Cálculo de Programas

2.º Ano de LCC (8504N1)
Ano Lectivo de 2007/08

Prova de avaliação individual — 25 de Junho 2008
14h30
Salas 2209 e 2210

NB: *Esta prova consta de **8** questões que valem, cada uma, 2.5 valores. Por favor utilize folhas de resposta diferentes para cada grupo.*

PROVA SEM CONSULTA (2 horas)

GRUPO I

Questão 1 Considere o tipo seguinte que define árvores binárias completas (isto é, com folhas — um misto de $BTree$ e de $LTree$):

$$\mathbf{data}\ FTree\ a\ c = Unit\ c \mid Comp\ a\ (FTree\ a\ c, FTree\ a\ c)$$

Usando o algoritmo de Hindley-Milner para inferência de tipos polimórficos estudado nas aulas práticas, deduza o tipo principal (ie. mais geral) da função

$$foldFTree\ f\ g\ (Unit\ c) = f\ c$$
$$foldFTree\ f\ g\ (Comp\ a\ (l, r)) = g\ a\ (foldFTree\ f\ g\ l, foldFTree\ f\ g\ r)$$

Sugestão: comece por abreviar $foldFTree\ f\ g$ em k, infira o tipo de k e deduza o de $foldFTree$ a partir deste.

Questão 2 Defina as funções $inFTree$, $outFTree$, $baseFTree$ e $cataFTree$ que fazem parte da biblioteca a construir à volta do tipo $FTree$ da questão anterior e use-as para completar a seguinte declaração desse tipo como instância da classe $BiFunctor$:

```
instance BiFunctor FTree
         where bmap f g = cataFTree ( ......................... )
```

NB: Recorde a declaração

```
class BiFunctor f where
      bmap :: (a -> b) -> (c -> d) -> (f a c -> f b d)
```

que consta da biblioteca $Cp.hs$.

Questão 3 A lei de recursividade mútua generaliza a mais do que duas funções mutuamente recursivas, por exemplo a três:

$$\left\{\begin{array}{l} f \cdot in = h \cdot \mathsf{F}\,\langle f, \langle g, j\rangle\rangle \\ g \cdot in = k \cdot \mathsf{F}\,\langle f, \langle g, j\rangle\rangle \\ j \cdot in = l \cdot \mathsf{F}\,\langle f, \langle g, j\rangle\rangle \end{array}\right. \equiv \langle f, \langle g, j\rangle\rangle = (\!|\langle h, \langle k, l\rangle\rangle|\!) \tag{1}$$

Justifique detalhadamente os passos do seguinte cálculo dessa versão da lei:

$$\begin{array}{cl} & \langle f, \langle g, j\rangle\rangle = (\!|\langle h, \langle k, l\rangle\rangle|\!) \\ \equiv & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\ & \langle f, \langle g, j\rangle\rangle \cdot in = \langle h, \langle k, l\rangle\rangle \cdot \mathsf{F}\,\langle f, \langle g, j\rangle\rangle \\ \equiv & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\ & \langle f \cdot in, \langle g, j\rangle \cdot in\rangle = \langle h \cdot \mathsf{F}\,\langle f, \langle g, j\rangle\rangle, \langle k, l\rangle \cdot \mathsf{F}\,\langle f, \langle g, j\rangle\rangle\rangle \end{array}$$

$$\equiv \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \}$$

$$\left\{ \begin{array}{l} f \cdot in = h \cdot \mathsf{F}\,\langle f, \langle g, j \rangle \rangle \\ \langle g, j \rangle \cdot in = \langle k, l \rangle \cdot \mathsf{F}\,\langle f, \langle g, j \rangle \rangle \end{array} \right.$$

$$\equiv \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \}$$

$$\left\{ \begin{array}{l} f \cdot in = h \cdot \mathsf{F}\,\langle f, \langle g, j \rangle \rangle \\ g \cdot in = k \cdot \mathsf{F}\,\langle f, \langle g, j \rangle \rangle \\ j \cdot in = l \cdot \mathsf{F}\,\langle f, \langle g, j \rangle \rangle \end{array} \right.$$

Questão 4 O $Prelude$ do Haskell inclui a definição da função

$$\begin{array}{l} \underline{\cdot} :: c \to b \to c \\ \underline{k}\ \_ = k \end{array}$$

que permite construir funções constantes. O diagrama

$$\begin{tikzcd} C \arrow[r, "const"] \arrow[d, "f"'] & C^B \arrow[d, "\ldots"] \\ A \arrow[r, "const"'] & \cdots \end{tikzcd} \tag{2}$$

esboça a propriedade *"grátis"* desta função. Complete o diagrama e mostre que a lei que ele desenha se pode escrever da forma

$$f \cdot \underline{c} \quad = \quad \underline{f\ c} \tag{3}$$

usando (como foi hábito ao longo da disciplina) a abreviatura $\underline{k}$ para $\underline{k}$.

