4. Converta a função vars do exercício 2 numa função com variáveis em Haskell sem quaisquer combinadores *pointfree*.

In [1]:

```
-- loading Cp.hs
:opt no-lint
:load ../src/Cp.hs
:set -XNPlusKPatterns
```

Resolução

Seja $\alpha = vars = (\!|g|\!), g = [singl, concat . p2], in = [Var, Op]$.

Temos então:

$$\alpha . [Var, Op] = [singl, concat . p2] . (id + id \times map\ \alpha)$$

{ fusão-$+$; absorção-$+$ }

$$[\alpha . Var, \alpha . Op] = [singl . id, concat . p2 . (id \times map\ \alpha)]$$

{ eq-$+$; natural-id }

$$\alpha . Var = singl$$

$$\alpha . Op = concat . p2 . (id \times map\ \alpha)$$

{ pointwise; def-comp }

$$\alpha\ (Var\ v) = singl\ v$$

$$\alpha\ (Op\ (o, l)) = concat\ (p2\ (o, map\ \alpha\ l))$$

{ def. π_2 }

$$\alpha\ (Var\ v) = singl\ v$$

$$\alpha\ (Op\ (o, l)) = concat\ (map\ \alpha\ l)$$

In [2]:

```
data Expr v o = Var v | Op (o, [Expr v o]) deriving (Show)

vars (Var v) = singl $ v
vars (Op (o,l)) = concat $ map vars l

:t vars
```

```
vars :: forall a o. Expr a o -> [a]
```

In [3]:

```
x = Op ("+",[Var "a",Var "b", Op("*",[Var "x", Var "y"])])
vars x
```

```
["a","b","x","y"]
```