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. 4

---

1. Considere a função $\mathsf{in} = [\underline{0}\ , \mathsf{succ}]$ (onde $\mathsf{succ}\ n = n + 1$) que exprime a forma como os números naturais ($\mathbb{N}_0$) são gerados a partir do número 0, de acordo com o diagrama seguinte:

   Let function $\mathsf{in} = [\underline{0}\ , \mathsf{succ}]$ *be given (where* $\mathsf{succ}\ n = n + 1$*) expressing the way natural numbers (*$\mathbb{N}_0$*) are generated from the number* 0*, according to the diagram below:*

$$\begin{tikzcd}
1 \arrow[r, "i_1"] \arrow[dr, "\underline{0}"'] & 1+\mathbb{N}_0 \arrow[d, "\mathsf{in}=[\underline{0}\ ,\mathsf{succ}]"] & \mathbb{N}_0 \arrow[l, "i_2"'] \arrow[dl, "\mathsf{succ}"] \\
& \mathbb{N}_0 &
\end{tikzcd} \tag{F1}$$

   Sabendo que o tipo 1 coincide com o tipo $()$ em Haskell e é habitado por um único elemento, também designado por $()$, calcule a inversa de in,

   Knowing that type 1 *matches type* $()$ *in Haskell and is inhabited by a single element, also denoted by* $()$*, find the inverse of* in*,*

$$\begin{array}{l}
\mathsf{out}\ 0 = i_1\ () \\
\mathsf{out}\ (n+1) = i_2\ n
\end{array}$$

   resolvendo em ordem a out a equação

   by solving the equation

$$\mathsf{out} \cdot \mathsf{in} = id \tag{F2}$$

   e introduzindo variáveis.

   for out *and adding variables.*

---

2. Na sequência da questão anterior, considere a equação em f

   As follow up to the previous question, consider the equation in f

$$f \cdot [\underline{0}\ , \mathsf{succ}] = [\underline{0}\ , \mathsf{add}] \cdot (id + \langle odd, f \rangle) \tag{F3}$$

a que corresponde o diagrama

captured by diagram

$$
\begin{tikzcd}
\mathbb{N}_0 \arrow[d, "f"'] & 1 + \mathbb{N}_0 \arrow[l, "{[\underline{0}\ ,\mathsf{succ}]}"'] \arrow[d, "id + \langle odd, f \rangle"] \\
\mathbb{N}_0 & 1 + \mathbb{N}_0 \times \mathbb{N}_0 \arrow[l, "{[\underline{0}\ ,\mathsf{add}]}"]
\end{tikzcd}
$$

onde $\mathsf{add}\ (x, y) = x + y$ e $odd\ n = 2\ n + 1$. Use as leis do cálculo de programas para mostrar que a única solução para essa equação é a função:

where $\mathsf{add}\ (x, y) = x + y$ *and* $odd\ n = 2\ n + 1$. *Show that the solution to this equation is the function (in Haskell syntax):*

$$
\left\{ \begin{array}{l} f\ 0 = 0 \\ f\ (n + 1) = (2\ n + 1) + f\ n \end{array} \right.
$$

"Corra"mentalmente a função f para vários casos, eg. $f\ 0$, $f\ 1$, $f\ 2$ e responda: o que faz (ou parece fazer) a função f? (A sua resposta, informal para já, será formalmente validada mais para a frente.)

"Run" function f mentally for various inputs, e.g. $f\ 0$, $f\ 1$, $f\ 2$ and answer: what does function f do (or seems *to do)? (Your answer, informal for now, will be formally validated later on.)*

---

3. Deduza o tipo mais geral da função $\alpha = (id + \pi_1) \cdot i_2 \cdot \pi_2$ e represente-o através de um diagrama.

Infer the most general type of function $\alpha = (id + \pi_1) \cdot i_2 \cdot \pi_2$ and draw it in a diagram of compositions.

---

4. No cálculo de programas, as definições condicionais do tipo

Conditional expressions of pattern

$$h\ x = \textbf{if}\ p\ x\ \textbf{then}\ f\ x\ \textbf{else}\ g\ x$$

são escritas usando o combinador ternário

are expressed in the algebra of programming by the ternary combinator

$$p \rightarrow f, g$$

conhecido pelo nome de *condicional de McCarthy*, cuja definição

known as the McCarthy conditionald*, whose definition*

$$p \rightarrow f, g \quad = \quad [f\ , g] \cdot p? \tag{F4}$$

vem no formulário. Baseie-se em leis desse formulário para demonstrar a chamada 2ª-lei de fusão do condicional:

can be found in reference sheet. Use this reference sheet to prove the so-called 2nd fusion-law of conditionals:

$$(p \rightarrow f, g) \cdot h \quad = \quad (p \cdot h) \rightarrow (f \cdot h), (g \cdot h)$$

---

5. Sabendo que as igualdades

Assuming

$$p \rightarrow k, k \ = \ k \tag{F5}$$

$$(p? + p?) \cdot p? \ = \ (i_1 + i_2) \cdot p? \tag{F6}$$

se verificam, demonstre as seguintes propriedades do mesmo combinador:

prove the following laws of the McCarthy conditional:

$$\langle (p \rightarrow f, h), (p \rightarrow g, i) \rangle \;=\; p \rightarrow \langle f, g\rangle, \langle h, i\rangle \tag{F7}$$

$$\langle f, (p \rightarrow g, h) \rangle \;=\; p \rightarrow \langle f, g\rangle, \langle f, h\rangle \tag{F8}$$

$$p \rightarrow (p \rightarrow a, b), (p \rightarrow c, d) \;=\; p \rightarrow a, d \tag{F9}$$

---

6. Considere a seguinte declaração de um tipo de *árvores binárias*, em Haskell:

Consider the following definition in Haskell of a particular type of binary tree*:*

$$\mathbf{data}\ \mathsf{LTree}\ a = \mathit{Leaf}\ a \mid \mathit{Fork}\ (\mathsf{LTree}\ a, \mathsf{LTree}\ a)$$

Indagando os tipos dos construtores Leaf e Fork, por exemplo no GHCi,

By querying the types of constructors Leaf and Fork in GHCi, for example,

```
*LTree> :t Fork
Fork :: (LTree a, LTree a) -> LTree a
*LTree> :t Leaf
Leaf :: a -> LTree a
```

faz sentido definir a função que mostra como construir árvores deste tipo:

one can define

$$\mathsf{in} = [\mathit{Leaf}\ , \mathit{Fork}] \tag{F10}$$

Desenhe um diagrama semelhante a (F1) para esta função e calcule a sua inversa

capturing how data of this type are built. Draw a diagram for this function similar to (F1) and find its inverse,

$$\mathsf{out}\ (\mathit{Leaf}\ a) = i_1\ a$$
$$\mathsf{out}\ (\mathit{Fork}\ (x, y)) = i_2\ (x, y)$$

de novo resolvendo a equação $\mathsf{out} \cdot \mathsf{in} = id$ em ordem a out, agora para o (F10). Finalmente, faça testes em Haskell que involvam a composição $\mathsf{in} \cdot \mathsf{out}$ e tire conclusões.

again solving the equation $\mathsf{out} \cdot \mathsf{in} = id$ *for* out*, but now with respect to (F10). Finally, run tests in Haskell involving the composition* $\mathsf{in} \cdot \mathsf{out}$ *and draw conclusions.*