

# CP - Ficha 3

## Exercício 1

Considere o diagrama

$$\begin{array}{ccc} (A \times B) \times C & \xrightarrow{\text{assocr}} & A \times (B \times C) \\ & \cong & \\ & \xleftarrow{\text{assocl}} & \end{array}$$

onde  $\text{assocl} = \langle id \times \pi_1, \pi_2 \cdot \pi_2 \rangle$ . Apresente justificações para o cálculo que se segue em que se resolve em ordem a  $\text{assocr}$  a equação  $\text{assocl} \cdot \text{assocr} = id$ :

## Resolução 1

$$\text{assocl} \cdot \text{assocr} = id$$

$$\equiv (\text{Def. assocl, 9: Fusão-}\times, 6: \text{Universal-}\times, 1: \text{Natural-id})$$

$$\left\{ \begin{array}{l} (id \times \pi_1) \cdot \text{assocr} = \pi_1 \\ \pi_2 \cdot \pi_2 \cdot \text{assocr} = \pi_2 \end{array} \right.$$

$$\equiv (10: \text{Def-}\times, 1: \text{Natural-id, 9: Fusão-}\times, 6: \text{Universal-}\times)$$

$$\left\{ \left\{ \begin{array}{l} \pi_1 \cdot \text{assocr} = \pi_1 \cdot \pi_1 \\ \pi_1 \cdot \pi_2 \cdot \text{assocr} = \pi_2 \cdot \pi_1 \end{array} \right. \right. \\ \left. \pi_2 \cdot \pi_2 \cdot \text{assocr} = \pi_2 \right.$$

$$\equiv (6: \text{Universal-}\times)$$

$$\left\{ \begin{array}{l} \pi_1 \cdot \text{assocr} = \pi_1 \cdot \pi_1 \\ \left\{ \begin{array}{l} \pi_1 \cdot \pi_2 \cdot \text{assocr} = \pi_2 \cdot \pi_1 \\ \pi_2 \cdot \pi_2 \cdot \text{assocr} = \pi_2 \end{array} \right. \end{array} \right.$$

$$\equiv (6: \text{Universal-}\times)$$

$$\left\{ \begin{array}{l} \pi_1 \cdot \text{assocr} = \pi_1 \cdot \pi_1 \\ \pi_2 \cdot \text{assocr} = \langle \pi_2 \cdot \pi_1, \pi_2 \rangle \end{array} \right.$$

$$\equiv (6: \text{Universal-}\times, 1: \text{Natural-id, 10: Def-}\times)$$

$$\text{assocr} = \langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle$$

(F1)

## Exercício 2

---

- a) Codifique (F1) diretamente em Haskell e verifique o comportamento dessa função no GHCi.
- b) De seguida, converta — por igualdade extensional — (F1) para notação Haskell pointwise que não recorra a nenhum combinador nem projecção e verifique no GHCi que as duas versões dão os mesmos resultados.

## Resolução 2

a)

```
ghci> assocr = split (p1 . p1) (p2 >< id)
ghci> assocr ((1,2),3)
(1,(2,3))
ghci> assocr (("Hi",True),3.14)
("Hi",(True,3.14))
```

b)

$$\begin{aligned} \text{assocr} &= \langle \pi_1 \cdot \pi_1, \pi_2 \times \text{id} \rangle \\ &\equiv (72: \text{Ig. Ext.}) \\ \text{assocr} ((a,b),c) &= \langle \pi_1 \cdot \pi_1, \pi_2 \times \text{id} \rangle ((a,b),c) \\ &\equiv (77: \text{Def-split}) \\ \text{assocr} ((a,b),c) &= (\pi_1 \cdot \pi_1 ((a,b),c), \pi_2 \times \text{id} ((a,b),c)) \\ &\equiv (78: \text{Def-}\times, 74: \text{Def-id}) \\ \text{assocr} ((a,b),c) &= (\pi_1 \cdot \pi_1 ((a,b),c), (\pi_2 (a,b),c)) \\ &\equiv (79: \text{Def-proj}, 73: \text{Def-comp}) \\ \text{assocr} ((a,b),c) &= (a, (b,c)) \end{aligned}$$

```
ghci> assocr ((a,b),c) = (a,(b,c))
ghci> assocr ((1,2),3)
(1,(2,3))
ghci> assocr (("Hi",True),3.14)
("Hi",(True,3.14))
```

