

# Cálculo de Programas

Class T01

Introduction to the course. Website and other administrative details.

# Cálculo de Programas

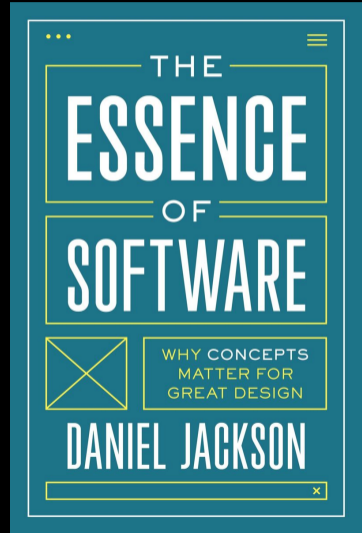
Class T02

**J.N. Oliveira**





”(...) The best services revolve around **a small number of concepts** that are **well designed and easy** (...) **to understand and use**, and their innovations often involve simple but compelling new concepts.”



In  
The Essence of Software  
by  
Prof. Daniel Jackson, MIT  
(2021)



... small number

... small number

... well defined

- ... small number
- ... well defined
- ... easy to understand



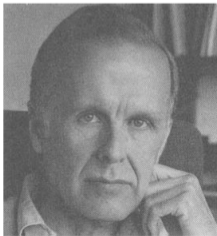
Urgently needed in software design



# John Backus – Turing Award (1978)

## Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs

John Backus  
IBM Research Laboratory, San Jose



Conventional programming languages are growing ever more enormous, but not stronger. Inherent defects at the most basic level cause them to be both fat and weak: their primitive word-at-a-time style of programming inherited from their common ancestor—the von Neumann computer, their close coupling of semantics to state transitions, their division of programming into a world of expressions and a world of statements, their inability to effectively use powerful combining forms for building new programs from existing ones, and their lack of useful mathematical properties for reasoning about programs.

An alternative functional style of programming is

# Motivation — back to the late 1990s





## From a mobile phone manufacturer

*(...) For each **list of calls** stored in the mobile phone (e.g. numbers dialed, SMS messages, lost calls), the **store** operation should work in a way such that **(a)** the more recently a call is made the more accessible it is; **(b)** no number appears twice in a list; **(c)** only the last 10 entries in each list are stored.*

## From a mobile phone manufacturer

```
store :: Call -> [Call] -> [Call]
```

```
store c l = take 10 (store' c l)
```

```
store' :: Call -> [Call] -> [Call]
```

```
store' c l = c : filter (/=c) l
```

## From a mobile phone manufacturer

```
store :: Call -> [Call] -> [Call]
store c l = take 10 (c : filter (/=c) l)
```

## Compare with ...

```
public void store10(string phoneNumber)
{
    System.Collections.ArrayList auxList =
        new System.Collections.ArrayList();
    auxList.Add(phoneNumber);
    auxList.AddRange(
        this.filteratmost9(phoneNumber) );
    this.callList = auxList;
}
```

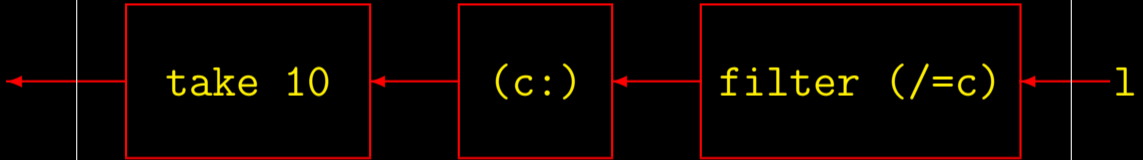
+ `filteratmost9` (next slide)

## Compare with ...

```
public System.Collections.ArrayList filteratmost9(string n)
{
    System.Collections.ArrayList retList =
        new System.Collections.ArrayList();
    int i=0, m=0;
    while((i < this.callList.Count) && (m < 9))
    {
        if ((string)this.callList[i] != n)
        {
            retList.Add(this.callList[i]);
            m++;
        }
        i++;
    }
    return retList;
}
```

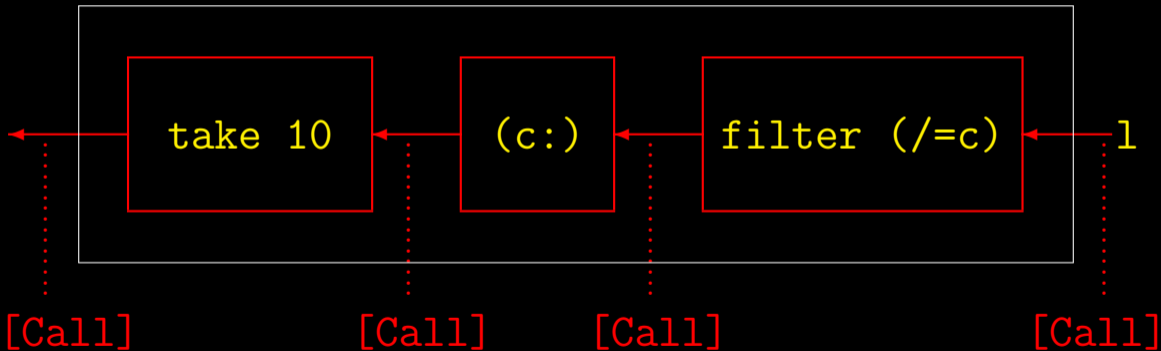
# From a mobile phone manufacturer

```
store c :: [Call] -> [Call]
```



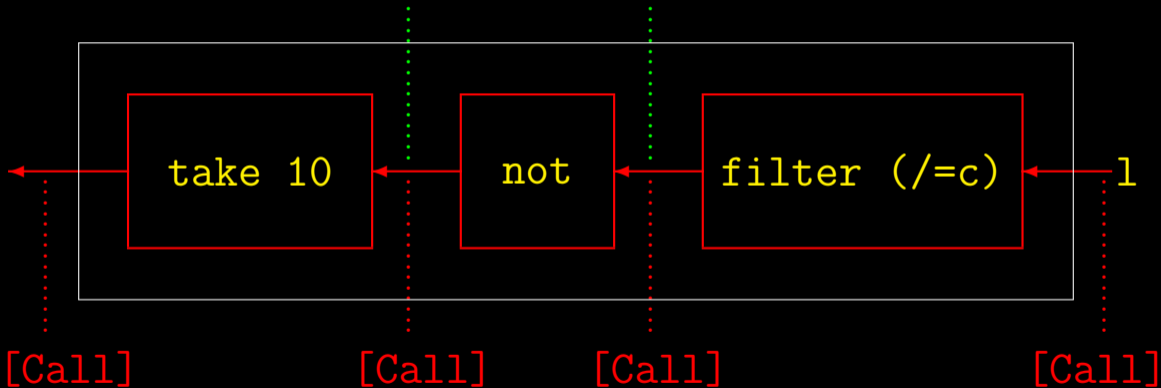
# From a mobile phone manufacturer

```
store c :: [Call] -> [Call]
```



Oops!

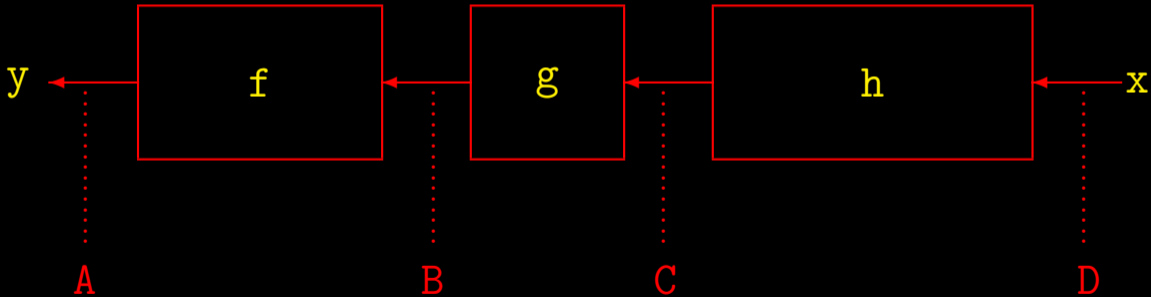
Bool (!) Bool (!)





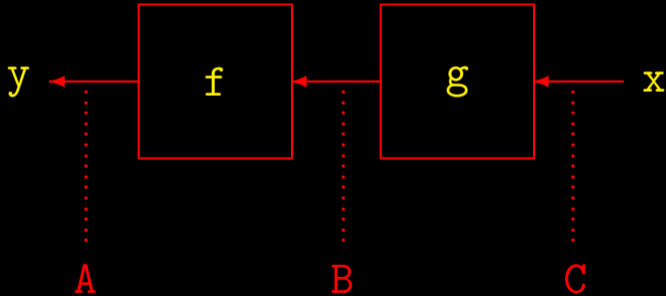
In general

$$y = f(g(h(x)))$$



In general

$$y = f(g x)$$



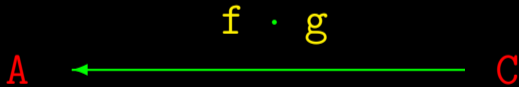
# Simplification

$$y = f(g x)$$



# Composition

$$y = f(g \ x)$$



$$y = (f \cdot g) \ x$$

# Composition

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

# Composition

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

$$(a + b) + c = a + (b + c)$$

# Composition

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

$$(a + b) + c = a + (b + c)$$

$$f \cdot g \cdot h$$

$$a + b + c$$

# Composition

$$\text{store } c = \text{take } 10 \cdot \underbrace{(\text{c:}) \cdot \text{filter } (\neq c)}_{\text{store}' c}$$



# Composition

$$\text{store } c = \text{take } 10 \cdot \underbrace{((c:) \cdot \text{filter } (\neq c))}_{\text{store}' c}$$

i.e.

$$\text{take } 10 \cdot ((c:) \cdot \text{filter } (\neq c))$$

# Composition

$$\text{store } c = \text{take } 10 \cdot \underbrace{((c:) \cdot \text{filter } (\neq c))}_{\text{store}' c}$$

i.e.

$$\text{take } 10 \cdot ((c:) \cdot \text{filter } (\neq c))$$

the same as

$$(take\ 10 \cdot (c:)) \cdot filter\ (\neq c)$$

# Composition

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

$$(a + b) + c = a + (b + c)$$

# Composition

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

$$(a + b) + c = a + (b + c)$$

$$a + 0 = 0 + a = a$$

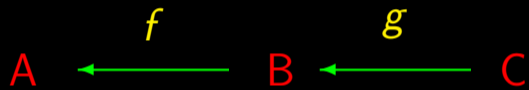
# Composition

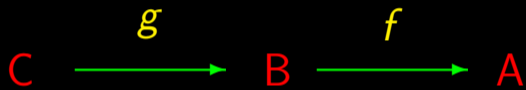
$$(f \cdot g) \cdot h = f \cdot (g \cdot h)$$

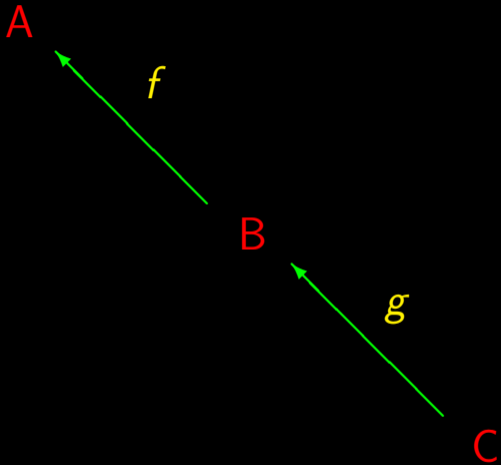
$$(a + b) + c = a + (b + c)$$

$$a + 0 = 0 + a = a$$

$$f \cdot ? = ? \cdot f = f$$













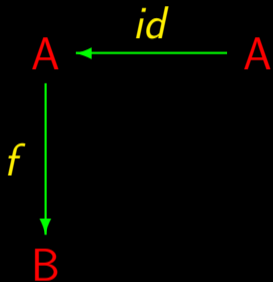




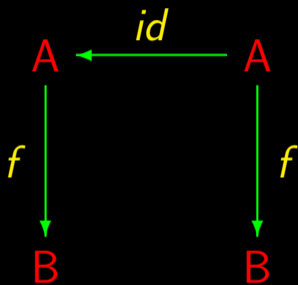


$$id\ a = a$$

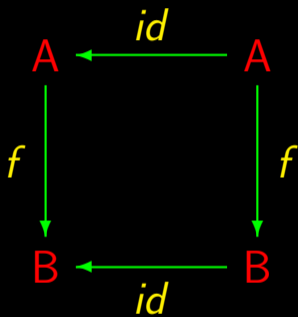
# Identity



# Identity

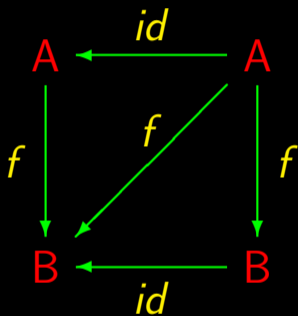


# Identity

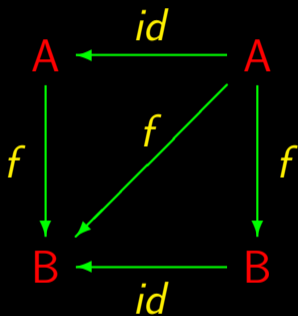




# Identity



# Identity



$$f \cdot id = f = id \cdot f$$

# Composition and identity

Associativity:

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

# Composition and identity

Associativity:

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

“Natural-*id*”:

$$f \cdot id = f = id \cdot f$$

$$f \cdot g$$

$$f \cdot g$$

$$f \times g ?$$

$$f \cdot g$$

$$f \times g ?$$

$$f + g ?$$

$$C \xrightarrow{f} B \quad \text{and} \quad A \xrightarrow{g} C$$



$$C \xrightarrow{f} B \quad \text{and} \quad A \xrightarrow{g} C$$

$$\text{Composition: } A \xrightarrow{f \cdot g} B$$

$$C \xrightarrow{f} B \quad \text{and} \quad A \xrightarrow{g} C$$

$$\text{Composition: } A \xrightarrow{f \cdot g} B$$

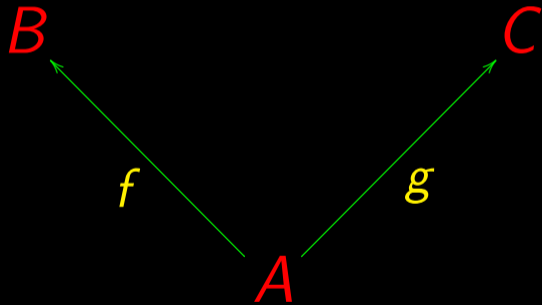
$$B \xleftarrow{f} C \quad \text{e} \quad C \xleftarrow{g} A$$

$$D \xrightarrow{f} B$$

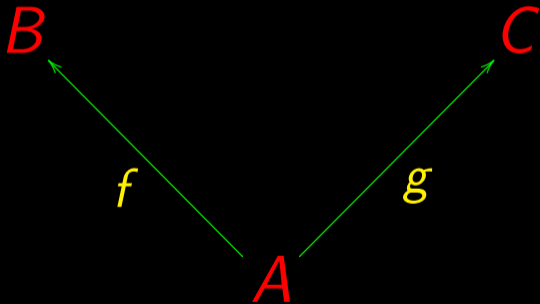
$$A \xrightarrow{g} C$$

?

