Cálculo de Programas

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

2020/21 - Ficha nr.° 8

1. Um *bifunctor* B é um functor **binário**

$$\begin{array}{ccccc} A & \cdots & C & \cdots & \mathsf{B}\ (A, C) \\ {\scriptstyle f}\downarrow & & {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle \mathsf{B}\ (f,g)} \\ D & \cdots & E & \cdots & \mathsf{B}\ (D, E) \end{array} \quad \text{tal que:} \quad \left\{ \begin{array}{l} \mathsf{B}\ (id, id) = id \\ \mathsf{B}\ (f \cdot g, h \cdot k) = \mathsf{B}\ (f, h) \cdot \mathsf{B}\ (g, k) \end{array} \right. \tag{F1}$$

   Mostre que $\mathsf{B}\ (X, Y) = X \times Y$, $\mathsf{B}\ (X, Y) = X + Y$ e $\mathsf{B}\ (X, Y) = X + Y \times Y$ são bifunctores.

2. O diagrama genérico de um catamorfismo de gene g sobre o *tipo paramétrico* $\mathsf{T}\ X \cong \mathsf{B}\ (X, \mathsf{T}\ X)$ cuja base é o bifunctor B, bem como a sua propriedade universal, são representados a seguir:

$$\begin{array}{ccc} \mathsf{T}\ X & \xleftarrow{\mathsf{in}} & \mathsf{B}\ (X, \mathsf{T}\ X) \\ {\scriptstyle (\!|\, g \,|\!)}\downarrow & & \downarrow{\scriptstyle \mathsf{B}\ (id, (\!|\, g \,|\!)) = \mathsf{F}\ (\!|\, g \,|\!)} \\ B & \xleftarrow[g]{} & \mathsf{B}\ (X, B) \end{array} \qquad k = (\!|\, g \,|\!) \ \equiv \ k \cdot \mathsf{in} = g \cdot \underbrace{\mathsf{B}\ (id, k)}_{\mathsf{F}\ k}$$

   (Repare-se que se tem sempre $\mathsf{F}\ k = \mathsf{B}\ (id, k)$.) Partindo da definição *genérica* de map associado ao tipo T,

$$\mathsf{T}\ f \quad = \quad (\!|\, \mathsf{in} \cdot \mathsf{B}\ (f, id) \,|\!)$$

   dada no formulário, mostre que o map das sequências finitas (vulg. listas) é a função

$$\begin{array}{l} f^*\ [\,] = [\,] \\ f^*\ (h : t) = f\ h : f^*\ t \end{array}$$

3. Considere a função $depth = (\!|\, [\mathsf{one}\ , \mathsf{succ} \cdot umax] \,|\!)$ que calcula a profundidade de árvores do tipo

$$\mathsf{T}\ X = \mathsf{LTree}\ X \qquad \left\{ \begin{array}{l} \mathsf{B}\ (X, Y) = X + Y^2 \\ \mathsf{B}\ (f, g) = f + g^2 \end{array} \right. \qquad \mathsf{in} = [Leaf\ , Fork]$$

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

   onde $umax\ (a, b) = max\ a\ b$. Mostre, por absorção-cata, que a profundidade de uma árvore t não é alterada quando aplica uma função f a todas as suas folhas:

$$depth \cdot \mathsf{LTree}\ f = depth \tag{F2}$$

4. Numa página de STACK OVERFLOW[1] alguém respondeu afirmativamente à pergunta

   Pode fazer-se unzip *num só passo?*

   com a versão

$$\begin{array}{l} \mathsf{unzip}\ [\,] = ([\,],[\,]) \\ \mathsf{unzip}\ ((a,b):xs) = (a:as, b:bs)\ \mathbf{where}\ (as,bs) = \mathsf{unzip}\ xs \end{array}$$

   O que essa página não faz é explicar como é que os dois passos de

$$\mathsf{unzip}\ xs = (\mathsf{map}\ \pi_1\ xs, \mathsf{map}\ \pi_2\ xs)$$

   se fundem num só. Como exemplo de aplicação da lei de *banana-split*,

$$\langle (\!|\, i \,|\!), (\!|\, j \,|\!) \rangle = (\!|\, (i \times j) \cdot \langle \mathsf{F}\ \pi_1, \mathsf{F}\ \pi_2 \rangle \,|\!)$$

