

# Cálculo Lambda

Paulo Torrens

paulotorrens@gnu.org

Departamento de Ciência da Computação  
Centro de Ciências e Tecnologias  
Universidade do Estado de Santa Catarina

2019/02





- O cálculo lambda foi *descoberto* em 1936 pelo matemático Alonzo Church...
  - ...orientador de doutorado de Alan Turing, com o qual trabalhou junto em sua tese de computabilidade
- O cálculo lambda foi criado como um formalismo matemático, a fim de fundamentar a matemática e se estudar o conceito do que é ser computável
- O cálculo lambda é um sistema de reescrita: “executamos” programas reescrevendo termos
- Pode ser entendido como uma linguagem de programação funcional extremamente simples
  - Serve como base para a maioria das linguagens funcionais, incluindo Haskell e CFG/SSA

# Introdução

- O cálculo lambda foi *descoberto* em 1936 pelo matemático Alonzo Church...
  - ...orientador de doutorado de Alan Turing, com o qual trabalhou junto em sua tese de computabilidade
- O cálculo lambda foi criado como um formalismo matemático, a fim de fundamentar a matemática e se estudar o conceito do que é ser computável
- O cálculo lambda é um sistema de reescrita: “executamos” programas reescrevendo termos
- Pode ser entendido como uma linguagem de programação funcional extremamente simples
  - Serve como base para a maioria das linguagens funcionais, incluindo Haskell e CFG/SSA

# Introdução

- O cálculo lambda foi *descoberto* em 1936 pelo matemático Alonzo Church...
  - ...orientador de doutorado de Alan Turing, com o qual trabalhou junto em sua tese de computabilidade
- O cálculo lambda foi criado como um formalismo matemático, a fim de fundamentar a matemática e se estudar o conceito do que é ser computável
- O cálculo lambda é um sistema de reescrita: “executamos” programas reescrevendo termos
- Pode ser entendido como uma linguagem de programação funcional extremamente simples
  - Serve como base para a maioria das linguagens funcionais, incluindo Haskell e CFG/SSA

- O cálculo lambda foi *descoberto* em 1936 pelo matemático Alonzo Church...
  - ...orientador de doutorado de Alan Turing, com o qual trabalhou junto em sua tese de computabilidade
- O cálculo lambda foi criado como um formalismo matemático, a fim de fundamentar a matemática e se estudar o conceito do que é ser computável
- O cálculo lambda é um sistema de reescrita: “executamos” programas reescrevendo termos
- Pode ser entendido como uma linguagem de programação funcional extremamente simples
  - Serve como base para a maioria das linguagens funcionais, incluindo Haskell e CFG/SSA

- O cálculo lambda foi *descoberto* em 1936 pelo matemático Alonzo Church...
  - ...orientador de doutorado de Alan Turing, com o qual trabalhou junto em sua tese de computabilidade
- O cálculo lambda foi criado como um formalismo matemático, a fim de fundamentar a matemática e se estudar o conceito do que é ser computável
- O cálculo lambda é um sistema de reescrita: “executamos” programas reescrevendo termos
- Pode ser entendido como uma linguagem de programação funcional extremamente simples
  - Serve como base para a maioria das linguagens funcionais, incluindo Haskell e CFG/SSA



- O cálculo lambda foi *descoberto* em 1936 pelo matemático Alonzo Church...
  - ...orientador de doutorado de Alan Turing, com o qual trabalhou junto em sua tese de computabilidade
- O cálculo lambda foi criado como um formalismo matemático, a fim de fundamentar a matemática e se estudar o conceito do que é ser computável
- O cálculo lambda é um sistema de reescrita: “executamos” programas reescrevendo termos
- Pode ser entendido como uma linguagem de programação funcional extremamente simples
  - Serve como base para a maioria das linguagens funcionais, incluindo Haskell e CFG/SSA

- Possui uma sintaxe bastante simples:

$$\begin{array}{lcl} e & ::= & x \\ & | & \lambda x. e \\ & | & e e \end{array}$$

- Em outras palavras, um termo pode ser:
  - 1 Uma variável (representada pela meta-variável “x”)
  - 2 Uma abstração, ou seja, uma função com um único argumento e seu corpo (um termo), representada pela letra grega lambda
  - 3 Uma aplicação, ou seja, uma chamada de função (à esquerda, um termo) com um argumento (à direita, um termo)

- Possui uma sintaxe bastante simples:

$$\begin{array}{lcl} e & ::= & x \\ & | & \lambda x. e \\ & | & e e \end{array}$$

