

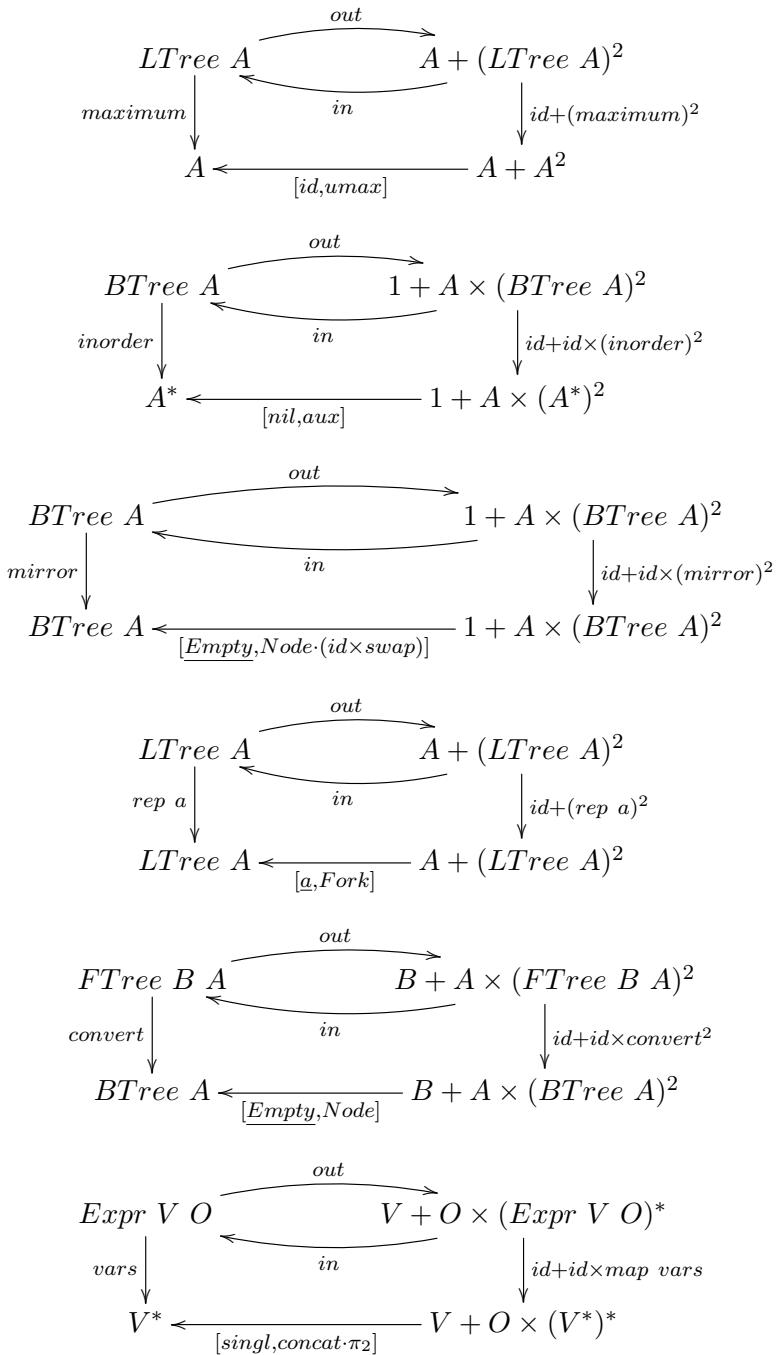
Cálculo de Programas

Resolução - Ficha 09

Eduardo Freitas Fernandes

2025

Exercício 1



Exercício 2

$$\begin{aligned}
tar &= (\lfloor [singl \cdot nil, g] \rfloor) \\
&\equiv \{\text{Universal-cata}\} \\
tar \cdot [\underline{\text{Empty}}, \underline{\text{Node}}] &= [singl \cdot nil, g] \cdot (id + id \times (tar \times tar)) \\
&\equiv \{\text{Fusão-+}, \text{Absorção-+}\} \\
[tar \cdot \underline{\text{Empty}}, tar \cdot \underline{\text{Node}}] &= [singl \cdot nil, g \cdot (id \times (tar \times tar))] \\
&\equiv \{\text{Eq-+}\} \\
&\quad \begin{cases} tar \cdot \underline{\text{Empty}} = singl \cdot nil \\ tar \cdot \underline{\text{Node}} = g \cdot (id \times (tar \times tar)) \end{cases} \\
&\equiv \{\text{pointwise, Def. comp}\} \\
&\quad \begin{cases} tar \text{ } \underline{\text{Empty}} = [[\]]] \\ tar (\text{Node}(x, (l, r))) = g (x, (tar l, tar r)) \end{cases} \\
&\equiv \{\text{Def. g}\} \\
&\quad \begin{cases} tar \text{ } \underline{\text{Empty}} = [[\]]] \\ tar (\text{Node}(x, (l, r))) = (\text{map cons} \cdot \text{lstr})(x, tar l ++ tar r) \end{cases} \\
&\equiv \{\text{Def. comp, Def. lstr}\} \\
&\quad \begin{cases} tar \text{ } \underline{\text{Empty}} = [[\]]] \\ tar (\text{Node}(x, (l, r))) = \text{map cons}[(x, a)|a \leftarrow tar l ++ tar r] \end{cases} \\
&\equiv \{\text{Def. comp, Def. lstr}\} \\
