Cálculo de Programas

Resolução - Ficha 10

Eduardo Freitas Fernandes

2026

Exercício 1

$$
\begin{tikzcd}
A^* \arrow[rr, bend left, "\mathsf{out}"] \arrow[d, "{[\!(g)\!]}"'] & & A + (\mathit{LTree}\ A)^2 \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "id + id \times [\!(g)\!]"] \\
A \arrow[rr, bend right, "g"'] & & A + A^2
\end{tikzcd}
$$

O anamorfismo de g inverte uma lista, ou seja, é a função *reverse*.

Exercício 2

$$
\begin{tikzcd}
\mathbb{N}_0 \arrow[rr, bend left, "\mathsf{out}"] \arrow[d, "\widehat{\theta}"'] & & 1 + \mathbb{N}_0 \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "id + [\!(g)\!]"] \\
\mathbb{N}_0 \times \mathbb{N}_0 \arrow[rr, bend right, "g"'] & & 1 + \mathbb{N}_0 \times \mathbb{N}_0
\end{tikzcd}
$$

$$
\begin{aligned}
& g :: (Int, Int) \to () + (Int, Int) \\
& g = \widehat{(\leqslant)} \to i_1 \cdot (!), i_2 \cdot (id \times \mathsf{succ})
\end{aligned}
$$

Exercício 3

$$
\begin{array}{cl}
& length \cdot concat = sum \cdot \mathsf{map}\ \ length \\
\equiv & \qquad \{ \text{ def. } concat\text{, def. } sum \ \} \\
& length \cdot \llparenthesis\, [nil, \mathsf{conc}]\, \rrparenthesis = \llparenthesis\, [\underline{0}, add]\, \rrparenthesis \cdot \mathsf{map}\ \ length \\
\equiv & \qquad \{ \text{ absorção-cata } \} \\
& length \cdot \llparenthesis\, [nil, \mathsf{conc}]\, \rrparenthesis = \llparenthesis\, [\underline{0}, add] \cdot (id + length \times id)\, \rrparenthesis \\
\Longleftarrow & \qquad \{ \text{ absorção-+, fusão-cata } \} \\
& length \cdot [nil, \mathsf{conc}] = [\underline{0}, add \cdot (length \times id)] \cdot (id + id \times length) \\
\equiv & \qquad \{ \text{ fusão-+, absorção-+, eq-+ } \} \\
& \left\{ \begin{array}{l} length \cdot nil = \underline{0} \\ length \cdot \mathsf{conc} = add \cdot (length \times id) \cdot (id \times length) \end{array} \right.
\end{array}
$$

$$\equiv \quad \{ \text{ functor-}\times \ \}$$

$$length \cdot \mathsf{conc} = add \cdot (length \times length)$$

$$\equiv \quad \{ \text{ ?????? } \}$$

$$true$$

$\square$

Exercício 4

$$length = sum \cdot (\mathsf{map}\ \ \underline{1})$$

$$\equiv \quad \{ \text{ def. } sum \ \}$$

$$length = (\!|\, [\underline{0}, add] \,|\!) \cdot \mathsf{map}\ \ \underline{1}$$

$$\equiv \quad \{ \text{ absorção-cata } \}$$

$$length = (\!|\, [\underline{0}, add] \cdot B\ (\underline{1}, id) \,|\!)$$

$$\equiv \quad \{ \text{ def. bi-functor de listas } \}$$

$$length = (\!|\, [\underline{0}, add] \cdot (id + \underline{1} \times id) \,|\!)$$

$$\equiv \quad \{ \text{ absorção-}+ \ \}$$

$$length = (\!|\, [\underline{0}, add \cdot (\underline{1} \times id)] \,|\!)$$

$$\equiv \quad \{ \ add \cdot (\underline{1} \times id) = \mathsf{succ} \cdot \pi_2 \ \}$$

$$length = (\!|\, [\underline{0}, \mathsf{succ} \cdot \pi_2] \,|\!)$$

Podemos verificar que $add \cdot (\underline{1} \times id)$ é equivalente a $\mathsf{succ} \cdot \pi_2$, pois esta recebe um par de valores, destroi o primeiro (transformando no valor 1) e mantém o segundo, de seguida soma o segundo a 1, obtendo assim o valor equivalente ao seu sucessor.