- Em outras palavras, um termo pode ser:
  - 1 Uma variável (representada pela meta-variável “x”)
  - 2 Uma abstração, ou seja, uma função com um único argumento e seu corpo (um termo), representada pela letra grega lambda
  - 3 Uma aplicação, ou seja, uma chamada de função (à esquerda, um termo) com um argumento (à direita, um termo)

- Possui uma sintaxe bastante simples:

$$\begin{array}{lcl} e & ::= & x \\ & | & \lambda x. e \\ & | & e e \end{array}$$

- Em outras palavras, um termo pode ser:
  - ① Uma variável (representada pela meta-variável “x”)
  - ② Uma abstração, ou seja, uma função com um único argumento e seu corpo (um termo), representada pela letra grega lambda
  - ③ Uma aplicação, ou seja, uma chamada de função (à esquerda, um termo) com um argumento (à direita, um termo)

- Possui uma sintaxe bastante simples:

$$\begin{array}{lcl} e & ::= & x \\ & | & \lambda x. e \\ & | & e e \end{array}$$

- Em outras palavras, um termo pode ser:
  - ① Uma variável (representada pela meta-variável “x”)
  - ② Uma abstração, ou seja, uma função com um único argumento e seu corpo (um termo), representada pela letra grega lambda
  - ③ Uma aplicação, ou seja, uma chamada de função (à esquerda, um termo) com um argumento (à direita, um termo)

- É útil lembrarmos algumas convenções sobre a notação de termos lambda:

- Aplicação associa à esquerda, ou seja:

$$a \ b \ c = (a \ b) \ c$$

$$a \ b \ c \ d = ((a \ b) \ c) \ d$$

- Assumimos que uma abstração lambda se estende até onde for possível à direita, ou seja:

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

- Exemplos:

- $\lambda x. x$
  - $a \ b \ c$
  - $a \ (b \ c)$
  - $\lambda f. \lambda x. f \ x$

- É útil lembrarmos algumas convenções sobre a notação de termos lambda:

- Aplicação associa à esquerda, ou seja:

$$a \ b \ c = (a \ b) \ c$$

$$a \ b \ c \ d = ((a \ b) \ c) \ d$$

- Assumimos que uma abstração lambda se estende até onde for possível à direita, ou seja:

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

- Exemplos:

- $\lambda x. x$
  - $a \ b \ c$
  - $a \ (b \ c)$
  - $\lambda f. \lambda x. f \ x$

- É útil lembrarmos algumas convenções sobre a notação de termos lambda:

- Aplicação associa à esquerda, ou seja:

$$a \ b \ c = (a \ b) \ c$$

$$a \ b \ c \ d = ((a \ b) \ c) \ d$$

- Assumimos que uma abstração lambda se estende até onde for possível à direita, ou seja:

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

- Exemplos:

- $\lambda x. x$
  - $a \ b \ c$
  - $a \ (b \ c)$
  - $\lambda f. \lambda x. f \ x$



- É útil lembrarmos algumas convenções sobre a notação de termos lambda:

- Aplicação associa à esquerda, ou seja:

$$a \ b \ c = (a \ b) \ c$$

$$a \ b \ c \ d = ((a \ b) \ c) \ d$$

- Assumimos que uma abstração lambda se estende até onde for possível à direita, ou seja:

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

- Exemplos:

- $\lambda x. x$
  - $a \ b \ c$
  - $a \ (b \ c)$
  - $\lambda f. \lambda x. f \ x$

- É útil lembrarmos algumas convenções sobre a notação de termos lambda:

- Aplicação associa à esquerda, ou seja:

$$a \ b \ c = (a \ b) \ c$$

$$a \ b \ c \ d = ((a \ b) \ c) \ d$$

- Assumimos que uma abstração lambda se estende até onde for possível à direita, ou seja:

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

- Exemplos:

- $\lambda x. x$
  - $a \ b \ c$
  - $a \ (b \ c)$
  - $\lambda f. \lambda x. f \ x$

- É útil lembrarmos algumas convenções sobre a notação de termos lambda:

- Aplicação associa à esquerda, ou seja:

$$a \ b \ c = (a \ b) \ c$$

$$a \ b \ c \ d = ((a \ b) \ c) \ d$$

- Assumimos que uma abstração lambda se estende até onde for possível à direita, ou seja:

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

- Exemplos:

