Cálculo λ

Profa. Dra. Gina. M. B. Oliveira

Alonzo Church (1903–1995)

Professor em Princeton, EUA (1929–1967) e UCLA (1967–1990)

Inventou um sistema formal, chamado ***λ-calculus*** (Lambda Cálculo) , e definiu a noção de função computável utilizando este sistema.

O Lambda Cálculo pode ser chamado “a menor linguagem de programação universal” do mundo.

As linguagens de programação funcionais, como Lisp, Miranda, ML, Haskell são baseadas no Lambda Cálculo.

Cálculo λ ou Lambda Cálculo (λ Calculus)

O lambda cálculo pode ser visto como uma linguagem de programação abstrata em que funções podem ser combinadas para formar outras funções, de uma forma *pura*.

O lambda cálculo trata funções como ***cidadãos de primeira classe***, isto é, entidades que podem, como um dado qualquer, ser utilizadas como argumentos e retornadas como valores de outras funções.

Sintaxe

Uma expressão simples do lambda cálculo:

```
(+ 4 5)
```

Todas as aplicações de funções no lambda cálculo são escritas no formato prefixo.

Avaliando a expressão: `(+ 4 5) = 9`

A avaliação da expressão procede por redução:

```
(+ (* 5 6)(* 4 3)) → (+ 30 (* 4 3))
                   → (+ 30 12) →  42
```

Sintaxe

Uma *abstração lambda* é um tipo de expressão que denota uma função:

```
(λx. + x 1)
```

O λ determina que existe uma função, e é imediatamente seguido por uma variável, denominada *parâmetro formal* da função.

(λ	x	.	+	x	1)
A função de	x	que	incrementa	x de	1

Sintaxe

Uma expressão lambda deve ter a forma:

```
<exp> ::=    <constante>
           | <variavel>
           | <exp> <exp>
           | λ<variavel>.<exp>
```

Os parênteses podem ser utilizados nas expressões para prevenir ambiguidades.

Sintaxe

Uma expressão lambda deve ter a forma:

```
<exp> ::=   <constante>
          | <variavel>
Aplicação → | <exp> <exp>
Abstração Lambda → | λ<variavel>.<exp>
```

Os parênteses podem ser utilizados nas expressões para prevenir ambiguidades.

Sintaxe

Derivação da expressão Lambda (λ x. ((+ 1) x)):

<exp> => (λ <variable>. <expr>)

⇒ (λ x. <expr>)

⇒ (λ x. (<expr> <expr>))

⇒ (λ x. ((<expr> <expr>) <expr>))

⇒ (λ x. ((<constant> <constant>) <expr>))

⇒ (λ x. ((+ 1) <expr>))

⇒ (λ x. ((+ 1) <variable>))

⇒ (λ x. ((+ 1) x))

Exemplos de expressões lambda

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

$(\lambda x.\ f\ \ x)$

$x\ y$

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

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

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

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

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

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

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

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

Sintaxe

Precisamos de regras claras para evitar ambiguidade na escrita de expressões lambda.

Exemplo: $\lambda x.x\ y$ representa $\lambda x.(x\ y)$ ou $(\lambda x.x)\ y$?

A expressão Lambda se estende para a direita

$$\lambda f.\ x\ y \quad \equiv \quad \lambda f.(x\ y)$$

A aplicação é associativa à esquerda

$$x\ y\ z \quad \equiv \quad (x\ y)\ z$$

Múltiplos lambdas podem ser omitidos

$$\lambda f\ g.\ x \quad \equiv \quad \lambda f\ .\ \lambda g.\ x$$

Variáveis Livres e Ligadas

Definição: seja a expressão $\lambda x.M$, dizemos que M é o **escopo** de λx.

Definição: uma ocorrência de x dentro do escopo de λx é dita **ligada**. Caso contrário, x ocorre **livre**.

Exemplo:

a) $\lambda x.x\ y$ — x é ligada, y é livre

b) $\lambda x.z\ \lambda z.z\ x$ — z é livre, z e x são ligadas

c) $\lambda x.x\ \lambda x.x\ y$ — x e x são ligadas, y é livre

Nota:

1. Um mesmo nome por ter ocorrências ligadas e livres na mesma expressão (b).
2. Uma ocorrência de x é sempre ligada ao λx mais interno (c).

Variáveis Livres e Ligadas

As variáveis livres de um termo são definidas como:

$$FV(x) = \{ x \}$$

$$FV(e_1\ e_2) = FV(e_1) \cup FV(e_2)$$

$$FV(\lambda x \,.\, e) = FV(e) - \{ x \}$$

<u>Exemplos:</u>

(λx. + x y) 4 // y é livre

λx. + ((λy. + y z) 7) x // z é livre

+ x ((λx. + x 1) 4) // o primeiro x é livre

β-redução

A aplicação de um argumento à uma abstração lambda implica na substituição das ocorrências das variáveis correspondentes ao argumento:

$$(\lambda x.\ +\ x\ 1)\ 4 \quad \rightarrow \quad +\ 4\ 1$$

Esta operação é denominada **β-redução.**

Funções também podem ser passadas como argumento:

$$
\begin{aligned}
(\lambda f.\ f\ 3)\ (\lambda x.\ +\ x\ 1) \quad &\rightarrow \quad (\lambda x.\ +\ x\ 1)\ 3 \\
&\rightarrow \quad +\ 3\ 1 \\
&\rightarrow \quad 4
\end{aligned}
$$

