Cálculo de Programas

2.° ano
Lic. Ciências da Computação e Mestrado Integrado em Engenharia Informática
UNIVERSIDADE DO MINHO

2018/19 - Ficha nr.° 11

1. O desenho que se segue

$$\begin{tikzcd}
\boxed{\text{Problema}} \arrow[r, "divide\ \text{(análise)}"] \arrow[d, "solve\ \text{(problema)}"'] & \boxed{\text{Sub-problemas}} \arrow[d, "solve\ \text{(sub-problemas)}"] \\
\boxed{\text{Solução}} & \boxed{\text{Sub-soluções}} \arrow[l, "conquer\ \text{(síntese)}"]
\end{tikzcd}$$

descreve aquela que deve ser a principal competência de um programador: saber *dividir* um problema complexo em partes e juntar as respectivas sub-soluções para assim resolver o problema inicial.

No Cálculo de Programas, esse desenho é captado pelo conceito de *hilomorfismo*,

$$solve = conquer \cdot (\mathsf{F}\ solve) \cdot divide \qquad \begin{tikzcd}
A \arrow[r, "divide"] \arrow[d, "solve"'] & \mathsf{F}\ A \arrow[d, "\mathsf{F}\ solve"] \\
B & \mathsf{F}\ B \arrow[l, "conquer"]
\end{tikzcd} \qquad \text{(F1)}$$

tendo-se[1]

$$solve = (\!|conquer|\!) \cdot [\!(divide)\!] \qquad \begin{tikzcd}
A \arrow[rr, "divide"] \arrow[d, "{[\!(divide)\!]}"] \arrow[dd, bend right=60, "solve"'] & & \mathsf{F}\ A \arrow[d, "\mathsf{F}[\!(divide)\!]"'] \arrow[dd, bend left=60, "\mathsf{F}\ solve"] \\
\mathsf{T} \arrow[rr, bend left, "\mathsf{out}"] \arrow[d, "(\!|conquer|\!)"] & \cong & \mathsf{F}\ \mathsf{T} \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "\mathsf{F}(\!|conquer|\!)"'] \\
B & & \mathsf{F}\ B \arrow[ll, "conquer"]
\end{tikzcd} \qquad \text{(F2)}$$

onde T é o tipo indutivo que tem F como base.[2] Apresente justificações para os passos seguintes da

[1] Normalmente escreve-se $solve = [\![conquer, divide]\!]$.

[2] Esta estrutura intermédia é designada normalmente por estrutura de dados **virtual** por "não ser ver" quando o algoritmo executa, ficando escondida no 'heap' do sistema de 'run-time' da linguagem recursiva que está a ser utilizada.

demonstração do princípio da *hilo-factorização*, isto é, da equivalência entre (F1) e (F2):

$$
\begin{array}{ll}
& solve = (\!|conquer|\!) \cdot [\!(divide)\!] \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& solve = (conquer \cdot \mathsf{F}\,(\!|conquer|\!) \cdot \mathsf{out}) \cdot (\mathsf{in} \cdot \mathsf{F}\,[\!(divide)\!] \cdot divide) \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& solve = conquer \cdot \mathsf{F}\,(\!|conquer|\!) \cdot \mathsf{F}\,[\!(divide)\!] \cdot divide \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& solve = conquer \cdot \mathsf{F}\,((\!|conquer|\!) \cdot [\!(divide)\!]) \cdot divide \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& solve = conquer \cdot \mathsf{F}\,solve \cdot divide \\
\Box &
\end{array}
$$

2. O algoritmo de "quick-sort" foi definido nas aulas teóricas como o hilomorfismo $qSort = (\!|inord|\!) \cdot [\!(qsep)\!]$ sobre árvores BTree, cujo catamorfismo recorre à função:

$$inord = [\mathsf{nil}\,, f] \textbf{ where } f\ (x, (l, r)) = l + [x] + r$$

Para um certo isomorfismo α, $h = (\!|inord \cdot \alpha|\!) \cdot [\!(qsep)\!]$ ordenará a lista de entrada por ordem inversa. (a) Identifique α, justificando informalmente. (b) Seja agora $h' = (\!|inord|\!) \cdot [\!(\alpha \cdot qsep)\!]$. Demonstre que $h' = h$.

3. Um mónade é um functor T equipado com duas funções μ e u,

$$A \xrightarrow{u} \mathsf{T}\ A \xleftarrow{\mu} \mathsf{T}\ (\mathsf{T}\ A)$$

que satisfazem (para além das naturais, ie. "grátis") as propriedades

$$\mu \cdot u = id = \mu \cdot \mathsf{T}\ u$$
$$\mu \cdot \mu = \mu \cdot \mathsf{T}\ \mu$$

— identifique-as no formulário — com base nas quais se pode definir a *composição monádica*:

$$f \bullet g = \mu \cdot \mathsf{T}\ f \cdot g.$$

(Identifique-a também no formulário.) Demonstre os factos seguintes:

$$\mu = id \bullet id \tag{F3}$$
$$f \bullet u = f \quad \wedge \quad f = u \bullet f \tag{F4}$$
$$(f \cdot g) \bullet h = f \bullet (\mathsf{T}\ g \cdot h) \tag{F5}$$
$$\mathsf{T}\,f = (u \cdot f) \bullet id \tag{F6}$$

4. A função $discollect : (A \times B^*)^* \rightarrow (A \times B)^*$ que apareceu (sem ser definida) numa questão da ficha nº2 não é mais do que

$$discollect = lstr \bullet id \tag{F7}$$

— onde $lstr\ (a, x) = [(a, b) \mid b \leftarrow x]$ — no mónade das listas, $\mathsf{T}\ A = A^*$,

$$A \xrightarrow{singl} A^* \xleftarrow{\mathsf{concat}} (A^*)^*$$

onde $u = singl$ e $\mu = \mathsf{concat} = (\!|[\mathsf{nil}\,, \mathsf{conc}]|\!)$. Recordando a lei de absorção-cata (para listas), derive uma definição recursiva para $discollect$ que não use nenhum dos combinadores 'point-free' estudados nesta disciplina.