## Extra - Chegue ao tipo mais geral de assocl através da sua definição (*point-free*)

$$\text{assocl} = \langle id \times \pi_1, \pi_2 \cdot \pi_2 \rangle$$

$$id \times \pi_1 :: A \times (B \times C) \rightarrow A \times B$$

$$\pi_2 \cdot \pi_2 :: A \times (B \times C) \rightarrow C$$

$$\text{assocl} :: A \times (B \times C) \rightarrow (A \times B) \times C$$

## Exercício 3

---

Recorde a propriedade universal do combinador  $[f, g]$ ,

$$k = [f, g] \equiv \begin{cases} k \cdot i_1 = f \\ k \cdot i_2 = g \end{cases}$$

Demonstre a igualdade

$$[\underline{k}, \underline{k}] = \underline{k} \quad (\text{F2})$$

recorrendo à propriedade universal acima e a uma lei que qualquer função constante  $\underline{k}$  satisfaz. (Ver no [formulário](#).)

## Resolução 3

$$[\underline{k}, \underline{k}] = \underline{k}$$

$\equiv$

(17: Universal-+)

$$\begin{cases} \underline{k} \cdot i_1 = \underline{k} \\ \underline{k} \cdot i_2 = \underline{k} \end{cases}$$

$\equiv$

(3: Fusão-const)

$$\begin{cases} \underline{k} = \underline{k} \\ \underline{k} = \underline{k} \end{cases} \text{ c.q.d.}$$

# Exercício 4

Os isomorfismos

$$\begin{array}{ccc} & \xrightarrow{\text{distr}} & \\ A \times (B + C) & \cong & A \times B + A \times C \\ & \xleftarrow{\text{undistr}} & \end{array}$$

estudados na aula teórica estão codificados na biblioteca *Cp.hs*.

Supondo  $A = \text{String}$ ,  $B = \mathbb{B}$  e  $C = \mathbb{Z}$ ,

a) aplique no GHCi `undistr`, alternativamente, aos pares ("CP", True) ou ("LEI", 1);

b) verifique que  $(\text{distr} \cdot \text{undistr})\ x = x$  para essas (e quaisquer outras) situações que possa testar.

## Resolução 4

TODO: verificar solução

a)

```
let alter1 = i1 ("CP", True) :: Either (String, Bool) (String, Int)
let alter2 = i2 ("LEI", 1)    :: Either (String, Bool) (String, Int)

ghci> f = undistr . either (const alter1) (const alter2)
ghci> f (i1 ())
("CP", Left True)
ghci> f (i2 ())
("LEI", Right 1)
```

b)

```
ghci> (distr . undistr) alter1
Right ("CP", True)
ghci> (distr . undistr) alter2
Left ("LEI", 1)
```

## Exercício 5

Recorde a função

$$\alpha = [\langle \underline{\text{False}}, id \rangle, \langle \underline{\text{True}}, id \rangle]$$

da ficha anterior. Mostre, usando a propriedade **Universal-+** (17), que  $\alpha$  se pode escrever em Haskell da forma seguinte:

$$\begin{aligned}\alpha (i_1 a) &= (\text{False}, a) \\ \alpha (i_2 a) &= (\text{True}, a)\end{aligned}$$

Codifique  $\alpha$  e teste-a no GHCi, onde  $i_1$  (resp.  $i_2$ ) se escreve `Left` (resp. `Right`).

