

# Lambda *Calculus*

## Definição

**Prof. Edson Alves**

Faculdade UnB Gama

2020

# Sumário

1. Introdução
2. Definição do cálculo  $\lambda$

## Características do cálculo $\lambda$

- ▶ O cálculo  $\lambda$  ( $\lambda$  *calculus*) pode ser chamada “a menor linguagem de programação do mundo”
- ▶ Ele consiste apenas em uma regra de transformação e um esquema de definição de funções
- ▶ Foi proposto do Alonzo Church na década de 1930, como uma maneira de formalizar a noção de computabilidade
- ▶ Qualquer função computável pode ser expressa e avaliada através do cálculo  $\lambda$ , de modo que ele é equivalente às máquinas de Turin
- ▶ Ao contrário das máquinas de Turin, o foco é o uso das regras de transformações, sendo mais próximo do software do que do hardware

# Cálculo $\lambda$

## Termos- $\lambda$

O conjunto  $\Lambda$  dos termos- $\lambda$  (ou expressões- $\lambda$ , ou simplesmente lambdas) é definido por meio de um conjunto de variáveis  $V$  através das regras de aplicação e abstração, dadas a seguir:

1.  $x \in V \Rightarrow x \in \Lambda$  (expressão)
2.  $M, N \in V \Rightarrow MN \in \Lambda$  (aplicação)
3.  $M \in \Lambda, x \in V \Rightarrow \lambda x.M$  (abstração)

**Observação:** informalmente, a aplicação equivale ao cálculo da função  $M$  com argumento  $N$ , isto é  $M(N)$ ; a abstração corresponde a definição da função  $f(x) = M$ .

## Exemplos de termos- $\lambda$

1. O termo- $\lambda$  mais simples possível é composto por uma única variável (por exemplo,  $x$ )
2. A função identidade  $\lambda x.x$  é um exemplo de abstração
3. Parêntesis podem ser utilizados para clarificar uma expressão, ou para remover ambiguidades
4. O termo  $(\lambda x.x)y$  é a aplicação da função identidade ao termo  $y$
5. A aplicação é associativa à esquerda:

$$M_1 M_2 \dots M_N = (((M_1 M_2) M_3) \dots M_N)$$

6. O termo  $\lambda y.(\lambda x.M)$  equivale a uma função de duas variáveis
7. Uma notação alternativa para o termo anterior é

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

8. A abstração é associativa à direita:

$$\lambda x_1 x_2 \dots x_N.M = \lambda x_1.(\lambda x_2.(\dots \lambda x_N.M))$$

## Variáveis livres e atadas (*bound*)

- ▶ A abstração  $\lambda x.M$  une (ata, *to bind*) a variável livre  $x$  ao termo (expressão) lambda  $M$
- ▶ Uma variável não precedida por um símbolo  $\lambda$  que a une a uma expressão é denominada variável **livre**
- ▶ Na expressão

$$\lambda x.xy$$

a variável  $x$  é atada e a variável  $y$  é livre

- ▶ Uma mesma variável pode ser livre e atada em uma mesma expressão. Por exemplo, na expressão

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

a variável  $y$  é livre no termo entre parêntesis à esquerda, e atada no termo da direita

# Substituições

## Substituição

A **substituição** de todas as ocorrências da variável livre  $x$  por  $N$  em  $M$ , cuja notação é  $M[x := N]$ , é definida por

- i.  $x[x := N] = N$
- ii.  $y[x := N] = y$ , se  $y \neq x$
- iii.  $(M_1 M_2)[x := N] = (M_1[x := N])(M_2[x := N])$
- iv.  $(\lambda y. M_1)[x := N] = \lambda y. (M_1[x := N])$

## Exemplos de substituição

1. Exemplo de substituição pela regra 3:

$$((\lambda x.xyz)(\lambda y.xzy))[z := N] = (\lambda x.xyN)(\lambda y.xNy)$$

2. Exemplo de substituição pela regra 4:

$$(\lambda x.xy)[y := N] = \lambda x.xN$$

3. Exemplo de substituição pelas regras 2 e 4:

$$(\lambda x.xy)[z := N] = \lambda x.xy$$

4.  $(\lambda x.xy)[x := N]$  não é uma expressão lambda válida, pois as substituições devem ser feitas em termos de variáveis livres, e  $x$  é atada na expressão entre parêntesis



# Reduções

## Axiomas de Redução

1. A conversão- $\alpha$  permite a troca das variáveis atadas de uma expressão, evitando colisões de nomes:

$$\lambda x.M \equiv_{\alpha} \lambda y.(M[x := y])$$

2. A redução- $\beta$  associa a aplicação com a substituição:

$$(\lambda x.M)N \equiv_{\beta} M[x := N]$$

3. A conversão- $\eta$  elimina de redundâncias em expressões cujo propósito é apenas passar um argumento para uma função:

$$(\lambda x.Mx) \equiv_{\eta} M,$$

se  $x$  não é uma variável livre em  $M$ .

