Cálculo de Programas

2.° ano da Licenciatura em Engenharia Informática da
Universidade do Minho

2010/11 - Ficha nr.° 10

1. O desenho que se segue

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

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

No Cálculo de Programas, o esquema desenhado acima é captado pelo diagrama de um *hilomorfismo*, cujo ingrediente principal é a fixação do padrão de organização das sub-soluções, captado pelo *functor* polinomial F:

$$\begin{array}{rcl} solve & = & [\![conquer, divide]\!] \\ & = & (\!| conquer |\!) \cdot [\!(divide)\!] \end{array}$$

a que corresponde o diagrama

$$\begin{tikzcd}
A \arrow[r, "divide"] \arrow[d, "solve"'] & \mathsf{F}A \arrow[d, "\mathsf{F}solve"] \\
B & \mathsf{F}B \arrow[l, "conquer"]
\end{tikzcd}$$

isto é, tem-se a equação

$$solve = conquer \cdot \mathsf{F}\, solve \cdot divide \qquad (1)$$

Mostre que

- para $divide = \mathsf{out}$ se tem $solve = (\!| conquer |\!)$
- para $conquer = \mathbf{in}$ se tem $solve = [\!(divide)\!]$.

2. No quadro que se segue mostra-se a classificação de algumas funções conhecidas de acordo com o respectivo F:

T	$\mathsf{F\,X}$	**Serialização**	**Ordenação**	**Inversão**	**Factorial**	**Quadrado**	**Outros**
$\mathbb{N}_0$	$1 + X$						$(a*), (`div`b)$
Listas	$1 + A \times X$		$iSort$	$invl$	fac	sq	$look$
BTree	$1 + A \times X^2$	*in/pré/pós*	$qSort$				$hanoi, traces$
LTree	$A + X^2$	$tips$	$mSort$	$invLTree$	$dfac$	dsq	fib

(a) A divisão inteira x ‘div‘ y obtém-se da equação (1) acima definindo

$$
\begin{array}{l}
div = flip\ aux\ \mathbf{where} \\
\quad aux\ y = [\![conquer, divide\ y]\!] \\
\quad conquer = \mathsf{in} \\
\quad divide\ y\ x \\
\qquad \mid x < y = i_1\ () \\
\qquad \mid x \geqslant y = i_2\ (x - y)
\end{array}
$$

Resolva a equação e introduza varáveis por forma a deduzir a versão habitual do algoritmo:

$$
\begin{array}{l}
x\ \text{‘}div\text{‘}\ y \mid x < y = 0 \\
\qquad\qquad\quad \mid x \geqslant y = (x - y)\ \text{‘}div\text{‘}\ y + 1
\end{array}
$$

(b) Identifique a linha e coluna onde deve, do quadro acima, colocar o hilomorfismo de *bubble sorting*, identificando para ele os genes $divide$ e $conquer$:

$$
\begin{array}{l}
bSort :: \mathsf{Ord}\ a \Rightarrow [a] \rightarrow [a] \\
bSort\ [\,] = [\,] \\
bSort\ l\ = \mathbf{let}\ (x, m) = \mathsf{bubble}\ l \\
\qquad\qquad \mathbf{in}\ \ x : bSort\ m \\
\mathsf{bubble} :: \mathsf{Ord}\ a \Rightarrow [a] \rightarrow (a, [a]) \\
\mathsf{bubble}\ [x] \quad\ = (x, [\,]) \\
\mathsf{bubble}\ (x : l) = \mathbf{let}\ (y, m) = \mathsf{bubble}\ l \\
\qquad\qquad\qquad \mathbf{in}\ \ \mathbf{if}\ x < y\ \mathbf{then}\ (x, y : m)\ \mathbf{else}\ (y, x : m)
\end{array}
$$

(c) Num dos outros algoritmos de ordenação do quadro acima (qual?), o passo de $divide$ é a função sep que se segue

$$
\begin{array}{l}
sep :: [a] \rightarrow ([a], [a]) \\
sep = (\![[\langle nil, nil \rangle\ , (\mathsf{cons} \times id) \cdot \mathsf{assocl} \cdot (id \times \mathsf{swap})]]\!) \\
\quad \mathbf{where}\ nil\ \_ = [\,] \\
\qquad \mathsf{cons}\ (a, b) = a : b
\end{array}
$$

cf. biblioteca `LTree.hs`. Mostre, usando a lei de recursividade múltipla para listas ($\mathsf{F}\ f = id + id \times f$) que sep se poderia ter escrito (ao nível *pointwise*) sob a forma de duas funções mutuamente recursivas $odds$ e $evens$,

$$
\begin{array}{l}
sep :: [a] \rightarrow ([a], [a]) \\
sep = \langle odds, evens \rangle\ \mathbf{where} \\
\quad odds\ [\,] = [\,] \\
\quad odds\ (h : t) = h : (evens\ t) \\
\quad evens\ [\,] = [\,] \\
\quad evens\ (h : t) = odds\ t
\end{array}
$$

a primeira seleccionando os elementos que ocupam as posições ímpares e a segunda fazendo o mesmo para as pares.