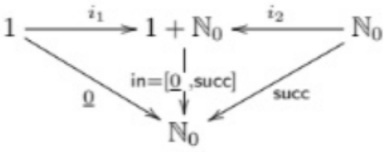
Ficha 4

1. Considere a função 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 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:*



(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,

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

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.*

1- out · in = id ⇔ out · [0, succ] = id ⇔ [out · 0, out · succ] = id

(17) (20) (1,72)

$$\Leftarrow \begin{cases} id \cdot i_1 = \mathsf{out} \cdot \underline{0} \\ id \cdot i_2 = \mathsf{out} \cdot \mathsf{succ} \end{cases} \Leftarrow \begin{cases} i_1 = \mathsf{out}\ 0 \\ i_2\ n = \mathsf{out} \cdot \mathsf{succ}\ n \end{cases} \Leftarrow$$

$$\Leftarrow \begin{cases} \mathsf{out}\ 0\ () = i_1\ () \\ \mathsf{out}\ (\mathsf{succ}\ n) = i_2\ n \end{cases} \Leftarrow \begin{cases} \mathsf{out}\ 0\ () = i_1\ () \\ \mathsf{out}\ (n+1) = i_2\ n \end{cases}$$

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

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

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

$$\begin{array}{ccc} \mathbb{N}_0 & \xleftarrow{[\underline{0}\,,\mathsf{succ}]} & 1 + \mathbb{N}_0 \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle id+\langle odd,f\rangle} \\ \mathbb{N}_0 & \xleftarrow[[\underline{0}\,,\mathsf{add}]]{} & 1 + \mathbb{N}_0 \times \mathbb{N}_0 \end{array}$$

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):*

$$\begin{cases} f\ 0 = 0 \\ f\ (n+1) = (2\ n+1) + f\ n \end{cases}$$

"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.)*

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

$\Leftarrow [f \cdot \underline{0}\ , f \cdot succ] = [\underline{0} \cdot id,\ add \cdot \langle odd, f \rangle]$ $\Leftarrow$ (20,22)

$\Leftarrow [f \cdot \underline{0}, f \cdot succ] = [\underline{0}, add \cdot \langle odd, f \rangle]$ (3,4)

$\Leftarrow \begin{cases} f\ 0 = 0 \\ f \cdot succ = add \cdot \langle odd, f \rangle \end{cases}$ (27)

$\Leftarrow \begin{cases} \\ f \cdot succ = add \cdot \langle odd, f \rangle \end{cases}$ (72)

$\Leftarrow \begin{cases} \\ (f \cdot succ)\ n = (add \cdot \langle odd, f \rangle)\ n \end{cases}$ (73)

$\Leftarrow \begin{cases} \\ f\ (succ\ n) = add\ (\langle odd, f \rangle\ n) \end{cases}$ (77)

$(\Leftrightarrow)$ $\begin{cases} \overline{\qquad\qquad} \\ f\,(n+1) = add\,(odd\ n, f\ n) \end{cases}$ $(\Leftrightarrow)$

$(\Leftrightarrow)$ $\begin{cases} f\ 0 = 0 \\ f\,(n+1) = 2n+1 + f\ n \end{cases}$

$f(1) = 1 + 0 = 1$ — 1×1

$f(2) = 4$ — 2×2

$f(3) = 9$ — 3×3

$f(4) = 16$ — 4×4

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.

3-

$\alpha = (id + \pi_1) \cdot i_2 \cdot \pi_2$

$A \xleftarrow{\pi_1} A \times (B \times C) \xrightarrow{\pi_2} B \times C$

$\downarrow i_2$

$D + B \times C$

$\downarrow id + \pi_1$

$D + B$

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

$$p \to f, g$$

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

known as the McCarthy conditionald, *whose definition*

$$p \to f, g \ = \ [f\ , g] \cdot p? \qquad \text{(F4)}$$

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:

$$(p \to f, g) \cdot h \ = \ (p \cdot h) \to (f \cdot h), (g \cdot h)$$

4- $(p \to f, g) \cdot h \overset{(30)}{=} ([f, g] \cdot p?) \cdot h \overset{(2)}{=}$

$= [f, g] \cdot (p? \cdot h) \overset{(29)}{=} [f, g] \cdot (h + h) \cdot (p \cdot h)? \overset{(22)}{=}$

$= [f \cdot h, g \cdot h] \cdot (p \cdot h)? \overset{(30)}{=}$

$= (p \cdot h) \to (f \cdot h), (g \cdot h)$

5. Sabendo que as igualdades

Assuming

$$p \to k, k \ = \ k \qquad \text{(F5)}$$
$$(p? + p?) \cdot p? \ = \ (i_1 + i_2) \cdot p? \qquad \text{(F6)}$$

se verificam, demonstre as seguintes propriedades do mesmo combinador:

prove the following laws of the McCarthy conditional:

$$\langle (p \to f, h), (p \to g, i) \rangle \ = \ p \to \langle f, g \rangle, \langle h, i \rangle \qquad \text{(F7)}$$
$$\langle f, (p \to g, h) \rangle \ = \ p \to \langle f, g \rangle, \langle f, h \rangle \qquad \text{(F8)}$$
$$p \to (p \to a, b), (p \to c, d) \ = \ p \to a, d \qquad \text{(F9)}$$

