Cálculo de Programas

2.° ano da Licenciatura em Engenharia Informática da Universidade do Minho

2010/11 - 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. Recorra ao algoritmo de Damas-Milner para inferir o tipo principal (polimórfico) da função $\langle \pi_1 + \pi_1, [\pi_2\ , \pi_2] \rangle$.

4. Suponha que tenta definir em Haskell o combinador seguinte:

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

   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 obtida aplicando a *comb* o algoritmo de Damas-Milner para a inferência de tipos polimórficos.

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 segunda alínea.

6. Seja dada a igualdade

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

onde $\mathsf{curry} :: ((a, b) \rightarrow c) \rightarrow a \rightarrow b \rightarrow c$ é um isomorfismo que conhece e a função de ordem superior $\mathsf{ap} :: (a \rightarrow b, a) \rightarrow b$ se define por $\mathsf{ap}\ (f, x) = f\ x$.

Mostre que a igualdade dada mais não é que a própria definição de curry que conhece:

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

7. 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

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

É possível mostrar que g satisfaz a equação

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

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.

8. Considere o seguinte sistema de equações

$$\mathit{fib} = \pi_2 \cdot \mathit{fib}'$$
$$\mathit{fib}' \cdot [\underline{0}\ , 1+] = [\langle \underline{1}, \underline{1} \rangle\ , \langle \mathsf{add}, \pi_2 \rangle] \cdot (id + \mathit{fib}')$$

em que se define a função de Fibonacci (*fib*) com recurso a uma função auxiliar (*fib*′), em que $\mathsf{add}\ (n, m) = n + m$. Converta esse sistema num programa Haskell *pointwise*.