CP - Ficha 9

Exercício 1

Considere o seguinte inventário de quatro tipos de árvores:

a) Árvores com informação de tipo A nas folhas:

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

Haskell: `data LTree a = Leaf a | Fork (LTree a, LTree a)`

b) Árvores com informação de tipo A nos nós:

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

Haskell: `data BTree a = Empty | Node (a, (BTree a, BTree a))`

c) Árvores com informação nos nós e nas folhas:

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

Haskell: `data FTree b a = Unit b | Comp (a, (FTree b a, FTree b a))`

d) Árvores de expressão:

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

Haskell: `data Expr v o = Var v | Term (o, [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 a).
- $inorder = (\!|g|\!)$ — faz a travessia inorder de uma árvore de tipo b).
- $mirror = (\!|g|\!)$ — espelha uma árvore de tipo b), i.e., roda-a de 180°.
- $rep\ a = (\!|g|\!)$ — substitui todas as folhas de uma árvore de tipo a) por um mesmo valor $a \in A$.
- $convert = (\!|g|\!)$ — converte árvores de tipo c) em árvores de tipo b) eliminando os Bs que estão na primeira.
- $vars = (\!|g|\!)$ — lista as variáveis de uma árvore expressão de tipo d).

Resolução 1

$$maximum$$

$$\begin{tikzcd}
\mathrm{LTree}\ A \arrow[r, "\mathrm{out}"] \arrow[d, "maximum"'] & A + (\mathrm{LTree}\ A)^2 \arrow[d, "id+maximum^2"] \\
A & A + A^2 \arrow[l, "{[id,\widehat{max}]}"']
\end{tikzcd}$$

$$inorder$$

$$\begin{tikzcd}
\mathrm{BTree}\ A \arrow[r, "\mathrm{out}"] \arrow[d, "inorder"'] & 1 + A \times (\mathrm{BTree}\ A)^2 \arrow[d, "id+id\times inorder^2"] \\
A^* & 1 + A \times (A^* \times A^*) \arrow[l, "{[\mathrm{nil},g_2]}"']
\end{tikzcd}$$

$$\text{where} \quad \mathrm{g}_2\ (a,(l,r)) = l \mathbin{+\!\!+} [a] \mathbin{+\!\!+} r$$

$$mirror$$

$$\begin{tikzcd}
\mathrm{BTree}\ A \arrow[r, "\mathrm{out}"] \arrow[d, "mirror"'] & 1 + A \times (\mathrm{BTree}\ A)^2 \arrow[d, "id+id\times mirror^2"] \\
A^* & 1 + A \times (A^* \times A^*) \arrow[l, "{[g_1,g_2]}"']
\end{tikzcd}$$

$$\text{where} \quad \begin{cases} g_1 = \underline{\mathrm{Empty}} \\ g_2 = \mathrm{Node} \cdot (id \times \mathrm{swap}) \end{cases}$$

$$rep\ a$$

$$\begin{tikzcd}
\mathrm{LTree}\ A \arrow[r, "\mathrm{out}"] \arrow[d, "rep\ a"'] & A + (\mathrm{LTree}\ A)^2 \arrow[d, "id+(rep\ a)^2"] \\
\mathrm{LTree}\ A & A + (\mathrm{LTree}\ A)^2 \arrow[l, "{[\mathrm{Leaf}\cdot\underline{a},\mathrm{Fork}]}"']
\end{tikzcd}$$

$$convert$$

$$
\begin{tikzcd}
\mathrm{FTree}\ B\ A \arrow[r, "\mathrm{out}"] \arrow[d, "convert"'] & B + A \times (\mathrm{FTree}\ B\ A)^2 \arrow[d, "id+id\times convert^2"] \\
\mathrm{BTree}\ A & B + A \times (\mathrm{BTree}\ A)^2 \arrow[l, "{[\underline{\mathrm{Empty}},\mathrm{Node}]}"']
\end{tikzcd}
$$

$$vars$$

$$
\begin{tikzcd}
\mathrm{Expr}\ V\ O \arrow[r, "\mathrm{out}"] \arrow[d, "vars"'] & V + O \times (\mathrm{Expr}\ V\ O)^* \arrow[d, "id+id\times \mathrm{map}\ vars"] \\
V^* & V + O \times (V^*)^* \arrow[l, "{[\mathrm{singl},\mathrm{concat}\cdot\pi_2]}"']
\end{tikzcd}
$$

