CP - Ficha 4

Exercício 1

> Considere o isomorfismo
>
> $$\begin{tikzcd} (A+B)+C \arrow[rr, bend left, "\text{coassocr}"] & \cong & A+(B+C) \arrow[ll, bend left, "\text{coassocl}"] \end{tikzcd}$$
>
> onde $\text{coassocr} = [id + i_1, i_2 \cdot i_2]$. Calcule a sua conversa resolvendo em ordem a coassocl a equação,
>
> $$\text{coassocl} \cdot \text{coassocr} = id$$
>
> isto é, a equação
>
> $$\text{coassocl} \cdot [id + i_1, i_2 \cdot i_2] = id$$
>
> Finalmente, exprima coassocl sob a forma de um programa em Haskell *não recorra* ao combinador `either` e teste as duas versões no GHCi.

Resolução 1

$$
\begin{array}{ll}
\text{coassocl} \cdot [id + i_1, i_2 \cdot i_2] = id \\
\equiv & \{ (20\text{: Fusão-+}) \} \\
[\text{coassocl} \cdot (id + i_1), \text{coassocl} \cdot (i_2 \cdot i_2)] = id \\
\equiv & (17\text{: Universal-+, 1: Natural-id}) \\
\begin{cases} \text{coassocl} \cdot (id + i_1) = i_1 \\ \text{coassocl} \cdot i_2 \cdot i_2 = i_2 \end{cases} \\
\equiv & (21\text{: Def-+, 1: Natural-id}) \\
\begin{cases} \text{coassocl} \cdot [i_1, i_2 \cdot i_1] = i_1 \\ \text{coassocl} \cdot i_2 \cdot i_2 = i_2 \end{cases} \\
\equiv & (20\text{: Fusão-+}) \\
\begin{cases} [\text{coassocl} \cdot i_1, \text{coassocl} \cdot i_2 \cdot i_1] = i_1 \\ \text{coassocl} \cdot i_2 \cdot i_2 = i_2 \end{cases} \\
\equiv & (17\text{: Universal-+}) \\
\begin{cases} \text{coassocl} \cdot i_1 = i_1 \cdot i_1 \\ \text{coassocl} \cdot i_2 \cdot i_1 = i_1 \cdot i_2 \\ \text{coassocl} \cdot i_2 \cdot i_2 = i_2 \end{cases} \\
\equiv & (17\text{: Universal-+}) \\
\begin{cases} \text{coassocl} \cdot i_1 = i_1 \cdot i_1 \\ \text{coassocl} \cdot i_2 = [i_1 \cdot i_2, i_2] \end{cases} \\
\equiv & (17\text{: Universal-+}) \\
\text{coassocl} = [i_1 \cdot i_1, [i_1 \cdot i_2, i_2]] \\
\equiv & (21\text{: Def-+, 1: Natural-id}) \\
\text{coassocl} = [i_1 \cdot i_1, i_2 + id]
\end{array}
$$

$$
\begin{cases} \text{coassocl} \cdot i_1 = i_1 \cdot i_1 \\ \text{coassocl} \cdot i_2 \cdot i_1 = i_1 \cdot i_2 \\ \text{coassocl} \cdot i_2 \cdot i_2 = i_2 \end{cases}
$$

```haskell
coassocl :: Either a (Either b c) -> Either (Either a b) c
coassocl (Left x) = Left (Left x)
coassocl (Right (Left x)) = Left (Right x)
coassocl (Right (Right x)) = Right x
```

Exercício 2

> Considere a seguinte declaração de um tipo de *árvores binárias*, em Haskell:
>
> ```haskell
> data LTree a = Leaf a | Fork (LTree a, LTree a)
> ```
>
> Indagando os tipos dos construtores $Leaf$ e $Fork$, por exemplo no GHCi,
>
> ```haskell
> *LTree> :t Fork
> Fork :: (LTree a, LTree a) -> LTree a
> *LTree> :t Leaf
> Leaf :: a -> LTree a
> ```
>
> faz sentido definir a função que mostra como construir árvores deste tipo:
>
> $$\mathrm{in} = [Leaf, Fork] \tag{F1}$$
>
> Desenhe um diagrama para esta função e calcule a sua inversa
>
> $$\begin{aligned} \mathrm{out}\ (Leaf\ a) &= i_1\ a \\ \mathrm{out}\ (Fork\ (x, y)) &= i_2\ (x, y) \end{aligned}$$
>
> de novo resolvendo a equação $\mathrm{out} \cdot \mathrm{in} = id$ em ordem a out, agora para o $(\mathrm{F1})$. Finalmente, faça testes em Haskell que involvam a composição $\mathrm{in} \cdot \mathrm{out}$ e tire conclusões.

