Semântica das Linguagens de Programação

2º Teste (29 de Maio de 2017)
Duração: 90 min

Questão 1 Considere os seguintes termos do lambda calculus puro:

$$\begin{array}{lcllcl} K & \equiv & (\lambda a.\lambda b.\, a) & \qquad F & \equiv & (\lambda a.\lambda b.\, b) \\ S & \equiv & (\lambda x.\lambda y.\lambda z.\, x\, z\, (y\, z)) & \qquad A & \equiv & (\lambda x.\, x\, x) \end{array}$$

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

$$S\, K\, K\, (F\, A\, A)$$

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

2. Acha que a expressão $S\, K\, K\, (F\, A\, A)$ é tipificável? Justifique a sua resposta.

Questão 2 Considere a seguinte expressão E

$$\begin{array}{l} \mathsf{let\ fun} \equiv \lambda a.\, \lambda l.\, \mathsf{listcase}\ l\ \mathsf{of}\ (\, a.1,\ \lambda h.\, \lambda t.\, h + a.2\,) \\ \mathsf{in\ fun}\ \langle 3, 7\rangle\ ((6-5) :: 9 :: \mathsf{nil}) \end{array}$$

1. Apresente, passo a passo, a sequência da avaliação "*call-by-value*" da expressão E, até à sua forma canónica.

2. Construa uma árvore de prova do juizo $\vdash \mathsf{E} : \mathbf{Int}$.

Questão 3 Pretende-se estender a linguagem de programação funcional, com um novo tipo de dados para representar uma sequência em que os elementos podem ser acrescentados à esquerda (inicio) ou à direita (fim) da sequência. Por exemplo, um equivalente em Haskell desta estrutura de dados seria:

```
data Seq a = Null | Inicio a (Seq a) | Fim (Seq a) 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-name"*.

3. Defina uma função que calcula o somatório de uma sequência de inteiros.