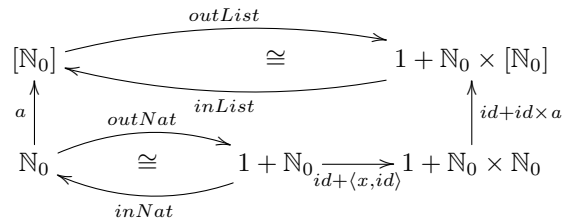


# Cálculo de Programas

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

2011/12 - Ficha nr.º 9

1. Considere o diagrama



que capta a seguinte propriedade da função  $a$ ,

$$a = inList \cdot (id + id \times a) \cdot (id + \langle x, id \rangle) \cdot outNat$$

para  $inNat = [\_0, succ]$  e  $inList = [nil, cons]$ , onde  $nil = []$  e  $cons (h, t) = h : t$ .

Funções com esta estrutura dizem-se *anamorfismos* do seu tipo de saída (listas de naturais neste caso).

(a) Explique por que é que a propriedade dada se pode escrever, alternativamente, sob a forma

$$a \cdot inNat = inList \cdot (id + \langle x, a \rangle) \tag{1}$$

(b) Diga o que faz a função  $a$  para  $x = succ$ .

2. Seja dada a função

$$odd = (id + \langle impar, id \rangle) \cdot outNat$$

**where**  $impar\ n = 2 * n + 1$

Faça o diagrama do anamorfismo de listas  $odds = [odd]$  e derive a correspondente versão em Haskell com variáveis.

3. Identifique o tipo do anamorfismo

$$suffixes = [g]$$

**where**  $g\ [] = i_1\ []$   
 $g\ (h : t) = i_2\ (h : t, t)$

representando-o sob a forma de um diagrama.

4. Relembre o cálculo que fez numa ficha anterior da lei de fusão-cata e adapte-o ao cálculo da lei de fusão-ana:

$$[g] \cdot f = [k] \iff g \cdot f = (F f) \cdot k$$

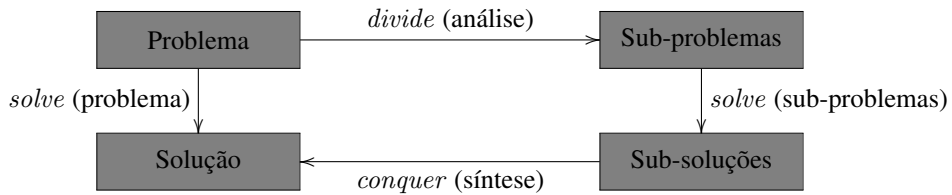
5. Investigue o comportamento do anamorfo *repeat* cujo diagrama é:

$$\begin{array}{ccc}
 [\mathbb{N}_0] & \xleftarrow{\text{inList}} & 1 + \mathbb{N}_0 \times [\mathbb{N}_0] \\
 \text{repeat} \uparrow & & \uparrow \text{id} + \text{id} \times \text{repeat} \\
 \mathbb{N}_0 & \xrightarrow{i_2 \cdot \langle \text{id}, \text{id} \rangle} & 1 + \mathbb{N}_0 \times \mathbb{N}_0
 \end{array}$$

Em particular, derive a definição *pointwise* de *repeat* a partir da sua definição

$$\text{repeat} = \llbracket i_2 \cdot \langle \text{id}, \text{id} \rangle \rrbracket$$

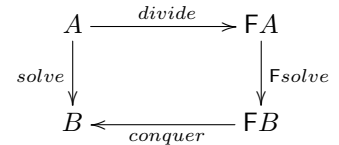
6. O desenho que se segue



descreve aquela que é talvez a principal competência de um bom programador: a capacidade de dividir um problema complexo em partes e a de saber juntar as respectivas sub-soluções para assim resolver o problema inicial.

No Cálculo de Programas, o esquema desenhado acima é captado pelo diagrama de um *hilomorfismo*, cujo ingrediente principal é a fixação do padrão de organização das sub-soluções, captado pelo *functor* polinomial *F*:

$$\begin{aligned}
 \text{solve} &= \llbracket \text{conquer}, \text{divide} \rrbracket \\
 &= \langle \text{conquer} \rangle \cdot \llbracket \text{divide} \rrbracket
 \end{aligned}
 \quad \text{a que corresponde o diagrama}$$



isto é, tem-se a equação

$$\text{solve} = \text{conquer} \cdot (\text{F solve}) \cdot \text{divide} \tag{2}$$

(a) Mostre que

- para *divide* = *out* se tem *solve* =  $\langle \text{conquer} \rangle$
- para *conquer* = *in* se tem *solve* =  $\llbracket \text{divide} \rrbracket$ .

(b) Derive a seguinte definição da função factorial,

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

a partir da sua construção como o hilomorfismo de listas ( $F f = \text{id} + \text{id} \times f$ ) cujo anamorfo *enumera* os primeiros *n* números naturais e cujo catamorfismo *multiplica* esses números. Após a identificação de *divide* e de *conquer*, faça um diagrama da composição  $\text{fac} = \langle \text{conquer} \rangle \cdot \llbracket \text{divide} \rrbracket$ . Finalmente, derive a definição dada por aplicação da equação (2).