B-redução Exemplo 1

```
(λx. λy. + x ((λx.- x 3) y)) 5 6
```

B-redução Exemplo 1

$$(\lambda x.\ \lambda y.\ +\ x\ ((\lambda x. -\ x\ 3)\ y))\ 5\ 6$$

$$(\lambda x.\ \lambda y.\ +\ x\ (\boxed{(\lambda x. -\ x\ 3)\ y}))\ 5\ 6$$

B-redução   Exemplo 1

$$(\lambda x.\ \lambda y.\ +\ x\ ((\lambda x.-\ x\ 3)\ y))\ 5\ 6$$

$$(\lambda x.\ \lambda y.\ +\ x\ (\boxed{(\lambda x.-\ x\ 3)\ y})\ 5\ 6$$

$$(\lambda x.\ \lambda y.\ +\ x\ (-\ y\ 3))\ 5\ 6$$

B-redução Exemplo 1

$$(\lambda x.\ \lambda y.\ +\ x\ ((\lambda x. -\ x\ 3)\ y))\ 5\ 6$$

$$(\lambda x.\ \lambda y.\ +\ x\ (\boxed{(\lambda x. -\ x\ 3)\ y})\ )\ 5\ 6$$

$$(\lambda x.\ \lambda y.\ +\ x\ (-\ y\ 3))\ 5\ 6$$

$$(\boxed{\lambda x}.\ \lambda y.\ \boxed{+\ x\ (-\ y\ 3)})\ \boxed{5}\ 6$$

B-redução Exemplo 1

$$(\lambda x.\ \lambda y.\ +\ x\ ((\lambda x.-\ x\ 3)\ y))\ 5\ 6$$

$$(\lambda x.\ \lambda y.\ +\ x\ (\boxed{(\lambda x.-\ x\ 3)\ y}))\ 5\ 6$$

$$(\lambda x.\ \lambda y.\ +\ x\ (-\ y\ 3))\ 5\ 6$$

$$(\boxed{\lambda x}.\ \lambda y.\ \boxed{+\ x\ (-\ y\ 3)})\ \boxed{5}\ 6$$

$$(\lambda y.\ +\ 5\ (-\ y\ 3))\ 6$$

B-redução Exemplo 1

$$(\lambda x.\ \lambda y.\ +\ x\ ((\lambda x.-\ x\ 3)\ y))\ 5\ 6$$

$$(\lambda x.\ \lambda y.\ +\ x\ (\boxed{(\lambda x.-\ x\ 3)\ y})\ )\ 5\ 6$$

$$(\lambda x.\ \lambda y.\ +\ x\ (-\ y\ 3))\ 5\ 6$$

$$(\boxed{\lambda x}.\ \lambda y.\ \boxed{+\ x\ (-\ y\ 3)})\ \boxed{5}\ 6$$

$$(\lambda y.\ +\ 5\ (-\ y\ 3))\ 6$$

$$(\boxed{\lambda y}.\ \boxed{+\ 5\ (-\ y\ 3)})\ \boxed{6}$$

B-redução Exemplo 1

$$(\lambda x.\ \lambda y.\ +\ x\ ((\lambda x. -\ x\ 3)\ y))\ 5\ 6$$

$$(\lambda x.\ \lambda y.\ +\ x\ (\boxed{(\lambda x. -\ x\ 3)\ y}))\ 5\ 6$$

$$(\lambda x.\ \lambda y.\ +\ x\ (-\ y\ 3))\ 5\ 6$$

$$(\boxed{\lambda x}.\ \lambda y.\ \boxed{+\ x\ (-\ y\ 3)})\ \boxed{5}\ 6$$

$$(\lambda y.\ +\ 5\ (-\ y\ 3))\ 6$$

$$(\boxed{\lambda y}.\ \boxed{+\ 5\ (-\ y\ 3)})\ \boxed{6}$$

$$+\ 5\ (-\ 6\ 3)$$

B-redução Exemplo 1

$$(\lambda x.\ \lambda y.\ +\ x\ ((\lambda x.-\ x\ 3)\ y))\ 5\ 6$$

$$(\lambda x.\ \lambda y.\ +\ x\ (\boxed{(\lambda x.-\ x\ 3)\ y}))\ 5\ 6$$

$$(\lambda x.\ \lambda y.\ +\ x\ (-\ y\ 3))\ 5\ 6$$

$$(\boxed{\lambda x}.\ \lambda y.\ \boxed{+\ x\ (-\ y\ 3)})\ \boxed{5}\ 6$$

$$(\lambda y.\ +\ 5\ (-\ y\ 3))\ 6$$

$$(\boxed{\lambda y}.\ \boxed{+\ 5\ (-\ y\ 3)})\ \boxed{6}$$

$$+\ 5\ (-\ 6\ 3)$$

$$+\ 5\ \ 3$$

$$8$$

B-redução Exemplo 2

$$((\lambda x.\ (\lambda y.\ +\ x\ y)\ 5)\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

B-redução Exemplo 2

$$((\lambda x.\ (\lambda y.\ +\ x\ y)\ 5)\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

$$((\lambda x.\ (\boxed{\lambda y.}\ \boxed{+\ x\ y)}\ \boxed{5})\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