**Observação:** Se a expressão- $\lambda$   $N$  pode ser obtida através de sucessivas aplicações dos três axiomas acima ao termo  $M$ , escreveremos  $M \equiv N$ .

## Exemplos de aplicação dos axiomas de redução

### 1. Aplicação da função identidade (redução- $\beta$ )

$$(\lambda x.x)y \equiv x[x := y] \equiv y$$

### 2. Aplicação em função de duas variáveis (redução- $\beta$ ):

$$\begin{aligned}(\lambda xy.yx)MN &\equiv (\lambda x.(\lambda y.yx))MN \\ &\equiv ((\lambda y.yx)[x := M])N \\ &\equiv (\lambda y.yM)N \\ &\equiv (yM)[y := N] \\ &\equiv NM\end{aligned}$$

### 3. Eliminação de redundância (conversão- $\eta$ ):

$$(\lambda x.zyx) \equiv zy$$

## Exemplos de aplicação dos axiomas de redução

4. Uso da conversão- $\alpha$  para evitar colisão de nomes, pois a variável  $y$ , que irá substituir a variável livre  $x$  no termo  $\lambda y.yx$ , tem mesmo nome que a variável atada  $y$  :

$$\begin{aligned}(\lambda x.(\lambda y.xy))y &\equiv (\lambda x.(\lambda z.xz))y \\ &\equiv (\lambda z.xz)[x := y] \\ &\equiv \lambda z.yz\end{aligned}$$

Observe que, sem o uso da conversão- $\alpha$ , a aplicação resultaria em  $(\lambda y.yy)$ , termo que não é equivalente ao resultado correto.

## Exemplos de aplicação dos axiomas de redução

5. Outro exemplo de que demanda o uso de conversão- $\alpha$ :

$$\begin{aligned}(\lambda x.(\lambda y.(x\lambda x.xy)))y &\equiv (\lambda x.(\lambda z.(x\lambda x.xz)))y \\ &\equiv (\lambda z.(x\lambda x.xz))[x := y] \\ &\equiv (\lambda z.(y\lambda x.xz))\end{aligned}$$

Aqui novamente a conversão- $\alpha$  foi usada por que  $y$  é uma variável atada na expressão  $\lambda y.(x\lambda x.xy)$ . Além disso, observe que somente a ocorrência livre de  $x$  é substituída, conforme a regra de substituição apresentada anteriormente.

# Combinadores e Igualdade Extensional

## Combinadores

- (a) O conjunto das variáveis livres  $FV(M)$  de  $M$  é definido por
- i.  $FV(x) = \{x\}$
  - ii.  $FV(MN) = FV(M) \cup FV(N)$
  - iii.  $FV(\lambda x.M) = FV(M) - \{x\}$
- (b)  $M$  é um termo fechado, ou **combinador**, se  $FV(M) = \emptyset$

## Igualdade Extensional

Duas expressões  $\lambda E_1, E_2$  são **extensionalmente iguais** se,  $\forall x \in \Lambda$ ,  $E_1x \equiv E_2x$ .

# Combinadores padrão e base S-K

## Combinadores padrão

Os combinadores padrão são enumerados a seguir:

1.  $\mathbf{I} \equiv \lambda x.x$
2.  $\mathbf{K} \equiv \lambda xy.x$
3.  $\mathbf{K}_* \equiv \lambda xy.y$
4.  $\mathbf{S} \equiv \lambda xyz.xz(yz)$

## Completude da base S-K

Dado uma expressão lambda  $E$ , é possível gerar um novo combinador  $C$  a partir dos combinadores  $\mathbf{S}$  e  $\mathbf{K}$  de tal modo que  $E$  e  $C$  são extensionalmente iguais.

## Exemplo da completude de S-K

A identidade **I** é extensionalmente igual ao combinador **SKK**:

$$\begin{aligned} ((\mathbf{SKK})x) &\equiv (\mathbf{SKK}x) \\ &\equiv (\mathbf{K}x(\mathbf{K}x)) \\ &\equiv x \\ &\equiv \mathbf{I}x \end{aligned}$$

# Referências

1. **BARENDREGT**, Henk; **BARENDSSEN**, Erik. *Introduction to Lambda Calculus*, March 2000.
2. **LOCZEWSKI**, Georg P. *A++ and the Lambda Calculus: Principles of Functional Programming*, tredition, 2018.
3. **ROJAS**, Raúl. *A Tutorial Introduction to the Lambda Calculus*, FU Berlin, WS-97/98.
4. Wikipédia. [Combinatory logic](#), acesso em 07/01/2020.
5. Wikipédia. [Lambda calculus](#), acesso em 03/01/2020.