

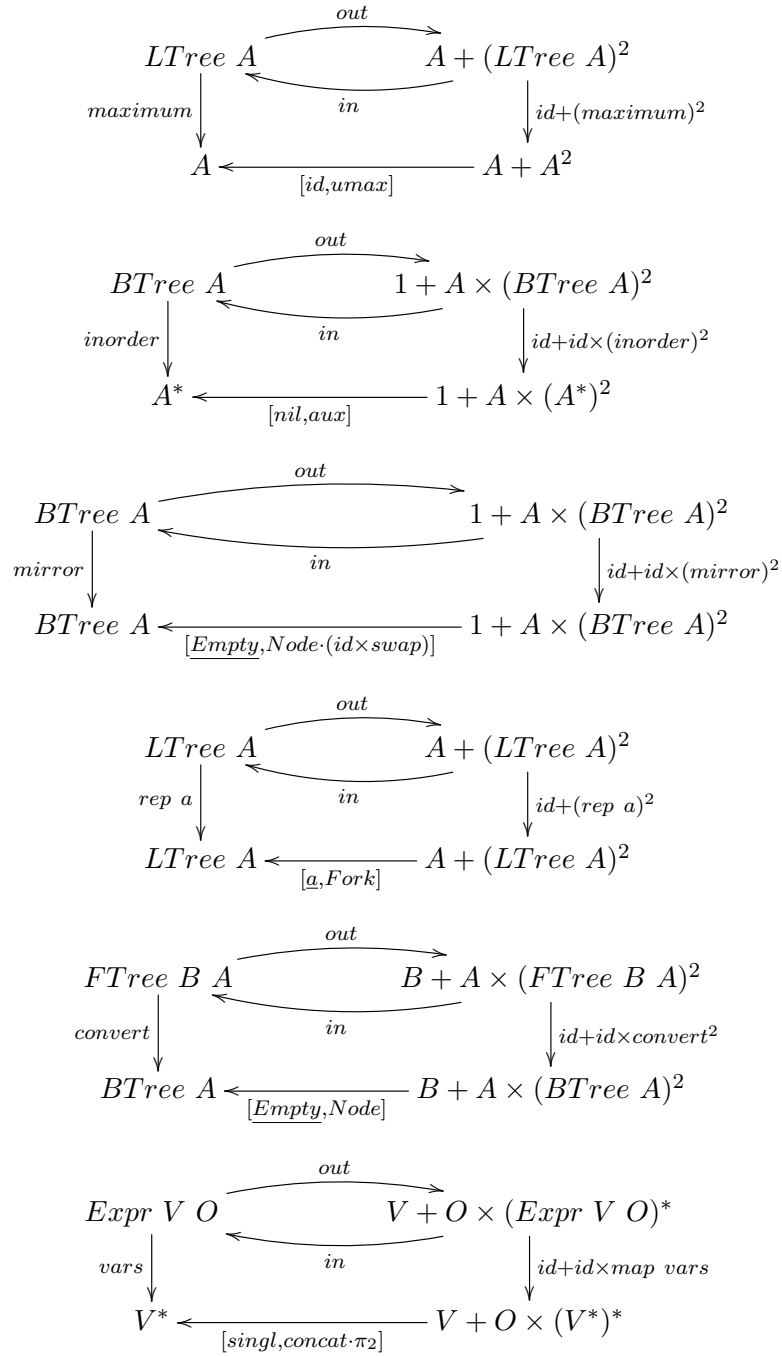
Cálculo de Programas

Resolução - Ficha 09

Eduardo Freitas Fernandes

2025

Exercício 1



Exercício 2

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

Exercício 3

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

Exercício 4

$$\begin{aligned}
& k = \llbracket (id + \langle f, id \rangle) \cdot out_{\mathcal{N}_0} \rrbracket \\
& \equiv \{\text{universal-ana}\} \\
& out_* \cdot k = (id + id \times k) \cdot (id + \langle f, id \rangle) \cdot out_{\mathcal{N}_0} \\
& \equiv \{\text{Shunt-left, Shunt-right}\} \\
& k \cdot in_{\mathcal{N}_0} = in_* \cdot (id + id \times k) \cdot (id + \langle f, id \rangle) \\
& \equiv \{\text{Functor-+}\} \\
& k \cdot [zero, succ] = [nil, cons] \cdot (id + (id \times k) \cdot \langle f, id \rangle) \\
& \equiv \{\text{Absorção-}\times, \text{Absorção-+}\} \\
& k \cdot [zero, succ] = [nil, cons \cdot \langle f, k \rangle] \\
& \equiv \{\text{Fusão-+}, \text{Eq-+}, \text{pointwise}\} \\
& \begin{cases} k \ 0 = [] \\ k \ (n + 1) = f \ n : k \ n \end{cases}
\end{aligned}$$

Exercício 5

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

Exercício 6

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

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)))
```