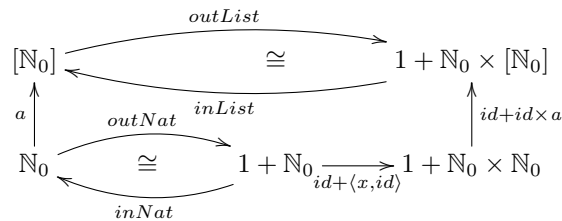


# Cálculo de Programas

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

2012/13 - Ficha nr.º 10

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  $in = [0, succ]$  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

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

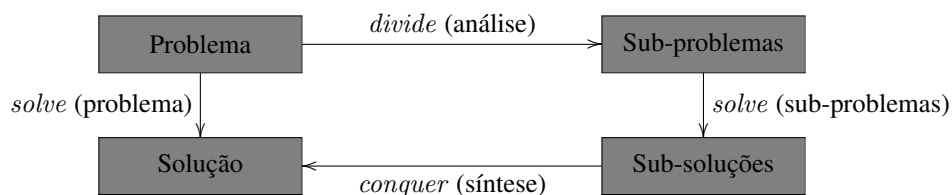
(b) Diga o que faz a função  $a$  para  $x = succ$ .

2. Considere os anamorfismos

$$\begin{array}{ll} odds = [odd] \text{ where} & suffixes = [g] \\ odd = (id + \langle impar, id \rangle) \cdot outNat & \text{e} \quad \text{where } g [] = i_1 [] \\ impar \ n = 2 * n + 1 & g(h : t) = i_2(h : t, t) \end{array}$$

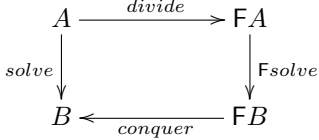
Faça o diagrama dos respectivos diagramas e derive as correspondentes versões em Haskell com variáveis.

3. O desenho que se segue

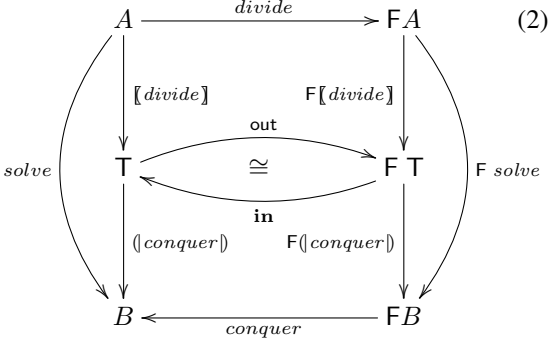


descreve aquela que é talvez a principal competência de um bom programador: a capacidade de dividir um problema complexo em partes e a de saber juntar as respectivas sub-soluções para assim resolver o problema inicial.

No Cálculo de Programas, o esquema desenhado acima é captado pelo conceito de *hilomorfismo*,

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

(1)

que normalmente se escreve  $solve = \llbracket conquer, divide \rrbracket$  e se factoriza na composição do anamorfismo  $\llbracket divide \rrbracket$  com o catamorfismo  $\llbracket conquer \rrbracket$ ,

$$\llbracket conquer, divide \rrbracket = \llbracket conquer \rrbracket \cdot \llbracket divide \rrbracket$$

(2)

mediados por uma estrutura de dados do tipo T associado ao functor F.<sup>1</sup>

- (a) Apresente justificações para os passos seguintes da demonstração do princípio da *hilo-factorização*, isto é, da equivalência entre (1) e (2):

$$\begin{aligned}
 & solve = \llbracket conquer, divide \rrbracket \\
 \equiv & \{ \dots\dots\dots \} \\
 & solve = \llbracket conquer \rrbracket \cdot \llbracket divide \rrbracket \\
 \equiv & \{ \dots\dots\dots \} \\
 & solve = (conquer \cdot F \llbracket conquer \rrbracket \cdot out) \cdot (in \cdot F \llbracket divide \rrbracket \cdot divide) \\
 \equiv & \{ \dots\dots\dots \} \\
 & solve = conquer \cdot F \llbracket conquer \rrbracket \cdot F \llbracket divide \rrbracket \cdot divide \\
 \equiv & \{ \dots\dots\dots \} \\
 & solve = conquer \cdot F (\llbracket conquer \rrbracket \cdot \llbracket divide \rrbracket) \cdot divide \\
 \equiv & \{ \dots\dots\dots \} \\
 & solve = conquer \cdot F solve \cdot divide
 \end{aligned}$$

- (b) Desenhe o diagrama (2) correspondente ao hilomorfismo  $f = \llbracket [g, h], p \rightarrow i_1, (i_2 \cdot k) \rrbracket$ , identificando o tipo T. Mostre ainda que  $f$  satisfaz a propriedade seguinte:

$$f = p \rightarrow g, (h \cdot f \cdot k)$$

<sup>1</sup>Esta estrutura intermédia é designada normalmente por estrutura de dados **virtual** por “não ser ver” quando o algoritmo executa, ficando escondida no ‘heap’ do sistema de ‘run-time’ da linguagem recursiva que está a ser utilizada.