Cálculo de Programas

Lic. C. Computação (2º ano)
Lic./Mest. em Engenharia Informática (3º ano)
UNIVERSIDADE DO MINHO

2021/22 - Ficha nr.° 8

1. Considere a função

$$\begin{array}{l} \mathsf{mirror}\ (Leaf\ \ a) = Leaf\ \ a \\ \mathsf{mirror}\ (Fork\ (x,y)) = Fork\ (\mathsf{mirror}\ y, \mathsf{mirror}\ x) \end{array}$$

que "espelha" árvores binárias do tipo LTree (ver fichas anteriores). Comece por mostrar que

$$\mathsf{mirror} = (\!|\, \mathsf{in} \cdot (id + \mathsf{swap}) \,|\!) \tag{F1}$$

desenhando o digrama que representa este catamorfismo.

Tal como swap, mirror é um isomorfismo de árvores pois é a sua própria inversa:

$$\mathsf{mirror} \cdot \mathsf{mirror} = id \tag{F2}$$

Complete a seguinte demonstração de (F2):

$$\begin{array}{ll} & \mathsf{mirror} \cdot \mathsf{mirror} = id \\ \equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\ & \mathsf{mirror} \cdot (\!|\, \mathsf{in} \cdot (id + \mathsf{swap}) \,|\!) = (\!|\, \mathsf{in} \,|\!) \\ \Leftarrow & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\ & \mathsf{mirror} \cdot \ldots = \ldots \\ \ldots & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\ & (etc) \end{array}$$

2. Mostre que a lei da recursividade mútua generaliza a mais do que duas funções, neste caso três:

$$\left\{ \begin{array}{l} f \cdot in = h \cdot \mathsf{F}\, \langle f, \langle g, j \rangle \rangle \\ g \cdot in = k \cdot \mathsf{F}\, \langle f, \langle g, j \rangle \rangle \\ j \cdot in = l \cdot \mathsf{F}\, \langle f, \langle g, j \rangle \rangle \end{array} \right. \quad \equiv \quad \langle f, \langle g, j \rangle \rangle = (\!|\, \langle h, \langle k, l \rangle \rangle \,|\!) \tag{F3}$$

3. Considere a função

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

escrita em Haskell. Numa página de STACK OVERFLOW[1] alguém respondeu afirmativamente à pergunta

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

Pode fazer-se unzip *num só passo?*

com a versão

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

O que essa página não faz é explicar como é que os dois passos de unzip 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, onde $\mathsf{B}\ (f,g) = id + f \times g$:

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

4. Mostre que, se F e G são functores, então também o serão $\mathsf{F} + \mathsf{G}$, $\mathsf{F} \times \mathsf{G}$ e $\mathsf{F} \cdot \mathsf{G}$ que a seguir se definem:

$$\begin{array}{l}(\mathsf{F} + \mathsf{G})\ X = (\mathsf{F}\ X) + (\mathsf{G}\ X)\\ (\mathsf{F} \times \mathsf{G})\ X = (\mathsf{F}\ X) \times (\mathsf{G}\ X)\\ (\mathsf{F} \cdot \mathsf{G})\ X = \mathsf{F}\ (\mathsf{G}\ X)\end{array}$$

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

$$\begin{array}{ccccc} A & \cdots & C & \cdots & \mathsf{B}\ (A,C)\\ {\scriptstyle f}\downarrow & & {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle \mathsf{B}\ (f,g)}\\ D & \cdots & E & \cdots & \mathsf{B}\ (D,E)\end{array} \quad \text{tal que:} \quad \left\{\begin{array}{l}\mathsf{B}\ (id,id) = id\\ \mathsf{B}\ (f \cdot g, h \cdot k) = \mathsf{B}\ (f,h) \cdot \mathsf{B}\ (g,k)\end{array}\right. \qquad \text{(F4)}$$

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.