Paradigmas de Programação

Prof. Maicon R. Zatelli

Python - Lambda Calculus

Universidade Federal de Santa Catarina

Florianópolis - Brasil

2018/1

Cálculo Lambda

Alonso Church (1936): usando lógica combinatória desenvolve o cálculo lambda e usando cálculo-λ prova a computabilidade.

Cálculo Lambda

O calculo-λ consiste em regras simples de transformação (substituição de argumentos ou variáveis) que pode representar qualquer função computável por meio deste formalismo.

Sintaxe do Cálculo Lambda

```
E ::- ID
E ::- λID.E
E ::- E E
E ::- (E)
```

* Cada ID é uma variável.

Pode ser considerada como uma linguagem universal de programação.

Cálculo Lambda

Exemplos

$\texttt{x}$

$\lambda\texttt{x.x}$

$\texttt{x y}$

$\lambda\texttt{x.yz}$

$\texttt{x}\lambda\texttt{y.(x(yz))}$

Cálculo Lambda

Alguns problemas... **ambiguidade**.

`a b c = (a b) c` ou `a b c = a (b c)`?

Removendo a ambiguidade: `E ::- E E` é associativo à esquerda

`a b c = (a b) c`

`a b c d = ((a b) c) d`

Cálculo Lambda

Alguns problemas... **ambiguidade**.

λ`x.xy = `λ`x.(xy)` ou λ`x.xy = (`λ`x.x) (y)`?

Removendo a ambiguidade: λ`ID.E` expande o máximo à direita o quanto possível a partir de λ`ID`.

λ`x.xy = `λ`x.(xy)`

λ`x.`λ`x.x = `λ`x.(`λ`x.x)`

λ`a.`λ`b.`λ`c.abc = `λ`a.(`λ`b.(`λ`c.(ab)c))`

Cálculo Lambda

Um pouco de semântica...

$E \; ::- \; \lambda ID.E$

É chamada de **abstração**

ID é a variável da abstração

E é o corpo da abstração

* Definindo uma função, assim

$E \; ::- \; E \; E$

É chamada de **aplicação** (da função)

* Chamando uma função

Cálculo Lambda

$\lambda ID.E$ é a definição de uma função anônima.

- ID é o parâmetro da função
- E é o corpo da função

$E_1\ E_2$ é a chamada da função, onde

- E_1 é a função
- E_2 é o parâmetro da função

Cálculo Lambda

- $\lambda x. + \ x\ 1$
  - Representa uma função que adiciona 1 ao seu argumento
- $(\lambda x. + \ x\ 1)2$
  - Representa a chamada da função substituindo x por 2, o que irá reduzir a função para $(+\ 2\ 1) = 3$

Cálculo Lambda

- $\lambda x. + \ x \ 1$
  - Representa uma função que adiciona 1 ao seu argumento
- $(\lambda x. + \ x \ 1)2$
  - Representa a chamada da função substituindo x por 2, o que irá reduzir a função para $(+ \ 2 \ 1) = 3$

Como usar mais de um argumento? $(+ \ 1 \ 2)$ É uma função lambda válida?

Cálculo Lambda

- $\lambda x. + \; x \; 1$
  - Representa uma função que adiciona 1 ao seu argumento
- $(\lambda x. + \; x \; 1)2$
  - Representa a chamada da função substituindo x por 2, o que irá reduzir a função para $(+ \; 2 \; 1) = 3$

Como usar mais de um argumento? $(+ \; 1 \; 2)$ É uma função lambda válida?

- Criar uma função que recebe um argumento e retorna uma função que recebe um argumento e adiciona ele ao número original. Exemplo: uma função que recebe um argumento x e retorna uma função que recebe um argumento y e soma x com y.
  - $\lambda x.(\lambda y((+x)y))$

Cálculo Lambda

- $\lambda x.(\lambda y((+x)y))$

Cálculo Lambda

- $\lambda x.(\lambda y((+x)y))$
  - $(\lambda x.(\lambda y((+x)y)))1$

Cálculo Lambda

- $\lambda x.(\lambda y((+x)y))$
  - $(\lambda x.(\lambda y((+x)y)))1$
  - $\lambda y((+1)y)$

Cálculo Lambda

- $\lambda x.(\lambda y((+x)y))$
  - $(\lambda x.(\lambda y((+x)y)))1$
  - $\lambda y((+1)y)$
  - $(\lambda y((+1)y))2$, retorna a função $(+1)$ aplicada ao argumento y

Cálculo Lambda

- $\lambda x.(\lambda y((+x)y))$
  - $(\lambda x.(\lambda y((+x)y)))1$
  - $\lambda y((+1)y)$
  - $(\lambda y((+1)y))2$, retorna a função $(+1)$ aplicada ao argumento y
  - $((+1)2)$

Cálculo Lambda

- $\lambda x.(\lambda y((+x)y))$
  - $(\lambda x.(\lambda y((+x)y)))1$
  - $\lambda y((+1)y)$
  - $(\lambda y((+1)y))2$, retorna a função $(+1)$ aplicada ao argumento y
  - $((+1)2)$
  - $(+1)2 = 3$, aplica a função $+1$ sobre 2

