Lambda Calculus

Now you can bring a computer to your tests!

Vincent Macri

William Lyon Mackenzie C.I. Math Club

Table of Contents

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

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

$$f(x)$$

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

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

In 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:

$$\lambda$$

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:

$$\lambda x.$$

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:

$$\lambda x.x + 1$$

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 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:

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

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

Currying
One Argument

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

Currying
One Argument

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:

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

And we use it like this:

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

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:

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

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:

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

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!

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!

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

Boolean logic
Booleans

Quote

"Any program can be written in lambda calculus."

Quote

"Any program can be written in lambda calculus."

— Me, 5 minutes ago

Quote

"Any program can be written in lambda calculus."

— Me, 5 minutes ago

So, let's bring on the Booleans!

We define TRUE as:

$$\mathrm{TRUE} = \lambda x.\lambda y.x = \lambda xy.x$$

We define TRUE as:

$$\mathrm{TRUE} = \lambda x.\lambda y.x = \lambda xy.x$$

And FALSE as:

$$\mathrm{FALSE} = \lambda x.\lambda y.y = \lambda xy.y$$

We define TRUE as:

$$\mathrm{TRUE} = \lambda x.\lambda y.x = \lambda xy.x$$

And FALSE as:

$$\mathrm{FALSE} = \lambda x.\lambda y.y = \lambda xy.y$$

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

True and false

Booleans

We define TRUE as:

$$\mathrm{TRUE} = \lambda x.\lambda y.x = \lambda xy.x$$

And FALSE as:

$$\mathrm{FALSE} = \lambda x.\lambda y.y = \lambda xy.y$$

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

We will use TRUE and FALSE and shorthand for their respective definitions shown here.

NOT can be written in lambda calculus as:

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

NOT can be written in lambda calculus as:

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

NOT TRUE

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

NOT can be written in lambda calculus as:

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

NOT TRUE

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

NOT can be written in lambda calculus as:

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

NOT TRUE

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

NOT can be written in lambda calculus as:

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

NOT TRUE

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