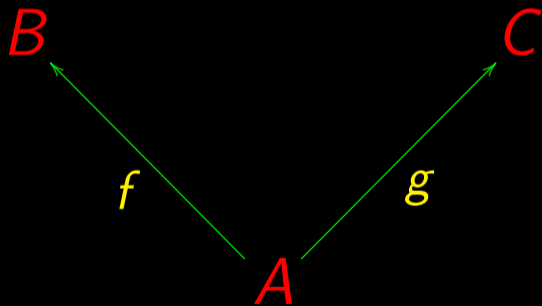
$D = A ?$



$D = A ?$



$f a \dots g a$



$(f\ a, g\ a)$

## Cartesian product

$$A \times B = \{(a, b) \mid a \in A \wedge b \in B\}$$

## Cartesian product

$$A \times B = \{(a, b) \mid a \in A \wedge b \in B\}$$

$$f : a \in B$$



## Cartesian product

$$A \times B = \{(a, b) \mid a \in A \wedge b \in B\}$$

$$f \ a \in B$$

$$g \ a \in C$$

## Cartesian product

$$A \times B = \{(a, b) \mid a \in A \wedge b \in B\}$$

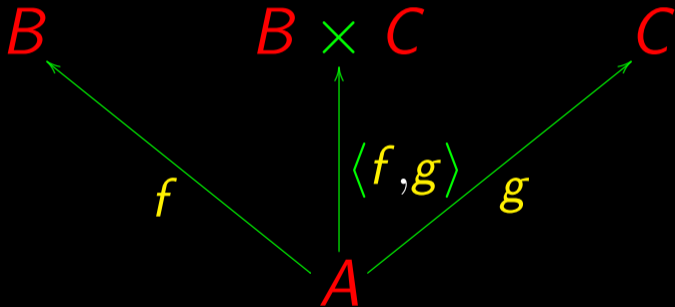
$$f \ a \in B$$

$$g \ a \in C$$

---

$$(f \ a, g \ a) \in B \times C$$

# “Split”



$$\langle f, g \rangle a = (f a, g a)$$

# Product

$$A \times B = \{(a, b) \mid a \in A \wedge b \in B\}$$

# Product

$$A \times B = \{(a, b) \mid a \in A \wedge b \in B\}$$

$$\pi_1 : A \times B \rightarrow A$$

$$\pi_1(a, b) = a$$

# Product

$$A \times B = \{(a, b) \mid a \in A \wedge b \in B\}$$

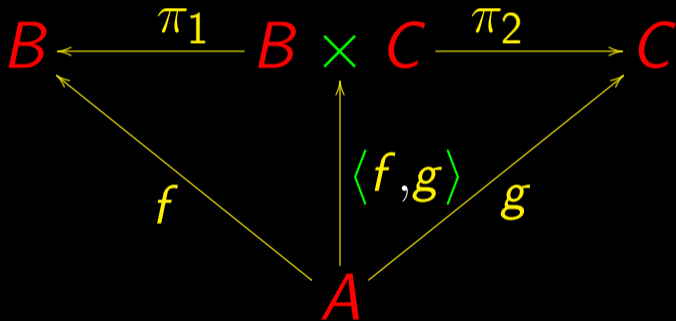
$$\pi_1 : A \times B \rightarrow A$$

$$\pi_2 : A \times B \rightarrow B$$

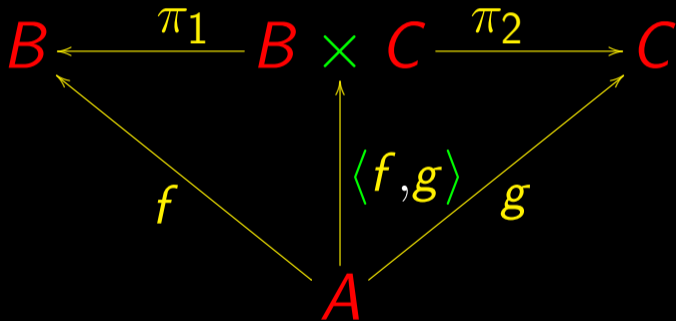
$$\pi_1(a, b) = a$$

$$\pi_2(a, b) = b$$

# Product



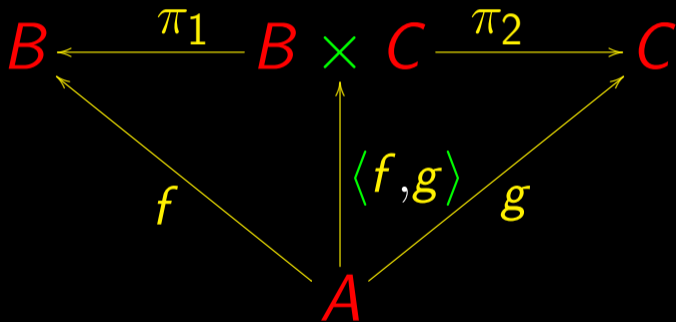
# Product



$$\pi_1 \cdot \langle f, g \rangle = f$$



# Product



$$\pi_1 \cdot \langle f, g \rangle = f$$

$$\pi_2 \cdot \langle f, g \rangle = g$$

Product

$$\langle f, g \rangle$$

# Product

$$\langle f, g \rangle$$

$f$  and  $g$  in parallel

$f$  “split”  $g$

# Product

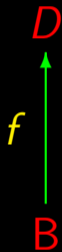
$$\langle f, g \rangle$$

$f$  and  $g$  in parallel

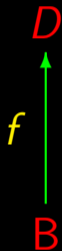
$f$  “split”  $g$

$$\langle f, g \rangle a = (f a, g a)$$

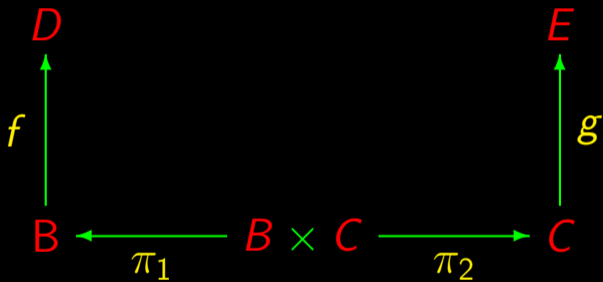
Product



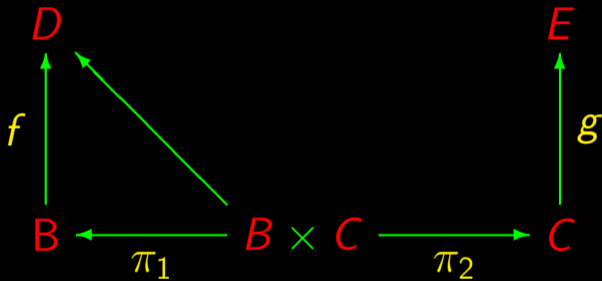
# Product



# Product

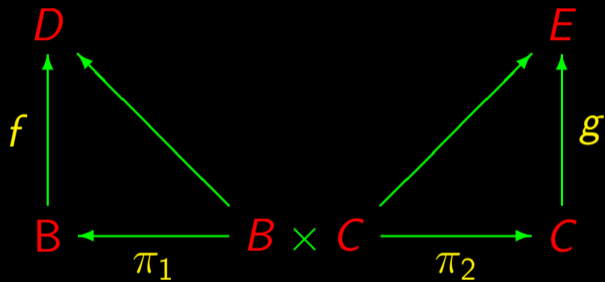


# Product

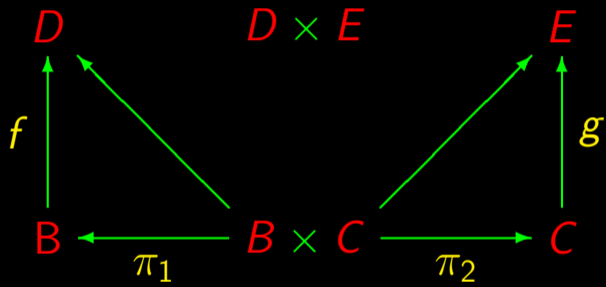




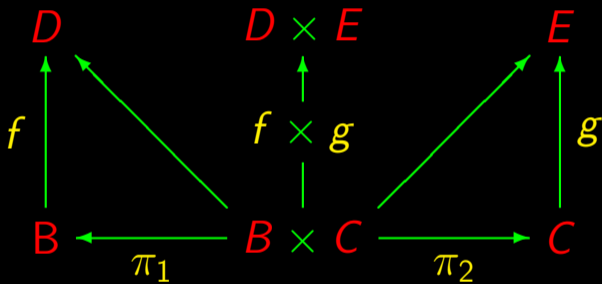
# Product



# Product

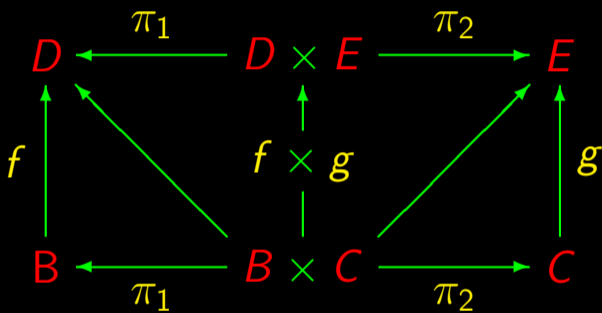


# Product



$$f \times g = \langle f \cdot \pi_1, g \cdot \pi_2 \rangle$$

# Product



$$f \times g = \langle f \cdot \pi_1, g \cdot \pi_2 \rangle$$

# Summing up

$$f \cdot g$$

## Summing up

$f \cdot g$

Sequential composition

## Summing up

$$f \cdot g$$
$$\langle f, g \rangle$$

Sequential composition

## Summing up

$f \cdot g$

$\langle f, g \rangle$

Sequential composition

Parallel composition



## Summing up

$f \cdot g$

$\langle f, g \rangle$

Sequential composition

Parallel composition (**synchronous**)

## Summing up

$f \cdot g$

$\langle f, g \rangle$

$f \times g$

Sequential composition

Parallel composition (**synchronous**)

## Summing up

$f \cdot g$

Sequential composition

$\langle f, g \rangle$

Parallel composition (**synchronous**)

$f \times g$

Parallel composition

## Summing up

$f \cdot g$

Sequential composition

$\langle f, g \rangle$

Parallel composition (**synchronous**)

$f \times g$

Parallel composition (**asynchronous**)

## Summing up

$f \cdot g$

Sequential composition

$\langle f, g \rangle$

Parallel composition (**synchronous**)

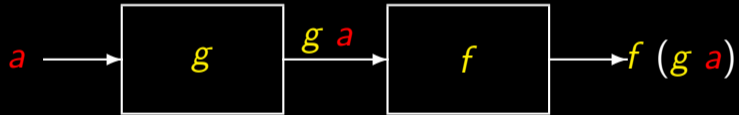
$f \times g$

Parallel composition (**asynchronous**)

**Compositional programming**

## Summing up

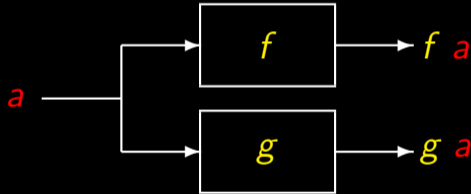
$$(f \cdot g) a = f (g a) \quad (2.6)$$



Function **composition**

## Summing up

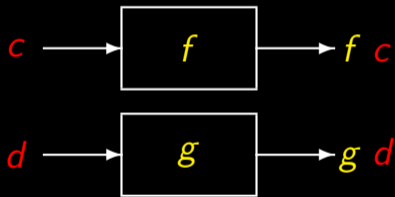
$$\langle f, g \rangle a = (f a, g a) \quad (2.20)$$



Functional “splits”

## Summing up

$$f \times g = \langle f \cdot \pi_1, g \cdot \pi_2 \rangle \quad (2.24)$$



Functional **products**



$f \cdot g$

Sequential composition

$\langle f, g \rangle$

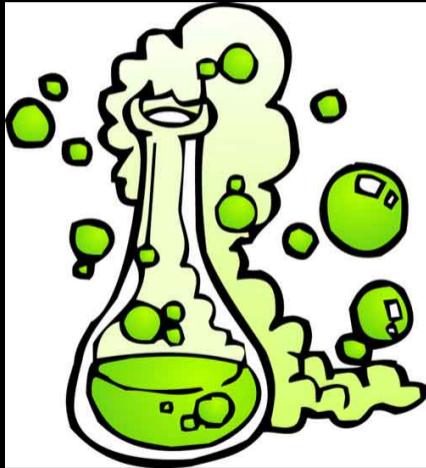
Parallel composition (**synchronous**)

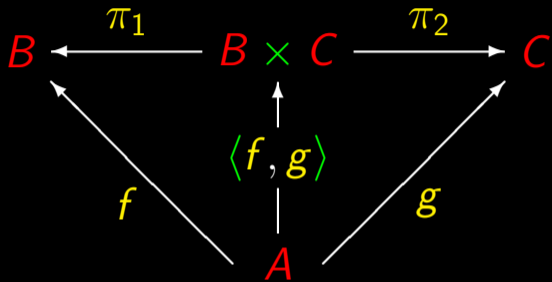
$f \times g$

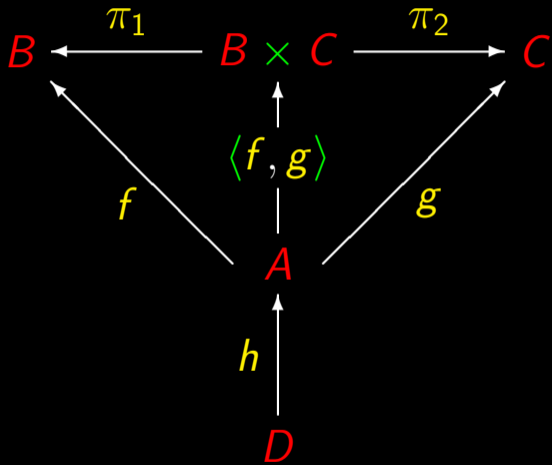
Parallel composition (**asynchronous**)

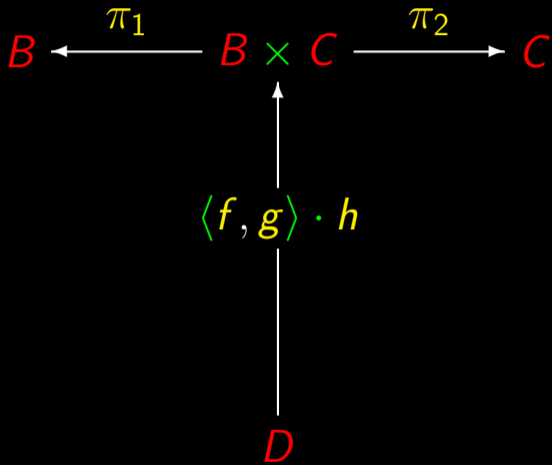
**Compositional programming**

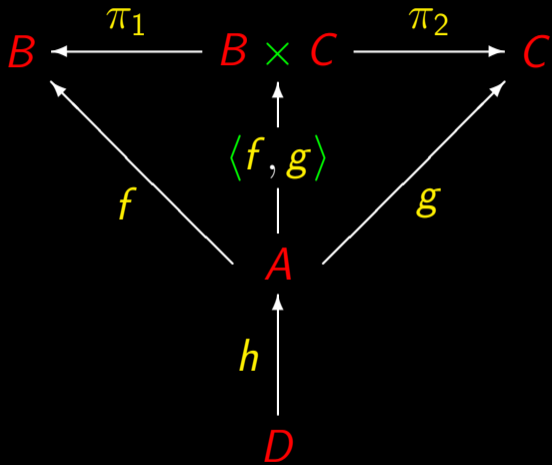
Calculus?