- $\lambda x. x$
  - $a \ b \ c$
  - $a \ (b \ c)$
  - $\lambda f. \lambda x. f \ x$

- É útil lembrarmos algumas convenções sobre a notação de termos lambda:

- Aplicação associa à esquerda, ou seja:

$$a \ b \ c = (a \ b) \ c$$

$$a \ b \ c \ d = ((a \ b) \ c) \ d$$

- Assumimos que uma abstração lambda se estende até onde for possível à direita, ou seja:

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

- Exemplos:

- $\lambda x. x$
  - $a \ b \ c$
  - $a \ (b \ c)$
  - $\lambda f. \lambda x. f \ x$

- O cálculo lambda é composto unicamente por funções
  - Uma abstração lambda é uma função que recebe uma função como argumento e retorna uma função como resultado
- Existe um conceito de escopo: uma variável é considerada ligada à abstração lambda mais próxima que contém o mesmo nome como parâmetro; por exemplo:
  - $x (\lambda x. x y)$
  - $\lambda x. \lambda y. x$
  - $\lambda x. \lambda y. y$
  - $\lambda x. \lambda x. x$
  - $\lambda x. y (\lambda y. a y x)$
- Uma variável que não está ligada é chamada de livre
- Importante notar que, num termo  $\lambda x. e$ , a abstração lambda irá ligar todas as variáveis  $x$  que estavam livres em  $e$

- O cálculo lambda é composto unicamente por funções
  - Uma abstração lambda é uma função que recebe uma função como argumento e retorna uma função como resultado
- Existe um conceito de escopo: uma variável é considerada ligada à abstração lambda mais próxima que contém o mesmo nome como parâmetro; por exemplo:
  - $x (\lambda x. x y)$
  - $\lambda x. \lambda y. x$
  - $\lambda x. \lambda y. y$
  - $\lambda x. \lambda x. x$
  - $\lambda x. y (\lambda y. a y x)$
- Uma variável que não está ligada é chamada de livre
- Importante notar que, num termo  $\lambda x. e$ , a abstração lambda irá ligar todas as variáveis  $x$  que estavam livres em  $e$

- O cálculo lambda é composto unicamente por funções
  - Uma abstração lambda é uma função que recebe uma função como argumento e retorna uma função como resultado
- Existe um conceito de escopo: uma variável é considerada ligada à abstração lambda mais próxima que contém o mesmo nome como parâmetro; por exemplo:
  - $x (\lambda x. x y)$
  - $\lambda x. \lambda y. x$
  - $\lambda x. \lambda y. y$
  - $\lambda x. \lambda x. x$
  - $\lambda x. y (\lambda y. a y x)$
- Uma variável que não está ligada é chamada de livre
- Importante notar que, num termo  $\lambda x. e$ , a abstração lambda irá ligar todas as variáveis  $x$  que estavam livres em  $e$

- O cálculo lambda é composto unicamente por funções
  - Uma abstração lambda é uma função que recebe uma função como argumento e retorna uma função como resultado
- Existe um conceito de escopo: uma variável é considerada ligada à abstração lambda mais próxima que contém o mesmo nome como parâmetro; por exemplo:
  - $x (\lambda x. x y)$
  - $\lambda x. \lambda y. x$
  - $\lambda x. \lambda y. y$
  - $\lambda x. \lambda x. x$
  - $\lambda x. y (\lambda y. a y x)$
- Uma variável que não está ligada é chamada de livre
- Importante notar que, num termo  $\lambda x. e$ , a abstração lambda irá ligar todas as variáveis  $x$  que estavam livres em  $e$



- O cálculo lambda é composto unicamente por funções
  - Uma abstração lambda é uma função que recebe uma função como argumento e retorna uma função como resultado
- Existe um conceito de escopo: uma variável é considerada ligada à abstração lambda mais próxima que contém o mesmo nome como parâmetro; por exemplo:
  - $x (\lambda x. x y)$
  - $\lambda x. \lambda y. x$
  - $\lambda x. \lambda y. y$
  - $\lambda x. \lambda x. x$
  - $\lambda x. y (\lambda y. a y x)$
- Uma variável que não está ligada é chamada de livre
- Importante notar que, num termo  $\lambda x. e$ , a abstração lambda irá ligar todas as variáveis  $x$  que estavam livres em  $e$

