Cálculo de Programas

2.º Ano de MiEI+LCC (Universidade do Minho)
Ano Lectivo de 2016/17

Exame da Época especial — 25 de Julho de 2017
9h00–11h00
Sala CP2-204

---

*Este teste consta de **8** questões que valem, cada uma, 2.5 valores. O tempo médio estimado para resolução de cada questão é de 12 min.*

PROVA SEM CONSULTA (1h30m)

Questão 1 Considere a função

$$\alpha \quad = \quad \langle ! + !, \beta \rangle \tag{E1}$$

onde $\beta = [id , id]$ e $! : A \to 1$ é a única função (constante) que pode ser definida com tipo $A \to 1$.

Identifique o isomorfismo que α testemunha, desenhando-o sob a forma de um diagrama de tipos, e derive a partir de (E1) uma definição de α em Haskell *pointwise* (i.e. Haskell *standard*, com variáveis).

Questão 2 No contexto da definição seguinte,

$$f \ h = \pi_1 \cdot \pi_2 \to \mathsf{swap}, h \tag{E2}$$

determine o tipo de h e derive a respectiva propriedade natural.

Questão 3 Considere as funções de ordem superior

$$\begin{array}{ll} junc : A^B \times A^C \to A^{B+C} & unjunc : A^{B+C} \to A^B \times A^C \\ junc \ (f, g) = [f \ , g] & unjunc \ k = (k \cdot i_1, k \cdot i_2) \end{array} \tag{E3}$$

Mostre que $junc \cdot unjunc = id$ e que $unjunc \cdot junc = id$ e que, portanto, o isomorfismo de exponenciais $A^{B+C} \cong A^B \times A^C$ se verifica.

Questão 4 Pretendendo-se uma função que conte o número de folhas de uma LTree apareceram duas soluções: uma é o catamorfismo

$$count = (\!|[\mathsf{one} , \mathsf{add}]|\!) \tag{E4}$$

onde $\mathsf{one} = \underline{1}$ e $\mathsf{add} \ (x, y) = x + y$; a outra,

$$count = \mathsf{length} \cdot tips \tag{E5}$$

baseia-se em duas funções que conhece das bibliotecas e trabalho prático da disciplina.

Recorrendo às leis dos catamorfismos, entre outras, mostre que as duas propostas (E4) e (E5) são a mesma função. **NB:** assuma a propriedade $\mathsf{length} \ (x +\!\!+ \ y) = (\mathsf{length} \ x) + (\mathsf{length} \ y)$, se dela precisar.

Questão 5 Numa página de Stack Overflow[1] alguém respondeu afirmativamente à pergunta

Pode fazer-se unzip *num só passo?*

com a versão

$$\begin{array}{l}\mathsf{unzip}\ [\,] = ([\,],[\,])\\ \mathsf{unzip}\ ((a,b):xs) = (a:as, b:bs)\ \mathbf{where}\ (as,bs) = \mathsf{unzip}\ xs\end{array}$$

Ora o que essa página não faz é explicar como é que os dois passos de

$$\mathsf{unzip}\ xs = (\mathsf{map}\ \pi_1\ xs, \mathsf{map}\ \pi_2\ xs)$$

se fundem num só. Complete com justificações baseadas do cálculo de programas que estudou nesta disciplina a derivação que se segue dessa evidência:

$$\begin{array}{ll}
& \mathsf{unzip}\ xs = (\mathsf{map}\ \pi_1\ xs, \mathsf{map}\ \pi_2\ xs)\\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \}\\
& \mathsf{unzip} = \langle \mathsf{map}\ \pi_1, \mathsf{map}\ \pi_2 \rangle\\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \}\\
& \mathsf{unzip} = \langle (\!| \mathsf{in} \cdot \mathsf{B}\ (\pi_1, id) |\!), (\!| \mathsf{in} \cdot \mathsf{B}\ (\pi_2, id) |\!) \rangle\\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \}\\
& \mathsf{unzip} = (\!| \langle \mathsf{in} \cdot \mathsf{B}\ (\pi_1, id) \cdot \mathsf{F}\ \pi_1, \mathsf{in} \cdot \mathsf{B}\ (\pi_2, id) \cdot \mathsf{F}\ \pi_2 \rangle |\!)\\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \}\\
& \mathsf{unzip} = (\!| \langle \mathsf{in} \cdot \mathsf{B}\ (\pi_1, \pi_1), \mathsf{in} \cdot \mathsf{B}\ (\pi_2, \pi_2) \rangle |\!)\\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \}\\
& \mathsf{unzip} \cdot \mathsf{in} = \langle [\mathsf{nil}, \mathsf{cons} \cdot (\pi_1 \times \pi_1)], [\mathsf{nil}, \mathsf{cons} \cdot (\pi_2 \times \pi_2)] \rangle \cdot (id + id \times \mathsf{unzip})\\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \}\\
& \left\{ \begin{array}{l} \mathsf{unzip} \cdot \mathsf{nil} = \langle \mathsf{nil}, \mathsf{nil} \rangle\\ \mathsf{unzip} \cdot \mathsf{cons} = (\mathsf{cons} \times \mathsf{cons}) \cdot \langle \pi_1 \times \pi_1, \pi_2 \times \pi_2 \rangle \cdot (id \times \mathsf{unzip}) \end{array} \right.\\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \}\\
& \left\{ \begin{array}{l} \mathsf{unzip}\ [\,] = ([\,],[\,])\\ \mathsf{unzip}\ ((a,b):xs) = ((\mathsf{cons} \times \mathsf{cons}) \cdot \langle \pi_1 \times \pi_1, \pi_2 \times \pi_2 \rangle)\ ((a,b), \mathsf{unzip}\ xs) \end{array} \right.\\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \}\\
& \left\{ \begin{array}{l} \mathsf{unzip}\ [\,] = ([\,],[\,])\\ \mathsf{unzip}\ ((a,b):xs) = ((\mathsf{cons} \times \mathsf{cons}) \cdot \langle \pi_1 \times \pi_1, \pi_2 \times \pi_2 \rangle)\ ((a,b), (as,bs))\ \mathbf{where}\ (as,bs) = \mathsf{unzip}\ xs \end{array} \right.\\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \}\\
& \left\{ \begin{array}{l} \mathsf{unzip}\ [\,] = ([\,],[\,])\\ \mathsf{unzip}\ ((a,b):xs) = (\mathsf{cons} \times \mathsf{cons})\ ((a,as),(b,bs))\ \mathbf{where}\ (as,bs) = \mathsf{unzip}\ xs \end{array} \right.\\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \}\\
& \left\{ \begin{array}{l} \mathsf{unzip}\ [\,] = ([\,],[\,])\\ \mathsf{unzip}\ ((a,b):xs) = ((a:as),(b:bs))\ \mathbf{where}\ (as,bs) = \mathsf{unzip}\ xs \end{array} \right.\\
\Box &
\end{array}$$

[1] Cf. `https://stackoverflow.com/questions/18287848/unzip-in-one-pass`.

Questão 6 A função

$$\begin{array}{l} takeWhile\ p\ [\,] = [\,] \\ takeWhile\ p\ (a : xs) \mid \neg\ (p\ a) = [\,] \\ takeWhile\ p\ (a : xs) \mid \mathsf{otherwise} = a : takeWhile\ xs \end{array}$$

é standard em Haskell e corresponde ao catamorfismo

$$takeWhile\ p = (\!|[\mathsf{nil}, (\tilde{p} \cdot \pi_1) \rightarrow \mathsf{nil}, \mathsf{cons}]|\!) \tag{E6}$$

onde $\tilde{p}$ designa a negação de p, isto é, $\tilde{p}\ x = \neg\ (p\ x)$. Mostre (por absorção-cata em listas) que

$$takeWhile\ p = f \cdot (\mathsf{map}\ \tilde{p}?) \tag{E7}$$

onde

$$f = (\!|[\mathsf{nil}, \mathsf{in}] \cdot (id + \mathsf{distl})|\!) \tag{E8}$$

NB: assuma a propriedade

$$\mathsf{distl} \cdot (p? \times id) = (p \cdot \pi_1)? \tag{E9}$$

se dela precisar.