$$length = length \cdot (\mathsf{map}\ \ f)$$

$$\equiv \quad \{ \text{ def. } length \ \}$$

$$length = (\!|\, [\underline{0}, \mathsf{succ} \cdot \pi_2] \,|\!) \cdot (\mathsf{map}\ \ f)$$

$$\equiv \quad \{ \text{ absorção-cata } \}$$

$$length = (\!|\, [\underline{0}, \mathsf{succ} \cdot \pi_2] \cdot B\ (f, id) \,|\!)$$

$$\equiv \quad \{ \text{ def. bi-functor de listas } \}$$

$$length = (\!|\, [\underline{0}, \mathsf{succ} \cdot \pi_2] \cdot (id + f \times id) \,|\!)$$

$$\equiv \quad \{ \text{ absorção-}+\text{, natural-}id \ \}$$

$$lengt = (\!|\, [\underline{0}, \mathsf{succ} \cdot \pi_2 \cdot (f \times id)] \,|\!)$$

$$\equiv \quad \{ \text{ natural-}\pi_2 \ \}$$

$$length = (\!|\, [\underline{0}, \mathsf{succ} \cdot id \cdot \pi_2] \,|\!)$$

$$\equiv \quad \{ \text{ natural-}id \ \}$$

$$length = (\!|\, [\underline{0}, \mathsf{succ} \cdot \pi_2] \,|\!)$$

Exercício 5

$$depth \cdot \mathsf{LTree}\ f = depth$$

$$\equiv \quad \{ \text{ def. } depth \ \}$$

$$
\begin{array}{ll}
& (\![[\underline{1}, \mathsf{succ} \cdot umax]]\!) \cdot T\ f = depth \\
\equiv & \quad \{ \text{ absorção-cata } \} \\
& (\![[\underline{1}, \mathsf{succ} \cdot umax] \cdot B\ (f, id)]\!) = depth \\
\equiv & \quad \{ \text{ def. bi-functor de LTree } \} \\
& (\![[\underline{1}, \mathsf{succ} \cdot umax] \cdot (f + (id \times id))]\!) = depth \\
\equiv & \quad \{ \text{ functor-id-}\times\text{, absorção-}+ \} \\
& (\![[\underline{1} \cdot f, \mathsf{succ} \cdot umax \cdot (id \times id)]]\!) = depth \\
\equiv & \quad \{ \text{ fusão-const, natural-}id \} \\
& (\![[\underline{1}, \mathsf{succ} \cdot umax]]\!) = depth
\end{array}
$$

Exercício 6

$$
\begin{array}{l}
bubble\ (x : y : xs) \\
\quad \mid x > y = y : bubble\ (x : xs) \\
\quad \mid otherwise = x : bubble\ (y : xs) \\
bubble\ x = x
\end{array}
$$

O primeiro passo será substituir o nome da função por *divide* e de seguida remover as chamadas recursivas:

$$
\begin{array}{l}
divide\ (x : y : xs) \\
\quad \mid x > y = y \ldots (x : xs) \\
\quad \mid otherwise = x \ldots (y : xs) \\
divide\ x = x
\end{array}
$$

De seguida emparelhamos o resultado e por fim injetamos o resultado num co-produto, dado que existem dois tipos de resultado:

$$
\begin{array}{l}
divide\ (x : y : xs) \\
\quad \mid x > y = i_2\ (y, (x : xs)) \\
\quad \mid otherwise = i_2\ (x, (y : xs)) \\
divide\ x = i_1\ x
\end{array}
$$

Podemos então inferir o tipo da função *divide*:

$$divide : A^* \rightarrow A{*}^* + A \times A^*$$

Verificamos que o bi-functor necessário para formar este hilomorfismo será o das **Listas com Sentinela**:

$$\mathbf{data}\ SList\ a\ b = Stl\ b \mid Cons\ a\ (SList\ a\ b)$$

Temos então:

$$B\ (Z, X, Y) = Z + X + Y$$

Pelo tipo de *divide* podemos inferir o tipo de *conquer*:

$$
\begin{array}{l}
conquer :: [a] + (a, [a]) \rightarrow [a] \\
conquer = [id, cons]
\end{array}
$$

Exercício 7

$\mathbf{data}\ \mathit{Point}\ a = \mathit{Point}\ \{x :: a, y :: a, z :: a\}\ \mathbf{deriving}\ (\mathit{Eq}, \mathit{Show})$

