Cálculo de Programas (8504N1)

2.º Ano de LCC (Lic. em Ciências da Computação da Universidade do Minho)
Ano Lectivo de 2008/09

Exame de recurso — 17 de Julho 2009
14h00
Sala 1201

NB*: Esta prova consta de **8** questões que valem, cada uma, 2.5 valores.*

PROVA SEM CONSULTA (2 horas)

GRUPO I

Questão 1 O combinador

```
const :: a -> b -> a
const a b = a
```

está disponível em Haskell para construir funções constantes, sendo habitual designarmos `const k` por $\underline{k}$, qualquer que seja k. Dão-se de seguida duas proprieades deste combinador:

$$\underline{k} \cdot g = \underline{k} \quad (1)$$
$$f \cdot \underline{k} = \underline{f\ k} \quad (2)$$

Demonstre a propriedade

$$\underline{(b, a)} = \langle \underline{b}, \underline{a} \rangle \quad (3)$$

a partir da propriedade universal do produto e das propriedades das funções constantes acima indicadas.

Questão 2 Demonstre as duas leis de fusão do condicional de McCarthy que conhece,

$$f \cdot (p \to g, h) = p \to f \cdot g, f \cdot h \quad (4)$$
$$(p \to f, g) \cdot h = (p \cdot h) \to (f \cdot h), (g \cdot h) \quad (5)$$

GRUPO II

Questão 3 Se multiplicar por um factor k o resultado do ciclo $for\ (a+)\ 0$, obterá o mesmo resultado que o do ciclo $for\ (ka+)\ 0$. Mostre que, de facto, a igualdade

$$(k*) \cdot (for\ (a+)\ 0) = for\ (k * a+)\ 0 \quad (6)$$

se verifica, usando uma lei de fusão que seja adequada ao problema.

Questão 4 Recorde a definição *pointfree* da função factorial:

$$fac = g \cdot (id + \langle id, fac \rangle) \cdot outNat$$
$$\textbf{where}\ g = [\underline{1}\ , \widehat{(*)} \cdot (succ \times id)]$$

Funções como esta, que fazem recursividade mútua com a função $id = (\!|in|\!)$, dizem-se *paramorfismos*, e são um caso particular dos mutumorfismos que foram assunto de uma questão do trabalho teorico-prático desta disciplina.

Nas aulas estudou-se como linearizar este tipo de funções, com recurso à lei de Fokkinga. Um outro caminho para lidar com esta classe de funções é transformá-las em hilomorfismos. Nesse sentido, complete o cálculo que se segue e que mostra como fac acima se converte no hilomorfismo

$$\begin{array}{l} fac = hylo\ g\ ((id + \langle id, id\rangle) \cdot outNat) \\ \quad \textbf{where}\ g = [\underline{1}\ , \widehat{(*)} \cdot (succ \times id)] \end{array}$$

que conhece da biblioteca `List.hs`. Generaliza-se fac a qualquer paramorfismo f de gene g e padrão de recursividade F:

$$\begin{array}{ll} & f = g \cdot \mathsf{F}\ \langle id, f\rangle \cdot out \\ = & \quad \{\ \ldots\ \} \\ & f = g \cdot \mathsf{F}\ ((id \times f) \cdot \langle id, id\rangle) \cdot out \\ = & \quad \{\ \ldots\ \} \\ & f = g \cdot \mathsf{F}\ (id \times f) \cdot \mathsf{F}\ \langle id, id\rangle \cdot out \\ = & \quad \{\ \ldots\ \} \\ & f = g \cdot (\mathsf{G}\ f) \cdot \mathsf{F}\ \langle id, id\rangle \cdot out \\ = & \quad \{\ \ldots\ \} \\ & f = [\!|\ g, \mathsf{F}\ \langle id, id\rangle \cdot out\ |\!] \\ = & \quad \{\ \ldots\ \} \\ & f = (\!|g|\!) \cdot [\!(\mathsf{F}\ \langle id, id\rangle \cdot out)|\!] \end{array}$$

Questão 5 Considere o seguinte tipo de dados paramétrico, em Haskell,

$$\textbf{data}\ MwTree\ a = Leaf\ a \mid Nest\ [MwTree\ a]$$

que nos permite escrever árvores de grau arbitrário ("*multi-way* trees") que generalizam as do tipo $LTree$.

1. Defina as funções $inMwTree$, $outMwTree$, $baseMwTree$, $recMwTree$ e $cataMwTree$ que deverão fazer parte da biblioteca a construir à volta do tipo $MwTree$ e use-as para declarar $MwTree$ como instância da

   ```
   class Functor t where
       fmap :: (a -> b) -> (t a -> t b)
   ```

2. Se tentar interpretar a expressão $cataMwTree\ [return\ , join]$, verá que ela está bem definida e exibe o tipo

   $$cataMwTree\ [return\ , join] :: MwTree\ a \to [a]$$

   Que funções são afinal $return$ e $join$, neste contexto? Acompanhe a sua resposta de um diagrama explicativo. E que faz a função $cataMwTree\ [return\ , join]$, afinal? Explique-o através de um exemplo.

3. Se se lembrar da questão 10 do trabalho teorico-prático da disciplina concerteza que identifica $MwTree$ como um tipo que instancia a classe mónade, para $return = inMwTree \cdot i_1$ e $join = cataMwTree\ [id\ , inMwTree \cdot i_2]$. Usando a propriedade universal de catamorfismos, calcule a versão *pointwise* de $join$.

GRUPO III

Questão 6 Analise a seguinte função em Haskell

$$k\ f\ l = [y \mid a \leftarrow l, y \leftarrow [f\ a]]$$

e responda às seguintes questões:

- Qual o tipo que um interpretador de Haskell devolve para a função k? E o que faz esta função?
- Escreva k generalizada a qualquer mónade, usando notação $\textbf{do}\ \{\ldots\}$. Não se esqueça de incluir o novo tipo da função.
