHENELSE}(\text{TRUE} \text{ "Math is great" "Math is kool"})$

$((\lambda b t f . b)(t \ f)) (\text{TRUE} \text{ "Math is kool" "Math is lit"})$



IFTHENELSE

Booleans

Definition (AND)

$$\text{IFTHENELSE} = (\lambda b t f. b)(t \ f)$$

Where b is a Boolean, t is the value to return if $b = \text{TRUE}$, and f is the value to return if $b = \text{FALSE}$.

$\text{IFTHENELSE}(\text{TRUE} \text{ "Math is great"} \text{ "Math is kool"})$

$$\begin{aligned} & ((\lambda b t f. b)(t \ f)) (\text{TRUE} \text{ "Math is kool"} \text{ "Math is lit"}) \\ &= ((\lambda b t f. b)(t \ f)) (\text{TRUE} \text{ "Math is kool"} \text{ "Math is lit"}) \end{aligned}$$