Exercício 2

> Derive a versão *pointwise* do seguinte catamorfismo de BTrees,
>
> $$\begin{aligned} \mathrm{tar} &= (\![[\mathrm{singl} \cdot \mathrm{nil}, g]]\!) \ \textbf{where} \\ & \quad g = \mathrm{map\ cons} \cdot \mathit{lstr} \cdot (id \times \mathrm{conc}) \\ & \quad \mathrm{lstr}\ (b, x) = [(b, a) \mid a \leftarrow x] \end{aligned}$$
>
> entregando no final uma versão da função em que não ocorrem os nomes das funções map, cons, singl, nil, conc e lstr. Pode usar $\mathrm{map}\ f\ x = [f\ a \mid a \leftarrow x]$ como definição *pointwise* de map em listas.

Resolução 2

$$\begin{aligned}
& \mathrm{tar} = (\![[\mathrm{singl} \cdot \mathrm{nil}, g]]\!) \\
\equiv & \qquad \{ \text{46: Universal-cata, Def. Functor BTree} \} \\
& \mathrm{tar} \cdot \mathrm{in}_{\mathrm{BTree}} = [\mathrm{singl} \cdot \mathrm{nil}, g] \cdot (id + id \times \mathrm{tar}^2) \\
\equiv & \qquad \{ \text{Def } \mathrm{in}_{\mathrm{BTree}}\text{, 20: Fusão-+, 22: Absorção-+, 1: Natural-id} \} \\
& [\mathrm{tar} \cdot \mathrm{Empty}, \mathrm{tar} \cdot \mathrm{Node}] = [\mathrm{singl} \cdot \mathrm{nil}, g \cdot (id \times \mathrm{tar}^2)] \\
\equiv & \qquad \{ \text{27: Eq-+, Def. g} \} \\
& \begin{cases} \mathrm{tar} \cdot \underline{\mathrm{Empty}} = \mathrm{singl} \cdot \mathrm{nil} \\ \mathrm{tar} \cdot \mathrm{Node} = \mathrm{map\ cons} \cdot \mathrm{lstr} \cdot (id \times \mathrm{conc}) \cdot (id \times \mathrm{tar}^2) \end{cases} \\
\equiv & \qquad \{ \text{72, 73, 75, Def. nil, Def. singl} \} \\
& \begin{cases} \mathrm{tar\ Empty} = [[\,]] \\ \mathrm{tar\ Node}\ (b, (l, r)) = \mathrm{map\ cons} \cdot \mathrm{lstr} \cdot (id \times \mathrm{conc}) \cdot (id \times \mathrm{tar}^2)\ (b, (l, r)) \end{cases} \\
\equiv & \qquad \{ \text{14: Functor-}\times\text{, 1: Natural-id} \} \\
& \begin{cases} \mathrm{tar\ Empty} = [[\,]] \\ \mathrm{tar\ Node}\ (b, (l, r)) = \mathrm{map\ cons} \cdot \mathrm{lstr} \cdot (id \times \mathrm{conc} \cdot (\mathrm{tar} \times \mathrm{tar}))\ (b, (l, r)) \end{cases} \\
\equiv & \qquad \{ \text{78: Def-}\times\ (2\times)\text{, 74: Def-id, 73: Def-comp} \} \\
& \begin{cases} \mathrm{tar\ Empty} = [[\,]] \\ \mathrm{tar\ Node}\ (b, (l, r)) = \mathrm{map\ cons} \cdot \mathrm{lstr}\ (b, \mathrm{conc} \cdot (\mathrm{tar}\ l, \mathrm{tar}\ r)) \end{cases} \\
\equiv & \qquad \{ \text{73: Def-comp, Def. lstr} \} \\
& \begin{cases} \mathrm{tar\ Empty} = [[\,]] \\ \mathrm{tar\ Node}\ (b, (l, r)) = \mathrm{map\ cons}\ [(b, a) \mid a \leftarrow \mathrm{conc} \cdot (\mathrm{tar}\ l, \mathrm{tar}\ r)] \end{cases} \\
\equiv & \qquad \{ \mathrm{map}\ f \cdot g = \mathrm{map}\ (f \cdot g)\text{, Def. cons} \} \\
& \begin{cases} \mathrm{tar\ Empty} = [[\,]] \\ \mathrm{tar\ Node}\ (b, (l, r)) = [\widehat{(:)}\ (b, a) \mid a \leftarrow \mathrm{conc} \cdot (\mathrm{tar}\ l, \mathrm{tar}\ r)] \end{cases}
\end{aligned}$$

