Cálculo de Programas

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

2017/18 - Ficha nr.° 9

1. Consultando as bibliotecas em Haskell disponíveis no material pedagógico, complete o seguinte quadro relativo aos tipos indutivos que aí se codificam:

T	**Descrição**	in	$\mathsf{B}\ (X, Y)$	$\mathsf{B}\ (f, g)$	$\mathsf{F}\ f$	$\mathsf{T}\ f$
$\mathbb{N}_0$	Números naturais					
A^*	Sequências finitas de A					
$\mathsf{BTree}\ A$	Árvores binárias de A					
$\mathsf{LTree}\ A$	Árvores com A nas folhas					

2. Recorra à lei da absorção-cata, entre outras, para verificar as seguintes propriedades sobre listas

$$\mathsf{length} = \mathsf{sum} \cdot (\mathsf{map}\ \underline{1}) \tag{F1}$$
$$\mathsf{length} = \mathsf{length} \cdot (\mathsf{map}\ f) \tag{F2}$$

onde length, sum e map são catamorfismos de listas que conhece.

3. A função concat, extraída do *Prelude* do Haskell, é o catamorfismo de listas

$$\mathsf{concat} = (\![\mathsf{nil}\ , \mathsf{conc}]\!) \tag{F3}$$

onde $\mathsf{conc}\ (x, y) = x +\!\!+\ y$ e $\mathsf{nil}\ \_ = [\,]$. Apresente justificações para a prova da propriedade

$$\mathsf{length} \cdot \mathsf{concat} = \mathsf{sum} \cdot \mathsf{map}\ \mathsf{length} \tag{F4}$$

que a seguir se apresenta, onde é de esperar que as leis de *fusão*-cata e *absorção*-cata desempenhem um papel importante:

$$
\begin{array}{cl}
& \mathsf{length} \cdot \mathsf{concat} = \mathsf{sum} \cdot \mathsf{map}\ \mathsf{length} \\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \} \\
& \mathsf{length} \cdot \mathsf{concat} = (\![\underline{0}\ , \mathsf{add}]\!) \cdot (\!\mathsf{in} \cdot \mathsf{B}\ (\mathsf{length}, id)\!) \\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \} \\
& \mathsf{length} \cdot \mathsf{concat} = (\![\underline{0}\ , \mathsf{add}] \cdot (id + \mathsf{length} \times id)\!) \\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \} \\
& \mathsf{length} \cdot (\![\mathsf{nil}\ , \mathsf{conc}]\!) = (\![\underline{0}\ , \mathsf{add} \cdot (\mathsf{length} \times id)]\!) \\
\Leftarrow & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \} \\
& \mathsf{length} \cdot [\mathsf{nil}\ , \mathsf{conc}] = [\underline{0}\ , \mathsf{add} \cdot (\mathsf{length} \times id)] \cdot (id + id \times \mathsf{length})
\end{array}
$$

$$
\begin{array}{ll}
\equiv & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \left\{ \begin{array}{l} \mathsf{length} \cdot \mathsf{nil} = \underline{0} \\ \mathsf{length} \cdot \mathsf{conc} = \mathsf{add} \cdot (\mathsf{length} \times id) \cdot (id \times \mathsf{length}) \end{array} \right. \\
\equiv & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{length} \cdot \mathsf{conc} = \mathsf{add} \cdot (\mathsf{length} \times \mathsf{length}) \\
\equiv & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& true \\
\Box &
\end{array}
$$

4. Recorra à lei de absorção-cata para demonstrar a propriedade

$$count \cdot (\mathsf{BTree}\ f) \quad = \quad count \tag{F5}$$

onde $\mathsf{BTree}\ A \xrightarrow{count} \mathbb{N}_0$ é o catamorfismo

$$count = (\![[\mathsf{zero}\ , \mathsf{succ} \cdot \mathsf{add} \cdot \pi_2]]\!)$$

onde $\mathsf{zero}\ \_ = 0$, $\mathsf{succ}\ n = n + 1$ e $\mathsf{add}\ (a, b) = a + b$. **NB:** recorda-se que a base do tipo BTree é $\mathsf{B}\ (f, g) = id + f \times (g \times g)$.

5. Recorra às leis dos catamorfismos para demonstrar a propriedade natural

$$(\mathsf{T}\ f) \cdot \mathsf{mirror} \quad = \quad \mathsf{mirror} \cdot (\mathsf{T}\ f) \tag{F6}$$

onde mirror é o catamorfismo

$$
\begin{array}{l}
\mathsf{mirror} :: \mathsf{LTree}\ a \to \mathsf{LTree}\ a \\
\mathsf{mirror} = (\![\mathsf{in} \cdot (id + \mathsf{swap})]\!)
\end{array}
$$

que "espelha"uma árvore e $\mathsf{T}\ f = (\![\mathsf{in} \cdot (f + id)]\!)$ é o correspondente functor de tipo.

6. Um *anamorfismo* é um *"catamorfismo ao contrário"*, isto é, uma função $k : A \to \mathsf{T}$ tal que

$$k = \mathsf{in} \cdot \mathsf{F}\ k \cdot g \tag{F7}$$

escrevendo-se $k = [\!(g)\!]$. Mostre que o anamorfismo de listas

$$k = [\!((id + \langle f, id \rangle) \cdot \mathsf{out}_{\mathbb{N}_0})\!] \tag{F8}$$

descrito pelo diagrama

$$
\begin{tikzcd}
{\mathbb{N}_0}^* & & 1 + \mathbb{N}_0 \times {\mathbb{N}_0}^* \arrow[ll, "\mathsf{in}"'] \\
\mathbb{N}_0 \arrow[u, "k"] \arrow[r, "\mathsf{out}_{\mathbb{N}_0}"'] & 1 + \mathbb{N}_0 \arrow[r, "id + \langle f, id \rangle)"'] & 1 + \mathbb{N}_0 \times \mathbb{N}_0 \arrow[u, "id + id \times k"']
\end{tikzcd}
$$

é a função

$$
\begin{array}{l}
k\ 0 = [\,] \\
k\ (n + 1) = (2\ n + 1) : k\ n
\end{array}
$$

para $f\ n = 2\ n + 1$. (Que faz esta função?)