Cálculo de Programas
Resolução - Ficha 07

Eduardo Freitas Fernandes

2025

Exercício 1

a)

```haskell
prod [] = 1
prod (h:t) = h * prod t
-- ou
prod = foldr (*) 1
```

$$prod = (\![\,\underline{1}, mul\,]\!)$$

b)

```haskell
reverse [] = []
reverse (h:t) = reverse t ++ [h]
-- ou
reverse = foldr (flip (++) . singl)
```

$$reverse = (\![\,nil, conc \cdot swap \cdot (singl \times id)\,]\!)$$

c)

```haskell
concat [] = []
concat (h:t) = h ++ concat t
-- ou
concat = foldr (++) []
```

$$concat = (\![\,nil, conc\,]\!)$$

d)

```haskell
map f [] = []
map f (h:t) = f h : map f t
```

$$map\ f = (\![\,nil, cons \cdot (f \times id)\,]\!)$$

e)

```haskell
maximum [x] = x
maximum (h:t) = max h (maximum t)
-- umax = uncurry max
```

$$maximum = (\![\,id, umax\,]\!)$$

f)

```haskell
filter p [] = []
filter p (h:t) = x ++ filter p t
    where x = if (p h)
              then [h]
              else []
```

$$filter\ p = (\![\,nil, conc \cdot (p \rightarrow singl, nil) \times id\,]\!)$$

Exercício 2

$$\begin{aligned}
& sumprod\ a = (a*) \cdot sum \\
\equiv\ & \{\text{Def. sum, Def. sumprod a}\} \\
& (\!|\, [zero, add \cdot ((a*) \times id)] \,|\!) = (a*) \cdot (\!|\, [zero, add] \,|\!) \\
\Longleftarrow\ & \{\text{Fusão-cata}\} \\
& (a*) \cdot [zero, add] = [zero, add \cdot ((a*) \times id)] \cdot (id + id \times (a*)) \\
\equiv\ & \{\text{Fusão-+, Absorção-+}\} \\
& [(a*) \cdot zero, (a*) \cdot add] = [zero, add \cdot ((a*) \times id) \cdot (id \times (a*))] \\
\equiv\ & \{\text{Eq-+}\} \\
& \begin{cases} (a*) \cdot zero = zero \\ (a*) \cdot add = add \cdot ((a*) \times id) \cdot \end{cases} \\
\equiv\ & \{\text{Functor-}\times\} \\
& \begin{cases} (a*) \cdot zero = zero \\ (a*) \cdot add = add \cdot ((a*) \times (a*)) \end{cases} \\
\equiv\ & \{\text{pointwise, def. add, def. zero}\} \\
& \begin{cases} a * 0 = 0 \\ a * (x + y) = (a * x) + (a * y) \end{cases}
\end{aligned}$$

Exercício 3

$$length = (\!|\, [zero, succ \cdot \pi_2] \,|\!)$$

$$\begin{aligned}
& f \cdot length = (\!|\, [zero, (2+).\pi_2] \,|\!) \\
\equiv\ & \{\text{Def. length}\} \\
& f \cdot (\!|\, [zero, succ.\pi_2] \,|\!) = (\!|\, [zero, (2+).\pi_2] \,|\!) \\
\Longleftarrow\ & \{\text{Fusão-cata}\} \\
& f \cdot [zero, succ \cdot \pi_2] = [zero, (2+) \cdot \pi_2] \cdot (id + id \times f) \\
\equiv\ & \{\text{Fusão-+, Absorção-+, Natural-id}\} \\
& [f \cdot zero, f \cdot succ \cdot \pi_2] = [zero, (2+) \cdot \pi_2 \cdot (id \times f)] \\
\equiv\ & \{\text{Eq-+, Natural-}\pi_2\} \\
& \begin{cases} f \cdot zero = zero \\ f \cdot succ \cdot \pi_2 = (2+) \cdot f \cdot \pi_2 \end{cases} \\
\equiv\ & \{\text{pointwise}\} \\
& \begin{cases} (f \cdot zero)\ n = zero\ n \\ (f \cdot succ \cdot \pi_2)\ (x, y) = ((2+) \cdot f \cdot \pi_2)\ (x, y) \end{cases} \\
\equiv\ & \{\text{Def. composição, Def. } \pi_2\} \\
& \begin{cases} f\ 0 = 0 \\ f\ (y + 1) = 2 + f\ y \end{cases}
\end{aligned}$$

Sendo que $2 + 0 = 2$ e $2 * 0 = 0$, concluimos que $f = (2*)$.

Exercício 4

$$
\begin{aligned}
& foldr\ \overline{\pi_2}\ i = f \\
\equiv\ & \{\text{Def. foldr}\} \\
& (\!|\, [\underline{i}, uncurry\ curry\ \pi_2] \,|\!) = f \\
\equiv\ & \{\text{Universal-cata}\} \\
& f \cdot in = [\underline{i}, \pi_2] \cdot (id + id \times f) \\
\equiv\ & \{\text{Def. in, Fusão-+, Absorção-+, Eq-+}\} \\
& \begin{cases} f \cdot nil = \underline{i} \\ f \cdot cons = \pi_2 \cdot (id \times f) \end{cases} \\
\equiv\ & \{\text{pointwise}\} \\
& \begin{cases} f\ [\,] = i \\ f\ (h : t) = f\ t \end{cases}
\end{aligned}
$$

Podemos concluir que $foldr\ \overline{\pi_2}\ i$ é a função constante i.

