Cálculo de Programas

2.º Ano de LCC+LEI (Universidade do Minho)
Ano Lectivo de 2014/15

Exame de Recurso — 7 de Julho de 2015
16h00
Edifício da cantina de Gualtar

---

*Este teste consta de **10** questões que valem, cada uma, 2 valores. O tempo médio estimado para resolução de cada questão é de 15 min.*

PROVA SEM CONSULTA (2h30m)

Questão 1 Dadas as funções

$$f = [id + i_1 \, , i_2 \cdot i_2] \tag{E1}$$

e

$$f^\circ = [i_1 \cdot i_1 \, , i_2 + id] \tag{E2}$$

identifique o isomorfismo que estas funções estabelecem, desenhando o diagrama respectivo.

Questão 2 Considere a propriedade "grátis" de um isomorfismo natural iso

$$((f + g) \times k) \cdot \mathsf{iso} \quad = \quad \mathsf{iso} \cdot (k \times (g + f)) \tag{E3}$$

Mostre a partir do respectivo diagrama que o tipo mais geral de iso é o mesmo que o da expressão $\mathsf{swap} \cdot (id \times \mathsf{coswap})$, em que swap e coswap são funções que conhece.

Questão 3 Demonstre a seguinte propriedade do combinador condicional de McCarthy

$$p \rightarrow (p \rightarrow a, b), (p \rightarrow c, d) \quad = \quad p \rightarrow a, d \tag{E4}$$

sabendo que

$$(p? + p?) \cdot p? \quad = \quad (i_1 + i_2) \cdot p? \tag{E5}$$

se verifica.

Questão 4 Pode mostrar-se que a seguinte variante do tipo "rose tree"

$$\mathbf{data}\ Rose\ a = L\ a \mid R\ [Rose\ a]$$

forma um mónade, tendo por base $\mathsf{B}\ (f, g) = f + \mathsf{map}\ g$. Construa as funções in / out para este tipo e desenhe a diagrama dos seus catamorfismos. Com base nesse diagrama, converta para Haskell com variáveis a função que a seguir se define:

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

Questão 5 A função

$$
\begin{array}{l}
f : [A + B] \to [B] \\
f\ [\,] = [\,] \\
f\ (i_1\ a : x) = [\,] \\
f\ (i_2\ b : x) = b : f\ x
\end{array}
$$

é uma espécie de $takeWhile$: copia para a saída todos os objectos do tipo B até à ocorrência do primeiro de tipo A, por exemplo: $f\ [i_2\ 3, i_1\ \texttt{'x'}, i_2\ 4] = [3]$.

Se quisermos contar quantos B são copiados podemos escrever a função

$$
\begin{array}{l}
g : [A + B] \to \mathbb{N}_0 \\
g\ [\,] = 0 \\
g\ (i_1\ a : x) = 0 \\
g\ (i_2\ b : x) = 1 + g\ x
\end{array}
$$

ou correr a expressão $\mathsf{length} \cdot f$. Mostre, por fusão-cata, que $\mathsf{length} \cdot f = g$ g que as duas funções são os catamorfismos:

$$f = (\![[\mathsf{nil}\ , \mathsf{in}] \cdot (id + \mathsf{distl})]\!) \tag{E7}$$

$$g = (\![([\mathsf{zero}\ , \mathsf{in}_{\mathbb{N}_0} \cdot (id + \pi_2)] \cdot (id + \mathsf{distl}))]\!) \tag{E8}$$

onde $\mathsf{in}_{\mathbb{N}_0} = [\mathsf{zero}\ , \mathsf{succ}]$, $\mathsf{in} = [\mathsf{nil}\ , \mathsf{cons}]$ e $(A \times C) + (B \times C) \xleftarrow{\mathsf{distl}} (A + B) \times C$ é um isomorfismo que conhece.

Sugestões: recorde que $\mathsf{length} = (\![\mathsf{in}_{\mathbb{N}_0} \cdot (id + \pi_2)]\!)$ e não desperdice a propriedade *grátis* de distl.

Questão 6 A seguinte função

$$
\begin{array}{l}
pascal = \mathsf{for}\ loop\ [1] \\
\quad \mathbf{where}\ loop\ x = 1 : \mathsf{soma}\ (x, \mathsf{tail}\ x)
\end{array}
$$

calcula a $n + 1$-ésima linha do triângulo de Pascal, onde

$$
\begin{array}{l}
\mathsf{soma}\ ([\,], x) = x \\
\mathsf{soma}\ (x, [\,]) = x \\
\mathsf{soma}\ ((a : as), (b : bs)) = (a + b) : (\mathsf{soma}\ (as, bs))
\end{array}
$$

Por exemplo, $pascal\ 0$ é a lista $[1]$, $pascal\ 2$ é a lista $[1, 2, 1]$, *etc.*, *cf.*