&\quad \begin{cases} tar \text{ } \underline{\text{Empty}} = [[\]]] \\ tar (\text{Node}(x, (l, r))) = [\text{cons } (h, t)|(h, t) \leftarrow [(x, a)|a \leftarrow tar l ++ tar r]] \end{cases}
\end{aligned}$$

Exercício 3

$$\begin{aligned}
vars &= (\lfloor [singl, concat \cdot \pi_2] \rfloor) \\
&\equiv \{\text{Universal-cata}\} \\
vars \cdot [Var, Term] &= [singl, concat \cdot \pi_2] \cdot (id + id \times map vars) \\
&\equiv \{\text{Fusão-+}, \text{Absorção-+}, \text{Eq-+}\} \\
&\quad \begin{cases} vars \cdot Var = singl \\ vars \cdot Term = concat \cdot \pi_2 \cdot (id \times map vars) \end{cases} \\
&\equiv \{\text{Natural-}\pi_2, \text{pointwise}\} \\
&\quad \begin{cases} vars (Varv) = [v] \\ vars (Term(o, l)) = concat (map vars l) \end{cases}
\end{aligned}$$

Exercício 4

$$\begin{aligned}
k &= [(id + \langle f, id \rangle) \cdot out_{N_0}] \\
&\equiv \{\text{universal-ana}\} \\
out_* \cdot k &= (id + id \times k) \cdot (id + \langle f, id \rangle) \cdot out_{N_0} \\
&\equiv \{\text{Shunt-left, Shunt-right}\} \\
k \cdot in_{N_0} &= in_* \cdot (id + id \times k) \cdot (id + \langle f, id \rangle) \\
&\equiv \{\text{Functor-+}\} \\
k \cdot [zero, succ] &= [nil, cons] \cdot (id + (id \times k) \cdot \langle f, id \rangle) \\
&\equiv \{\text{Absorção-}\times, \text{Absorção-+}\} \\
k \cdot [zero, succ] &= [nil, cons \cdot \langle f, k \rangle] \\
&\equiv \{\text{Fusão-+}, \text{Eq-+}, \text{pointwise}\} \\
&\quad \left\{ \begin{array}{l} k \ 0 = [] \\ k \ (n+1) = f \ n : k \ n \end{array} \right.
\end{aligned}$$

