Cálculo de Programas
Teste

18 Junho de 2008, 18h00

O teste tem a duração de 1h45m. Todas as perguntas valem 3 pontos, com excepção da última questão que vale apenas 2.

Questão 1 Identifique o isomorfismo que a seguinte função testemunha, desenhando o diagrama respectivo.

$$\mathsf{iso} = (\mathsf{bang} + \mathsf{bang}) \triangle (\mathsf{id} \triangledown \mathsf{id})$$

Defina esta função em Haskell no estilo *point-wise*. Demonstre a seguinte propriedade da função iso.

$$(\mathsf{id} \times f) \circ \mathsf{iso} = \mathsf{iso} \circ (f + f)$$

Justifique todos os cálculos que efectuar.

Questão 2 Considere o seguinte tipo de dados.

$$\mathbf{data}\ \mathsf{LTree}\ a = \mathsf{Leaf}\ a \mid \mathsf{Fork}\ (\mathsf{LTree}\ a, \mathsf{LTree}\ a)$$

Defina as funções out_{LT} em Haskell no estilo *point-wise* e in_{LT} no estilo *point-free*. Desenhe o diagrama dos catamorfismos para este tipo, e identifique a respeciva lei universal. Codifique o catamorfismo em Haskell no estilo *point-wise*.

Questão 3 Considere a seguinte função sobre árvores.

$$\begin{aligned}&\mathsf{join} :: \mathsf{LTree}\ (\mathsf{LTree}\ a) \to \mathsf{LTree}\ a\\ &\mathsf{join}\ (\mathsf{Leaf}\ x) \quad = x\\ &\mathsf{join}\ (\mathsf{Fork}\ (l, r)) = \mathsf{Fork}\ (\mathsf{join}\ l, \mathsf{join}\ r)\end{aligned}$$

Derive uma versão *point-free* de join usando um catamorfismo. Justifique todos os cálculos que efectuar.

Questão 4 Pretende-se implementar a função $\mathsf{gera} :: a \to \mathsf{Int} \to \mathsf{LTree}\ a$ que, dado um elemento x e um inteiro $n > 0$, gera uma árvore com n folhas todas com o valor x. Defina o padrão de recursividade *unfold* para o tipo LTree a e implemente gera x usando o *unfold*.

Questão 5 Demonstre a seguinte propriedade da função unzip.

$$\mathsf{fst} \circ \mathsf{unzip} = \mathsf{map}\ \mathsf{fst}$$

Considere as seguintes definições para as funções unzip e map f.

$$\begin{aligned}\mathsf{unzip} &= (\!|(\underline{[\,]} \triangle \underline{[\,]}) \triangledown (\mathsf{cons} \circ (\mathsf{fst} \times \mathsf{fst}) \triangle \mathsf{cons} \circ (\mathsf{snd} \times \mathsf{snd}))|\!)_L\\ \mathsf{map}\ f &= (\!|\mathsf{in}_L \circ (\mathsf{id} + f \times \mathsf{id})|\!)_L\end{aligned}$$

Justifique todos os cálculos que efectuar.

Questão 6 Para contabilizar o número que participações que cada aluno teve nas aulas práticas precisamos da seguinte função.

```
cadastro :: [Int] → [(Int, Int)]
```

Dada uma lista com todas as participações (cada ocorrência de um número de aluno nesta lista corresponde a uma ida ao quadro), a função cadastro gera uma lista que associa cada aluno ao número de vezes que foi ao quadro. Implemente esta função usando o *monad* estado.

Questão 7 À semelhança das listas, é possível ver o tipo LTree *a* como um *monad* que pode conter vários valores do tipo *a*. Implemente a respectiva instância da classe Monad.

Boa sorte!