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

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

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

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

é fácil desenhar o diagrama que explica a construção da função

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

Desenhe-o e calcule a sua inversa

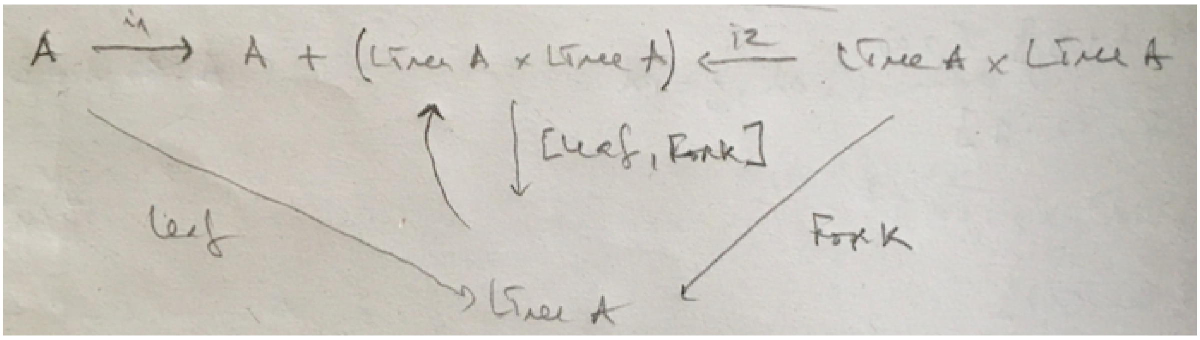
$$\mathsf{out} :: \mathsf{LTree}\ a \to a + \mathsf{LTree}\ a \times \mathsf{LTree}\ a$$
$$\mathsf{out}\ (\mathit{Leaf}\ \ a) = i_1\ a$$
$$\mathsf{out}\ (\mathit{Fork}\ (x, y)) = i_2\ (x, y)$$

resolvendo a equação

$$\mathsf{out} \cdot \mathsf{in} \quad = \quad id$$

em ordem a out.

Finalmente, faça testes em Haskell que involvam a composição in · out. Que "conclusão"tira desses testes?[2]



Resolução

$out \cdot in = id$

{ substituindo in e id por: $in = [Leaf, Fork], id = [i_1, i_2]$ ***(lei 19)*** *}*

$out \cdot [Leaf, Fork] = [i_1, i_2]$

*{**lei (20)** }*

$[out \cdot Leaf, out \cdot Fork] = [i_1, i_2]$

*{**lei (27)** }*

$$out \cdot Leaf = i_1 \tag{1}$$
$$out \cdot Fork = i_2 \tag{2}$$

*{**lei (71)** }*

$$\begin{aligned}(out \cdot Leaf)\ a &= i_1\ a \\ (out \cdot Fork)\ (a,b) &= i_2\ (a,b)\end{aligned}$$

*{**lei (72)** }*

$$\begin{aligned}out\ (Leaf\ a) &= i_1\ a \\ out\ (Fork\ (a,b)) &= i_2\ (a,b)\end{aligned}$$

In [1]:
```
data LTree a = Leaf a | Fork (LTree a, LTree a) deriving Show
```

In [2]:
```
-- type checking

:t Fork
:t Leaf
```

```
Fork :: forall a. (LTree a, LTree a) -> LTree a
Leaf :: forall a. a -> LTree a
```

In [22]:
```
fIN = either Leaf Fork
```

In [23]:
```
fOUT (Leaf a) = Left a
fOUT (Fork a) = Right a
```

In [24]:
```
-- type checking

:t (fOUT . fIN)
:t (fIN . fOUT)
```

```
(fOUT . fIN) :: forall a. Either a (LTree a, LTree a) -> Either a
(LTree a, LTree a)
(fIN . fOUT) :: forall a. LTree a -> LTree a
```

Conclusão

$in \cdot out = id_{LTree}$