Cálculo de Programas

2.º Ano de LCC e da LEI (Universidade do Minho)
Ano Lectivo de 2009/10

Exame de recurso — 12 de Julho 2010
14h30
Salas 2201, 2202 e 2203

---

*Esta prova consta de **10** questões que valem, cada uma, 2 valores. O tempo estimado para resolução de cada questão é, em média, de 15 min.*

PROVA SEM CONSULTA (2h30m)

Questão 1 Identifique, justificadamente, os tipos das funções

$$f = \langle \pi_1 \cdot \pi_1, (\pi_2 \times id) \rangle$$
$$g = \langle id \times \pi_1, (\pi_2 \cdot \pi_2) \rangle$$

e verifique se há lugar à igualdade $g \cdot f = id$.

Questão 2 Demonstre a lei do condicional

$$p \to (q \to c, d), c \quad = \quad (p \Rightarrow q) \to c, d$$

sabendo que

$$(p \Rightarrow q)? \quad = \quad [q? \, , i_1] \cdot p? \tag{1}$$

é uma propriedade da implicação de predicados.

Questão 3 O diagrama seguinte esboça a propriedade natural (i.é, "grátis") de um isomorfismo α que deve conhecer bem:

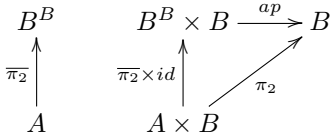
$$\begin{array}{ccc} \cdots & \xleftarrow{\alpha} & 1 + \cdots \\ {\scriptstyle \mathsf{map}\ f}\downarrow & & \downarrow{\scriptstyle id + f \times \mathsf{map}\ f} \\ \cdots & \xleftarrow[\alpha]{} & 1 + \cdots \end{array}$$

Identifique esse isomorfismo α, complete o diagrama e infira a propriedade natural do inverso de α, também com base no diagrama.

Questão 4 Considerando a seguinte propriedade da exponenciação

$$ap \cdot \langle \underline{f}, g \rangle \quad = \quad f \cdot g \tag{2}$$

e o diagrama



$$\overline{\pi_2} = \underline{id} \tag{3}$$

que mostra uma propriedade interessante da projecção π_2, obtida por transposição, apresente justificações para os passos que se seguem da dedução de uma bem conhecida propriedade de cancelamento:

$$\begin{array}{ll} & \pi_2 \cdot \langle f, g \rangle = g \\ \equiv & \quad \{ \text{ Passo A } \} \\ & ap \cdot (\underline{id} \times id) \cdot \langle f, g \rangle = g \\ \equiv & \quad \{ \text{ Passo B } \} \\ & ap \cdot \langle \underline{id}, g \rangle = g \\ \equiv & \quad \{ \text{ Passo C } \} \\ & id \cdot g = g \\ \equiv & \quad \{ \text{ Passo D } \} \\ & \text{TRUE} \end{array}$$

Questão 5 Mostre que $(\!|[nil\ , \pi_2]|\!) = nil$, em que $nil\ \_ = [\,]$.

Questão 6 Considere o tipo das *listas não vazias*, em Haskell

$$\mathbf{data}\ NEL\ a = Sing\ a \mid Add\ (a, NEL\ a)$$

com $\mathbf{in} = [Sing\ , Add]$ e base $\mathsf{B}(f, g) = f + f \times g$, e sobre ele definido o predicado que testa se uma lista está estritamente ordenada,

$$\begin{array}{l} ord\ (Sing\ a) = True \\ ord\ (Add\ (a, x)) = a > maxl\ x \wedge ord\ x \end{array}$$

em que $maxl = (\!|[id\ , \widehat{max}]|\!)$ e $max :: (Ord\ a) \Rightarrow a \to a \to a$ é uma função *standard* em Haskell.

Sendo fácil derivar

$$ord \cdot \mathbf{in} \quad = \quad [true\ , h] \cdot \mathsf{B}(id, \langle maxl, ord \rangle) \tag{4}$$

a partir da definição dada, para $true\ \_ = True$ e $h\ (a, (b, c)) = a > b \wedge c$, mostre que ord se pode implementar da forma mais eficiente

$$ord = \pi_2 \cdot aux\ \mathbf{where}\ aux = (\!|[\langle id, true \rangle\ , \langle \widehat{max} \cdot (id \times \pi_1), h \rangle]|\!)$$

tendo-se ainda que $maxl = \pi_1 \cdot aux$.