   — identifique-a no formulário — complete a derivação que se dá a seguir dessa evidência:

$$\begin{array}{cl}
& \mathsf{unzip}\ xs = (\mathsf{map}\ \pi_1\ xs, \mathsf{map}\ \pi_2\ xs) \\
\equiv & \quad \{ \quad \ldots\ldots\ldots \quad \} \\
& \mathsf{unzip} = \langle \mathsf{map}\ \pi_1, \mathsf{map}\ \pi_2 \rangle \\
\equiv & \quad \{ \quad \ldots\ldots\ldots \quad \} \\
& \mathsf{unzip} = \langle (\!|\, \mathsf{in} \cdot \mathsf{B}\ (\pi_1, id) \,|\!), (\!|\, \mathsf{in} \cdot \mathsf{B}\ (\pi_2, id) \,|\!) \rangle \\
\equiv & \quad \{ \quad \ldots\ldots\ldots \quad \} \\
& \mathsf{unzip} = (\!|\, \langle \mathsf{in} \cdot \mathsf{B}\ (\pi_1, id) \cdot \mathsf{F}\ \pi_1, \mathsf{in} \cdot \mathsf{B}\ (\pi_2, id) \cdot \mathsf{F}\ \pi_2 \rangle \,|\!) \\
\equiv & \quad \{ \quad \ldots\ldots\ldots \quad \} \\
& \mathsf{unzip} = (\!|\, \langle \mathsf{in} \cdot \mathsf{B}\ (\pi_1, \pi_1), \mathsf{in} \cdot \mathsf{B}\ (\pi_2, \pi_2) \rangle \,|\!) \\
\equiv & \quad \{ \quad \ldots\ldots\ldots \quad \} \\
& \mathsf{unzip} \cdot \mathsf{in} = \langle [\mathsf{nil}\ , \mathsf{cons} \cdot (\pi_1 \times \pi_1)], [\mathsf{nil}\ , \mathsf{cons} \cdot (\pi_2 \times \pi_2)] \rangle \cdot (id + id \times \mathsf{unzip}) \\
\equiv & \quad \{ \quad \ldots\ldots\ldots \quad \} \\
& \left\{ \begin{array}{l} \mathsf{unzip} \cdot \mathsf{nil} = \langle \mathsf{nil}, \mathsf{nil} \rangle \\ \mathsf{unzip} \cdot \mathsf{cons} = (\mathsf{cons} \times \mathsf{cons}) \cdot \langle \pi_1 \times \pi_1, \pi_2 \times \pi_2 \rangle \cdot (id \times \mathsf{unzip}) \end{array} \right. \\
\equiv & \quad \{ \text{ ... alguns passos mais ... } \} \\
& \ldots. \\
\equiv & \quad \{ \quad \ldots\ldots\ldots \quad \} \\
& \left\{ \begin{array}{l} \mathsf{unzip}\ [\,] = ([\,],[\,]) \\ \mathsf{unzip}\ ((a,b):xs) = ((a:as),(b:bs))\ \mathbf{where}\ (as,bs) = \mathsf{unzip}\ xs \end{array} \right.
\end{array}$$

5. Defina-se a função $average = ratio \cdot \langle \mathsf{sum}, \mathsf{length} \rangle$ para calcular a média de uma lista (não vazia), onde $ratio\ (n,d) = n\ /\ d$, para $d \neq 0$. Sabendo que sum e length são catamorfismos (recorde quais são os seus genes), recorra à lei "banana-split"para derivar

$$\begin{array}{l} average\ l = x\ /\ y\ \mathbf{where} \\ \quad (x,y) = aux\ l \\ \quad aux\ [\,] = (0,0) \\ \quad aux\ (a:l) = (a+x, y+1)\ \mathbf{where}\ (x,y) = aux\ l \end{array}$$

   em Haskell.

[1] Cf. `https://stackoverflow.com/questions/18287848/unzip-in-one-pass`.