Cálculo de Programas

2.º Ano de LCC (8504N1)
Ano Lectivo de 2006/07

Exame (1.ª chamada da época normal) de Junho 2007
14h00
Salas 2303, 2304

NB: *Esta prova consta de **8** alíneas que valem, cada uma, 2.5 valores. Utilize folhas de resposta diferentes para cada grupo.*

PROVA SEM CONSULTA (2 horas)

GRUPO I

Questão 1 Dadas as funções

$$f = [id + i_1 \ , i_2 \cdot i_2] \tag{1}$$

e

$$f^\circ = [i_1 \cdot i_1 \ , i_2 + id] \tag{2}$$

identifique o isomorfismo que estas funções estabelecem (desenhando o diagrama respectivo) e derive a versão *pointwise* de f.

Questão 2 Demonstre a seguinte propriedade do combinador condicional de McCarthy

$$f \times (p \rightarrow g, h) \quad = \quad p \cdot \pi_2 \rightarrow f \times g, f \times h \tag{3}$$

sabendo que

$$\langle f, (p \rightarrow q, h) \rangle \quad = \quad p \rightarrow \langle f, q \rangle, \langle h, f \rangle \tag{4}$$

se verifica.

Questão 3 Faça o diagrama de um hilomorfismo com as seguintes características:

- a sua estrutura de dados virtual (intermédia) é uma árvore binária de procura
- o catamorfismo que ele inclui conta o número de nós do seu argumento
- o anamorfismo que ele inclui é o mesmo do algoritmo "quick sort".

Explicite os genes do hilomorfismo que desenhou e diga, sumariamente, que função é que ele implementa.

GRUPO II

Questão 4 Derive a versão *pointwise* em Haskell da função f caracterizada pela seguinte equação

$$f \cdot [\underline{0}, succ] \quad = \quad [\langle \underline{0}, \underline{1} \rangle, \langle \pi_2, uncurry(+) \rangle] \cdot (id + f)$$

Mostre que f é um catamorfismo e desenhe o respectivo diagrama.

Questão 5 Uma das operações conhecidas sobre listas é a da inversão

```
invl [] = []
invl (a:l) = (invl l) ++ [a]
```

que facilmente se mostra ser o catamorfismo

$$invl \overset{\text{def}}{=} (\!|\underbrace{[nil\ , uconc \cdot swap \cdot (singl \times id)]}_{g}|\!) \tag{5}$$

onde $nil = \underline{[\,]}$, $singl\ a = [a]$ e $uconc = uncurry(++)$. Partindo das propriedades

$$invl \cdot uconc \quad = \quad uconc \cdot (invl \times invl) \cdot swap \tag{6}$$
$$invl \cdot singl \quad = \quad singl \tag{7}$$
$$uconc \cdot (singl \times id) \quad = \quad cons \tag{8}$$

em que $cons(a, l) = a : l$, complete as justificações da seguinte prova da propriedade involutiva de $invl$:

$$
\begin{array}{cl}
 & invl \cdot invl = id \\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \} \\
 & invl \cdot (\!|g|\!) = (\!|in|\!) \\
\Leftarrow & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \} \\
 & invl \cdot g = in \cdot (id + id \times invl) \\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \} \\
 & invl \cdot [nil\ , uconc \cdot swap \cdot (singl \times id)] = in \cdot (id + id \times invl) \\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \} \\
 & [invl \cdot nil\ , invl \cdot uconc \cdot swap \cdot (singl \times id)] = in \cdot (id + id \times invl) \\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \} \\
 & [nil\ , uconc \cdot (invl \times invl) \cdot swap \cdot swap \cdot (singl \times id)] = in \cdot (id + id \times invl) \\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \} \\
 & [nil\ , uconc \cdot (invl \cdot singl \times invl)] = in \cdot (id + id \times invl) \\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \} \\
 & [nil\ , uconc \cdot (singl \times invl)] = in \cdot (id + id \times invl) \\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \} \\
 & [nil\ , cons \cdot (id \times invl)] = in \cdot (id + id \times invl) \\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \} \\
 & [nil\ , cons] \cdot (id + id \times invl) = in \cdot (id + id \times invl) \\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \} \\
 & \textsc{True}
\end{array}
$$

Questão 6 As estruturas de dados recursivas (vulg. árvores) que linguagens como o Haskell admitem são traduzidas para estruturas em memória RAM usando *heaps*. Um *heap* é um *array* asssociativo: em cada célula de memória que ocupa, associa a um endereço um nó da estrutura de dados que está a armazenar, expressa em termos de endereços (vulg. *apontadores*). Assim, basta ter um endereço e um *heap* para ser possível reconstituir a árvore que ele representa.

Por exemplo, um *heap* para árvores do tipo

```
data LTree a = Leaf a | Fork (LTree a, LTree a)
```

poderá ser de tipo

```
data Heap a k = Heap [(k,(Either a (k, k)))] k
```

se se representar por uma lista de pares a associação entre apontadores (k) e nós ($a + k \times k$) envolvendo apontadores. Assim, a árvore

```
t = Fork(Leaf "azul",Fork(Leaf "verde",Leaf "amarelo"))
```

por exemplo, poderá representada pelo *heap*

```
h = Heap [(1,Right (2,3)),
          (2,Left "azul"),
          (3,Right (5,7)),
          (5,Left "verde"),
          (7,Left "amarelo")]
         1
```

Defina sob a forma de um anamorfismo de tipo $LTree$ a função

```
absf :: (Eq k) => Heap a k -> LTree a
```

que, dado um *heap*, reconstitui a árvore que ele representa.

GRUPO III

Questão 7 Nesta disciplina foram estudadas três mónades: $Maybe$, listas e estado. Uma outra estrutura paramétrica que forma uma mónade é $LTree$. Para isso, bastará definir, com sabe, os operadores

$$\mu \quad : \quad LTree(LTree\ A) \longrightarrow LTree\ A \tag{9}$$

$$u \quad : \quad A \longrightarrow LTree\ A \tag{10}$$

e mostrar que eles satisfazem duas propriedades.

Concretize os operadores μ e u acima, tendo cuidado em mostrar que o primeiro é um catamorfismo (por exemplo através de um diagrama).

Questão 8 Complete a demonstração que se segue do facto

$$(\mathsf{T}\ f)\ x = do\ \{\ a\ \leftarrow x\ ;\ return(f\ a)\ \}$$

válido para toda a mónade T:

$$
\begin{array}{ll}
& do\,\{\, a\ \leftarrow x\,;\, return(f\ a)\,\} \\
= & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& x >\!\!>\!\!= \lambda a.\, return(f\ a) \\
= & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& x >\!\!>\!\!= (return \cdot f) \\
= & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& (\mu \cdot \mathsf{T}\ (return \cdot f))\ x \\
= & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& (\mu \cdot (\mathsf{T}\ return) \cdot (\mathsf{T}\ f))\ x \\
= & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& (id \cdot (\mathsf{T}\ f))x \\
= & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& (\mathsf{T}\ f)x
\end{array}
$$