B-redução Exemplo 2

$$((\lambda x.\ (\lambda y.\ +\ x\ y)\ 5)\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

$$((\lambda x.\ (\boxed{\lambda y.}\ \boxed{+\ x\ y)}\ \boxed{5})\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

$$((\lambda x.\ +\ x\ 5)\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

B-redução   Exemplo 2

$$((\lambda x.\ (\lambda y.\ +\ x\ y)\ 5)\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

$$((\lambda x.\ (\boxed{\lambda y.}\ \boxed{+\ x\ y)}\ \boxed{5})\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

$$((\lambda x.\ +\ x\ 5)\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

$$((\lambda x.\ +\ x\ 5)\ ((\boxed{\lambda y.}\ \boxed{-\ y\ 3)}\ \boxed{7})) =$$

B-redução Exemplo 2

$$((\lambda x.\ (\lambda y.\ +\ x\ y)\ 5)\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

$$((\lambda x.\ (\boxed{\lambda y.}\ \boxed{+\ x\ y)}\ \boxed{5})\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

$$((\lambda x.\ +\ x\ 5)\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

$$((\lambda x.\ +\ x\ 5)\ ((\boxed{\lambda y.}\ \boxed{-\ y\ 3)}\ \boxed{7})) =$$

$$((\lambda x.\ +\ x\ 5)\ (-\ 7\ 3)) =$$

B-redução Exemplo 2

$$((\lambda x.\ (\lambda y.\ +\ x\ y)\ 5)\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

$$((\lambda x.\ (\boxed{\lambda y.}\ \boxed{+\ x\ y)}\ \boxed{5})\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

$$((\lambda x.\ +\ x\ 5)\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

$$((\lambda x.\ +\ x\ 5)\ ((\boxed{\lambda y.}\ \boxed{-\ y\ 3)}\ \boxed{7})) =$$

$$((\lambda x.\ +\ x\ 5)\ (-\ 7\ 3)) =$$

$$((\lambda x.\ +\ x\ 5)\ \boxed{(-\ 7\ 3)}) =$$

B-redução   Exemplo 2

$$((\lambda x.\ (\lambda y.\ +\ x\ y)\ 5)\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

$$((\lambda x.\ (\boxed{\lambda y.}\ \boxed{+\ x\ y})\ \boxed{5})\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

$$((\lambda x.\ +\ x\ 5)\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

$$((\lambda x.\ +\ x\ 5)\ ((\boxed{\lambda y.}\ \boxed{-\ y\ 3})\ \boxed{7})) =$$

$$((\lambda x.\ +\ x\ 5)\ (-\ 7\ 3)) =$$

$$((\lambda x.\ +\ x\ 5)\ \boxed{(-\ 7\ 3)}) =$$

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

B-redução   Exemplo 2

$$((\lambda x.\ (\lambda y.\ +\ x\ y)\ 5)\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

$$((\lambda x.\ (\boxed{\lambda y.}\ \boxed{+\ x\ y)}\ \boxed{5})\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

$$((\lambda x.\ +\ x\ 5)\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

$$((\lambda x.\ +\ x\ 5)\ ((\boxed{\lambda y.}\ \boxed{-\ y\ 3)}\ \boxed{7})) =$$

$$((\lambda x.\ +\ x\ 5)\ (-\ 7\ 3)) =$$

$$((\lambda x.\ +\ x\ 5)\ \boxed{(-\ 7\ 3)}) =$$

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

$$((\boxed{\lambda x.}\ \boxed{+\ x\ 5)}\ \boxed{4}) =$$

Β-redução Exemplo 2

$$((\lambda x.\ (\lambda y.\ +\ x\ y)\ 5)\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

$$((\lambda x.\ (\boxed{\lambda y.}\ \boxed{+\ x\ y)}\ \boxed{5})\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

$$((\lambda x.\ +\ x\ 5)\ ((\lambda y.\ -\ y\ 3)\ 7)) =$$

$$((\lambda x.\ +\ x\ 5)\ ((\boxed{\lambda y.}\ \boxed{-\ y\ 3)}\ \boxed{7})) =$$

$$((\lambda x.\ +\ x\ 5)\ (-\ 7\ 3)) =$$

$$((\lambda x.\ +\ x\ 5)\ \boxed{(-\ 7\ 3)}) =$$

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

$$((\boxed{\lambda x.}\ \boxed{+\ x\ 5)}\ \boxed{4}) =$$

$$(+\ 4\ 5) = 9$$

B-redução   Exemplo 3

- twice f x = f(f(x));
  - twice square 3 = 81

B-redução   Exemplo 3

- twice f x = f(f(x));
  - twice square 3 = 81
  - square in lambda calculus:
    (λy. (∗ y y))

B-redução   Exemplo 3

- twice f x = f(f(x));
  - twice square 3 = 81
  - square in lambda calculus:
    (λy. (* y y))
  - twice in lambda calculus:
    (λf. (λx. f (f x)))

B-redução Exemplo 3

- twice f x = f(f(x));
  - twice square 3 = 81
  - square in lambda calculus:
    $(\lambda y.\ (*\ y\ y))$
  - twice in lambda calculus:
    $(\lambda f.\ (\lambda x.\ f\ (f\ x)))$
  - twice square 3 =>
    $(((\lambda f.\ (\lambda x.\ f\ (f\ x)))\ (\lambda y.\ (*\ y\ y)))\ 3)$

