

# Cálculo Lambda

Linguagens de Programação

2016.2017

*Teresa Gonçalves*  
[tcg@uevora.pt](mailto:tcg@uevora.pt)

Departamento de Informática, ECT-UÉ

# Sumário

Funções e expressões funcionais

Expressões lambda

Redução, confluência e formas normais

Programação em cálculo- $\lambda$

# Cálculo Lambda

## História

Alonzo Church, 1936

## O que é?

Sistema matemático

Permite ilustrar conceitos importantes de LP de forma simples e pura



## Tradicionalmente

Notação para definição de funções

Sistema de prova para equações entre expressões

Conjunto de regras de cálculo (**redução**)

# Cálculo Lambda

## Significado

### Lambda

Letra Grega  $\lambda$

### Cálculo

Redução utilizada para **calcular** o resultado da aplicação duma função a um ou mais argumentos

Forma de avaliação simbólica de expressões

## Conceitos de LP

### Parametrização

Através de expressões funcionais

### Ligação (*binding*)

Através de declarações

### Estratégias de avaliação

# **Funções e expressões funcionais**

# Cálculo- $\lambda$ puro

## O que é uma função?

Uma regra para determinar um valor a partir de um ou mais argumentos

Exemplos

$$f(x) = x^2 + 3$$

$$g(x, y) = \sqrt{x^2 + y^2}$$

## cálculo- $\lambda$

2 operadores

Definição da função (abstração- $\lambda$ )

Aplicação

Permite definir funções do tipo

$$h(x) = f(g(x))$$

# Abstração Lambda

## O que é?

Operador que define uma função

$\lambda x.M$

Se  $M$  for uma expressão, é a função obtida ao tratar  $M$  como função da variável  $x$

## Exemplo

Função identidade

$\lambda x.x$

Equivalente a “ $I(x)=x$ ”

obriga dar um nome à função

# Aplicação

## O que é?

Operador que aplica da função a um argumento

$(\lambda x.M_1) M_2$

Se  $M_2$  for uma expressão, é aplicação da função  $(\lambda x.M_1)$  a  $M_2$

## Exemplos

$(\lambda x.x) M$

Aplicação da função identidade à expressão  $M$

O resultado desta expressão é  $M$

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

Dadas duas funções **f** e **g**, esta função produz a composição  $\lambda x.f(g x)$  de **f** e **g**



# Expressões lambda

# Sintaxe duma expressão lambda

## Assume-se

um conj. infinito  $V$  de **variáveis**, sendo  $x, y, z, \dots$  variáveis arbitrárias

## Gramática BNF

$$\langle M \rangle \rightarrow x \mid \langle M \rangle \langle M \rangle \mid \lambda x. \langle M \rangle$$

$x$  pode ser qualquer variável (elemento de  $V$ )

## Significado

$x$  refere uma variável (determinada pelo contexto)

$M_1 M_2$  é a aplicação de  $M_1$  ao argumento  $M_2$

$\lambda x. M$  é a função que, dado o argumento  $x$ , devolve o valor  $M$

# Termo lambda

É uma expressão de cálculo- $\lambda$

## Exemplos

$\lambda x.x$  Uma abstracção- $\lambda$  designada função identidade

$\lambda x.f(g\ x)$  Outra abstracção- $\lambda$

$(\lambda x.x)\ 5$  Uma aplicação

# Convenções sintáticas

O âmbito de  $\lambda$ , numa abstração- $\lambda$ , estende-se o mais para a direita possível

Exemplo

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

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

A aplicação associa à esquerda

Exemplo

$$x y z = (x y) z$$

$$x y z \neq x (y z)$$

# Ocorrências de variáveis

## Livre

Ocorrência fora do âmbito do operador de ligação

## Ligadora

Ocorrência onde a variável se torna ligada

## Ligada

Outras ocorrências não livres

## Exemplo

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

x: 1ª ocorrência → ocorrência ligadora

2ª ocorrência → ocorrência ligada

y: 1ª ocorrência → ocorrência ligadora

2ª ocorrência → ocorrência ligada

3ª ocorrência → ocorrência livre

# Ligação de uma variável

## Ocorrência livre

A variável não foi declarada na expressão

Não é possível avaliar a expressão sem colocá-la noutra (maior) que associe um valor à variável

Exemplo

Na expressão  $x+3$ ,  $x$  é livre

## Ocorrência ligada

É uma ocorrência não livre

$\lambda$  é o operador de ligação

Porque liga uma variável num âmbito específico

Exemplo

Na expressão  $\lambda x.M$ ,  $x$  é ligada

$M$  é o âmbito da ligação  $\lambda x$

# Variável ligada- $\lambda$

## Propriedade

É possível renomear uma variável ligada- $\lambda$   $x$  para  $y$  sem alterar o significado da expressão