Cálculo Lambda

- $\lambda x.(\lambda y((+x)y))$
  - $(\lambda x.(\lambda y((+x)y)))1$
  - $\lambda y((+1)y)$
  - $(\lambda y((+1)y))2$, retorna a função $(+1)$ aplicada ao argumento y
  - $((+1)2)$
  - $(+1)2 = 3$, aplica a função $+1$ sobre 2

- $\lambda x.(\lambda y((+x)y))$

Cálculo Lambda

- $\lambda x.(\lambda y((+x)y))$
  - $(\lambda x.(\lambda y((+x)y)))1$
  - $\lambda y((+1)y)$
  - $(\lambda y((+1)y))2$, retorna a função $(+1)$ aplicada ao argumento y
  - $((+1)2)$
  - $(+1)2 = 3$, aplica a função $+1$ sobre 2

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

Cálculo Lambda

- $\lambda x.(\lambda y((+x)y))$
  - $(\lambda x.(\lambda y((+x)y)))1$
  - $\lambda y((+1)y)$
  - $(\lambda y((+1)y))2$, retorna a função $(+1)$ aplicada ao argumento y
  - $((+1)2)$
  - $(+1)2 = 3$, aplica a função $+1$ sobre 2

- $\lambda x.(\lambda y((+x)y))$
  - $(\lambda x.(\lambda y((+x)y)))2\ 3$
  - $((\lambda x.(\lambda y((+x)y)))2)3$, removendo a ambiguidade

Cálculo Lambda

- $\lambda x.(\lambda y((+x)y))$
  - $(\lambda x.(\lambda y((+x)y)))1$
  - $\lambda y((+1)y)$
  - $(\lambda y((+1)y))2$, retorna a função $(+1)$ aplicada ao argumento y
  - $((+1)2)$
  - $(+1)2 = 3$, aplica a função $+1$ sobre 2

- $\lambda x.(\lambda y((+x)y))$
  - $(\lambda x.(\lambda y((+x)y)))2\ 3$
  - $((\lambda x.(\lambda y((+x)y)))2)3$, removendo a ambiguidade
  - $(\lambda y((+2)y))3$

Cálculo Lambda

- $\lambda x.(\lambda y((+x)y))$
  - $(\lambda x.(\lambda y((+x)y)))1$
  - $\lambda y((+1)y)$
  - $(\lambda y((+1)y))2$, retorna a função $(+1)$ aplicada ao argumento y
  - $((+1)2)$
  - $(+1)2 = 3$, aplica a função $+1$ sobre 2

- $\lambda x.(\lambda y((+x)y))$
  - $(\lambda x.(\lambda y((+x)y)))2\ 3$
  - $((\lambda x.(\lambda y((+x)y)))2)3$, removendo a ambiguidade
  - $(\lambda y((+2)y))3$
  - $((+2)3) = 5$

Esta técnica de traduzir a avaliação de uma função que recebe múltiplos argumentos para uma sequência de funções que cada uma recebe um único argumento é chamada de **currying**.

Cálculo Lambda

Uma variavel é livre (independente), se for diferente do termo-λ.

Exemplo 1

$\lambda x.(xy)$

y não esta vinculado a abstração λx.

Exemplo 2

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

O primeiro y em λx é livre, enquanto o outro y em λy é dependente.

Cálculo Lambda

Assim, uma variável x é livre em E se:

- $E = x$
- $E = \lambda y.E_1$ onde $y \neq x$ e x é livre em E_1
- $E = E_1\ E_2$, onde x é livre em E_1 ou x é livre em E_2

Uma expressão é um **combinador** se ela não possui nenhuma variável livre.

$\lambda\texttt{x}.\lambda\texttt{y}.\texttt{xyx}$

$\lambda\texttt{x}.\texttt{x}$

$\lambda\texttt{x}.\lambda\texttt{y}.\texttt{xyx}$

Cálculo Lambda

Variáveis **dependentes** (ou vinculadas) possuem escopo.

- Se uma ocorrência de x é livre em E, então x é dependente de $\lambda x.$ em $\lambda x.E$
- Se uma ocorrência de x é dependente de uma $\lambda x.$ em E, então x é dependente da mesma $\lambda x.$ em $\lambda z.E$, ou seja, adicionando uma abstração $\lambda z.$ não altera a dependência de x em E, mesmo se $z = x$.
  - $\lambda\texttt{z}.\lambda\texttt{x}.\texttt{xz}$
  - $\lambda\texttt{x}\lambda\texttt{x}.\texttt{x}$ - x é dependente da segunda λx.
- Se uma ocorrência de x é dependente de uma $\lambda x.$ em E_1, então aquela ocorrência de x em E_1 é dependente da mesma abstração $\lambda x.$ em E_1 E_2 e E_2 E_1, ou seja, aplicação não muda a dependência de x.

