

# Cálculo de Programas

2.º ano

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

2019/20 - Ficha nr.º 8

1. Um *bifunctor*  $B$  é um functor **binário**

$$\begin{array}{ccc} A & \dots\dots & C & \dots\dots & B(A, C) \\ f \downarrow & & g \downarrow & & \downarrow B(f, g) \\ D & \dots\dots & E & \dots\dots & B(D, E) \end{array} \quad \text{tal que: } \begin{cases} B(id, id) = id \\ B(f \cdot g, h \cdot k) = B(f, h) \cdot B(g, k) \end{cases} \quad (F1)$$

Mostre que  $B(X, Y) = X \times Y$ ,  $B(X, Y) = X + Y$  e  $B(X, Y) = X + Y \times Y$  são bifunctors.

2. O diagrama genérico de um catamorfismo de gene  $g$  sobre o *tipo paramétrico*  $T X \cong B(X, T X)$  cuja base é o bifunctor  $B$ , bem como a sua propriedade universal, são representados a seguir:

$$\begin{array}{ccc} T X & \xleftarrow{\text{in}} & B(X, T X) \\ \downarrow \llbracket g \rrbracket & & \downarrow B(id, \llbracket g \rrbracket) = F \llbracket g \rrbracket \\ B & \xleftarrow{g} & B(X, B) \end{array} \quad k = \llbracket g \rrbracket \equiv k \cdot \text{in} = g \cdot \underbrace{B(id, k)}_{F k}$$

(Repare-se que se tem sempre  $F k = B(id, k)$ .) Partindo da definição *genérica* de map associado ao tipo  $T$ ,

$$T f = \llbracket \text{in} \cdot B(f, id) \rrbracket$$

dada no formulário, mostre que o map das sequências finitas (vulg. listas) é a função

$$\begin{aligned} f^* [] &= [] \\ f^* (h : t) &= f h : f^* t \end{aligned}$$

3. Considere a função  $depth = \llbracket [one, succ \cdot umax] \rrbracket$  que calcula a profundidade de árvores do tipo

$$T X = LTree X \quad \begin{cases} B(X, Y) = X + Y^2 \\ B(f, g) = f + g^2 \end{cases} \quad \text{in} = [Leaf, Fork]$$

Haskell: `data LTree a = Leaf a | Fork (LTree a, LTree a)`

onde  $umax(a, b) = max a b$ . Mostre, por absorção-cata, que a profundidade de uma árvore  $t$  não é alterada quando aplica uma função  $f$  a todas as suas folhas:

$$depth \cdot LTree f = depth \quad (F2)$$

4. Numa página de STACK OVERFLOW<sup>1</sup> alguém respondeu afirmativamente à pergunta

*Pode fazer-se unzip num só passo?*

com a versão

```
unzip [] = ([], [])
unzip ((a, b) : xs) = (a : as, b : bs) where (as, bs) = unzip xs
```

O que essa página não faz é explicar como é que os dois passos de

```
unzip xs = (map π1 xs, map π2 xs)
```

se fundem num só. Como exemplo de aplicação da lei de *banana-split*,

$$\langle \langle i \rangle, \langle j \rangle \rangle = \langle (i \times j) \cdot \langle F \pi_1, F \pi_2 \rangle \rangle$$

— identifique-a no formulário — complete a derivação que se dá a seguir dessa evidência:

$$\begin{aligned} & \text{unzip } xs = (\text{map } \pi_1 \text{ } xs, \text{map } \pi_2 \text{ } xs) \\ \equiv & \quad \{ \dots \} \\ & \text{unzip} = \langle \text{map } \pi_1, \text{map } \pi_2 \rangle \\ \equiv & \quad \{ \dots \} \\ & \text{unzip} = \langle \langle \text{in} \cdot B(\pi_1, id) \rangle, \langle \text{in} \cdot B(\pi_2, id) \rangle \rangle \\ \equiv & \quad \{ \dots \} \\ & \text{unzip} = \langle \langle \text{in} \cdot B(\pi_1, id) \cdot F \pi_1, \text{in} \cdot B(\pi_2, id) \cdot F \pi_2 \rangle \rangle \\ \equiv & \quad \{ \dots \} \\ & \text{unzip} = \langle \langle \text{in} \cdot B(\pi_1, \pi_1), \text{in} \cdot B(\pi_2, \pi_2) \rangle \rangle \\ \equiv & \quad \{ \dots \} \\ & \text{unzip} \cdot \text{in} = \langle [\text{nil}, \text{cons} \cdot (\pi_1 \times \pi_1)], [\text{nil}, \text{cons} \cdot (\pi_2 \times \pi_2)] \rangle \cdot (id + id \times \text{unzip}) \\ \equiv & \quad \{ \dots \} \\ & \left\{ \begin{array}{l} \text{unzip} \cdot \text{nil} = \langle \text{nil}, \text{nil} \rangle \\ \text{unzip} \cdot \text{cons} = (\text{cons} \times \text{cons}) \cdot \langle \pi_1 \times \pi_1, \pi_2 \times \pi_2 \rangle \cdot (id \times \text{unzip}) \end{array} \right. \\ \equiv & \quad \{ \dots \text{alguns passos mais} \dots \} \\ & \dots \\ \equiv & \quad \{ \dots \} \\ & \left\{ \begin{array}{l} \text{unzip } [] = ([], []) \\ \text{unzip } ((a, b) : xs) = ((a : as), (b : bs)) \text{ **where** } (as, bs) = \text{unzip } xs \end{array} \right. \end{aligned}$$

5. Defina-se a função  $average = ratio \cdot \langle \text{sum}, \text{length} \rangle$  para calcular a média de uma lista (não vazia), onde  $ratio(n, d) = n / d$ , para  $d \neq 0$ . Sabendo que  $\text{sum}$  e  $\text{length}$  são catamorfismos (recorde quais são os seus genes), recorra à lei “banana-split” para derivar

```
average l = x / y where
  (x, y) = aux l
  aux [] = (0, 0)
  aux (a : l) = (a + x, y + 1) where (x, y) = aux l
```

em Haskell.

<sup>1</sup>Cf. <https://stackoverflow.com/questions/18287848/unzip-in-one-pass>.