

Cálculo de Programas

2.º ano

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

2017/18 - Ficha nr.º 3

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

$$k = \langle f, g \rangle \equiv \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases}$$
$$k = [f, g] \equiv \begin{cases} k \cdot i_1 = f \\ k \cdot i_2 = g \end{cases}$$

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

$$\langle f, g \rangle = \langle h, k \rangle \equiv \begin{cases} f = h \\ g = k \end{cases}$$

que também consta desse formulário sob a designação *Eq-×*.

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 $\underline{k} : A \rightarrow K$, para k um valor de K , que satisfaz sempre a propriedade

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

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 $\text{coswap} = [i_2, i_1]$. Faça um diagrama que explique o tipo de *coswap* e mostre, usando o cálculo de programas, que $\text{coswap} \cdot \text{coswap} = id$.

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

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

$$\begin{aligned} fac\ 0 &= 1 \\ fac\ (n + 1) &= (n + 1) * fac\ n \end{aligned}$$

é equivalente à equação seguinte

$$fac \cdot [0, succ] = [1, mul \cdot \langle succ, fac \rangle].$$

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

6. Considere o isomorfismo

$$(A + B) + C \begin{array}{c} \xrightarrow{\text{coassocr}} \\ \cong \\ \xleftarrow{\text{coassocl}} \end{array} A + (B + C)$$

onde $coassocr = [id + i_1, i_2 \cdot i_2]$. Calcule a sua conversa $coassocl$ a partir da equação

$$coassocl \cdot coassocr = id$$

(**Sugestão:** Faça-o resolvendo em ordem a x, y e z a seguinte versão

$$\underbrace{[x, [y, z]]}_{coassocl} \cdot \underbrace{[id + i_1, i_2 \cdot i_2]}_{coassocr} = id$$

dessa equação.) Finalmente, calcule da equação $coassocl = [x, [y, z]]$ que tiver obtido um programa em Haskell que implemente $coassocl$ *sem recurso* ao combinador "either".

7. Considere a função

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

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

$$\begin{aligned} \alpha\ (i_1\ a) &= (\text{False}, a) \\ \alpha\ (i_2\ a) &= (\text{True}, a) \end{aligned}$$

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

```
data LTree a = Leaf a | Fork (LTree a, 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

$$in = [Leaf, Fork]$$

Desenhe-o e calcule a sua inversa

$$\begin{aligned} out &:: LTree\ a \rightarrow a + (LTree\ a, LTree\ a) \\ out\ (Leaf\ a) &= i_1\ a \\ out\ (Fork\ (x, y)) &= i_2\ (x, y) \end{aligned}$$

resolvendo a equação

$$out \cdot in = id$$

em ordem a out.