GRUPO II

Questão 5 Considere a definição, em Haskell, da função que junta (concatena) duas sequências:

$$\begin{array}{l} [\,] +\!\!+\, l = l \\ (h : t) +\!\!+\, l = h : (t +\!\!+\, l) \end{array}$$

Pretendendo-se demonstrar a propriedade associativa desta operação,

$$(a +\!\!+\, b) +\!\!+\, c \quad = \quad a +\!\!+\, (b +\!\!+\, c)$$

representa-se a referida função sob a forma de um catamorfismo parametrizado por um dos seus argumentos,

$$(+\!\!+ x) \quad = \quad \llparenthesis \underbrace{[\underline{x}\,, \widehat{(:)}]}_{g_x} \rrparenthesis \tag{4}$$

e a referida propriedade sob a forma

$$(+\!\!+ c) \cdot (+\!\!+ b) \quad = \quad (+\!\!+ (b +\!\!+\, c)) \tag{5}$$

Justifique detalhadamente os passos da seguinte prova de (5):

$$(+\!\!+ c) \cdot (+\!\!+ b) \quad = \quad (+\!\!+ (b +\!\!+\, c))$$

$$\equiv \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \}$$

$$(+\!\!+ c) \cdot \llparenthesis g_b \rrparenthesis \quad = \quad \llparenthesis g_{b +\!\!+ c} \rrparenthesis$$

$$\Leftarrow \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \}$$

$$(+\!\!+ c) \cdot g_b \quad = \quad g_{b +\!\!+ c} \cdot (id + id \times (+\!\!+ c))$$

$$\equiv \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \}$$

$$(+\!\!+ c) \cdot [\underline{b}\,, in \cdot i_2] \quad = \quad [\underline{b +\!\!+\, c}\,, in \cdot i_2] \cdot (id + id \times (+\!\!+ c))$$

$$
\begin{array}{ll}
\equiv & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& (\mathbin{+\!\!+} c) \cdot \underline{b} = \underline{b \mathbin{+\!\!+} c} \quad \wedge \quad (\mathbin{+\!\!+} c) \cdot in \cdot i_2 = in \cdot i_2 \cdot (id \times (\mathbin{+\!\!+} c)) \\
\equiv & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& g_c \cdot (id + id \times (\mathbin{+\!\!+} c)) \cdot i_2 \quad = \quad in \cdot i_2 \cdot (id \times (\mathbin{+\!\!+} c)) \\
\equiv & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& g_c \cdot i_2 \cdot (id \times (\mathbin{+\!\!+} c)) \quad = \quad in \cdot i_2 \cdot (id \times (\mathbin{+\!\!+} c)) \\
\Leftarrow & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& g_c \cdot i_2 \quad = \quad in \cdot i_2 \\
\equiv & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \widehat{(:)} \quad = \quad \widehat{(:)} \\
\equiv & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \textsc{True}
\end{array}
$$

Questão 6 A função que calcula quadrados perfeitos

$$
\begin{array}{l}
sq\ 0 = 0 \\
sq\ (n+1) = 2 * n + 1 + sq\ n
\end{array}
$$

foi apresentada nesta disciplina como um hilomorfismo de listas. Contudo, ela pode exprimir-se indirectamente através de um catamorfismo de números naturais se inventarmos a função $impar\ \ n \stackrel{\mathrm{def}}{=} 2 * n + 1$, calcularmos para esta última as cláusulas (óbvias)

$$
\begin{array}{l}
impar\ 0 = 1 \\
impar\ (n+1) = 2 + impar\ n
\end{array}
$$

e redefinirmos sq de forma a depender mutuamente de $impar$:

$$
\begin{array}{l}
sq'\ 0 = 0 \\
sq'\ (n+1) = impar\ n + sq'\ n
\end{array}
$$

Demonstre, por aplicação da lei de recursividade mútua (para $\mathsf{F}\, g = id + g$), que sq, sq' e a função que se segue

$$
\begin{array}{l}
sq''\ n = \mathbf{let}\ (a, b) = aux\ n\ \mathbf{in}\ a\ \mathbf{where} \\
\quad aux\ 0 = (0, 1) \\
\quad aux\ (n+1) = \mathbf{let}\ (a, b) = aux\ n\ \mathbf{in}\ (a + b, 2 + b)
\end{array}
$$

são a mesma função.

Questão 7 Recorde o tipo de dados que é central à biblioteca $LTree.hs$:

```
data LTree a = Leaf a | Fork (LTree a, LTree a) deriving (Show, Eq)
```

Sabendo que

$$
foldLTree\ f\ g \quad = \quad cataLTree\ [f\ , \widehat{g}] \tag{6}
$$

escreva a definição com variáveis de $foldLTree\ f\ g$ e, a partir desta última, a sua variante monádica, de tipo

$$
foldLTreeM :: (Monad\ m) \Rightarrow (a \to b) \to (b \to b \to b) \to LTree\ a \to m\ b
$$

NB: recorde que $\widehat{g}$ abrevia $uncurry\ g$.

Questão 8 Com base nas definições e propriedades dos operadores monádicos $\gg\!=$ e $\bullet$ que conhece, demonstre a igualdade

$$
x \gg\!= (f \bullet g) \quad = \quad (x \gg\!= g) \gg\!= f \tag{7}
$$