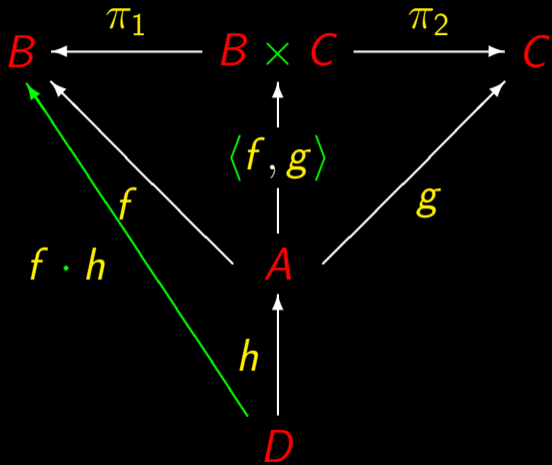


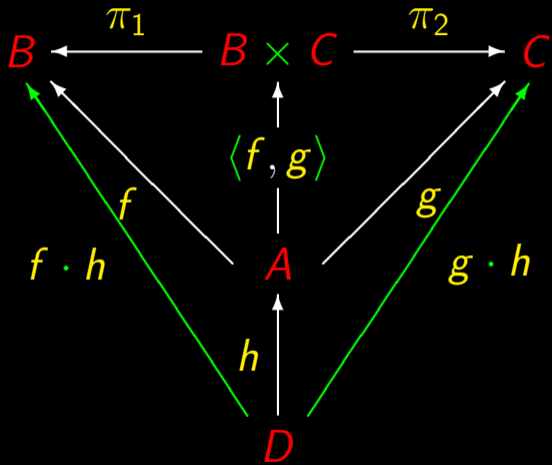




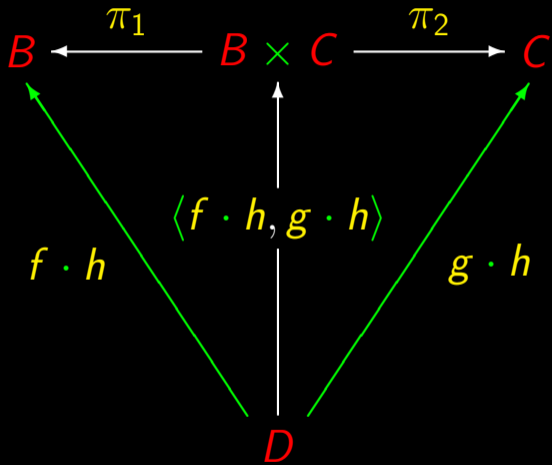












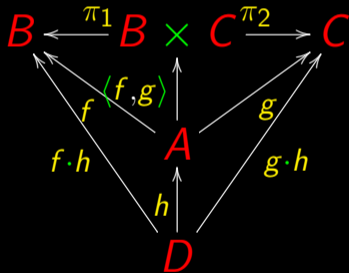
$$B \xleftarrow{\pi_1} B \times C \xrightarrow{\pi_2} C$$

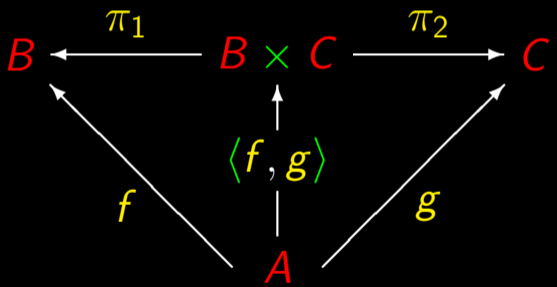
$$\langle f, g \rangle \cdot h = \langle f \cdot h, g \cdot h \rangle$$

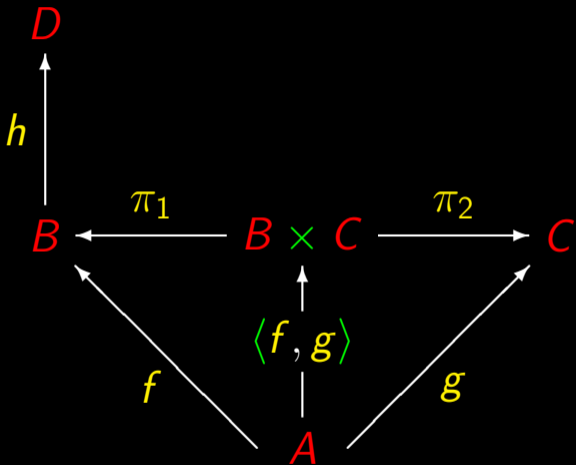
$D$

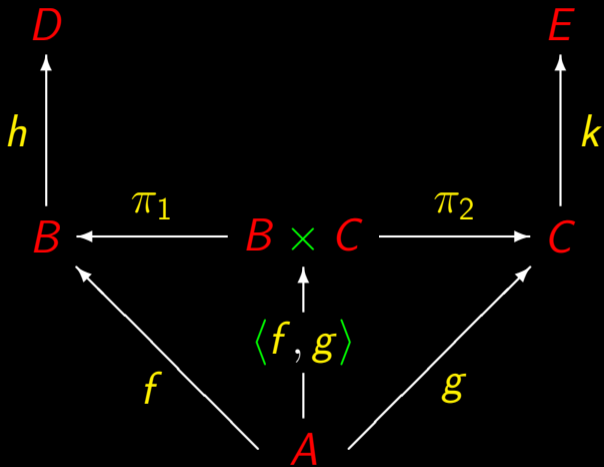
## $\times$ -Fusion

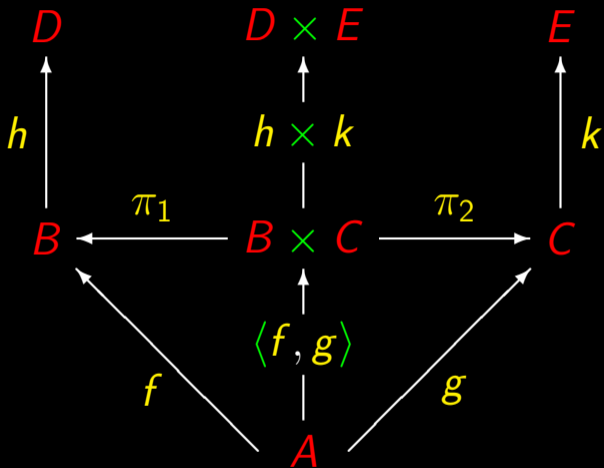
$$\langle f, g \rangle \cdot h = \langle f \cdot h, g \cdot h \rangle \quad (2.26)$$

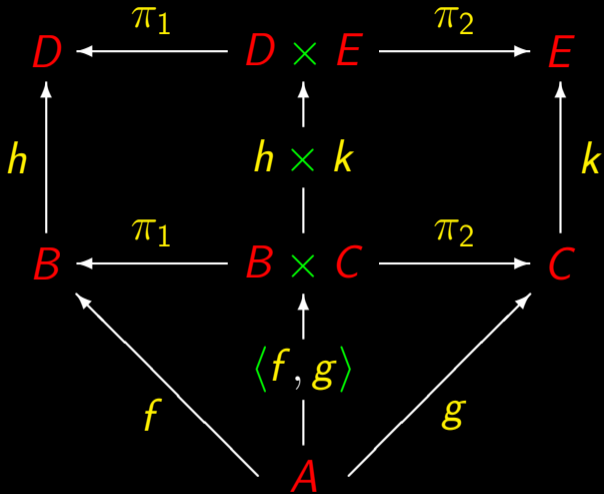




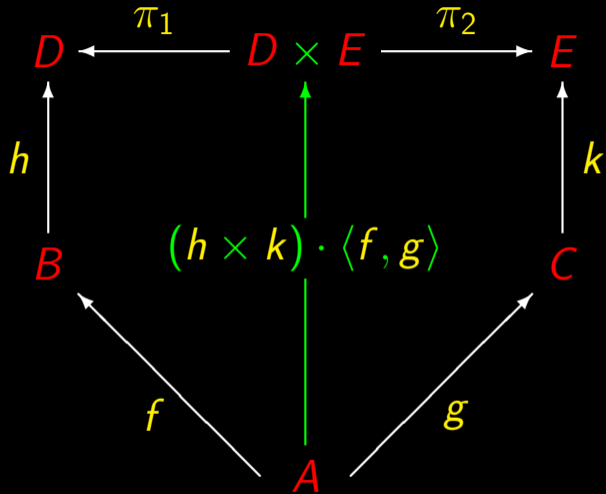


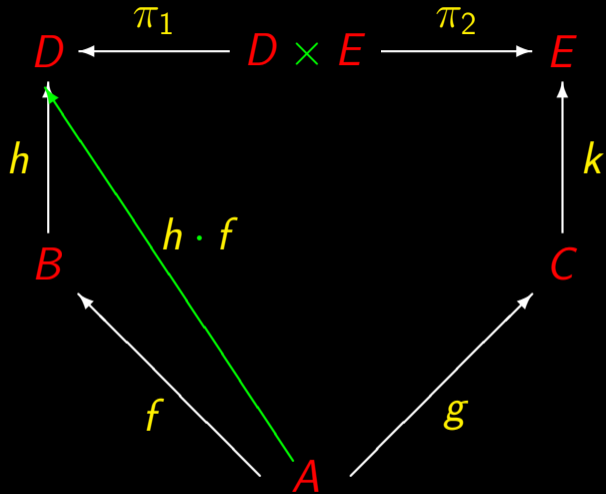


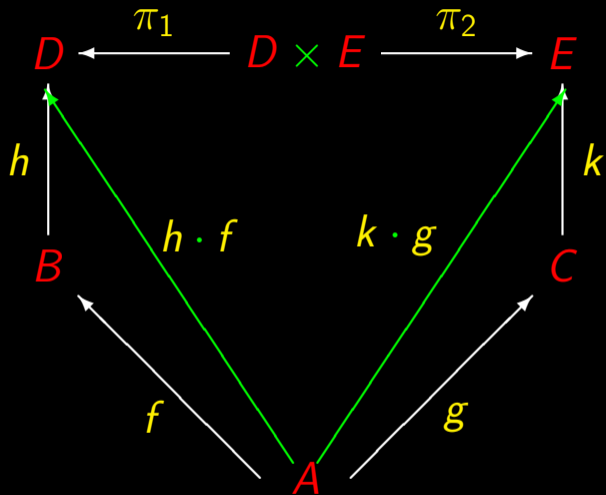


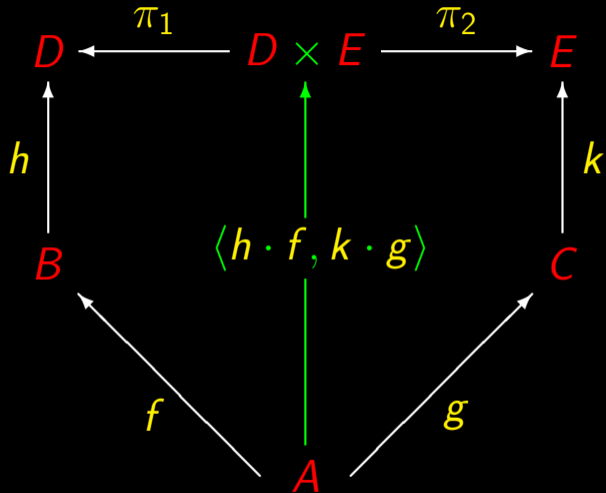


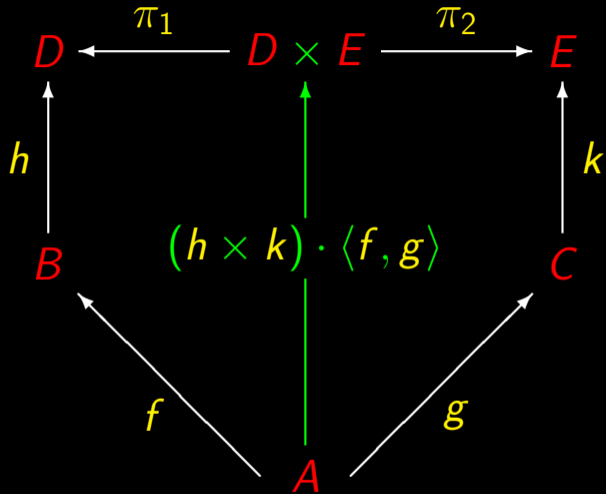






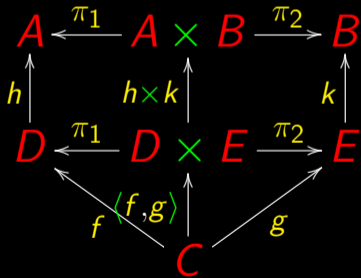


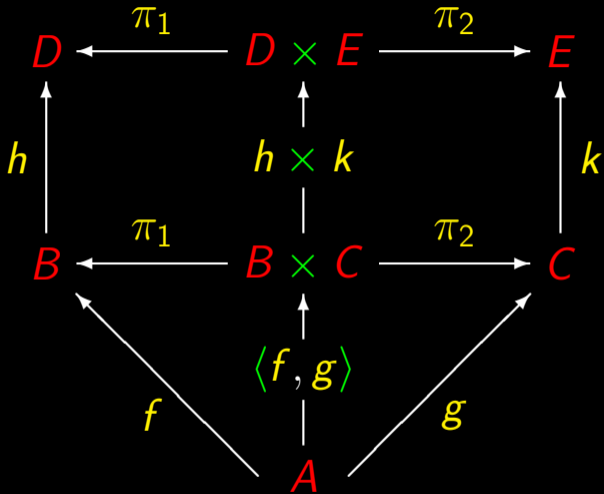


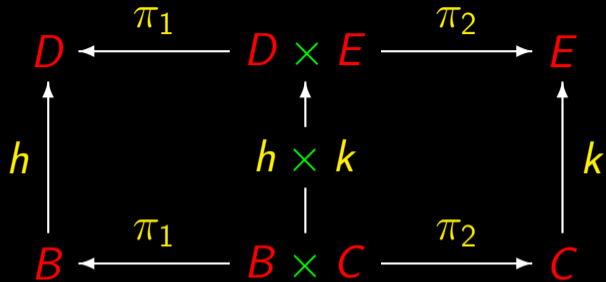


## $\times$ -Absorption

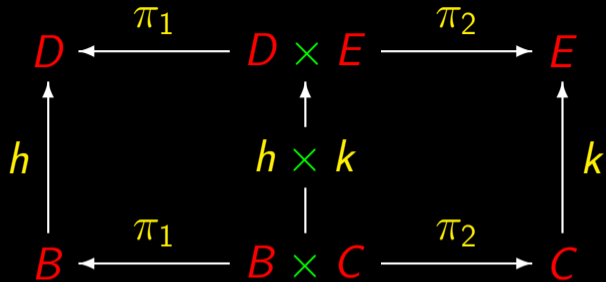
$$(h \times k) \cdot \langle f, g \rangle = \langle h \cdot f, k \cdot g \rangle \quad (2.27)$$



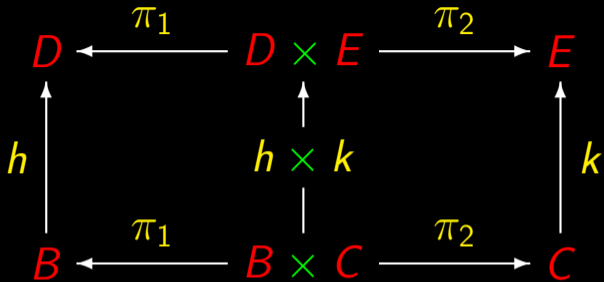








$$\pi_1 \cdot (h \times k) = h \cdot \pi_1$$



$$\pi_1 \cdot (h \times k) = h \cdot \pi_1$$

$$\pi_2 \cdot (h \times k) = k \cdot \pi_2$$

## Natural- $\pi_1$ , natural- $\pi_2$

$$\pi_1 \cdot (h \times k) = h \cdot \pi_1 \quad (2.28)$$

$$\pi_2 \cdot (h \times k) = k \cdot \pi_2 \quad (2.29)$$

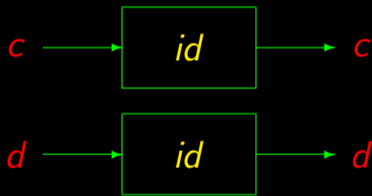
$$\begin{array}{ccccc} D & \xleftarrow{\pi_1} & D \times E & \xrightarrow{\pi_2} & E \\ \uparrow h & & \uparrow h \times k & & \uparrow k \\ B & \xleftarrow{\pi_1} & B \times C & \xrightarrow{\pi_2} & C \end{array}$$

# Cálculo de Programas

Class T03

## Functor- $id$ - $\times$

$$id \times id = id \tag{2.31}$$



Product of two identities is an identity.

## $\times$ -Functor

$$(f \times h) \cdot (g \times k) = (f \cdot g) \times (h \cdot k) \quad (2.30)$$

**Composition** of products is a **product** of compositions.

## Two laws still missing

×-Reflexion

$$\langle \pi_1, \pi_2 \rangle = id \quad (2.32)$$

## Two laws still missing

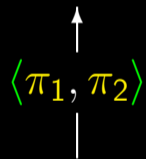
×-Reflexion

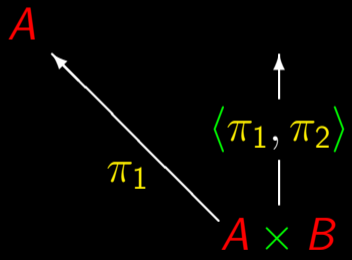
$$\langle \pi_1, \pi_2 \rangle = id \quad (2.32)$$

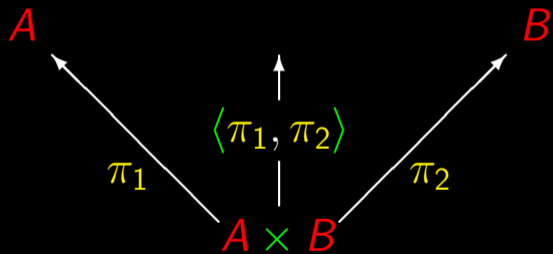
×-Eq

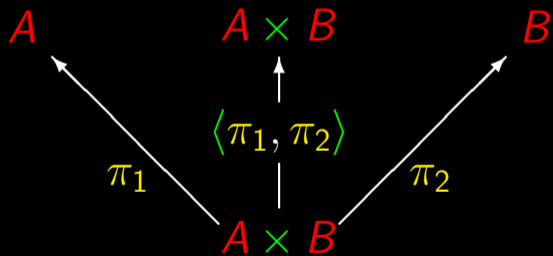
$$\langle i, j \rangle = \langle f, g \rangle \Leftrightarrow \begin{cases} i = f \\ j = g \end{cases} \quad (2.64)$$

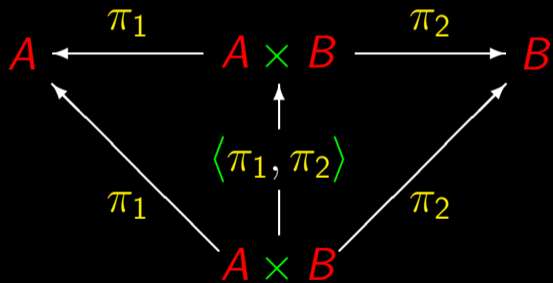


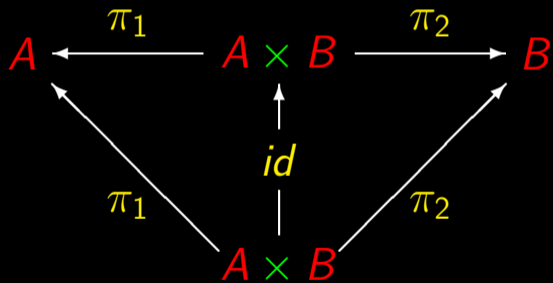
$$\langle \pi_1, \pi_2 \rangle$$
A diagram consisting of a central mathematical expression  $\langle \pi_1, \pi_2 \rangle$  enclosed in green angle brackets. A vertical white line passes through the center of the expression, extending both above and below. At the top end of this line is a white arrowhead pointing upwards.