Exercício 3

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

Resolução 3

$$
\begin{array}{l}
\mathrm{vars} = (\!|[\mathrm{singl}, \mathrm{concat} \cdot \pi_2]|\!) \\
\equiv \qquad \{ \text{46: Universal-cata, Def. Functor Expr} \} \\
\mathrm{vars} \cdot \mathrm{in}_{\mathrm{Expr}} = [\mathrm{singl}, \mathrm{concat} \cdot \pi_2] \cdot (id + id \times \mathrm{map\ vars}) \\
\equiv \qquad \{ \text{Def. } \mathrm{in}_{\mathrm{Expr}}\text{, 20: Fusão-+, 22: Absorção-+, 1: Natural-id} \} \\
[\mathrm{vars} \cdot \mathrm{Var}, \mathrm{vars} \cdot \mathrm{Term}] = [\mathrm{singl}, \mathrm{concat} \cdot \pi_2 \cdot (id \times \mathrm{map\ vars})] \\
\equiv \qquad \{ \text{13: Natural-}\pi_2\text{, 27: Eq-+} \} \\
\begin{cases} \mathrm{vars} \cdot \mathrm{Var} = \mathrm{singl} \\ \mathrm{vars} \cdot \mathrm{Term} = \mathrm{concat} \cdot \mathrm{map\ vars} \cdot \pi_2 \end{cases} \\
\equiv \qquad \{ \text{72: Ig. Ext., 73: Def-comp, Def. singl} \} \\
\begin{cases} \mathrm{vars}\ (\mathrm{Var}\ v) = [v] \\ \mathrm{vars}\ (\mathrm{Term}\ (o, exprs)) = \mathrm{concat} \cdot \mathrm{map\ vars} \cdot \pi_2\ (o, exprs) \end{cases} \\
\equiv \qquad \{ \text{79: Def-proj} \} \\
\begin{cases} \mathrm{vars}\ (\mathrm{Var}\ v) = [v] \\ \mathrm{vars}\ (\mathrm{Term}\ (o, exprs)) = \mathrm{concat} \cdot \mathrm{map\ vars}\ exprs \end{cases} \\
\equiv \qquad \{ \text{73: Def-comp, Def. map} \} \\
\begin{cases} \mathrm{vars}\ (\mathrm{Var}\ v) = [v] \\ \mathrm{vars}\ (\mathrm{Term}\ (o, exprs)) = \mathrm{concat}\ [\mathrm{vars}\ e \mid e \leftarrow exprs] \end{cases} \\
\equiv \qquad \{ \text{Def. concat} \} \\
\begin{cases} \mathrm{vars}\ (\mathrm{Var}\ v) = [v] \\ \mathrm{vars}\ (\mathrm{Term}\ (o, exprs)) = \mathrm{foldr}\ (++)\ [\,]\ [\mathrm{vars}\ e \mid e \leftarrow exprs] \end{cases} \\
\equiv \qquad \{ f\ (o, exprs) = \mathrm{foldr}\ (++)\ [\,]\ [\mathrm{vars}\ e \mid e \leftarrow exprs] \} \\
\begin{cases} \mathrm{vars}\ (\mathrm{Var}\ v) = [v] \\ \mathrm{vars}\ (\mathrm{Term}\ (o, exprs)) = f\ (o, exprs) \\ \quad \textbf{where} \\ \qquad f\ (\_, [\,]) = [\,] \\ \qquad f\ (o, \mathrm{e{:}es}) = \mathrm{vars\ e} \mathbin{++} f\ (o, \mathrm{es}) \end{cases}
\end{array}
$$

