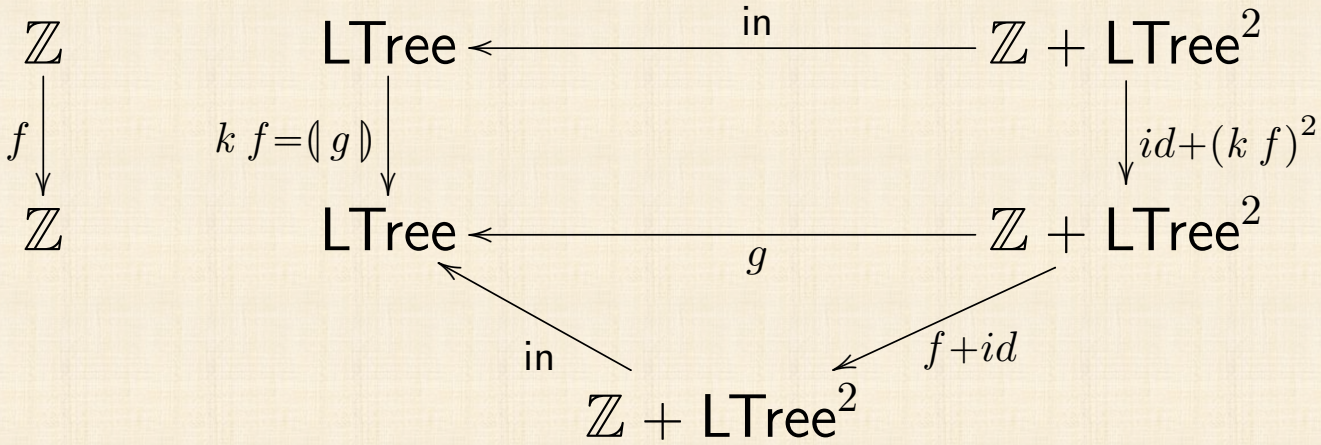
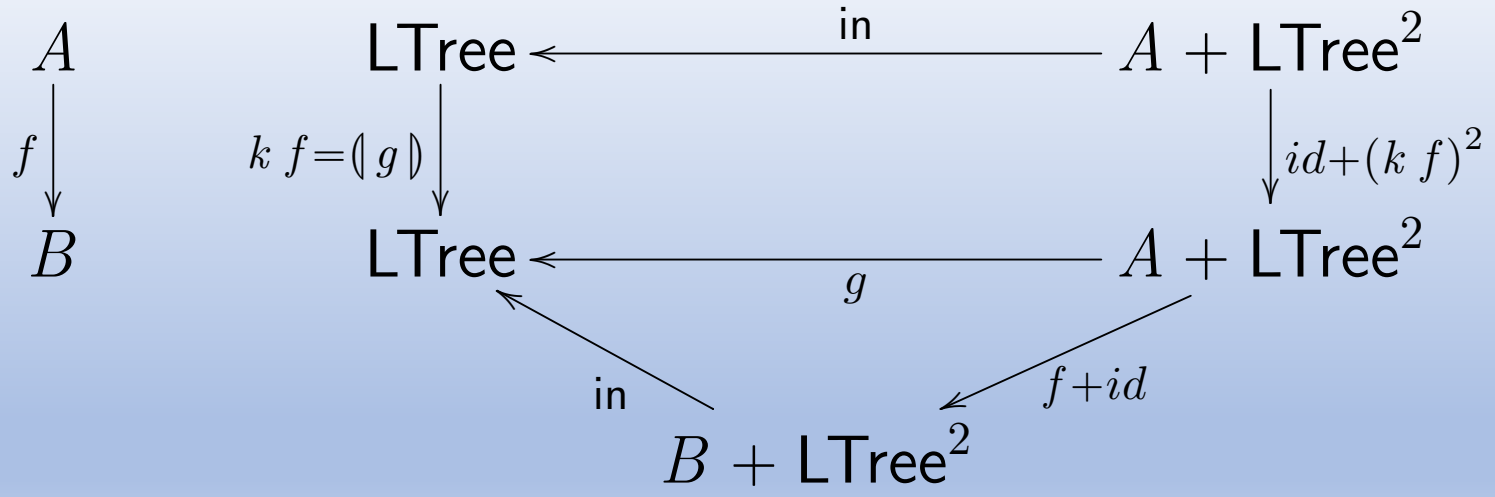
Cálculo de Programas
T09 (cntd)

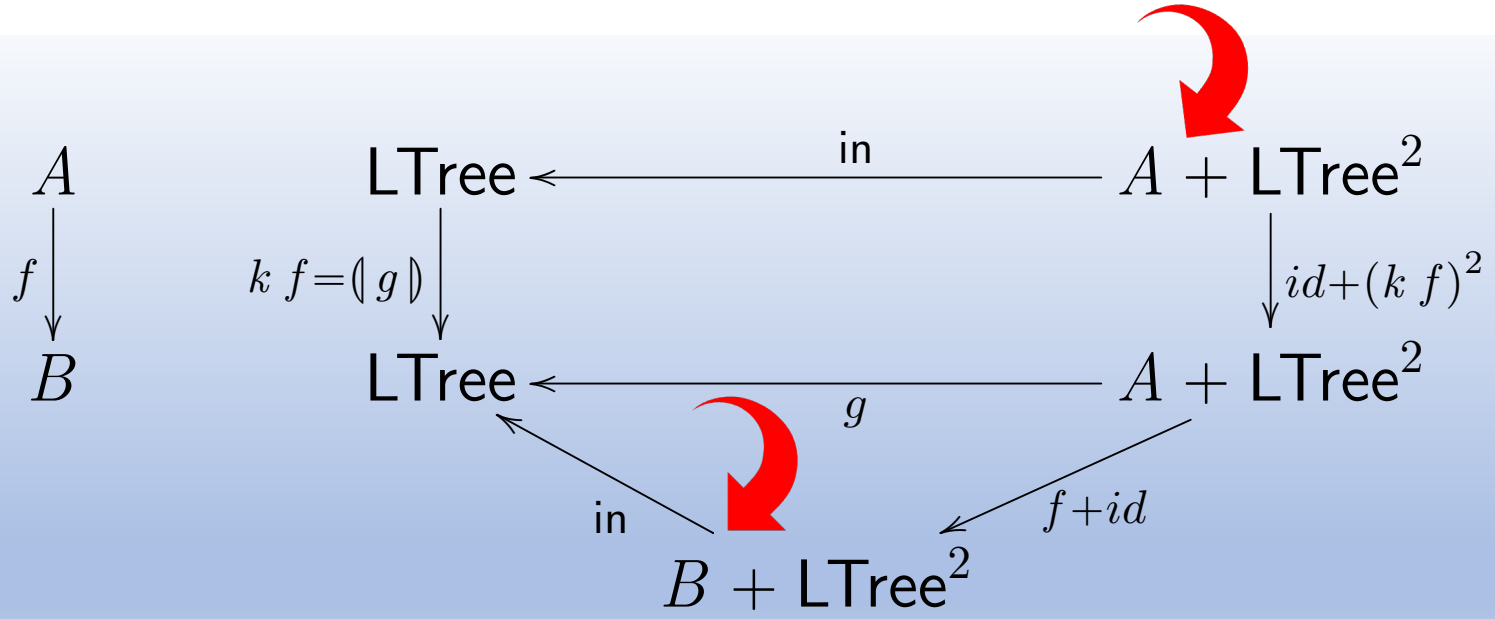
$$k\ f \cdot \mathsf{in} = \mathsf{in} \cdot (f + id) \cdot (id + k\ f \times k\ f)$$



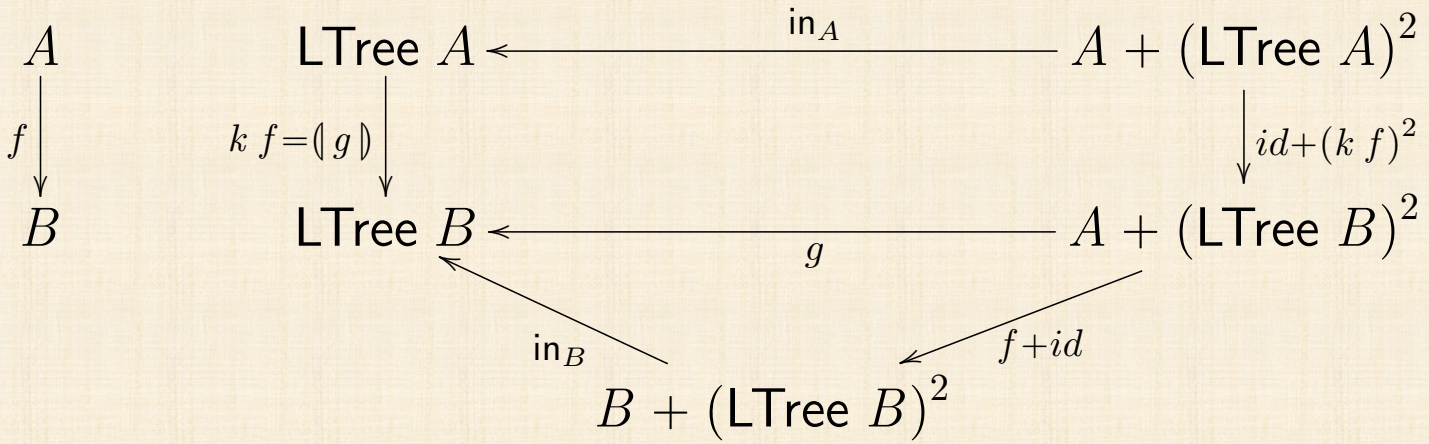
$$k\ f \cdot \mathsf{in} = \mathsf{in} \cdot (f + id) \cdot (id + k\ f \times k\ f)$$



$$k\ f \cdot \mathsf{in} = \mathsf{in} \cdot (f + id) \cdot (id + k\ f \times k\ f)$$



$$k\ f = (\!|\ \mathsf{in} \cdot (f + id)\ |\!)$$



$$k\ f = (\!|\ \mathsf{in} \cdot (f + id)\ |\!)$$

$$k\ f = (\!|\ \mathsf{in} \cdot (f + id)\ |\!)$$

$$k\ id = (\!|\ \mathsf{in}\ |\!) = id$$

Absorption:

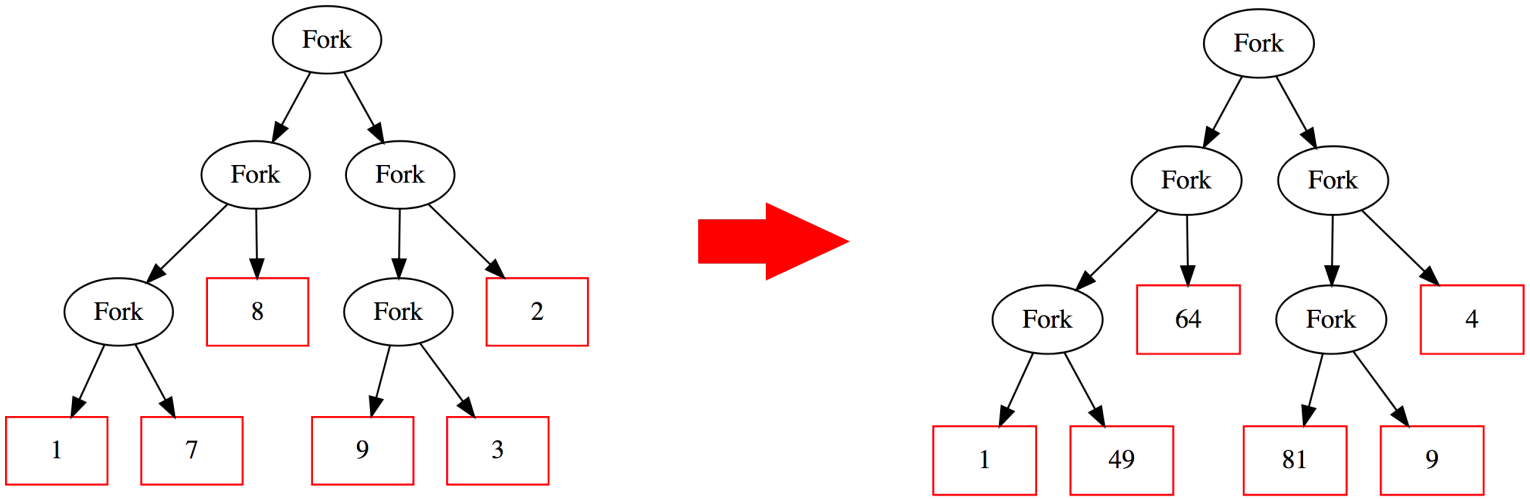
$$(\!|\ g\ |\!) \cdot k\ f = (\!|\ g \cdot (f + id)\ |\!)$$

$$
\begin{aligned}
& (\!|\, g \,|\!) \cdot k\, f = (\!|\, g \cdot (f + id) \,|\!) \\
\equiv & \qquad \{\ k\, f = (\!|\, \mathsf{in} \cdot (f + id) \,|\!)\ \} \\
& (\!|\, g \,|\!) \cdot (\!|\, \mathsf{in} \cdot (f + id) \,|\!) = (\!|\, g \cdot (f + id) \,|\!) \\
\Leftarrow & \qquad \{\ \text{cata-fusion}\ \} \\
& (\!|\, g \,|\!) \cdot \mathsf{in} \cdot (f + id) = g \cdot (f + id) \cdot (id + (\!|\, g \,|\!)^2) \\
\equiv & \qquad \{\ \text{cata-cancellation}\ \} \\
& g \cdot (id + (\!|\, g \,|\!)^2) \cdot (f + id) = g \cdot (f + id) \cdot (id + (\!|\, g \,|\!)^2) \\
\equiv & \qquad \{\ \text{functor-}+ \text{ twice; natural-}id \text{ four times}\ \} \\
& g \cdot (f + (\!|\, g \,|\!)^2) = g \cdot (f + (\!|\, g \,|\!)^2) \\
\equiv & \qquad \{\ \text{trivial}\ \} \\
& true
\end{aligned}
$$

$$
\begin{aligned}
& k\ (f \cdot g) \\
= & \quad \{\ k\ f = (\!|\ \mathsf{in} \cdot (f + id)\ |\!)\ \} \\
& (\!|\ \mathsf{in} \cdot (f \cdot g + id)\ |\!) \\
= & \quad \{\ +\text{-functor etc}\ \} \\
& (\!|\ \mathsf{in} \cdot (f + id) \cdot (g + id)\ |\!) \\
= & \quad \{\ \text{absorption (previous slides)}\ \} \\
& (\!|\ \mathsf{in} \cdot (f + id)\ |\!) \cdot (\!|\ \mathsf{in} \cdot (g + id)\ |\!) \\
= & \quad \{\ k\ f = (\!|\ \mathsf{in} \cdot (f + id)\ |\!)\ \text{twice}\ \} \\
& k\ f \cdot k\ g
\end{aligned}
$$

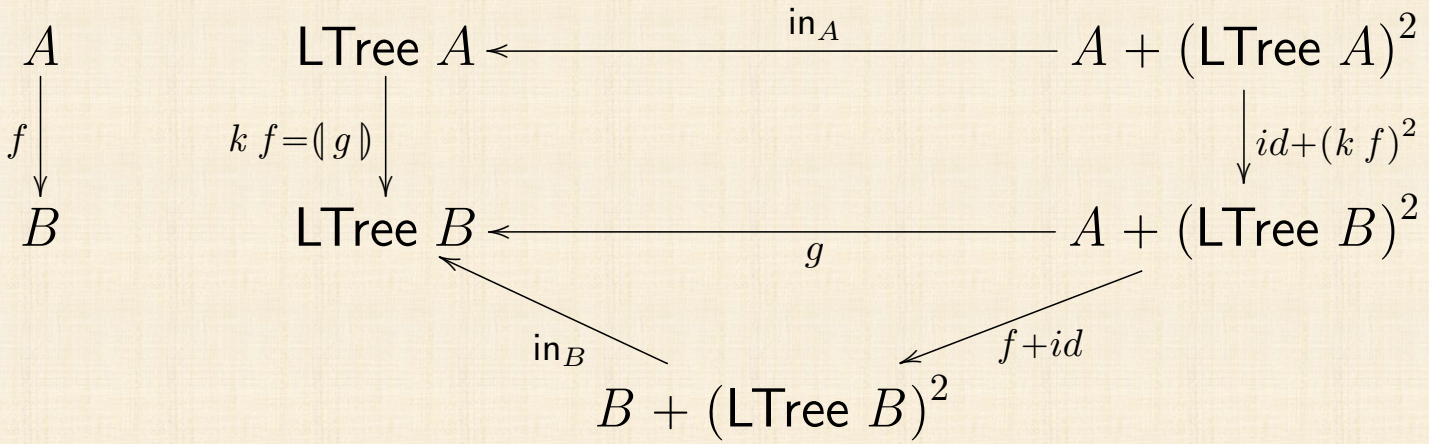
$$k\ id = (\!|\ \mathsf{in}\ |\!) = id$$

$$k\ (f \cdot g) = k\ f \cdot k\ g$$

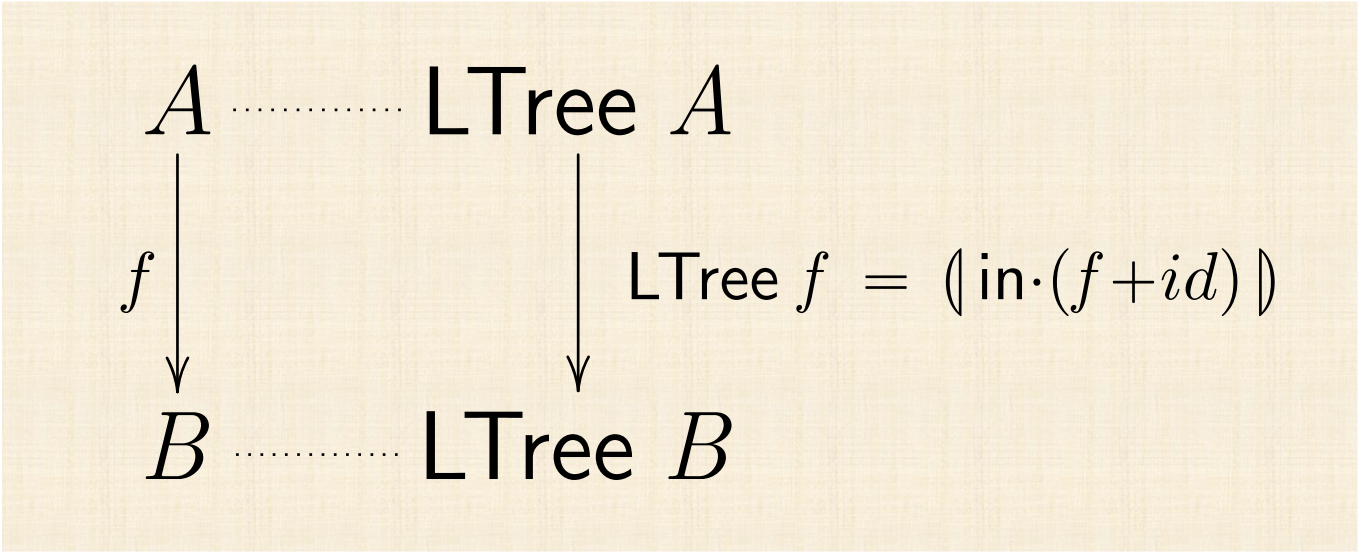



```
data LTree = Leaf Int | Fork (LTree, LTree)

k (Leaf x) = Leaf (x^2)
k (Fork (l,r)) = Fork (k l, k r)
```