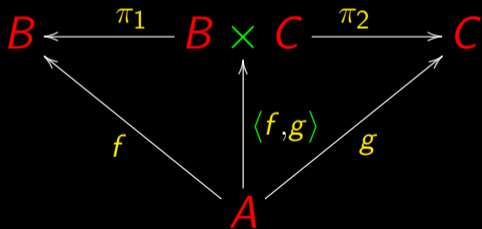






Finally the more important...

Recall  $\times$ -cancellation:



$$\pi_1 \cdot \langle f, g \rangle = f$$

$$\pi_2 \cdot \langle f, g \rangle = g$$

$$\begin{cases} \pi_1 \cdot \langle f, g \rangle = f \\ \pi_2 \cdot \langle f, g \rangle = g \end{cases}$$



$$\begin{cases} \pi_1 \cdot \langle f, g \rangle = f \\ \pi_2 \cdot \langle f, g \rangle = g \end{cases}$$

$$k = \langle f, g \rangle \Rightarrow \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases}$$

$$\begin{cases} \pi_1 \cdot \langle f, g \rangle = f \\ \pi_2 \cdot \langle f, g \rangle = g \end{cases}$$

$$k = \langle f, g \rangle \Leftrightarrow \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases}$$

## x-Universal

$$k = \langle f, g \rangle \Leftrightarrow \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases}$$

## ×-Universal

### Existence

$$k = \langle f, g \rangle \Rightarrow \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases}$$

“There exists a **solution** —  $k = \langle f, g \rangle$  — for the equations on the right”

## x-Universal

### Unicity

$$k = \langle f, g \rangle \iff \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases}$$

“Such a solution,  $k = \langle f, g \rangle$ , is **unique**”

# Equations!

$$\begin{cases} x = 2y \\ z = \frac{y}{3} \\ x + y + z = 10 \end{cases}$$

# Equations!

$$\begin{cases} x = 2y \\ z = \frac{y}{3} \\ x + y + z = 10 \end{cases} \Leftrightarrow \begin{cases} x = 6 \\ z = 1 \\ y = 3 \end{cases}$$

# Equations!

## Problem

*Solve the equation*

$$\langle f, g \rangle = id$$

*for  $f$  and  $g$ .*



# Equations!

## Calculation

$$\text{In } k = \langle f, g \rangle \Leftrightarrow \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases}$$

# Equations!

## Calculation

$$\text{In } k = \langle f, g \rangle \Leftrightarrow \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases} \quad \text{let } k = id$$

# Equations!

## Calculation

$$\text{In } k = \langle f, g \rangle \Leftrightarrow \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases} \quad \text{let } k = id$$

$$id = \langle f, g \rangle \Leftrightarrow \begin{cases} \pi_1 \cdot id = f \\ \pi_2 \cdot id = g \end{cases}$$

# Equations!

## Calculation

$$\text{In } k = \langle f, g \rangle \Leftrightarrow \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases} \quad \text{let } k = id$$

$$id = \langle f, g \rangle \Leftrightarrow \begin{cases} \pi_1 = f \\ \pi_2 = g \end{cases}$$

# Equations!

$$id = \langle f, g \rangle \Leftrightarrow \begin{cases} \pi_1 = f \\ \pi_2 = g \end{cases}$$

# Equations!

$$id = \langle f, g \rangle \Leftrightarrow \begin{cases} \pi_1 = f \\ \pi_2 = g \end{cases}$$

Substituting:

$$id = \langle \pi_1, \pi_2 \rangle$$

# Equations!

$$id = \langle f, g \rangle \Leftrightarrow \begin{cases} \pi_1 = f \\ \pi_2 = g \end{cases}$$

Substituting:

$$id = \langle \pi_1, \pi_2 \rangle$$

x-Reflexion



# Problem

*Solve the equation*

$$\langle h, k \rangle = \langle f, g \rangle$$



# Problem

*Solve the equation*

$$\langle h, k \rangle = \langle f, g \rangle$$

*(1 equation, 4 unknowns)*

# Calculation

$$\langle h, k \rangle = \langle f, g \rangle$$

# Calculation

$$\langle h, k \rangle = \langle f, g \rangle$$

$$\Leftrightarrow \{ \text{x-universal} \}$$

$$\begin{cases} \pi_1 \cdot \langle h, k \rangle = f \\ \pi_2 \cdot \langle h, k \rangle = g \end{cases}$$

# Calculation

$$\langle h, k \rangle = \langle f, g \rangle$$

$$\Leftrightarrow \{ \text{x-universal} \}$$

$$\begin{cases} \pi_1 \cdot \langle h, k \rangle = f \\ \pi_2 \cdot \langle h, k \rangle = g \end{cases}$$

$$\Leftrightarrow \{ \text{x-cancellation} \}$$


$$\begin{cases} h = f \\ k = g \end{cases}$$

# Calculation

$$\langle h, k \rangle = \langle f, g \rangle$$

$$\Leftrightarrow \{ \text{x-universal} \}$$

$$\begin{cases} \pi_1 \cdot \langle h, k \rangle = f \\ \pi_2 \cdot \langle h, k \rangle = g \end{cases}$$

x-Eq ! 

$$\Leftrightarrow \{ \text{x-cancellation} \}$$

$$\begin{cases} h = f \\ k = g \end{cases}$$

$$\langle \pi_1, \pi_2 \rangle = id$$

$$\langle \pi_1, \pi_2 \rangle = id$$

$$\langle \pi_2, \pi_1 \rangle ?$$

$$\langle \pi_1, \pi_2 \rangle = id$$

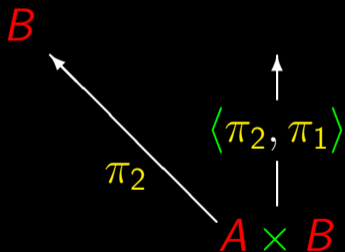
$$\langle \pi_2, \pi_1 \rangle ?$$

$$\begin{array}{c} \uparrow \\ \langle \pi_2, \pi_1 \rangle \\ \downarrow \end{array}$$



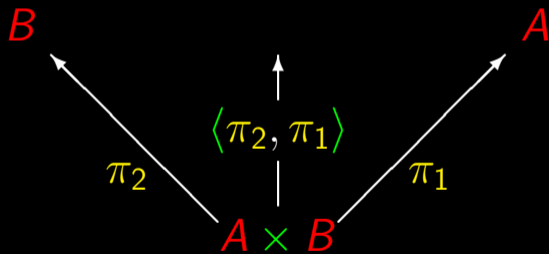
$$\langle \pi_1, \pi_2 \rangle = id$$

$$\langle \pi_2, \pi_1 \rangle ?$$



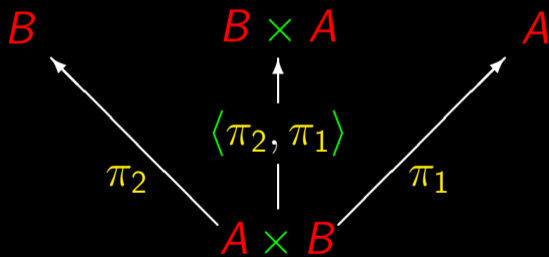
$$\langle \pi_1, \pi_2 \rangle = id$$

$$\langle \pi_2, \pi_1 \rangle ?$$



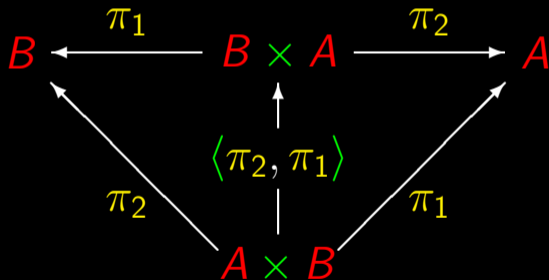
$$\langle \pi_1, \pi_2 \rangle = id$$

$$\langle \pi_2, \pi_1 \rangle ?$$



$$\langle \pi_1, \pi_2 \rangle = id$$

$$\langle \pi_2, \pi_1 \rangle ?$$



# Problem

Solve

$$\langle \pi_2, \pi_1 \rangle \cdot k = id$$

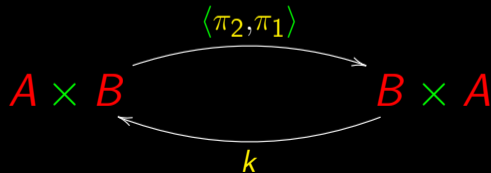
for  $k$

# Problem

Solve

$$\langle \pi_2, \pi_1 \rangle \cdot k = id$$

for  $k$



# Calculation

$$\langle \pi_2, \pi_1 \rangle \cdot k = id$$

# Calculation

$$\langle \pi_2, \pi_1 \rangle \cdot k = id$$

$$\Leftrightarrow \left\{ \begin{array}{c} \times\text{-fusion} \end{array} \right\}$$

$$\langle \pi_2 \cdot k, \pi_1 \cdot k \rangle = id$$



# Calculation

$$\langle \pi_2, \pi_1 \rangle \cdot k = id$$

$$\Leftrightarrow \left\{ \begin{array}{l} \text{x-fusion} \end{array} \right\}$$

$$\langle \pi_2 \cdot k, \pi_1 \cdot k \rangle = id$$

$$\Leftrightarrow \left\{ \begin{array}{l} \text{x-universal} \end{array} \right\}$$

$$\left\{ \begin{array}{l} \pi_2 \cdot k = \pi_1 \\ \pi_1 \cdot k = \pi_2 \end{array} \right.$$

# Calculation

$$\langle \pi_2, \pi_1 \rangle \cdot k = id$$

 $\Leftrightarrow$ 

$$\{ \text{x-fusion} \}$$

 $\Leftrightarrow$ 

$$\{ \text{trivial} \}$$

$$\langle \pi_2 \cdot k, \pi_1 \cdot k \rangle = id$$

 $\Leftrightarrow$ 

$$\{ \text{x-universal} \}$$

$$\begin{cases} \pi_1 \cdot k = \pi_2 \\ \pi_2 \cdot k = \pi_1 \end{cases}$$

$$\begin{cases} \pi_2 \cdot k = \pi_1 \\ \pi_1 \cdot k = \pi_2 \end{cases}$$

# Calculation

$$\langle \pi_2, \pi_1 \rangle \cdot k = id$$

$$\Leftrightarrow \left\{ \begin{array}{l} \times\text{-fusion} \end{array} \right\}$$

$$\langle \pi_2 \cdot k, \pi_1 \cdot k \rangle = id$$

$$\Leftrightarrow \left\{ \begin{array}{l} \times\text{-universal} \end{array} \right\}$$

$$\left\{ \begin{array}{l} \pi_2 \cdot k = \pi_1 \\ \pi_1 \cdot k = \pi_2 \end{array} \right.$$

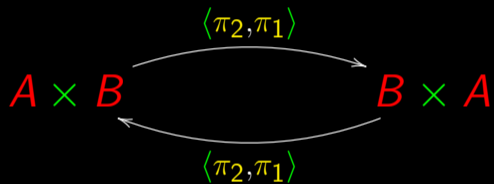
$$\Leftrightarrow \left\{ \begin{array}{l} \text{trivial} \end{array} \right\}$$

$$\left\{ \begin{array}{l} \pi_1 \cdot k = \pi_2 \\ \pi_2 \cdot k = \pi_1 \end{array} \right.$$

$$\Leftrightarrow \left\{ \begin{array}{l} \times\text{-universal} \end{array} \right\}$$

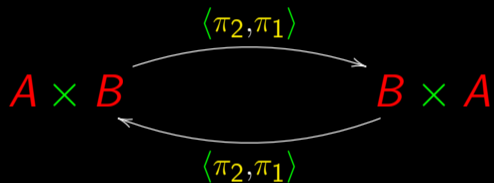
$$k = \langle \pi_2, \pi_1 \rangle$$

# Swap



$$\text{swap} = \langle \pi_2, \pi_1 \rangle$$

# Swap



$$\text{swap} = \langle \pi_2, \pi_1 \rangle$$

$$\text{swap} \cdot \text{swap} = \text{id}$$

Até agora

$f \cdot g$

Sequential composition

Até agora

$f \cdot g$

$\langle f, g \rangle$

Sequential composition

Parallel composition

## Até agora

$f \cdot g$

Sequential composition

$\langle f, g \rangle$

Parallel composition

Associativity

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



## Até agora

$f \cdot g$

Sequential composition

$\langle f, g \rangle$

Parallel composition

Associativity

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

Associativity?

$$\langle \langle f, g \rangle, h \rangle = \langle f, \langle g, h \rangle \rangle?$$

## Até agora

$f \cdot g$

Sequential composition

$\langle f, g \rangle$

Parallel composition

Associativity

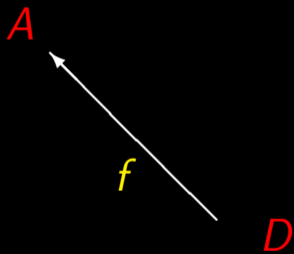
$$(f \cdot g) \cdot h = f \cdot (g \cdot h)$$

Associativity?

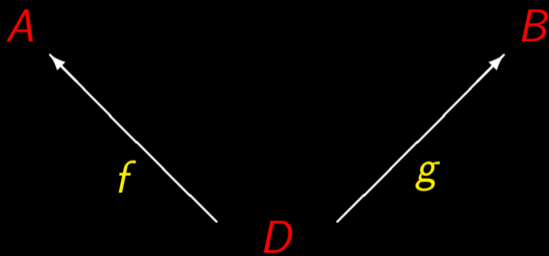
$$\langle \langle f, g \rangle, h \rangle = \langle f, \langle g, h \rangle \rangle?$$

No! but...