Exercício 4

> Um *anamorfismo* é um *"catamorfismo ao contrário"*, i.e., uma função $k : A \to \mathrm{T}$ tal que
>
> $$k = \mathrm{in} \cdot \mathrm{F}\ k \cdot g \tag{F1}$$
>
> escrevendo-se $k = [\!(g)\!]$. Mostre que o anamorfismo de listas
>
> $$k = [\!((id + \langle f, id \rangle) \cdot \mathrm{out}_{\mathbb{N}_0})\!] \tag{F2}$$
>
> descrito pelo diagrama
>
> $$\begin{tikzcd} {\mathbb{N}_0}^* & & & 1 + \mathbb{N}_0 \times {\mathbb{N}_0}^* \arrow[lll, "\mathsf{in}_*"'] \\ \mathbb{N}_0 \arrow[u, "k"] \arrow[r, "\mathsf{out}_{\mathbb{N}_0}"'] & 1 + \mathbb{N}_0 \arrow[rr, "id + \langle f, id \rangle"'] & & 1 + \mathbb{N}_0 \times \mathbb{N}_0 \arrow[u, "id + id \times k"'] \end{tikzcd}$$
>
> é a função
>
> $$\begin{cases} k\ 0 = [\ ] \\ k\ (n+1) = (2\ n + 1) : k\ n \end{cases}$$
>
> para $f\ n = 2\ n + 1$. (Que faz esta função?)

Resolução 4

$$
\begin{aligned}
& k = [\![(id + \langle f, id \rangle) \cdot \mathrm{out}_{\mathbb{N}_0}]\!] \\
\equiv & \qquad \{ \text{55: Universal-ana, Def. Functor } \mathbb{N}_0 \} \\
& \mathrm{out}_{List} \cdot k = (id + id \times k) \cdot (id + \langle f, id \rangle) \cdot \mathrm{out}_{\mathbb{N}_0} \\
\equiv & \qquad \{ \text{34: Shunt-right, Iso. } \mathrm{out}_{List}\text{, 33: Shunt-left, 25: Iso. } \mathrm{out}_{\mathbb{N}_0} \} \\
& k \cdot \mathrm{in}_{\mathbb{N}_0} = \mathrm{in}_{List} \cdot (id + id \times k) \cdot (id + \langle f, id \rangle) \\
\equiv & \qquad \{ \text{Def. } \mathrm{in}_{\mathbb{N}_0}\text{, Def, } \mathrm{in}_{List} \} \\
& k \cdot [\underline{0}, \mathrm{succ}] = [\mathrm{nil}, \mathrm{cons}] \cdot (id + id \times k) \cdot (id + \langle f, id \rangle) \\
\equiv & \qquad \{ \text{20: Fusão-+, 22: Absorção-+ } (2\times)\text{, 1: Natural-id } (2\times) \} \\
& [k \cdot \underline{0}, k \cdot \mathrm{succ}] = [\mathrm{nil}, \mathrm{cons} \cdot (id \times k) \cdot \langle f, id \rangle] \\
\equiv & \qquad \{ \text{27: Eq-+, 72: Ig. Ext., 73: Def-comp, 75: Def-const, Def. nil, Def. succ} \} \\
& \begin{cases} k\ 0 = [\,] \\ k\ (\mathrm{succ}\ n) = \mathrm{cons} \cdot (id \times k) \cdot \langle f, id \rangle\ n \end{cases} \\
\equiv & \qquad \{ \text{Def. succ, 73 } (2\times)\text{, 77: Def-split, 74 } (2\times)\text{, 78: Def-}\times\text{, Def. cons} \} \\
& \begin{cases} k\ 0 = [\,] \\ k\ (n+1) = f\ n : k\ n \end{cases} \\
\equiv & \qquad \{ \text{Def. f} \} \\
& \begin{cases} k\ 0 = [\,] \\ k\ (n+1) = (2\ n + 1) : k\ n \end{cases}
\end{aligned}
$$

A função $k\ n$ devolve uma lista com os primeiros n números ímpares.