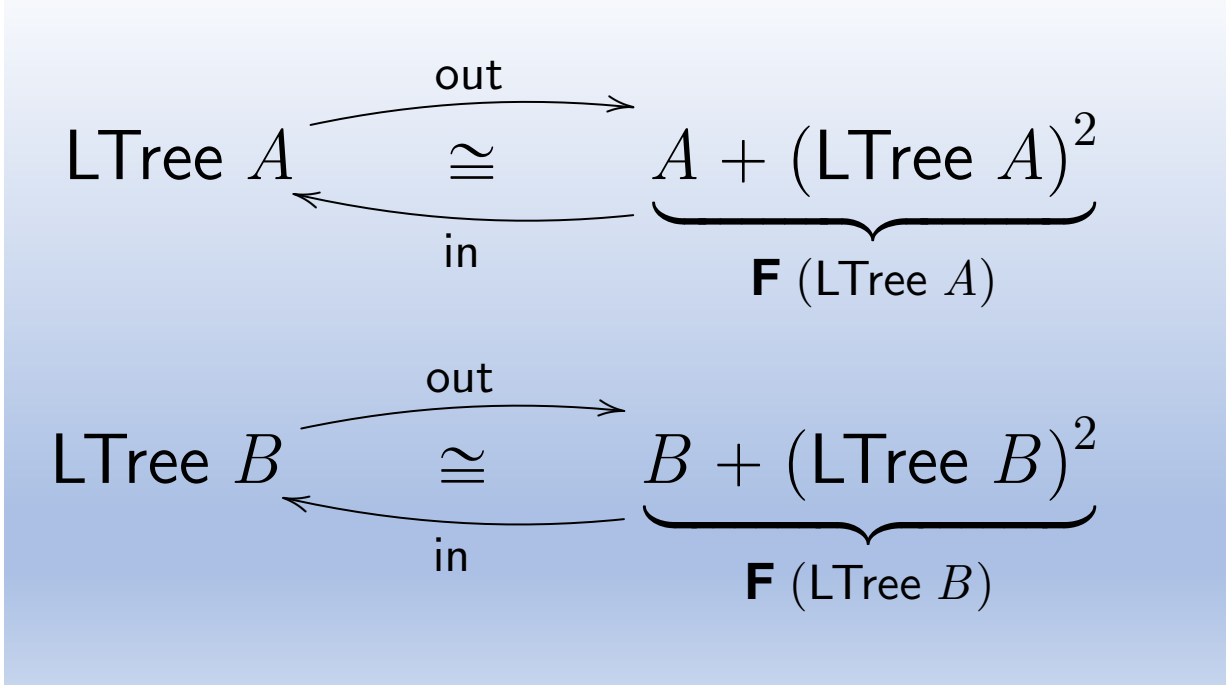
$$k\ f = (\!|\ \mathsf{in} \cdot (f + id)\ |\!)$$



LTREE TYPE FUNCTOR



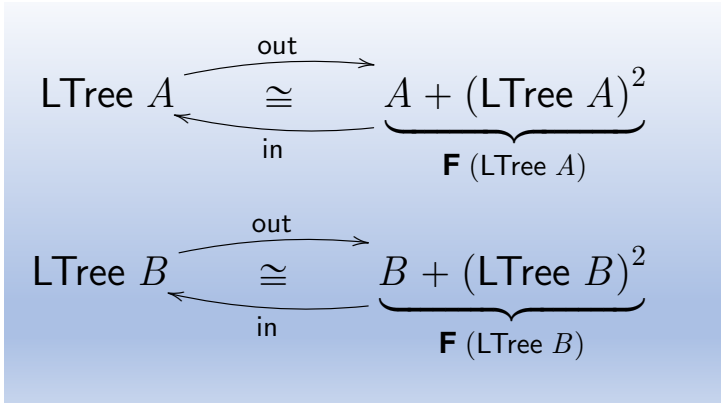
LTREE TYPE FUNCTOR



LTREE TYPE FUNCTOR

$$\mathbf{F}\ X = A + X^2 \quad ?$$

$$\mathbf{F}\ X = B + X^2 \quad ?$$

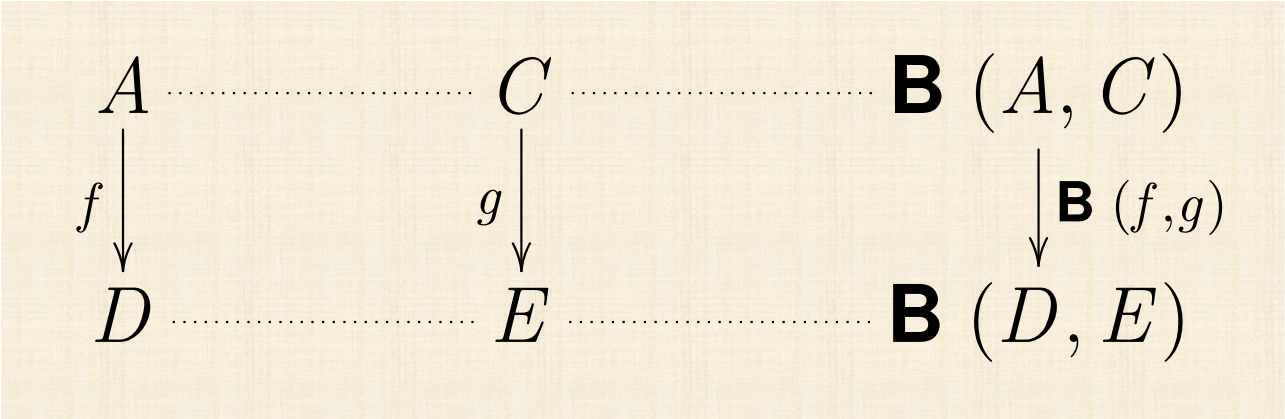


LTREE base bifunctor B

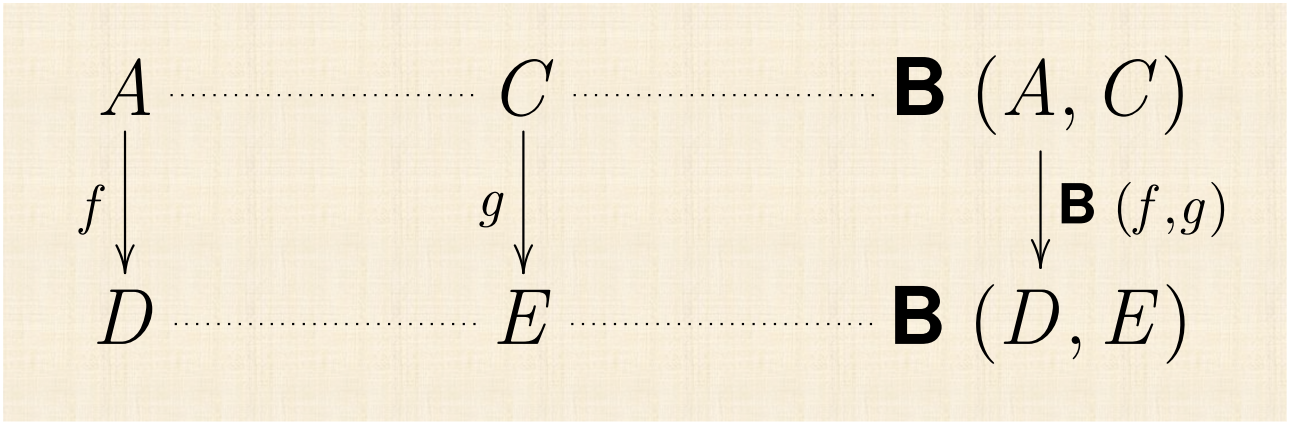
$$\mathsf{LTree}\ X \underset{\mathsf{in}}{\overset{\mathsf{out}}{\cong}} \underbrace{X + (\mathsf{LTree}\ X)^2}_{\mathbf{B}\ (X, \mathsf{LTree}\ X)}$$

$$\mathbf{B}\ (X, Y) = X + Y^2$$

BIFUNCTORS



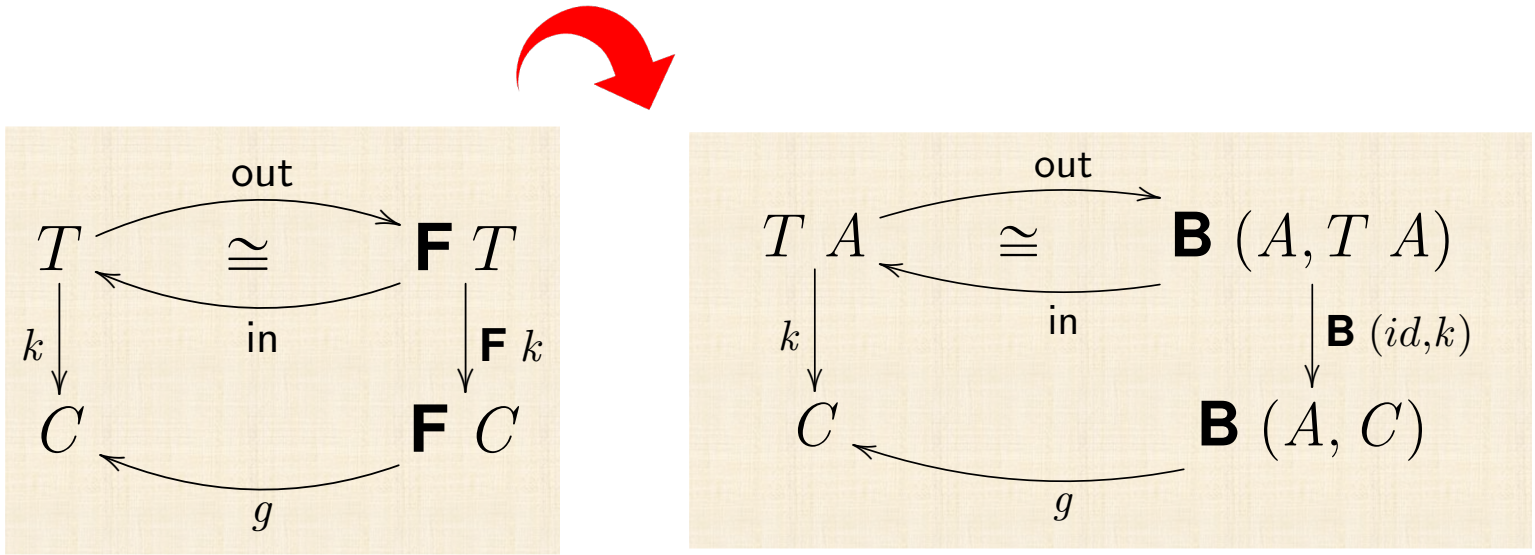
BIFUNCTORS (Laws)



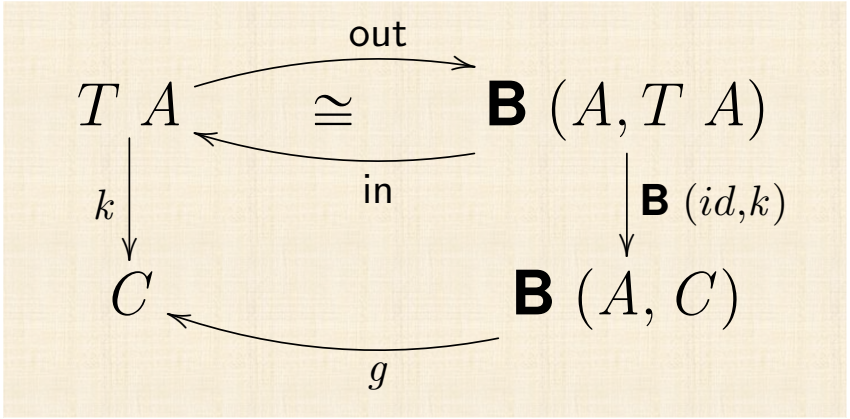
$$\mathbf{B}\,(id, id) = id$$

$$\mathbf{B}\,(h \cdot f, k \cdot g) = \mathbf{B}\,(h, k) \cdot \mathbf{B}\,(f, g)$$

CATAMORPHISMS (generalization)



CATAMORPHISMS (in general)



Universal property

$$k = (\!|\, g \,|\!) \ \Leftrightarrow\ k \cdot \mathsf{in} = g \cdot \mathbf{B}\ (id, k)$$

Type functor:

$$T\ f = (\!|\, \mathsf{in} \cdot \mathbf{B}\ (f, id) \,|\!)$$

Abbreviation:

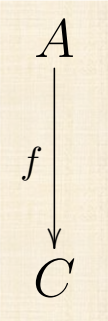
$$\mathbf{F}\ k = \mathbf{B}\ (id, k)$$

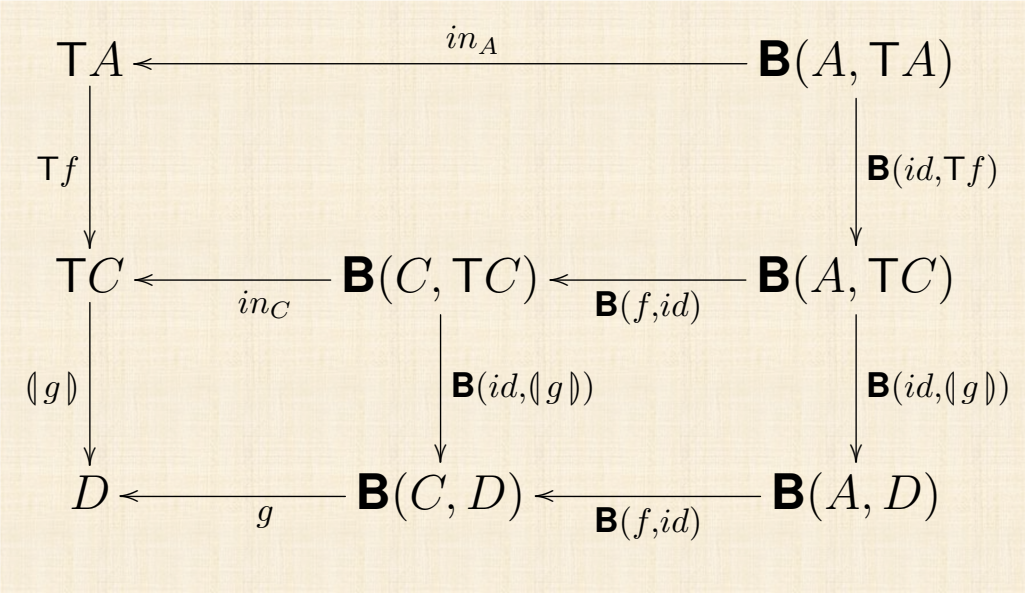
CATAMORPHISMS (Laws)

Universal-cata	$k = (\!\mid g \mid\!) \quad \Leftrightarrow \quad k \cdot \mathsf{in} = g \cdot \mathsf{F}\, k$	(43)
Cancelamento-cata	$(\!\mid g \mid\!) \cdot \mathsf{in} = g \cdot \mathsf{F}\, (\!\mid g \mid\!)$	(44)
Reflexão-cata	$(\!\mid \mathsf{in} \mid\!) = id_{\mathsf{T}}$	(45)
Fusão-cata	$f \cdot (\!\mid g \mid\!) = (\!\mid h \mid\!) \quad \Leftarrow \quad f \cdot g = h \cdot \mathsf{F}\, f$	(46)
Base-cata	$\mathsf{F}\, f \quad = \quad \mathsf{B}\ (id, f)$	(47)
Def-map-cata	$\mathsf{T}\, f \quad = \quad (\!\mid \mathsf{in} \cdot \mathsf{B}(f, id) \mid\!)$	(48)
Absorção-cata	$(\!\mid g \mid\!) \cdot \mathsf{T}\, f \quad = \quad (\!\mid g \cdot \mathsf{B}(f, id) \mid\!)$	(49)

Cata-absorption

$$(\!|\, g \,|\!) \cdot \mathsf{T} f = (\!|\, g \cdot \mathbf{B}\,(f, id) \,|\!)$$





(a) Trees whose data of type A are stored in their nodes:

$$\mathsf{T} = \mathsf{BTree}\ A \qquad \begin{cases} \mathsf{F}\ X = 1 + A \times X^2 \\ \mathsf{F}\ f = id + id \times f^2 \end{cases} \qquad \mathsf{in} = [\underline{Empty}\ , Node]$$

Haskell: $\mathbf{data}\ \mathsf{BTree}\ a = Empty \mid Node\ (a, (\mathsf{BTree}\ a, \mathsf{BTree}\ a))$

(b) Trees with data in their leafs :

$$\mathsf{T} = \mathsf{LTree}\ A \qquad \begin{cases} \mathsf{F}\ X = A + X^2 \\ \mathsf{F}\ f = id + f^2 \end{cases} \qquad \mathsf{in} = [Leaf\ , Fork]$$

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

(c) Full trees — data in both leaves and nodes:

$$\mathsf{T} = \mathsf{FTree}\ B\ A \qquad \begin{cases} \mathsf{F}\ X = B + A \times X^2 \\ \mathsf{F}\ f = id + id \times f^2 \end{cases} \qquad \mathsf{in} = [Unit\ , Comp]$$

Haskell: $\mathbf{data}\ \mathsf{FTree}\ b\ a = Unit\ b \mid Comp\ (a, (\mathsf{FTree}\ b\ a, \mathsf{FTree}\ b\ a))$

(d) Expression trees:

$$\mathsf{T} = \mathsf{Expr}\ V\ O \qquad \begin{cases} \mathsf{F}\ X = V + O \times X^* \\ \mathsf{F}\ f = id + id \times \mathsf{map}\ f \end{cases} \qquad \mathsf{in} = [Var\ , Op]$$

Haskell: $\mathbf{data}\ \mathsf{Expr}\ v\ o = Var\ v \mid Op\ (o, [\mathsf{Expr}\ v\ o])$

(a) Trees whose data of type A are stored in their nodes:

$$\mathsf{T} = \mathsf{BTree}\ A \qquad \begin{cases} \mathsf{B}\ (X, Y) = 1 + X \times Y^2 \\ \mathsf{B}\ (g, f) = id + g \times f^2 \end{cases} \qquad \mathsf{in} = [\underline{Empty}\ , Node]$$

Haskell: $\mathbf{data}\ \mathsf{BTree}\ a = Empty \mid Node\ (a, (\mathsf{BTree}\ a, \mathsf{BTree}\ a))$

(b) Trees with data in their leafs :

$$\mathsf{T} = \mathsf{LTree}\ A \qquad \begin{cases} \mathsf{B}\ (X, Y) = X + Y^2 \\ \mathsf{B}\ (g, f) = g + f^2 \end{cases} \qquad \mathsf{in} = [Leaf\ , Fork]$$

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

(c) Full trees — data in both leaves and nodes:

$$\mathsf{T} = \mathsf{FTree}\ B\ A \qquad \begin{cases} \mathsf{B}\ (Z, X, Y) = Z + X \times Y^2 \\ \mathsf{B}\ (h, g, f) = h + g \times f^2 \end{cases} \qquad \mathsf{in} = [Unit\ , Comp]$$

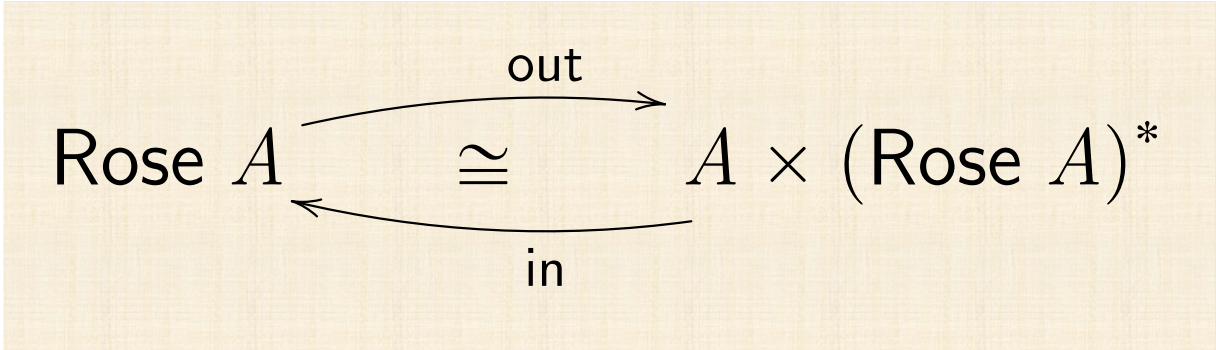
Haskell: $\mathbf{data}\ \mathsf{FTree}\ b\ a = Unit\ b \mid Comp\ (a, (\mathsf{FTree}\ b\ a, \mathsf{FTree}\ b\ a))$

(d) Expression trees:

$$\mathsf{T} = \mathsf{Expr}\ V\ O \qquad \begin{cases} \mathsf{B}\ (Z, X, Y) = Z + X \times Y^* \\ \mathsf{B}\ (h, g, f) = h + g \times \mathsf{map}\ f \end{cases} \qquad \mathsf{in} = [Var\ , Op]$$

