# Cálculo de Programas

2.° ano das Licenciaturas em
Engenharia Informática e Ciências da Computação
UNIVERSIDADE DO MINHO

2011/12 - Ficha nr.° 11

1. O tipo $1 + A$ ("apontador" para $A$) foi o primeiro exemplo de mónade apresentado nesta disciplina, em que

$$\mu = [i_1\,, id] \tag{1}$$
$$u = i_2 \tag{2}$$

Mostre que $\mu$ e $u$ satisfazem as duas propriedades que caracterizam um mónade, neste caso:

$$\mu \cdot u = \mu \cdot (id + u) = id \tag{3}$$
$$\mu \cdot \mu = \mu \cdot (id + \mu) \tag{4}$$

2. O tipo

$$\mathbf{data}\ \mathsf{Error}\ a = \mathsf{Error\ String} \mid \mathsf{Ok}\ a$$

que vamos querer usar para gerir a emissão de mensagens de erro em funções parciais, mostra-se facilmente ser um functor definindo

$$\mathsf{Error}\ f = inE \cdot (id + f) \cdot outE \tag{5}$$

onde $inE = [\mathsf{Error}\,, \mathsf{Ok}]$ e

$$outE\ (\mathsf{Error}\ s) = i_1\ s$$
$$outE\ (\mathsf{Ok}\ a) = i_2\ a$$

(Verifique-o como trabalho de casa.) O tipo Error forma, ainda, um mónade desde que equipado com unidade $u = \mathsf{Ok}$ e multiplicação

$$\mu :: \mathsf{Error}\ (\mathsf{Error}\ a) \to \mathsf{Error}\ a$$
$$\mu\ (\mathsf{Error}\ s) = \mathsf{Error}\ s$$
$$\mu\ (\mathsf{Ok}\ a) = a$$

(a) Complete o cálculo que se segue mais abaixo da derivação do código acima a partir da sua definição *pointfree*

$$\begin{tikzcd}
\mathsf{Error}\ a & \mathsf{S}+a \arrow[l, "inE"'] & \mathsf{S}+(\mathsf{S}+a) \arrow[l, "{[i_1\,,id]}"'] & \mathsf{Error}\ (\mathsf{S}+a) \arrow[l, "outE"'] & \mathsf{Error}\ (\mathsf{Error}\ a) \arrow[l, "(\mathsf{Error}\ outE)"'] \arrow[llll, bend left, "\mu"]
\end{tikzcd}$$

onde $\mathsf{S}$ abbrevia $\mathsf{String}$:

$$
\begin{array}{ll}
& \mu = inE \cdot [i_1\ , id] \cdot outE \cdot (\mathsf{Error}\ outE) \\
= & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mu = inE \cdot [i_1\ , id] \cdot outE \cdot (inE \cdot (id + outE) \cdot outE) \\
= & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \vdots \\
\equiv & \quad \{ \ \ldots \text{alguns passos depois} \ldots \ \} \\
& \vdots \\
= & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \left\{ \begin{array}{l} \mu \cdot \mathsf{Error} = \mathsf{Error} \\ \mu \cdot \mathsf{Ok} = id \end{array} \right.
\end{array}
$$

(b) Recorra à mesma definição *pointfree* de $\mu$ para calcular definições *pointwise* para a composição monádica $f \bullet g$ e a operação de *binding*, $x \gg\!= f$.

3. O functor de tipo $\mathsf{LTree}$ forma um mónade cuja unidade $u$ é o construtor $\mathsf{Leaf}$ e cuja multiplicação $\mu$ é a função

   $\mathsf{join} :: \mathsf{LTree}\ (\mathsf{LTree}\ a) \rightarrow \mathsf{LTree}\ a$
   $\mathsf{join} = (\!|[id\ , \mathsf{Fork}]|\!)$

   Recorra às leis de cálculo de catamorfismos que conhece para mostrar que join satisfaz as duas leis (**Multiplicação** e **Unidade** no formulário) que definem um mónade.

4. Aplique as regras para "monadificação" de programas Haskell (escritos ao nível *pointwise*) apresentadas nas aulas teóricas aos combinadores

   $foldr\ f\ b\ [\,] = b$
   $foldr\ f\ b\ (a : x) = f\ a\ (foldr\ f\ b\ x)$

   e

   $\mathsf{map}\ f\ [\,] = [\,]$
   $\mathsf{map}\ f\ (a : x) = (f\ a) : \mathsf{map}\ f\ x$

   por forma a obter as correspondentes versões monádicas, com tipos

   $mmap :: (\mathsf{Monad}\ m) \Rightarrow (a \rightarrow m\ b) \rightarrow [a] \rightarrow m\ [b]$

   e

   $mfoldr :: (\mathsf{Monad}\ m) \Rightarrow (a \rightarrow b \rightarrow m\ b) \rightarrow b \rightarrow [a] \rightarrow m\ b$

   respectivamente.