Cálculo de Programas

2.° ano das Licenciaturas em
Engenharia Informática e Ciências da Computação
UNIVERSIDADE DO MINHO

2012/13 - Ficha nr.° 5

1. Formule a lei natural ("grátis") da função

   $$pwnil :: a \to (a, ())$$
   $$pwnil = \langle id, ! \rangle$$

   (extraída de `Cp.hs`) com recurso ao diagrama respectivo. Confirme-a com uma demonstração analítica.

2. Considere o isomorfismo

   $$A \times (B + 1) \quad \cong \quad A \times B + A$$

   (a) Apresente a definição *pointfree* da função que o testemunha da direita para a esquerda.

   (b) Formule a propriedade natural (*grátis*) dessa função, através de um diagrama.

   (c) Demonstre analiticamente essa propriedade.

3. Faça a inferência do tipo polimórfico principal (isto é, mais geral) da função $\langle \pi_1 + \pi_1, [\pi_2 \ , \pi_2] \rangle$ através de um diagrama.

4. Se tentar definir em Haskell o combinador

   $$comb\ f\ g \quad = \quad \langle f, [g\ , f] \rangle$$

   obterá a seguinte mensagem de erro:

```
comb.hs:3:29:
    Occurs check: cannot construct the infinite type: b = Either a b
      Expected type: b -> c
      Inferred type: Either a b -> b1
    In the second argument of `either', namely `f'
    In the second argument of `split', namely `(either g f)'
Failed, modules loaded: Cp.
```

   Explique a mensagem de erro acima mostrando que é impossível construir um diagrama para o tipo de $comb\ f\ g$.

5. Defina funções em Haskell que testemunhem os seguintes isomorfismos:

   (a) $\mathsf{Maybe}\ a \cong \mathsf{Either}\ a\ ()$

   (b) $\mathsf{Either}\ ()\ () \cong \mathsf{Bool}$

   Investigue a propriedade "grátis" das funções que escreveu na alínea (b).

6. Seja dada a função

$$\mathsf{ap} :: (a \to b, a) \to b$$
$$\mathsf{ap}\ (f, x) = f\ x$$

(a) Mostre, através da adição de variáveis, que a função f definida a seguir

$$f\ k = \mathsf{ap} \cdot (k \times id)$$

é a função

$$\mathsf{uncurry} :: (a \to b \to c) \to (a, b) \to c$$
$$\mathsf{uncurry}\ f\ (a, b) = f\ a\ b$$

que conhece de Programação Funcional.

(b) Mostre que a igualdade

$$\mathsf{ap} \cdot ((\mathsf{curry}\ f) \times id) = f$$

corresponde à definição

$$\mathsf{curry}\ f\ a\ b \quad = \quad f\ (a, b)$$

da função $\mathsf{curry} :: ((a, b) \to c) \to a \to b \to c$ que também conhece de Programação Funcional.

7. Sejam dadas f e g satisfazendo a propriedade

$$f\ y = x \quad \equiv \quad y = g\ x$$

Mostre que f e g são inversas uma da outra,

$$\begin{array}{rcl} id & = & g \cdot f \\ f \cdot g & = & id \end{array}$$

e que, portanto, são ambas isomorfismos.

8. Considere a função, em Haskell

$$g\ (\mathsf{Leaf}\ a) = \mathsf{Leaf}\ (\mathsf{succ}\ a)$$
$$g\ (\mathsf{Fork}\ (x, y)) = \mathsf{Fork}\ (g\ x, g\ y)$$

definida sobre uma instância do tipo de dados

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

do qual se infere a existência da função

$$inLTree = [\mathsf{Leaf}\ , \mathsf{Fork}]$$

É possível mostrar que g satisfaz a equação

$$g \cdot inLTree = inLTree \cdot k \cdot (id + g \times g)$$

para uma dada função k. Calcule k.
Sugestão: retire as variáveis a, x e y à definição dada, converta-a numa igualdade *pointfree* e compare o resultado como a equação acima.