Haskell: $\mathbf{data}\ \mathsf{Expr}\ v\ o = Var\ v \mid Op\ (o, [\mathsf{Expr}\ v\ o])$

```haskell
data Rose a = Rose a [Rose a] deriving Show

inRose = uncurry Rose

outRose (Rose a x) = (a,x)
```



$$\text{Rose } A \underset{\text{in}}{\overset{\text{out}}{\cong}} A \times (\text{Rose } A)^*$$

$$\mathbf{B}\,(X, Y) = X \times Y^*$$

$$\mathbf{B}\,(f, g) = f \times g^*$$

$$A \times Y^*$$

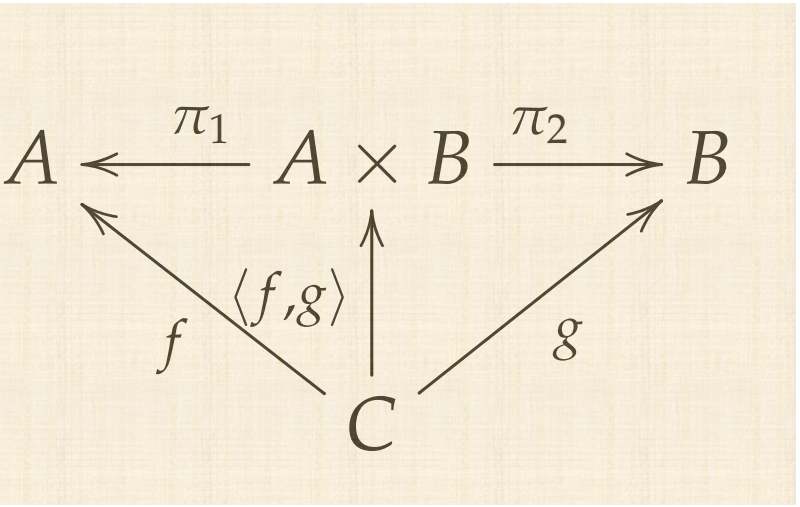
$$X \times Y^*$$

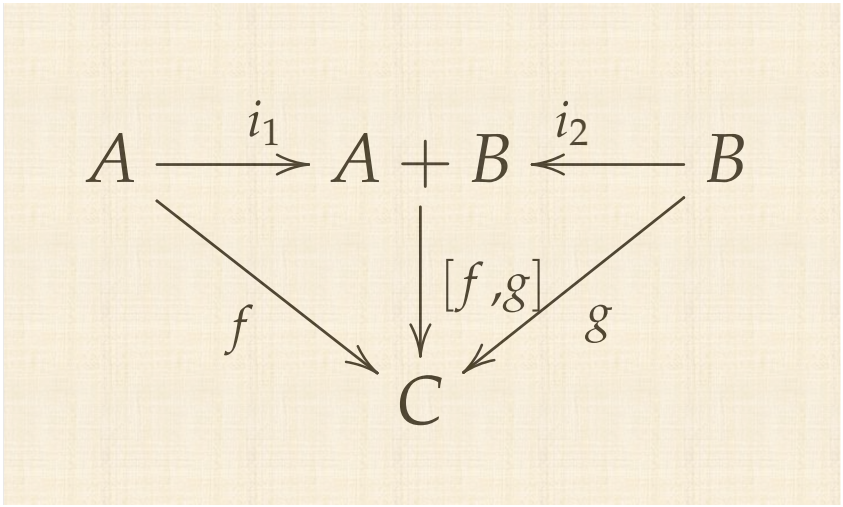
```haskell
data Rose a = Rose a [Rose a] deriving Show

inRose = uncurry Rose

outRose (Rose a x) = (a,x)
```

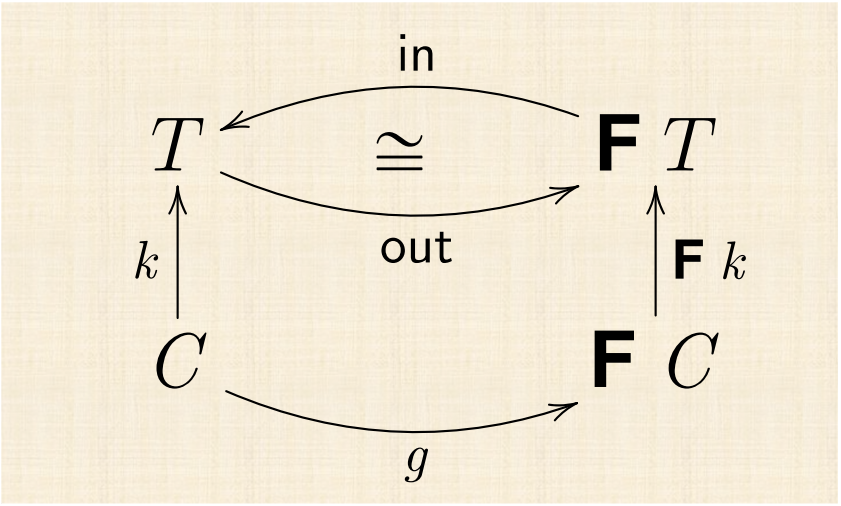
$$\mathbf{B}\,(X, Y) = X \times Y^*$$
$$\mathbf{B}\,(f, g) = f \times g^*$$

```haskell
f >< map g
```

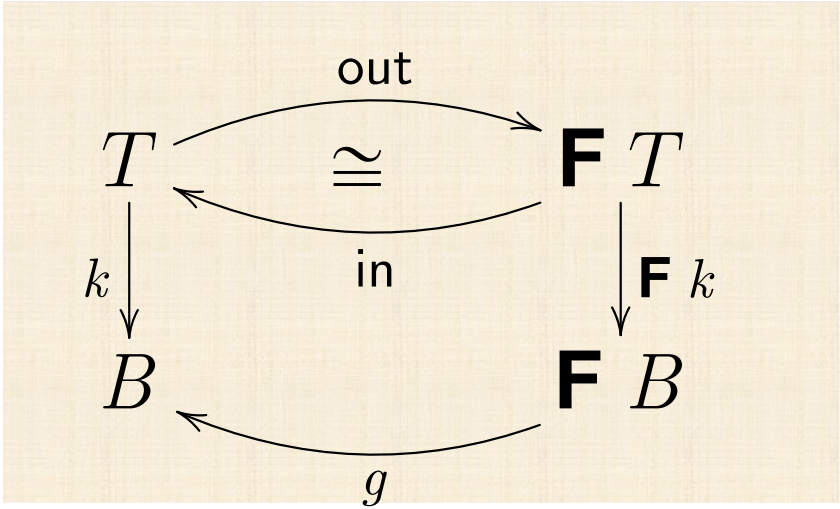
Cálculo de Programas
T10


A
π1
A × B
π2
B
⟨f,g⟩
f
g
C


A
i1
A + B
i2
B
[f,g]
f
g
C



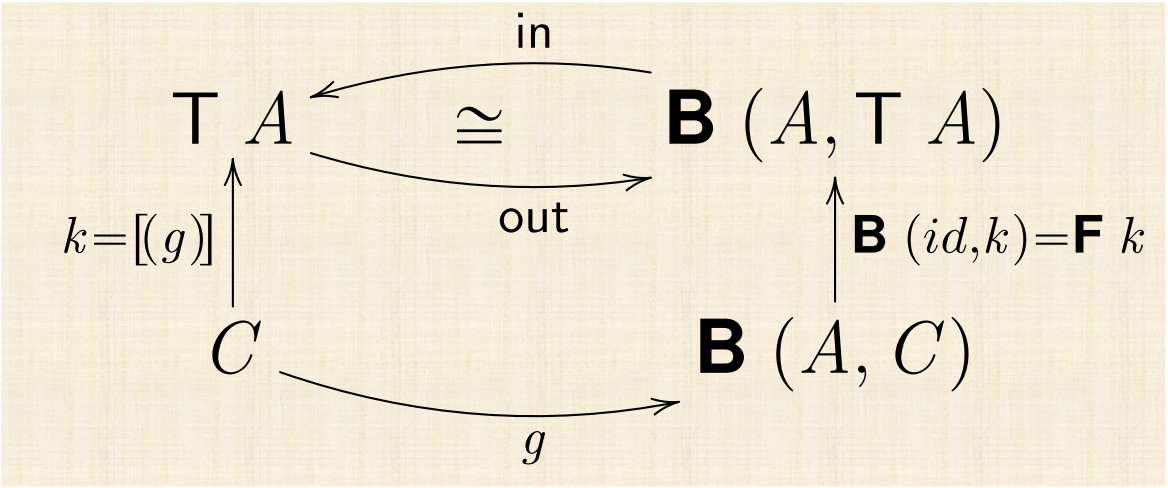
$\alpha\nu\alpha$ (ana)



$\kappa\alpha\tau\alpha$ (cata)

Anamorphisms

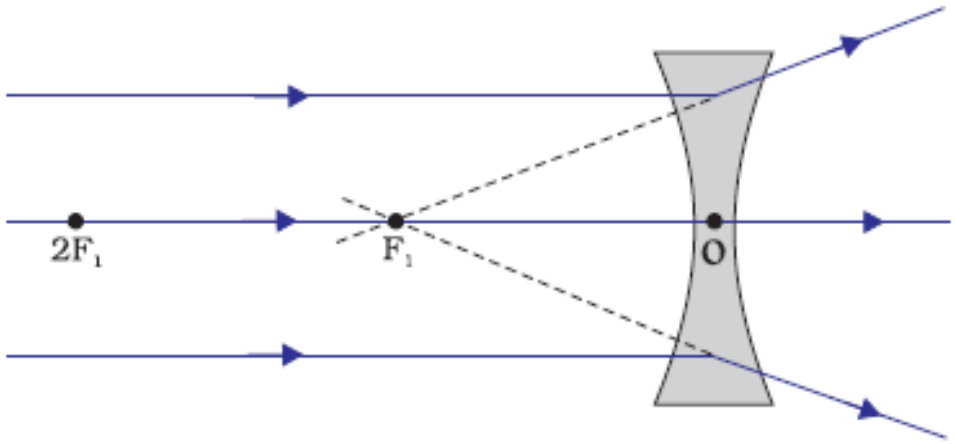
ANAMORPHISMS



$$k = [\!(g)\!] \Leftrightarrow \mathsf{in} \cdot \mathbf{F}\ k \cdot g$$

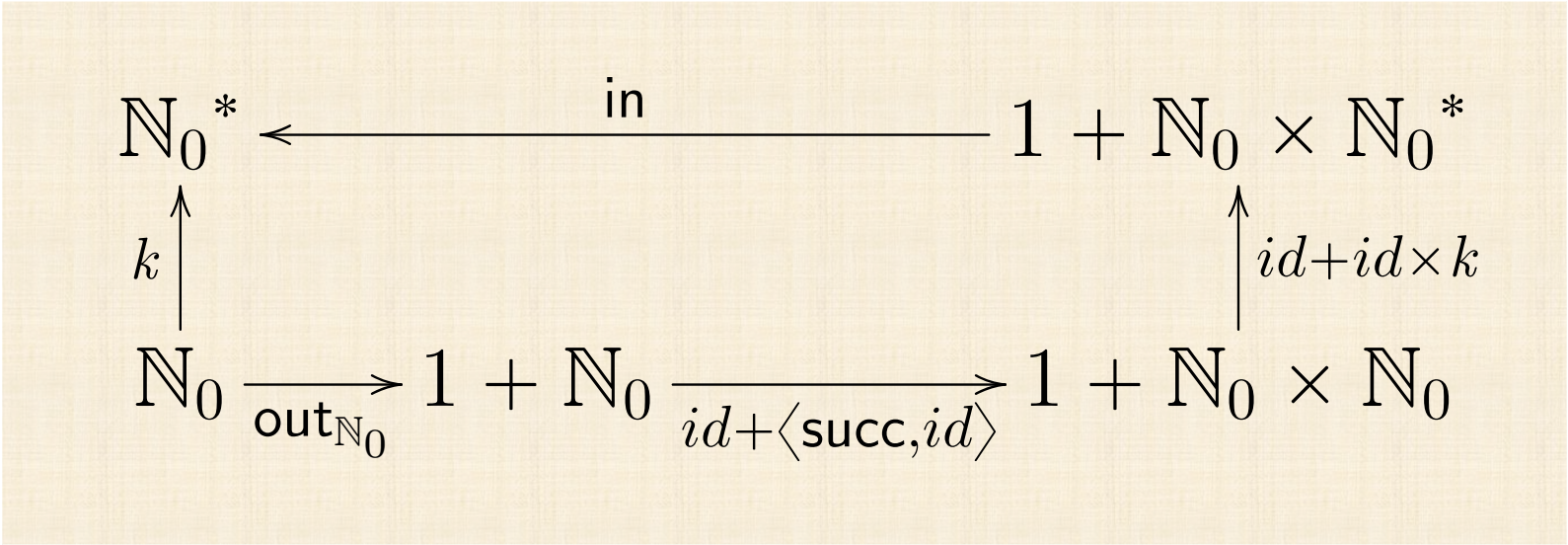
ANAMORPHISM LAWS

Universal-ana	$k = [\!(g)\!] \quad \Leftrightarrow \quad \mathsf{out} \cdot k = (\mathsf{F}\, k) \cdot g$
Cancelamento-ana	$\mathsf{out} \cdot [\!(g)\!] = \mathsf{F}\, [\!(g)\!] \cdot g$
Reflexão-ana	$[\!(\mathsf{out})\!] = id_{\mathsf{T}}$
Fusão-ana	$[\!(g)\!] \cdot f = [\!(h)\!] \quad \Leftarrow \quad g \cdot f = (\mathsf{F}\, f) \cdot h$
Base-ana	$\mathsf{F}\, f \quad = \quad \mathsf{B}\ (id, f)$
Def-map-ana	$\mathsf{T}\, f \quad = \quad [\!(\mathsf{B}(f, id) \cdot \mathsf{out})\!]$
Absorção-ana	$\mathsf{T}\, f \cdot [\!(g)\!] \quad = \quad [\!(\mathsf{B}(f, id) \cdot g)\!]$


2F1
F1
O

[(–)]

Example


$\mathbb{N}_0{}^*$
in
$1 + \mathbb{N}_0 \times \mathbb{N}_0{}^*$
k
$id+id\times k$
$\mathbb{N}_0$
$\mathsf{out}_{\mathbb{N}_0}$
$1 + \mathbb{N}_0$
$id+\langle \mathsf{succ}, id \rangle$
$1 + \mathbb{N}_0 \times \mathbb{N}_0$

$$
\begin{aligned}
& k = [\!(\,(id + \langle \mathsf{succ}, id \rangle) \cdot \mathsf{out}_{\mathbb{N}_0}\,)\!] \\
\equiv & \quad \{ \text{ ana-universal } \} \\
& k = \mathsf{in} \cdot (id + id \times k) \cdot (id + \langle \mathsf{succ}, id \rangle) \cdot \mathsf{out}_{\mathbb{N}_0} \\
\equiv & \quad \{ \text{ isomorphism } \mathsf{in}_{\mathbb{N}_0} \,/\, \mathsf{out}_{\mathbb{N}_0} \} \\
& k \cdot \mathsf{in}_{\mathbb{N}_0} = \mathsf{in} \cdot (id + id \times k) \cdot (id + \langle \mathsf{succ}, id \rangle) \\
\equiv & \quad \{ \text{ functor-}+\text{; absorption-}\times \} \\
& k \cdot \mathsf{in}_{\mathbb{N}_0} = \mathsf{in} \cdot (id + \langle \mathsf{succ}, k \rangle)
\end{aligned}
$$

$$\equiv \quad \{ \text{ definitions of } \mathsf{in} \text{ and } \mathsf{in}_{\mathbb{N}_0}\text{; fusion and absorption-}+ \}$$

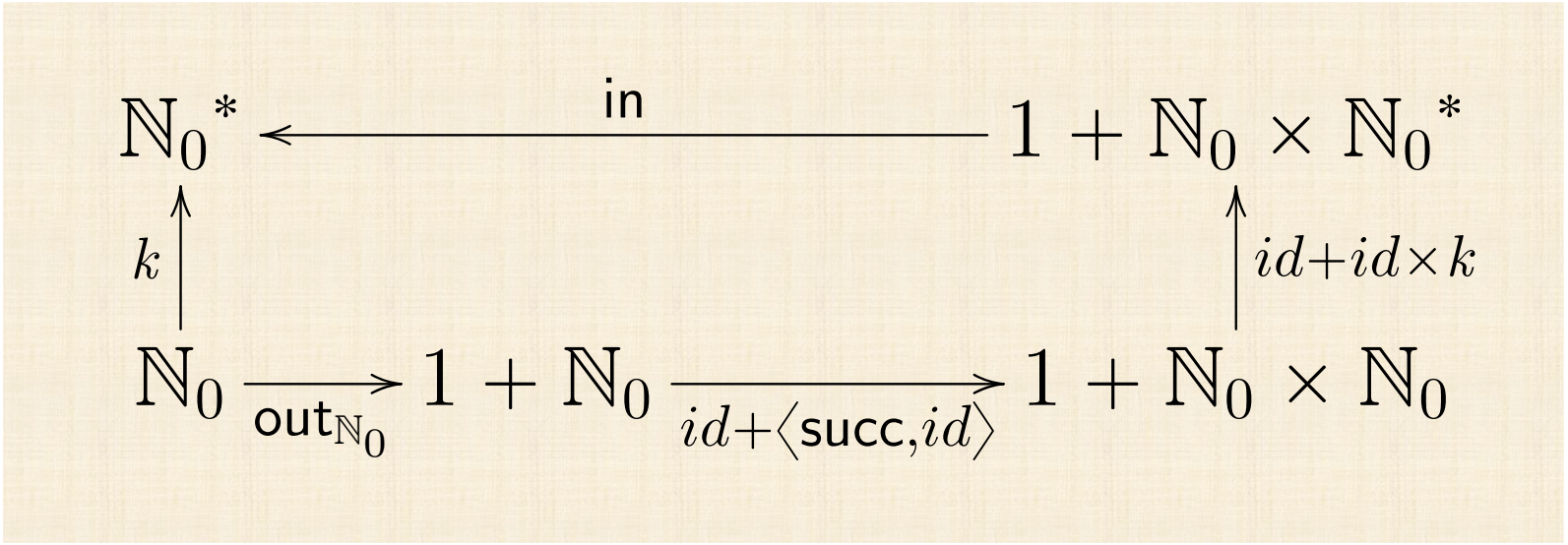
$$[k \cdot \underline{0}, k \cdot \mathsf{succ}] = [\mathsf{nil}, \mathsf{cons} \cdot \langle \mathsf{succ}, k \rangle]$$

$$\equiv \quad \{ \text{ eq-}+ \}$$

$$\begin{cases} k \cdot \underline{0} = \mathsf{nil} \\ k \cdot \mathsf{succ} = \mathsf{cons} \cdot \langle \mathsf{succ}, k \rangle \end{cases}$$