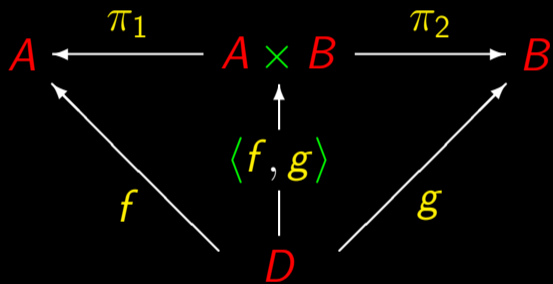
$\langle\langle f, g \rangle, h\rangle$



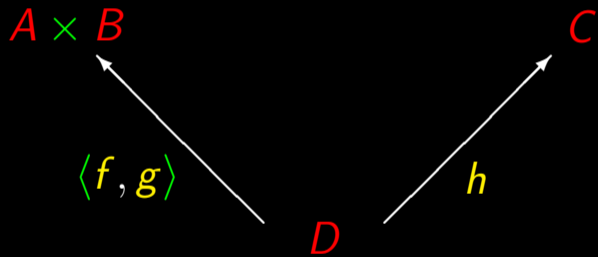
$\langle\langle f, g \rangle, h\rangle$



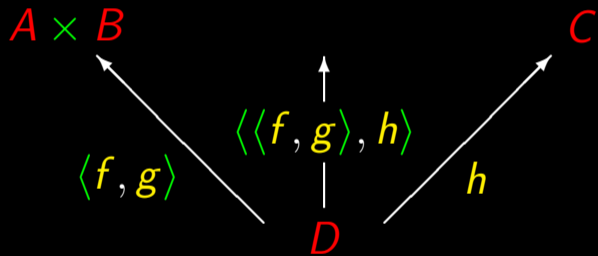
$\langle \langle f, g \rangle, h \rangle$



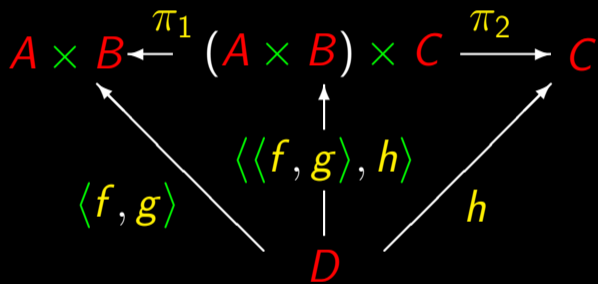
$\langle \langle f, g \rangle, h \rangle$



$\langle \langle f, g \rangle, h \rangle$

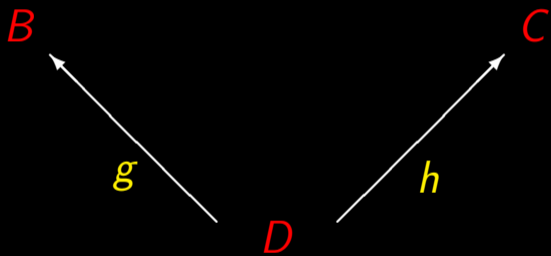


$\langle \langle f, g \rangle, h \rangle$

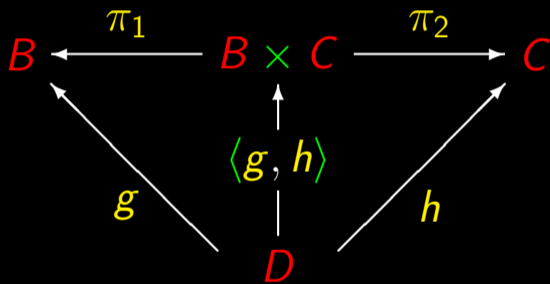




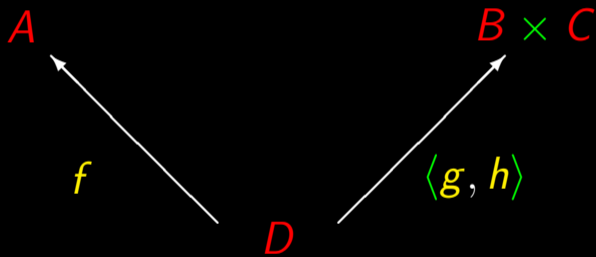
$\langle f, \langle g, h \rangle \rangle$



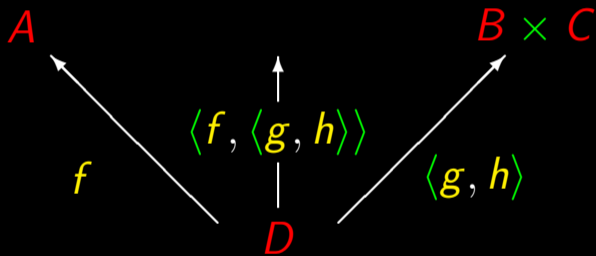
$\langle f, \langle g, h \rangle \rangle$



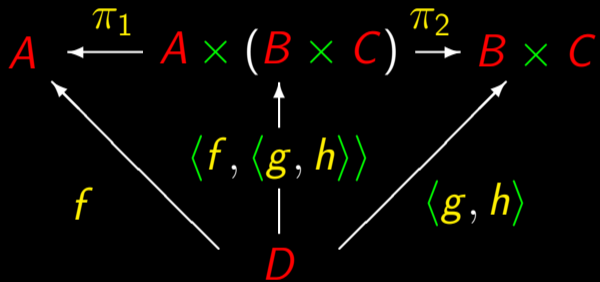
$\langle f, \langle g, h \rangle \rangle$



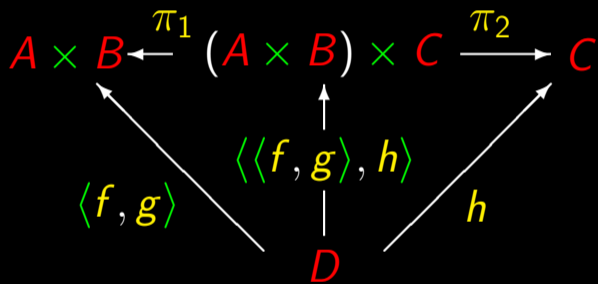
$\langle f, \langle g, h \rangle \rangle$

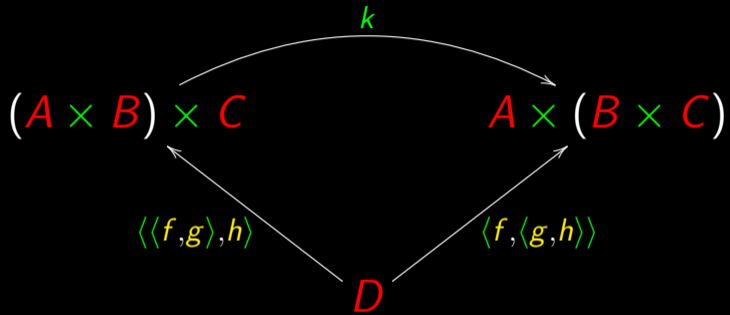


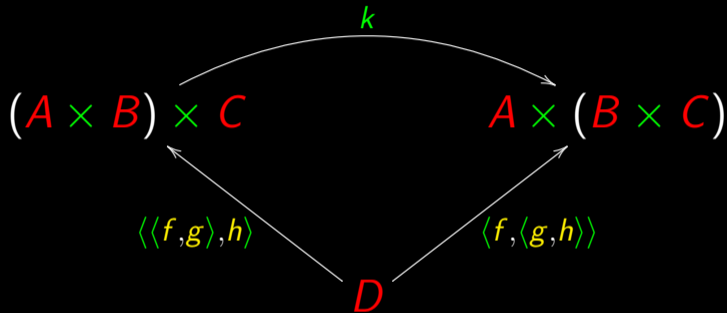
$\langle f, \langle g, h \rangle \rangle$



$\langle \langle f, g \rangle, h \rangle$







$$k \cdot \langle \langle f, g \rangle, h \rangle = \langle f, \langle g, h \rangle \rangle$$



$$k \cdot \langle \langle f, g \rangle, h \rangle = \langle f, \langle g, h \rangle \rangle$$

$$k \cdot \langle \langle f, g \rangle, h \rangle = \langle f, \langle g, h \rangle \rangle$$

$$k \cdot \underbrace{\langle \langle f, g \rangle, h \rangle}_{id?} = \langle f, \langle g, h \rangle \rangle$$

$$k \cdot \langle \langle f, g \rangle, h \rangle = \langle f, \langle g, h \rangle \rangle$$

$$k \cdot \underbrace{\langle \langle f, g \rangle, h \rangle}_{id?} = \langle f, \langle g, h \rangle \rangle$$



Solve  $\langle \langle f, g \rangle, h \rangle = id$

$$\langle \langle f, g \rangle, h \rangle = id$$

$$\langle \langle f, g \rangle, h \rangle = id$$

$$\Leftrightarrow \{ \text{x-universal} \}$$

$$\begin{cases} \pi_1 = \langle f, g \rangle \\ \pi_2 = h \end{cases}$$

$$\langle \langle f, g \rangle, h \rangle = id$$

$$\Leftrightarrow \{ \text{x-universal} \}$$

$$\begin{cases} \pi_1 = \langle f, g \rangle \\ \pi_2 = h \end{cases}$$

$$\Leftrightarrow \{ \text{x-universal} \}$$

$$\begin{cases} \pi_1 \cdot \pi_1 = f \\ \pi_2 \cdot \pi_1 = g \\ \pi_2 = h \end{cases}$$

## Substitute solutions

$$\begin{cases} \pi_1 \cdot \pi_1 = f \\ \pi_2 \cdot \pi_1 = g \\ \pi_2 = h \end{cases}$$

# Substitute solutions

$$\begin{cases} \pi_1 \cdot \pi_1 = f \\ \pi_2 \cdot \pi_1 = g \\ \pi_2 = h \end{cases}$$

$$k \cdot \underbrace{\langle \langle f, g \rangle, h \rangle}_{id} = \langle f, \langle g, h \rangle \rangle$$

*id* 😊



## Substitute solutions

$$\begin{cases} \pi_1 \cdot \pi_1 = f \\ \pi_2 \cdot \pi_1 = g \\ \pi_2 = h \end{cases}$$

$$k = \langle \pi_1 \cdot \pi_1, \langle \pi_2 \cdot \pi_1, \pi_2 \rangle \rangle$$

Slight improvement...

$$k = \langle \pi_1 \cdot \pi_1, \langle \pi_2 \cdot \pi_1, \pi_2 \rangle \rangle$$

## Slight improvement...

$$\begin{aligned} k &= \langle \pi_1 \cdot \pi_1, \langle \pi_2 \cdot \pi_1, \pi_2 \rangle \rangle \\ \Leftrightarrow & \quad \left\{ \pi_2 = id \cdot \pi_2 \right\} \\ k &= \langle \pi_1 \cdot \pi_1, \langle \pi_2 \cdot \pi_1, id \cdot \pi_2 \rangle \rangle \end{aligned}$$

## Slight improvement...

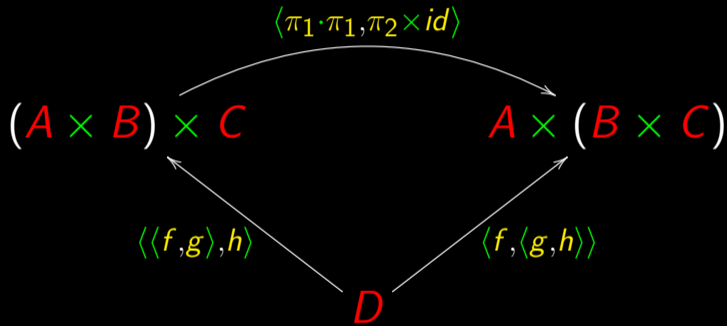
$$k = \langle \pi_1 \cdot \pi_1, \langle \pi_2 \cdot \pi_1, \pi_2 \rangle \rangle$$

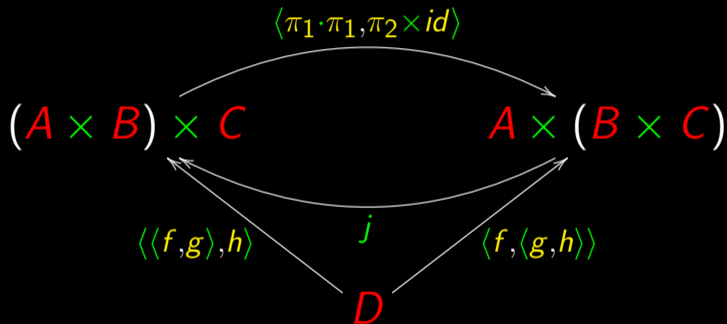
$$\Leftrightarrow \left\{ \pi_2 = id \cdot \pi_2 \right\}$$

$$k = \langle \pi_1 \cdot \pi_1, \langle \pi_2 \cdot \pi_1, id \cdot \pi_2 \rangle \rangle$$

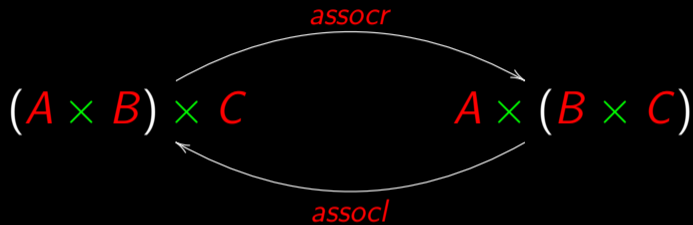
$$\Leftrightarrow \left\{ f \times g = \langle f \cdot \pi_1, g \cdot \pi_2 \rangle \right\}$$

$$k = \langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle$$





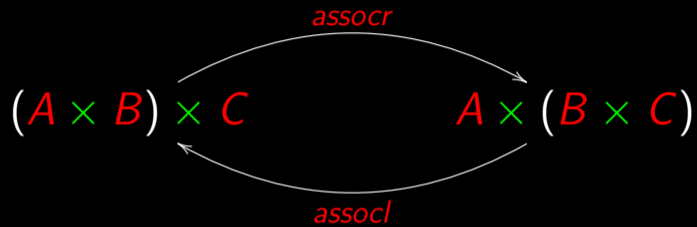
$$\begin{array}{ccc} & \langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle & \\ & \curvearrowright & \\ (A \times B) \times C & & A \times (B \times C) \\ & \curvearrowleft & \\ & \langle id \times \pi_1, \pi_2 \cdot \pi_2 \rangle & \end{array}$$



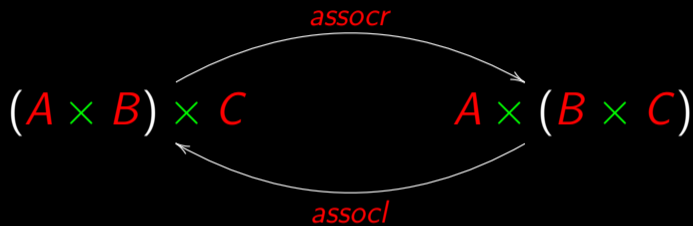
$$assocr = \langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle$$

$$assocl = \langle id \times \pi_1, \pi_2 \cdot \pi_2 \rangle$$





$$\text{assocr} \cdot \text{assocl} = \text{id}$$



$$assocr \cdot assocl = id$$

$$assocl \cdot assocr = id$$

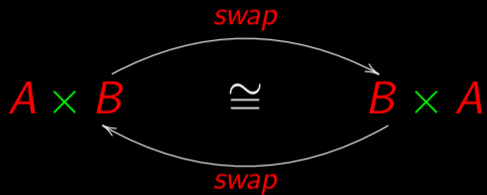
# Isomorphism

$$\begin{array}{ccc} & \xrightarrow{\text{assocr}} & \\ (A \times B) \times C & \cong & A \times (B \times C) \\ & \xleftarrow{\text{assocl}} & \end{array}$$

$$\text{assocr} \cdot \text{assocl} = \text{id}$$

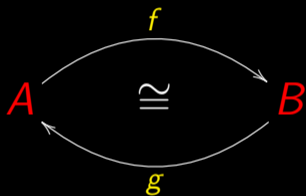
$$\text{assocl} \cdot \text{assocr} = \text{id}$$

# Isomorphism



$$\text{swap} \cdot \text{swap} = \text{id}$$

# Isomorphism



$$f \cdot g = id$$

$$g \cdot f = id$$

# Isomorphism

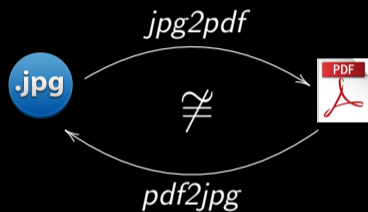
*iso* (ISO)  
the same

# Isomorphism

*iso* (*ισο*) + *morphism* (*μορφισμοζ*)  
*the same* *shape*

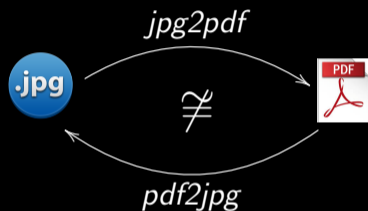
“Similar shape”

# Practical problem!





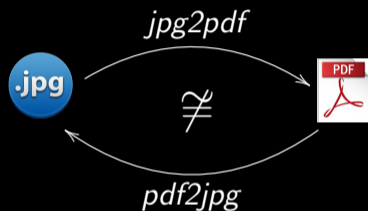
# Practical problem!