- O cálculo lambda é composto unicamente por funções
  - Uma abstração lambda é uma função que recebe uma função como argumento e retorna uma função como resultado
- Existe um conceito de escopo: uma variável é considerada ligada à abstração lambda mais próxima que contém o mesmo nome como parâmetro; por exemplo:
  - $x (\lambda x. x y)$
  - $\lambda x. \lambda y. x$
  - $\lambda x. \lambda y. y$
  - $\lambda x. \lambda x. x$
  - $\lambda x. y (\lambda y. a y x)$
- Uma variável que não está ligada é chamada de livre
- Importante notar que, num termo  $\lambda x. e$ , a abstração lambda irá ligar todas as variáveis  $x$  que estavam livres em  $e$

- O cálculo lambda é composto unicamente por funções
  - Uma abstração lambda é uma função que recebe uma função como argumento e retorna uma função como resultado
- Existe um conceito de escopo: uma variável é considerada ligada à abstração lambda mais próxima que contém o mesmo nome como parâmetro; por exemplo:
  - $x (\lambda x. x y)$
  - $\lambda x. \lambda y. x$
  - $\lambda x. \lambda y. y$
  - $\lambda x. \lambda x. x$
  - $\lambda x. y (\lambda y. a y x)$
- Uma variável que não está ligada é chamada de livre
- Importante notar que, num termo  $\lambda x. e$ , a abstração lambda irá ligar todas as variáveis  $x$  que estavam livres em  $e$

- O cálculo lambda é composto unicamente por funções
  - Uma abstração lambda é uma função que recebe uma função como argumento e retorna uma função como resultado
- Existe um conceito de escopo: uma variável é considerada ligada à abstração lambda mais próxima que contém o mesmo nome como parâmetro; por exemplo:
  - $x (\lambda x. x y)$
  - $\lambda x. \lambda y. x$
  - $\lambda x. \lambda y. y$
  - $\lambda x. \lambda x. x$
  - $\lambda x. y (\lambda y. a y x)$
- Uma variável que não está ligada é chamada de livre
- Importante notar que, num termo  $\lambda x. e$ , a abstração lambda irá ligar todas as variáveis  $x$  que estavam livres em  $e$

- O cálculo lambda é composto unicamente por funções
  - Uma abstração lambda é uma função que recebe uma função como argumento e retorna uma função como resultado
- Existe um conceito de escopo: uma variável é considerada ligada à abstração lambda mais próxima que contém o mesmo nome como parâmetro; por exemplo:
  - $x (\lambda x. x y)$
  - $\lambda x. \lambda y. x$
  - $\lambda x. \lambda y. y$
  - $\lambda x. \lambda x. x$
  - $\lambda x. y (\lambda y. a y x)$
- Uma variável que não está ligada é chamada de livre
- Importante notar que, num termo  $\lambda x. e$ , a abstração lambda irá ligar todas as variáveis  $x$  que estavam livres em  $e$

- O cálculo lambda é composto unicamente por funções
  - Uma abstração lambda é uma função que recebe uma função como argumento e retorna uma função como resultado
- Existe um conceito de escopo: uma variável é considerada ligada à abstração lambda mais próxima que contém o mesmo nome como parâmetro; por exemplo:
  - $x (\lambda x. x y)$
  - $\lambda x. \lambda y. x$
  - $\lambda x. \lambda y. y$
  - $\lambda x. \lambda x. x$
  - $\lambda x. y (\lambda y. a y x)$
- Uma variável que não está ligada é chamada de livre
- Importante notar que, num termo  $\lambda x. e$ , a abstração lambda irá ligar todas as variáveis  $x$  que estavam livres em  $e$

- Dois termos são considerados  $\alpha$ -equivalentes se a renomeação de parâmetros (junto aos seus usos ligados) pode tornar dois termos idênticos; por exemplo:
  - $\lambda x.x$
  - $a (\lambda a.b a)$
- Um conceito importante no cálculo lambda é o conceito de substituição, usado na reescrita de termos
- Uma substituição, anotada como  $a[b/x]$ , representa a reescrita do termo  $a$ , com todas as ocorrências da variável livre  $x$  substituídas por  $b$ ... por exemplo:
  - $x[y/x]$
  - $(x (\lambda z.z x))[y/x]$
  - $(\lambda x.x)[y/x]$
  - $(a b c b)[\lambda x.x/b]$