$\mathit{outPoint} = \langle\langle x, y\rangle, z\rangle$

-- inPoint = uncurry (uncurry Point)

$\mathit{inPoint}\ ((a, b), c) = \mathit{Point}\ a\ b\ c$

Exercício 8

$\mathbf{module}\ \mathit{B\_Tree}\ \mathbf{where}$

$\mathbf{import}\ \mathit{Cp}$

-- (1) Datatype definition

$\mathbf{data}\ \mathit{B\_Tree}\ a = \mathit{Nil} \mid \mathit{Block}\ \{\mathit{leftmost} :: \mathit{B\_Tree}\ a, \mathit{block} :: [(a, \mathit{B\_Tree}\ a)]\}$
$\quad \mathbf{deriving}\ (\mathit{Show})$

$\mathit{inB\_Tree} :: () + (\mathit{B\_Tree}\ a, [(a, \mathit{B\_Tree}\ a)]) \to \mathit{B\_Tree}\ a$

$\mathit{inB\_Tree} = [\underline{\mathit{Nil}}, \widehat{\mathit{Block}}]$

$\mathit{outB\_Tree} :: \mathit{B\_Tree}\ a \to () + (\mathit{B\_Tree}\ a, [(a, \mathit{B\_Tree}\ a)])$
$\mathit{outB\_Tree}\ \mathit{Nil} = i_1\ ()$
$\mathit{outB\_Tree}\ (\mathit{Block}\ l\ b) = i_2\ (l, b)$

$\mathit{baseB\_Tree}\ g\ f = \mathit{id} + (f \times (\mathsf{map}\ \ (g \times f)))$

-- (2) Ana + cata + hylo

$\mathit{recB\_Tree}\ f = \mathit{baseB\_Tree}\ \mathit{id}\ f$

$(\!|g|\!) = g \cdot (\mathit{recB\_Tree}\ (\!|g|\!)) \cdot \mathit{outB\_Tree}$

$\mathit{anaB\_Tree}\ g = \mathit{inB\_Tree} \cdot (\mathit{recB\_Tree}\ (\mathit{anaB\_Tree}\ g)) \cdot g$

$\mathit{hyloB\_Tree}\ f\ g = (\!|f|\!) \cdot \mathit{anaB\_Tree}\ g$

-- (3) Map

$\mathbf{instance}\ \mathit{Functor}\ \mathit{B\_Tree}\ \mathbf{where}$
$\mathsf{fmap}\ f = (\!|\mathit{inB\_Tree} \cdot \mathit{baseB\_Tree}\ f\ \mathit{id}|\!)$

-- (4) Examples

-- (4.1) Count and depth

$\mathit{countB\_Tree} = (\!|[\mathit{zero},$
$\quad \mathit{add} \cdot (\mathit{id} \times \mathit{sum} \cdot \mathsf{map}\ \ (\mathsf{succ} \cdot \pi_2))]|\!)$

$\mathit{depthB\_Tree} = (\!|[\mathit{zero},$
$\quad \mathsf{succ} \cdot \mathit{umax} \cdot (\mathit{id} \times \mathit{maximum} \cdot \mathsf{map}\ \ \pi_2)]|\!)$

-- (4.2) Serialization

-- in-order traversal

$\mathit{inordtB\_Tree} = (\!|[\mathit{nil}, \mathsf{conc} \cdot (\mathit{id} \times (\mathit{concat} \cdot \mathsf{map}\ \ \mathit{cons}))]|\!)$

-- pre-order traversal

$\mathit{preordtB\_Tree} = (\!|[\mathit{nil}, \mathit{aux}]|\!)$
$\quad \mathbf{where}$
$\quad\quad \mathit{aux}\ (l, [\,]) = l$
$\quad\quad \mathit{aux}\ (l, (h, a) : t) = (h : l) +\!\!+\ a +\!\!+ (\mathit{concat}\ (\mathsf{map}\ \ \mathit{cons}\ t))$

-- post-order traversal

$\mathit{postordtB\_Tree} = (\!|[\mathit{nil}, \mathsf{conc} \cdot (\mathit{id} \times (\mathit{concat} \cdot \mathsf{map}\ \ \mathit{aux}))]|\!)$
$\quad \mathbf{where}$
$\quad\quad \mathit{aux}\ (x, l) = l +\!\!+ [x]$