# CP - Ficha 3

## Exercício 1

Considere o diagrama

$$(A \times B) \times C \quad \cong \quad A \times (B \times C)$$

com as setas $\xrightarrow{\text{assocr}}$ e $\xleftarrow{\text{assocl}}$.

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 assocr a equação $\text{assocl} \cdot \text{assocr} = id$:

## Resolução 1

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

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

$$\begin{cases} (id \times \pi_1) \cdot \text{assocr} = \pi_1 \\ \pi_2 \cdot \pi_2 \cdot \text{assocr} = \pi_2 \end{cases}$$

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

$$\begin{cases} \begin{cases} \pi_1 \cdot \text{assocr} = \pi_1 \cdot \pi_1 \\ \pi_1 \cdot \pi_2 \cdot \text{assocr} = \pi_2 \cdot \pi_1 \end{cases} \\ \pi_2 \cdot \pi_2 \cdot \text{assocr} = \pi_2 \end{cases}$$

$\equiv \quad$ (6: Universal-$\times$)

$$\begin{cases} \pi_1 \cdot \text{assocr} = \pi_1 \cdot \pi_1 \\ \begin{cases} \pi_1 \cdot \pi_2 \cdot \text{assocr} = \pi_2 \cdot \pi_1 \\ \pi_2 \cdot \pi_2 \cdot \text{assocr} = \pi_2 \end{cases} \end{cases}$$

$\equiv \quad$ (6: Universal-$\times$)

$$\begin{cases} \pi_1 \cdot \text{assocr} = \pi_1 \cdot \pi_1 \\ \pi_2 \cdot \text{assocr} = \langle \pi_2 \cdot \pi_1, \pi_2 \rangle \end{cases}$$

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

$$\text{assocr} = \langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle \tag{F1}$$

# 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)**

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

$$\equiv \qquad \qquad \qquad \qquad \text{(72: Ig. Ext.)}$$

$$assocr\ ((a,b),c) = \langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle\ ((a,b),c)$$

$$\equiv \qquad \qquad \qquad \qquad \text{(77: Def-split)}$$

$$assocr\ ((a,b),c) = (\pi_1 \cdot \pi_1\ ((a,b),c), \pi_2 \times id\ ((a,b),c))$$

$$\equiv \qquad \qquad \qquad \text{(78: Def-}\times\text{, 74: Def-id)}$$

$$assocr\ ((a,b),c) = (\pi_1 \cdot \pi_1\ ((a,b),c), (\pi_2\ (a,b),c))$$

$$\equiv \qquad \qquad \text{(79: Def-proj, 73: Def-comp)}$$

$$assocr\ ((a,b),c) = (a,(b,c))$$

```
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** $\mathrm{assocl}$ **através da sua definição (*point-free*)**

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

$$id \times \pi_1 :: A \times (B \times C) \to A \times B$$
$$\pi_2 \cdot \pi_2 :: A \times (B \times C) \to C$$
$$\mathrm{assocl} :: A \times (B \times C) \to (A \times B) \times C$$

# 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

$$[\underline{k}, \underline{k}] = \underline{k} \tag{F2}$$

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

# Resolução 3

$$[\underline{k}, \underline{k}] = \underline{k}$$
$$\equiv \qquad \qquad \text{(17: Universal-+)}$$
$$\begin{cases} \underline{k} \cdot i_1 = \underline{k} \\ \underline{k} \cdot i_2 = \underline{k} \end{cases}$$
$$\equiv \qquad \qquad \text{(3: Fusão-const)}$$
$$\begin{cases} \underline{k} = \underline{k} \\ \underline{k} = \underline{k} \end{cases} \quad \text{c.q.d.}$$

# Exercício 4

Os isomorfismos

$$A \times (B + C) \underset{\text{undistr}}{\overset{\text{distr}}{\cong}} A \times B + A \times C$$

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

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

a) aplique no GHCi $\text{undistr}$, alternativamente, aos pares $(\text{"CP"}, \text{True})$ ou $(\text{"LEI"}, 1)$ ;

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)