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5  10  10   5   1
```

Identifique o *gene* da função soma quando escrita sob a forma de um *anamorfismo* do tipo SList que foi assunto do trabalho prático, recordando:

$$\mathbf{data}\ \mathsf{SList}\ a\ b = Sent\ b \mid Cons\ (a, \mathsf{SList}\ a\ b)$$

Questão 7 Para qualquer tipo de dados paramétrico $\mathsf{T}\ X$ é definível a função

$\mathsf{shape} = \mathsf{T}\ !$

que calcula a *forma* dos seus habitantes — e.g. árvores — isto é, deita fora toda a informação útil através da função $! : A \to 1$ mas mantém a topologia da estrutura. Ora verifica-se que a *"forma da forma é a mesma forma"*, isto é

$$\mathsf{shape} \cdot \mathsf{shape} = \mathsf{shape} \tag{E9}$$

Demonstre esta igualdade para o caso $\mathsf{T} = \mathsf{BTree}$. **NB:** Sugere-se o recurso directo à lei de absorção-cata e recorda-se que a base do tipo BTree é $\mathsf{B}\ (f, g) = id + f \times (g \times g)$.

Questão 8 Considere o catamorfismo $\mathsf{LTree}\ (A \times B) \xrightarrow{unzp} (\mathsf{LTree}\ A) \times (\mathsf{LTree}\ B)$ que divide uma árvore de pares num par de árvores

$$\begin{array}{l} unzp = (\!|\langle \mathsf{in}_1 \cdot (\mathsf{F}\ \pi_1), \mathsf{in}_2 \cdot (\mathsf{F}\ \pi_2)\rangle|\!)\ \mathbf{where} \\ \quad \mathsf{in}_1 = \mathsf{in} \cdot \mathsf{B}\ (\pi_1, id) \\ \quad \mathsf{in}_2 = \mathsf{in} \cdot \mathsf{B}\ (\pi_2, id) \end{array}$$

onde, como sabe, $\mathsf{B}\ (f, g) = f + g \times g$. Recorra a uma lei que conhece (e cujo nome é bastante sugestivo) para demonstrar a seguinte propriedade de cancelamento:

$$\pi_1 \cdot unzp \quad = \quad \mathsf{LTree}\ \pi_1 \tag{E10}$$

Questão 9 Apresente justificações para o cálculo da igualdade

$$\underline{g} = \overline{g \cdot \pi_2}$$

envolvendo exponenciais e funções constantes, onde $\mathsf{exp}\ f$ abrevia $\overline{f \cdot \mathsf{ap}}$:

$$\begin{array}{ll} & \underline{g} = \overline{g \cdot \pi_2} \\ \equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \} \\ & \underline{g} = \mathsf{exp}\ g \cdot \underline{id} \\ \equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \} \\ & \underline{g} = \underline{g \cdot id} \\ \equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \} \\ & true \\ \Box & \end{array}$$

Sugestão: Os factos seguintes, estudados nas aulas, podem ser-lhe úteis nas justificações acima:

$$\overline{\pi_2} = \underline{id} \tag{E11}$$

$$\mathsf{exp}\ f\ \underline{g} = \underline{f \cdot g} \tag{E12}$$

Questão 10 O tipo

$\mathbf{data}\ \mathsf{Error}\ a = Err\ String \mid Ok\ a$

que podemos usar para gerir mensagens de erro, mostra-se facilmente ser um functor definindo

$$\mathsf{Error}\ f = \mathsf{in} \cdot (id + f) \cdot \mathsf{out} \tag{E13}$$

onde $\mathsf{in} = [Err\ , Ok]$ e $\mathsf{out} = \mathsf{in}^{\circ}$. O tipo Error forma, ainda, um mónade

$$X \xrightarrow{Ok} \mathsf{Error}\ X \xleftarrow{\mu} \mathsf{Error}\ (\mathsf{Error}\ X)$$

onde $\mu = [\mathsf{in} \cdot i_1\ , id] \cdot \mathsf{out}$, isto é

$$\begin{array}{l}
\mu :: \mathsf{Error}\ (\mathsf{Error}\ a) \rightarrow \mathsf{Error}\ a \\
\mu\ (Err\ s) = Err\ s \\
\mu\ (Ok\ a) = a
\end{array}$$

em sintaxe Haskell.

1. Justifique o cálculo que se segue da derivação da definição de μ:

$$\begin{tikzcd}
\mathsf{Error}\ X & S + X \arrow[l, "\mathsf{in}"'] & S + (S + X) \arrow[l, "{[i_1\ , id]}"'] & \mathsf{Error}\ (S + X) \arrow[l, "\mathsf{out}"'] & \mathsf{Error}\ (\mathsf{Error}\ X) \arrow[l, "\mathsf{Error\ out}"'] \arrow[llll, bend left, "\mu"]
\end{tikzcd}$$

   onde S abrevia $String$:

$$\begin{array}{ll}
 & \mu = \mathsf{in} \cdot [i_1\ , id] \cdot \mathsf{out} \cdot (\mathsf{Error\ out}) \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & \mu = [\mathsf{in} \cdot i_1\ , \mathsf{in}] \cdot (id + \mathsf{out}) \cdot \mathsf{out} \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & \mu = [\mathsf{in} \cdot i_1\ , id] \cdot \mathsf{out} \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & \left\{ \begin{array}{l} \mu \cdot Err = Err \\ \mu \cdot Ok = id \end{array} \right.
\end{array}$$

2. Mostre que, para $\mathsf{T} = \mathsf{Error}$, a lei monádica $\mu \cdot (\mathsf{T}\ u) = id$ se verifica.
