CP - Ficha 4

Exercício 1

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

Resolução 1

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

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

```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:

```
data LTree a = Leaf a | Fork (LTree a, LTree a)
```

Indagando os tipos dos construtores $Leaf$ e $Fork$, por exemplo no GHCi,

```
*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:

$$\text{in} = [Leaf, Fork] \tag{F1}$$

Desenhe um diagrama para esta função e calcule a sua inversa

$$\begin{aligned}&\text{out}\ (Leaf\ a) = i_1\ a\\&\text{out}\ (Fork\ (x,y)) = i_2\ (x,y)\end{aligned}$$

de novo resolvendo a equação $\text{out} \cdot \text{in} = id$ em ordem a out, agora para o (F1). Finalmente, faça testes em Haskell que involvam a composição $\text{in} \cdot \text{out}$ e tire conclusões.

Resolução 2

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

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

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

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

Exercício 4

> Considere a função
>
> $$\alpha = \mathrm{swap} \cdot (id \times \mathrm{swap}) \tag{F2}$$
>
> 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

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

$$\begin{array}{ccc} (A \times B) \times C & \xrightarrow{\alpha} & (C \times B) \times A \\ \downarrow & & \downarrow \\ f \times (g \times h) & & (h \times g) \times f \\ \downarrow & & \downarrow \\ A' \times (B' \times C') & \xrightarrow{\alpha} & (C' \times B') \times A' \end{array}$$

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

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 $\alpha = dup \cdot join$

$$\begin{array}{ccc} A + A & \xrightarrow{\alpha} & A \times A \\ \downarrow{\scriptstyle f + f} & & \downarrow{\scriptstyle f \times f} \\ A' + A' & \xrightarrow{\alpha} & A' \times A' \end{array}$$

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

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
>
> $$f \cdot \nabla = \nabla \cdot (f + f) \qquad \text{(F5)}$$

Resolução 6

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

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

Exercício 7

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

Resolução 7

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

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

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

$$\text{Prova analítica:}$$

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

Exercício 8

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

Resolução 8

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

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

$$\begin{array}{ccc} A \times (B + C) & \xrightarrow{\text{distr}} & (A \times B) + (A \times C) \\ \downarrow{\scriptstyle f \times (g + h)} & & \downarrow{\scriptstyle (f \times g) + (f \times h)} \\ A' \times (B' + C') & \xrightarrow{\text{distr}} & (A' \times B') + (A' \times C') \end{array}$$

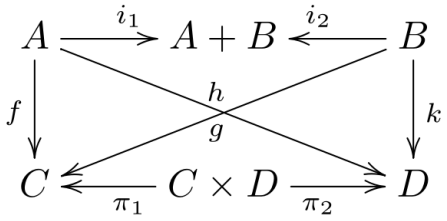
Propriedade natural de distr:

$$((f \times g) + (f \times h)) \cdot \text{distr} = \text{distr} \cdot (f \times (g + h))$$

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

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:
>
> $$[\langle f, g\rangle, \langle h, k\rangle] = \langle [f, h], [g, k]\rangle \tag{F9}$$
>
> 
>
>
> Demonstre esta lei recorrendo às propriedades (e.g. universais) dos produtos e dos coprodutos.

Resolução 9

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

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,*

$$((K + K') \times V)^* \underset{glue}{\overset{unglue}{\rightleftarrows}} (K \times V)^* \times (K' \times V)^*$$

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.*

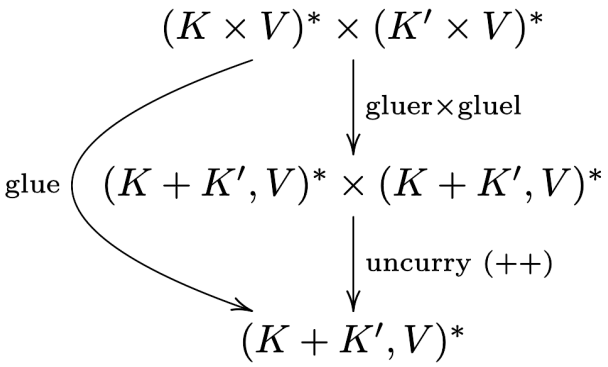
Resolução 10

```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
```



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

Diagrama da função glue

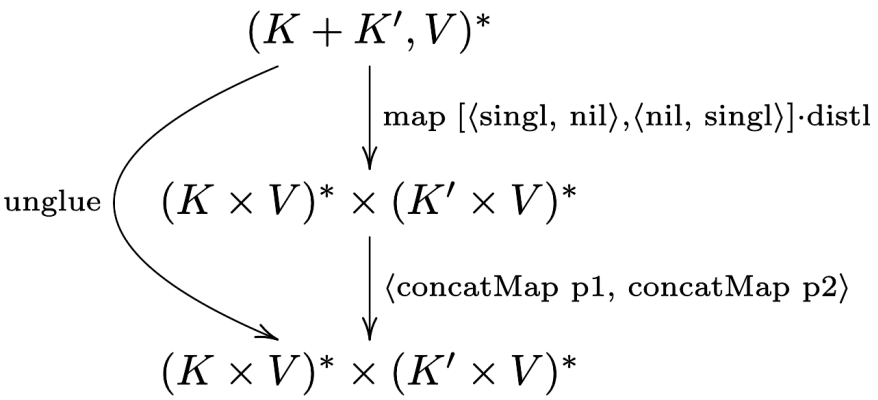


Diagrama da função unglue

□