Semântica das Linguagens de Programação

2º Teste (28 de Maio de 2021)

Questão 1 Considere os seguintes termos do lambda calculus puro:

$$\begin{array}{lcllcl} N & \equiv & (\lambda x.\lambda y.\lambda z.\, x\, z\, y) & \quad I & \equiv & (\lambda x.\, x) \\ K & \equiv & (\lambda a.\lambda b.\, a) & \quad Y & \equiv & (\lambda f.(\lambda x.f\,(x\,x))\,(\lambda x.f\,(x\,x))) \end{array}$$

1. Apresente a sequência da *ordem aplicativa* de redução até à forma normal da expressão

$$N\,K\,(K\,I)\,(I\,N)$$

   Sublinhe o β-redex que é seleccionado em cada passo de redução.

2. Considere agora o termo $K\,I\,(Y\,I)$. Como classifica este termo quanto normalização? Justifique a sua resposta.

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

$$\begin{array}{l} \mathsf{let}\ \ \mathsf{fun} \equiv \lambda x.\lambda l.\, \mathsf{listcase}\ l\ \mathsf{of}\ (\,0, \lambda h.\lambda t.\, \mathsf{if}\ h{>}0\ \mathsf{then}\ h{-}x\ \mathsf{else}\ h+x\,)\,, \\ \qquad \mathsf{lis} \equiv 5 :: (8{+}2) :: \mathsf{nil} \\ \mathsf{in}\ \ \mathsf{fun}\,((\lambda y.\, y{*}y)\,7)\,\mathsf{lis} \end{array}$$

1. Calcule, passo a passo, a sua forma canónica usando a semântica de avaliação *call-by-name*.

2. Construa uma árvore de prova do juízo

$$f : \mathsf{Bool} \to \mathsf{List\,Int} \to \mathsf{Int},\ l : \mathsf{List\,Int} \vdash \lambda x.\, \mathsf{liscase}\ l\ \mathsf{of}\ (\,0, \lambda h.\lambda t.\, h + (f\,x\,t)\,) : \mathsf{Bool} \to \mathsf{Int}$$

3. Considere a seguinte função escrita em Haskell:

```
alt [] = []
alt [x] = [x]
alt (x:y:xys) = x : alt xys
```

   (a) Defina uma função alt, equivalente a esta, na linguagem funcional que estudou.

   (b) Apresente uma expressão da linguagem funcional que estudou (versão *call-by-name*) equivalente à expressão Haskell `alt [10,15..]`

Questão 3 Pretende-se estender a linguagem de programação funcional, com um novo tipo de dados para representar o equivalente ao tipo `(Maybe a)` do Haskell:

```
data Maybe a = Nothing | Just a
```

1. Defina a sintaxe abstracta das novas expressões e do novo tipo, e as regras de inferência de tipo para as novas expressões.

2. Indique as novas formas canónicas da linguagem e as novas regras de avaliação *call-by-value*.

3. Defina uma função soma, de tipo Maybe Int → Maybe Int → Maybe Int, que calcula a soma de dois Maybe Int. Note que, se uma das parcelar for Nothing, a soma deve dar Nothing.