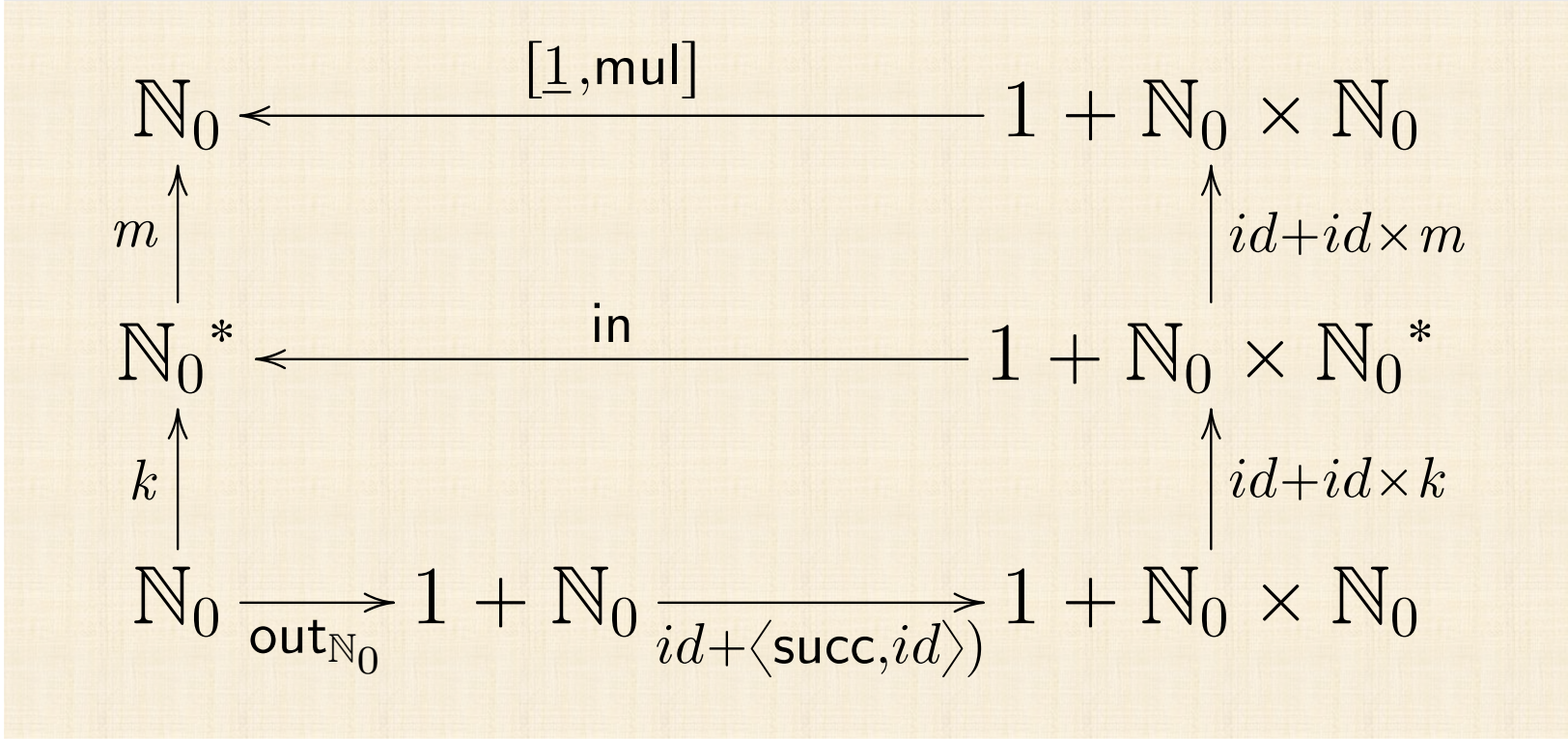
$$\equiv \quad \{ \text{ going pointwise } \}$$

$$\begin{cases} k\ 0 = [\,] \\ k\ (n+1) = (n+1) : k\ n \end{cases}$$



$$\begin{cases} k\ 0 = [\,] \\ k\ (n+1) = (n+1) : k\ n \end{cases}$$

Hylomorphisms


$\mathbb{N}_0$
$[\underline{1},\mathsf{mul}]$
$1 + \mathbb{N}_0 \times \mathbb{N}_0$
m
$id+id\times m$
$\mathbb{N}_0{}^*$
in
$1 + \mathbb{N}_0 \times \mathbb{N}_0{}^*$
k
$id+id\times k$
$\mathbb{N}_0$
$\mathsf{out}_{\mathbb{N}_0}$
$1 + \mathbb{N}_0$
$id+\langle \mathsf{succ},id\rangle)$
$1 + \mathbb{N}_0 \times \mathbb{N}_0$

$$
\begin{aligned}
& f = m \cdot k \\
\equiv & \quad \{\ m = (\!|\, [\underline{1}, \mathsf{mul}]\, |\!) \text{ and } k = [\!(\, (id + \langle \mathsf{succ}, id \rangle) \cdot \mathsf{out}_{\mathbb{N}_0}\,)\!]\ \} \\
& f = (\!|\, [\underline{1}, \mathsf{mul}]\, |\!) \cdot [\!(\, (id + \langle \mathsf{succ}, id \rangle) \cdot \mathsf{out}_{\mathbb{N}_0}\,)\!] \\
\equiv & \quad \{\ \text{cancellations (ana and cata)}\ \} \\
& f = [\underline{1}, \mathsf{mul}] \cdot \mathbf{F}\ m \cdot \mathsf{out} \cdot \mathsf{in} \cdot \mathbf{F}\ k \cdot (id + \langle \mathsf{succ}, id \rangle) \cdot \mathsf{out}_{\mathbb{N}_0} \\
\equiv & \quad \{\ \mathsf{in} \cdot \mathsf{out} = id \text{ ; functor } \mathbf{F}\text{: } (\mathbf{F}\ m) \cdot (\mathbf{F}\ k) = \mathbf{F}\ (m \cdot k)\ \} \\
& f = [\underline{1}, \mathsf{mul}] \cdot \mathbf{F}\ (m \cdot k) \cdot (id + \langle \mathsf{succ}, id \rangle) \cdot \mathsf{out}_{\mathbb{N}_0} \\
\equiv & \quad \{\ \text{isomorphism } \mathsf{in}_{\mathbb{N}_0} \,/\, \mathsf{out}_{\mathbb{N}_0} ;\ m \cdot k = f \text{ ; } \mathbf{F}\ f = id + id \times f\ \} \\
& f \cdot \mathsf{in}_{\mathbb{N}_0} = [\underline{1}, \mathsf{mul}] \cdot (id + id \times f) \cdot (id + \langle \mathsf{succ}, id \rangle)
\end{aligned}
$$

$$f \cdot \mathsf{in}_{\mathbb{N}_0} = [\underline{1}, \mathsf{mul}] \cdot (id + id \times f) \cdot (id + \langle \mathsf{succ}, id \rangle)$$

$\equiv$ { $+$-absorption ; $\times$-absorption ; etc }

$$f \cdot \mathsf{in}_{\mathbb{N}_0} = [\underline{1}, \mathsf{mul} \cdot \langle \mathsf{succ}, f \rangle]$$

$\equiv$ { Eq-$+$; $\mathsf{in}_{\mathbb{N}_0} = [\underline{0}, \mathsf{succ}]$ }

$$\begin{cases} f \cdot \underline{0} = \underline{1} \\ f \cdot \mathsf{succ} = \mathsf{mul} \cdot \langle \mathsf{succ}, f \rangle \end{cases}$$

$\equiv$ { go pointwise }

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

Factorial (“rediscovered”)

$$fac = (\!|\, [\underline{1}, \mathsf{mul}] \,|\!) \cdot [\!(\, (id + \langle \mathsf{succ}, id \rangle) \cdot \mathsf{out}_{\mathbb{N}_0} \,)\!]$$

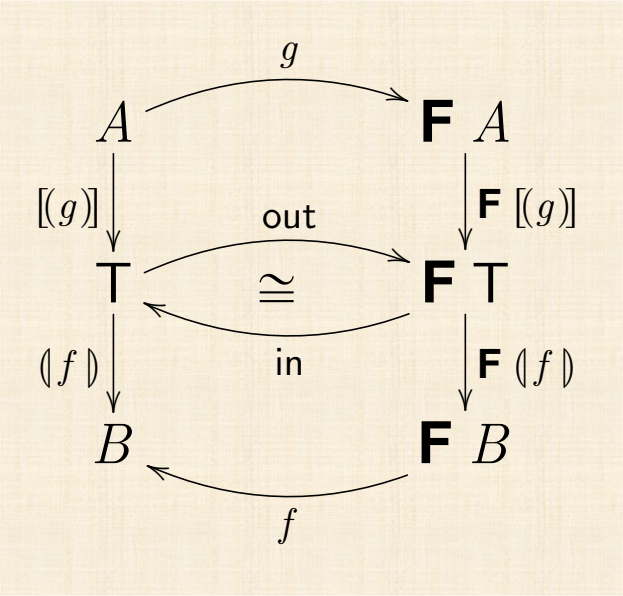
HILOMORPHISM

"Hylo + morphism"

ξύλο = matter, thing

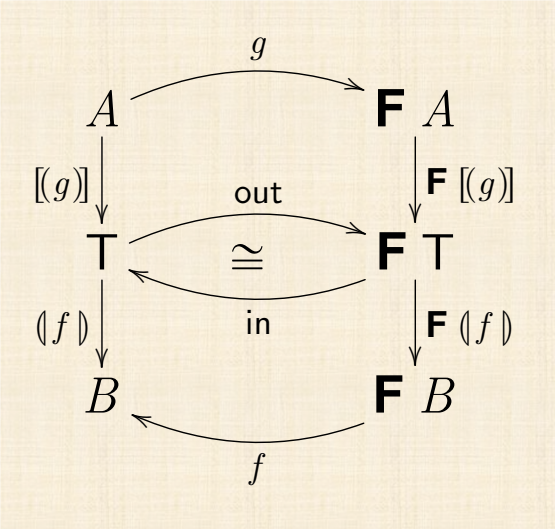
$$[\![f, g]\!] = (\!|\, f \,|\!) \cdot [\!(g)\!]$$

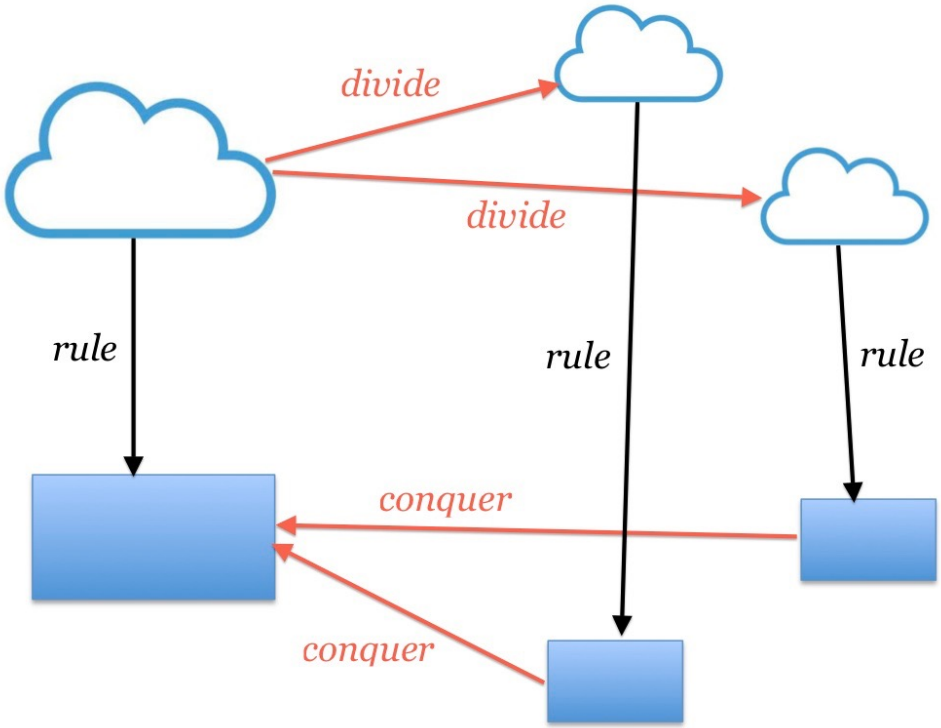
HILOMORPHISM



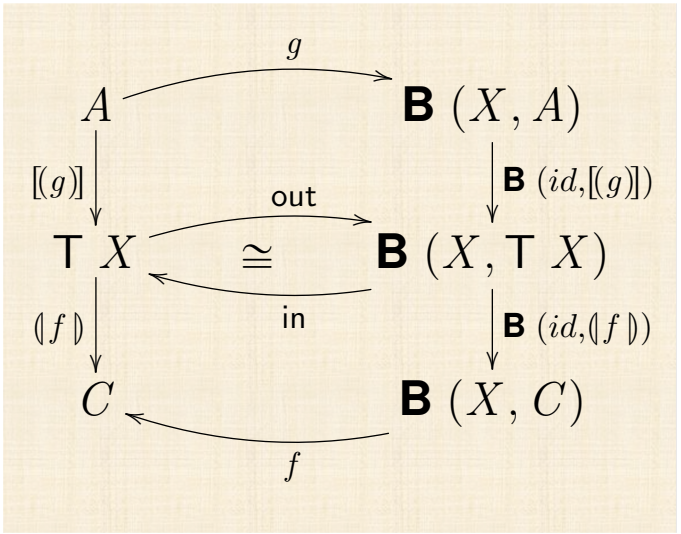
$$[\![f, g]\!] = (\!|\, f \,|\!) \cdot [\!(g)\!]$$

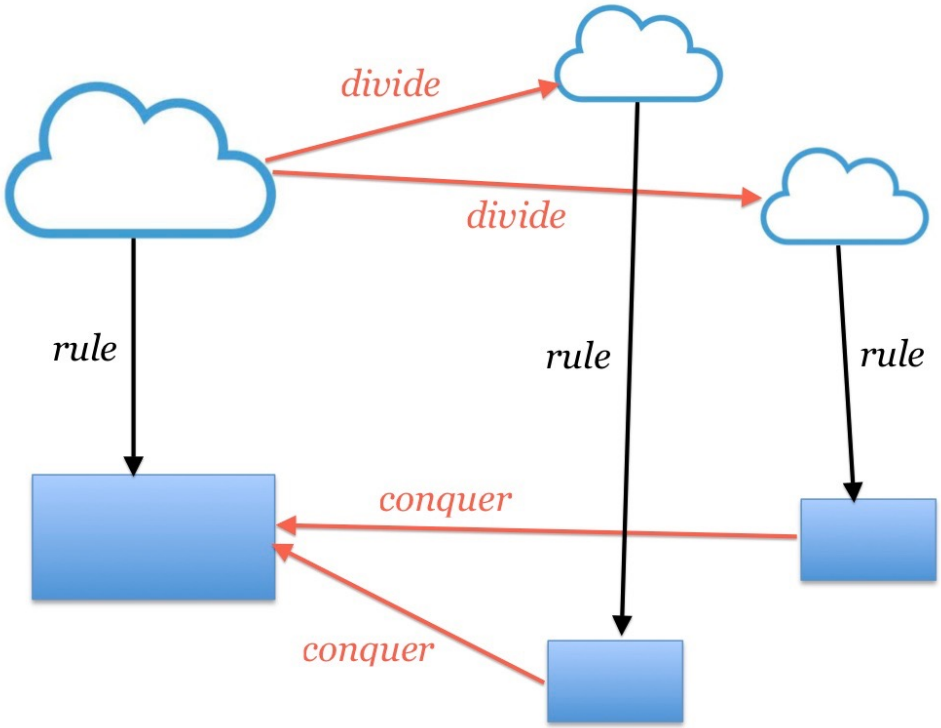
‘DIVIDE & CONQUER’





‘DIVIDE & CONQUER’





`DIVIDE & CONQUER’

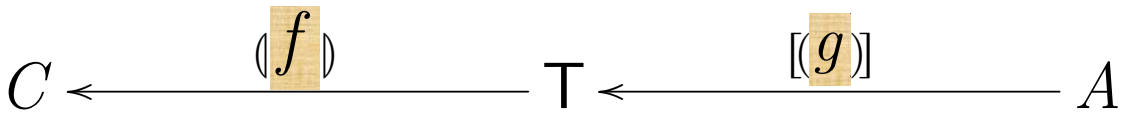
$$C \xleftarrow{(\!|-|\!)} \mathsf{T} \xleftarrow{[\![-]\!]} A$$

ANA, CATA & HYLO

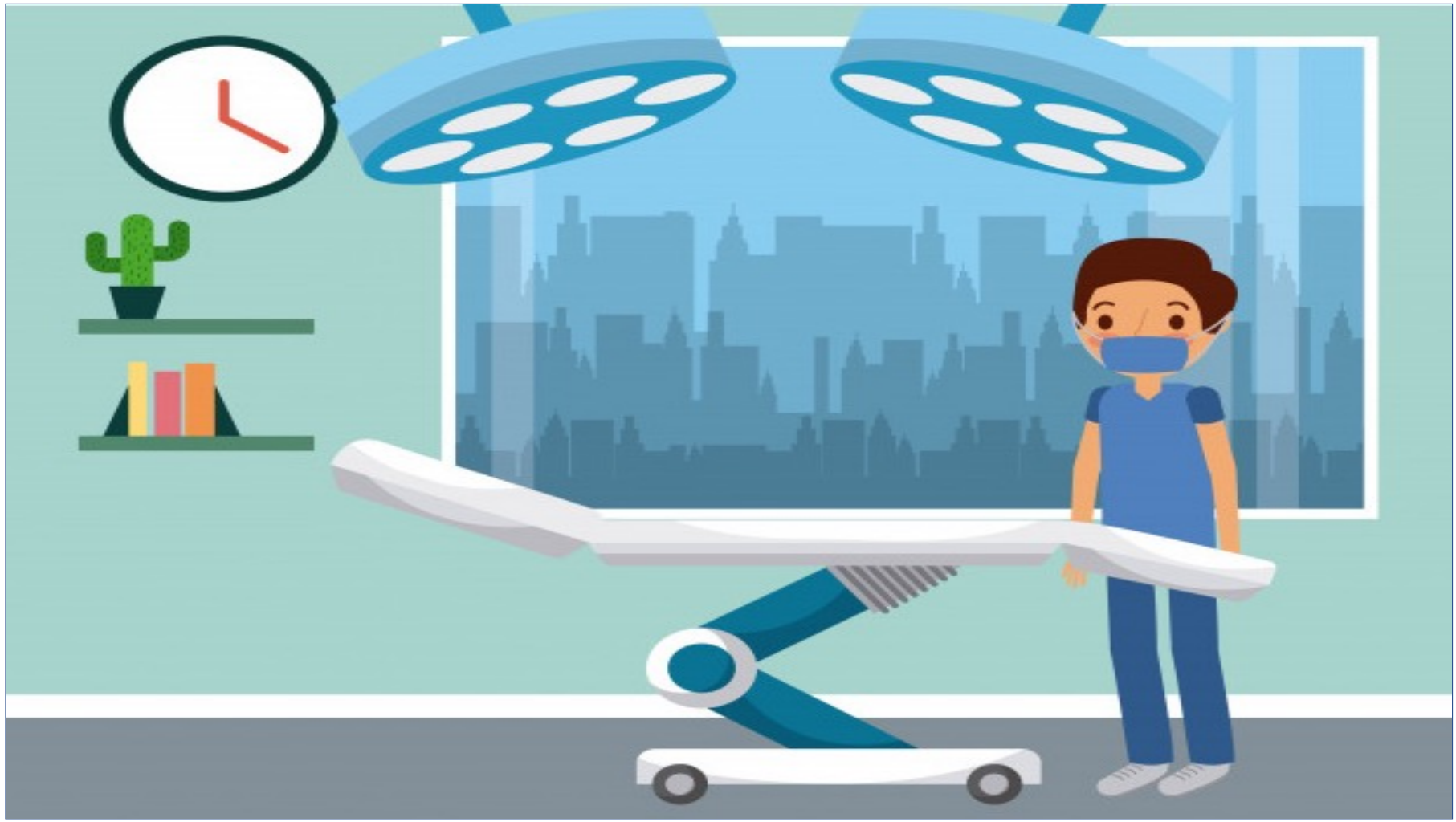
$$[\![\mathsf{in}, g]\!] = [\!(g)\!]$$
$$[\![f, \mathsf{out}]\!] = (\!|\, f \,|\!)$$