B-redução   Exemplo 3

- twice f x = f(f(x));
  - twice square 3 = 81
  - square in lambda calculus:
    (λy. (∗ y y))
  - twice in lambda calculus:
    (λf. (λx. f (f x)))
  - twice square 3 =>
    (((λf. (λx. f (f x))) (λy. (∗ y y))) 3)

(((λf. (λx. f (f x))) (λy. (∗ y y))) 3)

B-redução Exemplo 3

- twice f x = f(f(x));
  - twice square 3 = 81
  - square in lambda calculus:
    (λy. (∗ y y))
  - twice in lambda calculus:
    (λf. (λx. f (f x)))
  - twice square 3 =>
    (((λf. (λx. f (f x))) (λy. (∗ y y))) 3)

$$(((\boxed{\lambda f.}\ \boxed{(\lambda x.\ f\ (f\ x))})\ \boxed{(\lambda y.\ (*\ y\ y))})\ 3)$$

β-redução Exemplo 3

- twice f x = f(f(x));
  - twice square 3 = 81
  - square in lambda calculus:
    (λy. (∗ y y))
  - twice in lambda calculus:
    (λf. (λx. f (f x)))
  - twice square 3 =>
    (((λf. (λx. f (f x))) (λy. (∗ y y))) 3)

$(((\boxed{\lambda f.}\ \boxed{(\lambda x.\ f\ (f\ x))})\ \boxed{(\lambda y.\ (*\ y\ y))})\ 3)$

$= ((\ (\lambda x.\ (\lambda y.\ (*\ y\ y))\ ((\lambda y.\ (*\ y\ y))\ x))\ )\ 3)$

B-redução   Exemplo 3

- twice f x = f(f(x));
  - twice square 3 = 81
  - square in lambda calculus:
    $(\lambda y. (* \ y \ y))$
  - twice in lambda calculus:
    $(\lambda f. (\lambda x. f \ (f \ x)))$
  - twice square 3 =>
    $(((\lambda f. (\lambda x. f \ (f \ x))) \ (\lambda y. (* \ y \ y))) \ 3)$

$$(((\boxed{\lambda f.} \ \boxed{(\lambda x. f \ (f \ x))}) \ \boxed{(\lambda y. (* \ y \ y))}) \ 3)$$

$$= ((\ (\lambda x. (\lambda y. (* \ y \ y)) \ ((\lambda y. (* \ y \ y)) \ x)) \ ) \ 3)$$

$$= (\ (\lambda x. (\lambda y. (* \ y \ y)) \ ((\lambda y. (* \ y \ y)) \ x)) \ \ 3)$$

Removendo parênteses ...

B-redução   Exemplo 3

- twice f x = f(f(x));
  - twice square 3 = 81
  - square in lambda calculus:
    (λy. (* y y))
  - twice in lambda calculus:
    (λf. (λx. f (f x)))
  - twice square 3 =>
    (((λf. (λx. f (f x))) (λy. (* y y))) 3)

(((λf. (λx. f (f x))) (λy. (* y y))) 3)
= (((λx. (λy. (* y y)) ((λy. (* y y)) x))) 3)
= ((λx. (λy. (* y y)) ((λy. (* y y)) x)) 3)

β-redução Exemplo 3

- twice f x = f(f(x));
  - twice square 3 = 81
  - square in lambda calculus:
    (λy. (∗ y y))
  - twice in lambda calculus:
    (λf. (λx. f (f x)))
  - twice square 3 =>
    (((λf. (λx. f (f x))) (λy. (∗ y y))) 3)

(((λf. (λx. f (f x))) (λy. (∗ y y))) 3)
= (((λx. (λy. (* y y)) ((λy. (* y y)) x))) 3)
= ((λx. (λy. (* y y)) ((λy. (* y y)) x)) 3)
= ((λy. (* y y)) ((λy. (* y y)) 3))

B-redução Exemplo 3

- twice f x = f(f(x));
  - twice square 3 = 81
  - square in lambda calculus:
    $(\lambda y.\ (*\ y\ y))$
  - twice in lambda calculus:
    $(\lambda f.\ (\lambda x.\ f\ (f\ x)))$
  - twice square 3 =>
    $(((\lambda f.\ (\lambda x.\ f\ (f\ x)))\ (\lambda y.\ (*\ y\ y)))\ 3)$

$$(((\boxed{\lambda f.}\ (\boxed{\lambda x.\ f\ (f\ x)}))\ \boxed{(\lambda y.\ (*\ y\ y))})\ 3)$$
$$= ((\ (\lambda x.\ (\lambda y.\ (*\ y\ y))\ ((\lambda y.\ (*\ y\ y))\ x))\ )\ 3)$$
$$= (\ (\boxed{\lambda x.}\ \boxed{(\lambda y.\ (*\ y\ y))\ ((\lambda y.\ (*\ y\ y))\ x)})\ \ \boxed{3})$$
$$= (\ (\lambda y.\ (*\ y\ y))\ ((\lambda y.\ (*\ y\ y))\ 3)\ )$$
$$= (\lambda y.\ (*\ y\ y))\ ((\lambda y.\ (*\ y\ y))\ 3)$$

Removendo parênteses ...

