

Cálculo de Programas

Algebra of Programming

UNIVERSIDADE DO MINHO
Lic. em Engenharia Informática (3º ano)
Lic. Ciências da Computação (2º ano)

2023/24 - Ficha (Exercise sheet) nr. 11

1. Considere o anamorfismo $r = \llbracket g \rrbracket$ em que

Consider the anamorphism $r = \llbracket g \rrbracket$ where g is

$$\begin{aligned} g [] &= i_1 () \\ g x &= i_2 (\text{last } x, \text{init } x) \end{aligned}$$

onde $\text{last } x$ dá o último elemento da lista x e $\text{init } x$ dá x sem esse último elemento. O que faz a função r ? Responda informalmente desenhando o diagrama de r .

in which $\text{last } x$ gives the last element of list x and $\text{init } x$ gives x without this last element. What does r do? Answer informally by drawing the diagram of r .

2. Considere a função:

Let function

$$x \ominus y = \text{if } x \leq y \text{ then } 0 \text{ else } 1 + x \ominus (y + 1)$$

Quais os valores das expressões $(3 \ominus 2) \ominus 3$ e $(3 \ominus 4) + 4$? Codifique $\widehat{\ominus} : \mathbb{N}_0 \times \mathbb{N}_0 \rightarrow \mathbb{N}_0$ como um anamorfismo de naturais e faça o respectivo diagrama.

be given. Evaluate $(3 \ominus 2) \ominus 3$ and $(3 \ominus 4) + 4$ and encode $\widehat{\ominus} : \mathbb{N}_0 \times \mathbb{N}_0 \rightarrow \mathbb{N}_0$ as an anamorphism over \mathbb{N}_0 . Draw the corresponding diagram.

3. O isomorfismo $\text{in} : B(A, T A) \rightarrow T A$ construtor dos habitantes de um tipo recursivo (paramétrico) de base B é ele próprio paramétrico em A . Complete o seguinte diagrama que capta a propriedade natural (grátis) de in :

Isomorphism $\text{in} : B(A, T A) \rightarrow T A$ constructing inhabitants of a recursive (parametric) base type B is itself parametric on A . Complete the following diagram that captures the natural (free) property of in :

$$\begin{array}{ccc} T A & \xleftarrow{\text{in}} & B(A, T A) \\ \tau f \downarrow & & \downarrow ? \\ T A' & \xleftarrow{\text{in}} & B(A', T A') \end{array}$$

Instancie essa propriedade para listas, em que

Instantiate this property for lists, where

$$\begin{cases} T A = A^* \\ B(X, Y) = 1 + X \times Y \\ T f = \text{map } f \end{cases}$$

Desenvolva essa igualdade até chegar à sua formulação sem qualquer dos construtores *pointfree* estudados nesta disciplina. O que é que obteve, afinal?

Unfold the equality until a formulation is reached involving none of the pointfree constructors studied in this course. What did you get, after all?

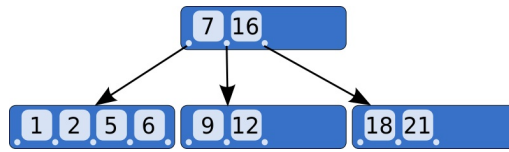
4. Uma “B-tree” é uma generalização das árvores binárias do módulo BTree a mais do que duas sub-árvores por nó:

A “B-tree” is a generalization of the binary trees of the BTree module to more than two subtrees per node:

data B_tree a = Nil | Block { leftmost :: B_tree a, block :: [(a, B_tree a)] }

Por exemplo, a B-tree

For instance, the B-tree



é representada no tipo acima por:

is represented by the data type above as:

```
t = Block {
  leftmost = Block {
    leftmost = Nil,
    block = [(1, Nil), (2, Nil), (5, Nil), (6, Nil)]},
  block = [
    (7, Block {
      leftmost = Nil,
      block = [(9, Nil), (12, Nil)]}),
    (16, Block {
      leftmost = Nil,
      block = [(18, Nil), (21, Nil)]})
  ]}
```

Identifique, justificando, o functor de base

Identify the base functor

$$\begin{cases} B(X, Y) = \dots \\ B(f, g) = \dots \end{cases} \quad (\text{F1})$$

que capta o padrão de recursividade da declaração de *B_tree* dada acima, em Haskell, bem como o isomorfismo:

which captures the recursion pattern of the declaration of B_tree given above, in Haskell, as well as the isomorphism:

$$\text{in} : B(A, B_tree A) \rightarrow B_tree A$$

5. O algoritmo “bubble-sort” é o ciclo-for

The “bubble-sort” algorithm is a for-loop:

```
bSort xs = for bubble xs (length xs) where
  bubble (x : y : xs)
    | x > y = y : bubble (x : xs)
    | otherwise = x : bubble (y : xs)
  bubble x = x
```

cujo corpo de ciclo é um hilomorfismo bubble = $\llbracket \text{conquer}, \text{divide} \rrbracket$. Identifique os genes *divide* e *conquer* desse hilomorfismo.

Its loop-body is a hylomorphism bubble = $\llbracket \text{conquer}, \text{divide} \rrbracket$. Identify the genes divide and conquer of this hylomorphism.

6. Todo o ciclo-*while* que termina pode ser definido por

Every terminating while-loop can be defined by

$$\mathbf{while} \ p \ f \ g = \mathbf{tailr} \ ((g + f) \cdot (\neg \cdot p)?) \quad (\text{F2})$$

recorrendo ao combinador de “tail recursion”

using the “tail recursion” combinator

$$\mathbf{tailr} \ f = \llbracket \text{join}, f \rrbracket \quad (\text{F3})$$

que é um hilomorfismo de base $B \ (X, Y) = X + Y$, para $\text{join} = [id, id]$.

which is a hylomorphism of basis $B \ (X, Y) = X + Y$, for $\text{join} = [id, id]$.

Derive a definição *pointwise* de $\mathbf{while} \ p \ f \ g$, sabendo que qualquer $h = \llbracket f, g \rrbracket$ que termina é tal que $h = f \cdot F \ h \cdot g$.

Derive the pointwise definition of $\mathbf{while} \ p \ f \ g$, knowing that any terminating $h = \llbracket f, g \rrbracket$ is such that $h = f \cdot F \ h \cdot g$.

7. Considere a seguinte lei de fusão de \mathbf{tailr} , válida sempre que $(\mathbf{tailr} \ g) \cdot f$ termina:

Consider the following fusion-law of \mathbf{tailr} , valid whenever $(\mathbf{tailr} \ g) \cdot f$ terminates:

$$(\mathbf{tailr} \ g) \cdot f = \mathbf{tailr} \ h \iff (id + f) \cdot h = g \cdot f \quad (\text{F4})$$

Complete a seguinte demonstração dessa lei.

Complete de following proof of (F4).

$$\begin{aligned}
 & (\mathbf{tailr} \ g) \cdot f = \mathbf{tailr} \ h \\
 \equiv & \quad \{ \dots\dots\dots \} \\
 & \llbracket \nabla \rrbracket \cdot \llbracket g \rrbracket \cdot f = \llbracket \nabla \rrbracket \cdot \llbracket h \rrbracket \\
 \Leftarrow & \quad \{ \dots\dots\dots \} \\
 & \llbracket g \rrbracket \cdot f = \llbracket h \rrbracket \\
 \Leftarrow & \quad \{ \dots\dots\dots \} \\
 & g \cdot f = (id + f) \cdot h \\
 & \square
 \end{aligned}$$