```haskell
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)

```haskell
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 $\text{Universal-}+$ $(17)$, que $\alpha$ se pode escrever em Haskell da forma seguinte:

$$\alpha \ (i_1 \ a) = (\text{False}, a)$$
$$\alpha \ (i_2 \ a) = (\text{True}, a)$$

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

## Resolução 5

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

$$= \qquad\qquad (17: \text{Universal-}+)$$

$$\begin{cases} \alpha \cdot i_1 = \langle \underline{\text{False}}, id \rangle \\ \alpha \cdot i_2 = \langle \underline{\text{True}}, id \rangle \end{cases}$$

$$\equiv \qquad\qquad (72: \text{Ig. Ext.})$$

$$\begin{cases} (\alpha \cdot i_1) \ a = \langle \underline{\text{False}}, id \rangle \ a \\ (\alpha \cdot i_2) \ a = \langle \underline{\text{True}}, id \rangle \ a \end{cases}$$

$$\equiv \qquad\qquad (73: \text{Def-comp}, 77: \text{Def-split})$$

$$\begin{cases} \alpha \ (i_1 \ a) = (\underline{\text{False}} \ a, id \ a) \\ \alpha \ (i_2 \ a) = (\underline{\text{True}} \ a, id \ a) \end{cases}$$

$$\equiv \qquad\qquad (75: \text{Def-const}, 74: \text{Def-id})$$

$$\begin{cases} \alpha \ (i_1 \ a) = (\text{False}, a) \\ \alpha \ (i_2 \ a) = (\text{True}, a) \end{cases} \quad \text{c.q.m.}$$

```
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,

$$fac\ 0 = 1$$
$$fac\ (n+1) = (n+1) * fac\ n$$

é equivalente à equação seguinte

$$fac \cdot [\underline{0}, \text{succ}] = [\underline{1}, \text{mul} \cdot \langle \text{succ}, fac \rangle]$$

onde

$$\text{succ}\ n = n + 1$$
$$\text{mul}\ (a,b) = a * b$$

# Resolução 6

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

# Exercício 7

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


$$\text{(F3)}$$

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 $\text{in}$,

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

resolvendo em ordem a $\text{out}$ a equação

$$\text{in} \cdot \text{out} = id \tag{F5}$$

e introduzindo variáveis.

# Resolução 7

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

# Exercício 8

Verifique no GHCi que a seguinte função

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

a que corresponde o diagrama



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

## Resolução 8

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

```haskell
{-# LANGUAGE NPlusKPatterns #-}

import Cp (split, i1, i2, (-|-))

mul (x, y) = x * y

out 0 = i1 ()
out (n + 1) = i2 n

fac = either (const 1) mul . (id -|- split succ fac) . out



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

# Exercício 9

**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 $mkInd : Bib \times Aux \to Ind$ 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

```haskell
import Cp (p1, p2, (><))

import Data.List (sortBy, groupBy)
import Data.Ord (comparing)
import Data.Function (on)


-- (1: deconsAux and deconsBib)

deconsAux :: [(pag, [key])] -> [(key, pag)]
deconsAux = concatMap aux  -- same as: concat . map aux
  where
    aux :: (a, [b]) -> [(b, a)]
    aux (p, []) = []
    aux (p, k:ks) = (k, p) : aux (p, ks)


deconsBib :: [(key, [aut])] -> [(key, aut)]
deconsBib = concatMap aux
  where
    aux :: (a, [b]) -> [(a, b)]
    aux (_, []) = []
    aux (k, a:as) = (k, a) : aux (k, as)


-- (2: sortByKey)

sortByKey :: Ord key => [(key, b)] -> [(key, b)]
sortByKey = sortBy (comparing p1)


-- (3: joinAutPag)

-- joinAutPag requires both lists to be sorted by the key
joinAutPag :: Ord key => ([(key, aut)], [(key, pag)]) -> [(aut, pag)]
joinAutPag (_, []) = []
joinAutPag ([], _) = []
joinAutPag ((ka, a):as, (kp, p):ps)
  | ka == kp  = (a, p) : joinAutPag (as, ps)
  | ka <  kp  = joinAutPag (as, (kp, p):ps)
  | otherwise = joinAutPag ((ka, a):as, ps)


-- (4: groupPagByAut)

groupPagByAut :: Eq pag => [(aut, pag)] -> [(aut, [pag])]
groupPagByAut = map aux . groupBy ((==) `on` p2)
  where
    aux :: [(a, b)] -> (a, [b])
    aux l@((a, _):_) = (a, map p2 l)  -- can sort the pages here
```

```
-- (mkInd)

mkInd :: (Ord key, Eq pag) => ([(key, [aut])], [(pag, [key])]) -> [(aut, [pag])]
mkInd = groupPagByAut              -- group pages by author
      . joinAutPag                 -- join authors and pages
      . (sortByKey >< sortByKey)   -- alternatively sort lists by key
      . (deconsBib >< deconsAux)   -- alternatively deconstruct authors and pages
```

$$
\begin{array}{ccc}
Bib \times Aux & \xrightarrow{\text{(sortByKey} \times \text{sortByKey)} \cdot \text{(deconsBib} \times \text{deconsAux)}} & (Key \times Aut)^* \times (Pag \times Key)^* \\
\Big\downarrow \text{mkInd} & & \Big\downarrow \text{joinAutPag} \\
Ind & \xleftarrow{\text{groupPagByAut}} & (Aut \times Pag)^*
\end{array}
$$