

CP - Ficha 4

Exercício 1

Considere o isomorfismo

$$\begin{array}{ccc} & \xrightarrow{\text{coassocr}} & \\ (A + B) + C & \cong & A + (B + C) \\ & \xleftarrow{\text{coassocl}} & \end{array}$$

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

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

TODO Haskell definition without using `either`

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:

$$\text{in} = [\text{Leaf}, \text{Fork}] \quad (\text{F1})$$

Desenhe um diagrama para esta função e calcule a sua inversa

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

de novo resolvendo a equação $\text{out} \cdot \text{in} = \text{id}$ em ordem a out, agora para o (F1).

Finalmente, faça testes em Haskell que envolvam a composição $\text{in} \cdot \text{out}$ e tire conclusões.

Resolução 2

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

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

$$\alpha = \text{swap} \cdot (id \times \text{swap}) \quad (\text{F2})$$

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

$$\begin{aligned} \text{swap} &: A \times B \rightarrow B \times A \\ (id \times \text{swap}) &: A \times (B \times C) \rightarrow A \times (C \times B) \\ \alpha &: A \times (B \times C) \rightarrow (C \times B) \times A \end{aligned}$$
$$\begin{array}{ccc} (A \times B) \times C & \xrightarrow{\alpha} & (C \times B) \times A \\ \downarrow f \times (g \times h) & & \downarrow (h \times g) \times f \\ A' \times (B' \times C') & \xrightarrow{\alpha} & (C' \times B') \times A' \end{array}$$
$$\alpha \cdot (f \times (g \times h)) = ((h \times g) \times f) \cdot \alpha$$

Exercício 5

Considere as funções elementares que respetivamente juntam ou duplicam informação:

$$join = [id, id] \quad (F3)$$

$$dup = \langle id, id \rangle \quad (F4)$$

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

$$join : A + A \rightarrow A$$

$$dup : B \rightarrow B \times B$$

Para a composição ser possível, $B = A$, logo:

$$dup : A \rightarrow A \times A$$

$$dup \cdot join : A + A \rightarrow A \times A$$

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

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

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

Não podemos calcular a propriedade natural para $join \cdot dup$ pois o codomínio de $join$ é incompatível 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

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

Resolução 6

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

$$f \cdot \nabla = \nabla \cdot (f + f)$$

$$\equiv$$

(Def. ∇)

$$f \cdot \nabla = [id, id] \cdot (f + f)$$

$$\equiv$$

(22: Absorção-+)

$$f \cdot \nabla = [id \cdot f, id \cdot f]$$

$$\equiv$$

(1: Natural-id)

$$f \cdot \nabla = [f \cdot id, f \cdot id]$$

$$\equiv$$

(20: Fusão-+)

$$f \cdot \nabla = f \cdot [id, id]$$

$$\equiv$$

(Def. ∇)

$$f \cdot \nabla = f \cdot \nabla \quad \text{c.q.m.}$$

Exercício 7

Seja dada uma função α cuja propriedade gráti é:

$$(f + h) \cdot \alpha = \alpha \cdot (f + g \times h) \quad (\text{F6})$$

Será esta propriedade suficiente para deduzir a definição de α ? Justifique analiticamente.

Resolução 7

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

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

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

$$\text{Deduz-se que: } \alpha = id + \pi_2$$

Prova analítica:

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

$$\equiv \quad (\text{Dedução da def. } \alpha)$$

$$(f + h) \cdot (id + \pi_2) = (id + \pi_2) \cdot (f + g \times h)$$

$$\equiv \quad (25: \text{Functor-}+)$$

$$(f \cdot id) + (h \cdot \pi_2) = (id \cdot f) + (\pi_2 \cdot (g \times h))$$

$$\equiv \quad (1: \text{Natural-id, } 13: \text{Natural-}\pi_2)$$

$$f + (h \cdot \pi_2) = f + (h \cdot \pi_2) \quad \text{c.q.d.}$$

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

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

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

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

Propriedade natural de distr :

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

$$h \cdot \text{distr} \cdot (g \times (id + \alpha)) = k$$

\equiv

(Prop. grátis de distr)

$$h \cdot ((g \times id) + (g \times \alpha)) \cdot \text{distr} = k$$

\equiv

(F8 [a.k.a.] 33: Shunt-left)

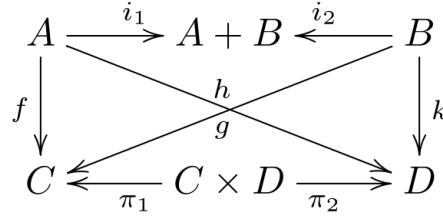
$$h \cdot ((g \times id) + (g \times \alpha)) = k \cdot \text{undistr}$$

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:

$$[\langle f, g \rangle, \langle h, k \rangle] = \langle [f, h], [g, k] \rangle \quad (\text{F9})$$



Demonstre esta lei recorrendo às propriedades (e.g. universais) dos produtos e dos coprodutos.

Resolução 9

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

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,

$$\begin{array}{ccc} & \xrightarrow{\text{unglue}} & \\ ((K + K') \times V)^* & & (K \times V)^* \times (K' \times V)^* \\ & \xleftarrow{\text{glue}} & \end{array}$$

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

TODO