Cálculo de Programas

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

2025/26 - Ficha nr.° 3

1. Considere o diagrama

$$\begin{tikzcd} (A \times B) \times C \arrow[r, bend left, "\mathsf{assocr}"] \arrow[r, phantom, "\cong"] & A \times (B \times C) \arrow[l, bend left, "\mathsf{assocl}"] \end{tikzcd}$$

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

$$
\begin{array}{ll}
& \mathsf{assocl} \cdot \mathsf{assocr} = id \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \left\{ \begin{array}{l} (id \times \pi_1) \cdot \mathsf{assocr} = \pi_1 \\ \pi_2 \cdot \pi_2 \cdot \mathsf{assocr} = \pi_2 \end{array} \right. \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \left\{ \begin{array}{l} \left\{ \begin{array}{l} \pi_1 \cdot \mathsf{assocr} = \pi_1 \cdot \pi_1 \\ \pi_1 \cdot \pi_2 \cdot \mathsf{assocr} = \pi_2 \cdot \pi_1 \end{array} \right. \\ \pi_2 \cdot \pi_2 \cdot \mathsf{assocr} = \pi_2 \end{array} \right. \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \left\{ \begin{array}{l} \pi_1 \cdot \mathsf{assocr} = \pi_1 \cdot \pi_1 \\ \left\{ \begin{array}{l} \pi_1 \cdot \pi_2 \cdot \mathsf{assocr} = \pi_2 \cdot \pi_1 \\ \pi_2 \cdot \pi_2 \cdot \mathsf{assocr} = \pi_2 \end{array} \right. \end{array} \right. \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \left\{ \begin{array}{l} \pi_1 \cdot \mathsf{assocr} = \pi_1 \cdot \pi_1 \\ \pi_2 \cdot \mathsf{assocr} = \langle \pi_2 \cdot \pi_1, \pi_2 \rangle \end{array} \right. \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{assocr} = \langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle
\end{array}
\tag{F1}
$$

---

2. (a) Codifique (F1) directamente em Haskell e verifique o comportamemto dessa função no GHCi; (b) De seguida, converta — por igualdade extensional — (F1) para notação Haskell *pointwise* que não recorra a nenhum combinador nem projecção e verifique no GHCi que as duas versões dão os mesmos resultados.

---

3. Considere a função

$$\beta = \mathsf{swap} \cdot (id \times \mathsf{swap}) \tag{F2}$$

Determine o tipo mais geral de β e mostre que a seguinte propriedade se verifica:

$$\beta \cdot (f \times (g \times h)) = ((h \times g) \times f) \cdot \beta \tag{F3}$$

NB: sugere-se a utilização de uma propriedade de swap que foi demonstrada na ficha anterior. .

---

4. Recorde da ficha anterior a noção de função constante $\underline{k}$, que é tal que

$\underline{k}\ x = k$

qualquer que seja x. Consulte as propriedades dessas funções no formlário e demonstre, de entre estas, a lei **Eq-const**.

---

5. Defina no GHCi o seguinte tipo de dados:

```
data X = B Bool | P (Bool,Int)
```

Peça ao GHCi informação sobre os tipos de B e de P e deduza que são funções tais que $f = [B\,, P]$ faz sentido. Qual é o tipo de f? NB: em Haskell a alternativa $[f\,, g]$ escreve-se `either f g`.

---

6. Determine o tipo mais geral da função α que se segue:

$$\alpha = [\langle \underline{\textsc{False}}, id \rangle\,, \langle \underline{\textsc{True}}, id \rangle] \tag{F4}$$

---

7. Procure no formulário a propriedade universal do combinador $[f\,, g]$:

$$k = [f\,, g] \quad \equiv \quad \left\{ \begin{array}{l} k \cdot i_1 = f \\ k \cdot i_2 = g \end{array} \right.$$

Usando essa propriedade, mostre que α acima (F4) se pode escrever em Haskell da forma seguinte:

$$\left\{ \begin{array}{l} \alpha\ (i_1\ a) = (\textsc{False}, a) \\ \alpha\ (i_2\ a) = (\textsc{True}, a) \end{array} \right.$$

Codifique α e teste-a no GHCi, onde i_1 (resp. i_2) se escreve `Left` (resp. `Right`).

---

8. Considere a função:

$$\mathsf{xr} = \langle \pi_1 \times id, \pi_2 \cdot \pi_1 \rangle \tag{F5}$$

(a) Determine o tipo de xr; (b) Mostre que xr satisfaz a propriedade

$$\mathsf{xr} \cdot \langle \langle f, g \rangle, h \rangle = \langle \langle f, h \rangle, g \rangle \tag{F6}$$

para todo o f, g e h.

---

9. **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.

   ***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 $mkInd : Bib \times Aux \to Ind$ 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 that can be found in library Cp.hs. Use **diagrams** to plan your proposed solution, which should avoid re-inventing functions over lists already available in the Haskell standard libraries.*
   $\square$