$$jpg2pdf \cdot pdf2jpg \neq id$$

$$pdf2jpg \cdot jpg2pdf \neq id$$

# Practical problem!

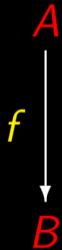


$jpg2pdf \cdot pdf2jpg \neq id$

$pdf2jpg \cdot jpg2pdf \neq id$

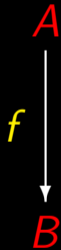
# Format conversion

Need



# Format conversion

Need



Reusable



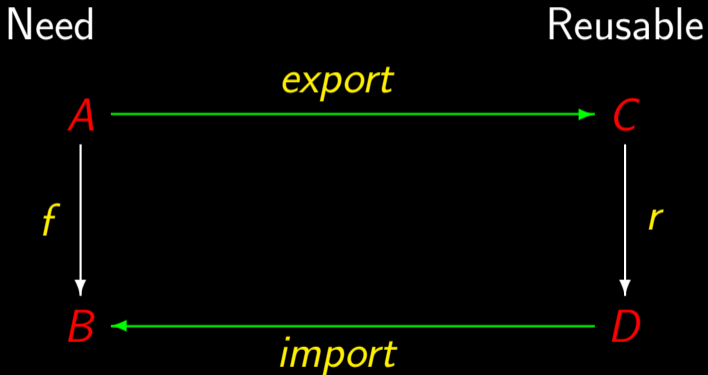
# Format conversion

Need

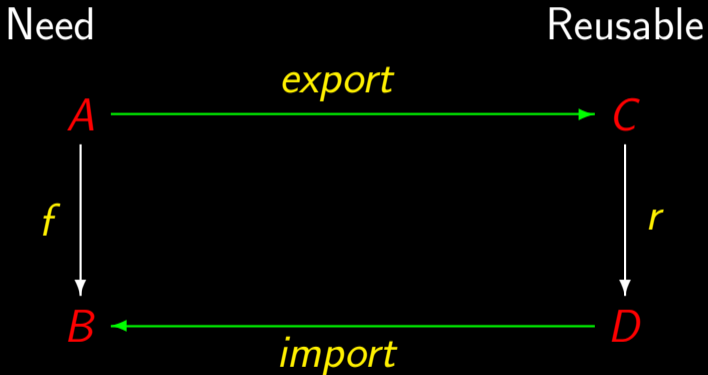
Reusable



# Format conversion

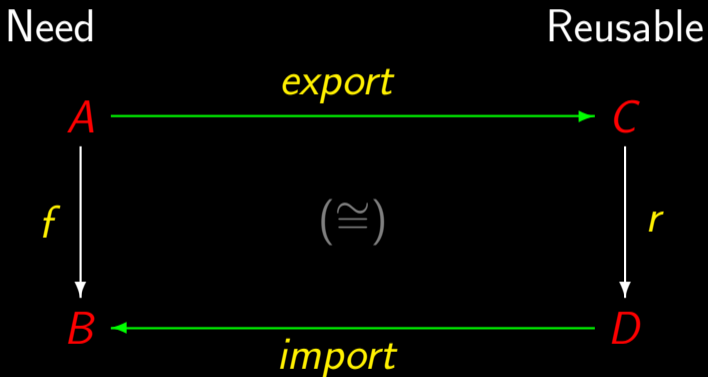


# Format conversion



$$f = \text{import} \cdot r \cdot \text{export}$$

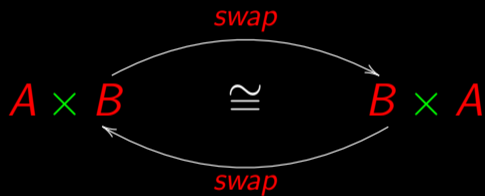
# Format conversion



$$f = \text{import} \cdot r \cdot \text{export}$$

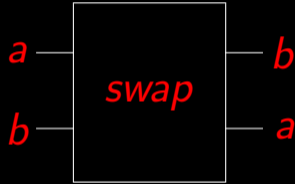


By the way — *swap*

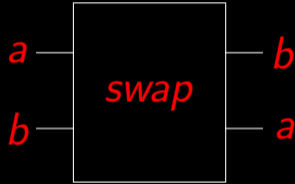


Isomorphisms are **reversible** computations

By the way — *swap*

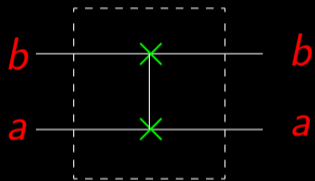


By the way — *swap*



*swap* is a basic gate in **quantum programming**.

By the way — *swap*



# Cálculo de Programas

Class T04: 13-Mar

## Problem

*Retrieve the address of a civil servant, knowing that she/he can be identified either by a citizen card number (**CC**) or a fiscal number (**NIF**).*

# Problem

*Retrieve the address of a civil servant, knowing that she/he can be identified either by a citizen card number (CC) or a fiscal number (NIF).*

*address : Iden  $\rightarrow$  Address*

*Iden = CC  $\cup$  NIF*

# Problem!

$$CC = N$$

$$NIF = N$$



# Problem!

$$CC = \mathbb{N}$$

$$NIF = \mathbb{N}$$

$$Iden = CC \cup NIF = \mathbb{N} \cup \mathbb{N} = \mathbb{N}$$

# Problem!

$$CC = \mathbb{N}$$

$$NIF = \mathbb{N}$$

$$Iden = CC \cup NIF = \mathbb{N} \cup \mathbb{N} = \mathbb{N}$$

$$address : \mathbb{N} \rightarrow Address (!)$$

In general

We need to fix:

$$m : A \cup B \rightarrow C$$

In general

We need to fix:

$$m : A \cup B \rightarrow C$$

Let us start from

$$A \cup B = \{a \mid a \in A\} \cup \{b \mid b \in B\}$$

## Disjoint union

In need of something like...

$$\{(1, a) \mid a \in A\} \cup \{(2, b) \mid b \in B\}$$

## Disjoint union

In need of something like...

$$\{(1, a) \mid a \in A\} \cup \{(2, b) \mid b \in B\}$$

... we define:

$$A + B = \{(1, a) \mid a \in A\} \cup \{(2, b) \mid b \in B\}$$

## Disjoint union

Clearly,

$$A + B = \{i_1 a \mid a \in A\} \cup \{i_2 b \mid b \in B\}$$

once we define:

$$i_1 a = (1, a)$$

$$i_2 b = (2, b)$$

# Disjoint union

Types:

$$i_1 : A \rightarrow A + B$$

$$i_2 : B \rightarrow A + B$$



## Disjoint union

Types:

$$i_1 : A \rightarrow A + B$$

$$i_2 : B \rightarrow A + B$$

Now think of:

$$m : A + B \rightarrow C$$

# Disjoint union

Types:

$$i_1 : A \rightarrow A + B$$

$$i_2 : B \rightarrow A + B$$

Now think of:

$$m : A + B \rightarrow C$$



# Disjoint union

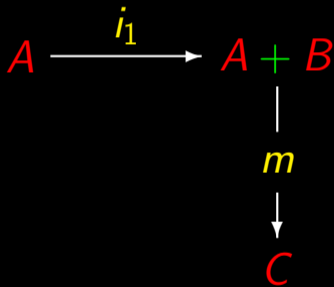
Types:

$$i_1 : A \rightarrow A + B$$

$$i_2 : B \rightarrow A + B$$

Now think of:

$$m : A + B \rightarrow C$$



# Disjoint union

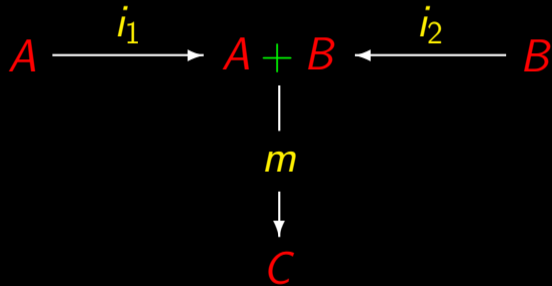
Types:

$$i_1 : A \rightarrow A + B$$

$$i_2 : B \rightarrow A + B$$

Now think of:

$$m : A + B \rightarrow C$$



# Disjoint union

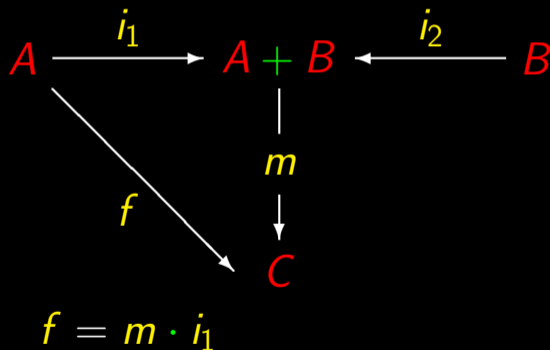
Types:

$$i_1 : A \rightarrow A + B$$

$$i_2 : B \rightarrow A + B$$

Now think of:

$$m : A + B \rightarrow C$$



# Disjoint union

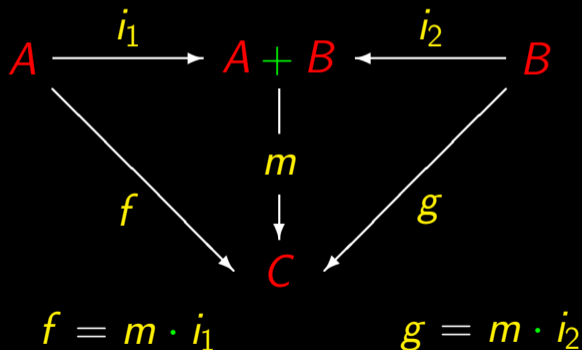
Types:

$$i_1 : A \rightarrow A + B$$

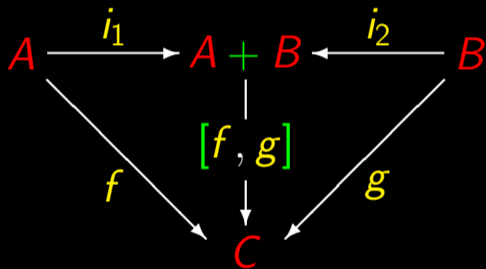
$$i_2 : B \rightarrow A + B$$

Now think of:

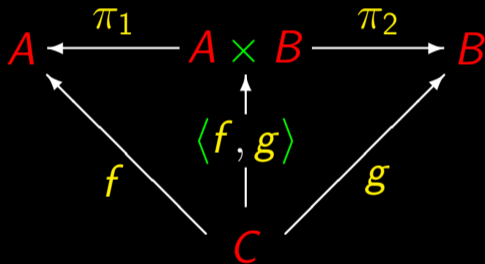
$$m : A + B \rightarrow C$$



Compare...



Compare...





## + - Universal

$$k = [f, g] \Leftrightarrow \begin{cases} k \cdot i_1 = f \\ k \cdot i_2 = g \end{cases}$$

## + - Universal

$$k = [f, g] \Leftrightarrow \begin{cases} k \cdot i_1 = f \\ k \cdot i_2 = g \end{cases}$$

Compare with

$$k = \langle f, g \rangle \Leftrightarrow \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases}$$

# “Alternating” functions

$$[f, g] : A + B \rightarrow C$$

$$[f, g] x = \begin{cases} x = i_1 a \Rightarrow f a \\ x = i_2 b \Rightarrow g b \end{cases}$$

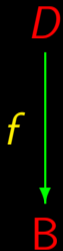
# “Alternating” functions

$$[f, g] : A + B \rightarrow C$$

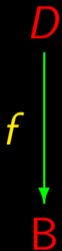
$$[f, g] x = \begin{cases} x = i_1 a \Rightarrow f a \\ x = i_2 b \Rightarrow g b \end{cases}$$

$$f + g \quad ?$$

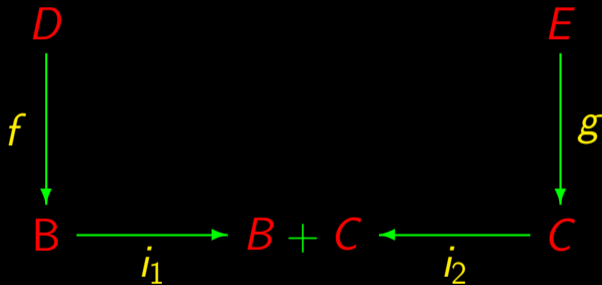
# Sum of two functions



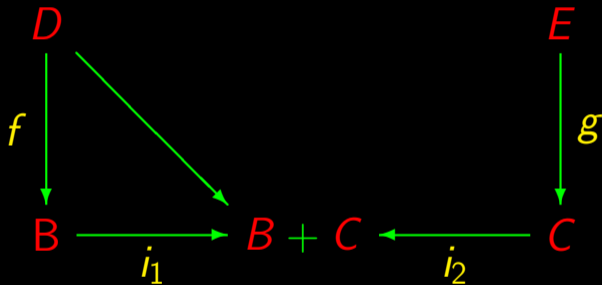
# Sum of two functions



# Sum of two functions

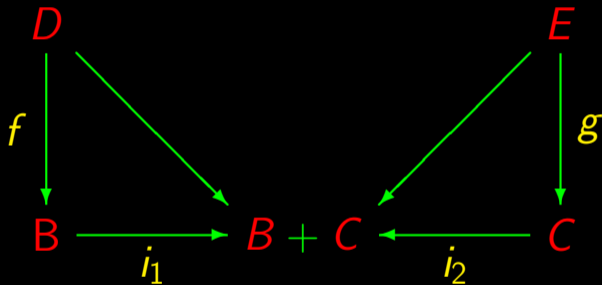


# Sum of two functions

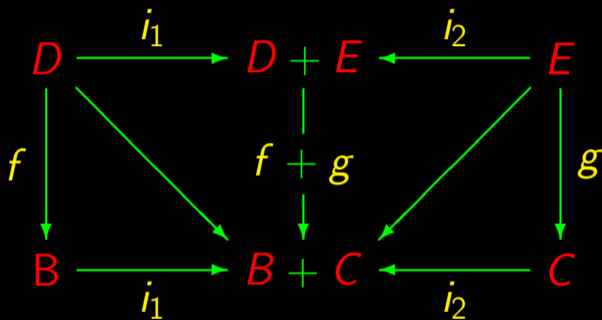




# Sum of two functions



# Sum of two functions



$$f + g = [i_1 \cdot f, i_2 \cdot g]$$

## Coproduct laws

“Just reverse the arrows”, cf.

$$\text{+-Absorption} \quad [h, k] \cdot (f + g) = [h \cdot f, k \cdot g] \quad (2.43)$$

## Coproduct laws

“Just reverse the arrows”, cf.

$$\text{+-Absorption} \quad [h, k] \cdot (f + g) = [h \cdot f, k \cdot g] \quad (2.43)$$

$$\text{+-Fusion} \quad f \cdot [h, k] = [f \cdot h, f \cdot k] \quad (2.42)$$

## Coproduct laws

“Just reverse the arrows”, cf.

$$\text{+-Absorption} \quad [h, k] \cdot (f + g) = [h \cdot f, k \cdot g] \quad (2.43)$$

$$\text{+-Fusion} \quad f \cdot [h, k] = [f \cdot h, f \cdot k] \quad (2.42)$$

$$\text{+-Reflexion} \quad [i_1, i_2] = id \quad (2.41)$$

## Coproduct laws

+Equality  $[h, k] = [f, g] \Leftrightarrow \begin{cases} h = f \\ k = g \end{cases} \quad (2.66)$

## Coproduct laws

**+ - Equality**       $[h, k] = [f, g] \Leftrightarrow \begin{cases} h = f \\ k = g \end{cases} \quad (2.66)$

**+ - Functor**       $(h + k) \cdot (f + g) = h \cdot f + k \cdot g \quad (2.44)$

## Coproduct laws

$$\text{+-Equality} \quad [h, k] = [f, g] \Leftrightarrow \begin{cases} h = f \\ k = g \end{cases} \quad (2.66)$$

$$\text{+-Functor} \quad (h + k) \cdot (f + g) = h \cdot f + k \cdot g \quad (2.44)$$

$$\text{+-Functor-id} \quad id + id = id \quad (2.45)$$

and so on



All these laws can be found in the **reference sheet**

(WWW → **Material**)

## Summary

$f \cdot g$

$\langle f, g \rangle$

$f \times g$

Sequential composition

Parallel composition

Product composition

## Summary

$f \cdot g$

$\langle f, g \rangle$

$f \times g$

$[f, g]$

Sequential composition

Parallel composition

Product composition

Alternative composition

## Summary

$f \cdot g$

Sequential composition

$\langle f, g \rangle$

Parallel composition

$f \times g$

Product composition

$[f, g]$

Alternative composition

$f + g$

Coproduct (function sum)

## Summary

$f \cdot g$

Sequential composition

$\langle f, g \rangle$

Parallel composition

$f \times g$

Product composition

$[f, g]$

Alternative composition

$f + g$

Coproduct (function sum)

**Compositional programming**



What about...

$$f \times (g + h) \stackrel{?}{=} f \times g + f \times h$$

What about...

$$f \times (g + h) \stackrel{?}{=} f \times g + f \times h$$

...?

$$A \times (B + C) \stackrel{?}{=} A \times B + A \times C$$

What about...

$$f \times (g + h) \stackrel{?}{=} f \times g + f \times h$$

...?

$$A \times (B + C) \stackrel{?}{\cong} A \times B + A \times C$$

Moreover, any “0” such that



What about...

$$f \times (g + h) \stackrel{?}{=} f \times g + f \times h$$

...?

