Cálculo de Programas
Algebra of Programming

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

2022/23 - Ficha *(Exercise sheet)* nr. 12 – última *(last)*

1. Considere a função:

   Let function

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

Quais os valores das expressões $(3 \ominus 2) \ominus 3$ e $(3 \ominus 4) + 4$? Codifique $\widehat{\ominus} : \mathbb{N}_0 \times \mathbb{N}_0 \to \mathbb{N}_0$ 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* $\widehat{\ominus} : \mathbb{N}_0 \times \mathbb{N}_0 \to \mathbb{N}_0$ *as an anamorphism over* $\mathbb{N}_0$. *Draw the corresponding diagram.*

---

2. (a) Mostre que a conhecida definição da função factorial segue o padrão recursivo

   (a) Show that the well-known definition of the factorial function follows the recursive pattern

$$f \cdot \mathsf{in} = g \cdot \mathsf{G}\ \langle id, f \rangle \qquad \text{(F1)}$$

identificando in, g e G. (b) A lei que se segue mostra que esse padrão é o hilomorfismo

by identifying in, g *and* G. *(b) The following law shows that pattern (F1) is a hylomorphism*

$$f \cdot \mathsf{in} = g \cdot \mathsf{G}\ \langle id, f \rangle \quad \equiv \quad f = (\!|\, g \,|\!) \cdot [\!(\mathsf{G}\ \langle id, id \rangle \cdot \mathsf{out})\!] \qquad \text{(F2)}$$

cuja base é o bifunctor:

whose basis is bifunctor:

$$\mathsf{B}\ (X, Y) = \mathsf{G}\ (X \times Y)$$

Apresente justificações para os passos do cálculo de (F2) que se segue:

Provide justifications for each step of the calculation of (F2) that follows:

$$\begin{array}{ll}
& f = (\!|\, g \,|\!) \cdot [\!(\mathsf{G}\ \langle id, id \rangle \cdot \mathsf{out})\!] \\
\equiv & \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& f = g \cdot \mathsf{F}\ f \cdot \mathsf{G}\ \langle id, id \rangle \cdot \mathsf{out} \\
\equiv & \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& f \cdot \mathsf{in} = g \cdot \mathsf{G}\ (id \times f) \cdot \mathsf{G}\ \langle id, id \rangle \\
\equiv & \qquad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& f \cdot \mathsf{in} = g \cdot \mathsf{G}\ \langle id, f \rangle \\
\Box &
\end{array}$$

---

3. O algoritmo "bubble-sort" é o ciclo-for — *The "bubble-sort" algorithm is the for-loop*

$$
\begin{array}{l}
bSort\ xs = \mathsf{for\ bubble}\ xs\ (\mathsf{length}\ xs)\ \mathbf{where} \\
\quad \mathsf{bubble}\ (x : y : xs) \\
\qquad \mid x > y = y : \mathsf{bubble}\ (x : xs) \\
\qquad \mid \mathsf{otherwise} = x : \mathsf{bubble}\ (y : xs) \\
\quad \mathsf{bubble}\ x = x
\end{array}
$$

cujo corpo de ciclo (bubble) é um hilomorfismo $\mathsf{bubble} = [\![conquer, divide]\!]$. Identifique os genes $divide$ e $conquer$ desse hilomorfismo. **Sugestão**: siga a heurística que foi usada nas aulas teóricas para fazer o mesmo para a função de Fibonacci.[a]

[a] Se não tiver vindo à aula teórica veja a mesma heurística a ser aplicada à função de Fibonacci no vídeo T9b, a começar em t=28:09.

*whose loop-body (*bubble*) is a hylomorphism* $\mathsf{bubble} = [\![conquer, divide]\!]$. *Identify the genes $divide$ and $conquer$ of this hylomorphism.* ***Suggestion**: follow the heuristics that were used in the lectures to do the same for the Fibonacci function.*[a]

[a] If you missed the lectures, see the same heuristics to be applied to the Fibonacci function in video T9b, starting at t=28:09.

---

4. Todo o ciclo-*while* que termina pode ser definido por — *Every terminating* while*-loop can be defined by*

$$\mathbf{while}\ p\ f\ g = \mathbf{tailr}\ ((g + f) \cdot (\neg \cdot p)?) \tag{F3}$$

recorrendo ao combinador de *"tail recursion"* — *using the* "tail recursion" *combinator*

$$\mathbf{tailr}\ f = [\![\mathsf{join}, f]\!] \tag{F4}$$

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

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

Derive a definição *pointwise* de $\mathbf{while}\ p\ f\ g$, sabendo que qualquer $h = [\![f, g]\!]$ que termine é tal que $h = f \cdot \mathsf{F}\ h \cdot g$.

Derive the pointwise *definition of* $\mathbf{while}\ p\ f\ g$*, knowing that any terminating* $h = [\![f, g]\!]$ *is such that* $h = f \cdot \mathsf{F}\ h \cdot g$.