Cálculo Lambda

Exemplos variáveis livres e dependentes

`(λ x.  x (λ y.  x y z y) x) x y`

Cálculo Lambda

Exemplos variáveis livres e dependentes

`(λ x. x (λ y. x y z y) x) x y`

- `(λ x. (x (λ y. (x y z y)) x)) x y`

Cálculo Lambda

Exemplos variáveis livres e dependentes

`(λ x.  x (λ y.  x y z y) x) x y`

- `(λ x.  (x (λ y.  (x y z y)) x)) x y`
- `(λ x.  x (λ y.  x y z y) x) x y`

Cálculo Lambda

Exemplos variáveis livres e dependentes

$(\lambda\ \mathtt{x.}\ \ \mathtt{x}\ (\lambda\ \mathtt{y.}\ \ \mathtt{x\ y\ z\ y})\ \mathtt{x})\ \mathtt{x\ y}$

- $(\lambda\ \mathtt{x.}\ \ (\mathtt{x}\ (\lambda\ \mathtt{y.}\ \ (\mathtt{x\ y\ z\ y}))\ \mathtt{x}))\ \mathtt{x\ y}$
- $(\lambda\ \mathtt{x.}\ \ \mathtt{x}\ (\lambda\ \mathtt{y.}\ \ \mathtt{x\ y\ z\ y})\ \mathtt{x})\ \mathtt{x\ y}$

$(\lambda\ \mathtt{x.}\ \ \lambda\ \mathtt{y.x\ y})\ (\lambda\ \mathtt{z.x\ z})$

Cálculo Lambda

Exemplos variáveis livres e dependentes

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

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

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

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

Cálculo Lambda

Exemplos variáveis livres e dependentes

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

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

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

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

Cálculo Lambda

Exemplos variáveis livres e dependentes

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

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

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

- $(\lambda$ x. λ y.(x y)) $(\lambda$ z.(x z))
- $(\lambda$ x. λ y.x y) $(\lambda$ z.x z)

$(\lambda$ x.x λ x.z x)

Exemplos variáveis livres e dependentes

`(λ x.  x (λ y.  x y z y) x) x y`

- `(λ x.  (x (λ y.  (x y z y)) x)) x y`
- `(λ x.  x (λ y.  x y z y) x) x y`

`(λ x.  λ y.x y) (λ z.x z)`

- `(λ x.  λ y.(x y)) (λ z.(x z))`
- `(λ x.  λ y.x y) (λ z.x z)`

`(λ x.x λ x.z x)`

- `(λ x.(x λ x.(z x)))`

Exemplos variáveis livres e dependentes

`(λ x.  x (λ y.  x y z y) x) x y`

- `(λ x.  (x (λ y.  (x y z y)) x)) x y`
- `(λ x.  x (λ y.  x y z y) x) x y`

`(λ x.  λ y.x y) (λ z.x z)`

- `(λ x.  λ y.(x y)) (λ z.(x z))`
- `(λ x.  λ y.x y) (λ z.x z)`

`(λ x.x λ x.z x)`

- `(λ x.(x λ x.(z x)))`
- `(λ x.x λ x.z x)`

Dizemos que duas funções são α-**equivalentes** se elas variam apenas no nome das variáveis dependentes.

- $\lambda x.x$ e $\lambda y.y$ são α-equivalentes, pois ambas x e y são variáveis dependentes, vinculadas a uma abstração.
- x e y não são α-equivalentes, pois ambas são variáveis livres

Cálculo Lambda

α-**conversão** ou α-**renomeação** é a operação de "trocar" (renomear) **apenas** variáveis dependentes (vinculadas).

- Devemos tomar cuidado quando renomeamos variáveis. A função anterior e a atual devem ainda ser α-**equivalentes**.

$E\ \{y/x\}$ - renomeie todas as ocorrências de x com y, se x é vinculada, e respeitando seu escopo

- E ::- ID
  - $x\ \{y/x\}\ =\ y$
  - $z\ \{y/x\}\ =\ z$, se $z \neq x$
- E ::- $E_1\ \ E_2$
  - $(E_1\ E_2)\{y/x\} = (E_1\{y/x\})\ (E_2\{y/x\})$
- E ::- $\lambda ID.E$
  - $(\lambda x.E)\ \{y/x\} = (\lambda y.E\{y/x\})$
  - $(\lambda z.E)\ \{y/x\} = (\lambda z.E\{y/x\})$, se $z \neq x$

Cálculo Lambda

α-**conversão** ou α-**renomeação**

Exemplo:

- $(\lambda x.x)\ \{f/x\}$

Cálculo Lambda

α-**conversão** ou α-**renomeação**

Exemplo:

- $(\lambda x.x)\ \{f/x\}$
  - $(\lambda f.(x)\ \{f/x\})$

α-**conversão** ou α-**renomeação**

Exemplo:

- $(\lambda x.x)\ \{f/x\}$
  - $(\lambda f.(x)\ \{f/x\})$
  - $(\lambda f.(f))$

Cálculo Lambda

α-**conversão** ou α-**renomeação**

Exemplo:

- $(\lambda x.x)\ \{f/x\}$
  - $(\lambda f.(x)\ \{f/x\})$
  - $(\lambda f.(f))$
  - $(\lambda f.f)$

Cálculo Lambda

α-conversão ou **α-renomeação**

Exemplo:

- $(\lambda x.x)\ \{f/x\}$
  - $(\lambda f.(x)\ \{f/x\})$
  - $(\lambda f.(f))$
  - $(\lambda f.f)$
- $((\lambda x.x(\lambda y.xyzy)x)xy)\ \{b/x\}$

