

Conceitos básicos de λ -calculus

1. Para cada uma das expressões seguintes:

- determina as que correspondem a um λ -termo.
- para cada uma que for um λ -termo e indica o conjunto das variáveis livres e o conjunto das variáveis ligadas.

1. $xyxx$
2. $\lambda x.\lambda x.x$
3. $y\lambda x.$
4. $x\lambda x.\lambda y.xy$
5. $xy\lambda x.x\lambda x.yxx$
6. $\lambda x.xy(\lambda y.xy)$

2. Sendo M e N dois λ -termos, $M \equiv_\alpha N$ se são o mesmo λ -termo ou se podem ser obtidos um do outro por mudança de variáveis ligadas em M ou em N . Por exemplo $(\lambda x.x)z \equiv_\alpha (\lambda y.y)z$ e $(\lambda x.x)z \not\equiv_\alpha (\lambda x.y)z$. Prova usando indução na estrutura dum λ -termo M que se $x \neq y$ e se $x \notin FV(L)$ então:

$$M[N/x][L/y] \equiv_\alpha M[L/y][N[L/y]/x]$$

3. Para cada um dos seguintes λ -termos reduz a forma normal β se possível:

1. xy
2. $(\lambda x.x)(yz)$
3. $(\lambda x.xy)(\lambda z.z)$
4. $(\lambda xy.xy)$
5. $(\lambda x.x\lambda x.xy)(yy)$
6. $(\lambda x.xx)(\lambda x.xx)$
7. $(\lambda x.xxx)(\lambda x.xxx)$
8. $(\lambda xy.y)((\lambda x.xx)(\lambda x.xx))b$
9. $M = AAx$ onde $A = \lambda axz.z(aax)$

4. O que está errado nas seguintes reduções?

1. $(\lambda xy.yx)y \rightarrow_\beta \lambda y.yy$
2. $(\lambda xx.xx)y \rightarrow_\beta \lambda x.yy$
3. $(\lambda x.xx)(\lambda y.ay)k \rightarrow_\beta (\lambda x.xx)ak \rightarrow_\beta akak$

5. Obtem reduções por ordem de aplicação e por ordem normal para cada um dos seguintes λ -termos:

- (a) $(\lambda x.xxx)(\lambda yz.yz)$
- (b) $A(AA)$ onde $A = \lambda x.xKSK$, $K = \lambda xy.x$ e $S = \lambda xyz.xz(yz)$.

6. Considerando os numerais de Church, para $n \geq 0$, $c_n = \lambda fx.\overbrace{f(f \cdots (fx) \cdots)}^{n \text{ f's}}$. Mostra que

- i. sendo $\mathbf{succ} = \lambda n f x. f(n f x)$, tem-se que $\mathbf{succ} c_n \equiv_{\beta} c_{n+1}$.
- ii. sendo $A_+ = \lambda m n. m \mathbf{succ} n$, tem-se que $A_+ c_m c_n \equiv_{\beta} c_{m+n}$.
- iii. sendo $A_* = \lambda n m f. n(m f)$, tem-se que $A_* c_n c_m \equiv_{\beta} c_{n*m}$.

7. Sendo

$\mathbf{true} = \lambda x y. x$
 $\mathbf{false} = \lambda x y. y$
 $\mathbf{pair} = \lambda x y f. f x y$
 $\mathbf{fst} = \lambda p. p \mathbf{true}$
 $\mathbf{snd} = \lambda p. p \mathbf{false}$

Mostra que

$\mathbf{fst}(\mathbf{pair} M N) \equiv_{\beta} M$
 $\mathbf{snd}(\mathbf{pair} M N) \equiv_{\beta} N$

- 8. Considerando ainda $\mathbf{if} = \lambda b x y. b x y$ define λ -termos que correspondam às as operações Booleanas $\neg$, $\wedge$ e $\vee$.
- 9. Considera as regras do sistema de tipos simples $\lambda \rightarrow$ dado:

$$\Gamma \vdash x : \alpha \quad \text{se } x : \alpha \in \Gamma \quad (Axioma)$$

$$\frac{\Gamma \vdash M : \alpha \rightarrow \beta \quad \Gamma \vdash N : \alpha}{\Gamma \vdash (MN) : \beta} (APP)$$

$$\frac{\Gamma, x : \alpha \vdash M : \beta}{\Gamma \vdash (\lambda x. M) : (\alpha \rightarrow \beta)} (ABS)$$

Usando essas regras constrói as inferências para as seguintes asserções:

- (a) $\vdash \lambda x y z. x(yz) : (a \rightarrow b) \rightarrow (c \rightarrow a) \rightarrow (c \rightarrow b)$
- (b) $\vdash \lambda x. x : (a \rightarrow a)$
- (c) $\vdash \lambda x. x : (a \rightarrow b) \rightarrow a \rightarrow b$
- (d) $\vdash \lambda x y. x : (a \rightarrow b \rightarrow a)$
- (e) $\vdash \lambda x y z. x z y : (a \rightarrow b \rightarrow c) \rightarrow b \rightarrow a \rightarrow c$
- (f) $\vdash \lambda x y. x y y : (a \rightarrow a \rightarrow b) \rightarrow a \rightarrow b$

Conceitos de Linguagens Funcionais

- 9. Identifica algumas características das linguagens funcionais na linguagem Haskell, em particular:
 - 1. definição de funções (recursivas ou não)

2. sistema de tipos
 3. encaixe de padrões
 4. mecanismos de passagem de parâmetros
 5. funções locais
 6. composição de funções
10. Resolve os problemas de implementação em Haskell das semânticas operacionais sugeridos na folha 2 (Exercício 12 e 19) e folha 4 (Exercício 7).
 11. Considera o `The Programming Languages Zoo`: <http://plzoo.andrej.com>. Compara a linguagem `miniml` com a `minihaskell` e indica algumas diferenças.