Exercício 5

$$\begin{aligned}
suffixes &= [(id + \langle cons, \pi_2 \rangle) \cdot out] \\
&\equiv \{\text{Universal-ana}\} \\
out \cdot suffixes &= (id + id \times suffixes) \cdot (id + \langle cons, \pi_2 \rangle) \cdot out \\
&\equiv \{\text{Shunt-left, Shunt-right}\} \\
suffixes \cdot in &= in \cdot (id + id \times suffixes) \cdot (id + \langle cons, \pi_2 \rangle) \\
&\equiv \{\text{Functor-+}, \text{Fusão-+}\} \\
[suffixes \cdot nil, suffixes \cdot cons] &= [nil, cons] \cdot (id + ((id \times suffixes) \cdot \langle cons, \pi_2 \rangle)) \\
&\equiv \{\text{Absorção-+}\} \\
[suffixes \cdot nil, suffixes \cdot cons] &= [nil, cons \cdot ((id \times suffixes) \cdot \langle cons, \pi_2 \rangle)] \\
&\equiv \{\text{Absorção-}\times\} \\
[suffixes \cdot nil, suffixes \cdot cons] &= [nil, cons \cdot \langle cons, suffixes \cdot \pi_2 \rangle] \\
&\equiv \{\text{Eq-+}, \text{pointwise}\} \\
&\quad \left\{ \begin{array}{l} suffixes \ [] = [] \\ suffixes \ (h : t) = (h : t) : suffixes \ t \end{array} \right.
\end{aligned}$$

Exercício 6

$$\begin{aligned} \langle [zero, succ \cdot \pi_2] \rangle &= \langle (id + \pi_2) \cdot out_* \rangle \\ &\equiv \{\text{Universal-ana}\} \\ \text{out}_{\mathbb{N}_0} \cdot \langle [zero, succ \cdot \pi_2] \rangle &= \mathbf{F}_{\mathbb{N}_0} \langle [zero, succ \cdot \pi_2] \rangle \cdot (id + \pi_2) \cdot out_* \\ &\equiv \{\text{Shunt-left, Shunt-right, Functor dos naturais}\} \\ \langle [zero, succ \cdot \pi_2] \rangle \cdot in_* &= in_{\mathbb{N}_0} \cdot (id + \langle [zero, succ \cdot \pi_2] \rangle) \cdot (id + \pi_2) \\ &\equiv \{\text{Functor-+}, \text{Def. in, Cancelamento-cata}\} \\ [zero, succ \cdot \pi_2] \cdot (id + id \times \langle [zero, succ \cdot \pi_2] \rangle) &= [zero, succ] \cdot (id + \langle [zero, succ \cdot \pi_2] \rangle \cdot \pi_2) \\ &\equiv \{\text{Absorção-+}\} \\ [zero, succ \cdot \pi_2] \cdot (id \times \langle [zero, succ \cdot \pi_2] \rangle) &= [zero, succ \cdot \langle [zero, succ \cdot \pi_2] \rangle \cdot \pi_2] \\ &\equiv \{\text{Eq-+}, \text{Natural-}\pi_2\} \\ \begin{cases} zero = zero \\ succ \cdot \langle [zero, succ \cdot \pi_2] \rangle \cdot \pi_2 = succ \cdot \langle [zero, succ \cdot \pi_2] \rangle \cdot \pi_2 \end{cases} \end{aligned}$$

Exercício 7

```
Data QTree a = Pixel a | Blocks ((QTree a, QTree a), (QTree a, QTree a))
deriving (Show)

inQTree :: Either a ((QTree a, QTree a), (QTree a, QTree a)) -> QTree a
inQTree = either Pixel Blocks

outQTree :: QTree a -> Either a ((QTree a, QTree a), (QTree a, QTree a))
outQTree (Pixel x) = i1 x
outQTree (Blocks ((x, y), (z, w))) = i2 ((x, y), (z, w))

-- Bi-Functor de QTree
baseQTree g f = g -|- ((f >< f) >< (f >< f))

-- Functor de QTree
recQTree f = baseQTree id f

-- catamorfismo de QTree
cataQTree g = g . (recQTree (cataQTree g)) . outQTree

-- anamorfismo de QTree
anaQTree g = inQTree . (recQTree (anaQTree g)) . g

-- hylomorfismo de QTree
hyloQTree f g = cataQTree f . anaQTree g

instance Functor QTree where
    fmap f = cataQTree (inQTree . baseQTree f id)

    mirrorQTree = cataQTree (either Pixel (Blocks . swap . (swap >< swap)))

    countQTree = cataQTree (either one (add . (add >< add)))

    depthQTree = cataQTree (either one (succ . umax . (umax >< umax)))

    rotate90 = cataQTree (either Pixel (Blocks . f))
        where f ((x, y), (z, w)) = ((w, x), (y, z))

    tips = cataQTree (either singl (conc . (conc >< conc)))
```