

Cálculo de Programas

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

2013/14 - Ficha nr.º 11

1. O algoritmo da divisão inteira,

$$m \div n \quad \begin{cases} m < n = 0 \\ \text{otherwise} = 1 + (m - n) \div n \end{cases}$$

corresponde ao anamorfismo $(\div n) = \llbracket g \ n \rrbracket$ em que $g \ n = (<n) \rightarrow (i_1 \cdot !), (i_2 \cdot (-n))$. É de esperar que o algoritmo dado satisfaça a propriedade $(m * n) \div n = m$, isto é, $(\div n) \cdot (*n) = id$. Complete a prova seguinte, por fusão-ana, dessa propriedade:

$$\begin{aligned} & (\div n) \cdot (*n) = id \\ \equiv & \quad \{ \dots \} \\ & \llbracket g \ n \rrbracket \cdot (*n) = \llbracket \text{out} \rrbracket \\ \Leftarrow & \quad \{ \dots \} \\ & (g \ n) \cdot (*n) = (id + (*n)) \cdot \text{out} \\ \equiv & \quad \{ \dots \} \\ & ((<n) \cdot (*n)) \rightarrow (i_1 \cdot !), (i_2 \cdot (-n)) \cdot (*n) = (id + (*n)) \cdot \text{out} \\ \equiv & \quad \{ \dots \} \\ & (= 0) \rightarrow (i_1 \cdot !), (i_2 \cdot (*n) \cdot \text{pred}) = (id + (*n)) \cdot \text{out} \\ \equiv & \quad \{ \dots \} \\ & (id + (*n)) \cdot \text{out} = (id + (*n)) \cdot \text{out} \\ \equiv & \quad \{ \dots \} \\ & \text{true} \end{aligned}$$

2. Recorde o diagrama genérico de um catamorfismo de gene g sobre o tipo T e a sua propriedade universal:

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

Nesta disciplina vimos vários exemplos de T , por exemplo os números naturais \mathbb{N}_0 , listas $[A]$ e dois tipos de árvores binárias,

```

data LTree a = Leaf a | Fork (LTree a, LTree a)
data BTree a = Empty | Node (a, (BTree a, BTree a))

```

A estes tipos podemos acrescentar outros como, por exemplo, o das listas não vazias

```

data NEList a = Sing a | Add (a, NEList a)

```

e o das chamadas “rose trees”:

```

data Rose a = Rose a [Rose a]

```

Preencha o quadro seguinte, em que a coluna da esquerda identifica funções sobre o tipo da coluna T, funções essas que conhece ou cujo significado facilmente identifica:

k	g	$F X$	$F f$	T	in	B
length		$1 + A \times X$		[A]	[nil, cons]	\mathbb{N}_0
length			$id + id \times f$	NEList A		\mathbb{N}_0
count					[Leaf, Fork]	\mathbb{N}
listify	$[singl, \widehat{(\++)}]$			LTree A		[A]
reverse					[nil, cons]	
sum		$1 + A \times X^2$				
sum					[Sing, Add]	
mirror	in · (id + swap)				[Leaf, Fork]	
mirror					[Empty, Node]	
filter p			$id + id \times f$	[A]		[A]
gmax	[id, max]	$A + A \times X$				A
gmax	[id, max]				[Leaf, Fork]	A

3. Defina como um catamorfismo a função seguinte, extraída do *Prelude* do Haskell,

```

concat :: [[a]] → [a]
concat = foldr (++) []

```

e mostre que a propriedade

$$\text{length} \cdot \text{concat} = \text{sum} \cdot \text{map length} \quad (1)$$

se verifica, recorrendo às leis de *fusão- e absorção-cata*

$$f \cdot \langle h \rangle = \langle k \rangle \iff f \cdot h = k \cdot (F f) \quad (2)$$

$$\langle h \rangle \cdot T f = \langle h \cdot B(f, id) \rangle \quad (3)$$

em que, para listas, se tem $B(f, g) = id + f \times g$, $F f = B(id, f)$ e $T f = \text{map } f$.

4. A função correspondente a concat para árvores é

```

join :: LTree (LTree a) → LTree a
join = \([id, Fork])

```

que junta uma *árvore de árvores* de tipo LTree numa só árvore. Conjecture a propriedade (1) para join e demonstre-a.

5. Recorra às leis dos catamorfismos para demonstrar a propriedade natural

$$(LTree f) \cdot \text{mirror} = \text{mirror} \cdot (LTree f) \quad (4)$$

onde mirror é o catamorfismo

```

mirror :: LTree a → LTree a
mirror = \([in \cdot (id + swap)])

```

que “espelha” uma árvore e $LTree f = \langle in \cdot (f + id) \rangle$ é o correspondente functor de tipo.