---

5. 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:*

$$(\mathbf{tailr}\ g) \cdot f = \mathbf{tailr}\ h \ \Leftarrow\ (id + f) \cdot h = g \cdot f \tag{F5}$$

Complete a seguinte demonstração dessa lei. — *Complete de following proof of (F5).*

$$
\begin{array}{ll}
 & (\mathbf{tailr}\ g) \cdot f = \mathbf{tailr}\ h \\
\equiv & \quad \{ \ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\ \} \\
 & (\!|\ \nabla\ |\!) \cdot [\!(g)\!] \cdot f = (\!|\ \nabla\ |\!) \cdot [\!(h)\!] \\
\Leftarrow & \quad \{ \ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\ \} \\
 & [\!(g)\!] \cdot f = [\!(h)\!] \\
\Leftarrow & \quad \{ \ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\ \} \\
 & g \cdot f = (id + f) \cdot h \\
\Box &
\end{array}
$$

---

6. Demonstre a seguinte propriedade da composição monádica:

Prove the following property of monadic composition:

$$f \bullet [g\ , h] = [f \bullet g\ , f \bullet h] \tag{F6}$$

---

7. Seja M um monad e T um functor. Em Haskell, a instância para listas ($\mathsf{T}\ X = X^*$) da função monádica

Let M be a monad and T a functor. In Haskell, the instance for lists ($\mathsf{T}\ X = X^*$) of the monadic function

$$\mathsf{sequence} : \mathsf{T}\ (\mathsf{M}\ X) \rightarrow \mathsf{M}\ (\mathsf{T}\ X)$$

é o catamorfismo

is the catamorphism

$$\begin{array}{l} \mathsf{sequence} = (\!|\, g \,|\!)\ \mathbf{where} \\ \quad g = [\mathsf{return}\ , id] \cdot (\mathsf{nil} + \lfloor \mathsf{cons} \rfloor) \\ \quad \lfloor f \rfloor\ (x, y) = \mathbf{do}\ \{ a \leftarrow x; b \leftarrow y; \mathsf{return}\ (f\ (a, b)) \} \end{array}$$

tal como se mostra neste diagrama:

as in the following diagram:

$$\begin{tikzcd}
(\mathsf{M}\ X)^* \arrow[d, "\mathsf{sequence}"'] & 1 + \mathsf{M}\ X \times (\mathsf{M}\ X)^* \arrow[l, "\mathsf{in}"'] \arrow[d, "id + id \times \mathsf{sequence}"] \\
\mathsf{M}\ (X^*) & 1 + \mathsf{M}\ X \times \mathsf{M}\ (X^*) \arrow[l, dotted, "g"'] \arrow[d, "\mathsf{nil} + \lfloor \mathsf{cons} \rfloor"] \\
 & X^* + \mathsf{M}\ (X^*) \arrow[ul, dotted, "{[\mathsf{return}\ , id]}"]
\end{tikzcd}$$

Partindo da propriedade universal-cata, derive uma versão de sequence em Haskell com variáveis que não recorra à composição de funções.

Starting from the universal-cata property, derive a version of sequence *in Haskell with variables that doesn't resort to function composition.*

---

8. Mostre que o anamorfismo $\mathsf{repeat} = [\!(\langle id, id \rangle)\!]$ definido pelo diagrama

Show that the anamorphism $\mathsf{repeat} = [\!(\langle id, id \rangle)\!]$ *defined by the diagram*

$$\begin{tikzcd}
A^\infty & A \times A^\infty \arrow[l, "\mathsf{cons}"'] \\
A \arrow[u, "\mathsf{repeat}"] \arrow[r, "\langle id, id \rangle"'] & A \times A \arrow[u, "id \times \mathsf{repeat}"']
\end{tikzcd}$$

é a função: $\mathsf{repeat}\ a = a : \mathsf{repeat}\ a$. De seguida, recorrendo às leis dos anamorfismos mostre que, apesar de não terminar[a], repeat satisfaz a propriedade $\mathsf{map}\ f \cdot \mathsf{repeat} = \mathsf{repeat} \cdot f$. [b]

[a] Por isso usamos, no diagrama, A^∞ em vez de A^*, para incluir também as listas infinitas.

[b] "Verifique" este facto comparando, por exemplo, $(\mathsf{take}\ 10 \cdot \mathsf{map}\ \mathsf{succ} \cdot \mathsf{repeat})\ 1$ com $(\mathsf{take}\ 10 \cdot \mathsf{repeat} \cdot \mathsf{succ})\ 1$.

is the function: $\mathsf{repeat}\ a = a : \mathsf{repeat}\ a$. *Then, resorting to the laws of anamorphisms, show that, despite not ending* [a]*,* repeat *satisfies the* $\mathsf{map}\ f$ *property* $\cdot\ \mathsf{repeat} = \mathsf{repeat} \cdot f$. [b]

[a] That's why we use, in the diagram, A^∞ instead of A^*, to also include infinite lists.

[b] "Verify" this fact by comparing, for example, $(\mathsf{take}\ 10 \cdot \mathsf{map}\ \mathsf{succ} \cdot \mathsf{repeat})\ 1$ with $(\mathsf{take}\ 10 \cdot \mathsf{repeat} \cdot \mathsf{succ})\ 1$.