Exercício 5

> Mostre que o anamorfismo que calcula os sufixos de uma lista
>
> $$suffixes = [\!(g)\!] \ \mathbf{where} \ g = (id + \langle \text{cons}, \pi_2 \rangle) \cdot \text{out}$$
>
> é a função:
>
> $$\begin{cases} suffixes \ [\,] = [\,] \\ suffixes \ (h : t) = (h : t) : suffixes \ t \end{cases}$$

Resolução 5

$$\begin{tikzcd}
(A^*)^* & & 1 + A^* \times (A^*)^* \arrow[ll, "\text{in}_\text{L}"'] \\
A^* \arrow[u, "suffixes"] \arrow[r, "\text{out}_\text{L}"'] & 1 + A \times A^* \arrow[r, "id + \langle \text{cons}, \pi_2 \rangle"'] & 1 + A^* \times A^* \arrow[u, "id + id \times suffixes"']
\end{tikzcd}$$

$$
\begin{aligned}
& suffixes = [\!((id + \langle \text{cons}, \pi_2 \rangle) \cdot \text{out})\!] \\
\equiv \quad & \{ \text{55: Universal-ana, Def. Functor List} \} \\
& \text{out} \cdot suffixes = (id + id \times suffixes) \cdot (id + \langle \text{cons}, \pi_2 \rangle) \cdot \text{out} \\
\equiv \quad & \{ \text{34: Shunt-right, 33: Shunt-left, 25: Iso. out, Def. in} \} \\
& suffixes \cdot [\text{nil}, \text{cons}] = [\text{nil}, \text{cons}] \cdot (id + id \times suffixes) \cdot (id + \langle \text{cons}, \pi_2 \rangle) \\
\equiv \quad & \{ \text{20: Fusão-+, 22: Absorção-+ } (2\times)\text{, 1: Natural-id } (2\times) \} \\
& [suffixes \cdot \text{nil}, suffixes \cdot \text{cons}] = [\text{nil}, \text{cons} \cdot (id \times suffixes) \cdot \langle \text{cons}, \pi_2 \rangle] \\
\equiv \quad & \{ \text{27: Eq-+, 72: Ig. Ext., 73, 75: Def-const, Def. nil, Def. cons} \} \\
& \begin{cases} suffixes \ [\,] = [\,] \\ suffixes \ (h : t) = \text{cons} \cdot (id \times suffixes) \cdot \langle \text{cons}, \pi_2 \rangle \ (h, t) \end{cases} \\
\equiv \quad & \{ \text{73, 77: Def-split, 79: Def-proj, Def. cons} \} \\
& \begin{cases} suffixes \ [\,] = [\,] \\ suffixes \ (h : t) = \text{cons} \cdot (id \times suffixes) \ (h : t, t) \end{cases} \\
\equiv \quad & \{ \text{78: Def-}\times\text{, 73: Def-comp, 74: Def-id, Def. cons} \} \\
& \begin{cases} suffixes \ [\,] = [\,] \\ suffixes \ (h : t) = (h : t) : suffixes \ t \end{cases} \qquad \text{c.q.m.}
\end{aligned}
$$

Exercício 6

> Mostre que o catamorfismo de listas $\text{length} = \llparenthesis [\text{zero}, \text{succ} \cdot \pi_2] \rrparenthesis$ é a mesma função que o anamorfismo de naturais $\llbracket (id + \pi_2) \cdot \text{out}_{\text{List}} \rrbracket$.