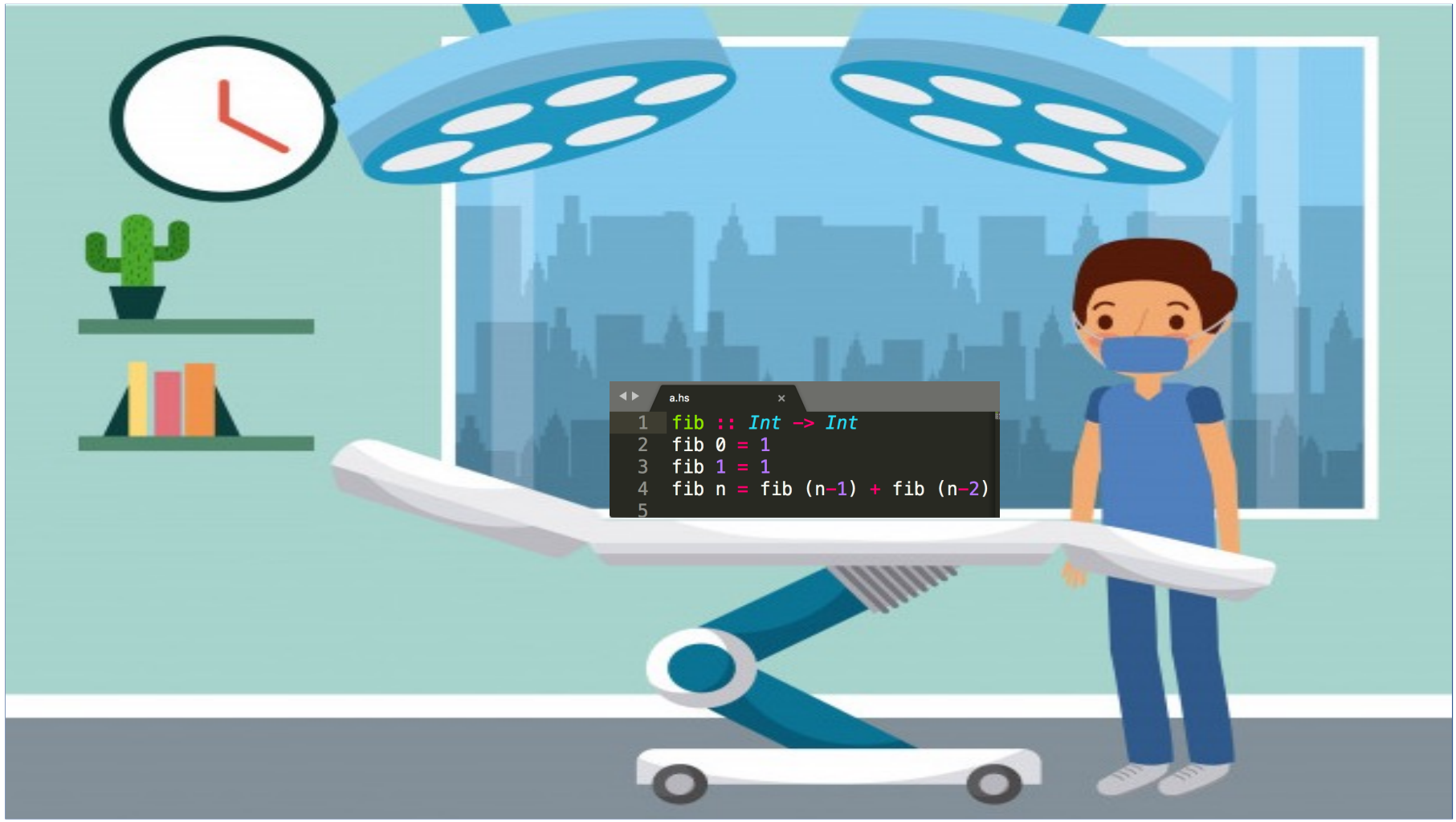
Reflexion laws:

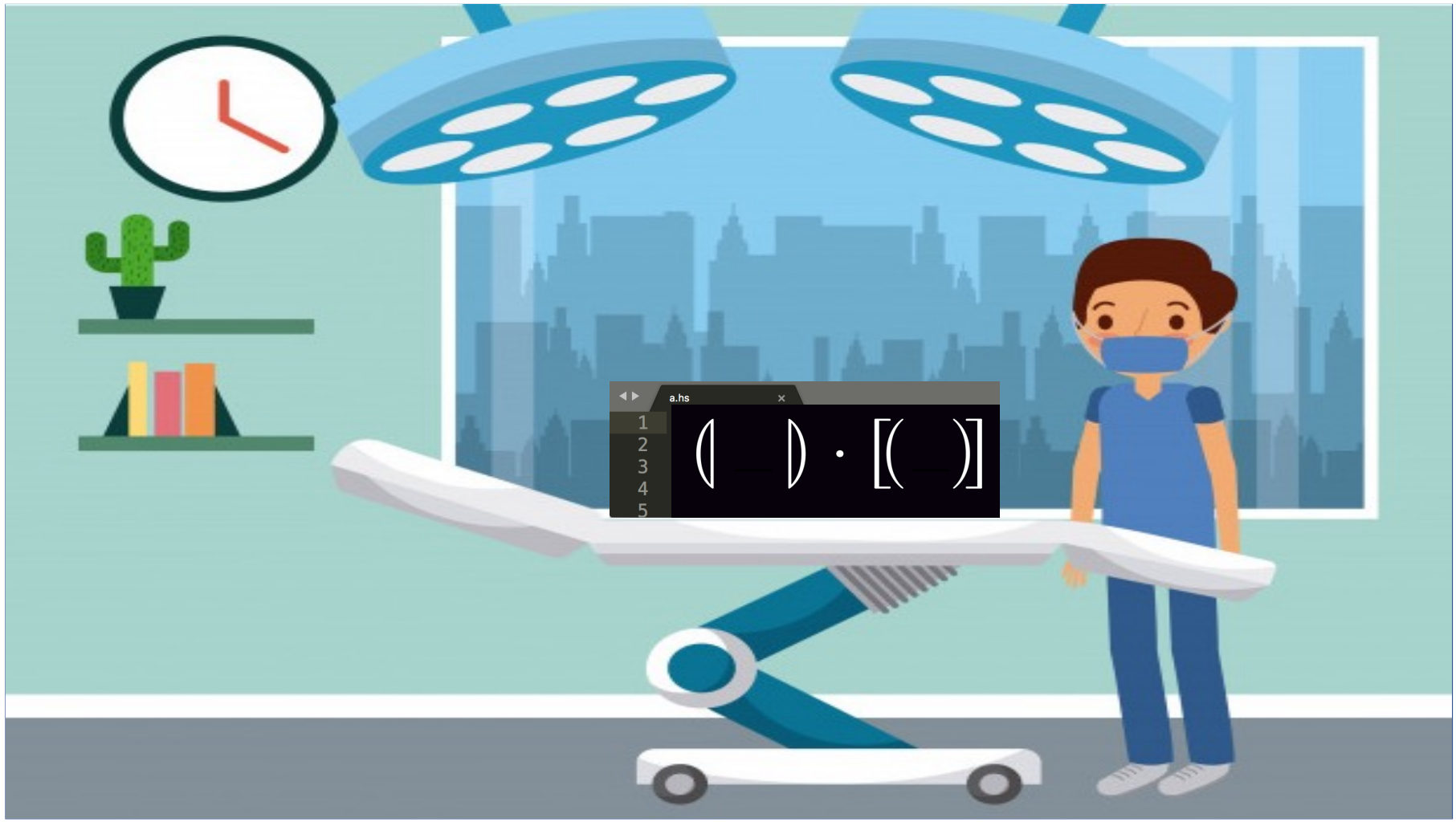
$$(\!|\,\mathsf{in}\,|\!) = id$$
$$[\!(\mathsf{out})\!] = id$$

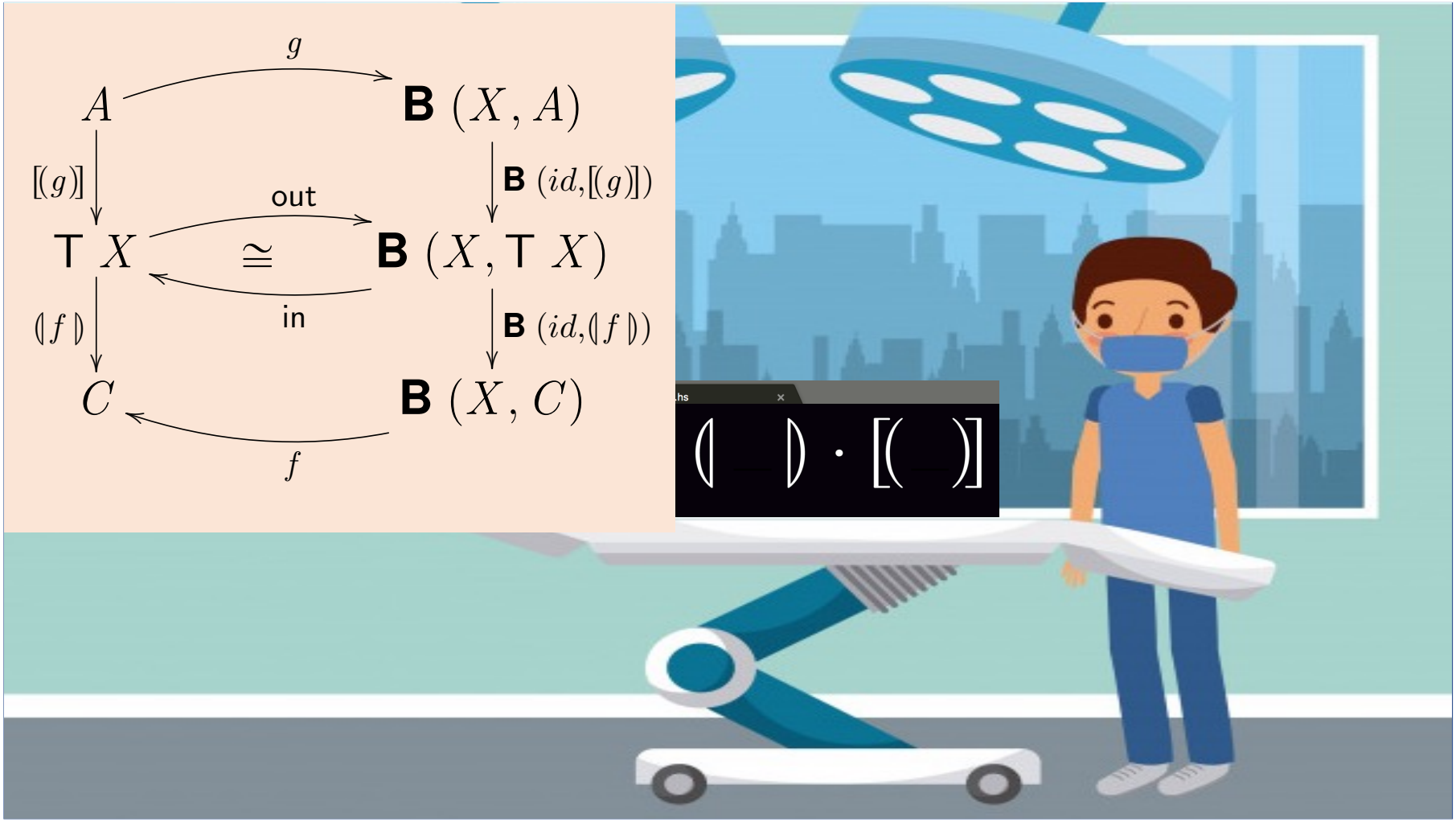


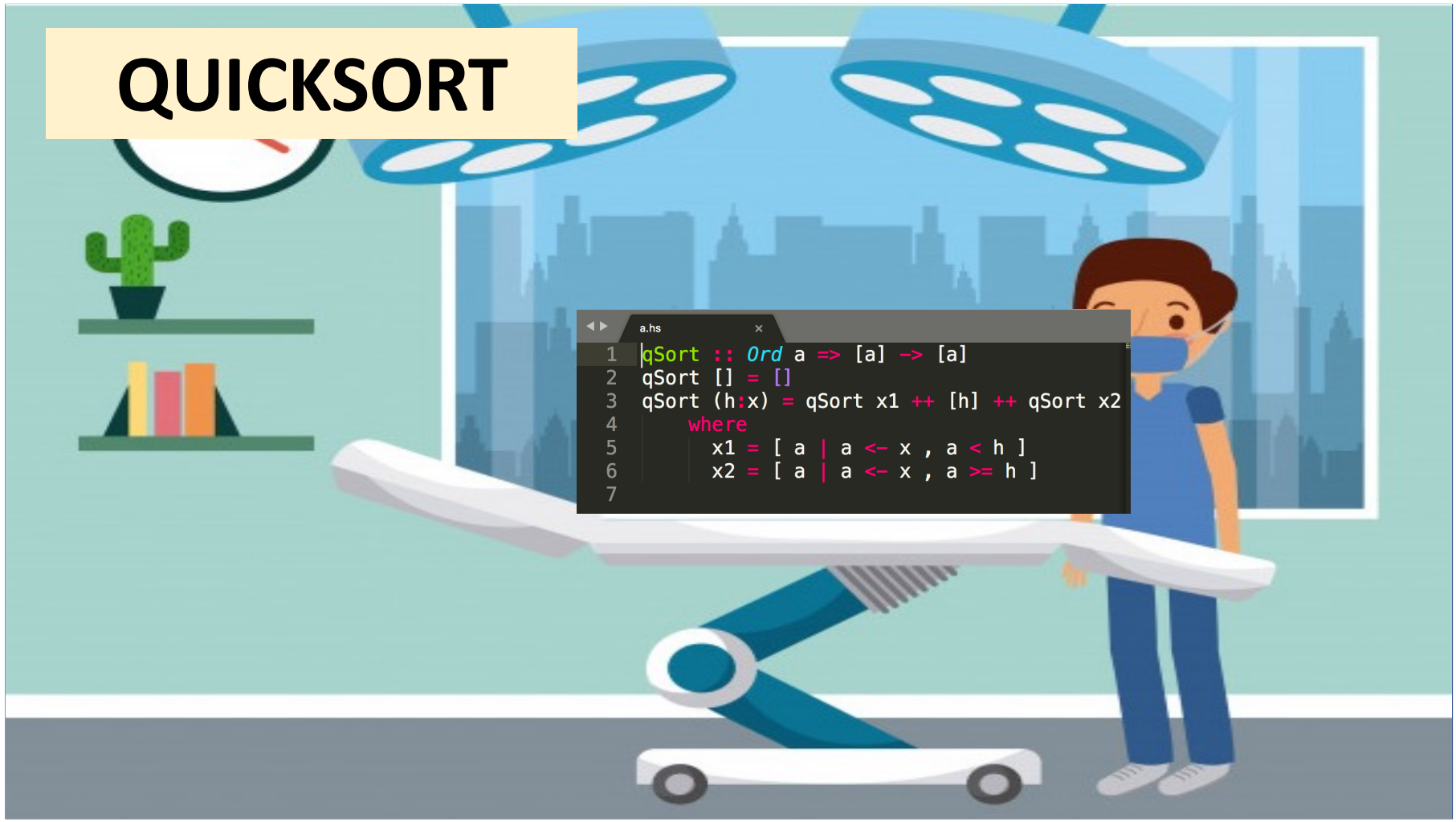
$$[\![f, g]\!] = (\!|\, f \,|\!) \cdot [\!(g)\!]$$




a.hs
1 fib :: Int -> Int
2 fib 0 = 1
3 fib 1 = 1
4 fib n = fib (n-1) + fib (n-2)
5


a.hs
1
2
3
4
5
( ) · [( )]

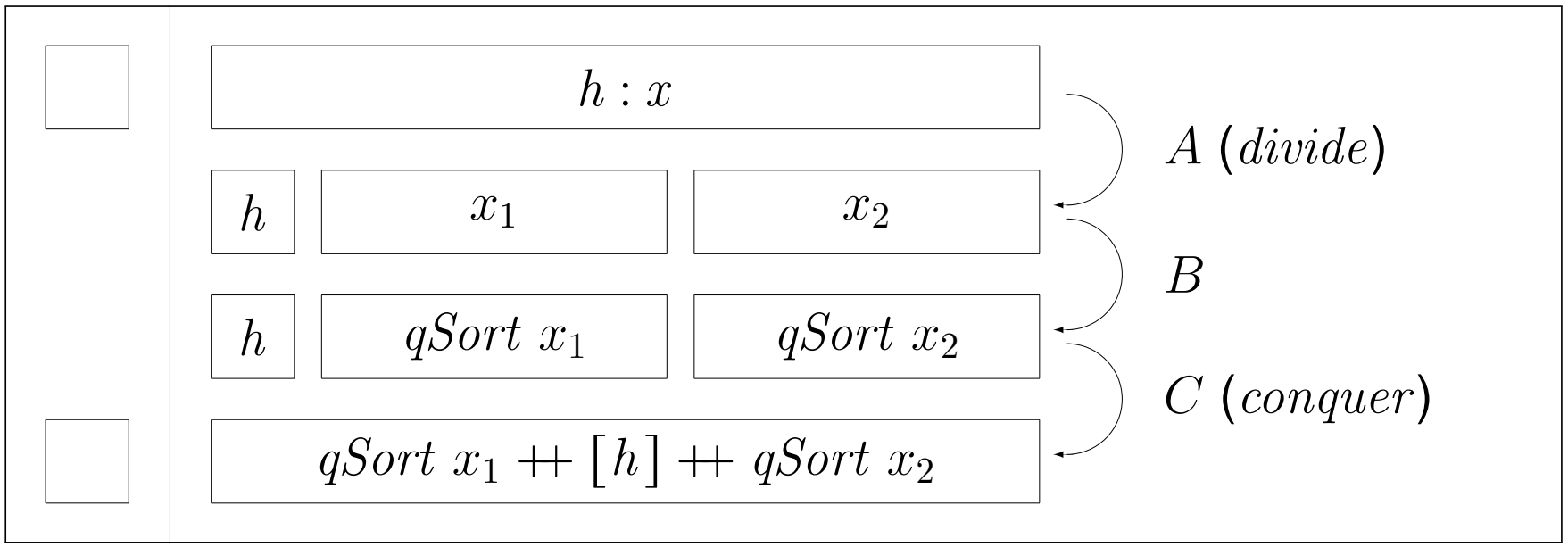

g
A
B (X, A)
[(g)]
B (id,[(g)])
out
T X
≅
B (X, T X)
in
(| f |)
B (id,(| f |))
C
B (X, C)
f
(| _ |) · [(_)]


QUICKSORT
a.hs
1 qSort :: Ord a => [a] -> [a]
2 qSort [] = []
3 qSort (h:x) = qSort x1 ++ [h] ++ qSort x2
4 where
5 x1 = [a | a <- x , a < h]
6 x2 = [a | a <- x , a >= h]
7

QUICKSORT

a.hs

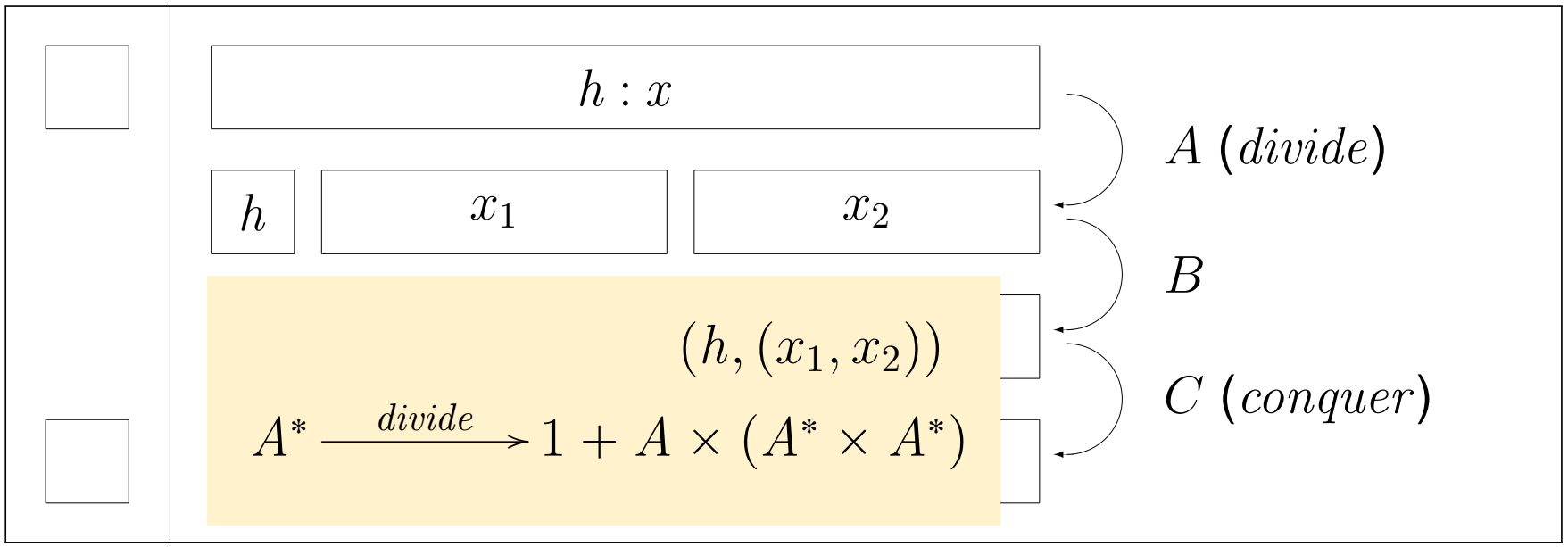
```haskell
qSort :: Ord a => [a] -> [a]
qSort [] = []
qSort (h:x) = qSort x1 ++ [h] ++ qSort x2
    where
      x1 = [ a | a <- x , a < h ]
      x2 = [ a | a <- x , a >= h ]