Resolução 2

$$\begin{aligned}
& \mathrm{out} \cdot \mathrm{in} = id \\
\equiv & \qquad \{ \text{(F1: Def. in)} \} \\
& \mathrm{out} \cdot [Leaf, Fork] = id \\
\equiv & \qquad \{ \text{(20: Fusão-+)} \} \\
& [\mathrm{out} \cdot Leaf, \mathrm{out} \cdot Fork] = id \\
\equiv & \qquad \{ \text{(17: Universal-+, 1: Natural-id)} \} \\
& \begin{cases} \mathrm{out} \cdot Leaf = i_1 \\ \mathrm{out} \cdot Fork = i_2 \end{cases} \\
\equiv & \qquad \{ \text{(72: Ig. Ext., 73: Def-comp)} \} \\
& \begin{cases} \mathrm{out}\ (Leaf\ a) = i_1\ a \\ \mathrm{out}\ (Fork\ (x, y)) = i_2\ (x, y) \end{cases}
\end{aligned}$$

Exercício 3

> Deduza o tipo mais geral da função $\alpha = (id + \pi_1) \cdot i_2 \cdot \pi_2$ e represente-o num diagrama.

Resolução 3

$$C + B \xleftarrow{id + \pi_1} C + (B \times D) \xleftarrow{i_2} B \times D \xleftarrow{\pi_2} A \times (B \times D)$$

$$\alpha : A \times (B \times D) \rightarrow C + B$$

Exercício 4

> Considere a função
>
> $$\alpha = \text{swap} \cdot (id \times \text{swap}) \tag{F2}$$
>
> Calcule o tipo mais geral de α e formule a sua propriedade natural (grátis) a inferir através de um diagrama, como se explicou na aula teórica.

Resolução 4

$$\begin{aligned} \text{swap} &: A \times B \rightarrow B \times A \\ (id \times \text{swap}) &: A \times (B \times C) \rightarrow A \times (C \times B) \\ \alpha &: A \times (B \times C) \rightarrow (C \times B) \times A \end{aligned}$$

$$\begin{tikzcd} (A \times B) \times C \arrow[r, "\alpha"] \arrow[d, "f \times (g \times h)"] & (C \times B) \times A \arrow[d, "(h \times g) \times f"] \\ A' \times (B' \times C') \arrow[r, "\alpha"] & (C' \times B') \times A' \end{tikzcd}$$

$$\alpha \cdot (f \times (g \times h)) = ((h \times g) \times f) \cdot \alpha$$

Exercício 5

> Considere as funções elementares que respetivamente juntam ou duplicam informação:
>
> $$join = [id, id] \tag{F3}$$
> $$dup = \langle id, id \rangle \tag{F4}$$
>
> Calcule (justificando) a propriedade grátis da função $\alpha = dup \cdot join$ e indique por que razão não pode calcular essa propriedade para $join \cdot dup$.

Resolução 5

$$join : A + A \to A$$
$$dup : B \to B \times B$$

Para a composição ser possível, $B = A$, logo:
$dup : A \to A \times A$
$dup \cdot join : A + A \to A \times A$

Propriedade grátis de $\alpha = dup \cdot join$

$$\begin{tikzcd}
A + A \arrow[r, "\alpha"] \arrow[d, "f + f"] & A \times A \arrow[d, "f \times f"] \\
A' + A' \arrow[r, "\alpha"] & A' \times A'
\end{tikzcd}$$

$$\alpha \cdot (f + f) = (f \times f) \cdot \alpha$$

Não podemos calcular a propriedade natural para $join \cdot dup$
pois o codomínio de $join$ é incompátivel com o domínio de dup.

Exercício 6

> Seja dada uma função ∇ da qual só sabe duas propriedades: $\nabla \cdot i_1 = id$ e $\nabla \cdot i_2 = id$. Mostre que, necessariamente, ∇ satisfaz também a propriedade natural
>
> $$f \cdot \nabla = \nabla \cdot (f + f) \tag{F5}$$

