Cálculo de Programas

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

2019/20 - Ficha nr.° 10

1. O formulário desta disciplina apresenta duas definições alternativas para o functor $\mathsf{T}\ f$ de um tipo indutivo, uma como *catamorfismo* e outra como *anamorfismo*. Identifique-as e acrescente justificações à seguinte prova de que essas definições são equivalentes:

$$
\begin{array}{ll}
& \mathsf{T}\ f = (\!|\ \mathsf{in} \cdot \mathsf{B}\ (f, id)\ |\!) \\
\equiv & \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{T}\ f \cdot \mathsf{in} = \mathsf{in} \cdot \mathsf{B}\ (f, id) \cdot \mathsf{F}\ (\mathsf{T}\ f) \\
\equiv & \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{T}\ f \cdot \mathsf{in} = \mathsf{in} \cdot \mathsf{B}\ (id, \mathsf{T}\ f) \cdot \mathsf{B}\ (f, id) \\
\equiv & \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{out} \cdot \mathsf{T}\ f = \mathsf{F}\ (\mathsf{T}\ f) \cdot \mathsf{B}\ (f, id) \cdot \mathsf{out} \\
\equiv & \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{T}\ f = [\!(\mathsf{B}\ (f, id) \cdot \mathsf{out})\!] \\
\Box &
\end{array}
$$

2. Mostre que o catamorfismo de listas $\mathsf{length} = (\!|\ [\mathsf{zero}\ , \mathsf{succ} \cdot \pi_2]\ |\!)$ é a mesma função que o *anamorfismo* de naturais $[\!((id + \pi_2) \cdot \mathsf{out}_{\mathsf{List}})\!]$.

3. Mostre que o anamorfismo que calcula os sufixos de uma lista

$$suffixes = [\!(g)\!]\ \textbf{where}\ g = (id + \langle \mathsf{cons}, \pi_2 \rangle) \cdot \mathsf{out}$$

é a função

$$
\begin{array}{l}
suffixes\ [\,] = [\,] \\
suffixes\ (h : t) = (h : t) : suffixes\ t
\end{array}
$$

4. Mostre que a função mirror da ficha nr.º 7 se pode definir como o anamorfismo

$$\mathsf{mirror} = [\!((id + \mathsf{swap}) \cdot \mathsf{out})\!] \tag{F1}$$

onde out é a conversa de in. Volte a demonstrar a propriedade $\mathsf{mirror} \cdot \mathsf{mirror} = id$, desta vez recorrendo à lei de fusão dos anamorfismos.

5. O algoritmo da divisão inteira (ao lado) corresponde ao anamorfismo de naturais $(\div n) = [\![g\ n]\!]$ em que

$$\begin{array}{l} g\ n\ x = \\ \quad \textbf{if}\ x < n \\ \quad \textbf{then}\ i_1\ ()\ \textbf{else}\ i_2\ (x - n) \end{array}$$

$$\boxed{\begin{array}{l} m \div n \\ \quad \mid m < n = 0 \\ \quad \mid \mathsf{otherwise} = 1 + (m - n) \div n \end{array}}$$

É de esperar que o algoritmo dado satisfaça a propriedade $(m * n) \div n = m$, isto é, $(\div n) \cdot (*n) = id$. Complete a prova seguinte dessa propriedade, que se baseia na lei de fusão dos anamorfismos (identifique-a no formulário):

$$\begin{array}{ll}
 & (\div n) \cdot (*n) = id \\
\equiv & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & [\![g\ n]\!] \cdot (*n) = [\![\mathsf{out}]\!] \\
\Leftarrow & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & (g\ n) \cdot (*n) = (id + (*n)) \cdot \mathsf{out} \\
\equiv & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & (g\ n) \cdot [\underline{0}\ , (n+) \cdot (*n)] = id + (*n) \\
\Leftarrow & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & (g\ n) \cdot [\underline{0}\ , (n+)] = id \\
\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} g\ n \cdot \underline{0} = i_1 \\ g\ n \cdot (n+) = i_2 \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 \} \\
 & \left\{ \begin{array}{l} g\ n\ 0 = i_1\ () \\ g\ n\ (n + x) = i_2\ x \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 \} \\
 & \left\{ \begin{array}{l} g\ n\ 0 = i_1\ () \\ g\ n\ (n + x) = i_2\ x \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 \} \\
 & \left\{ \begin{array}{l} i_1\ () = i_1\ () \\ \textbf{if}\ n + x < n\ \textbf{then}\ i_1\ ()\ \textbf{else}\ i_2\ ((n + x) - n) = i_2\ x \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 \} \\
 & \textbf{if}\ x < 0\ \textbf{then}\ i_1\ ()\ \textbf{else}\ i_2\ x = i_2\ x \\
\equiv & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & true
\end{array}$$

6. 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. Identifique α, justificando informalmente.