Cálculo Lambda

α-**conversão** ou α-**renomeação**

Exemplo:

- $(\lambda x.x)\ \{f/x\}$
  - $(\lambda f.(x)\ \{f/x\})$
  - $(\lambda f.(f))$
  - $(\lambda f.f)$

- $((\lambda x.x(\lambda y.xyzy)x)xy)\ \{b/x\}$
  - $(\lambda x.x(\lambda y.xyzy)x)\{b/x\}\ (x)\{b/x\}\ (y)\{b/x\}$

α-**conversão** ou α-**renomeação**

Exemplo:

- $(\lambda x.x)\ \{f/x\}$
  - $(\lambda f.(x)\ \{f/x\})$
  - $(\lambda f.(f))$
  - $(\lambda f.f)$

- $((\lambda x.x(\lambda y.xyzy)x)xy)\ \{b/x\}$
  - $(\lambda x.x(\lambda y.xyzy)x)\{b/x\}\ (x)\{b/x\}\ (y)\{b/x\}$
  - $(\lambda x.x(\lambda y.xyzy)x)\{b/x\}\ (x)\{b/x\}\ y$

Cálculo Lambda

α-conversão ou **α-renomeação**

Exemplo:

- $(\lambda x.x)\ \{f/x\}$
  - $(\lambda f.(x)\ \{f/x\})$
  - $(\lambda f.(f))$
  - $(\lambda f.f)$

- $((\lambda x.x(\lambda y.xyzy)x)xy)\ \{b/x\}$
  - $(\lambda x.x(\lambda y.xyzy)x)\{b/x\}\ (x)\{b/x\}\ (y)\{b/x\}$
  - $(\lambda x.x(\lambda y.xyzy)x)\{b/x\}\ (x)\{b/x\}\ y$
  - $(\lambda x.x(\lambda y.xyzy)x)\{b/x\}\ x\ y$

Cálculo Lambda

α-conversão ou **α-renomeação**

Exemplo:

- $(\lambda x.x)\ \{f/x\}$
  - $(\lambda f.(x)\ \{f/x\})$
  - $(\lambda f.(f))$
  - $(\lambda f.f)$

- $((\lambda x.x(\lambda y.xyzy)x)xy)\ \{b/x\}$
  - $(\lambda x.x(\lambda y.xyzy)x)\{b/x\}\ (x)\{b/x\}\ (y)\{b/x\}$
  - $(\lambda x.x(\lambda y.xyzy)x)\{b/x\}\ (x)\{b/x\}\ y$
  - $(\lambda x.x(\lambda y.xyzy)x)\{b/x\}\ x\ y$
  - $(\lambda b.(x(\lambda y.xyzy)x)\{b/x\})\ x\ y$

α-conversão ou **α-renomeação**

Exemplo:

- $(\lambda x.x)\ \{f/x\}$
  - $(\lambda f.(x)\ \{f/x\})$
  - $(\lambda f.(f))$
  - $(\lambda f.f)$

- $((\lambda x.x(\lambda y.xyzy)x)xy)\ \{b/x\}$
  - $(\lambda x.x(\lambda y.xyzy)x)\{b/x\}\ (x)\{b/x\}\ (y)\{b/x\}$
  - $(\lambda x.x(\lambda y.xyzy)x)\{b/x\}\ (x)\{b/x\}\ y$
  - $(\lambda x.x(\lambda y.xyzy)x)\{b/x\}\ x\ y$
  - $(\lambda b.(x(\lambda y.xyzy)x)\{b/x\})\ x\ y$
  - $(\lambda b.(b(\lambda y.xyzy)\{b/x\}b))\ x\ y$

Cálculo Lambda

α-**conversão** ou α-**renomeação**

Exemplo:

- $(\lambda x.x)\ \{f/x\}$
  - $(\lambda f.(x)\ \{f/x\})$
  - $(\lambda f.(f))$
  - $(\lambda f.f)$

- $((\lambda x.x(\lambda y.xyzy)x)xy)\ \{b/x\}$
  - $(\lambda x.x(\lambda y.xyzy)x)\{b/x\}\ (x)\{b/x\}\ (y)\{b/x\}$
  - $(\lambda x.x(\lambda y.xyzy)x)\{b/x\}\ (x)\{b/x\}\ y$
  - $(\lambda x.x(\lambda y.xyzy)x)\{b/x\}\ x\ y$
  - $(\lambda b.(x(\lambda y.xyzy)x)\{b/x\})\ x\ y$
  - $(\lambda b.(b(\lambda y.xyzy)\{b/x\}b))\ x\ y$
  - $(\lambda b.(b(\lambda y.(xyzy)\{b/x\})x))\ x\ y$

α-conversão ou **α-renomeação**

Exemplo:

- $(\lambda x.x)\ \{f/x\}$
  - $(\lambda f.(x)\ \{f/x\})$
  - $(\lambda f.(f))$
  - $(\lambda f.f)$

