Cálculo de Programas

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

2020/21 - Ficha nr.° 6

1. Os diagramas seguintes representam as **propriedades universais** que definem o combinador **catamorfismo** para dois tipos de dados — números naturais $\mathbb{N}_0$ à esquerda e listas finitas A^* à direita:

$$\begin{tikzcd}
\mathbb{N}_0 \arrow[d, "\llparenthesis g \rrparenthesis"'] & 1 + \mathbb{N}_0 \arrow[l, "\mathsf{in}"'] \arrow[d, "id + \llparenthesis g \rrparenthesis"] \\
B & 1 + B \arrow[l, "g"]
\end{tikzcd}
\qquad
\begin{tikzcd}
A^* \arrow[d, "\llparenthesis g \rrparenthesis"'] & 1 + A \times A^* \arrow[l, "\mathsf{in}"'] \arrow[d, "id + id \times \llparenthesis g \rrparenthesis"] \\
B & 1 + A \times B \arrow[l, "g"]
\end{tikzcd}$$

$$\begin{cases} \mathsf{in} = [\mathsf{zero}\ , \mathsf{succ}] \\ \mathsf{zero}\ _ = 0 \\ \mathsf{succ}\ n = n + 1 \end{cases}
\qquad
\begin{cases} \mathsf{in} = [\mathsf{nil}\ , \mathsf{cons}] \\ \mathsf{nil}\ _ = [\,] \\ \mathsf{cons}\ (a, x) = a : x \end{cases}$$

$$k = \llparenthesis g \rrparenthesis \;\equiv\; k \cdot \mathsf{in} = g \cdot (id + k) \qquad k = \llparenthesis g \rrparenthesis \;\equiv\; k \cdot \mathsf{in} = g \cdot (id + id \times k)$$

$$\mathsf{for}\ b\ i = \llparenthesis [\underline{i}\ , b] \rrparenthesis \qquad \mathsf{foldr}\ f\ u = \llparenthesis [\underline{u}\ , \widehat{f}] \rrparenthesis$$

onde $\widehat{f}$ abrevia $\mathsf{uncurry} f$.

   (a) Tendo em conta o diagrama da esquerda, codifique, em Haskell

   $$\llparenthesis g \rrparenthesis = g \cdot (id + \llparenthesis g \rrparenthesis) \cdot \mathsf{out}$$

   e

   $$\mathsf{for}\ b\ i = \llparenthesis [\underline{i}\ , b] \rrparenthesis$$

   em que out foi calculada numa ficha anterior. De seguida, codifique

   $$f = \pi_2 \cdot aux \ \textbf{where}\ aux = \mathsf{for}\ \langle \mathsf{succ} \cdot \pi_1, \mathsf{mul} \rangle\ (1, 1)$$

   e inspeccione o comportamento de f. Que função é essa?

   (b) Identifique como catamorfismos de listas as funções seguintes, indicando o gene g para cada caso:[1]

      i. k é a função que multiplica todos os elementos de uma lista
      ii. $k = reverse$
      iii. $k = \mathsf{concat}$
      iv. k é a função $\mathsf{map}\ f$, para um dado $f : A \to B$
      v. k é a função que calcula o máximo de uma lista de números naturais ($\mathbb{N}_0{}^*$).
      vi. $k = \mathsf{filter}\ p$ onde

$$\begin{array}{l} \mathsf{filter}\ p\ [\,] = [\,] \\ \mathsf{filter}\ p\ (h : t) = x +\!\!+\ \mathsf{filter}\ p\ t \\ \quad \textbf{where}\ x = \textbf{if}\ (p\ h)\ \textbf{then}\ [h]\ \textbf{else}\ [\,] \end{array}$$

[1] Apoie a sua resolução com diagramas.