- $\langle (p \to f, h), (p \to g, i) \rangle =$ (30)

$\langle \underbrace{[f,h]}_{g} \cdot \underbrace{p?}_{f}, \underbrace{[g,i]}_{h} \cdot \underbrace{p?}_{f} \rangle \overset{(9)}{=} \langle [f,h],[g,i] \rangle \cdot p? \overset{(28)}{=}$

$= [\underbrace{\langle f,g \rangle}_{f}, \underbrace{\langle h,i \rangle}_{g}] \cdot p? \overset{(30)}{=} p \to \langle f,g \rangle, \langle h,i \rangle$

- $\langle f, (p \to g, h) \rangle = \overbrace{\langle (p \to f, f), (p \to g, h) \rangle}^{(F5)}$ (30)

$= \langle [f,f] \cdot p?, [g,h] \cdot p? \rangle \overset{(9)}{=} \langle [f,f],[g,h] \rangle \cdot p? =$ (28)

$= [\langle f,g \rangle, \langle f,h \rangle] \cdot p? \overset{(30)}{=} p \to \langle f,g \rangle, \langle f,h \rangle]$

- $p \to (\underbrace{p \to a, b}_{f}), (\underbrace{p \to c, d}_{g}) =$ (30)

$= [p \to a, b, p \to c, d] \cdot p? \overset{(30)}{=} [\overbrace{[a,b] \cdot p?}^{g}, \overbrace{[c,d] \cdot p?}^{h}] \cdot p?$

(22)

$= [[a,b],[c,d]] \cdot (p! + p?) \cdot p?$ (FG)

$= [[a,b],[c,d]] \cdot (i_1 + i_2) \cdot p?$ (22)

$= [[a,b] \cdot i_1, [c,d] \cdot i_2] \cdot p? =$ (18)

$= [a, d] \cdot p? \overset{(30)}{=} p \to a, d$

c.q.d

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*:*

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

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

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

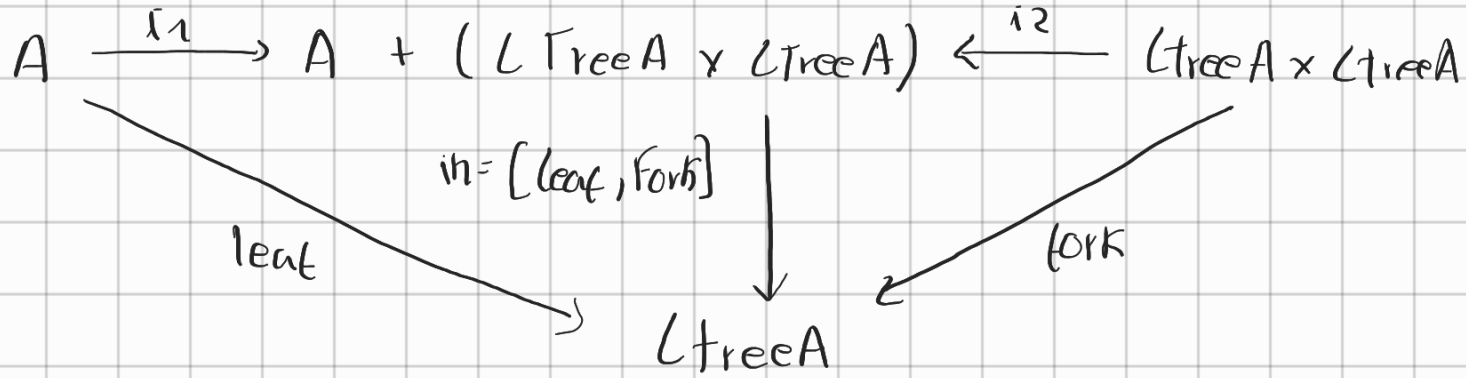
Desenhe um diagrama semelhante a (F1) para esta função e calcule a sua inversa

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

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

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 in · 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* in · out *and draw conclusions.*



$$\mathsf{LTree}\,A \xrightarrow{\mathsf{out}} A + (\mathsf{LTree}\,A)^2$$

$$\mathsf{out} \cdot \mathsf{in} = id \equiv \mathsf{out} \cdot [\mathit{Leaf}, \mathit{Fork}] = id \equiv \text{(20)}$$

$$
\begin{aligned}
&\Leftarrow [out.leaf, out.Fork] = id \quad \Leftarrow (17)\\
&\Leftarrow \begin{cases} id \cdot i_1 = out.leaf \\ id \cdot i_2 = out.Fork \end{cases} \overset{(1)}{\Leftarrow} \begin{cases} i_1 = out.leaf \\ i_2 = out.Fork \end{cases} \overset{(72)}{\Leftarrow}\\
&\Leftarrow \begin{cases} \forall x \mid (out.leaf(x)) = i_1\, x \\ \forall x,y \mid (out.Fork(x,y)) = i_2(x,y) \end{cases} \overset{(73)}{\Leftarrow} \begin{cases} out\,(leaf\, x) = i_1\, x \\ out\,(Fork(x,y)) = i_2(x,y) \end{cases}
\end{aligned}
$$