Cálculo de Programas
Algebra of Programming

Universidade do Minho
Lic. Ciências da Computação (3º ano)
Lic. em Engenharia Informática (3º ano)

2025/26 - Ficha *(Exercise sheet)* nr. 8

1. Considere o functor

Consider functor

$$\begin{cases} \mathsf{T}\ X = X \times X \\ \mathsf{T}\ f = f \times f \end{cases} \tag{F1}$$

e as funções

and functions

$$\begin{cases} \mu = \pi_1 \times \pi_2 \\ u = \langle id, id \rangle \end{cases} \tag{F2}$$

(a) Mostre que T é de facto um functor:

(a) Show that T is indeed a functor:

$$\mathsf{T}\ id = id \tag{F3}$$

$$\mathsf{T}\ (f \cdot g) = \mathsf{T}\ f \cdot \mathsf{T}\ g \tag{F4}$$

(b) Demonstre a propriedade:

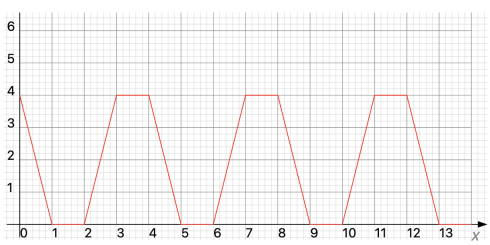
(b) Prove the following property:

$$\mu \cdot \mathsf{T}\ u = id = \mu \cdot u$$

---

2. A figura representa a função $\pi_1 \cdot aux$, para aux definida ao lado:

The figure plots $\pi_1 \cdot aux$, for aux defined aside:



$$\begin{array}{l} aux = \mathsf{for}\ loop\ (4, -2)\ \mathbf{where} \\ \quad loop\ (a, b) = (2 + b, 2 - a) \end{array}$$

Partindo da definição do combinador $\mathsf{for}\ b\ i = (\!|\,[\underline{i}\ , b]\,|\!)$, para $\mathsf{F} = id + f$ e $\mathsf{in} = [\underline{0}\ , \mathsf{succ}]$, resolva em ordem a f e g a equação

Starting from the definition of $\mathsf{for}\ b\ i = (\!|\,[\underline{i}\ , b]\,|\!)$, for $\mathsf{F} = id + f$ and $\mathsf{in} = [\underline{0}\ , \mathsf{succ}]$, solve for f and g the equation

$$\langle f, g \rangle = aux$$

por aplicação da lei de recursividade mútua, entregando as definições de f e g em notação *pointwise*.

by the mutual recursion law, delivering the definitions of f and g in pointwise *notation.*

---

3. Mostre que a lei da recursividade mútua generaliza a mais do que duas funções, neste caso três:

Show that the mutual recursion law generalizes to more than two functions (three, in the following case):

$$\left\{\begin{array}{l} f \cdot \mathsf{in} = h \cdot \mathsf{F}\ \langle\langle f, g\rangle, j\rangle \\ g \cdot \mathsf{in} = k \cdot \mathsf{F}\ \langle\langle f, g\rangle, j\rangle \\ j \cdot \mathsf{in} = l \cdot \mathsf{F}\ \langle\langle f, g\rangle, j\rangle \end{array}\right. \quad\equiv\quad \langle\langle f, g\rangle, j\rangle = (\!|\, \langle\langle h, k\rangle, l\rangle \,|\!) \tag{F5}$$

---

4. As seguintes funções mutuamente recursivas testam a paridade de um número natural:

The following mutually recursive functions test the parity of a natural number:

$$\left\{\begin{array}{l} impar\ 0 = \text{FALSE} \\ impar\ (n+1) = par\ n \end{array}\right. \qquad \left\{\begin{array}{l} par\ 0 = \text{TRUE} \\ par\ (n+1) = impar\ n \end{array}\right.$$

Assumindo o functor $\mathsf{F}\ f = id + f$, mostre que esse par de definições é equivalente ao sistema de equações

Assuming the functor $\mathsf{F}\ f = id + f$, show that this pair of definitions is equivalent to the system of equations

$$\left\{\begin{array}{l} impar \cdot \mathsf{in} = h \cdot \mathsf{F}\ \langle impar, par\rangle \\ par \cdot \mathsf{in} = k \cdot \mathsf{F}\ \langle impar, par\rangle \end{array}\right.$$

para um dado h e k (deduza-os). De seguida, recorra às leis da recursividade mútua e da troca para mostrar que

for a given h and k (calculate these). Then use the mutual recursion and exchange laws to show that

$$imparpar = \langle impar, par\rangle = \mathsf{for}\ \mathsf{swap}\ (\text{FALSE}, \text{TRUE})$$

