Cálculo de Programas
Algebra of Programming

UNIVERSIDADE DO MINHO
Lic. em Engenharia Informática (3º ano)
Lic. Ciências da Computação (2º ano)

2024/25 - Ficha *(Exercise sheet)* nr. 9

1. Considere o seguinte inventário de quatro tipos de árvores: *Consider the following inventory of four types of trees:*

   (a) Árvores com informação de tipo A nas folhas *(Trees with data in their leaves)* :

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

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

   (b) Árvores com informação de tipo A nos nós *(Trees whose data of type A are stored in their nodes)*:

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

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

   (c) Árvores com informação nos nós e nas folhas *(Full trees — data in both leaves and nodes)*:

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

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

   (d) Árvores de expressão *(Expression trees)*:

   $$\mathsf{T} = \mathsf{Expr}\ V\ O \qquad \left\{\begin{array}{l}\mathsf{F}\ X = V + O \times X^*\\ \mathsf{F}\ f = id + id \times \mathsf{map}\ f\end{array}\right. \qquad \mathsf{in} = [Var\ , Term]$$

   Haskell: $\mathbf{data}\ \mathsf{Expr}\ v\ o = Var\ v \mid Term\ (o, [\mathsf{Expr}\ v\ o])$

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

   - $maximum = (\!|\, g \,|\!)$ — devolve a maior folha de uma árvore de tipo (1a).
   - $inorder = (\!|\, g \,|\!)$ — faz a travessia inorder de uma árvore de tipo (1b).
   - $mirror = (\!|\, g \,|\!)$ — espelha uma árvore de tipo (1b), i.e., roda-a de 180º.
   - $rep\ \ a = (\!|\, g \,|\!)$ — substitui todas as folhas de uma árvore de tipo (1a) por um mesmo valor $a \in A$.

   Define the "gene" g for each of the following catamorphisms by drawing, for each case, the corresponding diagram:

   - $maximum = (\!|\, g \,|\!)$ — *returns the largest leaf of a tree of type (1a).*
   - $inorder = (\!|\, g \,|\!)$ — *performs a traversal of a type tree (1b).*
   - $mirror = (\!|\, g \,|\!)$ — *mirrors a tree of type (1b), i.e., rotates it 180º.*
   - $rep\ \ a = (\!|\, g \,|\!)$ — *replaces all leaves of a tree of type (1a) by the same value $a \in A$.*

- $convert = (\!| g |\!)$ — converte árvores de tipo (1c) em árvores de tipo (1b) eliminando os Bs que estão na primeira.
- $vars = (\!| g |\!)$ — lista as variáveis de uma árvore expressão de tipo (1d).

- *$convert = (\!| g |\!)$ — converts trees of type (1c) into trees of type (1b) eliminating the Bs that can be found in the first.*
- *$vars = (\!| g |\!)$ — lists the variables of an expression tree of type (1d).*

---

2. Derive a versão *pointwise* do seguinte catamorfismo de BTrees,

   Derive the pointwise *version of the following catamorphism of* BTree*s*

$$
\begin{aligned}
&tar = (\!| [\mathsf{singl} \cdot \mathsf{nil}\ , g] |\!)\ \textbf{where}\\
&\quad g = \mathsf{map\ cons} \cdot lstr \cdot (id \times \mathsf{conc})\\
&\quad lstr\ (b, x) = [(b, a) \mid a \leftarrow x]
\end{aligned}
$$

   entregando no final uma versão da função em que não ocorrem os nomes das funções map, cons, singl, nil, conc e $lstr$. Pode usar $\mathsf{map}\ f\ x = [f\ a \mid a \leftarrow x]$ como definição *pointwise* de map em listas.

   eventually delivering a version of the function in which the function names map, cons, singl, nil, conc *do not occur. and lstr. You can use* $\mathsf{map}\ f\ x = [f\ a \mid a \leftarrow x]$ *as* pointwise *definition of* map *in lists.*

---

3. Converta o catamorfismo $vars$ do exercício 1 numa função em Haskell sem quaisquer combinadores *pointfree*.

   Unfold catamorphism vars (exercise 1) towards a function in Haskell without any pointfree combinator.

