Cálculo de Programas

2.° ano das Licenciaturas em
Engenharia Informática e Ciências da Computação da
Universidade do Minho

2009/10 - Ficha nr.° 2

1. Apresente justificações para cada um dos passos dados no cálculo seguinte da propriedade **Functor-id-**$+$:

$$
\begin{array}{ll}
& id = id + id \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& id = [i_1 \cdot id \,, i_2 \cdot id] \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& id = [i_1 \,, i_2] \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& id = id \\
\equiv & \quad \{ \text{ igualdade é reflexiva } \} \\
& true
\end{array}
$$

2. Considere a função:

$$\mathsf{coassocr} \quad = \quad [id + i_1 \,, i_2 \cdot i_2] \tag{1}$$

(a) Represente-a sob a forma de um diagrama.

(b) Sabendo que em Haskell as injecções i_1 e i_2 são as funções Left e Right, respectivamente, complete o cálculo que a seguir se sugere para derivar a declaração de coassocr na linguagem Haskell:

$$
\begin{array}{ll}
& \mathsf{coassocr} = [id + i_1 \,, i_2 \cdot i_2] \\
\equiv & \quad \{ \text{ propriedade } \textbf{universal-}+ \} \\
& \left\{ \begin{array}{l} \mathsf{coassocr} \cdot i_1 = id + i_1 \\ \mathsf{coassocr} \cdot i_2 = i_2 \cdot i_2 \end{array} \right. \\
\equiv & \quad \{ \text{ .... vários passos depois .... } \} \\
& \left\{ \begin{array}{l} \mathsf{coassocr} :: \mathsf{Either}\ (\mathsf{Either}\ a\ c)\ b \rightarrow \mathsf{Either}\ a\ (\mathsf{Either}\ c\ b) \\ \mathsf{coassocr}\ (\mathsf{Left}\ (\mathsf{Left}\ a)) = \mathsf{Left}\ a \\ \mathsf{coassocr}\ (\mathsf{Left}\ (\mathsf{Right}\ b)) = \mathsf{Right}\ (\mathsf{Left}\ b) \\ \mathsf{coassocr}\ (\mathsf{Right}\ c) = \mathsf{Right}\ (\mathsf{Right}\ c) \end{array} \right.
\end{array}
$$

3. A lei de **absorção-**$\times$,

$$\begin{tikzcd}
A & A \times B \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & B \\
D \arrow[u, "i"] & D \times E \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[u, "i \times j"] & E \arrow[u, "j"'] \\
& C \arrow[ul, "g"] \arrow[u, "\langle g,h \rangle"] \arrow[ur, "h"'] &
\end{tikzcd} \qquad (i \times j) \cdot \langle g,h \rangle = \langle i \cdot g, j \cdot h \rangle \tag{2}$$

pode deduzir-se da propriedade **universal-**$\times$ resolvendo a equação

$$(i \times j) \cdot \langle g,h \rangle \quad = \quad \langle x,y \rangle \tag{3}$$

em ordem a x e y. Faça-o.

4. Suponha que declara em Haskell o tipo de dados

$$\mathbf{data}\ \mathsf{T}\ a\ b = \mathsf{A}\ a \mid \mathsf{B}\ b$$

Por inspecção, pode verificar os seguintes tipos dos construtores A e B,

$$\mathsf{A} :: a \rightarrow \mathsf{T}\ a\ b$$
$$\mathsf{B} :: b \rightarrow \mathsf{T}\ a\ b$$

usando o comando `:t(ype)` disponível no interpretador GHCi da linguagem. Faça o diagrama do coproduto em que que $in = [\mathsf{A}\ , \mathsf{B}]$ participa e resolva a equação

$$out \cdot in \quad = \quad id \tag{4}$$

em ordem a out, completando o cálculo que a seguir se sugere:

$$\begin{array}{cl}
& out \cdot [\mathsf{A}\ , \mathsf{B}] = id \\
\equiv & \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& out \cdot [\mathsf{A}\ , \mathsf{B}] = [i_1\ , i_2] \\
\equiv & \qquad \{ \text{ .... vários passos depois .... } \} \\
& \left\{ \begin{array}{l} out(\mathsf{A}\ a) = i_1\ a \\ out(\mathsf{B}\ b) = i_2\ b \end{array} \right.
\end{array}$$

5. Mostre que a função factorial

$$fac :: (\mathsf{Integral}\ a) \Rightarrow a \rightarrow a$$
$$fac\ 0 = 1$$
$$fac\ (n+1) = (n+1) * fac\ n$$

satisfaz a equação

$$fac \cdot [\underline{0}\ , (1+)] \quad = \quad [\underline{1}\ , mul] \cdot (id + \langle (1+), fac \rangle) \tag{5}$$

onde $mul\ (a,b) = a * b$ e $\underline{k}$ designa a função "constante-k", isto é tal que $\underline{k}\ x = k$ qualquer que seja x.
Sugestão: derive o código Haskell dado a partir da equação dada.