Questão 7 A função em Haskell

$sq\ 0 = 0$
$sq\ (n+1) = 2 * n + 1 + sq\ n$

calcula o quadrado de um número, satisfazendo a equação

$$sq \cdot in \quad = \quad [\underline{0}\ , add \cdot \langle odd, sq \rangle] \tag{5}$$

para $in = [\underline{0}\ , succ]$, $\overline{add} = (+)$ e $odd = suc \cdot add \cdot \langle id, id \rangle$. Complete as justificações do seguinte processo de cálculo que exprime sq sob a forma de um hilomorfismo de listas, partindo de (5):

$$
\begin{array}{ll}
 & sq \cdot in = [\underline{0}\ , add \cdot \langle odd, sq \rangle] \\
\equiv & \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \} \\
 & sq \cdot in = [\underline{0}\ , add] \cdot (id + \langle odd, sq \rangle) \\
\equiv & \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \} \\
 & sq \cdot in = [\underline{0}\ , add] \cdot (id + odd \times sq) \cdot (id + \langle id, id \rangle) \\
\equiv & \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \} \\
 & sq \cdot in = [\underline{0}\ , add] \cdot (id + id \times sq) \cdot (id + odd \times id) \cdot (id + \langle id, id \rangle) \\
\equiv & \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \} \\
 & sq = [\underline{0}\ , add] \cdot (\mathsf{F}\ sq) \cdot (id + \langle odd, id \rangle) \cdot out \\
\equiv & \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \} \\
 & sq = [\![\ [\underline{0}\ , add], (id + \langle odd, id \rangle) \cdot out\ ]\!]
\end{array}
$$

Questão 8 Se substituirmos todas as folhas de uma árvore por um mesmo valor k e a seguir calcularmos a maior dessas folhas, com certeza que vamos ter k como resultado. Isto é, tem-se a igualdade

$$maxlt \cdot \mathsf{LTree}\ (\underline{k}) = \underline{k} \tag{6}$$

em que $maxlt = (\![[id\ , \widehat{max}]]\!)$ é a função que calcula a maior folha de uma árvore.

Recorrendo a propriedades de catamorfismos que conhece, entre outras, prove (6).

Questão 9 Como sabe, o operador monádico de *binding* é dado, para qualquer mónade T, pela definição

$$x \gg\!= f \quad \stackrel{\text{def}}{=} \quad (\mu \cdot \mathsf{T}\ f) x$$

que, no caso do mónade das listas, instancia em

$$x \gg\!= f \quad \stackrel{\text{def}}{=} \quad (concat \cdot \mathsf{map}\ f)\ x$$

— para $concat = (\![[nil\ , cat]]\!)$, $nil\ \_ = [\,]$ e $cat\ (x, y) = x +\!\!+ y$ — isto é,

$$(\gg\!= f) \quad = \quad concat \cdot \mathsf{map}\ f \tag{7}$$

Calcule a versão *pointwise* de $(\gg\!=)$ que aparece na bilioteca `Control.Monad.hs`:

$\mathbf{instance}\ Monad\ [\,]\ \mathbf{where}$
$\quad (x : xs) \gg\!= f = f\ x +\!\!+ (xs \gg\!= f)$
$\quad [\,] \gg\!= f = [\,]$

Sugestão: Recorra às leis de catamorfismos que conhece antes de introduzir variáveis.

Questão 10 Os ciclos-**while** típicos da programação imperativa podem exprimir-se funcionalmente como hilomorfismos, como é o caso do seguinte combinador,

$$\begin{aligned} &while :: (b \to Bool) \to (b \to b) \to (b \to c) \to b \to c \\ &while\ p\ f\ g = [\![[id\ , g], ((f + id) \cdot p?)]\!] \end{aligned}$$

que se converte no seguinte Haskell com variáveis:

$$\begin{aligned} &while\ p\ f\ g\ x \\ &\quad \mid \neg\ (p\ x) = g\ x \\ &\quad \mid otherwise = while\ p\ f\ g\ (f\ x) \end{aligned}$$

Defina a função

$$mwhile :: (Monad\ m) \Rightarrow (b \to Bool) \to (b \to m\ b) \to (b \to m\ c) \to b \to m\ c$$

como resultado da "monadificação" de $while$. Jutifique a sua resposta.