Cálculo de Programas
Algebra of Programming

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

2023/24 - Ficha nr.° 2

1. Recorde a propriedade universal

   Recall the universal property

$$k = \langle f, g \rangle \quad \equiv \quad \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases}$$

   da qual, como se disse na aula teórica, podem ser derivadas todas as outras que aparecem no respectivo grupo, no formulário. Use-a para demonstrar a lei

   from which all the others that appear in the respective group (in the reference sheet) can be derived. Use it to prove the law

$$\langle h, k \rangle \cdot f = \langle h \cdot f, k \cdot f \rangle$$

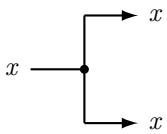
   que também consta desse formulário sob a designação *fusão-*$\times$.

   that also appears in the reference sheet under the name $\times$-fusion.

---

2. Uma das operações essenciais em processamento da informação é a sua *duplicação*:

   A core operation in information processing is data duplication*:*



$$dup\ x = (x, x)$$

   Recorra à lei de fusão-$\times$ para demonstrar a seguinte propriedade da duplicação de informação:

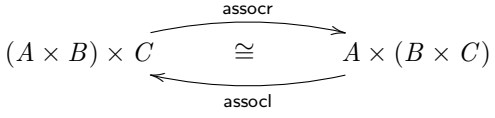
   Derive from the $\times$*-fusion law the following property of data duplication:*

$$dup \cdot f = \langle f, f \rangle$$

---

3. Considere o diagrama

   Consider the following diagram



onde $\mathsf{assocl} = \langle id \times \pi_1, \pi_2 \cdot \pi_2 \rangle$. Apresente justificações para o cálculo que se segue em que se resolve em ordem a assocr a equação $\mathsf{assocl} \cdot \mathsf{assocr} = id$:

where $\mathsf{assocl} = \langle id \times \pi_1, \pi_2 \cdot \pi_2 \rangle$. *The reasoning below solves the equation* $\mathsf{assocl} \cdot \mathsf{assocr} = id$ *for variable* assocr. *Fill in justifications for each step in the reasoning:*

$$\mathsf{assocl} \cdot \mathsf{assocr} = id$$

$$\equiv \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \}$$

$$\left\{ \begin{array}{l} (id \times \pi_1) \cdot \mathsf{assocr} = \pi_1 \\ \pi_2 \cdot \pi_2 \cdot \mathsf{assocr} = \pi_2 \end{array} \right.$$

$$\equiv \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \}$$

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} \pi_1 \cdot \mathsf{assocr} = \pi_1 \cdot \pi_1 \\ \pi_1 \cdot \pi_2 \cdot \mathsf{assocr} = \pi_2 \cdot \pi_1 \end{array} \right. \\ \pi_2 \cdot \pi_2 \cdot \mathsf{assocr} = \pi_2 \end{array} \right.$$

$$\equiv \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \}$$

$$\left\{ \begin{array}{l} \pi_1 \cdot \mathsf{assocr} = \pi_1 \cdot \pi_1 \\ \left\{ \begin{array}{l} \pi_1 \cdot \pi_2 \cdot \mathsf{assocr} = \pi_2 \cdot \pi_1 \\ \pi_2 \cdot \pi_2 \cdot \mathsf{assocr} = \pi_2 \end{array} \right. \end{array} \right.$$

$$\equiv \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \}$$

$$\left\{ \begin{array}{l} \pi_1 \cdot \mathsf{assocr} = \pi_1 \cdot \pi_1 \\ \pi_2 \cdot \mathsf{assocr} = \langle \pi_2 \cdot \pi_1, \pi_2 \rangle \end{array} \right.$$

$$\equiv \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \}$$

$$\mathsf{assocr} = \langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle$$

Finalmente, converta a definição de assocr para notação Haskell *pointwise* que não recorra a nenhum combinador nem projecção.

Finally, convert the definition of assocr *to* pointfree *Haskell so that no combinator or projection is used.*

---

4. Sabendo que uma dada função xr satisfaz a propriedade

Knowing that a given function xr *satisfies the property*

$$\mathsf{xr} \cdot \langle \langle f, g \rangle, h \rangle = \langle \langle f, h \rangle, g \rangle \tag{F1}$$

para todo o f, g e h, derivar de (F1) a definição de xr:

for all f, g *and* h, *derive from (F1) the definition of* xr*:*

$$\mathsf{xr} = \langle \pi_1 \times id, \pi_2 \cdot \pi_1 \rangle \tag{F2}$$

---

5. O combinador

The following combinator

$$\mathsf{const} :: a \rightarrow b \rightarrow a$$
$$\mathsf{const}\ a\ b = a$$

está disponível em Haskell para construir funções constantes, sendo habitual designarmos $\mathsf{const}\ k$ por $\underline{k}$. Demonstre a igualdade

is available in Haskell to build constant functions. (One usually abbreviates $\mathsf{const}\ k$ *to* $\underline{k}$*.) Prove the equality*

$$\underline{(b, a)} = \langle \underline{b}, \underline{a} \rangle \tag{F3}$$

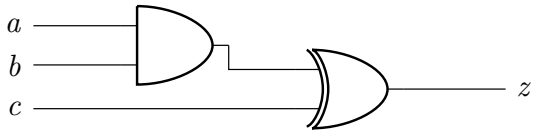
a partir da propriedade universal do produto e das propriedades das funções constantes que constam do formulário.

based on the $\times$-universal property and the properties of constant functions that appear in the reference sheet.

---

6. Considere o circuito booleano

The following Boolean circuit



descreve a função

describes the function

$$f\ ((a, b), c) = (a \wedge b) \oplus c$$

onde $\oplus$ é a operação "exclusive-or"

where $\oplus$ is the "exclusive-or" operation.

(a) Escreva uma definição de

(a) Write a definition of

$$(\mathbb{B} \times \mathbb{B}) \times \mathbb{B} \xrightarrow{f} \mathbb{B}$$

que não recorra às variáveis a, b ou c:

that does not use the variables a, b or c:

(b) Qual é o tipo da função $g = \langle \pi_1, f \rangle$?

(b) What is the type of function $g = \langle \pi_1, f \rangle$?

---

7. **Questão prática** — Este problema não irá ser abordado em sala de aula. Os alunos devem tentar resolvê-lo em casa e, querendo, publicarem a sua solução no canal **#geral** do Slack, com vista à sua discussão com colegas.
Os requisitos do problema são dados abaixo.
NB: usa-se a notação X^* para designar o tipo $[X]$ em Haskell.

***Open assignment** — This assignment will not be addressed in class. Students should try to solve it at home and, if they wish, publish their solution in the **#geral** Slack channel, so that it can be discussed among colleagues.*
The requirements of the problem are given below.
***NB**: notation X^* is used to denote the type $[X]$ in 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 $mkInd : Bib \times Aux \rightarrow Ind$ 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 avilable from library Cp.hs. Solutions should avoid re-inventing functions over lists already available in the Haskell standard libraries.*
□