Cálculo de Programas

3.º Ano de LCC+LEI+MiEI (Universidade do Minho)
Ano Lectivo de 2025/26

Exame de Recurso — 31 de Janeiro de 2026 , 09h00–11h00
Salas E1-0.08 + E1-0.22

PROVA PRESENCIAL INDIVIDUAL SEM CONSULTA (2h)

Importante — *Ler antes de iniciar a prova:*

- *Esta prova consta de **8** questões que valem, cada uma, 2.5 valores. O tempo médio estimado para resolução de cada questão é de 15 min.*
- *Recomenda-se que os alunos leiam a prova antes de decidirem por que ordem querem responder às questões que são colocadas.*

Questão 1 Considere a função:

$$\beta = \mathsf{swap} \cdot (id \times \mathsf{swap}) \tag{E1}$$

Infira o seu tipo mais geral e formule a sua propriedade natural (grátis) com base no respectivo diagrama, como se explicou nas aulas teóricas.

Questão 2 Demonstre a igualdade

$$p \to (p \to f\ ,\ g)\ ,\ g \quad = \quad p \to f\ ,\ g \tag{E2}$$

com base na lei do condicional de McCarthy que se segue

$$p \to (q \to a\ ,\ b)\ ,\ b \quad = \quad (p \wedge q) \to a\ ,\ b \tag{E3}$$

— que deverá também demonstrar — sabendo que

$$(p \wedge q)? \quad = \quad p \to q?\ ,\ i_2 \tag{E4}$$

é uma propriedade da conjunção de predicados.

Questão 3 Mostre que, para o caso dos naturais (em que $\mathsf{F}\ X = 1 + X$ e $\mathsf{in}_{\mathbb{N}_0} = [\underline{0}, \mathsf{succ}]$), a lei de recursividade mútua toma a forma:

$$\langle f, g\rangle = \mathsf{for}\ \langle \alpha, \beta\rangle\ (a, b) \quad \equiv \quad \begin{cases} f\ 0 = a \\ f\ (n+1) = \alpha\ (f\ n, g\ n) \\ g\ 0 = b \\ g\ (n+1) = \beta\ (f\ n, g\ n) \end{cases}$$

Questão 4 Considere as árvores binárias:

$$\mathsf{T} = \mathsf{BTree}\ A \qquad \begin{cases} \mathsf{F}\ X = 1 + A \times X^2 \\ \mathsf{F}\ f = id + id \times f^2 \end{cases} \qquad \mathsf{in} = [\underline{Empty}, Node]$$

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

Segue-se uma solução para a pergunta que, no teste, pedia uma função $traces$ que calculasse percursos descendentes:

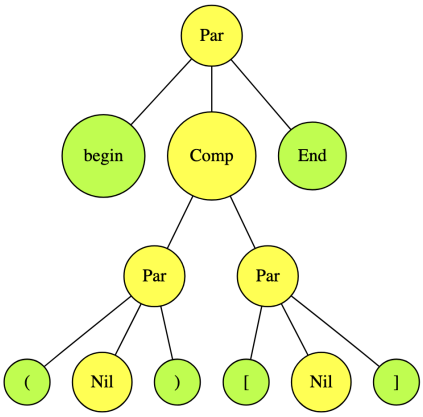
$$\begin{aligned} &traces = (\!|\, [g_1, g_2] \,|\!)\ \mathbf{where} \\ &\quad g_1 = \underline{\mathsf{singl}\ [\,]} \\ &\quad g_2\ (a, (l, r)) = nub\ (\mathsf{map}\ (a:)\ l +\!\!+ \mathsf{map}\ (a:)\ r) \end{aligned}$$

Mostre que, quem por acaso cometer o erro de definir $g_1 = \mathsf{nil}$, obterá $traces = \mathsf{nil}$. **Sugestão**: use a propriedade universal dos catamorfismos.

Questão 5 Em análise de código usam-se as chamadas *árvores de Dyck* para gerir o bom uso de parênteses e de blocos "begin" $\cdots$ "end", etc. Partindo do seguinte tipo recursivo em Haskell,

$$\mathbf{data}\ \mathsf{DyckT}\ a = Nil \mid Comp\ (\mathsf{DyckT}\ a)\ (\mathsf{DyckT}\ a) \mid Par\ a\ (\mathsf{DyckT}\ a)\ a$$

a figura mostra uma dessas árvores.



Identifique, justificando, o functor de base

$$\begin{cases} \mathsf{B}\ (X, Y) = \ldots \\ \mathsf{B}\ (f, g) = ... \end{cases}$$

que capta o padrão de recursividade da declaração de DyckT dada acima, em Haskell, bem como o isomorfismo:

$$\mathsf{in} : \mathsf{B}\ (A, \mathsf{DyckT}\ A) \to \mathsf{DyckT}\ A.$$

Questão 6 Mostre, por absorção-cata, que

$$concatMap\ f = \mathsf{concat} \cdot \mathsf{map}\ f \tag{E5}$$

é a função:

$$\begin{aligned} &concatMap :: (a \to [b]) \to [a] \to [b] \\ &concatMap\ f\ [\,] = [\,] \\ &concatMap\ f\ (h : t) = f\ h +\!\!+ concatMap\ f\ t \end{aligned}$$

NB: recorde que $\mathsf{concat} = (\!|\, [\mathsf{nil}, \mathsf{conc}] \,|\!)$.

Questão 7 A função

$$\theta :: ((a,a) \to a) \to ([a],[a]) \to [a]$$
$$\theta\ f\ ([\,], r) = r$$
$$\theta\ f\ (l, [\,]) = l$$
$$\theta\ f\ (a : l, b : r) = f\ (a,b) : \theta\ f\ (l, r)$$

pode ser vista como o hilomorfismo:

$$\begin{array}{ccc} (A^*)^2 & \xrightarrow{\quad divide \quad} & A^* + A^2 \times (A^*)^2 \\ {\scriptstyle \theta\ f}\downarrow & & \downarrow{\scriptstyle id + id \times \theta\ f} \\ A^* & \xleftarrow[\quad conquer\ f \quad]{} & A^* + A^2 \times A^* \end{array}$$

Infira os genes $divide$ e $conquer$ deste hilomorfismo.

Questão 8 Foi estudado nesta disciplina o chamado *mónade livre* induzido por um qualquer functor G,

$$X \xrightarrow{\mathsf{in} \cdot i_1} \mathsf{T_G}\ X \xleftarrow{(\!|\, [id, \mathsf{in} \cdot i_2] \,|\!)} \mathsf{T_G}\ (\mathsf{T_G}\ X)$$

onde $\mathsf{T_G}\ X$ tem a base $\mathsf{B}\ (X, Y) = X + \mathsf{G}\ Y$, cf:

$$\mathsf{T_G}\ X \underset{\mathsf{in}}{\overset{\mathsf{out} = \mathsf{in}^\circ}{\cong}} X + \mathsf{G}\ (\mathsf{T_G}\ X)$$

$$u = \mathsf{in} \cdot i_1 \tag{E6}$$
$$\mu = (\!|\, [id, \mathsf{in} \cdot i_2] \,|\!) \tag{E7}$$

Considere o seguinte caso particular:

$$\mathsf{G}\ X = 1$$
$$\mathsf{T_G}\ X = Maybe\ X$$
$$\mathsf{in} = [\mathsf{Just}, \underline{\mathsf{Nothing}}]$$

Usando a propriedade universal-cata, calcule:

$$\mu\ (\mathsf{Just}\ x) = x$$
$$\mu\ \mathsf{Nothing} = \mathsf{Nothing}$$
