4. Considere a seguinte definição, em Haskell, de um tipo para descrever pontos no espaço tridimensional:

$$\textbf{data}\ Point\ a = Point\ \{x :: a, y :: a, z :: a\}\ \textbf{deriving}\ (Eq, Show)$$

Considere a seguinte formulação da correspondência entre a sintaxe concreta acima, em Haskell, e a correspondente sintaxe abstracta

$$\begin{tikzcd} Point\ A \arrow[rr, bend left, "\mathsf{out} = \langle\langle x,y\rangle,z\rangle"] \arrow[rr, phantom, "\cong"] & & (A \times A) \times A \arrow[ll, bend left, "\mathsf{in} = \widehat{\widehat{Point}}"] \end{tikzcd}$$

onde $\widehat{f}$ abrevia uncurry f. Preencha os "?" na seguinte alternativa à situação anterior:

$$\begin{tikzcd} Point\ A \arrow[rr, bend left, "\mathsf{out} = ?"] \arrow[rr, phantom, "\cong"] & & ? \arrow[ll, bend left, "\mathsf{in} = \widehat{flip\ \widehat{Point}}"] \end{tikzcd}$$

NB: peça ajuda ao GHCi na inferência dos tipos em causa.

Haskell

In [1]:

```haskell
-- loading Cp.hs

:load ../src/Cp.hs
```

In [2]:

```haskell
data Point a = Point { x :: a , y :: a , z :: a } deriving (Eq , Show )

:t uncurry (uncurry Point)
```

```
uncurry (uncurry Point) :: forall a. ((a, a), a) -> Point a
```

Resolução

Vamos começar por verificar o tipo de $in = \widehat{flip\ \widehat{Point}}$

$in = \widehat{flip\ \widehat{Point}}$

{ def. $flip$, $flip\ f = \overline{\widehat{f}\,.\,swap}$ }

$= \widehat{\overline{\widehat{\widehat{Point}}.\,swap}}$

{ uncurry . curry = id }

$= \widehat{\widehat{Point}}.\,swap$

Sabemos, pelo primeiro diagrama, que

$$(A \times A) \times A \xrightarrow{\widehat{Point}} Point A$$

Sabemos ainda, pelo segundo diagrama, que

$$? \xrightarrow{\widehat{Point}.swap} Point A$$

Ou seja, pela definição de $swap$ podemos então concluir que $? \;=\; A \times (A \times A)$.

E quanto a $out = \;?$

Analisando os dois diagramas vemos que:

- [Primeiro diagrama]: $Point A \xrightarrow{out=<<x,y>,z>} (A \times A) \times A$
- [Segundo diagrama]: $Point A \xrightarrow{out=?} A \times (A \times A)$

Facilmente se conclui que no segundo diagrama $out = swap\,.<< x, y >, z > \;=\; < z, < x, y >>$, ou seja, temos finalmente:

- [Primeiro diagrama]: $Point A \xrightarrow{out=<<x,y>,z>} (A \times A) \times A$
- [Segundo diagrama]: $Point A \xrightarrow{out=<z,<x,y>>} A \times (A \times A)$

Haskell

In [3]:

```
out' = split (split x y) z

in' = uncurry (uncurry Point)

-- type checking

:t out'
:t in'
```

```
out' :: forall c. Point c -> ((c, c), c)
in' :: forall a. ((a, a), a) -> Point a
```

In [4]:

```
out'' = split z (split x y)

in'' = uncurry (flip (uncurry Point))

-- type checking

:t out''
:t in''
```

```
out'' :: forall c. Point c -> (c, (c, c))
in'' :: forall a. (a, (a, a)) -> Point a
```

In [5]:

```
-- checking with ((2,3),4) and (2,(3,4))

out' . in' $ ((2,3),4)
out'' . in'' $ (2,(3,4))
```

```
((2,3),4)
(2,(3,4))
```

In [6]:

```
-- checking with p = Point 2 3 4

p = Point 2 3 4

in' . out' $ p
in'' . out'' $ p
```

```
Point {x = 2, y = 3, z = 4}
Point {x = 2, y = 3, z = 4}
```