## Resolução 5

$$\begin{aligned}& [\langle \underline{\text{False}}, id \rangle, \langle \underline{\text{True}}, id \rangle] \\& = \hspace{15em} (17: \text{Universal-+}) \\& \left\{ \begin{array}{l} \alpha \cdot i_1 = \langle \underline{\text{False}}, id \rangle \\ \alpha \cdot i_2 = \langle \underline{\text{True}}, id \rangle \end{array} \right. \\& \equiv \hspace{15em} (72: \text{Ig. Ext.}) \\& \left\{ \begin{array}{l} (\alpha \cdot i_1) a = \langle \underline{\text{False}}, id \rangle a \\ (\alpha \cdot i_2) a = \langle \underline{\text{True}}, id \rangle a \end{array} \right. \\& \equiv \hspace{10em} (73: \text{Def-comp}, 77: \text{Def-split}) \\& \left\{ \begin{array}{l} \alpha (i_1 a) = (\underline{\text{False}} a, id a) \\ \alpha (i_2 a) = (\underline{\text{True}} a, id a) \end{array} \right. \\& \equiv \hspace{10em} (75: \text{Def-const}, 74: \text{Def-id}) \\& \left\{ \begin{array}{l} \alpha (i_1 a) = (\text{False}, a) \\ \alpha (i_2 a) = (\text{True}, a) \end{array} \right. \text{ c.q.m.}\end{aligned}$$

```
ghci> alpha = either (split (const False) id) (split (const True) id)
ghci> alpha (Left 42)
(False,42)
ghci> alpha (Right 42)
(True,42)
```

## Exercício 6

Recorra às leis dos coprodutos para mostrar que a definição que conhece da função factorial,

$$\begin{aligned} fac\ 0 &= 1 \\ fac\ (n + 1) &= (n + 1) * fac\ n \end{aligned}$$

é equivalente à equação seguinte

$$fac \cdot [0, succ] = [1, mul \cdot \langle succ, fac \rangle]$$

onde

$$\begin{aligned} succ\ n &= n + 1 \\ mul\ (a, b) &= a * b \end{aligned}$$

## Resolução 6

$$\begin{aligned} fac \cdot [0, succ] &= [1, mul \cdot \langle succ, fac \rangle] \\ &\equiv (20: \text{Fusão-+}) \\ [fac \cdot 0, fac \cdot succ] &= [1, mul \cdot \langle succ, fac \rangle] \\ &\equiv (27: \text{Eq-+}) \\ \begin{cases} fac \cdot 0 = 1 \\ fac \cdot succ = mul \cdot \langle succ, fac \rangle \end{cases} \\ &\equiv (72: \text{Ig. Ext.}) \\ \begin{cases} (fac \cdot 0)\ n = 1\ n \\ (fac \cdot succ)\ n = mul \cdot \langle succ, fac \rangle\ n \end{cases} \\ &\equiv (73: \text{Def-comp}, 75: \text{Def-const}) \\ \begin{cases} fac\ 0 = 1 \\ fac\ (succ\ n) = mul\ (\langle succ, fac \rangle\ n) \end{cases} \\ &\equiv (\text{Def. succ}, 77: \text{Def-split}) \\ \begin{cases} fac\ 0 = 1 \\ fac\ (n + 1) = mul\ (succ\ n, fac\ n) \end{cases} \\ &\equiv (\text{Def. mul}, \text{Def succ}) \\ \begin{cases} fac\ 0 = 1 \\ fac\ (n + 1) = (n + 1) * fac\ n \end{cases} \quad \text{c.q.m.} \end{aligned}$$

## Exercício 7

A função  $\text{in} = [\underline{0}, \text{succ}]$  da questão anterior exprime, para  $\text{succ } n = n + 1$ , a forma como os números naturais ( $\mathbb{N}_0$ ) são gerados a partir do número 0, de acordo com o diagrama seguinte:

$$\begin{array}{ccccc}
 1 & \xrightarrow{i_1} & 1 + \mathbb{N}_0 & \xleftarrow{i_2} & \mathbb{N}_0 \\
 & \searrow \underline{0} & \downarrow \text{in} = [\underline{0}, \text{succ}] & \swarrow \text{succ} & \\
 & & \mathbb{N}_0 & & 
 \end{array} \quad (\text{F3})$$

Sabendo que o tipo 1 coincide com o tipo () em Haskell e é habitado por um único elemento, também designado por (), calcule a inversa de  $\text{in}$ ,

$$\begin{cases} \text{out } 0 = i_1 () \\ \text{out } (n + 1) = i_2 n \end{cases} \quad (\text{F4})$$

resolvendo em ordem a out a equação

$$\text{in} \cdot \text{out} = id \quad (\text{F5})$$

e introduzindo variáveis.