Exercício 5

$$
\begin{aligned}
& f \cdot (for\ f\ i) = for\ f (f\ i) \\
\equiv\ & \{\text{Def. for b i}\} \\
& f \cdot (\!|\, [\underline{i}, f] \,|\!) = (\!|\, [\underline{f\ i}, f] \,|\!) \\
\Longleftarrow\ & \{\text{Fusão-cata}\} \\
& f \cdot [\underline{i}, f] = [\underline{f\ i}, f] \cdot (id + f) \\
\equiv\ & \{\text{Fusão-+, Absorção-+}\} \\
& [f \cdot \underline{i}, f \cdot f] = [\underline{f\ i} \cdot id, f \cdot f] \\
\equiv\ & \{\text{Eq-+}\} \\
& \begin{cases} f \cdot \underline{i} = \underline{f\ i} \\ f \cdot f = f \cdot f \end{cases} \\
\equiv\ & \{\text{Absorção-const}\} \\
& \begin{cases} \underline{f\ i} = \underline{f\ i} \\ f \cdot f = f \cdot f \end{cases}
\end{aligned}
$$

Exercício 6

$$
\begin{aligned}
& for\ id\ i = for \underline{i} i \\
\equiv\ & \{\text{Def. for (twice)}\} \\
& \llparenthesis [\underline{i}, id] \rrparenthesis = \llparenthesis [\underline{i}, \underline{i}] \rrparenthesis \\
\equiv\ & \{\text{Universal-cata}\} \\
& \llparenthesis [\underline{i}, id] \rrparenthesis \cdot in = [\underline{i}, \underline{i}] \cdot (id + \llparenthesis [\underline{i}, id] \rrparenthesis) \\
\equiv\ & \{\text{Cancelamento-cata, Absorção-+}\} \\
& [\underline{i}, id] \cdot (id + \llparenthesis [\underline{i}, id] \rrparenthesis) = [\underline{i}, \underline{i}] \\
\equiv\ & \{\text{Absorção-+}\} \\
& [\underline{i}, \llparenthesis [\underline{i}, id] \rrparenthesis] = [\underline{i}, \underline{i}] \\
\equiv\ & \{\text{Eq-+}\} \\
& \begin{cases} \underline{i} = \underline{i} \\ \llparenthesis [\underline{i}, id] \rrparenthesis = \underline{i} \end{cases} \\
\equiv\ & \{\text{Universal-cata}\} \\
& \begin{cases} true \\ \underline{i} \cdot in = [\underline{i}, id] \cdot (id + \underline{i}) \end{cases} \\
\equiv\ & \{\text{Absorção-+}\} \\
& \underline{i} = [\underline{i}, \underline{i}]
\end{aligned}
$$

A última expressão é verdadeira, pois o resultado de $[\underline{i}, \underline{i}]$ será sempre $\underline{i}$.

Exercício 7

```
ghci> rep f = cataNat (either (const id) (f.))
ghci> rep (2*) 0 3
3
ghci> rep ("a"++) 10 "b"
"aaaaaaaaaab"
```

$$
\begin{aligned}
\underline{id} &: B \to A^A \\
f &: C \to D \\
(f\cdot) &: C^E \to D^E
\end{aligned}
$$

Temos de ter igualdade de tipos, ou seja:

$$
A^A = D^E \implies \begin{cases} A = E \\ A = D \end{cases}
$$

Logo:

$$
\begin{aligned}
\underline{id} &: B \to A^A \\
f &: C \to A \\
(f\cdot) &: C^A \to A^A
\end{aligned}
$$

$$
\begin{tikzcd}
\mathbb{N}_0 \arrow[r, bend left, "out"] \arrow[d, "rep\ f"'] & 1 + \mathbb{N}_0 \arrow[l, bend left, "in"] \arrow[d, "id + rep\ f"] \\
A^A & 1 + C^A \arrow[l, "{[\underline{id}, (f\cdot)]}"]
\end{tikzcd}
$$

Exercício 8

```
ghci> out 0 = i1 () ; out (n+1) = i2 n
ghci> cata g = g . (id -|- (cata g)) . out
ghci> rep a = cata (either nil (a:))
ghci> :t +d rep
rep :: a -> Integer -> [a]
ghci> rep 3 4
[3,3,3,3]
ghci> rep 'l' 8
"llllllll"
ghci> -- rep é a função que repete algo n vezes
```

Exercício 9

```haskell
import Data.List (sort, nub)

type Date = String
type Player = String
type Game = String

db1 = [
    ("2023-10-01", ["Game1", "Game2"]),
    ("2023-10-02", ["Game2", "Game3"])
]

db2 = [
    ("Game1", ["PlayerA", "PlayerB"]),
    ("Game2", ["PlayerA", "PlayerC"]),
    ("Game3", ["PlayerB", "PlayerC"])
]

f :: [(Date, [Game])] -> [(Game, [Player])] -> [(Player, [Date])]
f = undefined

main :: IO ()
main = do
    let result = f db1 db2
    print result
```