B-redução Exemplo 3

- twice f x = f(f(x));
  - twice square 3 = 81
  - square in lambda calculus:
    (λy. (∗ y y))
  - twice in lambda calculus:
    (λf. (λx. f (f x)))
  - twice square 3 =>
    (((λf. (λx. f (f x))) (λy. (∗ y y))) 3)

(((λf. (λx. f (f x))) (λy. (∗ y y))) 3)
= (((λx. (λy. (* y y)) ((λy. (* y y)) x))) 3)
= ((λx. (λy. (* y y)) ((λy. (* y y)) x)) 3)
= ((λy. (* y y)) ((λy. (* y y)) 3))
= (λy. (* y y)) ((λy. (* y y)) 3)

B-redução Exemplo 3

- twice f x = f(f(x));
  - twice square 3 = 81
  - square in lambda calculus:
    (λy. (∗ y y))
  - twice in lambda calculus:
    (λf. (λx. f (f x)))
  - twice square 3 =>
    (((λf. (λx. f (f x))) (λy. (∗ y y))) 3)

$$
\begin{aligned}
&(((\boxed{\lambda f.}\ \boxed{(\lambda x.\ f\ (f\ x))})\ \boxed{(\lambda y.\ (*\ y\ y))})\ 3) \\
&= ((\ (\lambda x.\ (\lambda y.\ (*\ y\ y))\ ((\lambda y.\ (*\ y\ y))\ x))\ )\ 3) \\
&= (\ (\boxed{\lambda x.}\ \boxed{(\lambda y.\ (*\ y\ y))\ ((\lambda y.\ (*\ y\ y))\ x)})\ \ \boxed{3}) \\
&= (\ (\lambda y.\ (*\ y\ y))\ ((\lambda y.\ (*\ y\ y))\ 3)\ ) \\
&= (\lambda y.\ (*\ y\ y))\ ((\boxed{\lambda y.}\ \boxed{(*\ y\ y)})\ \boxed{3}) \\
&= (\lambda y.\ (*\ y\ y))\ ((*\ 3\ 3))
\end{aligned}
$$

B-redução Exemplo 3

- twice f x = f(f(x));
  - twice square 3 = 81
  - square in lambda calculus:
    (λy. (∗ y y))
  - twice in lambda calculus:
    (λf. (λx. f (f x)))
  - twice square 3 =>
    (((λf. (λx. f (f x))) (λy. (∗ y y))) 3)

(((λf. (λx. f (f x))) (λy. (∗ y y))) 3)
= (((λx. (λy. (* y y)) ((λy. (* y y)) x))) 3)
= ((λx. (λy. (* y y)) ((λy. (* y y)) x)) 3)
= ((λy. (* y y)) ((λy. (* y y)) 3))
= (λy. (* y y)) ((λy. (* y y)) 3)
= (λy. (* y y)) ((* 3 3))
= (λy. (* y y)) (* 3 3)

Removendo parênteses ...

B-redução Exemplo 3

- twice f x = f(f(x));
  - twice square 3 = 81
  - square in lambda calculus:
    (λy. (∗ y y))
  - twice in lambda calculus:
    (λf. (λx. f (f x)))
  - twice square 3 =>
    (((λf. (λx. f (f x))) (λy. (∗ y y))) 3)

$$(((\boxed{\lambda f.}\ \boxed{\lambda x.\ f\ (f\ x)})\ \boxed{(\lambda y.\ (*\ y\ y))})\ 3)$$
$$= ((\ (\lambda x.\ (\lambda y.\ (*\ y\ y))\ ((\lambda y.\ (*\ y\ y))\ x))\ )\ 3)$$
$$= (\ (\boxed{\lambda x.}\ \boxed{(\lambda y.\ (*\ y\ y))\ ((\lambda y.\ (*\ y\ y))\ x)})\ \ \boxed{3})$$
$$= (\ (\lambda y.\ (*\ y\ y))\ ((\lambda y.\ (*\ y\ y))\ 3)\ )$$
$$= (\lambda y.\ (*\ y\ y))\ ((\boxed{\lambda y.}\ \boxed{(*\ y\ y)})\ \boxed{3})$$
$$= (\lambda y.\ (*\ y\ y))\ ((*\ 3\ 3))$$
$$= (\lambda y.\ (*\ y\ y))\ \boxed{(*\ 3\ 3)}$$

Β-redução   Exemplo 3

- twice f x = f(f(x));
  - twice square 3 = 81
  - square in lambda calculus:
    (λy. (* y y))
  - twice in lambda calculus:
    (λf. (λx. f (f x)))
  - twice square 3 =>
    (((λf. (λx. f (f x))) (λy. (* y y))) 3)

(((λf. (λx. f (f x))) (λy. (* y y))) 3)
= (((λx. (λy. (* y y)) ((λy. (* y y)) x))) 3)
= ((λx. (λy. (* y y)) ((λy. (* y y)) x)) 3)
= ((λy. (* y y)) ((λy. (* y y)) 3))
= (λy. (* y y)) ((λy. (* y y)) 3)
= (λy. (* y y)) ((* 3 3))
= (λy. (* y y)) (* 3 3) = (λy. (* y y)) 9