```



QUICKSORT

a.hs

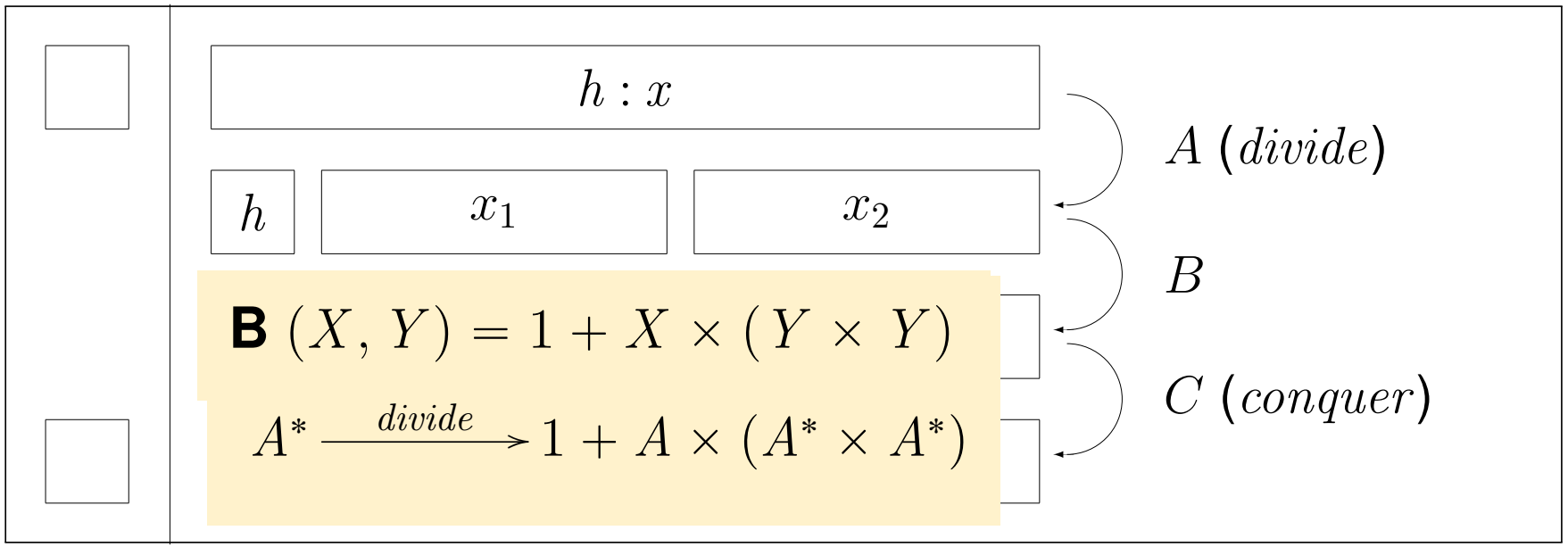
```haskell
qSort :: Ord a => [a] -> [a]
qSort [] = []
qSort (h:x) = qSort x1 ++ [h] ++ qSort x2
    where
      x1 = [ a | a <- x , a < h ]
      x2 = [ a | a <- x , a >= h ]

```



QUICKSORT

a.hs

```haskell
qSort :: Ord a => [a] -> [a]
qSort [] = []
qSort (h:x) = qSort x1 ++ [h] ++ qSort x2
    where
      x1 = [ a | a <- x , a < h ]
      x2 = [ a | a <- x , a >= h ]

```



QUICKSORT

a.hs

```haskell
qSort :: Ord a => [a] -> [a]
qSort [] = []
qSort (h:x) = qSort x1 ++ [h] ++ qSort x2
    where
      x1 = [ a | a <- x , a < h ]
      x2 = [ a | a <- x , a >= h ]

```

$$\begin{cases} qSort \cdot \mathsf{nil} = \mathsf{nil} \\ qSort \cdot \mathsf{cons} = f_2 \cdot (id \times (qSort \times qSort)) \cdot g_2 \end{cases}$$

$$f_2\ (h, (y_1, y_2)) = y_1 +\!\!+ [h] +\!\!+ y_2$$

$$g_2\ (h, x) = (h, (x_1, x_2))$$
$$\textbf{where}$$
$$x_1 = [a \mid a \leftarrow x, a < h]$$
$$x_2 = [a \mid a \leftarrow x, a \geqslant h]$$

QUICKSORT

$$\mathbf{B}\ (X, Y) = 1 + X \times (Y \times Y)$$

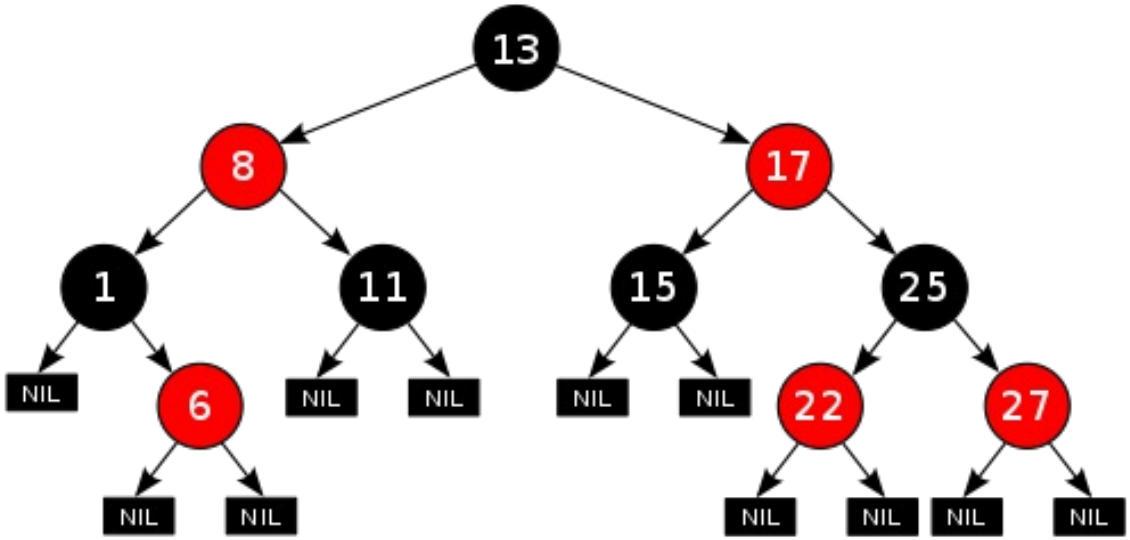
$$\begin{cases} qSort \cdot \mathsf{nil} = \mathsf{nil} \\ qSort \cdot \mathsf{cons} = f_2 \cdot (id \times (qSort \times qSort)) \cdot g_2 \end{cases}$$

$$\equiv \quad \{ \text{ fusão-+, absorção-+, eq-+ etc } \}$$

$$qSort \cdot \mathsf{in} = [\mathsf{nil}, f_2] \cdot (id + id \times qSort^2) \cdot (id + g_2)$$

$$\equiv \quad \{ \text{ isomorfismo in / out } \}$$

$$qSort = \underbrace{[\mathsf{nil}, f_2]}_{conquer} \cdot (\underbrace{id + id \times qSort^2}_{\mathbf{B}\ (id, qSort)} \cdot \underbrace{(id + g_2) \cdot \mathsf{out}}_{divide}$$




```
data BTree a = Empty | Node (a, (BTree a, BTree a))
```

QUICKSORT

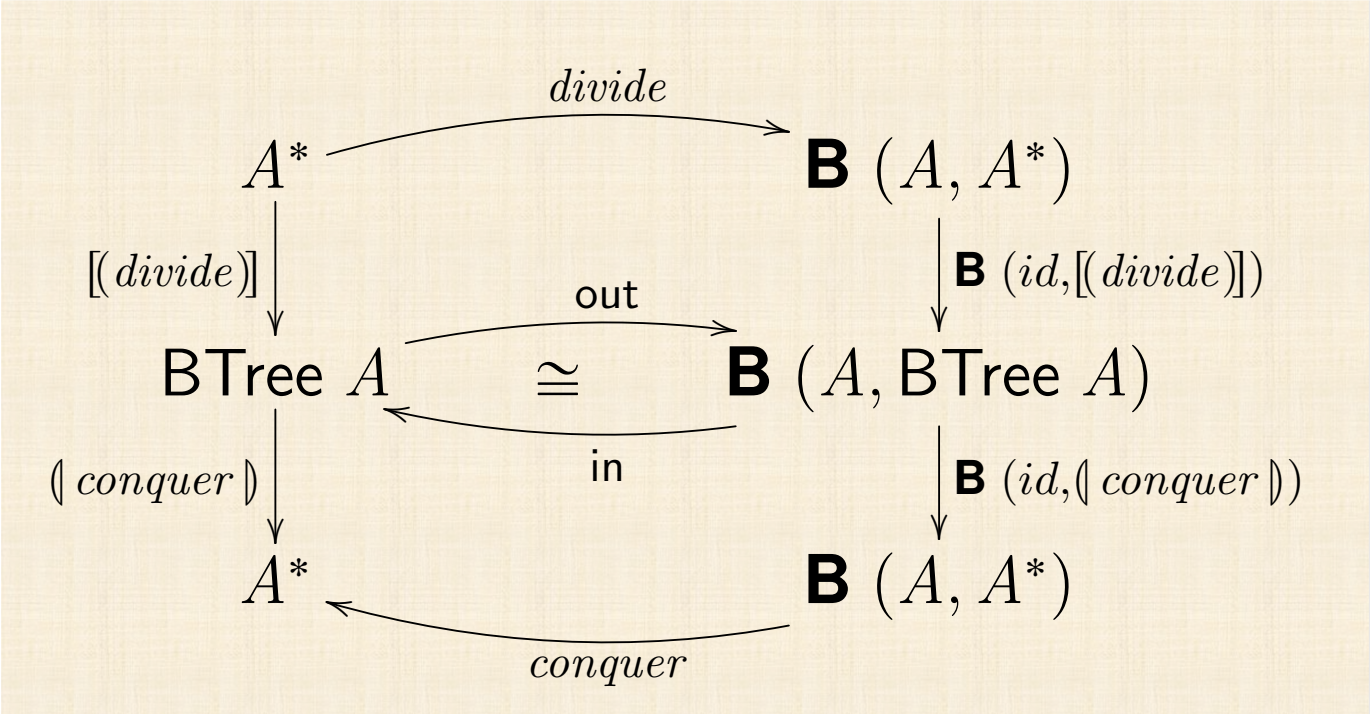
$$\mathbf{B}\,(X, Y) = 1 + X \times (Y \times Y)$$

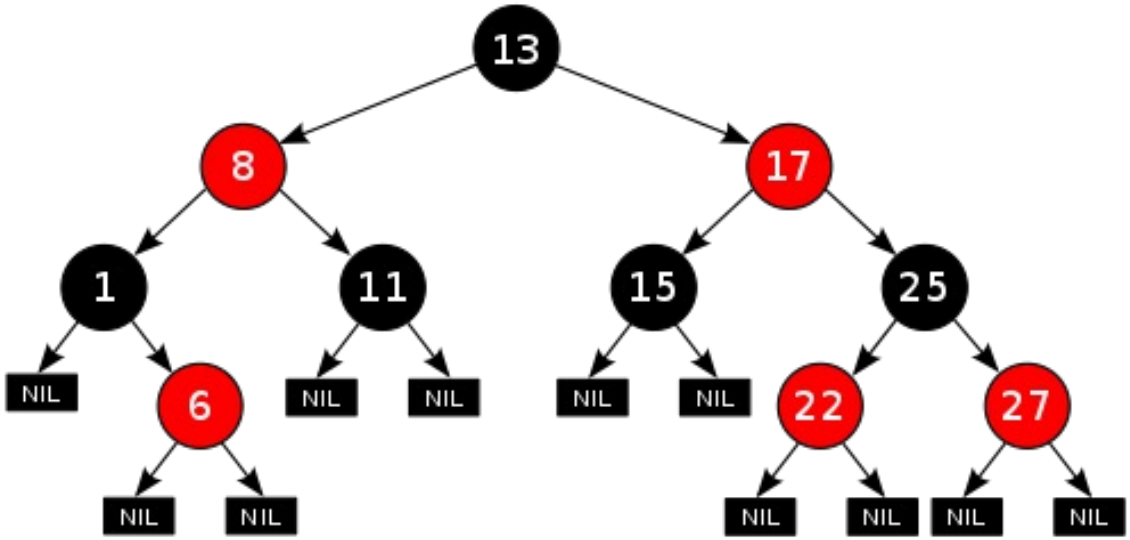
Description	T X	B (X, Y)	B (id, f)	B (f, id)
"Right" Lists	List X	$1 + X \times Y$	$id + id \times f$	$id + f \times id$
"Left" Lists	LList X	$1 + Y \times X$	$id + f \times id$	$id + id \times f$
Non-empty Lists	NList X	$X + X \times Y$	$id + id \times f$	$f + f \times id$
Binary Trees	BTree X	$1 + X \times Y^2$	$id + id \times f^2$	$id + f \times id$
"Leaf" Trees	LTree X	$X + Y^2$	$id + f^2$	$f + id$

QUICKSORT

a.hs

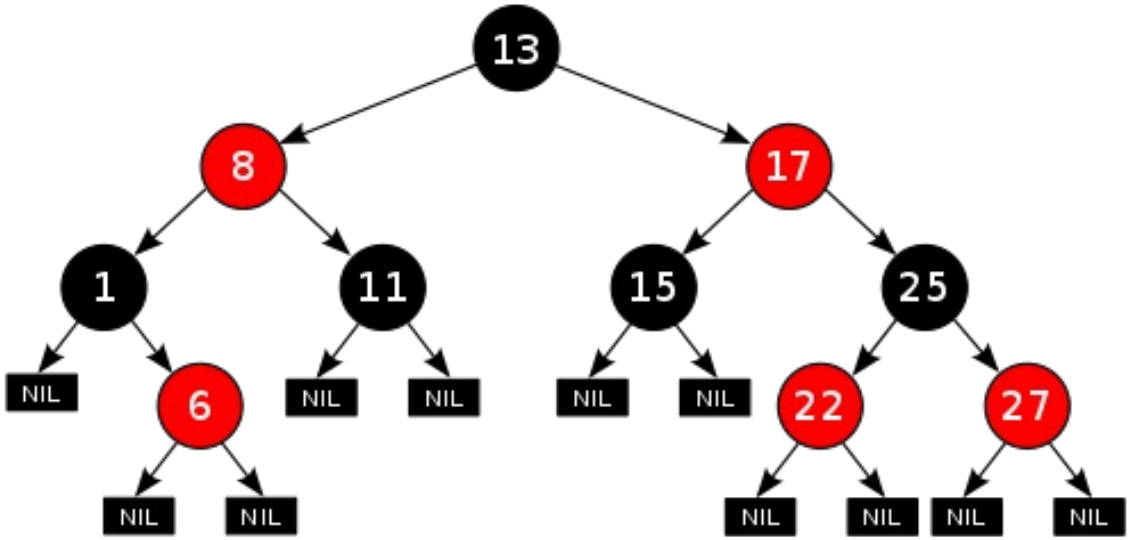
```haskell
qSort :: Ord a => [a] -> [a]
qSort [] = []
qSort (h:x) = qSort x1 ++ [h] ++ qSort x2
    where
      x1 = [ a | a <- x , a < h ]
      x2 = [ a | a <- x , a >= h ]

```



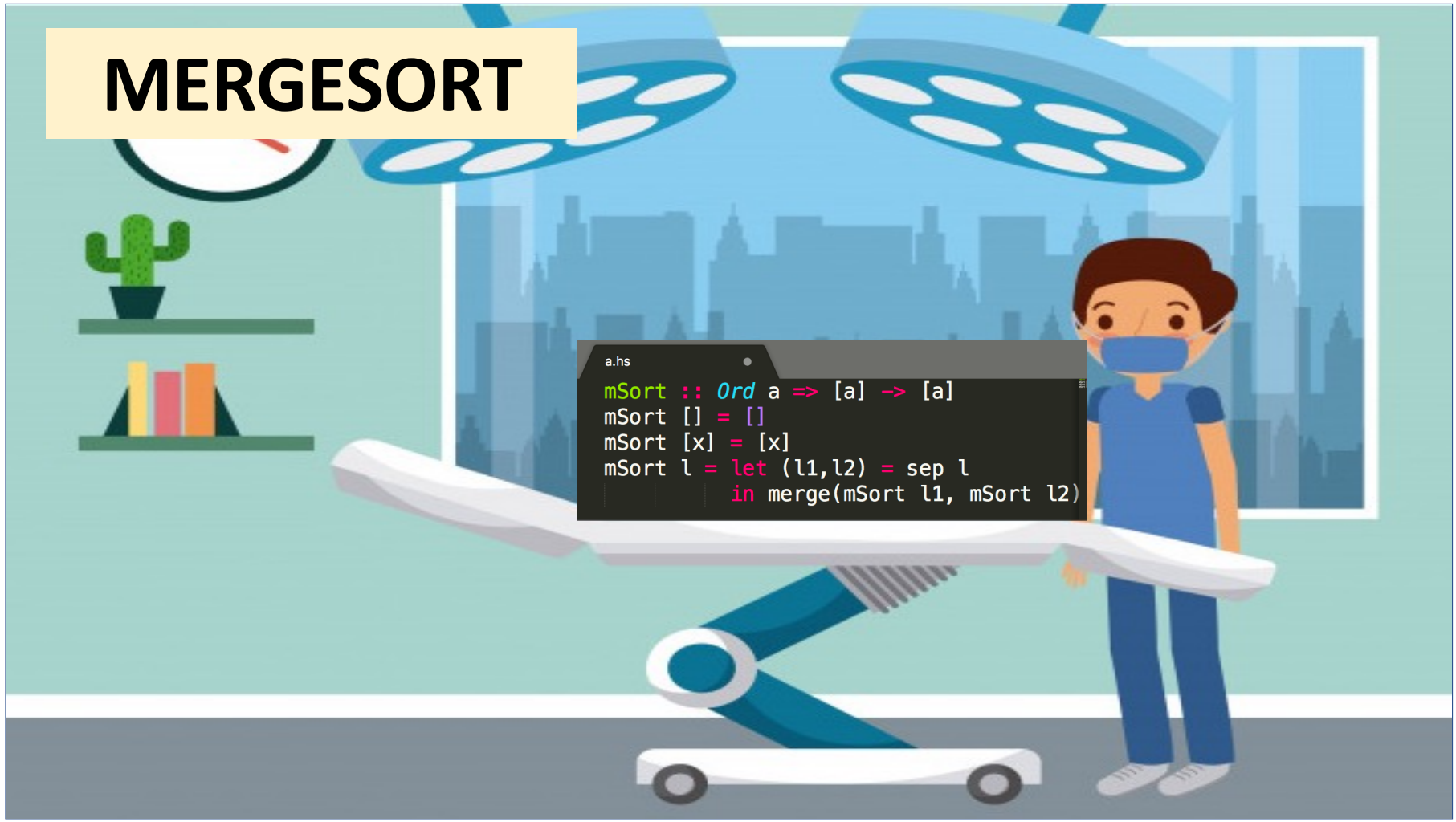



```
t = anaB divide [13,8,17,1,6,11,25,15,27,22]
```

```
data BTree a = Empty | Node (a, (BTree a, BTree a))
```



