

Cálculo de Programas

Algebra of Programming

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

2024/25 - Ficha (*Exercise sheet*) nr. 8

1. A igualdade que se segue

The following equality

$$f \cdot \text{length} = ([\text{zero}, (2+) \cdot \pi_2])$$

verifica-se para $f = (2*)$ ou $f = (2+)$? Use a lei de fusão-cata para justificar, por cálculo, a sua resposta.

holds for $f = (2)$ or $f = (2+)$? Use the cata-fusion law to justify, by calculation, your answer.*

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

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

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

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

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

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

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

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

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

3. 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:

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

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

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

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

$$\begin{cases} \text{fsuc } 0 = 1 \\ \text{fsuc } (n + 1) = \text{fsuc } n + 1 \end{cases}$$

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

Then, using the law of mutual recursion, derive:

$$\text{insg} = \pi_2 \cdot \text{insgfor}$$

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

4. Considere o par de funções mutuamente recursivas

Consider the pair of mutually recursive functions

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

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

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?

5. Sejam dados os funtores elementares seguintes:

Consider the following basic functors:

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

$$\begin{cases} G X = X \\ G f = f \end{cases}$$

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.

6. 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.

7. **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.

□