- $((\lambda x.x(\lambda y.xyzy)x)xy)\ \{b/x\}$
  - $(\lambda x.x(\lambda y.xyzy)x)\{b/x\}\ (x)\{b/x\}\ (y)\{b/x\}$
  - $(\lambda x.x(\lambda y.xyzy)x)\{b/x\}\ (x)\{b/x\}\ y$
  - $(\lambda x.x(\lambda y.xyzy)x)\{b/x\}\ x\ y$
  - $(\lambda b.(x(\lambda y.xyzy)x)\{b/x\})\ x\ y$
  - $(\lambda b.(b(\lambda y.xyzy)\{b/x\}b))\ x\ y$
  - $(\lambda b.(b(\lambda y.(xyzy)\{b/x\})x))\ x\ y$
  - $(\lambda b.(b(\lambda y.(byzy))b))\ x\ y$

α-conversão ou **α-renomeação**

Exemplo:

- $(\lambda x.x)\ \{f/x\}$
  - $(\lambda f.(x)\ \{f/x\})$
  - $(\lambda f.(f))$
  - $(\lambda f.f)$

- $((\lambda x.x(\lambda y.xyzy)x)xy)\ \{b/x\}$
  - $(\lambda x.x(\lambda y.xyzy)x)\{b/x\}\ (x)\{b/x\}\ (y)\{b/x\}$
  - $(\lambda x.x(\lambda y.xyzy)x)\{b/x\}\ (x)\{b/x\}\ y$
  - $(\lambda x.x(\lambda y.xyzy)x)\{b/x\}\ x\ y$
  - $(\lambda b.(x(\lambda y.xyzy)x)\{b/x\})\ x\ y$
  - $(\lambda b.(b(\lambda y.xyzy)\{b/x\}b))\ x\ y$
  - $(\lambda b.(b(\lambda y.(xyzy)\{b/x\})x))\ x\ y$
  - $(\lambda b.(b(\lambda y.(byzy))b))\ x\ y$
  - $(\lambda b.(b(\lambda y.byzy)b))\ x\ y$

Substituição

É a operação que permite trocar uma **variável livre** por uma expressão lambda. Assim, $E[x \rightarrow N]$ significa que deve-se substituir todas as ocorrências da variável livre x em E pela expressão N.

- E ::- ID
  - $x\ [x \rightarrow N]\ =\ N$
  - $z\ [x \rightarrow N]\ =\ z$, se $z \neq x$
- E ::- $E_1\ \ E_2$
  - $(E_1\ E_2)[x \rightarrow N] = (E_1[x \rightarrow N])\ (E_2[x \rightarrow N])$
- E ::- $\lambda ID.E$
  - $(\lambda x.E)[x \rightarrow N] = (\lambda x.E)$
  - $(\lambda z.E)[x \rightarrow N] = (\lambda z.E[x \rightarrow N])$, se $z \neq x$ e z não é uma variável livre em N
  - $(\lambda z.E)[x \rightarrow N] = (\lambda z'.E\{z'/z\}[x \rightarrow N])$, se $z \neq x$, z é uma variável livre em N e y' é um novo nome para a variável dependente z da abstração $\lambda z.$, antes de executar a substituição

Substituição

Exemplo:

- $(\lambda x.x)[x \rightarrow f]$

Cálculo Lambda

Substituição

Exemplo:

- $(\lambda x.x)[x \to f]$
  - $(\lambda x.x)$, pois x é uma variável dependente

Cálculo Lambda

Substituição

Exemplo:

- $(\lambda x.x)[x \rightarrow f]$
  - $(\lambda x.x)$, pois x é uma variável dependente
- $(+1x)[x \rightarrow 2]$

Substituição

Exemplo:

- $(\lambda x.x)[x \rightarrow f]$
  - $(\lambda x.x)$, pois x é uma variável dependente

- $(+1x)[x \rightarrow 2]$
  - $(+[x \rightarrow 2]1[x \rightarrow 2]x[x \rightarrow 2])$

Substituição

Exemplo:

- $(\lambda x.x)[x \rightarrow f]$
    - $(\lambda x.x)$, pois x é uma variável dependente

- $(+1x)[x \rightarrow 2]$
    - $(+[x \rightarrow 2]1[x \rightarrow 2]x[x \rightarrow 2])$
    - $(+1\ 2)$

Cálculo Lambda

Substituição

Exemplo:

- $(\lambda x.x)[x \rightarrow f]$
  - $(\lambda x.x)$, pois x é uma variável dependente
- $(+1x)[x \rightarrow 2]$
  - $(+[x \rightarrow 2]1[x \rightarrow 2]x[x \rightarrow 2])$
  - $(+1\ 2)$
- $(\lambda x.yx)[y \rightarrow \lambda z.xz]$

Substituição

Exemplo:

- $(\lambda x.x)[x \rightarrow f]$
  - $(\lambda x.x)$, pois x é uma variável dependente

- $(+1x)[x \rightarrow 2]$
  - $(+[x \rightarrow 2]1[x \rightarrow 2]x[x \rightarrow 2])$
  - $(+1\ 2)$

- $(\lambda x.yx)[y \rightarrow \lambda z.xz]$
  - Note: x é uma variável livre em N

Substituição

Exemplo:

- $(\lambda x.x)[x \rightarrow f]$
  - $(\lambda x.x)$, pois x é uma variável dependente

