

Cálculo de Programas

2.º ano

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

2019/20 - Ficha nr.º 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	B (X, Y)	B (f, g)	F f	T f
A^*	Sequências finitas de A					
BTree A	Árvores binárias de A					
LTree A	Árvores com A nas folhas					
\mathbb{N}_0	Números naturais					

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

$$\text{length} = \text{sum} \cdot (\text{map } \underline{1}) \quad (\text{F1})$$

$$\text{length} = \text{length} \cdot (\text{map } f) \quad (\text{F2})$$

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

$$\text{concat} = ([\text{nil}, \text{conc}]) \quad (\text{F3})$$

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

$$\text{length} \cdot \text{concat} = \text{sum} \cdot \text{map } \text{length} \quad (\text{F4})$$

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

$$\begin{aligned} & \text{length} \cdot \text{concat} = \text{sum} \cdot \text{map } \text{length} \\ \equiv & \quad \{ \dots\dots\dots \} \\ & \text{length} \cdot \text{concat} = ([\underline{0}, \text{add}] \cdot (\text{id} + \text{length} \times \text{id})) \\ \equiv & \quad \{ \dots\dots\dots \} \\ & \text{length} \cdot ([\text{nil}, \text{conc}]) = ([\underline{0}, \text{add} \cdot (\text{length} \times \text{id})]) \\ \leftarrow & \quad \{ \dots\dots\dots \} \\ & \text{length} \cdot [\text{nil}, \text{conc}] = [\underline{0}, \text{add} \cdot (\text{length} \times \text{id})] \cdot (\text{id} + \text{id} \times \text{length}) \\ \equiv & \quad \{ \dots\dots\dots \} \end{aligned}$$

$$\begin{aligned}
& \left\{ \begin{array}{l} \text{length} \cdot \text{nil} = \underline{0} \\ \text{length} \cdot \text{conc} = \text{add} \cdot (\text{length} \times \text{id}) \cdot (\text{id} \times \text{length}) \end{array} \right. \\
\equiv & \quad \{ \dots \dots \dots \} \\
& \text{length} \cdot \text{conc} = \text{add} \cdot (\text{length} \times \text{length}) \\
\equiv & \quad \{ \dots \dots \dots \} \\
& \text{true} \\
& \square
\end{aligned}$$

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

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

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

$$\text{count} = \llbracket [\text{zero}, \text{succ} \cdot \text{add} \cdot \pi_2] \rrbracket$$

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

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

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

onde mirror é o catamorfismo

$$\begin{aligned}
\text{mirror} &:: \text{LTree } a \rightarrow \text{LTree } a \\
\text{mirror} &= \llbracket \text{in} \cdot (\text{id} + \text{swap}) \rrbracket
\end{aligned}$$

que “espelha” uma árvore e $\text{T } f = \llbracket \text{in} \cdot (f + \text{id}) \rrbracket$ é o correspondente functor de tipo.

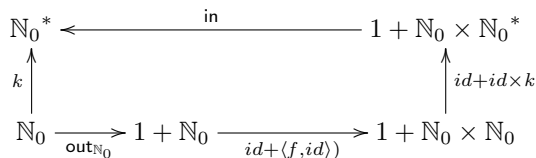
6. Um *anamorfismo* é um “*catamorfismo ao contrário*”, isto é, uma função $k : A \rightarrow \text{T}$ tal que

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

escrevendo-se $k = \llbracket (g) \rrbracket$. Mostre que o anamorfismo de listas

$$k = \llbracket ((\text{id} + \langle f, \text{id} \rangle) \cdot \text{out}_{\mathbb{N}_0}) \rrbracket \tag{F8}$$

descrito pelo diagrama



é a função

$$\begin{aligned}
k \ 0 &= [] \\
k \ (n + 1) &= (2 \ n + 1) : k \ n
\end{aligned}$$

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