---

5. A seguinte função em Haskell lista os primeiros n números naturais por ordem inversa:

The following Haskell function lists the n first natural numbers in reverse order:

$$\left\{\begin{array}{l} insg\ 0 = [\,] \\ insg\ (n+1) = (n+1) : insg\ n \end{array}\right.$$

Mostre que $insg$ pode ser definida por recursividade mútua tal como se segue:

Show that $insg$ can be defined by mutual recursion as follows:

$$\left\{\begin{array}{l} insg\ 0 = [\,] \\ insg\ (n+1) = (fsuc\ n) : insg\ n \end{array}\right.$$
$$\left\{\begin{array}{l} fsuc\ 0 = 1 \\ fsuc\ (n+1) = fsuc\ n + 1 \end{array}\right.$$

A seguir, usando a lei de recursividade mútua, derive:

Then, using the law of mutual recursion, derive:

$$insg = \pi_2 \cdot insgfor$$
$$insgfor = \mathsf{for}\ \langle (1+) \cdot \pi_1, \mathsf{cons}\rangle\ (1, [\,])$$

---

6. Considere o par de funções mutuamente recursivas

Consider the pair of mutually recursive functions

$$\left\{ \begin{array}{l} f_1 \; [\,] = [\,] \\ f_1 \; (h : t) = h : (f_2 \; t) \end{array} \right. \qquad \left\{ \begin{array}{l} f_2 \; [\,] = [\,] \\ f_2 \; (h : t) = f_1 \; t \end{array} \right.$$

Mostre por recursividade mútua que $\langle f_1, f_2 \rangle$ é um catamorfismo de listas (onde $\mathsf{F}\ f = id + id \times f$) e desenhe o respectivo diagrama. Que faz cada uma destas funções f_1 e f_2?

Show by mutual recursion that $\langle f_1, f_2 \rangle$ is a list catamorphism (for $\mathsf{F}\ f = id + id \times f$) and draw the corresponding diagram. What do functions f_1 and f_2 actually do?

---

7. Sejam dados os functores elementares seguintes:

Consider the following basic functors:

$$\left\{ \begin{array}{l} \mathsf{F}\ X = \mathbb{Z} \\ \mathsf{F}\ f = id \end{array} \right. \qquad \left\{ \begin{array}{l} \mathsf{G}\ X = X \\ \mathsf{G}\ f = f \end{array} \right.$$

Mostre que H e K definidos por

Show that H and K defined by

$$\mathsf{H}\ X = \mathsf{F}\ X + \mathsf{G}\ X$$
$$\mathsf{K}\ X = \mathsf{G}\ X \times \mathsf{F}\ X$$

são functores.

are functors.

---

8. Mostre que, sempre que F e G são functores, então a sua composição $\mathsf{H} = \mathsf{F} \cdot \mathsf{G}$ é também um functor.

Show that wherever F and G are functors, then their composition $\mathsf{H} = \mathsf{F} \cdot \mathsf{G}$ is also a functor.

---

9. **Questão prática** — Este problema não irá ser abordado em sala de aula. Os alunos devem tentar resolvê-lo em casa e, querendo, publicarem a sua solução no canal **#geral** do Slack, com vista à sua discussão com colegas.

Open assignment *— This assignment will not be addressed in class. Students should try to solve it at home and, whishing so, publish their solutions in the* ***#geral*** *Slack channel, so as to trigger discussion among other colleagues.*

Problem definition*: Page* UNZIP IN ONE PASS? *of* STACK OVERFLOW *addresses the question as to whether*

$$\mathsf{unzip}\ xs = (\mathsf{map}\ \pi_1\ xs, \mathsf{map}\ \pi_2\ xs)$$

can do one traversal only. The answer is affirmative:

$$\begin{array}{l} \mathsf{unzip}\ [\,] = ([\,],[\,]) \\ \mathsf{unzip}\ ((a,b) : xs) = (a : as, b : bs)\ \mathbf{where}\ (as, bs) = \mathsf{unzip}\ xs \end{array}$$

What is missing from STACK OVERFLOW *is the explanation of how the two steps of* unzip *merge into one.*

Show that the banana-split *law is what needs to be known for the one traversal version to be derived from the two traversal one.*

$\square$