- Dois termos são considerados  $\alpha$ -equivalentes se a renomeação de parâmetros (junto aos seus usos ligados) pode tornar dois termos idênticos; por exemplo:
  - $\lambda x.x$
  - $a (\lambda a.b a)$
- Um conceito importante no cálculo lambda é o conceito de substituição, usado na reescrita de termos
- Uma substituição, anotada como  $a[b/x]$ , representa a reescrita do termo  $a$ , com todas as ocorrências da variável livre  $x$  substituídas por  $b$ ... por exemplo:
  - $x[y/x]$
  - $(x (\lambda z.z x))[y/x]$
  - $(\lambda x.x)[y/x]$
  - $(a b c b)[\lambda x.x/b]$



- Dois termos são considerados  $\alpha$ -equivalentes se a renomeação de parâmetros (junto aos seus usos ligados) pode tornar dois termos idênticos; por exemplo:
  - $\lambda x.x = \lambda y.y$
  - $a (\lambda a.b a)$
- Um conceito importante no cálculo lambda é o conceito de substituição, usado na reescrita de termos
- Uma substituição, anotada como  $a[b/x]$ , representa a reescrita do termo  $a$ , com todas as ocorrências da variável livre  $x$  substituídas por  $b$ ... por exemplo:
  - $x[y/x]$
  - $(x (\lambda z.z x))[y/x]$
  - $(\lambda x.x)[y/x]$
  - $(a b c b)[\lambda x.x/b]$

- Dois termos são considerados  $\alpha$ -equivalentes se a renomeação de parâmetros (junto aos seus usos ligados) pode tornar dois termos idênticos; por exemplo:
  - $\lambda x.x = \lambda y.y$
  - $a (\lambda a.b a)$
- Um conceito importante no cálculo lambda é o conceito de substituição, usado na reescrita de termos
- Uma substituição, anotada como  $a[b/x]$ , representa a reescrita do termo  $a$ , com todas as ocorrências da variável livre  $x$  substituídas por  $b$ ... por exemplo:
  - $x[y/x]$
  - $(x (\lambda z.z x))[y/x]$
  - $(\lambda x.x)[y/x]$
  - $(a b c b)[\lambda x.x/b]$

- Dois termos são considerados  $\alpha$ -equivalentes se a renomeação de parâmetros (junto aos seus usos ligados) pode tornar dois termos idênticos; por exemplo:
  - $\lambda x.x = \lambda y.y$
  - $a (\lambda a.b a) = a (\lambda x.b x)$
- Um conceito importante no cálculo lambda é o conceito de substituição, usado na reescrita de termos
- Uma substituição, anotada como  $a[b/x]$ , representa a reescrita do termo  $a$ , com todas as ocorrências da variável livre  $x$  substituídas por  $b$ ... por exemplo:
  - $x[y/x]$
  - $(x (\lambda z.z x))[y/x]$
  - $(\lambda x.x)[y/x]$
  - $(a b c b)[\lambda x.x/b]$

- Dois termos são considerados  $\alpha$ -equivalentes se a renomeação de parâmetros (junto aos seus usos ligados) pode tornar dois termos idênticos; por exemplo:
  - $\lambda x.x = \lambda y.y$
  - $a (\lambda a.b a) = a (\lambda x.b x)$
- Um conceito importante no cálculo lambda é o conceito de substituição, usado na reescrita de termos
- Uma substituição, anotada como  $a[b/x]$ , representa a reescrita do termo  $a$ , com todas as ocorrências da variável livre  $x$  substituídas por  $b$ ... por exemplo:
  - $x[y/x]$
  - $(x (\lambda z.z x))[y/x]$
  - $(\lambda x.x)[y/x]$
  - $(a b c b)[\lambda x.x/b]$

- Dois termos são considerados  $\alpha$ -equivalentes se a renomeação de parâmetros (junto aos seus usos ligados) pode tornar dois termos idênticos; por exemplo:
  - $\lambda x.x = \lambda y.y$
  - $a (\lambda a.b a) = a (\lambda x.b x)$
- Um conceito importante no cálculo lambda é o conceito de substituição, usado na reescrita de termos
- Uma substituição, anotada como  $a[b/x]$ , representa a reescrita do termo  $a$ , com todas as ocorrências da variável livre  $x$  substituídas por  $b$ ... por exemplo:
  - $x[y/x]$
  - $(x (\lambda z.z x))[y/x]$
  - $(\lambda x.x)[y/x]$
  - $(a b c b)[\lambda x.x/b]$

