Cálculo de Programas

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

2021/22 - Ficha nr.° 7

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

   (a) Árvores com informação de tipo A nos nós:
   $\mathsf{T} = \mathsf{BTree}\ A \qquad \left\{ \begin{array}{l} \mathsf{F}\ X = 1 + A \times X^2 \\ \mathsf{F}\ f = id + id \times f^2 \end{array} \right. \qquad \mathsf{in} = [\underline{Empty}\ , Node]$
   Haskell: $\mathbf{data}\ \mathsf{BTree}\ a = Empty \mid Node\ (a, (\mathsf{BTree}\ a, \mathsf{BTree}\ a))$

   (b) Árvores com informação de tipo A nas folhas:
   $\mathsf{T} = \mathsf{LTree}\ A \qquad \left\{ \begin{array}{l} \mathsf{F}\ X = A + X^2 \\ \mathsf{F}\ f = id + f^2 \end{array} \right. \qquad \mathsf{in} = [Leaf\ , Fork]$
   Haskell: $\mathbf{data}\ \mathsf{LTree}\ a = Leaf\ a \mid Fork\ (\mathsf{LTree}\ a, \mathsf{LTree}\ a)$

   (c) Árvores com informação nos nós e nas folhas:
   $\mathsf{T} = \mathsf{FTree}\ B\ A \qquad \left\{ \begin{array}{l} \mathsf{F}\ X = B + A \times X^2 \\ \mathsf{F}\ f = id + id \times f^2 \end{array} \right. \qquad \mathsf{in} = [Unit\ , Comp]$
   Haskell: $\mathbf{data}\ \mathsf{FTree}\ b\ a = Unit\ b \mid Comp\ (a, (\mathsf{FTree}\ b\ a, \mathsf{FTree}\ b\ a))$

   (d) Árvores de expressão:
   $\mathsf{T} = Expr\ V\ O \qquad \left\{ \begin{array}{l} \mathsf{F}\ X = V + O \times X^* \\ \mathsf{F}\ f = id + id \times \mathsf{map}\ f \end{array} \right. \qquad \mathsf{in} = [Var\ , Op]$
   Haskell: $\mathbf{data}\ Expr\ v\ o = Var\ v \mid Op\ (o, [Expr\ v\ o])$

   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 (1b) por zero.
   - $conta = (\!| g |\!)$ — conta o número de nós de uma árvore de tipo (1a).
   - $\mathsf{mirror} = (\!| g |\!)$ — espelha uma árvore de tipo (1b), i.e., roda-a de 180º.
   - $converte = (\!| g |\!)$ — converte árvores de tipo (1c) em árvores de tipo (1a) eliminando os Bs que estão na primeira.
   - $\mathsf{vars} = (\!| g |\!)$ — lista as variáveis de uma árvore expressão de tipo (1d).

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

3. Converta a função vars do exercício 1 numa função com variáveis em Haskell sem quaisquer combinadores *pointfree*.

4. As seguintes funções mutuamente recursivas testam a paridade de um número natural:

$$\left\{ \begin{array}{l} impar\ 0 = \text{FALSE} \\ impar\ (n+1) = par\ n \end{array} \right. \qquad \left\{ \begin{array}{l} par\ 0 = \text{TRUE} \\ par\ (n+1) = impar\ n \end{array} \right.$$

Assumindo o functor $\mathsf{F}\ f = id + f$, mostre que esse par de definições é equivalente ao sistema de equações

$$\left\{ \begin{array}{l} impar \cdot \mathsf{in} = h \cdot \mathsf{F}\ \langle impar, par \rangle \\ par \cdot \mathsf{in} = k \cdot \mathsf{F}\ \langle impar, par \rangle \end{array} \right.$$

para um dado h e k (deduza-os). De seguida, recorra às leis da recursividade mútua e da troca para mostrar que

$$imparpar = \langle impar, par \rangle = \mathsf{for\ swap}\ (\textsc{False}, \textsc{True})$$

5. A seguinte função em Haskell calcula a lista dos primeiros n números naturais por ordem inversa:

$$\begin{array}{l} insg\ 0 = [\,] \\ insg\ (n+1) = (n+1) : insg\ n \end{array}$$

Mostre que $insg$ pode ser definida por recursividade mútua tal como se segue,

$$\begin{array}{l} insg\ 0 = [\,] \\ insg\ (n+1) = (fsuc\ n) : insg\ n \\ fsuc\ 0 = 1 \\ fsuc\ (n+1) = fsuc\ n + 1 \end{array}$$

e, usando a lei de recursividade mútua, derive:

$$\begin{array}{l} insg = \pi_2 \cdot insgfor \\ insgfor = \mathsf{for}\ \langle (1+) \cdot \pi_1, \mathsf{cons} \rangle\ (1, [\,]) \end{array}$$

6. Considere o par de funções mutuamente recursivas

$$\left\{ \begin{array}{l} f_1\ [\,] = [\,] \\ f_1\ (h : t) = h : (f_2\ t) \end{array} \right. \qquad \left\{ \begin{array}{l} f_2\ [\,] = [\,] \\ f_2\ (h : t) = f_1\ t \end{array} \right.$$

Use a lei de recursividade mútua para definir $\langle f_1, f_2 \rangle$ como um catamorfismo de listas (onde o functor de trabalho é $\mathsf{F}\ f = id + id \times f$) e desenhe o respectivo diagrama. Que faz cada uma destas funções f_1 e f_2?

7. A função $k = \mathsf{for}\ f\ i$ pode ser codificada em sintaxe C escrevendo

```
int k(int n) {
  int r=i;
  int j;
  for (j=1;j<n+1;j++) {r=f(r);}
  return r;
};
```

Escreva em sintaxe C as funções $(a*) = \mathsf{for}\ (a+)\ 0$ e outros catamorfismos de naturais de que se tenha falado nas aulas da disciplina.