Exemplo

$\lambda x.x$  define a mesma função que  $\lambda y.y$

## Expressões $\alpha$ -equivalentes

São expressões que diferem apenas nos nomes das variáveis ligadas

Escreve-se

$$\lambda x.x =_{\alpha} \lambda y.y$$

# Conj. variáveis livres

## VL(M)

conj. de variáveis livres na expressão M

definido por indução na estrutura das expressões

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

$$VL(M N) = VL(M) \cup VL(N)$$

$$VL(\lambda x.M) = VL(M) - \{x\}, \text{ onde } - \text{ significa diferença conj.}$$

## Exemplos

$$VL(\lambda x.x) = \emptyset$$

$$VL(\lambda f.\lambda x.f(g(x))) = \{g\}$$



# Sistema de prova do cálculo- $\lambda$

## Axiomas básicos

equivalência- $\alpha$

Equivalência de expressões com variáveis ligadas

equivalência- $\beta$

Equivalência de expressões através do cálculo do valor de uma aplicação por substituição

# Equivalência- $\alpha$

$$\lambda x.M = \lambda y.[y/x]M$$

$[y/x]M$  é o resultado de substituir as ocorrências livres de  $x$  em  $M$  por  $y$   
 $y$  não pode aparecer em  $M$

## Exemplo

$$(\lambda x.(\lambda x.x) x) x =_{\alpha} (\lambda y.(\lambda z.z) y) x$$

# Equivalência- $\beta$

$$(\lambda x.M)N = [N/x]M$$

$\lambda x.M$  é a função obtida ao tratar  $M$  como função de  $x$

$[N/x] M$  : o valor da aplicação em  $N$  obtém-se, substituindo as ocorrências livres de  $x$  em  $M$  por  $N$

## Exemplo

$$(\lambda f.f x) (\lambda y.y) =_{\beta} (\lambda y.y) x =_{\beta} x$$

# Substituição

$[N/x] M$

Resultado da substituição de  $x$  por  $N$  em  $M$  ( $N$  substitui  $x$  em  $M$ )

## Regras

Definidas por indução na estrutura de  $M$

$$[N/x] x = N$$

$$[N/x] y = y \quad y \neq x$$

$$[N/x] (M_1 M_2) = [N/x] M_1 [N/x] M_2$$

$$[N/x] (\lambda x. M) = \lambda x. M$$

$$[N/x] (\lambda y. M) = \lambda y. ([N/x] M) \quad y \neq x, y \notin VL(N)$$

# Colisão de variáveis

## Quando acontece?

Sempre que na substituição  $[N/x] M$ , as ligações- $\lambda$  em  $M$  têm o mesmo nome que variáveis livres de  $N$

## Como evitar?

Renomear as variáveis ligadas em  $(\lambda x.M) N$

Todas as variáveis ligadas devem ser diferentes umas das outras e das variáveis livres

## Exemplo

$$(\lambda f.\lambda x.f(fx))(\lambda y.y + x)$$

# **Redução, forma normal e confluência**

# Redução- $\beta$

## O que é?

equivalência- $\beta$  que permite avaliar uma função

Para indicar a direção, utiliza  $\rightarrow$  em vez de  $=$

$$(\lambda x.M) N \rightarrow [N/x] M$$

## $M \rightarrow N$

$M$  reduz- $\beta$  para  $N$

Quando um subtermo de  $M$  reduz- $\beta$  para criar  $N$

## Redex (reduction expression)

É um termo do tipo  $(\lambda x.M) N$

# Forma normal

## O que é?

Uma expressão que não é possível reduzir mais

## Exemplos

$\lambda z.zz$

$(\lambda f.\lambda x.f(fx))(\lambda y.y + 1)2$   
 $\rightarrow \lambda x.((\lambda y.y + 1)((\lambda y.y + 1)x))2$   
 $\rightarrow (\lambda x.(\lambda y.y + 1)(x + 1))2$   
 $\rightarrow (\lambda x.x + 1 + 1)2$   
 $\rightarrow (2 + 1 + 1) \quad \leftarrow \text{Forma normal}$

Se adicionarmos regra cálculo da adição

$2 + 1 + 1$   
 $\rightarrow 3 + 1$   
 $\rightarrow 4$



# Reduções alternativas

Podem existir várias reduções alternativas

Exemplo

$$\begin{aligned} & (\lambda f. \lambda z. f(fz))(\lambda y. y + x) \\ \rightarrow & \lambda z. (\lambda y. y + x)((\lambda y. y + x)z) \\ \rightarrow & \lambda z. (\lambda y. y + x)(z + x) \\ \rightarrow & \lambda z. (z + x) + x \end{aligned}$$

$$\begin{aligned} & (\lambda f. \lambda z. f(fz))(\lambda y. y + x) \\ \rightarrow & \lambda z. (\lambda y. y + x)((\lambda y. y + x)z) \\ \rightarrow & \lambda z. ((\lambda y. y + x)z) + x \\ \rightarrow & \lambda z. (z + x) + x \end{aligned}$$

# Reduções intermináveis

Existem termos onde é possível reduzir sempre (sem nunca chegar à forma normal)

Exemplo

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

# Confluência

## O que é?

Se existir, a forma normal é única

## O que significa?

Se

M puder ser reduzida para uma forma normal

Então

Existe uma única forma normal de M, independente/ da ordem de avaliação das sub-expressões