- Dois termos são considerados  $\alpha$ -equivalentes se a renomeação de parâmetros (junto aos seus usos ligados) pode tornar dois termos idênticos; por exemplo:
  - $\lambda x.x = \lambda y.y$
  - $a (\lambda a.b a) = a (\lambda x.b x)$
- Um conceito importante no cálculo lambda é o conceito de substituição, usado na reescrita de termos
- Uma substituição, anotada como  $a[b/x]$ , representa a reescrita do termo  $a$ , com todas as ocorrências da variável livre  $x$  substituídas por  $b$ ... por exemplo:
  - $x[y/x]$
  - $(x (\lambda z.z x))[y/x]$
  - $(\lambda x.x)[y/x]$
  - $(a b c b)[\lambda x.x/b]$

- Dois termos são considerados  $\alpha$ -equivalentes se a renomeação de parâmetros (junto aos seus usos ligados) pode tornar dois termos idênticos; por exemplo:
  - $\lambda x.x = \lambda y.y$
  - $a (\lambda a.b a) = a (\lambda x.b x)$
- Um conceito importante no cálculo lambda é o conceito de substituição, usado na reescrita de termos
- Uma substituição, anotada como  $a[b/x]$ , representa a reescrita do termo  $a$ , com todas as ocorrências da variável livre  $x$  substituídas por  $b$ ... por exemplo:
  - $x[y/x] = y$
  - $(x (\lambda z.z x))[y/x]$
  - $(\lambda x.x)[y/x]$
  - $(a b c b)[\lambda x.x/b]$

- Dois termos são considerados  $\alpha$ -equivalentes se a renomeação de parâmetros (junto aos seus usos ligados) pode tornar dois termos idênticos; por exemplo:
  - $\lambda x.x = \lambda y.y$
  - $a (\lambda a.b a) = a (\lambda x.b x)$
- Um conceito importante no cálculo lambda é o conceito de substituição, usado na reescrita de termos
- Uma substituição, anotada como  $a[b/x]$ , representa a reescrita do termo  $a$ , com todas as ocorrências da variável livre  $x$  substituídas por  $b$ ... por exemplo:
  - $x[y/x] = y$
  - $(x (\lambda z.z x))[y/x]$
  - $(\lambda x.x)[y/x]$
  - $(a b c b)[\lambda x.x/b]$



- Dois termos são considerados  $\alpha$ -equivalentes se a renomeação de parâmetros (junto aos seus usos ligados) pode tornar dois termos idênticos; por exemplo:
  - $\lambda x.x = \lambda y.y$
  - $a (\lambda a.b a) = a (\lambda x.b x)$
- Um conceito importante no cálculo lambda é o conceito de substituição, usado na reescrita de termos
- Uma substituição, anotada como  $a[b/x]$ , representa a reescrita do termo  $a$ , com todas as ocorrências da variável livre  $x$  substituídas por  $b$ ... por exemplo:
  - $x[y/x] = y$
  - $(x (\lambda z.z x))[y/x] = y (\lambda z.z y)$
  - $(\lambda x.x)[y/x]$
  - $(a b c b)[\lambda x.x/b]$

- Dois termos são considerados  $\alpha$ -equivalentes se a renomeação de parâmetros (junto aos seus usos ligados) pode tornar dois termos idênticos; por exemplo:
  - $\lambda x.x = \lambda y.y$
  - $a (\lambda a.b a) = a (\lambda x.b x)$
- Um conceito importante no cálculo lambda é o conceito de substituição, usado na reescrita de termos
- Uma substituição, anotada como  $a[b/x]$ , representa a reescrita do termo  $a$ , com todas as ocorrências da variável livre  $x$  substituídas por  $b$ ... por exemplo:
  - $x[y/x] = y$
  - $(x (\lambda z.z x))[y/x] = y (\lambda z.z y)$
  - $(\lambda x.x)[y/x]$
  - $(a b c b)[\lambda x.x/b]$

- Dois termos são considerados  $\alpha$ -equivalentes se a renomeação de parâmetros (junto aos seus usos ligados) pode tornar dois termos idênticos; por exemplo:
  - $\lambda x.x = \lambda y.y$
  - $a (\lambda a.b a) = a (\lambda x.b x)$
- Um conceito importante no cálculo lambda é o conceito de substituição, usado na reescrita de termos
- Uma substituição, anotada como  $a[b/x]$ , representa a reescrita do termo  $a$ , com todas as ocorrências da variável livre  $x$  substituídas por  $b$ ... por exemplo:
  - $x[y/x] = y$
  - $(x (\lambda z.z x))[y/x] = y (\lambda z.z y)$
  - $(\lambda x.x)[y/x] = (\lambda x.x)$
  - $(a b c b)[\lambda x.x/b]$