---

4. Um *anamorfismo* é um *"catamorfismo ao contrário"*, i.é uma função $k : A \to \mathsf{T}$ tal que

   An anamorphism *is a* "catamorphism upside-down"*, i.e. a function $k : A \to \mathsf{T}$ such that*

$$k = \mathsf{in} \cdot \mathsf{F}\ k \cdot g \tag{F1}$$

   escrevendo-se $k = [\!(g)\!]$. Mostre que o anamorfismo de listas

   One writes $k = [\!(g)\!]$. Show that the list-anamorphism

$$k = [\!((id + \langle f, id \rangle) \cdot \mathsf{out}_{\mathbb{N}_0})\!] \tag{F2}$$

   descrito pelo diagrama

   depicted in diagram

$$
\begin{tikzcd}
\mathbb{N}_0{}^* & & 1 + \mathbb{N}_0 \times \mathbb{N}_0{}^* \arrow[ll, "\mathsf{in}_*"'] \\
\mathbb{N}_0 \arrow[u, "k"] \arrow[r, "\mathsf{out}_{\mathbb{N}_0}"'] & 1 + \mathbb{N}_0 \arrow[r, "id + \langle f, id \rangle"'] & 1 + \mathbb{N}_0 \times \mathbb{N}_0 \arrow[u, "id + id \times k"']
\end{tikzcd}
$$

   é a função

   is the function

$$
\begin{aligned}
&k\ 0 = [\,]\\
&k\ (n + 1) = (2\ n + 1) : k\ n
\end{aligned}
$$

   para $f\ n = 2\ n + 1$. (Que faz esta função?)

   for $f\ n = 2\ n + 1$. (What does this function do?)

---

5. Mostre que o anamorfismo que calcula os sufixos de uma lista

   Show that the anamorphism that computes the suffixes of a list

$$suffixes = [\![g]\!] \textbf{ where } g = (id + \langle \mathsf{cons}, \pi_2 \rangle) \cdot \mathsf{out}$$

é a função:

is the function:

$$suffixes\ [\,] = [\,]$$
$$suffixes\ (h : t) = (h : t) : suffixes\ t$$

---

6. Mostre que o catamorfismo de listas $\mathsf{length} = (\![\,[\mathsf{zero}\ , \mathsf{succ} \cdot \pi_2]\,]\!)$ é a mesma função que o *anamorfismo* de naturais $[\!((id + \pi_2) \cdot \mathsf{out}_{\mathsf{List}})\!]$.

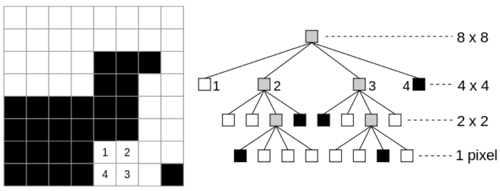
Show that the list catamorphism $\mathsf{length} = (\![\,[\mathsf{zero}\ , \mathsf{succ} \cdot \pi_2]\,]\!)$ *is the same function as the* $\mathbb{N}_0$-anamorphism $[\!((id + \pi_2) \cdot \mathsf{out}_{\mathsf{List}})\!]$.

---

7. **Questão prática** — Este problema não irá ser abordado em sala de aula. Os alunos devem tentar resolvê-lo em casa e, querendo, publicarem a sua solução no canal **#geral** do Slack, com vista à sua discussão com colegas.
Dão-se a seguir os requisitos do problema.

***Open assignment** — This assignment will not be addressed in class. Students should try to solve it at home and, whishing so, publish their solutions in the **#geral** Slack channel, so as to trigger discussion among other colleagues.*
The requirements of the problem are given below.

***Problem requirements**: The figure below*



(Source: Wikipedia) shows how an image (in this case in black and white) is represented in the form of a quaternary tree (vulg. quadtree*) by successive divisions of the 2D space into four regions, until reaching the resolution of one pixel.*
Let the following Haskell definition of a quadtree *be given, for a given type $Pixel$ predefined:*

$$\textbf{data}\ \mathsf{QTree} = Pixel \mid Blocks\ (\mathsf{QTree})\ (\mathsf{QTree})\ (\mathsf{QTree})\ (\mathsf{QTree})$$

Having chosen for this type the base functor

$$\mathsf{F}\ Y = Pixel + Y^2 \times Y^2 \qquad \text{(F3)}$$

where Y^2 abbreviates $Y \times Y$, as usual, define the usual construction and decomposition functions of this type, cf.:

$$\begin{tikzcd} \mathsf{QTree} \arrow[r, bend left, "\mathsf{out}_{\mathsf{QTree}}"] \arrow[r, phantom, "\cong"] & \mathsf{F}\ (\mathsf{QTree}) \arrow[l, bend left, "\mathsf{in}_{\mathsf{QTree}}"] \end{tikzcd}$$

Then, write the Haskell code of Quad.hs, a Haskell library similar to others already available, e.g. LTree.hs. Finally, implement as a QTree *catamorphism the operation that rotates an image 90º clockwise.*
□