Resolução 6

$$\begin{cases} \nabla \cdot i_1 = id \\ \nabla \cdot i_2 = id \end{cases} \equiv \quad \nabla = [id, id] \qquad \text{(17: Universal-+)}$$

$$\begin{aligned}
& f \cdot \nabla = \nabla \cdot (f + f) \\
\equiv & \qquad \text{(Def. } \nabla\text{)} \\
& f \cdot \nabla = [id, id] \cdot (f + f) \\
\equiv & \qquad \text{(22: Absorção-+)} \\
& f \cdot \nabla = [id \cdot f, id \cdot f] \\
\equiv & \qquad \text{(1: Natural-id)} \\
& f \cdot \nabla = [f \cdot id, f \cdot id] \\
\equiv & \qquad \text{(20: Fusão-+)} \\
& f \cdot \nabla = f \cdot [id, id] \\
\equiv & \qquad \text{(Def. } \nabla\text{)} \\
& f \cdot \nabla = f \cdot \nabla \qquad \text{c.q.m.}
\end{aligned}$$

Exercício 7

> Seja dada uma função α cuja propriedade grátis é:
>
> $$(f + h) \cdot \alpha = \alpha \cdot (f + g \times h) \tag{F6}$$
>
> Será esta propriedade suficiente para deduzir a definição de α? Justifique analiticamente.

Resolução 7

$$(f + h) \cdot \alpha = \alpha \cdot (f + g \times h)$$

$$\begin{tikzcd}
A + (B \times C) \arrow[r, "\alpha"] \arrow[d, "f + g \times h"] & A + C \arrow[d, "f + h"] \\
A' + (B' \times C') \arrow[r, "\alpha"] & A' + C'
\end{tikzcd}$$

$$\alpha : A + (B \times C) \to A + C$$
$$\text{Deduz-se que: } \alpha = id + \pi_2$$

Prova analítica:

$$\begin{aligned}
& (f + h) \cdot \alpha = \alpha \cdot (f + g \times h) \\
\equiv \quad & \{\text{Dedução da def. } \alpha\} \\
& (f + h) \cdot (id + \pi_2) = (id + \pi_2) \cdot (f + g \times h) \\
\equiv \quad & \{\text{25: Functor-+}\} \\
& (f \cdot id) + (h \cdot \pi_2) = (id \cdot f) + (\pi_2 \cdot (g \times h)) \\
\equiv \quad & \{\text{1: Natural-id, 13: Natural-}\pi_2\} \\
& f + (h \cdot \pi_2) = f + (h \cdot \pi_2) \qquad \text{c.q.d.}
\end{aligned}$$

Exercício 8

> O formulário inclui as duas equivalências seguintes, válidas para qualquer isomorfismo α:
>
> $$\alpha \cdot g = h \equiv g = \alpha^\circ \cdot h \tag{F7}$$
> $$g \cdot \alpha = h \equiv g = h \cdot \alpha^\circ \tag{F8}$$
>
> Recorra a essas propriedades para mostrar que a igualdade
>
> $$h \cdot \mathrm{distr} \cdot (g \times (id + \alpha)) = k$$
>
> é equivalente à igualdade
>
> $$h \cdot (g \times id + g \times \alpha) = k \cdot \mathrm{undistr}$$
>
> (**Sugestão**: não ignore a propriedade natural (i.e. grátis) do isomorfismo distr.)

Resolução 8

$$\mathrm{distr}^\circ = \mathrm{undistr}$$

$$\mathrm{distr} : A \times (B + C) \to (A \times B) + (A \times C)$$

$$\begin{array}{ccc}
A \times (B + C) & \xrightarrow{\quad \mathrm{distr} \quad} & (A \times B) + (A \times C) \\
\Big| & & \Big| \\
f \times (g + h) & & (f \times g) + (f \times h) \\
\Big\downarrow & & \Big\downarrow \\
A' \times (B' + C') & \xrightarrow{\quad \mathrm{distr} \quad} & (A' \times B') + (A' \times C')
\end{array}$$

$$\text{Propriedade natural de distr:}$$
$$((f \times g) + (f \times h)) \cdot \mathrm{distr} = \mathrm{distr} \cdot (f \times (g + h))$$

$$\begin{array}{ll}
h \cdot \mathrm{distr} \cdot (g \times (id + \alpha)) = k & \\
\equiv & \text{(Prop. grátis de distr)} \\
h \cdot ((g \times id) + (g \times \alpha)) \cdot \mathrm{distr} = k & \\
\equiv & \text{(F8 [a.k.a.] 33: Shunt-left)} \\
h \cdot ((g \times id) + (g \times \alpha)) = k \cdot \mathrm{undistr} &
\end{array}$$

c.q.m.