- Dois termos são considerados  $\alpha$ -equivalentes se a renomeação de parâmetros (junto aos seus usos ligados) pode tornar dois termos idênticos; por exemplo:
  - $\lambda x.x = \lambda y.y$
  - $a (\lambda a.b a) = a (\lambda x.b x)$
- Um conceito importante no cálculo lambda é o conceito de substituição, usado na reescrita de termos
- Uma substituição, anotada como  $a[b/x]$ , representa a reescrita do termo  $a$ , com todas as ocorrências da variável livre  $x$  substituídas por  $b$ ... por exemplo:
  - $x[y/x] = y$
  - $(x (\lambda z.z x))[y/x] = y (\lambda z.z y)$
  - $(\lambda x.x)[y/x] = (\lambda x.x)$  A variável  $x$  não era livre!
  - $(a b c b)[\lambda x.x/b]$

- Dois termos são considerados  $\alpha$ -equivalentes se a renomeação de parâmetros (junto aos seus usos ligados) pode tornar dois termos idênticos; por exemplo:
  - $\lambda x.x = \lambda y.y$
  - $a (\lambda a.b a) = a (\lambda x.b x)$
- Um conceito importante no cálculo lambda é o conceito de substituição, usado na reescrita de termos
- Uma substituição, anotada como  $a[b/x]$ , representa a reescrita do termo  $a$ , com todas as ocorrências da variável livre  $x$  substituídas por  $b$ ... por exemplo:
  - $x[y/x] = y$
  - $(x (\lambda z.z x))[y/x] = y (\lambda z.z y)$
  - $(\lambda x.x)[y/x] = (\lambda x.x)$  A variável  $x$  não era livre!
  - $(a b c b)[\lambda x.x/b]$

- Dois termos são considerados  $\alpha$ -equivalentes se a renomeação de parâmetros (junto aos seus usos ligados) pode tornar dois termos idênticos; por exemplo:
  - $\lambda x.x = \lambda y.y$
  - $a (\lambda a.b a) = a (\lambda x.b x)$
- Um conceito importante no cálculo lambda é o conceito de substituição, usado na reescrita de termos
- Uma substituição, anotada como  $a[b/x]$ , representa a reescrita do termo  $a$ , com todas as ocorrências da variável livre  $x$  substituídas por  $b$ ... por exemplo:
  - $x[y/x] = y$
  - $(x (\lambda z.z x))[y/x] = y (\lambda z.z y)$
  - $(\lambda x.x)[y/x] = (\lambda x.x)$  A variável  $x$  não era livre!
  - $(a b c b)[\lambda x.x/b] = a (\lambda x.x) c (\lambda x.x)$

- Para “executarmos” o termo, usamos a chamada redução  $\beta$ :

$$\underbrace{(\lambda x.e) y}_{\beta\text{-redex}} \longrightarrow \overbrace{e[y/x]}^{\text{substituição}}$$

- Itens sujeitos à redução são chamados de  $\beta$ -redexes, que são apenas uma aplicação cujo subtermo à esquerda é uma abstração
- Em outras palavras, devemos reescrever o termo à esquerda removendo a abstração lambda, e substituindo todos os casos ligados de seu parâmetro com o argumento fornecido

- Para “executarmos” o termo, usamos a chamada redução  $\beta$ :

$$\underbrace{(\lambda x.e) y}_{\beta\text{-redex}} \longrightarrow \overbrace{e[y/x]}^{\text{substituição}}$$

- Itens sujeitos à redução são chamados de  $\beta$ -redexes, que são apenas uma aplicação cujo subtermo à esquerda é uma abstração
- Em outras palavras, devemos reescrever o termo à esquerda removendo a abstração lambda, e substituindo todos os casos ligados de seu parâmetro com o argumento fornecido



- Para “executarmos” o termo, usamos a chamada redução  $\beta$ :

$$\underbrace{(\lambda x.e) y}_{\beta\text{-redex}} \longrightarrow \overbrace{e[y/x]}^{\text{substituição}}$$

- Itens sujeitos à redução são chamados de  $\beta$ -redexes, que são apenas uma aplicação cujo subtermo à esquerda é uma abstração
- Em outras palavras, devemos reescrever o termo à esquerda removendo a abstração lambda, e substituindo todos os casos ligados de seu parâmetro com o argumento fornecido