B-redução   Exemplo 3

- twice f x = f(f(x));
  - twice square 3 = 81
  - square in lambda calculus:
    (λy. (∗ y y))
  - twice in lambda calculus:
    (λf. (λx. f (f x)))
  - twice square 3 =>
    (((λf. (λx. f (f x))) (λy. (∗ y y))) 3)

```
(((λf. (λx. f (f x))) (λy. (∗ y y))) 3)
= (( (λx. (λy. (* y y)) ((λy. (* y y)) x)) ) 3)
= ( (λx. (λy. (* y y)) ((λy. (* y y)) x)) 3)
= ( (λy. (* y y)) ((λy. (* y y)) 3) )
= (λy. (* y y)) ((λy. (* y y)) 3)
= (λy. (* y y)) ((* 3 3))
= (λy. (* y y)) (* 3 3) = (λy. (* y y)) 9
```

B-redução Exemplo 3

- twice f x = f(f(x));
  - twice square 3 = 81
  - square in lambda calculus:
    $(\lambda y.\ (*\ y\ y))$
  - twice in lambda calculus:
    $(\lambda f.\ (\lambda x.\ f\ (f\ x)))$
  - twice square 3 =>
    $(((\lambda f.\ (\lambda x.\ f\ (f\ x)))\ (\lambda y.\ (*\ y\ y)))\ 3)$

$$(((\lambda f.\ (\lambda x.\ f\ (f\ x)))\ (\lambda y.\ (*\ y\ y)))\ 3)$$
$$= ((\ (\lambda x.\ (\lambda y.\ (*\ y\ y))\ ((\lambda y.\ (*\ y\ y))\ x))\ )\ 3)$$
$$= (\ (\lambda x.\ (\lambda y.\ (*\ y\ y))\ ((\lambda y.\ (*\ y\ y))\ x))\ 3)$$
$$= (\ (\lambda y.\ (*\ y\ y))\ ((\lambda y.\ (*\ y\ y))\ 3)\ )$$
$$= (\lambda y.\ (*\ y\ y))\ ((\lambda y.\ (*\ y\ y))\ 3)$$
$$= (\lambda y.\ (*\ y\ y))\ ((*\ 3\ 3))$$
$$= (\lambda y.\ (*\ y\ y))\ (*\ 3\ 3) = (\lambda y.\ (*\ y\ y))\ 9$$
$$= (*\ 9\ 9) = 81$$

B-redução   Exemplo 4

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

B-redução   Exemplo 4

$$(\lambda x.\ +\ ((\lambda y.\ (\boxed{(\lambda x.\ *\ x\ y)\ 2}))\ x)\ y)$$

Β-redução Exemplo 4

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

$\Rightarrow (\lambda x. + ((\lambda y. (* \ 2\ y))\ x)\ y)$

B-redução   Exemplo 4

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

$$\Rightarrow (\lambda x.\ +\ ((\boxed{\lambda y.(*\ 2\ y))\ x})\ y)$$

B-redução Exemplo 4

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

$\Rightarrow (\lambda x.\ +\ ((\lambda y.\ (*\ 2\ y))\ x)\ y)$

$\Rightarrow (\lambda x.\ +\ ((*\ 2\ x))\ y)$

B-redução   Exemplo 4

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

$$\Rightarrow (\lambda x.\ +\ ((\lambda y.\ (*\ 2\ y))\ x)\ y)$$

$$\Rightarrow (\boxed{\lambda x.\ +\ ((*\ 2\ x))\ y})$$

B-redução Exemplo 4

$(\lambda x.\ +\ ((\lambda y.\ ((\lambda x.\ *\ x\ y)\ 2))\ x\ )\ y)$
$\Rightarrow (\lambda x.\ +\ ((\lambda y.\ (*\ 2\ y))\ x)\ y)$
$\Rightarrow (\lambda x.\ +\ ((*\ 2\ x))\ y)$
$\Rightarrow (\lambda x.\ +\ (*\ 2\ x)\ y)$

Ordem de Aplicação da β-redução

As regras de avaliação não especificam a ordem exata que uma expressão deve ser reduzida, uma possível ordem de avaliação é reduzir completamente o argumento antes de substituí-lo no corpo da função, essa avaliação e chamada **avaliação por valor** (*eager evaluation ou applicative-order*). Uma outra alternativa de avaliação é substituir o argumento sem avalia-lo, nesse caso o argumento será reduzido apenas se necessário, essa ordem de avaliação e chamada de **avaliação preguiçosa** ou **ordem normal** (*normal-order evaluation ou lazy evaluation*).

A avaliação de uma expressão pode resultar em uma expressão na forma normal ou a computação pode não terminar.

Teorema de Church e Rosser: *Se v é o resultado da avaliação de uma expressão M aplicando a ordem de avaliação preguiçosa, então qualquer que seja a ordem de avaliação aplicada, ou o resuldado da avaliação é v ou a avaliação falha (não termina). Se a avaliação de M não termina usando a ordem de avaliação preguiçosa a avaliação não termina usando qualquer ordem de avaliação.*

Ordem de Aplicação da β-redução

$$((\lambda x. (* \; x \; x)) \; (+ \; 2 \; 3))$$

- We may use pass by value or pass by name

Ordem de Aplicação da β-redução

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

- We may use pass by value or pass by name
  - **Pass by value**:

    $((\lambda x.\ (*\ x\ x))\ (+\ 2\ 3)) \Rightarrow ((\lambda x.\ (*\ x\ x))\ 5) \Rightarrow$
    $(*\ 5\ 5) \Rightarrow 25$