Resolução 6

$$
\begin{aligned}
& \text{length} = \llparenthesis [\underline{0}, \text{succ} \cdot \pi_2] \rrparenthesis \\
\equiv \quad & \{ \text{46: Universal-cata} \} \\
& \text{length} \cdot \text{in}_{\text{List}} = [\underline{0}, \text{succ} \cdot \pi_2] \cdot (id + id \times \text{length}) \\
\equiv \quad & \{ \text{33: Shunt-left, 22: Absorção-+, 1: Natural-id} \} \\
& \text{length} = [\underline{0}, \text{succ}] \cdot (id + \pi_2) \cdot (id + id \times \text{length}) \cdot \text{out}_{\text{List}} \\
\equiv \quad & \{ \text{Def. in}_{\mathbb{N}_0}\text{, 34: Shunt-right, 25: Functor-+, 1: Natural-id} \} \\
& \text{out}_{\mathbb{N}_0} \cdot \text{length} = (id + \pi_2 \cdot (id \times \text{length})) \cdot \text{out}_{\text{List}} \\
\equiv \quad & \{ \text{13: Natural-}\pi_2 \} \\
& \text{out}_{\mathbb{N}_0} \cdot \text{length} = (id + \text{length} \cdot \pi_2) \cdot \text{out}_{\text{List}} \\
\equiv \quad & \{ \text{1: Natural-id, 25: Functor-+} \} \\
& \text{out}_{\mathbb{N}_0} \cdot \text{length} = (id + \text{length}) \cdot (id + \pi_2) \cdot \text{out}_{\text{List}} \\
\equiv \quad & \{ \text{F}_{\mathbb{N}_0}\ \text{length} = id + \text{length}\text{, 55: Universal-ana} \} \\
& \text{length} = \llbracket (id + \pi_2) \cdot \text{out}_{\text{List}} \rrbracket
\end{aligned}
$$

$$
\begin{tikzcd}
A^* \arrow[r, "\text{out}_{\text{List}}"] \arrow[d, "\text{length}"'] & 1 + A \times A^* \arrow[d, "id + id \times \text{length}"] \\
\mathbb{N}_0 & 1 + A \times \mathbb{N}_0 \arrow[l, "{[\underline{0}, \text{succ} \cdot \pi_2]}"]
\end{tikzcd}
$$

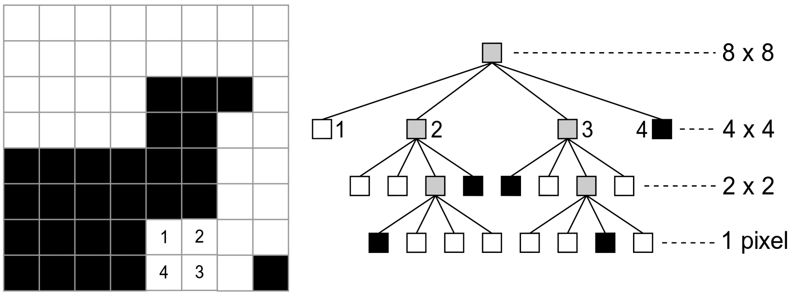
$$
\begin{tikzcd}
A^* \arrow[r, "\text{out}_{\text{List}}"] \arrow[d, "\text{length}"'] & 1 + A \times A^* \arrow[r, "id + \pi_2"] & 1 + A^* \arrow[d, "id + \text{length}"] \\
\mathbb{N}_0 & & 1 + \mathbb{N}_0 \arrow[ll, "{\text{in}_{\text{List}} = [\underline{0}, \text{succ}]}"]
\end{tikzcd}
$$

Exercício 7

Questão prática

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:

```haskell
data QTree = Pixel | Blocks (QTree) (QTree) (QTree) (QTree)
```

Having chosen for this type the base functor

$$\mathsf{F}\ Y = Pixel + Y^2 \times Y^2 \tag{F3}$$

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

QTree diagram

Then, write the Haskell code of $\mathrm{Quad.hs}$, a Haskell library similar to others already available, e.g., $\mathrm{LTree.hs}$. Finally, implement as a QTree catamorphism the operation that rotates an image 90° clockwise.

Resolução 7

TODO

☐