- Alguns exemplos de reduções  $\beta$ :
  - $(\lambda x.x) a$
  - $(\lambda x.y) a$
- Assumindo que podemos usar números e operações:
  - $(\lambda x.x * x) 5$
  - $(\lambda x.\lambda y.x + y) 10 20$

- Alguns exemplos de reduções  $\beta$ :
  - $(\lambda x.x) a$
  - $(\lambda x.y) a$
- Assumindo que podemos usar números e operações:
  - $(\lambda x.x * x) 5$
  - $(\lambda x.\lambda y.x + y) 10 20$

- Alguns exemplos de reduções  $\beta$ :
  - $(\lambda x.x) a \rightarrow a$
  - $(\lambda x.y) a$
- Assumindo que podemos usar números e operações:
  - $(\lambda x.x * x) 5$
  - $(\lambda x.\lambda y.x + y) 10 20$

- Alguns exemplos de reduções  $\beta$ :
  - $(\lambda x.x) a \rightarrow a$
  - $(\lambda x.y) a$
- Assumindo que podemos usar números e operações:
  - $(\lambda x.x * x) 5$
  - $(\lambda x.\lambda y.x + y) 10 20$

- Alguns exemplos de reduções  $\beta$ :
  - $(\lambda x.x) a \rightarrow a$
  - $(\lambda x.y) a \rightarrow y$
- Assumindo que podemos usar números e operações:
  - $(\lambda x.x * x) 5$
  - $(\lambda x.\lambda y.x + y) 10 20$

- Alguns exemplos de reduções  $\beta$ :
  - $(\lambda x.x) a \rightarrow a$
  - $(\lambda x.y) a \rightarrow y$
- Assumindo que podemos usar números e operações:
  - $(\lambda x.x * x) 5$
  - $(\lambda x.\lambda y.x + y) 10 20$

- Alguns exemplos de reduções  $\beta$ :
  - $(\lambda x.x) a \rightarrow a$
  - $(\lambda x.y) a \rightarrow y$
- Assumindo que podemos usar números e operações:
  - $(\lambda x.x * x) 5$   
 $\rightarrow 5 * 5$
  - $(\lambda x.\lambda y.x + y) 10 20$



- Alguns exemplos de reduções  $\beta$ :
  - $(\lambda x.x) a \rightarrow a$
  - $(\lambda x.y) a \rightarrow y$
- Assumindo que podemos usar números e operações:
  - $(\lambda x.x * x) 5$   
 $\rightarrow 5 * 5$   
 $\rightarrow 25$
  - $(\lambda x.\lambda y.x + y) 10 20$

- Alguns exemplos de reduções  $\beta$ :
  - $(\lambda x.x) a \rightarrow a$
  - $(\lambda x.y) a \rightarrow y$
- Assumindo que podemos usar números e operações:
  - $(\lambda x.x * x) 5$   
 $\rightarrow 5 * 5$   
 $\rightarrow 25$
  - $(\lambda x.\lambda y.x + y) 10 20$

- Alguns exemplos de reduções  $\beta$ :
  - $(\lambda x.x) a \rightarrow a$
  - $(\lambda x.y) a \rightarrow y$
- Assumindo que podemos usar números e operações:
  - $(\lambda x.x * x) 5$   
 $\rightarrow 5 * 5$   
 $\rightarrow 25$
  - $(\lambda x.\lambda y.x + y) 10 20$   
 $\rightarrow (\lambda y.10 + y) 20$

- Alguns exemplos de reduções  $\beta$ :
  - $(\lambda x.x) a \rightarrow a$
  - $(\lambda x.y) a \rightarrow y$
- Assumindo que podemos usar números e operações:
  - $(\lambda x.x * x) 5$   
 $\rightarrow 5 * 5$   
 $\rightarrow 25$
  - $(\lambda x.\lambda y.x + y) 10 20$   
 $\rightarrow (\lambda y.10 + y) 20$   
 $\rightarrow 10 + 20$

- Alguns exemplos de reduções  $\beta$ :
  - $(\lambda x.x) a \rightarrow a$
  - $(\lambda x.y) a \rightarrow y$
- Assumindo que podemos usar números e operações:
  - $(\lambda x.x * x) 5$   
 $\rightarrow 5 * 5$   
 $\rightarrow 25$
  - $(\lambda x.\lambda y.x + y) 10 20$   
 $\rightarrow (\lambda y.10 + y) 20$   
 $\rightarrow 10 + 20$   
 $\rightarrow 30$