Cálculo de Programas

3/2.º Ano de LEI/MiEI (Universidade do Minho)
Ano Lectivo de 2023/24

2º Teste — 11 de Dezembro de 2023, 17h00–19h00
Salas 1.03 + 1.01 + 0.20 + 1.13 do Edifício 2.

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 Os tipos $\mathbb{B}$ e $1 + 1$ — onde $1 = \{()\}$ — são isomorfos, podendo a função $\mathsf{out} : \mathbb{B} \to 1 + 1$ ser escrita em Haskell da seguinte maneira:

$$\begin{array}{l} \mathsf{out} : \mathbb{B} \to 1 + 1 \\ \mathsf{out}\ \textsc{False} = i_1\ () \\ \mathsf{out}\ \textsc{True} = i_2\ () \end{array}$$

Apresente uma definição, sem recorrer a variáveis, para a função in (inversa de out), derivada por cálculo analítico a partir da definição dada acima de out.

Questão 2 Considere a função

$$\alpha\ p = \mathsf{swap} \cdot (p \to \pi_1\ ,\ \pi_2)$$

Determine o tipo mais geral de $\alpha\ p$ e, a partir dele, a sua propriedade grátis.

Questão 3 Considere a estrutura de dados que se segue, estudada nas aulas:

Árvores com informação de tipo A nos nós :

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

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

Defina como um catamorfismo a função

$$\begin{array}{l} f : \mathsf{BTree}\ A \to A^* \\ f = (\!|\, g \,|\!) \end{array}$$

isto é, identifique g tal que $f\ t$ seja o caminho a percorrer na árvore t para atingir o seu nó terminal mais à direita.

Questão 4 A função nr (="*no repeats*") que se segue testa se uma lista tem elementos repetidos:

$$
\begin{array}{l}
nr : A^* \to \mathbb{B} \\
nr = \pi_2 \cdot aux \ \mathbf{where} \\
\quad aux = (\!|\, [m\,, \langle n, h \rangle] \,|\!) \\
\quad m\ \_ = ([\,], \textsc{True}) \\
\quad n = \mathsf{cons} \cdot (id \times \pi_1) \\
\quad h\ (a, (t, b)) = \neg\ (a \in t) \wedge b
\end{array}
$$

Resolva em ordem a f e g a equação

$$\langle f\,, g \rangle = aux \tag{E1}$$

entregando essas funções definidas sem recurso a quaisquer combinadores pointfree estudados na disciplina. **Sugestão**: use a lei de recursividade mútua.

Questão 5 Nas aulas teórico-práticas demonstrou-se o seguinte resultado sobre a composição de catamorfismos:

$$(\!|\, g \,|\!) \cdot (\!|\, \mathsf{in} \cdot k \,|\!) = (\!|\, g \cdot m \,|\!) \quad \Leftarrow \quad m \cdot \mathsf{F}\ f = \mathsf{F}\ f \cdot k \tag{E2}$$

Use (E2) para provar a lei de absorção-cata:

$$(\!|\, g \,|\!) \cdot \mathsf{T}\ h = (\!|\, g \cdot \mathsf{B}\ (h, id) \,|\!)$$

NB: recordam-se as leis functoriais estendidas a bifunctores:

$$\mathsf{B}\ (id, id) = id \tag{E3}$$

$$\mathsf{B}\ (h \cdot f, k \cdot g) = \mathsf{B}\ (h, k) \cdot \mathsf{B}\ (f, g) \tag{E4}$$

Questão 6 Considere a função:

$$
\begin{array}{l}
stake\ \_\ [\,] = [\,] \\
stake\ 0\ \_ = [\,] \\
stake\ y\ (x : t) \\
\quad \mid y \equiv x = [x] \\
\quad \mid y < x = [y] \\
\quad \mid y > x = x : stake\ (y - x)\ t
\end{array}
$$

$stake\ x\ y$ dá o maior prefixo de y cuja soma não excede x, truncando se necessário o último elemento. Por exemplo, $stake\ 5\ [1, 2] = [1, 2]$ e $stake\ 5\ [-1, 2, 6] = [-1, 2, 4]$.

Defina $divide$ tal que $\widehat{stake} = [\!(\, divide \,)\!]$ seja um anamorfismo de listas. Apoie a sua resolução num diagrama.

Questão 7 O desenho que se segue descreve a estratégia de programação conhecida por *divide & conquer*:

$$
\begin{tikzcd}
\boxed{\text{Problema}} \arrow[rr, "divide\ \text{(análise)}"] \arrow[d, "solve\ \text{(problema)}"'] & & \boxed{\text{Sub-problemas}} \arrow[d, "solve\ \text{(sub-problemas)}"] \\
\boxed{\text{Solução}} & & \boxed{\text{Sub-soluções}} \arrow[ll, "conquer\ \text{(síntese)}"]
\end{tikzcd}
$$

No Cálculo de Programas, esta estratégia é captada pelo conceito de *hilomorfismo*, definido como a composição

$$solve = \llparenthesis\, conquer\, \rrparenthesis \cdot [\!(\, divide\,)\!] \tag{E5}$$

que se pode demonstrar ser tal que:

$$solve = conquer \cdot (\mathsf{F}\, solve) \cdot divide \qquad \begin{tikzcd} A \arrow[r, "divide"] \arrow[d, "solve"'] & \mathsf{F}A \arrow[d, "\mathsf{F} solve"] \\ B & \mathsf{F}B \arrow[l, "conquer"] \end{tikzcd} \tag{E6}$$

Complete o raciocínio que se segue em que se converte (E5) em (E6):

$$
\begin{array}{ll}
& solve = \llparenthesis\, conquer\, \rrparenthesis \cdot [\!(\, divide\,)\!] \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& solve = conquer \cdot \mathsf{F}\, \llparenthesis\, conquer\, \rrparenthesis \cdot \mathsf{out} \cdot [\!(\, divide\,)\!] \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \vdots \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& solve = conquer \cdot \mathsf{F}\, solve \cdot divide
\end{array}
$$

Questão 8 Em qualquer monad T faz sentido definir a operação que emparelha cada elemento b de um seu habitante t com um determinado valor a:

$$str\ a\ t \quad = \quad do\ \{\ b\ \leftarrow t\ ;\ return(a, b)\ \} \tag{E7}$$

Mostre que (E7) e a definição *pointfree* (E8) que se segue coincidem:

$$str\ a \quad = \quad \mathsf{T}\ \langle \underline{a}, id \rangle \tag{E8}$$

Sugestão: recorde, das aulas práticas, o facto $\mathsf{T}\ f = (u \cdot f) \bullet id$.
