

Cálculo de Programas

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

2021/22 - Ficha nr.º 12 (última)

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

$$\begin{array}{ccc}
 Maybe\ (Maybe\ a) \xleftarrow{\text{in}} (Maybe\ a) + 1 & \mu \cdot \text{in} = [id, \text{in} \cdot i_2] \cdot (id + !) & \\
 \mu \downarrow & & \downarrow id+! \\
 Maybe\ a \xleftarrow{[id, \text{in} \cdot i_2]} (Maybe\ a) + 1 & &
 \end{array}$$

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

$$\begin{aligned}
 \mu\ (Just\ a) &= a \\
 \mu\ Nothing &= Nothing
 \end{aligned}$$

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:

$$\overline{\pi_1} = \text{const} \tag{F1}$$

$$\overline{\pi_2} = \underline{id} \tag{F2}$$

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

$\overline{\pi_1} = \text{const}$ $\equiv \{ \dots \}$ $\text{ap} \cdot (\text{const} \times id) = \pi_1$ $\equiv \{ \dots \}$ $(\text{ap} \cdot (\text{const} \times id))\ (a, b) = \pi_1\ (a, b)$ $\equiv \{ \dots \}$ $\text{ap}\ (\text{const}\ a, b) = a$ $\equiv \{ \dots \}$ $\text{const}\ a\ b = a$ $\equiv \{ \dots \}$ $\underline{a}\ b = a$ \square	$\overline{\pi_2} = \underline{id}$ $\equiv \{ \dots \}$ $\text{ap} \cdot (\underline{id} \times id) = \pi_2$ $\equiv \{ \dots \}$ $\text{ap}\ ((\underline{id} \times id)\ (a, b)) = b$ $\equiv \{ \dots \}$ $\text{ap}\ (id, b) = b$ $\equiv \{ \dots \}$ $b = b$ \square
--	---

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

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

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:

$$T f = (u \cdot f) \bullet id$$

Definindo-se

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

mostre que

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

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

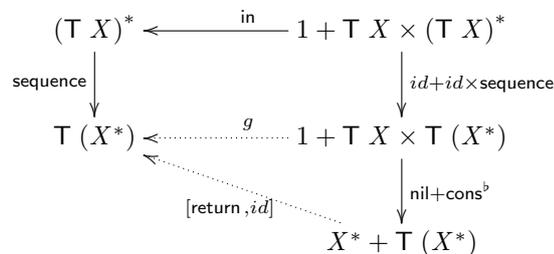
$$(f \bullet g) a = \mathbf{do} \{ b \leftarrow g a; f b \} \tag{F8}$$

e outras leis que conhece do cálculo de mónades.)

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

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

tal como se mostra neste diagrama:



Partindo da propriedade universal-cata, derive uma versão de *sequence* em Haskell com variáveis que não recorra à composição de funções.