- $(+1x)[x \rightarrow 2]$
  - $(+[x \rightarrow 2]1[x \rightarrow 2]x[x \rightarrow 2])$
  - $(+1\ 2)$

- $(\lambda x.yx)[y \rightarrow \lambda z.xz]$
  - Note: x é uma variável livre em N
  - $(\lambda w.(yx)\{w/x\}[y \rightarrow \lambda z.xz])$

Substituição

Exemplo:

- $(\lambda x.x)[x \rightarrow f]$
  - $(\lambda x.x)$, pois x é uma variável dependente

- $(+1x)[x \rightarrow 2]$
  - $(+[x \rightarrow 2]1[x \rightarrow 2]x[x \rightarrow 2])$
  - $(+1\ 2)$

- $(\lambda x.yx)[y \rightarrow \lambda z.xz]$
  - Note: x é uma variável livre em N
  - $(\lambda w.(yx)\{w/x\}[y \rightarrow \lambda z.xz])$
  - $(\lambda w.(yw)[y \rightarrow \lambda z.xz])$

Cálculo Lambda

Substituição

Exemplo:

- $(\lambda x.x)[x \rightarrow f]$
  - $(\lambda x.x)$, pois x é uma variável dependente

- $(+1x)[x \rightarrow 2]$
  - $(+[x \rightarrow 2]1[x \rightarrow 2]x[x \rightarrow 2])$
  - $(+1\ 2)$

- $(\lambda x.yx)[y \rightarrow \lambda z.xz]$
  - Note: x é uma variável livre em N
  - $(\lambda w.(yx)\{w/x\}[y \rightarrow \lambda z.xz])$
  - $(\lambda w.(yw)[y \rightarrow \lambda z.xz])$
  - $(\lambda w.(y[y \rightarrow \lambda z.xz]w[y \rightarrow \lambda z.xz]))$

Substituição

Exemplo:

- $(\lambda x.x)[x \rightarrow f]$
  - $(\lambda x.x)$, pois x é uma variável dependente

- $(+1x)[x \rightarrow 2]$
  - $(+[x \rightarrow 2]1[x \rightarrow 2]x[x \rightarrow 2])$
  - $(+1\ 2)$

- $(\lambda x.yx)[y \rightarrow \lambda z.xz]$
  - Note: x é uma variável livre em N
  - $(\lambda w.(yx)\{w/x\}[y \rightarrow \lambda z.xz])$
  - $(\lambda w.(yw)[y \rightarrow \lambda z.xz])$
  - $(\lambda w.(y[y \rightarrow \lambda z.xz]w[y \rightarrow \lambda z.xz]))$
  - $(\lambda w.(\lambda z.xz)w)$

Execução

Execução é a sequência de termos resultantes das chamadas de funções, onde cada passo nessa sequência é uma β-**Redução**.

Quando não for mais possível efetuar nenhuma redução, então dizemos que estamos na **forma normal**, ou seja, a forma normal é o máximo de reduções de uma abstração lambda. Assim, na forma normal, não temos mais nenhuma subexpressão na forma $(\lambda x.E)N$.

Cálculo Lambda

Redução-Beta (β-**Redução**)

$$(\lambda x.E)N \Longrightarrow E[x \rightarrow N]$$

É o principal conceito do Cálculo-λ. E é uma expressão para x, isto é, $\lambda x.E$ é uma função de x. Assim, $(\lambda x.E)N$ refere-se à aplicação da função para a entrada N. Podemos então **reduzir** uma aplicação $(\lambda x.E)N$ para alguma outra coisa simplesmente plugando N para todas as ocorrências de x em E.
Formalmente: $(\lambda x.E)N$ β-reduz para $E[x \rightarrow N]$.

Cálculo Lambda

Redução-Beta (β-**Redução**)

Exemplo:

- $(\lambda x.x)y$

Redução-Beta (β-**Redução**)

Exemplo:

- $(\lambda x.x)y$
  - $x[x \rightarrow y]$

Cálculo Lambda

Redução-Beta (β-**Redução**)

Exemplo:

- $(\lambda x.x)y$
  - $x[x \rightarrow y]$
  - y

Cálculo Lambda

Redução-Beta (β-**Redução**)

Exemplo:

- $(\lambda x.x)y$
  - $x[x \rightarrow y]$
  - y

- $(\lambda x.x\ y)1$

Cálculo Lambda

Redução-Beta (β-**Redução**)

Exemplo:

- $(\lambda x.x)y$
  - $x[x \rightarrow y]$
  - y

- $(\lambda x.x\ y)1$
  - $(x\ y)[x \rightarrow 1]$

Cálculo Lambda

Redução-Beta (β-**Redução**)

Exemplo:

- $(\lambda x.x)y$
  - $x[x \rightarrow y]$
  - y
- $(\lambda x.x\ y)1$
  - $(x\ y)[x \rightarrow 1]$
  - $1\ y$

Redução-Beta (β-**Redução**)

Exemplo:

- $(\lambda x.x)y$
  - $x[x \rightarrow y]$
  - y

- $(\lambda x.x\ y)1$
  - $(x\ y)[x \rightarrow 1]$
  - $1\ y$

- $(\lambda x.x(\lambda x.x))(u\ r)$

Cálculo Lambda

Redução-Beta (β-**Redução**)

Exemplo:

- $(\lambda x.x)y$
  - $x[x \to y]$
  - y

- $(\lambda x.x\ y)1$
  - $(x\ y)[x \to 1]$
  - $1\ y$

- $(\lambda x.x(\lambda x.x))(u\ r)$
  - $(x(\lambda x.x))[x \to (u\ r)]$

Cálculo Lambda

Redução-Beta (β-**Redução**)

Exemplo:

- $(\lambda x.x)y$
  - $x[x \to y]$
  - y

- $(\lambda x.x\ y)1$
  - $(x\ y)[x \to 1]$
  - $1\ y$

- $(\lambda x.x(\lambda x.x))(u\ r)$
  - $(x(\lambda x.x))[x \to (u\ r)]$
  - $(u\ r)(\lambda x.x)$, note que o outro x é vinculado a outra abstração

Cálculo Lambda

Redução-Beta (β-**Redução**)

Exemplo:

- $(\lambda x.y)((\lambda z.zz)(\lambda w.w))$

Cálculo Lambda

Redução-Beta (β-**Redução**)

Exemplo:

- $(\lambda x.y)((\lambda z.zz)(\lambda w.w))$
  - $(\lambda x.y)((\lambda z.zz)(\lambda w.w))$

Cálculo Lambda

Redução-Beta (β-**Redução**)

Exemplo:

- $(\lambda x.y)((\lambda z.zz)(\lambda w.w))$
  - $(\lambda x.y)((\lambda z.zz)(\lambda w.w))$
  - $(\lambda x.y)((zz)[z \rightarrow (\lambda w.w)])$

Cálculo Lambda

Redução-Beta (β-**Redução**)

Exemplo:

- $(\lambda x.y)((\lambda z.zz)(\lambda w.w))$
  - $(\lambda x.y)((\lambda z.zz)(\lambda w.w))$
  - $(\lambda x.y)((zz)[z \longrightarrow (\lambda w.w)])$
  - $(\lambda x.y)((\lambda w.w)(\lambda w.w))$

Cálculo Lambda

Redução-Beta (β-**Redução**)

Exemplo:

- $(\lambda x.y)((\lambda z.zz)(\lambda w.w))$
  - $(\lambda x.y)((\lambda z.zz)(\lambda w.w))$
  - $(\lambda x.y)((zz)[z \rightarrow (\lambda w.w)])$
  - $(\lambda x.y)((\lambda w.w)(\lambda w.w))$
  - $(\lambda x.y)(w[w \rightarrow (\lambda w.w)])$

Redução-Beta (β-**Redução**)

Exemplo:

- $(\lambda x.y)((\lambda z.zz)(\lambda w.w))$
  - $(\lambda x.y)((\lambda z.zz)(\lambda w.w))$
  - $(\lambda x.y)((zz)[z \longrightarrow (\lambda w.w)])$
  - $(\lambda x.y)((\lambda w.w)(\lambda w.w))$
  - $(\lambda x.y)(w[w \longrightarrow (\lambda w.w)])$
  - $(\lambda x.y)(\lambda w.w)$

Cálculo Lambda

Redução-Beta (β-**Redução**)

Exemplo:

- $(\lambda x.y)((\lambda z.zz)(\lambda w.w))$
  - $(\lambda x.y)((\lambda z.zz)(\lambda w.w))$
  - $(\lambda x.y)((zz)[z \longrightarrow (\lambda w.w)])$
  - $(\lambda x.y)((\lambda w.w)(\lambda w.w))$
  - $(\lambda x.y)(w[w \longrightarrow (\lambda w.w)])$
  - $(\lambda x.y)(\lambda w.w)$
  - $y[x \longrightarrow (\lambda w.w)]$

Cálculo Lambda

Redução-Beta (β-**Redução**)

Exemplo:

- $(\lambda x.y)((\lambda z.zz)(\lambda w.w))$
  - $(\lambda x.y)((\lambda z.zz)(\lambda w.w))$
  - $(\lambda x.y)((zz)[z \longrightarrow (\lambda w.w)])$
  - $(\lambda x.y)((\lambda w.w)(\lambda w.w))$
  - $(\lambda x.y)(w[w \longrightarrow (\lambda w.w)])$
  - $(\lambda x.y)(\lambda w.w)$
  - $y[x \longrightarrow (\lambda w.w)]$
  - y

Cálculo Lambda

Note: até o momento temos apenas definição de funções e chamada de funções.