## Resolução 7

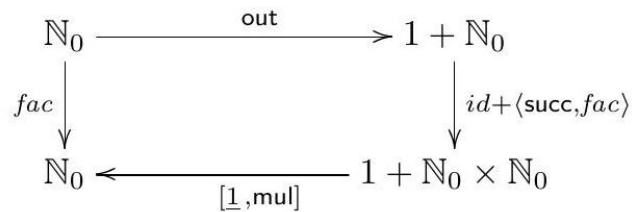
$$\begin{aligned}
 & \text{out} \cdot \text{in} = id \\
 & \equiv \quad (\text{Def. in}) \\
 & \text{out} \cdot [\underline{0}, \text{succ}] = id \\
 & \equiv \quad (20: \text{Fusão-+}) \\
 & [\text{out} \cdot \underline{0}, \text{out} \cdot \text{succ}] = id \\
 & \equiv \quad (4: \text{Absorção-const}) \\
 & [\underline{\text{out } 0}, \text{out} \cdot \text{succ}] = id \\
 & \equiv \quad (17: \text{Universal-+}, 1: \text{Natural-id}) \\
 & \begin{cases} \underline{\text{out } 0} = i_1 \\ \text{out} \cdot \text{succ} = i_2 \end{cases} \\
 & \equiv \quad (72: \text{Ig. Ext.}) \\
 & \begin{cases} \underline{\text{out } 0} () = i_1 () \\ \text{out} \cdot \text{succ } n = i_2 n \end{cases} \\
 & \equiv \quad (75: \text{Def-const}, \text{Def. succ}) \\
 & \begin{cases} \text{out } 0 = i_1 () \\ \text{out } (n + 1) = i_2 n \end{cases} \quad \text{c.q.d.}
 \end{aligned}$$

## Exercício 8

Verifique no GHCi que a seguinte função

$$fac = [\underline{1}, mul] \cdot (id + \langle succ, fac \rangle) \cdot out$$

a que corresponde o diagrama



calcula o factorial da sua entrada, assumindo  $out$  (F4) e  $mul(a, b) = a * b$  já definidas.

## Resolução 8

$$fac = [\underline{1}, mul] \cdot (id + \langle succ, fac \rangle) \cdot out$$

```
mul :: Num a => (a, a) -> a
mul (x, y) = x * y
```

```
out :: (Eq b, Num b) => b -> Either () b
out x = case x of
  0 -> i1 ()
  n -> i2 (n - 1)
```

```
fac :: Int -> Int
fac = either (const 1) mul . (id |- split succ fac) . out
```

```
fac 0 = 1
fac 3 = 6
fac 5 = 120
```



# Exercício 9

## Questão prática (...)

**NB:** usa-se a notação  $X^*$  para designar o tipo  $[X]$  em Haskell.

### **Problem requirements:**

The automatic generation of *bibliographies* in the LATEX text preparation system is based bibliographic databases from which the following information can be extracted:

$$Bib = (Key \times Aut^*)^*$$

It associates authors ( $Aut$ ) to citation keys ( $Key$ ).

Whenever LATEX processes a text document, it compiles all occurrences of citation keys in an auxiliary file

$$Aux = (Pag \times Key^*)^*$$

associating pages ( $Pag$ ) to the citation keys that occur in them.

An **author index** is an appendix to a text (e.g. book) indicating, in alphabetical order, the names of authors mentioned and the ordered list of pages where their works are cited, for example:

```
Arbib, M. A. – 10, 11
Bird, R. – 28
Horowitz, E. – 2, 3, 15, 16, 19
Hudak, P. – 11, 12, 29
Jones, C. B. – 3, 7, 28
Manes, E. G. – 10, 11
Sahni, S. – 2, 3, 15, 16, 19
Spivey, J.M. – 3, 7
Wadler, P. – 2, 3
```

The above structure can be represented by the type

$$Ind = (Aut \times Pag^*)^*$$

listing authors ( $Aut$ ) and the respective pages where they are mentioned ( $Pag$ ).

Write a Haskell function  $mkInd : Bib \times Aux \rightarrow Ind$  that generates author indices ( $Ind$ ) from  $Bib$  and  $Aux$ .

**Important:** Structure your solution across the  $f \cdot g$ ,  $\langle f, g \rangle$  and  $f \times g$  combinators that can be found in library [Cp.hs](#). Use **diagrams** to plan your proposed solution, which should avoid re-inventing functions over lists already available in the Haskell [standard libraries](#).

# Resolução 9

TODO