Cálculo de Programas

2.° ano
Lic. Ciências da Computação e Mestrado Integrado em Engenharia Informática
UNIVERSIDADE DO MINHO

2019/20 - Ficha nr.° 2

1. Recorde as propriedades universais dos combinadores $\langle f\,,g\rangle$ e $[f\,,g]$,

$$k = \langle f, g\rangle \quad \equiv \quad \left\{ \begin{array}{l} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{array} \right.$$

$$k = [f\,, g] \quad \equiv \quad \left\{ \begin{array}{l} k \cdot i_1 = f \\ k \cdot i_2 = g \end{array} \right.$$

das quais, como sabe, podem ser derivadas todas as outras que aparecem no respectivo grupo, no formulário.

   (a) Use a segunda para demonstrar a lei $[i_1\,, i_2] = id$ conhecida por *Reflexão*-$+$.

   (b) Use a primeira para demonstrar a lei

$$\langle h, k\rangle = \langle f\,, g\rangle \quad \equiv \quad \left\{ \begin{array}{l} h = f \\ k = g \end{array} \right.$$

   que também consta desse formulário sob a designação *Eq*-$\times$.

2. Uma função diz-se *constante* sempre que o seu resultado é o mesmo, qualquer que seja o argumento. Por isso se designa uma tal função sublinhando o valor do seu resultado: se este for k, por exemplo, ter-se-á a função $\underline{k} : A \to K$, para k um valor de K, que satisfaz sempre a propriedade

$$\underline{k} \cdot f \quad = \quad \underline{k}$$

qualquer que seja k e f.[1]

Mostre que $[\underline{k}\,, \underline{k}] = \underline{k}$ aplicando a segunda lei universal dada acima.

3. O combinador funcional *soma* define-se por: $f + g = [i_1 \cdot f\,, i_2 \cdot g]$. Identifique os nomes das seguintes propriedades

$$id + id = id$$
$$(f + g) \cdot i_1 = i_1 \cdot f$$
$$(f + g) \cdot i_2 = i_2 \cdot g$$

no **formulário** da disciplina e demonstre-as usando o cálculo de programas.

4. Seja dada a função $\mathsf{coswap} = [i_2\,, i_1]$. Faça um diagrama que explique o tipo de coswap e mostre, usando o cálculo de programas, que $\mathsf{coswap} \cdot \mathsf{coswap} = id$.

---

[1] A função $\underline{k}$ escreve-se `const k` em Haskell.

5. Considere a função

$$\alpha = [\langle \underline{\mathsf{False}}, id \rangle, \langle \underline{\mathsf{True}}, id \rangle]$$

Determine o tipo de α e mostre, usando a propriedade *universal*-$+$, que α se pode escrever em Haskell da forma seguinte:

$$\alpha\ (i_1\ a) = (\mathsf{False}, a)$$
$$\alpha\ (i_2\ a) = (\mathsf{True}, a)$$

6. Recorra à lei Eq-$+$ (entre várias outras) para mostrar que a definição que conhece da função factorial,

$$fac\ 0 = 1$$
$$fac\ (n+1) = (n+1) * fac\ n$$

é equivalente à equação seguinte

$$fac \cdot [\underline{0}, \mathsf{succ}] = [\underline{1}, \mathsf{mul} \cdot \langle \mathsf{succ}, fac \rangle].$$

onde $\mathsf{succ}\ n = n + 1$ e $\mathsf{mul}\ (a, b) = a * b$.

7. Considere a seguinte declaração de um tipo de *árvores binárias*, em Haskell:

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

Indagando os tipos dos construtores Leaf e Fork, por exemplo no GHCi,

```
*LTree> :t Fork
Fork :: (LTree a, LTree a) -> LTree a
*LTree> :t Leaf
Leaf :: a -> LTree a
```

é fácil desenhar o diagrama que explica a construção da função

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

Desenhe-o e calcule a sua inversa

$$\mathsf{out} :: \mathsf{LTree}\ a \to a + \mathsf{LTree}\ a \times \mathsf{LTree}\ a$$
$$\mathsf{out}\ (\mathit{Leaf}\ a) = i_1\ a$$
$$\mathsf{out}\ (\mathit{Fork}\ (x, y)) = i_2\ (x, y)$$

resolvendo a equação

$$\mathsf{out} \cdot \mathsf{in} \quad = \quad id$$

em ordem a out.

Finalmente, faça testes em Haskell que involvam a composição $\mathsf{in} \cdot \mathsf{out}$. Que “conclusão”tira desses testes?[2]

[2]Escreve-se “conclusão”pois um teste nunca prova um resultado geral. Mas pode ajudar a *intuir* qual é esse resultado.