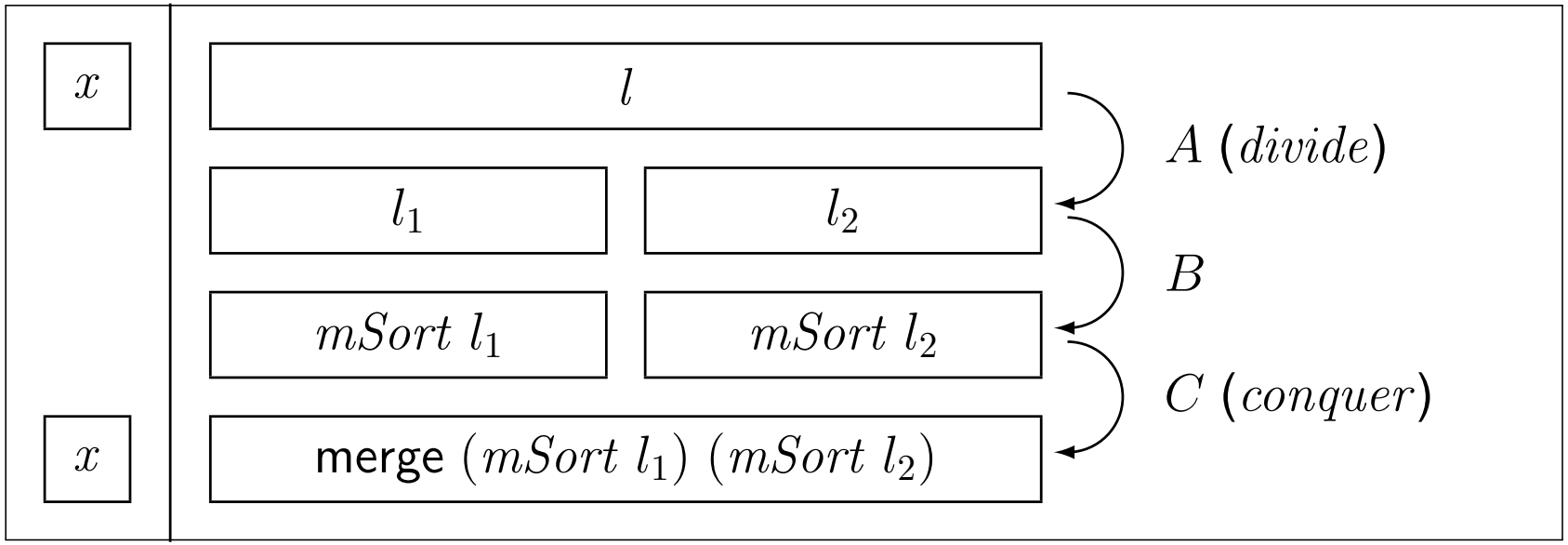

```
*Cp> anaB divide [13,8,17,1,6,11,25,15,27,22]
Node (13,(Node (8,(Node (1,(Empty,Node (6,(Empty,Empty)))),Node
(11,(Empty,Empty)))),Node (17,(Node (15,(Empty,Empty)),Node (25,
(Node (22,(Empty,Empty)),Node (27,(Empty,Empty)))))))
```


MERGESORT
a.hs
mSort :: Ord a => [a] -> [a]
mSort [] = []
mSort [x] = [x]
mSort l = let (l1,l2) = sep l
in merge(mSort l1, mSort l2)

MERGESORT

a.hs

```
mSort :: Ord a => [a] -> [a]
mSort [] = []
mSort [x] = [x]
mSort l = let (l1,l2) = sep l
          in merge(mSort l1, mSort l2)
```

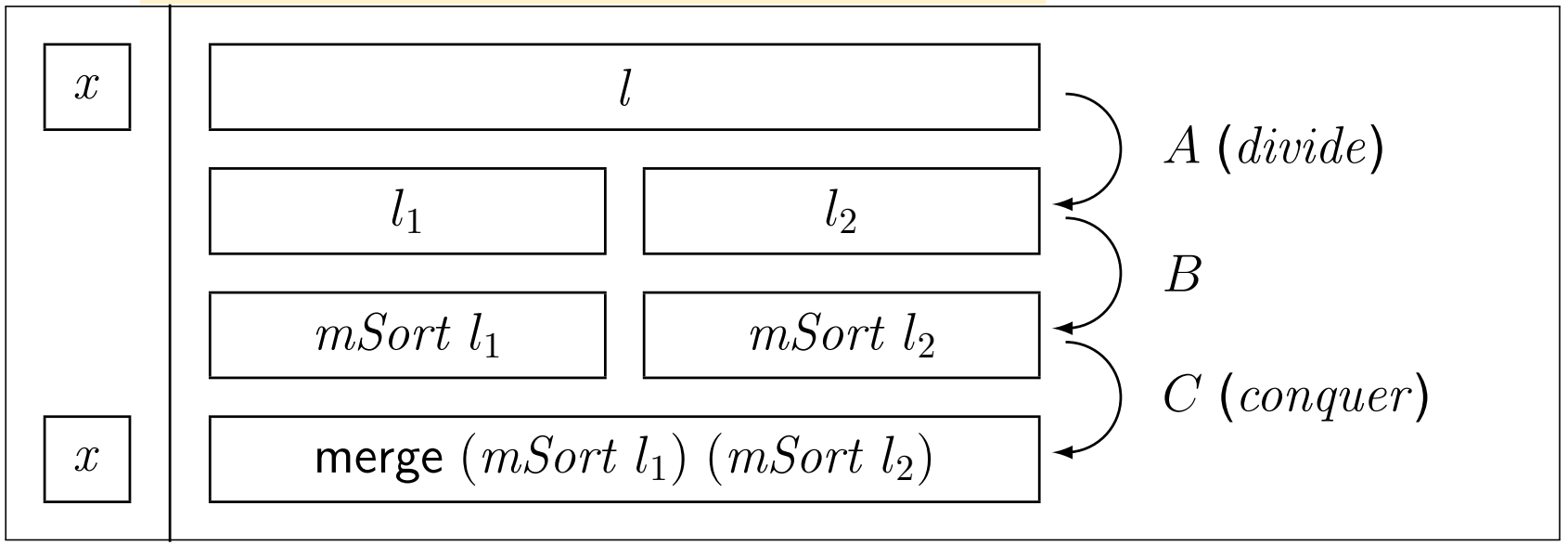


MERGESORT

a.hs

```
mSort :: Ord a => [a] -> [a]


l1,l2) = sep l
rge(mSort l1, mSort l2)
```

$$divide : A^* \rightarrow A + (A^* \times A^*)$$

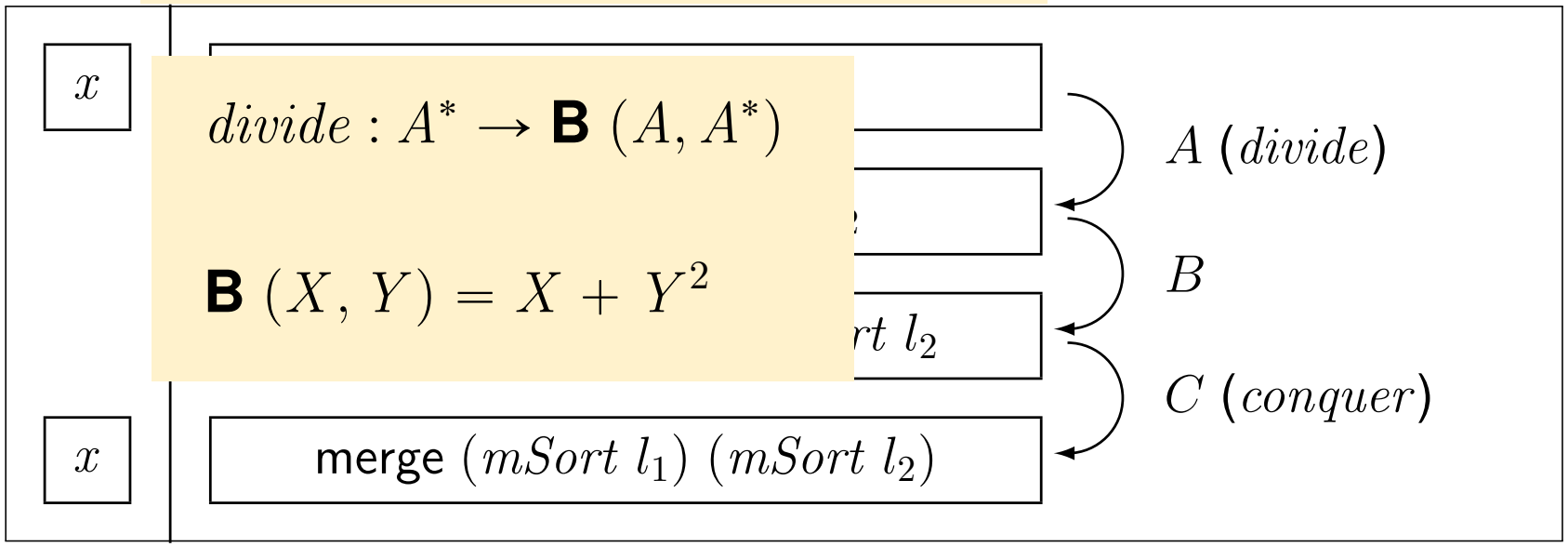


MERGESORT

a.hs

```
mSort :: Ord a => [a] -> [a]


l1,l2) = sep l
rge(mSort l1, mSort l2)
```

$$divide : A^* \rightarrow A + (A^* \times A^*)$$



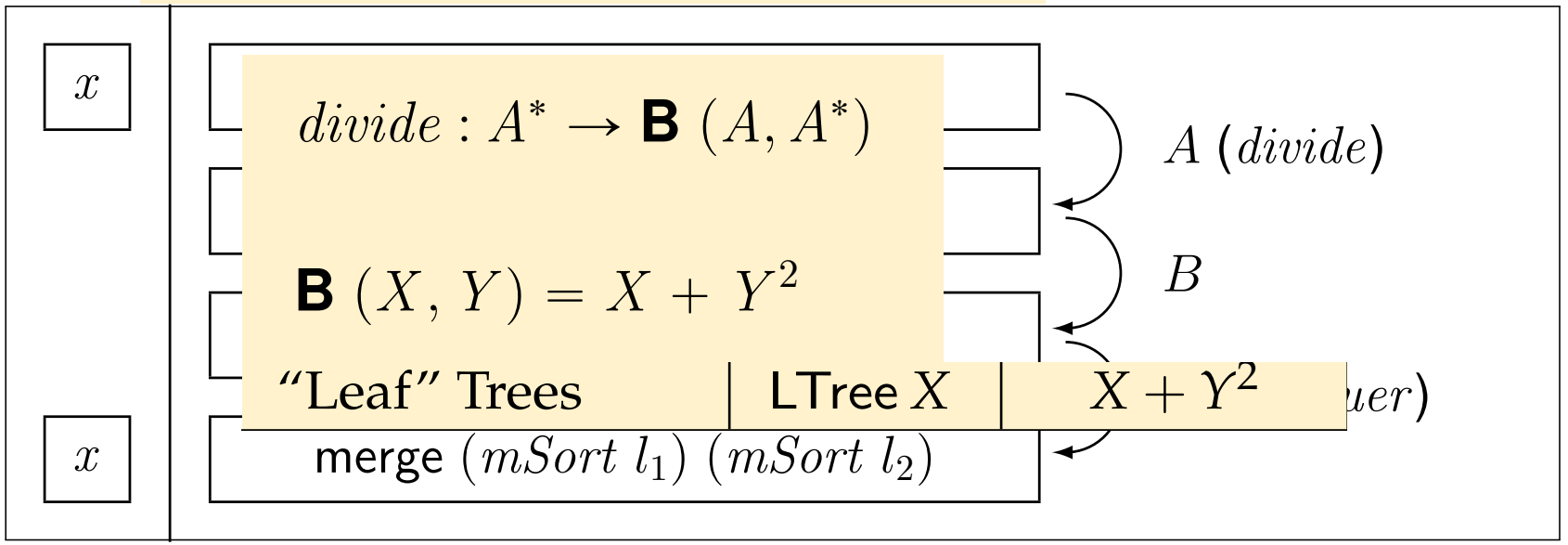
$$divide : A^* \rightarrow \mathbf{B}\ (A, A^*)$$

$$\mathbf{B}\ (X, Y) = X + Y^2$$

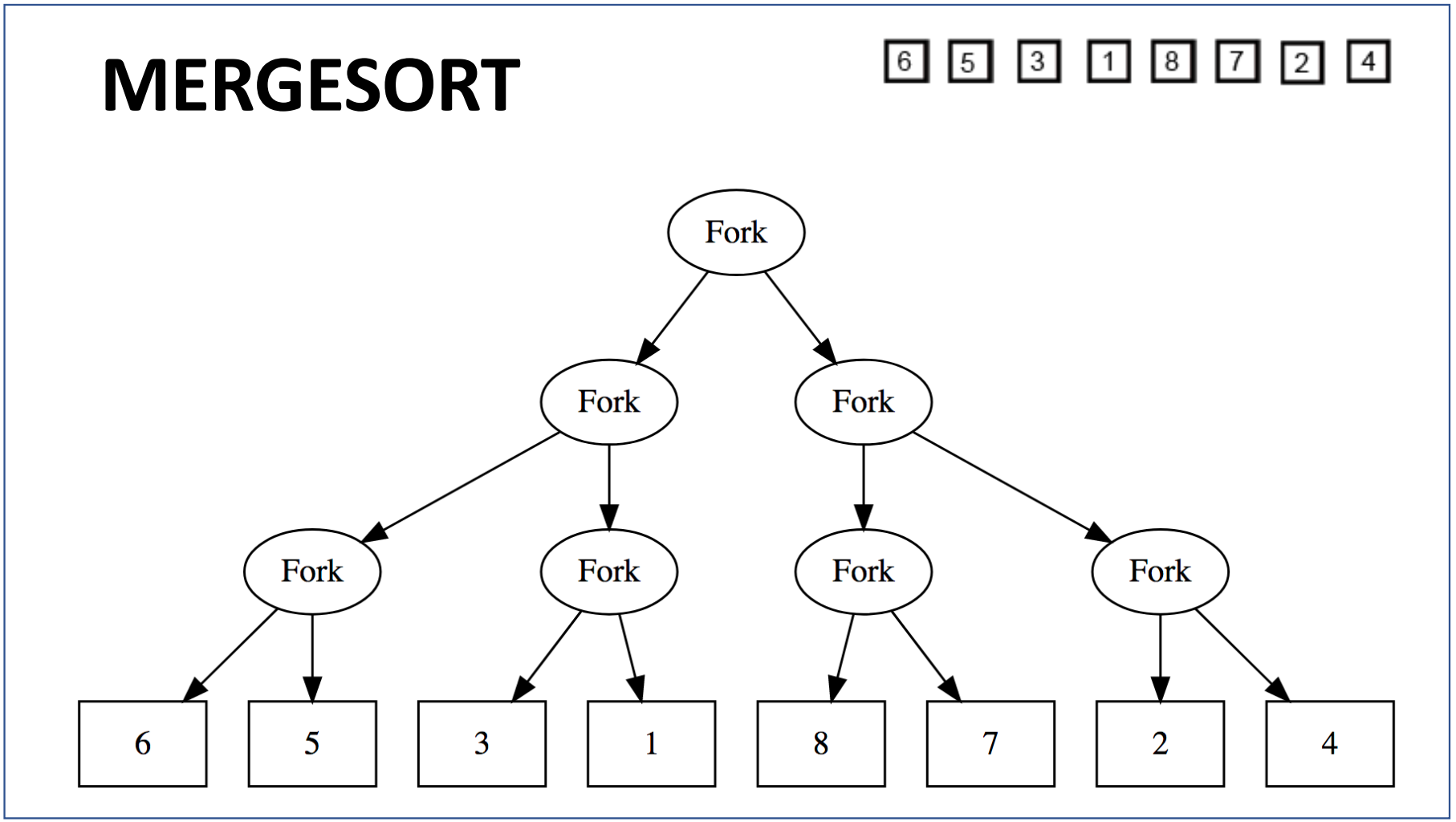
MERGESORT

```
a.hs
mSort :: Ord a => [a] -> [a]


l1,l2) = sep l
rge(mSort l1, mSort l2)
```

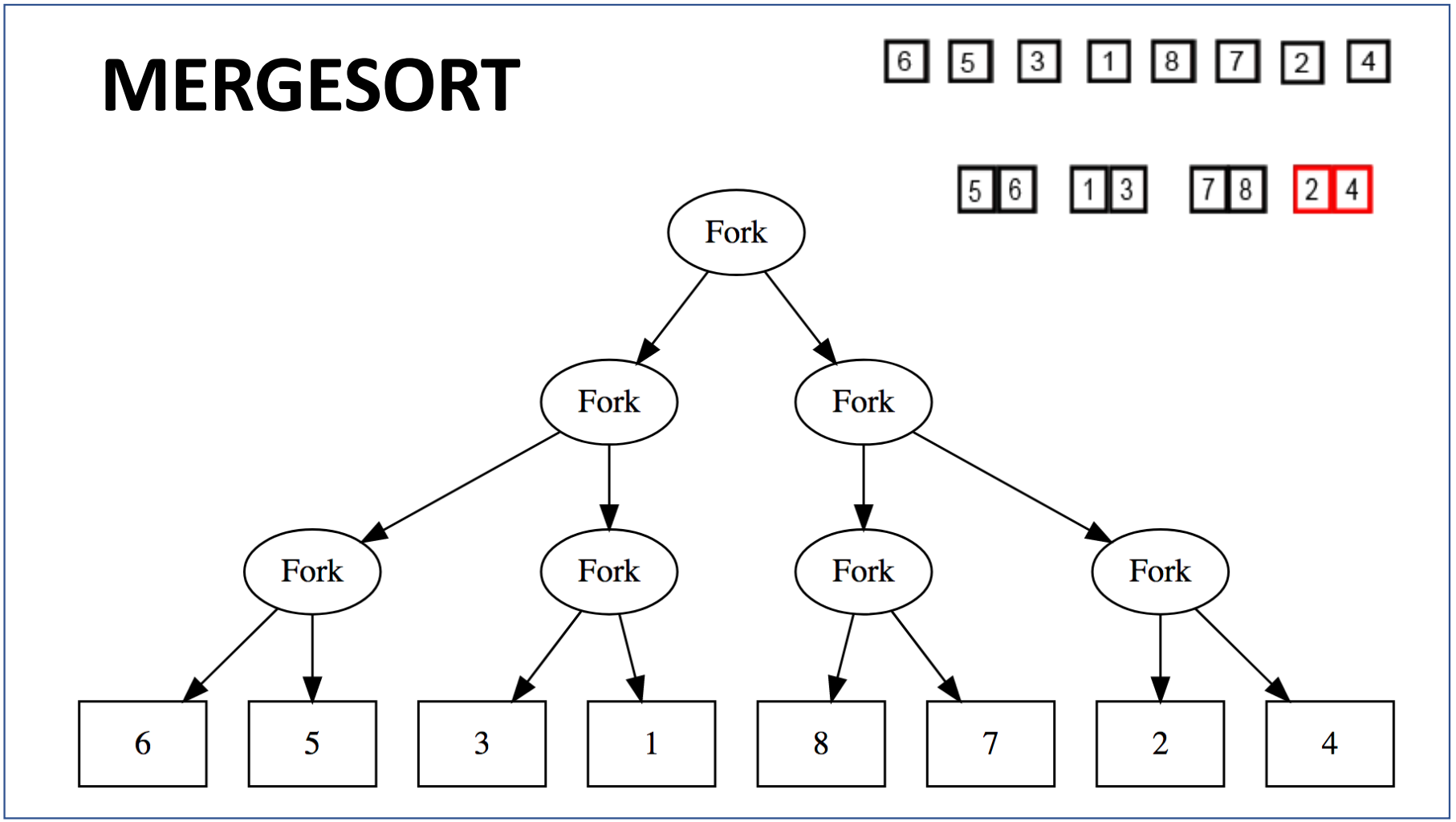
$$divide : A^* \rightarrow A + (A^* \times A^*)$$