Exercício 9

> A *lei da troca* (identifique-a no formulário) permite-nos exprimir determinadas funções sob duas formas alternativas, conforme desenhado no respectivo diagrama:
>
> $$[\langle f, g\rangle, \langle h, k\rangle] = \langle [f, h], [g, k]\rangle \tag{F9}$$
>
> $$\begin{tikzcd} A \arrow[r, "i_1"] \arrow[d, "f"'] \arrow[drr, "h" near start] & A + B & B \arrow[l, "i_2"'] \arrow[d, "k"] \arrow[dll, "g"' near end] \\ C & C \times D \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] & D \end{tikzcd}$$
>
> Demonstre esta lei recorrendo às propriedades (e.g. universais) dos produtos e dos coprodutos.

Resolução 9

$$\begin{array}{l}
[\langle f, g\rangle, \langle h, k\rangle] = \langle [f, h], [g, k]\rangle \\
= \qquad\qquad\qquad\qquad (\text{6: Universal-}\times) \\
\begin{cases} [f, h] = \pi_1 \cdot [\langle f, g\rangle, \langle h, k\rangle] \\ [g, k] = \pi_2 \cdot [\langle f, g\rangle, \langle h, k\rangle] \end{cases} \\
= \qquad\qquad\qquad\qquad (\text{20: Fusão-}+\text{, 7: Cancelamento-}\times) \\
\begin{cases} [f, h] = [f, h] \\ [g, k] = [g, k] \end{cases} \quad \text{c.q.d.}
\end{array}$$

Exercício 10

> **Questão prática** ***Problem requirements:***
>
> *Well-known services such as Google Maps, Google Analytics, YouTube, MapReduce etc. run on top of Bigtable or successors thereof. Such data systems rely on the so-called key-value* NoSQL *data model, which is widely adopted because of its efficiency and flexibility.*
>
> *Key-value stores can be regarded abstractly as lists of pairs* $(K \times V)^*$ *in which* K *is a datatype of* keys *and* V *is a type of data* values. *Keys uniquely identify values. Key-value stores with the same type* V *of values can be glued together as the diagram suggests,*
>
> $$\begin{tikzcd} ((K + K') \times V)^* \arrow[rr, bend left, "unglue"] & & (K \times V)^* \times (K' \times V)^* \arrow[ll, bend left, "glue"] \end{tikzcd}$$
>
> *where* $unglue$ *performs the action opposite to* $glue$.
>
> *Define* $glue$ *and* $unglue$ *in Haskell structured along the functional combinators (*$f \cdot g$, $\langle f, g \rangle$, $f \times g$*, and so on) studied in this course and available from library Cp.hs. Use* ***diagrams*** *to plan your solutions, in which you should avoid re-inventing functions over lists already available from the Haskell standard libraries.*

Resolução 10

```haskell
import Cp (split, p1, p2, i1, i2, (><), singl, nil, distl)

glue :: ([(k, v)], [(k, v)]) -> [(Either k k, v)]
glue = uncurry (++) . (gluer >< gluel)
  where
    gluer = map (i1 >< id)
    gluel = map (i2 >< id)

unglue :: [(Either k k, v)] -> ([(k, v)], [(k, v)])
unglue = split (concatMap p1) (concatMap p2)
         . map (either (split singl nil) (split nil singl) . distl)

-- test
glued = ([(1, "one"), (3, "three")], [(2, "two"), (4, "four")])
unglued = [(Left 1, "one"), (Left 3, "three"), (Right 2, "two"), (Right 4, "four")]
test = print $ unglue unglued == glued && glue glued == unglued
```

$$\begin{tikzcd}
(K \times V)^* \times (K' \times V)^* \arrow[d, "\mathrm{gluer} \times \mathrm{gluel}"] \arrow[dd, bend right=60, swap, "\mathrm{glue}"] \\
(K + K', V)^* \times (K + K', V)^* \arrow[d, "\mathrm{uncurry}\ (++)"] \\
(K + K', V)^*
\end{tikzcd}$$

$$\text{where} \begin{cases} \mathrm{gluer} = \mathrm{map}\ (i_1 \times id) \\ \mathrm{gluel} = \mathrm{map}\ (i_2 \times id) \end{cases}$$

Diagrama da função glue

$$\begin{tikzcd}
(K + K', V)^* \arrow[d, "\mathrm{map}\ {[\langle \mathrm{singl},\ \mathrm{nil} \rangle, \langle \mathrm{nil},\ \mathrm{singl} \rangle]} \cdot \mathrm{distl}"] \arrow[dd, bend right=60, swap, "\mathrm{unglue}"] \\
(K \times V)^* \times (K' \times V)^* \arrow[d, "\langle \mathrm{concatMap}\ \pi_1,\ \mathrm{concatMap}\ \pi_2 \rangle"] \\
(K \times V)^* \times (K' \times V)^*
\end{tikzcd}$$

Diagrama da função unglue

$\square$