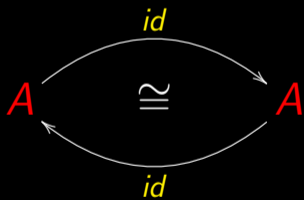
$$A \times (B + C) \stackrel{?}{\cong} A \times B + A \times C$$

Moreover, any “0” such that

$$A \times 0 = 0 \quad ? \quad A + 0 = A \quad ??$$

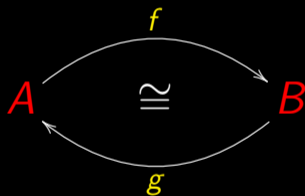
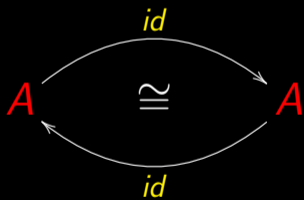
# Recall

$$\begin{cases} id : A \rightarrow A \\ id \ a = a \end{cases}$$



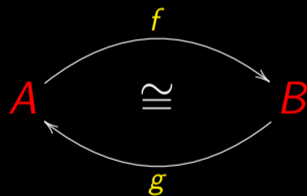
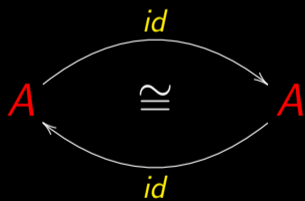
# Recall

$$\begin{cases} id : A \rightarrow A \\ id \ a = a \end{cases}$$



# Recall

$$\begin{cases} id : A \rightarrow A \\ id \ a = a \end{cases}$$



$$f \cdot g = id$$

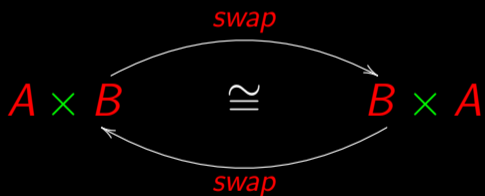
$$g \cdot f = id$$

## Recall

$$\begin{cases} \text{swap} : A \times B \rightarrow B \times A \\ \text{swap} (a, b) = (b, a) \end{cases}$$

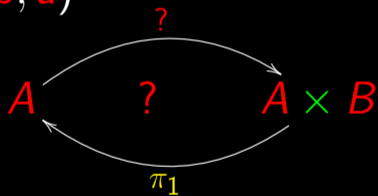
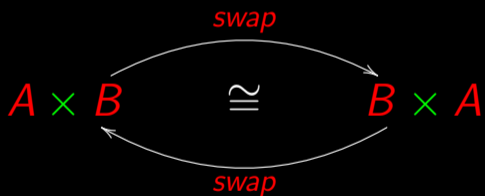
# Recall

$$\begin{cases} \text{swap} : A \times B \rightarrow B \times A \\ \text{swap} (a, b) = (b, a) \end{cases}$$



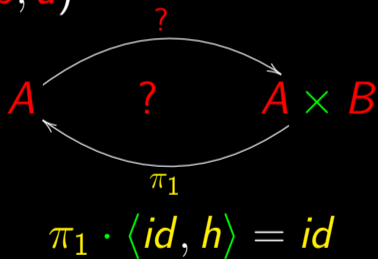
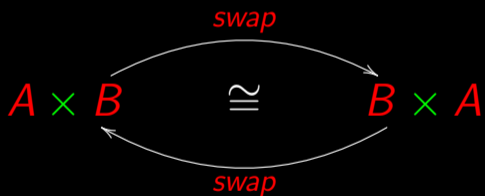
# Recall

$$\begin{cases} \text{swap} : A \times B \rightarrow B \times A \\ \text{swap}(a, b) = (b, a) \end{cases}$$



# Recall

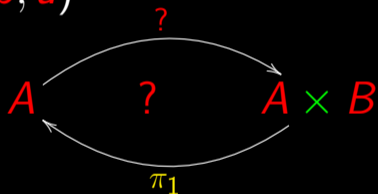
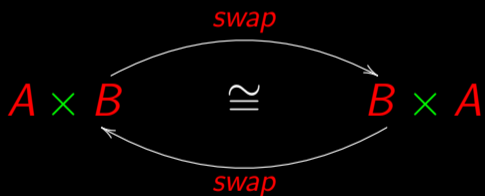
$$\begin{cases} \text{swap} : A \times B \rightarrow B \times A \\ \text{swap}(a, b) = (b, a) \end{cases}$$





# Recall

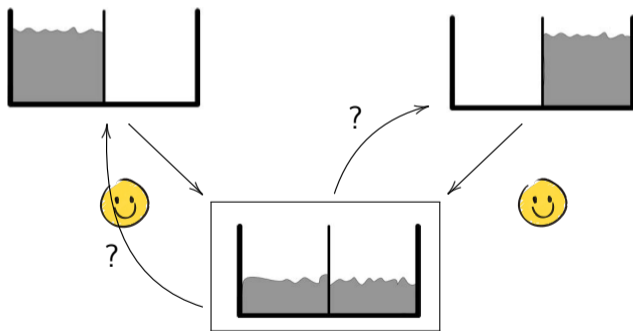
$$\begin{cases} \text{swap} : A \times B \rightarrow B \times A \\ \text{swap}(a, b) = (b, a) \end{cases}$$



$$\pi_1 \cdot \langle id, h \rangle = id$$

$$\langle id, h \rangle \cdot \pi_1 \neq id$$

# Irreversibility...



Rolf Landauer (1927–99)

**“Information is  
physical”**



Even worse than  $\pi_1 \dots$

$$\begin{cases} \text{zero } x = 0 \\ \text{one } x = 1 \end{cases}$$

Even worse than  $\pi_1 \dots$

$$\begin{cases} \text{zero } x = 0 \\ \text{one } x = 1 \end{cases}$$

$$\text{one } 10 = 1$$

Even worse than  $\pi_1\dots$

$$\begin{cases} \text{zero } x = 0 \\ \text{one } x = 1 \end{cases}$$

*one* 10 = 1

*one* "string" = 1

Even worse than  $\pi_1\dots$

$$\begin{cases} \text{zero } x = 0 \\ \text{one } x = 1 \end{cases}$$

*one* 10 = 1

*one* "string" = 1

*zero* "string" = 0

Even worse than  $\pi_1 \dots$

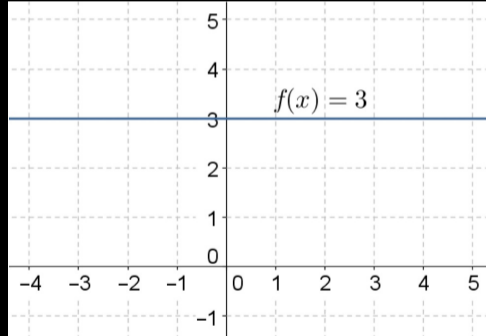
$$\begin{cases} \text{zero } x = 0 \\ \text{one } x = 1 \end{cases}$$

$$\text{one } 10 = 1$$

$$\text{one } \text{"string"} = 1$$

$$\text{zero } \text{"string"} = 0$$

$$f \ x = 3$$





# Constant functions

$$3 \in \mathbb{N}_0$$

$$f(x) = 3$$

# Constant functions

$$3 \in \mathbb{N}_0$$

$$f(x) = 3$$

$$A \xrightarrow{f} \mathbb{N}_0$$

# Constant functions

$$3 \in \mathbb{N}_0$$

$$b_0 \in B$$

$$f(x) = 3$$

$$A \xrightarrow{f} \mathbb{N}_0$$

# Constant functions

$$3 \in \mathbb{N}_0$$

$$b_0 \in B$$

$$f(x) = 3$$

$$A \xrightarrow{f} \mathbb{N}_0$$

$$\left\{ \begin{array}{l} \underline{b_0} : A \rightarrow B \\ \underline{b_0} \quad a = b_0 \end{array} \right.$$

# Constant functions

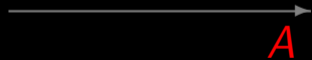
$$3 \in \mathbb{N}_0$$

$$b_0 \in B$$

$$f(x) = 3$$

$$A \xrightarrow{f} \mathbb{N}_0$$

$$\left\{ \begin{array}{l} \underline{b_0} : A \rightarrow B \\ \underline{b_0} \quad a = b_0 \end{array} \right.$$



# Constant functions

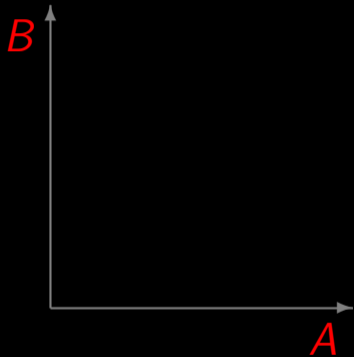
$$3 \in \mathbb{N}_0$$

$$f(x) = 3$$

$$A \xrightarrow{f} \mathbb{N}_0$$

$$b_0 \in B$$

$$\left\{ \begin{array}{l} \underline{b_0} : A \rightarrow B \\ \underline{b_0} \quad a = b_0 \end{array} \right.$$



# Constant functions

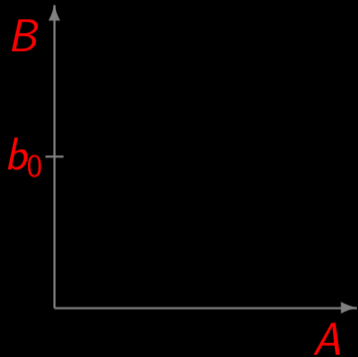
$$3 \in \mathbb{N}_0$$

$$f(x) = 3$$

$$A \xrightarrow{f} \mathbb{N}_0$$

$$b_0 \in B$$

$$\left\{ \begin{array}{l} \underline{b_0} : A \rightarrow B \\ \underline{b_0} \quad a = b_0 \end{array} \right.$$



# Constant functions

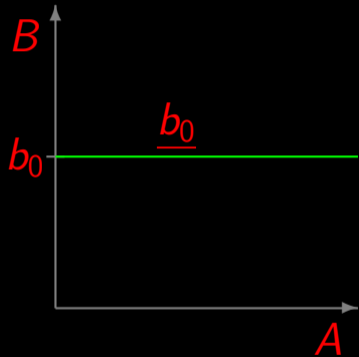
$$3 \in \mathbb{N}_0$$

$$f(x) = 3$$

$$A \xrightarrow{f} \mathbb{N}_0$$

$$b_0 \in B$$

$$\left\{ \begin{array}{l} \underline{b_0} : A \rightarrow B \\ \underline{b_0} \quad a = b_0 \end{array} \right.$$





# Constant functions

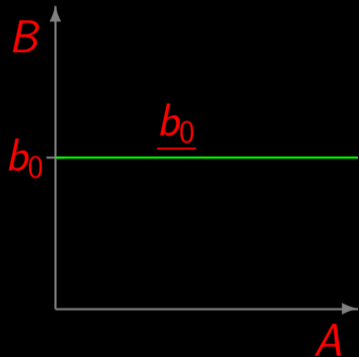
$$3 \in \mathbb{N}_0$$

$$f \ x = 3$$

$$A \xrightarrow{f} \mathbb{N}_0$$

$$b_0 \in B$$

$$\left\{ \begin{array}{l} \underline{b_0} : A \rightarrow B \\ \underline{b_0} \ a = b_0 \end{array} \right.$$



(Haskell: `const  $b_0$` )

# Properties

$$\underline{b} \cdot f = \underline{b}$$

# Properties

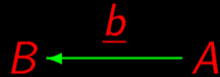
$$\underline{b} \cdot f = \underline{b}$$

$$g \cdot \underline{b} = \underline{g b}$$

# Properties

$$\underline{b} \cdot f = \underline{b}$$

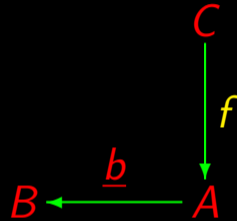
$$g \cdot \underline{b} = \underline{g b}$$



# Properties

$$\underline{b} \cdot f = \underline{b}$$

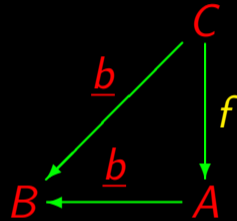
$$g \cdot \underline{b} = \underline{g \ b}$$



# Properties

$$\underline{b} \cdot f = \underline{b}$$

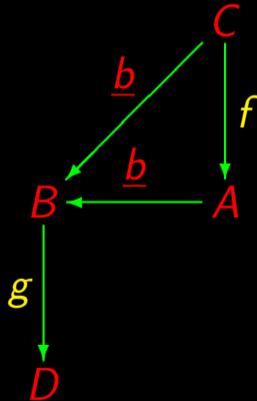
$$g \cdot \underline{b} = \underline{g \ b}$$



# Properties

$$\underline{b} \cdot f = \underline{b}$$

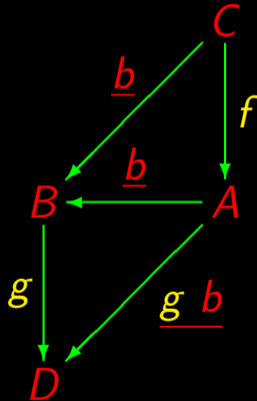
$$g \cdot \underline{b} = \underline{g \ b}$$



# Properties

$$\underline{b} \cdot f = \underline{b}$$

$$g \cdot \underline{b} = \underline{g b}$$

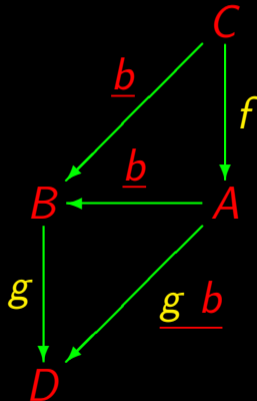




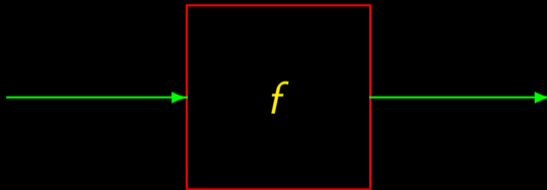
# Properties

$$\underline{b} \cdot f = \underline{b}$$

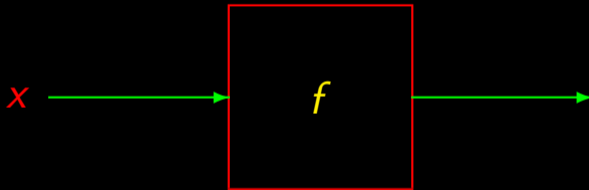
$$g \cdot \underline{b} = \underline{g \ b}$$



