

Ficha 9

Semântica das Linguagens de Programação

2023/24

1. Usando a semântica de avaliação *call-by-name*, calcule o valor de cada uma das seguintes expressões:

- (a) $(\lambda u. \lambda l. \text{listcase } l \text{ of } (\text{nil}, \lambda h. \lambda t. u :: t)) (7+2) (1 :: 2 :: \text{nil})$
- (b) $\langle \text{True}, \lambda x. x + x, 5 * 3 \rangle. 2 ((\lambda y. y + 1) 3)$
- (c) $(\lambda f. \lambda u. \text{sumcase } (f u) \text{ of } (\lambda x. x, \lambda x. x * 2, \lambda x. 10)) (\lambda x. @2 x) ((\lambda y. y * y) 3)$

2. Considere a seguinte expressão B da linguagem de programação funcional estudada:

$(\lambda \langle x, y \rangle. \text{if } x \leq y \text{ then } x \text{ else } y) (\lambda x. x + 1) 3, (\lambda x. x - 1) 4$

- (a) Construa uma árvore de prova do juízo $\vdash B : \text{Int}$.
- (b) Calcule o valor de B , usando a semântica de avaliação *call-by-name* da linguagem (deve começar por traduzir o açúcar sintático utilizado).

3. Considere a seguinte expressão da linguagem de programação funcional estudada:

$\text{let posroot} \equiv \lambda a. \text{sumcase } a \text{ of } (\lambda x. \text{False}, \lambda y. (y.1) > 0)$
 $\text{in posroot } (@2 \langle 7 + 3, @1 \langle \rangle, @1 \langle \rangle)$

- (a) Calcule o seu valor, usando a semântica de avaliação *call-by-name*.
- (b) Construa uma árvore de prova do juízo

$a : \text{Unit} + \text{Int} \times \mathbf{A} \times \mathbf{A} \vdash \text{sumcase } a \text{ of } (\lambda x. \text{False}, \lambda y. (y.1) > 0) : \text{Bool}$

4. Usando a semântica de avaliação *call-by-name* da linguagem funcional que estudou, calcule o valor da seguinte expressão:

$\text{letrec comp} \equiv \lambda l. \text{listcase } l \text{ of } (0, \lambda h. \lambda t. 1 + \text{comp } t)$
 $\text{in comp } ((3 * 4) :: \text{nil})$

Construa também a árvore de tipificação desta expressão.

5. Considere a seguinte definição na linguagem de programação funcional estudada:

$\text{letrec map} \equiv \lambda f. \lambda l. \text{listcase } l \text{ of } (\text{nil}, \lambda h. \lambda t. f h :: \text{map } f t) \dots$

- (a) Apresente a avaliação CBN da expressão $\text{letrec map} \equiv \dots \text{ in map } (\lambda x. 2 * x) (7 :: \text{nil})$ até à sua forma canónica.

- (b) No âmbito da semântica CBN que estudou, a lista infinita de números naturais pode ser codificada por $\text{letrec map} \equiv \dots \text{ in } \text{rec } (\lambda l. 0 :: \text{map } (\lambda x. x + 1) l)$
 Apresente uma definição alternativa para a lista de naturais chamada *natlist*.

6. Considere a seguinte função que testa se duas listas são iguais

eqlist : List Int \rightarrow List Int \rightarrow Bool

e compare a avaliação CBN e CBV do seguinte programa

```
letrec eqlist  $\equiv$   $\lambda l_1. \lambda l_2.$ 
  listcase  $l_1$  of (
    listcase  $l_2$  of (True,  $\lambda h_2. \lambda t_2. \text{False}$ ),
       $\lambda h_1. \lambda t_1. \text{listcase } l_2 \text{ of } (\text{False}, \lambda h_2. \lambda t_2. h_1 = h_2 \wedge \text{eqlist } t_1 t_2)$ 
    )
  in eqlist (1 :: 2 :: 3 :: nil) (3 :: 2 :: 1 :: nil)
```

7. Construa uma extensão da linguagem de programação funcional, por forma a incluir um tipo de árvores binárias com informação nos nodos intermédios e nas folhas ("*full trees*").

Defina a sintaxe abstracta das novas expressões e do novo tipo, as novas regras de inferência de tipo, as novas formas canónicas da linguagem e as novas regras de avaliação "*call-by-name*".