- Precisamos suportar expressões booleanas;

Cálculo Lambda

Lógica Booleana

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

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

and $= (\lambda a.(\lambda b.(a\ b\ FALSE)))$

- se a é $TRUE$, retorna o primeiro parâmetro, que é b (agora depende só de b)
- se a é $FALSE$, retorna o segundo parâmetro, que é $FALSE$

or $= (\lambda a.(\lambda b.(a\ TRUE\ b)))$

- se a é $TRUE$, retorna o primeiro parâmetro, que é $TRUE$
- se a é $FALSE$, retorna o segundo parâmetro, que é b (agora depende só de b)

not $= (\lambda a.a\ FALSE\ \ TRUE)$,

- se a é $TRUE$, retorna o primeiro parâmetro, que é $FALSE$
- se a é $FALSE$, retorna o segundo parâmetro, que é $TRUE$

Lógica Booleana

Função identidade

```
if e then
  a
else
  b
```

- if $TRUE\ a\ b = a$
- if $FALSE\ a\ b = b$
- Assim, if $= \lambda e.e$

Lógica Booleana

Função identidade

```
if e then
  a
else
  b
```

- if $TRUE\ a\ b = a$
- if $FALSE\ a\ b = b$
- Assim, if $= \lambda e.e$

Como funciona?

Lógica Booleana

Como funciona?

if e a b

Cálculo Lambda

Lógica Booleana

Como funciona?

if e a b

- $(\lambda e.e) a\ b$

Lógica Booleana

Como funciona?

if $e\ a\ b$

- $(\lambda e.e)a\ b$
- Suponha $e == TRUE$

Lógica Booleana

Como funciona?

if $e\ a\ b$

- $(\lambda e.e)a\ b$
- Suponha $e == TRUE$
- $(\lambda TRUE.TRUE)a\ b$

Lógica Booleana

Como funciona?

if $e\ a\ b$

- $(\lambda e.e)a\ b$
- Suponha $e == TRUE$
- $(\lambda TRUE.TRUE)a\ b$
- $TRUE\ a\ b$

Lógica Booleana

Como funciona?

if $e\ a\ b$

- $(\lambda e.e)a\ b$
- Suponha $e == TRUE$
- $(\lambda TRUE.TRUE)a\ b$
- $TRUE\ a\ b$
- a

Funções anônimas

São funções que não possuem um identificador.

Para entender como funcionam, vamos entender primeiramente o conceito de **funções de alta ordem**. Uma função de alta ordem pode ser de dois tipos:

- Recebem uma ou mais funções como argumentos;
- Devolvem outra função como valor de retorno;

Ou seja, pode-se passar uma função A como parâmetro para uma função B e retornar uma função C baseada na função A passada como parâmetro para B.

Lambda Calculus - Aplicações em Programação

Diversas linguagens funcionais derivam do Cálculo-λ.

- LISP
- Scheme
- ML
- Haskell
- Clojure
- F#
- ...

Python - Exemplo Lambda Calculus

Função identidade

```
print ((lambda x: x) (2))
```

Saída

2

Tente executar este código em:

http://www.compileonline.com/execute_python_online.php

Python - Exemplo Lambda Calculus

Soma de dois valores

```
print ((lambda x, y: x+y)(1,2))
```

Saída

3

Tente executar este código em:

http://www.compileonline.com/execute_python_online.php

Python - Exemplo Lambda Calculus

Retornando uma tupla.

Soma de dois valores

```
print ((lambda x: (x + 1, x + 2)) (1))
```

Saída

(2, 3)

Tente executar este código em:

http://www.compileonline.com/execute_python_online.php

Python - Exemplo Lambda Calculus

Guardando uma expressão lambda em uma variável.

Soma de dois valores

```
soma = (lambda x, y: x+y)(1,2)
print (soma(2 3))
```

Saída

5

Tente executar este código em:

http://www.compileonline.com/execute_python_online.php

Python - Exemplo Lambda Calculus

Aplicando uma expressão lambda a uma sequência.

```
print (map(lambda x: x*x, [1, 2, 3, 4, 5, 6]))
```

Saída

[1, 4, 9, 16, 25, 36]

* A função `map(função ou None, seq1, seq2, ..., seqN)`, recebe uma função ou None e aplica ela sobre uma sequência de valores.

Tente executar este código em:

http://www.compileonline.com/execute_python_online.php

Python - Exemplo Lambda Calculus

Aplicando uma expressão lambda a uma sequência.

```
print (map(lambda x: x*x, range(1,7)))
```

Saída

[1, 4, 9, 16, 25, 36]

* A função `map(função ou None, seq1, seq2, ..., seqN)`, recebe uma função ou None e aplica ela sobre uma sequência de valores.

Tente executar este código em:

http://www.compileonline.com/execute_python_online.php

Python - Exemplo Lambda Calculus

Ordenação

```
cientistas = ["Nicolau Copernico", "Galileu Galilei",
              "Johannes Kepler", "Isaac Newton",
              "Antoine Laurent Lavoisier", "Michael Faraday",
              "Charles Darwin", "Louis Pasteur",
              "James Clerk Maxwell", "Nikola Tesla",
              "Albert Einstein", "Niels Bohr"]
#L.sort(key=None, reverse=False)
cientistas.sort(key = lambda x: x.split(" ")[-1].lower())
print cientistas
```

Tente executar este código em:

http://www.compileonline.com/execute_python_online.php

Python - Exemplo Lambda Calculus

Retorna sim, se a maior que b, caso contrário retorna não.

```
print ((lambda a, b: "sim" if a > b else "nao")(3, 2))
```

Tente executar este código em:

http://www.compileonline.com/execute_python_online.php

Atividade

Exercícios para a aula no Moodle.