Ordem de Aplicação da β-redução

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

- We may use pass by value or pass by name

  – **Pass by name, “delayed evaluation”, “outermost evaluation”, “normal order”**

    ((λx. (* x x)) (+ 2 3)) => (* (+ 2 3) (+ 2 3)) =>
      (* 5 5) => 25

Ordem de Aplicação da β-redução

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

- We may use pass by value or pass by name
  - **Pass by value**:
    $((\lambda x.\ (* \ x\ x))\ (+\ 2\ 3)) \Rightarrow ((\lambda x.\ (* \ x\ x))\ 5) \Rightarrow$
    $(*\ 5\ 5) \Rightarrow 25$
  - **Pass by name, "delayed evaluation", "outermost evaluation", "normal order"**
    $((\lambda x.\ (* \ x\ x))\ (+\ 2\ 3)) \Rightarrow (*\ (+\ 2\ 3)\ (+\ 2\ 3)) \Rightarrow$
    $(*\ 5\ 5) \Rightarrow 25$

β-redução   Exemplo de aplicação por valor

– **Call-by-value**

$$((\lambda x.\ ((\lambda y.\ (*\ 2\ y))\ (+\ x\ y)))\ y)$$
$$= (\lambda x.\ ((\lambda y.\ (*\ 2\ y))\ (+\ x\ y)))\ y$$
$$= ((\lambda y.\ (*\ 2\ y))\ (+\ y\ y))$$
$$= (\lambda y.\ (*\ 2\ y))\ (+\ y\ y)$$
$$= (\lambda y.\ (*\ 2\ y))\ 2y$$
$$= (*\ 2\ 2y) = 4y$$

β-redução   Exemplo de aplicação por nome

– Call-by-name

```
((λx. ((λy. (* 2 y)) (+ x y))) y)
= (λx. ((λy. (* 2 y)) (+ x y))) y
= ((λy. (* 2 y)) (+ y y))
= (λy. (* 2 y)) (+ y y)
= (* 2 (+ y y) )
```

<- Here is the difference

```
= (* 2 2y) = 4y
```

α-conversão

Considere as duas abstrações lambda:

```
(λx. + x 1)
(λy. + y 1)
```

Claramente as duas abstrações acima são equivalentes e uma **α-conversão** nos permite mudar o nome do parâmetro formal de uma abstração lambda.

Então:

$$(\lambda x.\ +\ x\ 1) \underset{\alpha}{\leftrightarrow} (\lambda y.\ +\ y\ 1)$$

η-redução

Considere as duas expressões:

```
(λx. + 1 x)
(+  1)
```

Ambas tem o mesmo comportamento quando aplicadas à argumento: adicionam 1 ao argumento.
Uma **η–redução** é uma regra expressando tal equivalência:

$$(\lambda x.\ +\ 1\ x) \underset{\eta}{\leftrightarrow} (+\ 1)$$

η-redução

Considere as duas expressões:

```
(λx. + 1 x)
(+ 1)
```

Ambas tem o mesmo comportamento quando aplicadas à argumento: adicionam 1 ao argumento.

Uma **η–redução** é uma regra expressando tal equivalência:

$$(\lambda x.\ +\ 1\ x) \underset{\eta}{\leftrightarrow} (+\ 1)$$

Operações em expressões Lambda

α–conversão (renomeação)	$\lambda x . e \leftrightarrow \lambda y . e$	*Quando y não é uma variável livre em e.*
β–redução (aplicação)	$(\lambda x . e1)\ e2 \rightarrow [\,e2/x\,]\ e1$	*Operações de renomeação e substituição.*
η–conversão	$\lambda x . e\ x \rightarrow e$	*Elimina uma abstração λ mais complexa.*

Exercícios

Reduza as expressões avaliando-as:

```
1)  (λ x.2*x + 1)  3
2)  (λ xy.x-y)  5 7
3)  (λ yx.x-y)  5 7
4)  (λ xy.x-y)  (λz.z/2)
5)  (λ xy.x-y) ((λz.z/2) 6) 1
6)  (λ x.λ y. - x y)  9  4
7)  (λ x.xx)  (λ y.y)
```

Expressões Lambda em Haskell

Da mesma maneira que um número inteiro, uma string ou uma tupla podem ser escritos sem ser nomeados, uma função também pode ser escrita em Haskell sem associá-la a um nome, através de uma expressão lambda.

O termo lambda provém do cálculo lambda, introduzido por Church.

No cálculo lambda, as expressões são introduzidas usando a letra grega λ. Em Haskell usa-se o caracter \, que se assemelha-se um pouco com λ.