MERGESORT



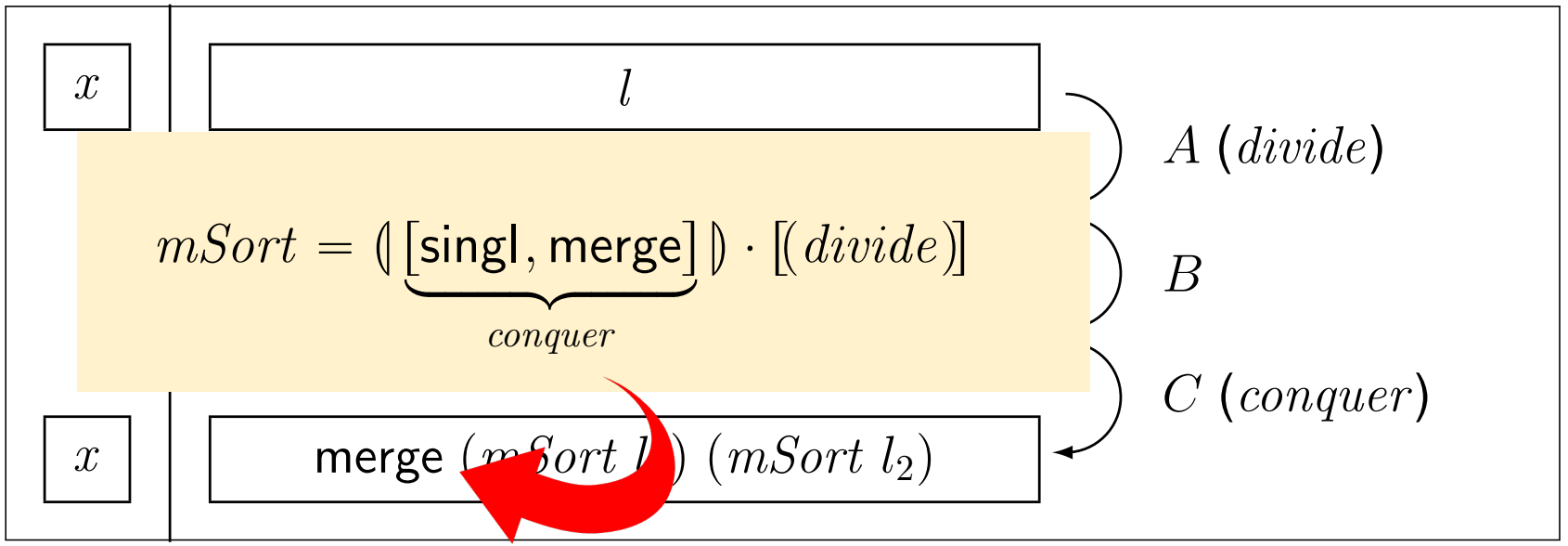
MERGESORT


6 5 3 1 8 7 2 4
5 6 1 3 7 8 2 4
Fork
Fork
Fork
Fork
Fork
Fork
Fork
6
5
3
1
8
7
2
4

MERGESORT

a.hs

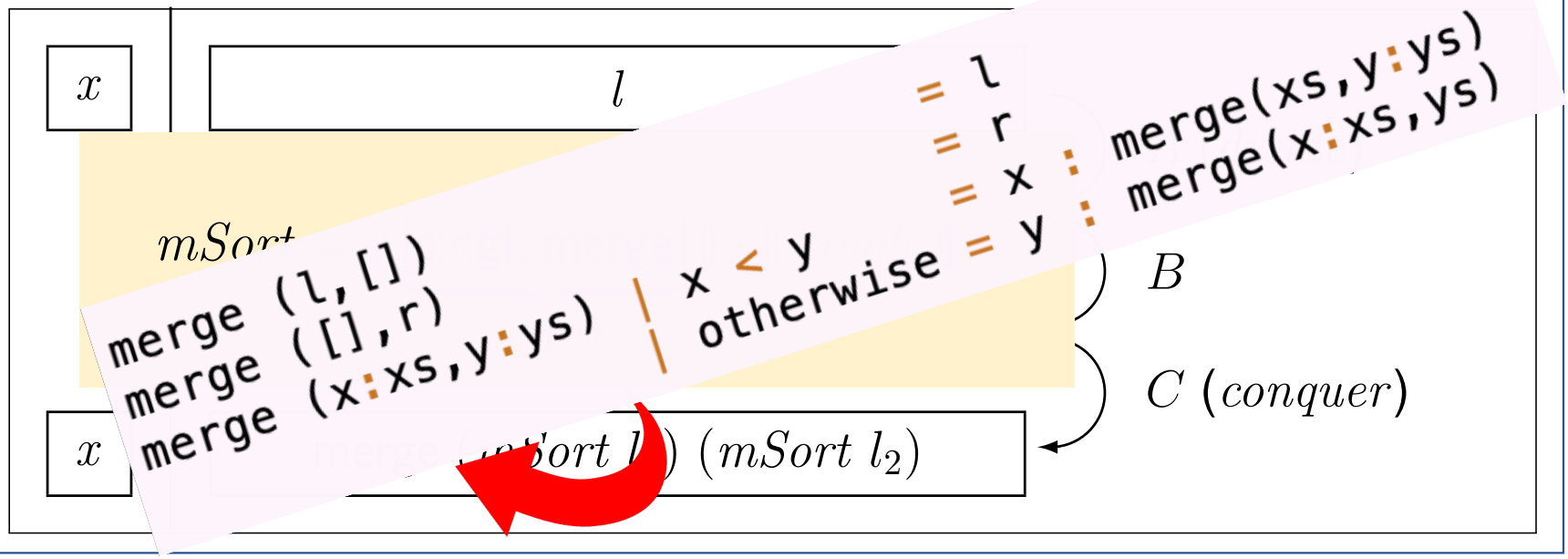
```
mSort :: Ord a => [a] -> [a]
mSort [] = []
mSort [x] = [x]
mSort l = let (l1,l2) = sep l
          in merge(mSort l1, mSort l2)
```



MERGESORT

a.hs

```
mSort :: Ord a => [a] -> [a]
mSort [] = []
mSort [x] = [x]
mSort l = let (l1,l2) = sep l
          in merge(mSort l1
```

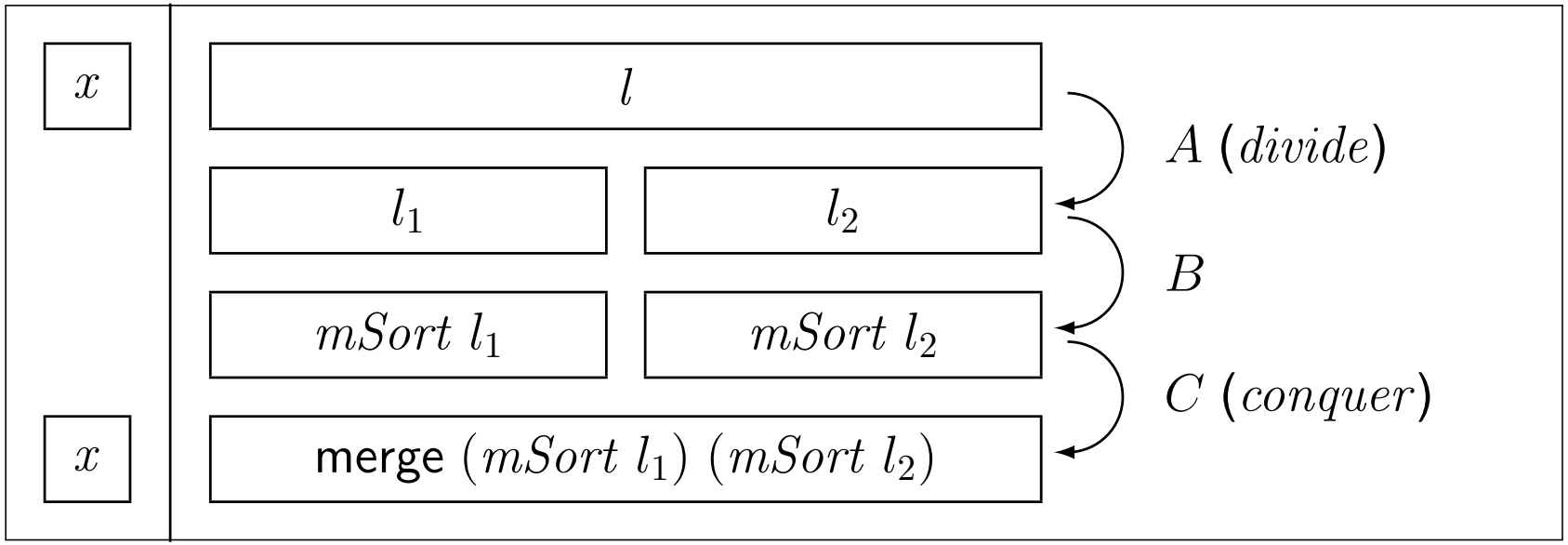



```
merge (l,[]) = l
merge ([],r) = r
merge (x:xs,y:ys) | x < y = x : merge(xs,y:ys)
                  | otherwise = y : merge(x:xs,ys)
```

MERGESORT

a.hs

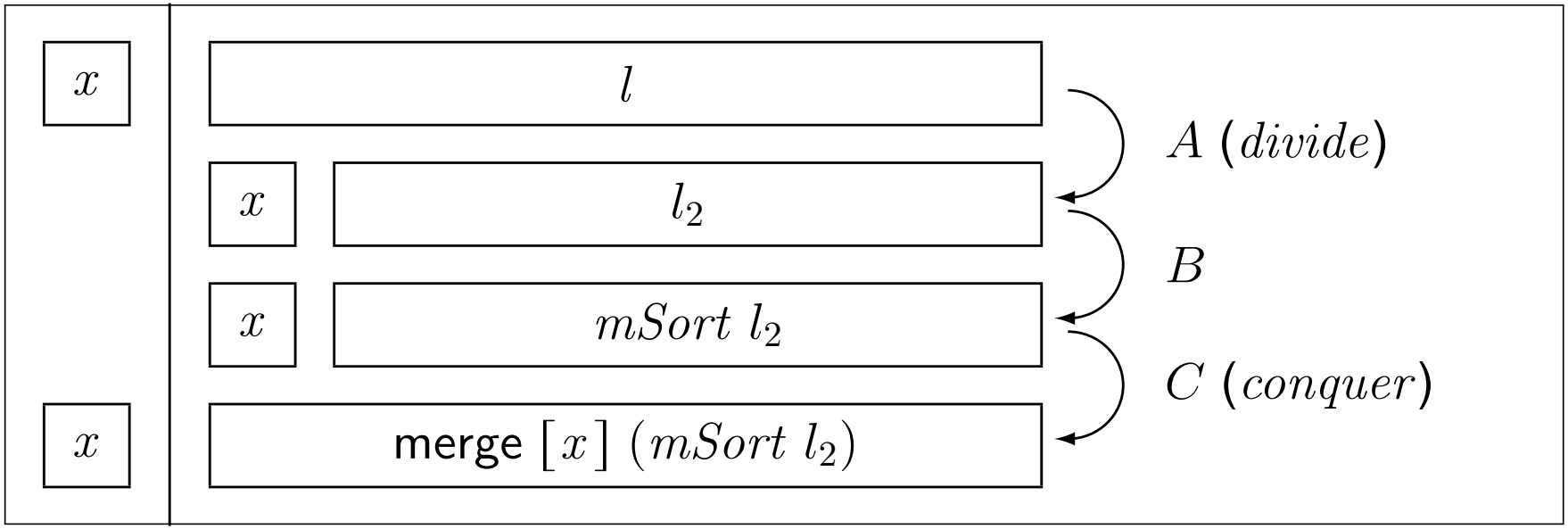
```
mSort :: Ord a => [a] -> [a]
mSort [] = []
mSort [x] = [x]
mSort l = let (l1,l2) = sep l
          in merge(mSort l1, mSort l2)
```



MERGESORT

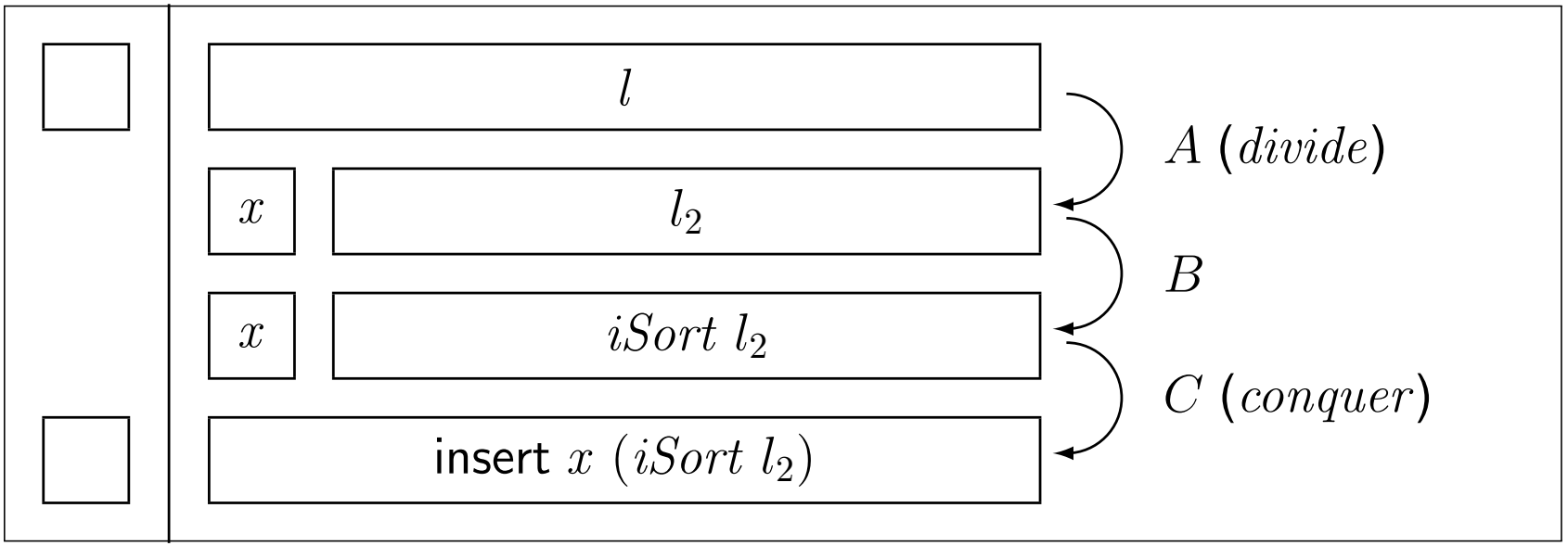
Particular case:

```
merge :: Ord a => ([a], [a]) -> [a]
merge ([x],[])                  = [x]
merge ([],r)                    = r
merge ([x],y:ys) | x < y        = x : merge([],y:ys)
                 ) | otherwise = y : merge(x:[],ys
```



INSERTION SORT

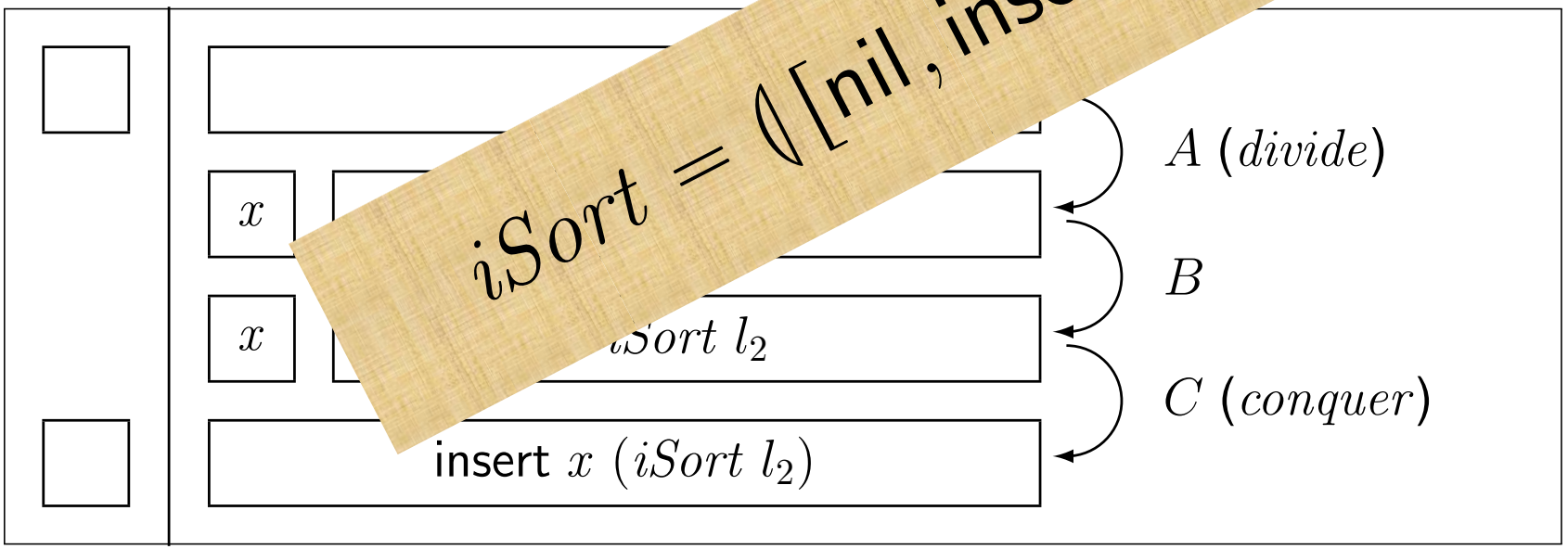
```haskell
insert :: Ord t => t -> [t] -> [t]
insert x []                     = [x]
insert x (y:ys) | x < y         = x:y:ys
                | otherwise = y:insert x ys
```



INSERTION SORT

```
insert :: Ord t => t -> [t] -> [
insert x []
insert x (y:ys) | x < y
                | ot
                                  x ys
```

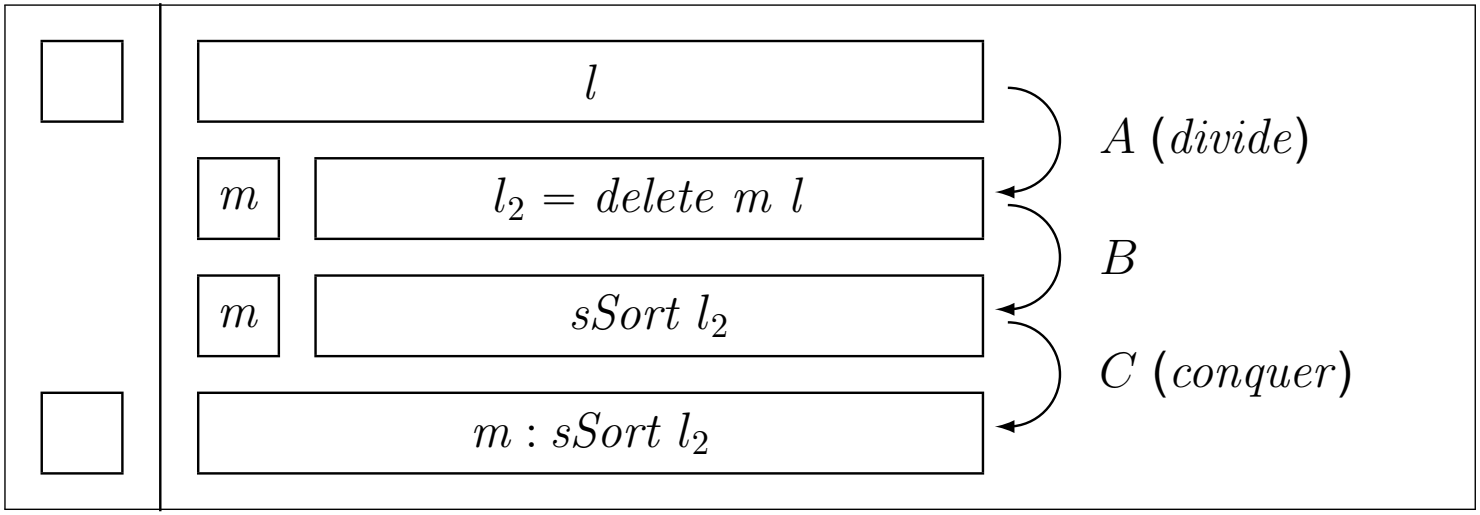
$$iSort = (\!|[nil, \widehat{insert}]|\!)$$



SELECTION SORT

```haskell
1
2  sSort :: Ord a => [a] -> [a]
3  sSort = anaL divide where
4     divide [] = i1()
5     divide (xs) = let m = minimum xs
6                   in  i2(m, delete m xs)
```

Selection sort ($m = minimum\ l$):



SELECTION SORT

```haskell
1
2  sSort :: Ord a => [a] -> [a]
3  sSort = anaL divide where
4     divide [] = i1()
5     divide (xs) = let m = minimum xs
6                   in  i2(m, delete m xs)
```

Selection sort ($m = minimum\ l$):

ANA, CATA & HILO

$$C \xleftarrow{(\!|f|\!)} \mathsf{T} \xleftarrow{[\!(g)\!]} A$$

$$[\![f, g]\!] = (\!|f|\!) \cdot [\!(g)\!]$$

ALGORITHM CLASSIFICATION

	Singleton	Equal-size
Easy Split/Hard Join	Insertion Sort	Merge Sort
Hard Split/Easy Join	Selection Sort	Quick Sort

NB:
'Split' = *divide*
'Join' = *conquer*

Cálculo de Programas
T11