

Cálculo de Programas

Algebra of Programming

Lic. Ciências da Computação (2º ano)
Lic./Mest.Int. em Engenharia Informática (3º ano)
UNIVERSIDADE DO MINHO

2024/25 - Ficha nr.º 4

1. Considere o isomorfismo

Consider the isomorphism

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

onde $\text{coassocr} = [id + i_1, i_2 \cdot i_2]$. Calcule a sua conversa resolvendo em ordem a coassocl a equação,

where $\text{coassocr} = [id + i_1, i_2 \cdot i_2]$. Find its converse coassocl by solving the equation,

$$\text{coassocl} \cdot \text{coassocr} = id$$

isto é, a equação

that is, the equation

$$\text{coassocl} \cdot \underbrace{[id + i_1, i_2 \cdot i_2]}_{\text{coassocr}} = id$$

Finalmente, exprima coassocl sob a forma de um programa em Haskell *não recorra* ao combinador “either” e teste as duas versões no GHCi.
..

Finally express coassocl in pointwise Haskell code not using the “either” combinator a test both versions on GHCi

2. Considere a seguinte declaração de um tipo de *árvores binárias*, em Haskell:

Consider the following definition in Haskell of a particular type of binary tree:

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

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

*By querying the types of constructors *Leaf* and *Fork* in GHCi, for example,*

```
*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:

one can define

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

Desenhe um diagrama para esta função e calcule a sua inversa

capturing how data of this type are built. Draw a diagram for this function and find its inverse,

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

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.

again solving the equation $\text{out} \cdot \text{in} = \text{id}$ for out, but now with respect to (F1). Finally, run tests in Haskell involving the composition $\text{in} \cdot \text{out}$ and draw conclusions.

3. Deduza o tipo mais geral da função $\alpha = (\text{id} + \pi_1) \cdot i_2 \cdot \pi_2$ e represente-o através de um diagrama.

Infer the most general type of function $\alpha = (\text{id} + \pi_1) \cdot i_2 \cdot \pi_2$ and draw it in a diagram of compositions.

4. Considere a função

Let

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

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.

be given. Infer the most general type of α and the associated natural (“free”) property using a diagram, as shown in the theory class.

5. Considere as seguintes funções elementares que respectivamente juntam ou duplicam informação:

Let the following basic functions be given that, respectively, gather or duplicate information:

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

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

Calcule (justificando) a propriedade grátis da função $\alpha = \text{dup} \cdot \text{join}$ e indique por que razão não pode calcular essa propriedade para $\text{join} \cdot \text{dup}$.

Calculate (justifying) the free property of the function $\alpha = \text{dup} \cdot \text{join}$ and indicate why you cannot calculate this property for $\text{join} \cdot \text{dup}$.

6. Seja dada uma função ∇ da qual só sabe duas propriedades: $\nabla \cdot i_1 = \text{id}$ e $\nabla \cdot i_2 = \text{id}$. Mostre que, necessariamente, ∇ satisfaz também a propriedade natural

Suppose that, about a function ∇ , you only know two properties: $\nabla \cdot i_1 = \text{id}$ and $\nabla \cdot i_2 = \text{id}$. Show that, necessarily, ∇ also satisfies the natural property

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

7. Seja dada uma função α cuja propriedade grátis é:

Let α be a function with free property:

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

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

Can a definition of α be inferred from (F6)? Justify.

8. O formulário inclui as duas equivalências seguintes, válidas para qualquer isomorfismo α :

Any isomorphism α satisfies the following equivalences (also given in the reference sheet),

$$\alpha \cdot g = h \equiv g = \alpha^\circ \cdot h \quad (\text{F7})$$

$$g \cdot \alpha = h \equiv g = h \cdot \alpha^\circ \quad (\text{F8})$$

Recorra a essas propriedades para mostrar que a igualdade

which can be useful to show that the equality

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

é equivalente à igualdade

is equivalent to:

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

(Sugestão: não ignore a propriedade natural (i.e. grátis) do isomorfismo distr .)

Prove this equivalence. (**Hint:** the free-property of distr shouldn't be ignored in the reasoning.)

9. A lei da troca (identifique-a no formulário) permite-nos exprimir determinadas funções sob duas formas alternativas, conforme desenhado no respectivo diagrama:

The exchange law (check this law in the reference sheet) allows one to express certain functions in two alternative forms, as given below:

$$[\langle f, g \rangle, \langle h, k \rangle] = \langle [f, h], [g, k] \rangle$$

$$\begin{array}{ccccc} A & \xrightarrow{i_1} & A + B & \xleftarrow{i_2} & B \\ f \downarrow & & \downarrow h & & \downarrow k \\ C & \xleftarrow{\pi_1} & C \times D & \xrightarrow{\pi_2} & D \end{array} \quad (\text{F9})$$

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

Prove this law using the (e.g. universal) properties of products and co-products.

10. **Questão prática** — Este problema não irá ser abordado em sala de aula. Os alunos devem tentar resolvê-lo em casa e, querendo, publicar a sua solução no canal **#geral** do Slack, com vista à sua discussão com colegas.

Open assignment — This assignment will not be addressed in class. Students should try to solve it at home and, wishing so, publish their solutions in the **#geral** Slack channel, so as to trigger discussion among other colleagues.

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.

□