Cálculo de Programas

2.° ano
Lic. Ciências da Computação e Mestrado Integrado em Engenharia Informática
UNIVERSIDADE DO MINHO

2016/17 - Ficha nr.° 3

1. Considere as funções seguintes:

$$f = \langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle$$
$$g = \langle id \times \pi_1, \pi_2 \cdot \pi_2 \rangle$$

   (a) Identifique os tipos de f e g. Justifique a sua resposta construindo os respectivos diagramas.

   (b) Use as leis to cálculo de programas para mostrar que $f \cdot g = id$ se verifica.

2. Considere uma função d da qual apenas conhece duas propriedades:

$$\begin{cases} \pi_1 \cdot d = id \\ \pi_2 \cdot d = id \end{cases} \tag{F1}$$

Mostre (recorrendo directamente à propriedade universal correspondente) que essa função é, necessariamente, a mesma que em Programação Funcional escreveria da forma seguinte, em Haskell:

$$d :: a \to (a, a)$$
$$d\ a = (a, a)$$

(Esta função, que duplica um valor, designa-se habitualmente por função *diagonal*.)

3. Identifique os tipos das expressões $\langle \pi_1, \langle id, \pi_2 \rangle \rangle$ e $\langle id, \langle \pi_1, \pi_2 \rangle \rangle$. Como compara este último com o tipo da função d da alínea anterior?

4. Considere a função

$$\alpha = [\langle \underline{\mathsf{False}}, id \rangle, \langle \underline{\mathsf{True}}, id \rangle]$$

Determine o tipo de α e mostre, usando a propriedade *Universal*-$+$, que α se pode escrever em Haskell da forma seguinte:

$$\alpha\ (\mathsf{Left}\ a) = (\mathsf{False}, a)$$
$$\alpha\ (\mathsf{Right}\ a) = (\mathsf{True}, a)$$

5. Recorra à lei Eq-$+$ (entre várias outras) para mostrar que a definição que conhece da função factorial,

$$\mathit{fac}\ 0 = 1$$
$$\mathit{fac}\ (n+1) = (n+1) * \mathit{fac}\ n$$

é equivalente à equação seguinte

$$\mathit{fac} \cdot [\underline{0}, \mathsf{succ}] = [\underline{1}, \mathit{mul} \cdot \langle \mathsf{succ}, \mathit{fac} \rangle].$$

onde $\mathsf{succ}\ n = n + 1$ e $\mathit{mul}\ (a, b) = a * b$.

6. Considere a seguinte declaração de um tipo de *árvores binárias*, em Haskell:

   $\mathbf{data}\ \mathsf{LTree}\ a = \mathsf{Leaf}\ a \mid \mathsf{Fork}\ (\mathsf{LTree}\ a, \mathsf{LTree}\ a)$

   Indagando os tipos dos construtores Leaf e Fork, por exemplo no GHCi,

   ```
   *LTree> :t Fork
   Fork :: (LTree a, LTree a) -> LTree a
   *LTree> :t Leaf
   Leaf :: a -> LTree a
   ```

   é fácil desenhar o diagrama que explica a construção da função

   $$inLTree = [\mathsf{Leaf}, \mathsf{Fork}]$$

   Desenhe-o e calcule a sua inversa

   $outLTree :: \mathsf{LTree}\ a \to \mathsf{Either}\ a\ (\mathsf{LTree}\ a, \mathsf{LTree}\ a)$
   $outLTree\ (\mathsf{Leaf}\ a) = i_1\ a$
   $outLTree\ (\mathsf{Fork}\ (x, y)) = i_2\ (x, y)$

   resolvendo a equação

   $$outLTree \cdot inLTree \quad = \quad id$$

   em ordem a $outLTree$.

7. No Cálculo de Programas, as definições condicionais do tipo $h\ x = \mathbf{if}\ p\ x\ \mathbf{then}\ f\ x\ \mathbf{else}\ g\ x$ ou

   $h\ x$
   $\quad \mid p\ x = f\ x$
   $\quad \mid \mathbf{otherwise} = g\ x$

   são escritas usando o combinador ternário

   $p \to f, g$

   conhecido pelo nome de *condicional de McCarthy*, cuja definição é (ver formulário e diagrama em baixo)

   $$p \to f, g \quad = \quad [f, g] \cdot p?$$

   onde

   $$p?\ x = \begin{cases} i_1\ x\ \mathbf{if}\ p\ x \\ i_2\ x\ \mathbf{if}\ \neg p\ x \end{cases}$$

   $$\begin{tikzcd}
   & A \arrow[d, "p?"'] & \\
   A \arrow[r, "i_1"] \arrow[dr, "f"'] & A + A \arrow[d, "{[f, g]}"] & A \arrow[l, "i_2"'] \arrow[dl, "g"] \\
   & B &
   \end{tikzcd}$$

   Demonstre a seguinte lei de fusão deste combinador condicional (identifique-a no formulário):

   $$(p \to f, g) \cdot h \quad = \quad (p \cdot h) \to (f \cdot h), (g \cdot h)$$