Expressões Lambda em Haskell

Expressão lambda é uma função anônima (sem nome), formada por uma seqüência de padrões representando os argumentos da função, e um corpo que especifica como o resultado pode ser calculado usando os argumentos:

$$\backslash\text{padrão}_1 \ldots \text{padrão}_n \text{ -> expressao}$$

para representar a Abstração Lambda:

$$\lambda x_1.\lambda x_2. \ldots \lambda x_n. <exp>$$

Ex: Função anônima que calcula o dobro de um número:

```
\x -> x + x
```

Funções anônimas em Haskell: exemplos

Função anônima que mapeia um número x a 2x + 1:

\x -> 2*x + 1

Função anônima que calcula o fatorial de um número:

\n -> product [1..n]

Função anônima que recebe 3 argumentos em uma tuple e calcula a soma dos três:

\(a,b,c) -> a + b + c

Função anônima que calcula a área de um círculo, a partir do valor do seu raio ($F=\pi * r^2$):

\r -> pi * r * r

Nomeando funções em Haskell: exemplos

```
f = \x -> 2*x + 1
somaTrio = \(x,y,z) -> x + y + z
fatorial =  \n -> product [1..n]
areaCirc = \r -> pi * r * r
```

Essas definições são equivalentes a:

```
f x = 2*x + 1
somaTrio (x,y,z) = x + y + z
fatorial n = product [1..n]
areaCirc r = pi * r * r
```

Uso de expressões Lambda em Haskell

Apesar de não terem um nome, funções anônimas podem ser usadas da mesma forma que outras funções:

```
(\x -> 2*x + 1) 8
↝ 17
```

```
(\a -> (a,2*a,3*a)) 5
↝ (5,10,15)
```

```
(\x y -> sqrt (x*x + y*y)) 3 4
↝ 5.0
```

```
(\xs -> let n = div (length xs) 2
        in (take n xs,drop n xs))
"Bom dia"
↝ ("Bom", " dia")
```

```
(\(x1,y1) (x2,y2) -> sqrt((x2-x1)^2 + (y2-y1)^2)) (6,7) (9,11)
↝ 5.0
```

Exemplo de aplicação

a) Implementar uma função que calcula o sucessor de um número inteiro usando expressão lambda ($\lambda x.\ x+1$).

b) Em seguida, definir uma função duasVezes para aplicar uma função nela mesma.

c) Finalmente, construir uma função para mapear a aplicação de duasVezes sobre uma lista de inteiros.

Exemplo de aplicação

a) Implementar uma função que calcula o sucessor de um número inteiro usando expressão lambda (λx. x+1).

b) Em seguida, definir uma função duasVezes para aplicar uma função nela mesma.

c) Finalmente, construir uma função para mapear a aplicação de duasVezes sobre uma lista de inteiros.

```
sucessor::(Int -> Int)
sucessor = \x -> x + 1

Main> sucessor 1
2
Main> sucessor 5
6
```

Exemplo de aplicação

a) Implementar uma função que calcula o sucessor de um número inteiro usando expressão lambda (λx. x+1).

b) Em seguida, definir uma função duasVezes para aplicar uma função nela mesma.

c) Finalmente, construir uma função para mapear a aplicação de duasVezes sobre uma lista de inteiros.

```
sucessor::(Int -> Int)
sucessor = \x -> x + 1

duasVezes :: (a->a) ->a ->a
duasVezes f x = f (f x)


Main> duasVezes sucessor 5
7
Main> duasVezes sucessor 100
102
```

Exemplo de aplicação

a) Implementar uma função que calcula o sucessor de um número inteiro usando expressão lambda (λx. x+1).

b) Em seguida, definir uma função duasVezes para aplicar uma função nela mesma.

c) Finalmente, construir uma função para mapear a aplicação de duasVezes sobre uma lista de inteiros.

```
sucessor::(Int -> Int)
sucessor = \x -> x + 1

duasVezes :: (a->a) ->a ->a
duasVezes f x = f (f x)

mapear::(a->b)->[a]->[b]
mapear f [] = []
mapear f (x:xs) = (f x): mapear f xs
```

Exemplo de aplicação

```
Main> mapear sucessor [1,3,5]
[2,4,6]

Main> mapear (duasVezes sucessor) [1,3,5]
[3,5,7]

Main> mapear (duasVezes (+ 1)) [1,3,5]
[3,5,7]

Main> mapear (duasVezes (\x -> x * x)) [3,4,5]
[81,256,625]

Main> mapear (\y -> y ++ y) ["na","ta","la"]
["nana","tata","lala"]

Main> mapear (duasVezes (\y -> y ++ y)) ["na","ta","la"]
["nananana","tatatata","lalalala"]
```

Exemplo de aplicação

```
sucessor::(Int -> Int)
sucessor = \x -> x + 1

duasVezes :: (a->a) ->a ->a
duasVezes f x = f (f x)

mapear::(a->b)->[a]->[b]
mapear f [] = []
mapear f (x:xs) = (f x): mapear f xs
```

```
Main> mapear sucessor [1,3,5]
[2,4,6]

Main> mapear (duasVezes sucessor) [1,3,5]
[3,5,7]

Main> mapear (duasVezes (+ 1)) [1,3,5]
[3,5,7]

Main> mapear (duasVezes (\x -> x * x)) [3,4,5]
[81,256,625]

Main> mapear (\y -> y ++ y) ["na","ta","la"]
["nana","tata","lala"]

Main> mapear (duasVezes (\y -> y ++ y)) ["na","ta","la"]
["nananana","tatatata","lalalala"]
```

Exercícios

Reduza as expressões avaliando-as e escreva o código correspondente em Haskell, testando no ambiente

```
1)  (λ x.2*x + 1) 3
2)  (λ xy.x-y) 5 7
3)  (λ yx.x-y) 5 7
4)  (λ xy.x-y) (λz.z/2)
5)  (λ xy.x-y) ((λz.z/2) 6) 1
6)  (λ x.λ y. - x y) 9 4
```