Questão 7 Recorde uma função bem conhecida do Prelude do Haskell:

$$\begin{array}{l} \mathsf{zip}\ [\,]\ \_ = [\,] \\ \mathsf{zip}\ \_\ [\,] = [\,] \\ \mathsf{zip}\ (a : x)\ (b : y) = (a, b) : \mathsf{zip}\ x\ y \end{array}$$

Mostre que $\widehat{\mathsf{zip}} = [\!(h)\!]$ identificando o gene h do anamorfismo no diagrama seguinte,

$$\begin{tikzcd} A^* \times B^* \arrow[r, "h"] \arrow[d, "\widehat{\mathsf{zip}}"'] & \cdots \arrow[d, "id + id \times \widehat{\mathsf{zip}}"] \\ (A \times B)^* & \cdots \arrow[l, "\mathsf{in}"] \end{tikzcd}$$

e preenchendo as reticências. Justifique a sua resposta.

Questão 8 Mostre que a função map monádica $mmap$, definida por

$$\left\{ \begin{array}{l} mmap\ f\ [\,] = \mathsf{return}\ [\,] \\ mmap\ f\ (h : t) = \mathbf{do}\ \{\, a \leftarrow f\ h; b \leftarrow mmap\ f\ t; \mathsf{return}\ (a : b) \} \end{array} \right. \tag{E10}$$

é o catamorfismo de listas

$$mmap\ f = (\!|[\mathsf{return} \cdot \mathsf{nil}, \mathit{lift}\ \mathsf{cons}] \cdot (id + f \times id)|\!) \tag{E11}$$

onde

$$\mathit{lift}\ h\ (x, y) = \mathbf{do}\ \{\, a \leftarrow x; b \leftarrow y; \mathsf{return}\ (\overline{h}\ a\ b) \} \tag{E12}$$

ANEXO — Catálogo de tipos de dados estudados na disciplina.

1. Números naturais:

$$\mathsf{T} = \mathbb{N}_0 \qquad \left\{ \begin{array}{l} \mathsf{F}\ X = 1 + X \\ \mathsf{F}\ f = id + f \end{array} \right. \qquad \mathsf{in} = [\underline{0}, \mathsf{succ}] \tag{E13}$$

   Haskell: Int inclui $\mathbb{N}_0$.

2. Listas de elementos em A:

$$\mathsf{T} = A^* \qquad \left\{ \begin{array}{l} \mathsf{F}\ X = 1 + A \times X \\ \mathsf{F}\ f = id + id \times f \end{array} \right. \qquad \mathsf{in} = [\mathsf{nil}, \mathsf{cons}] \tag{E14}$$

   Haskell: $[\,a\,]$.

3. Árvores com informação de tipo A nos nós:

$$\mathsf{T} = \mathsf{BTree}\ A \qquad \left\{ \begin{array}{l} \mathsf{F}\ X = 1 + A \times X^2 \\ \mathsf{F}\ f = id + id \times f^2 \end{array} \right. \qquad \mathsf{in} = [\underline{Empty}, Node] \tag{E15}$$

   Haskell: $\mathbf{data}\ \mathsf{BTree}\ a = Empty \mid Node\ (a, (\mathsf{BTree}\ a, \mathsf{BTree}\ a))$.

4. Árvores com informação de tipo A nas folhas:

$$\mathsf{T} = \mathsf{LTree}\ A \qquad \left\{ \begin{array}{l} \mathsf{F}\ X = A + X^2 \\ \mathsf{F}\ f = id + f^2 \end{array} \right. \qquad \mathsf{in} = [Leaf, Fork] \tag{E16}$$

   Haskell: $\mathbf{data}\ \mathsf{LTree}\ a = Leaf\ \ a \mid Fork\ (\mathsf{LTree}\ a, \mathsf{LTree}\ a)$.

5. Árvores com informação nos nós e nas folhas:

$$\mathsf{T} = \mathsf{FTree}\ B\ A \qquad \left\{ \begin{array}{l} \mathsf{F}\ X = B + A \times X^2 \\ \mathsf{F}\ f = id + id \times f^2 \end{array} \right. \qquad \mathsf{in} = [Unit, Comp] \tag{E17}$$

   Haskell: $\mathbf{data}\ \mathsf{FTree}\ b\ a = Unit\ b \mid Comp\ (a, (\mathsf{FTree}\ b\ a, \mathsf{FTree}\ b\ a))$.