2. Considere o seguinte inventário de quatro tipos de árvores:

   (a) Árvores com informação de tipo A nos nós:

   $$\mathsf{T} = \mathsf{BTree}\ A \qquad \begin{cases} \mathsf{F}\ X = 1 + A \times X^2 \\ \mathsf{F}\ f = id + id \times f^2 \end{cases} \qquad \mathsf{in} = [\underline{Empty}\ , Node]$$

   Haskell: **data** $\mathsf{BTree}\ a = Empty \mid Node\ (a, (\mathsf{BTree}\ a, \mathsf{BTree}\ a))$

   (b) Árvores com informação de tipo A nas folhas:

   $$\mathsf{T} = \mathsf{LTree}\ A \qquad \begin{cases} \mathsf{F}\ X = A + X^2 \\ \mathsf{F}\ f = id + f^2 \end{cases} \qquad \mathsf{in} = [Leaf\ , Fork]$$

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

   (c) Árvores com informação nos nós e nas folhas:

   $$\mathsf{T} = \mathsf{FTree}\ B\ A \qquad \begin{cases} \mathsf{F}\ X = B + A \times X^2 \\ \mathsf{F}\ f = id + id \times f^2 \end{cases} \qquad \mathsf{in} = [Unit\ , Comp]$$

   Haskell: **data** $\mathsf{FTree}\ b\ a = Unit\ b \mid Comp\ (a, (\mathsf{FTree}\ b\ a, \mathsf{FTree}\ b\ a))$

   (d) Árvores de expressão:

   $$\mathsf{T} = Expr\ V\ O \qquad \begin{cases} \mathsf{F}\ X = V + O \times X^* \\ \mathsf{F}\ f = id + id \times \mathsf{map}\ f \end{cases} \qquad \mathsf{in} = [Var\ , Op]$$

   Haskell: **data** $Expr\ v\ o = Var\ v \mid Op\ (o, [Expr\ v\ o])$

   Defina o gene g para cada um dos catamorfismos seguintes desenhando, para cada caso, o diagrama correspondente:

   - $zeros = (\!|\ g\ |\!)$ — substitui todas as folhas de uma árvore de tipo (2b) por zero.
   - $conta = (\!|\ g\ |\!)$ — conta o número de nós de uma árvore de tipo (2a).
   - $\mathsf{mirror} = (\!|\ g\ |\!)$ — espelha uma árvore de tipo (2b), i.e., roda-a de 180º.
   - $converte = (\!|\ g\ |\!)$ — converte árvores de tipo (2c) em árvores de tipo (2a) eliminando os Bs que estão na primeira.
   - $\mathsf{vars} = (\!|\ g\ |\!)$ — lista as variáveis de uma árvore expressão de tipo (2d).

3. Implemente $\mathsf{mirror} = (\!|\ g\ |\!)$ em Haskell definindo previamente `outLTree` e o combinador `cataLTree` (catamorfismo de LTrees).

4. Converta a função vars do exercício 2 numa função com variáveis em Haskell sem quaisquer combinadores *pointfree*.

5. A função seguinte, em Haskell

   $$\begin{array}{l} sumprod\ a\ [\,] = 0 \\ sumprod\ a\ (h : t) = a * h + sumprod\ a\ t \end{array}$$

   é o catamorfismo de listas

   $$sumprod\ a \quad = \quad (\!|\ [\mathsf{zero}\ , \mathsf{add} \cdot ((a*) \times id)]\ |\!) \tag{F1}$$

   onde $\mathsf{zero} = \underline{0}$ e $\mathsf{add}\ (x, y) = x + y$. Mostre, como exemplo de aplicação da propriedade de **fusão-cata** para listas, que

   $$sumprod\ a \quad = \quad (a*) \cdot \mathsf{sum} \tag{F2}$$

   onde $\mathsf{sum} = (\!|\ [\mathsf{zero}\ , \mathsf{add}]\ |\!)$. **NB:** não ignore propriedades elementares da aritmética que lhe possam ser úteis.