

Cálculo de Programas

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

2011/12 - Ficha nr.º 5

1. Formule a lei natural (“grátis”) da função

$$\begin{aligned} pwnil &:: a \rightarrow (a, ()) \\ pwnil &= \langle id, ! \rangle \end{aligned}$$

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

2. Considere o isomorfismo

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

- Apresente a definição *pointfree* da função que o testemunha da direita para a esquerda.
 - Formule a propriedade natural (*grátis*) dessa função, através de um diagrama.
 - 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

$$\text{comb } f \ g = \langle f, [g, f] \rangle$$

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:

- Maybe $a \cong$ Either $a ()$
- Either $() () \cong$ Bool

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

6. Seja dada a função

$$\begin{aligned} \text{ap} &:: (a \rightarrow b, a) \rightarrow b \\ \text{ap} (f, x) &= f x \end{aligned}$$

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

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

é a função

$$\begin{aligned} \text{uncurry} &:: (a \rightarrow b \rightarrow c) \rightarrow (a, b) \rightarrow c \\ \text{uncurry } f (a, b) &= f a b \end{aligned}$$

que conhece de Programação Funcional.

(b) Mostre que a igualdade

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

corresponde à definição

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

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

7. Sejam dadas f e g satisfazendo a propriedade

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

Mostre que f e g são inversas uma da outra,

$$\begin{aligned} \text{id} &= g \cdot f \\ f \cdot g &= \text{id} \end{aligned}$$

e que, portanto, são ambas isomorfismos.

8. Considere a função, em Haskell

$$\begin{aligned} g (\text{Leaf } a) &= \text{Leaf } (\text{succ } a) \\ g (\text{Fork } (x, y)) &= \text{Fork } (g x, g y) \end{aligned}$$

definida sobre uma instância do tipo de dados

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

do qual se infere a existência da função

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

É possível mostrar que g satisfaz a equação

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

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.