**A ordem de avaliação não afecta o resultado final da expressão!**

# Estratégia de redução

É uma regra para escolher redexes

## Estratégias

Call-by-name (ou ordem normal de redução)

Escolhe o redex mais exterior, mais à esquerda

Call-by-value

Escolhe o redex mais interior, mais à esquerda

## Exemplo

Call-by-name

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

Call-by-value

$(\lambda x \lambda z. \underline{(\lambda x. x)} z (yz)) (\lambda x. x)$

# Garantia de chegada à forma normal

Call-by-name: sim

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

Call-by-value: não

$$\begin{aligned} & (\lambda y. z) ((\lambda x. xx)(\lambda x. xx)) \\ \rightarrow & (\lambda y. z) ((\lambda x. xx)(\lambda x. xx)) \\ \rightarrow & (\lambda y. z) ((\lambda x. xx)(\lambda x. xx)) \\ \rightarrow & \dots \end{aligned}$$

# Rapidez de chegada à forma normal

Call-by-value: mais rápido

$$\begin{aligned} & (\lambda x.xx)((\lambda y.y)(\lambda z.z)) \\ \rightarrow & (\lambda x.xx)(\lambda z.z) \\ \rightarrow & (\lambda z.z)(\lambda z.z) \\ \rightarrow & (\lambda z.z) \end{aligned}$$

Call-by-name: mais lento

$$\begin{aligned} & (\lambda x.xx)((\lambda y.y)(\lambda z.z)) \\ \rightarrow & ((\lambda y.y)(\lambda z.z)) ((\lambda y.y)(\lambda z.z)) \\ \rightarrow & (\lambda z.z) ((\lambda y.y)(\lambda z.z)) \\ \rightarrow & (\lambda y.y)(\lambda z.z) \\ \rightarrow & (\lambda z.z) \end{aligned}$$

# Computação em cálculo lambda

# Declaração let

## Construção que permite fazer declarações locais

$let\ x = M\ in\ N$

## **let** é açúcar sintático

$let\ x = M\ in\ N \Leftrightarrow (\lambda x. N)\ M$

Não adiciona poder ao cálculo, mas torna-o mais “amigável”

## Exemplo

$let\ compose = \lambda f. \lambda g. \lambda x. f\ (g\ x)\ in\ compose\ h\ h$

$\equiv_{\beta}$

$\lambda x. h\ (h\ x) \leftarrow$  composição de  $h$  com  $h$



# Múltiplas variáveis

## cálculo- $\lambda$

permite tratar uma expressão  $M$  como função de qualquer variável  $x$

### Tratar $M$ como função de 2 variáveis $x$ e $y$

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

Função de um único argumento que, quando aplicada, devolve uma segunda função que aceita um 2º argumento

### Exemplo

$f(g,x)=g(x)$

argumento: par  $(g,x)$

$f_{\text{curry}}=\lambda g.\lambda x.g\ x$

argumento:  $g$

## Técnica que simula funções de várias variáveis

$g$  é uma versão *curried* de  $f$  se

$$f(x_1, x_2, \dots, x_k) = g\ x_1\ x_2 \dots x_k$$

A função  $f$  tem  $k$  argumentos que  $g$  aceita à vez

## Útil para avaliação parcial

quando um dos argumentos não está disponível

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

Uma função que multiplica o seu argumento  $y$  por 2

## Convenção

$$\lambda x. \lambda y. M = \lambda xy. M$$

# Exemplos

## Operador \*

$$\lambda x. \lambda y. x * y$$
$$(\lambda x. \lambda y. x * y) 2 3 \rightarrow (\lambda y. 2 * y) 3 \rightarrow 2 * 3$$

## f(x, y)=x-y

$$\lambda x. \lambda y. x - y$$
$$f(4, 3)$$
$$(\lambda x. \lambda y. x - y) 4 3$$

# Abstração- $\lambda$ em ML e C

## ML

*fn x => corpo\_função;*

*$\lambda x$ .corpo\_função*

## C

*int f (int x) {return x};*

*invoca\_f;*

*let f = ( $\lambda x$ .x) in invoca\_f*

# Exemplo

```
int function f(x,y){  
    return 2*x+y;  
}  
int main(){  
    f(3,4);  
}
```

```
let f =  $\lambda xy. 2 \cdot x + y$  in  
    f 3 4
```

```
( $\lambda f. f$  3 4) ( $\lambda xy. 2 \cdot x + y$ )  $\rightarrow$   
( $\lambda xy. 2 \cdot x + y$ ) 3 4  $\rightarrow$   
( $\lambda y. 2 \cdot 3 + y$ ) 4  $\rightarrow$   
2*3+4
```

# Cálculo- $\lambda$ aplicado

## O que é?

Uma extensão do cálculo- $\lambda$  puro, que adiciona um conjunto de outras operações e tipos

## Exemplo

Adição dos operadores básicos de cálculo: +, -, \*, /...

**LP = cálculo- $\lambda$  aplicado**

**= cálculo- $\lambda$  puro + tipos de dados adicionais**