

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. 8

1. Considere o functor

Consider functor

$$\begin{cases} \mathbb{T} X = X \times X \\ \mathbb{T} f = f \times f \end{cases}$$

e as funções

and functions

$$\mu = \pi_1 \times \pi_2$$

$$u = \langle id, id \rangle.$$

Demonstre a propriedade:

Prove the following property:

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

2. Sejam dados os funtores elementares seguintes:

Consider the following basic functors:

$$\begin{cases} F X = \mathbb{Z} \\ F f = id \end{cases} \quad \begin{cases} G X = X \\ G f = f \end{cases}$$

Mostre que H e K definidos por

Show that H and K defined by

$$H X = F X + G X$$

$$K X = G X \times F X$$

são funtores.

are functors.

3. Mostre que, sempre que F e G são funtores, então a sua composição $H = F \cdot G$ é também um functor.

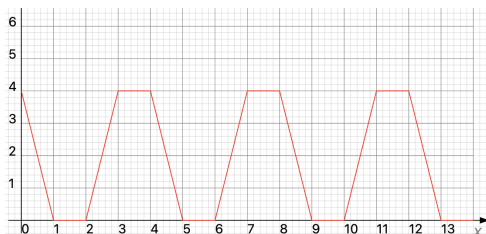
Show that wherever F and G are functors, then their composition $H = F \cdot G$ is also a functor.

4. 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):

$$\begin{cases} f \cdot in = h \cdot F \langle \langle f, g \rangle, j \rangle \\ g \cdot in = k \cdot F \langle \langle f, g \rangle, j \rangle \\ j \cdot in = l \cdot F \langle \langle f, g \rangle, j \rangle \end{cases} \equiv \langle \langle f, g \rangle, j \rangle = \langle \langle \langle h, k \rangle, l \rangle \rangle \quad (F1)$$

5. 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:

$$aux = \text{for loop } (4, -2) \text{ where} \\ \text{loop } (a, b) = (2 + b, 2 - a)$$

Partindo da definição do combinador for $b \ i = \llbracket [i, b] \rrbracket$, para $F = id + f$ e $in = [0, succ]$, resolva em ordem a f e g a equação

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

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

Starting from the definition of for $b \ i = \llbracket [i, b] \rrbracket$, for $F = id + f$ and $in = [0, succ]$, solve for f and g the equation

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

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

$$\begin{cases} impar \ 0 = \text{FALSE} \\ impar \ (n + 1) = par \ n \end{cases} \quad \begin{cases} par \ 0 = \text{TRUE} \\ par \ (n + 1) = impar \ n \end{cases}$$

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

$$\begin{cases} impar \cdot in = h \cdot F \langle impar, par \rangle \\ par \cdot in = k \cdot F \langle impar, par \rangle \end{cases}$$

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

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

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

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

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

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

$$\begin{cases} insg \ 0 = [] \\ insg \ (n + 1) = (n + 1) : insg \ n \end{cases}$$

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

$$\begin{cases} insg \ 0 = [] \\ insg \ (n + 1) = (fsuc \ n) : insg \ n \\ fsuc \ 0 = 1 \\ fsuc \ (n + 1) = fsuc \ n + 1 \end{cases}$$

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

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

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

Then, using the law of mutual recursion, derive:

$$\begin{aligned} \text{insg} &= \pi_2 \cdot \text{insgfor} \\ \text{insgfor} &= \text{for } \langle (1+) \cdot \pi_1, \text{cons} \rangle (1, []) \end{aligned}$$

-
8. Considere o par de funções mutuamente recursivas

Consider the pair of mutually recursive functions

$$\begin{cases} f_1 [] = [] \\ f_1 (h : t) = h : (f_2 t) \end{cases} \quad \begin{cases} f_2 [] = [] \\ f_2 (h : t) = f_1 t \end{cases}$$

Mostre por recursividade mútua que $\langle f_1, f_2 \rangle$ é um catamorfismo de listas (onde $F f = id + id \times f$) 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 $F f = id + id \times f$) and draw the corresponding diagram. What do functions f_1 and f_2 actually do?

-
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, wishing 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

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

can do one traversal only. The answer is affirmative:

$$\begin{aligned} \text{unzip } [] &= ([], []) \\ \text{unzip } ((a, b) : xs) &= (a : as, b : bs) \textbf{ where } (as, bs) = \text{unzip } xs \end{aligned}$$

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.

□