Cálculo de Programas

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

2021/22 - Ficha nr.° 12 (última)

1. Recorde que o tipo $\mathit{Maybe}\ a = \mathsf{Just}\ a \mid \mathsf{Nothing}$ forma um mónade cuja operação de multiplicação pode ser captada pelo diagrama seguinte:

$$\begin{tikzcd}
\mathit{Maybe}\ (\mathit{Maybe}\ a) \arrow[d, "\mu"'] & (\mathit{Maybe}\ a) + 1 \arrow[l, "\mathsf{in}"'] \arrow[d, "id + !"] \\
\mathit{Maybe}\ a & (\mathit{Maybe}\ a) + 1 \arrow[l, "{[id\,,\,\mathbf{in} \cdot i_2]}"']
\end{tikzcd}
\qquad \mu \cdot \mathsf{in} = [id, \mathsf{in} \cdot i_2] \cdot (id + !)$$

onde $\mathsf{in} = [\mathsf{Just}, \underline{\mathsf{Nothing}}]$. Derive deste diagrama a definição *pointwise* dessa função:

$\mu\ (\mathsf{Just}\ a) = a$
$\mu\ \mathsf{Nothing} = \mathsf{Nothing}$

2. Repare que as projecções $A \xleftarrow{\pi_1} A \times B \xrightarrow{\pi_2} B$ são funções binárias e como tal podem ser "curried", $A^B \xleftarrow{\overline{\pi_1}} A \xrightarrow{\overline{\pi_2}} B^B$. Verifica-se que:

$$\overline{\pi_1} = \mathsf{const} \tag{F1}$$
$$\overline{\pi_2} = \underline{id} \tag{F2}$$

onde $\mathsf{const}\ a = \underline{a}$, a função constante que dá a como resultado. Apresente justificações para os passos das provas respectivas que se seguem:

$$\begin{array}{ll}
& \overline{\pi_1} = \mathsf{const} \\
\equiv & \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{ap} \cdot (\mathsf{const} \times id) = \pi_1 \\
\equiv & \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& (\mathsf{ap} \cdot (\mathsf{const} \times id))\ (a, b) = \pi_1\ (a, b) \\
\equiv & \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{ap}\ (\mathsf{const}\ a, b) = a \\
\equiv & \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{const}\ a\ b = a \\
\equiv & \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \underline{a}\ b = a \\
\Box &
\end{array}$$

$$\begin{array}{ll}
& \overline{\pi_2} = \underline{id} \\
\equiv & \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{ap} \cdot (\underline{id} \times id) = \pi_2 \\
\equiv & \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{ap}\ ((\underline{id} \times id)\ (a, b)) = b \\
\equiv & \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{ap}\ (id, b) = b \\
\equiv & \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& b = b \\
\Box &
\end{array}$$

3. Em Haskell, um mónade declara-se instanciando a classe $Monad$, onde se define a unidade u (que aí se designa por return) e uma operação $x \gg\!= f$, conhecida como aplicação monádica, ou *"binding"* de f a x, que é tal que

$$x \gg\!= f = (f \bullet id)\ x = (\mu \cdot \mathsf{T}\ f)\ x \tag{F3}$$

Mostre que:

$$\mu = (\gg\!= id) \tag{F4}$$
$$g \bullet f = (\gg\!= g) \cdot f \tag{F5}$$
$$x \gg\!= (f \bullet g) = (x \gg\!= g) \gg\!= f \tag{F6}$$

4. Sempre que um functor T é um mónade tem-se:

$$\mathsf{T}\ f = (u \cdot f) \bullet id$$

Definindo-se

$$\theta\ b = \mathsf{T}\ \langle \underline{b}, id \rangle$$

mostre que

$$\theta\ b\ x = \mathbf{do}\ \{ a \leftarrow x; \mathsf{return}\ (b, a) \} \tag{F7}$$

O que faz o operador θ? E qual a sua relação com o operador $lstr$ que consta da biblioteca Cp.hs? (**Sugestão**: use

$$(f \bullet g)\ a \quad = \quad \mathbf{do}\ \{ b \leftarrow g\ a; f\ \ b \} \tag{F8}$$

e outras leis que conhece do cálculo de mónades.)

5. Em Haskell, a instância para listas da função monádica $\mathsf{sequence} : \mathsf{F}\ \ (\mathsf{T}\ \ X) \rightarrow \mathsf{T}\ \ (\mathsf{F}\ \ X)$ é o catamorfismo

$$\begin{array}{l} \mathsf{sequence} = (\!|\, g\, |\!)\ \mathbf{where} \\ \quad g = [\mathsf{return}, id] \cdot (\mathsf{nil} + \mathsf{cons}^\flat) \\ \quad f^\flat\ (x, y) = \mathbf{do}\ \{ a \leftarrow x; b \leftarrow y; \mathsf{return}\ (f\ \ (a, b)) \} \end{array}$$

tal como se mostra neste diagrama:

$$\begin{tikzcd}
(\mathsf{T}\ X)^* \arrow[d, "\mathsf{sequence}"'] & 1 + \mathsf{T}\ X \times (\mathsf{T}\ X)^* \arrow[l, "\mathsf{in}"'] \arrow[d, "id + id \times \mathsf{sequence}"] \\
\mathsf{T}\ (X^*) & 1 + \mathsf{T}\ X \times \mathsf{T}\ (X^*) \arrow[l, dotted, "g"'] \arrow[d, "\mathsf{nil} + \mathsf{cons}^\flat"] \\
 & X^* + \mathsf{T}\ (X^*) \arrow[ul, dotted, "{[\mathsf{return}, id]}"]
\end{tikzcd}$$

Partindo da propriedade universal-cata, derive uma versão de $\mathsf{sequence}$ em Haskell com variáveis que não recorra à composição de funções.