CP - Ficha 3

Exercício 1

> Considere o diagrama
>
> $$\begin{tikzcd} (A \times B) \times C \arrow[rr, bend left, "\mathsf{assocr}"] \arrow[rr, phantom, "\cong"] & & A \times (B \times C) \arrow[ll, bend left, "\mathsf{assocl}"] \end{tikzcd}$$
>
> onde $\mathrm{assocl} = \langle id \times \pi_1, \pi_2 \cdot \pi_2 \rangle$. Apresente justificações para o cálculo que se segue em que se resolve em ordem a assocr a equação $\mathrm{assocl} \cdot \mathrm{assocr} = id$:

Resolução 1

$$
\begin{aligned}
& \mathrm{assocl} \cdot \mathrm{assocr} = id \\
\equiv & \quad (\text{Def. assocl, 9: Fusão-}\times\text{, 6: Universal-}\times\text{, 1: Natural-id}) \\
& \begin{cases} (id \times \pi_1) \cdot \mathrm{assocr} = \pi_1 \\ \pi_2 \cdot \pi_2 \cdot \mathrm{assocr} = \pi_2 \end{cases} \\
\equiv & \quad (\text{10: Def-}\times\text{, 1: Natural-id, 9: Fusão-}\times\text{, 6: Universal-}\times) \\
& \begin{cases} \begin{cases} \pi_1 \cdot \mathrm{assocr} = \pi_1 \cdot \pi_1 \\ \pi_1 \cdot \pi_2 \cdot \mathrm{assocr} = \pi_2 \cdot \pi_1 \end{cases} \\ \pi_2 \cdot \pi_2 \cdot \mathrm{assocr} = \pi_2 \end{cases} \\
\equiv & \quad (\text{6: Universal-}\times) \\
& \begin{cases} \pi_1 \cdot \mathrm{assocr} = \pi_1 \cdot \pi_1 \\ \begin{cases} \pi_1 \cdot \pi_2 \cdot \mathrm{assocr} = \pi_2 \cdot \pi_1 \\ \pi_2 \cdot \pi_2 \cdot \mathrm{assocr} = \pi_2 \end{cases} \end{cases} \\
\equiv & \quad (\text{6: Universal-}\times) \\
& \begin{cases} \pi_1 \cdot \mathrm{assocr} = \pi_1 \cdot \pi_1 \\ \pi_2 \cdot \mathrm{assocr} = \langle \pi_2 \cdot \pi_1, \pi_2 \rangle \end{cases} \\
\equiv & \quad (\text{6: Universal-}\times\text{, 1: Natural-id, 10: Def-}\times) \\
& \mathrm{assocr} = \langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle \qquad (\text{F1})
\end{aligned}
$$

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)

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

```
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*)

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

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

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

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

Exercício 4

> Os isomorfismos
>
> $$\begin{tikzcd} A \times (B + C) \arrow[rr, bend left, "\mathsf{distr}"] \arrow[rr, phantom, "\cong"] & & A \times B + A \times C \arrow[ll, bend left, "\mathsf{undistr}"] \end{tikzcd}$$
>
> estudados na aula teórica estão codificados na biblioteca $Cp.hs$.
> Supondo $A = String$, $B = \mathbb{B}$ e $C = \mathbb{Z}$,
> a) aplique no GHCi $\mathrm{undistr}$, alternativamente, aos pares $(\text{"CP"}, \mathrm{True})$ ou $(\text{"LEI"}, 1)$;
> b) verifique que $(\mathrm{distr} \cdot \mathrm{undistr})\ x = x$ 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 Universal-+ (17), que α 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 α e teste-a no GHCi, onde i_1 (resp. i_2) se escreve `Left` (resp. `Right`).

Resolução 5

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

```
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
>
> $$fac \cdot [\underline{0}, \mathrm{succ}] = [\underline{1}, \mathrm{mul} \cdot \langle \mathrm{succ}, fac \rangle]$$
>
> onde
>
> $$\begin{aligned} &\mathrm{succ}\ n = n + 1 \\ &\mathrm{mul}\ (a, b) = a * b \end{aligned}$$

Resolução 6

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

Exercício 7

> A função $\mathrm{in} = [\underline{0}, succ]$ da questão anterior exprime, para $\mathrm{succ}\ n = n + 1$, a forma como os números naturais ($\mathbb{N}_0$) são gerados a partir do número 0, de acordo com o diagrama seguinte:
>
> $$\begin{tikzcd} 1 \arrow[r, "i_1"] \arrow[dr, "\underline{0}"'] & 1 + \mathbb{N}_0 \arrow[d, "\mathsf{in}=[\underline{0},\mathsf{succ}]"] & \mathbb{N}_0 \arrow[l, "i_2"'] \arrow[dl, "\mathsf{succ}"] \\ & \mathbb{N}_0 & \end{tikzcd} \tag{F3}$$
>
> 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 in,
>
> $$\begin{cases} \mathrm{out}\ 0 = i_1\ () \\ \mathrm{out}\ (n+1) = i_2\ n \end{cases} \tag{F4}$$
>
> resolvendo em ordem a out a equação
>
> $$\mathrm{in} \cdot \mathrm{out} = id \tag{F5}$$
>
> e introduzindo variáveis.

Resolução 7

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

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
>
> $$\begin{tikzcd}
\mathbb{N}_0 \arrow[rr, "\mathsf{out}"] \arrow[d, "fac"'] & & 1 + \mathbb{N}_0 \arrow[d, "id + \langle \mathsf{succ}, fac \rangle"] \\
\mathbb{N}_0 & & 1 + \mathbb{N}_0 \times \mathbb{N}_0 \arrow[ll, "{[\underline{1}, \mathsf{mul}]}"]
\end{tikzcd}$$
>
> calcula o factorial da sua entrada, assumindo 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
```

```haskell
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 $mkInd : Bib \times Aux \to Ind$ 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
```

```haskell
-- (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
```

$$
\begin{tikzcd}
Bib \times Aux \arrow[rr, "(\mathrm{sortByKey} \times \mathrm{sortByKey}) \cdot (\mathrm{deconsBib} \times \mathrm{deconsAux})"] \arrow[d, "\mathrm{mkInd}"] & & (Key \times Aut)^* \times (Pag \times Key)^* \arrow[d, "\mathrm{joinAutPag}"] \\
Ind & & (Aut \times Pag)^* \arrow[ll, "\mathrm{groupPagByAut}"']
\end{tikzcd}
$$

$\square$