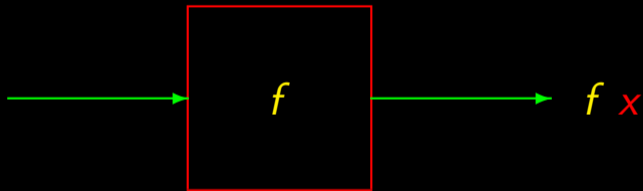
# Data flow



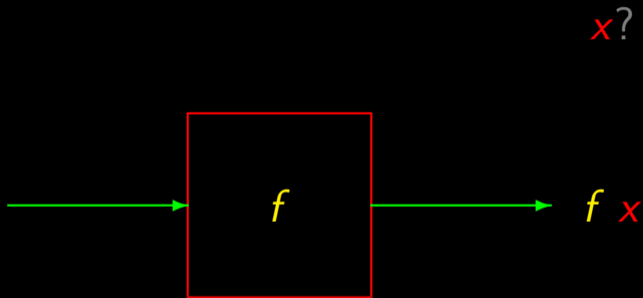
# Data flow



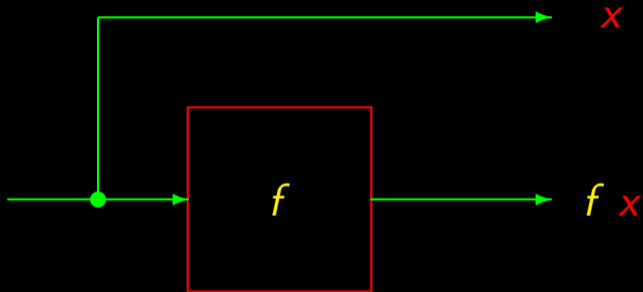
# Data flow



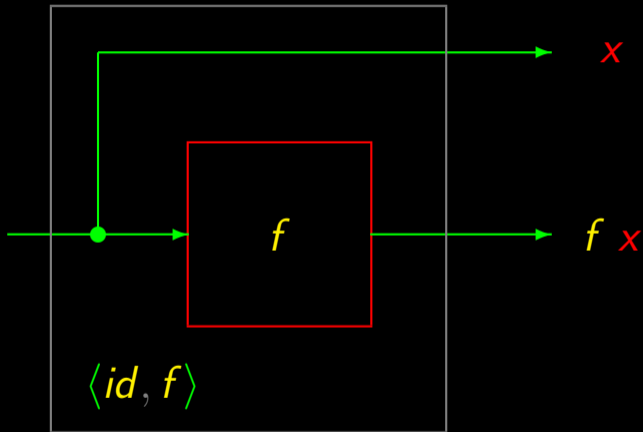
# Data flow



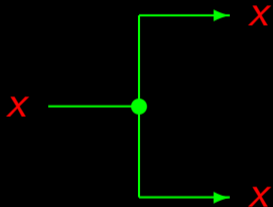
# Data flow



# Data flow

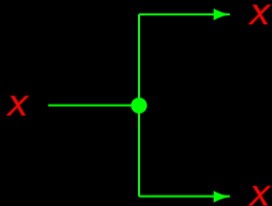


# Data duplication



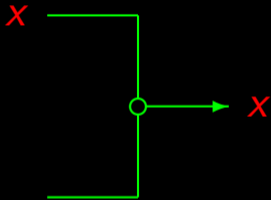


# Data duplication



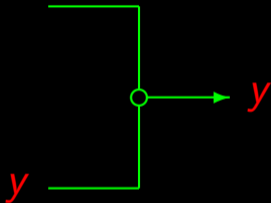
$$dup = \langle id, id \rangle$$

# Data joins



$join = [id, id]$

# Data joins

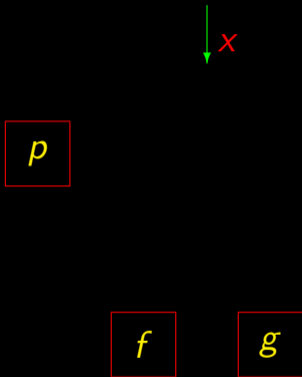


$join = [id, id]$

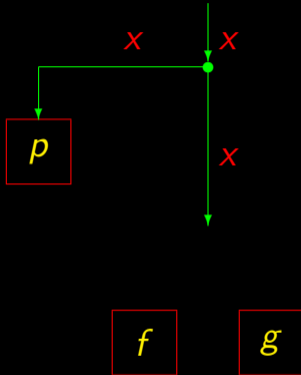
## if-then-else

$y = \text{if } p \ x \ \text{then } f \ x \ \text{else } g \ x$

# if-then-else

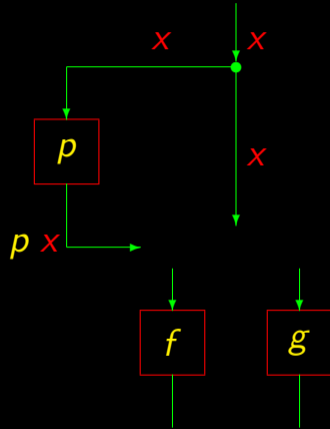


# if-then-else



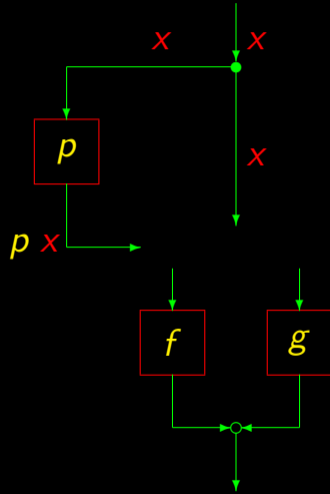


# if-then-else

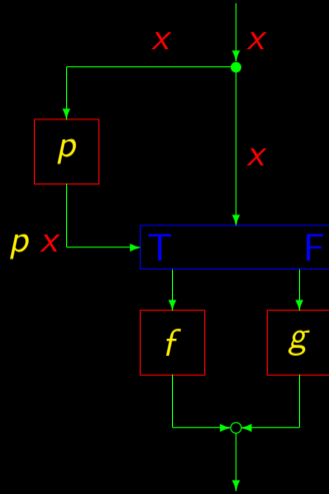




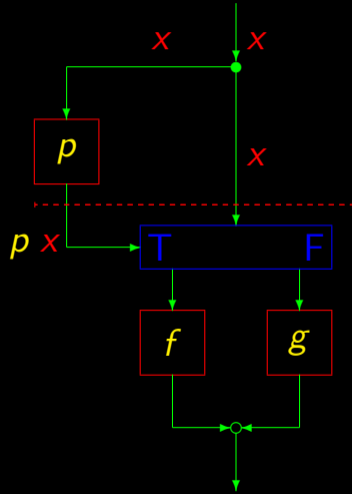
# if-then-else



# if-then-else



# if-then-else



if-then-else (“point-free”)

$x \in A$

if-then-else (“point-free”)

$$x \in A$$

$$(p\ x, x) \in B \times A$$

if-then-else (“point-free”)

$$x \in A$$

$$(p\ x, x) \in B \times A$$

$$B = \{F, T\}$$

## if-then-else (“point-free”)

**B** in Haskell:

$$x \in A$$

```
data Bool = False | True
```

$$(p\ x, x) \in B \times A$$

$$B = \{F, T\}$$

## if-then-else (“point-free”)

$$x \in A$$

$$(p\ x, x) \in B \times A$$

$$B = \{F, T\}$$

**B** in Haskell:

```
data Bool = False | True
```

*T* represented by `True`

*F* represented by `False`

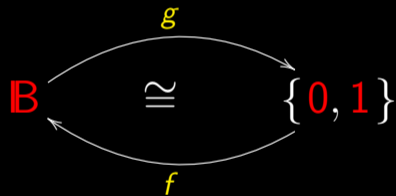


The type 2

$$\mathbb{B} = \{F, T\} \cong \{\text{False}, \text{True}\}$$

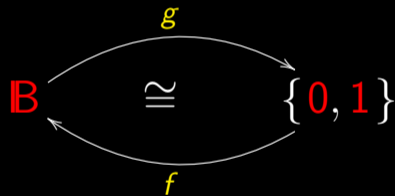
## The type 2

$$\mathbb{B} = \{F, T\} \cong \{\text{False}, \text{True}\} \cong \{0, 1\} \cong \dots \cong 2$$



## The type 2

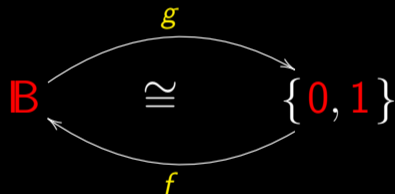
$$\mathbb{B} = \{F, T\} \cong \{\text{False}, \text{True}\} \cong \{0, 1\} \cong \dots \cong 2$$



$$\begin{cases} f\ 0 = F \\ f\ 1 = T \end{cases}$$

## The type 2

$$\mathbb{B} = \{F, T\} \cong \{\text{False}, \text{True}\} \cong \{0, 1\} \cong \dots \cong 2$$



$$\begin{cases} f\ 0 = F \\ f\ 1 = T \end{cases}$$

$$\begin{cases} g\ F = 0 \\ g\ T = 1 \end{cases}$$

$$a + a = 2 a$$

$$A + A \xrightarrow{\text{join}} A$$

$$a + a = 2 a$$

$$A + A \xrightarrow{\text{join}} A$$

$$A \xrightarrow{\langle p, id \rangle} 2 \times A$$

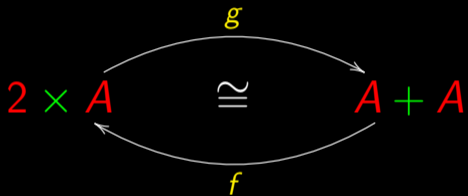
$$a + a = 2 a$$

$$A + A \xrightarrow{\text{join}} A$$

$$A \xrightarrow{\langle p, \text{id} \rangle} 2 \times A$$

Is it the case that...?

$$\begin{array}{ccc} & g & \\ & \curvearrowright & \\ 2 \times A & \cong & A + A \\ & \curvearrowleft & \\ & f & \end{array}$$

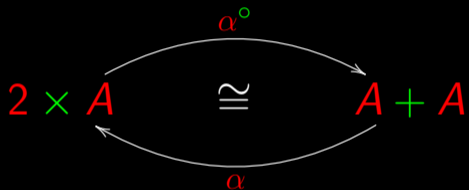


$$f = [\langle \underline{T}, id \rangle, \langle \underline{F}, id \rangle]$$

$$g = \dots (?)$$



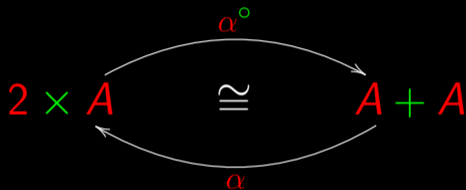
# Notation



$$\alpha = [\langle \underline{T}, id \rangle, \langle \underline{F}, id \rangle]$$

$$\alpha^\circ = \dots$$

# Notation

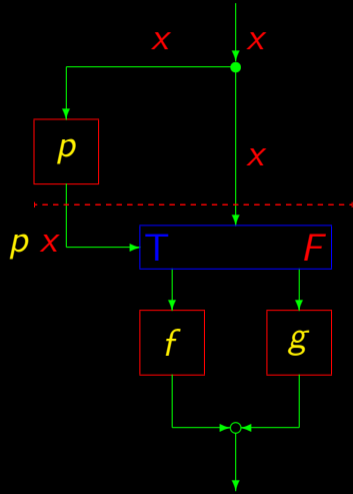


$$\alpha \cdot \alpha^\circ = id$$

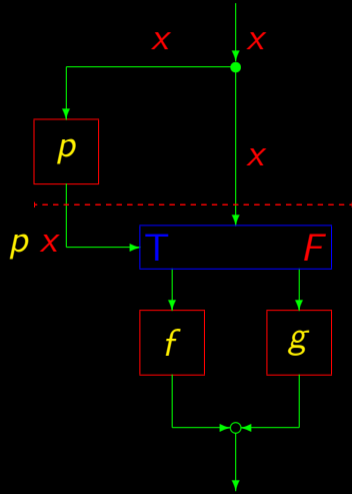
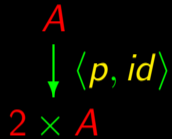
$$\alpha^\circ \cdot \alpha = id$$

$\alpha$  determines  $\alpha^\circ$

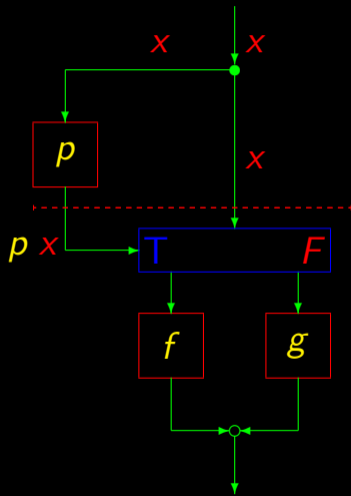
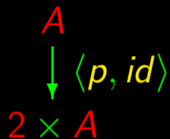
# if-then-else



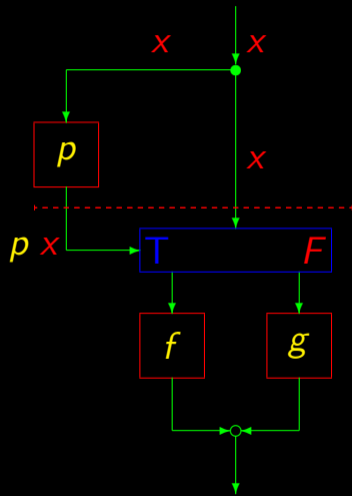
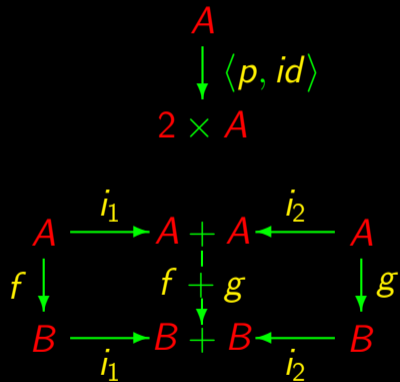
# if-then-else



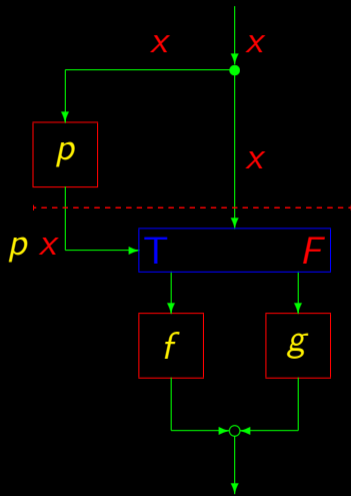
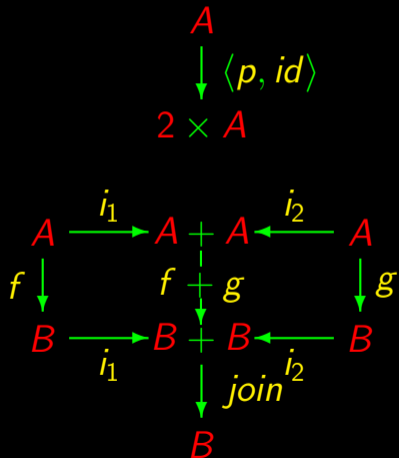
# if-then-else



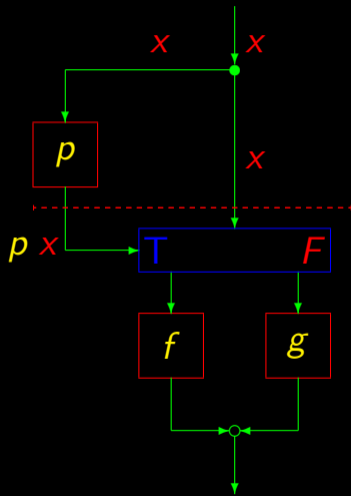
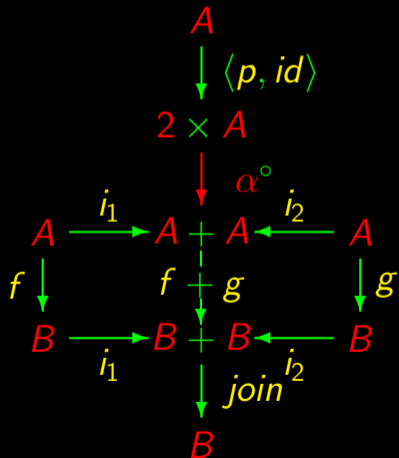
# if-then-else



# if-then-else



# if-then-else





## McCarthy Conditional $p \rightarrow f, g$

From the diagram:

$$p \rightarrow f, g = \text{join} \cdot (f + g) \cdot \alpha^\circ \cdot \langle p, \text{id} \rangle$$

## McCarthy Conditional $p \rightarrow f, g$

From the diagram:

$$p \rightarrow f, g = \text{join} \cdot (f + g) \cdot \alpha^\circ \cdot \langle p, \text{id} \rangle$$

Since  $\text{join} \cdot (f + g) = [\text{id}, \text{id}] \cdot (f + g) = [f, g]$

## McCarthy Conditional $p \rightarrow f, g$

From the diagram:

$$p \rightarrow f, g = \text{join} \cdot (f + g) \cdot \alpha^\circ \cdot \langle p, \text{id} \rangle$$

Since  $\text{join} \cdot (f + g) = [\text{id}, \text{id}] \cdot (f + g) = [f, g]$

we have:

$$p \rightarrow f, g = [f, g] \cdot \underbrace{\alpha^\circ \cdot \langle p, \text{id} \rangle}_{p?}$$

# McCarthy Conditional $p \rightarrow f, g$

$p$ -guard:

$$A \xrightarrow{\langle p, id \rangle} 2 \times A \xrightarrow{\alpha^\circ} A + A$$

$\xrightarrow{p?}$

Conditional:

$$p \rightarrow f, g = [f, g] \cdot p?$$

# Pointwise

$$p? : A \rightarrow A + A$$

$$p? a = \begin{cases} p a \Rightarrow i_1 a \\ \neg (p a) \Rightarrow i_2 a \end{cases}$$

## Pointwise

$$p? : A \rightarrow A + A$$

$$p? a = \begin{cases} p a \Rightarrow i_1 a \\ \neg (p a) \Rightarrow i_2 a \end{cases}$$

wherefrom

$$(p \rightarrow f, g) a = \begin{cases} p a \Rightarrow f a \\ \neg (p a) \Rightarrow g a \end{cases}$$

## Pointwise

$$p? : A \rightarrow A + A$$

$$p? a = \begin{cases} p a \Rightarrow i_1 a \\ \neg (p a) \Rightarrow i_2 a \end{cases}$$

wherefrom

$$(p \rightarrow f, g) a = \begin{cases} p a \Rightarrow f a \\ \neg (p a) \Rightarrow g a \end{cases}$$

Case analysis?

## Pointwise

$$p? : A \rightarrow A + A$$

$$p? a = \begin{cases} p a \Rightarrow i_1 a \\ \neg (p a) \Rightarrow i_2 a \end{cases}$$

wherefrom

$$(p \rightarrow f, g) a = \begin{cases} p a \Rightarrow f a \\ \neg (p a) \Rightarrow g a \end{cases}$$

Case analysis?

No!



# Between conditional and composition

Any “chemistry”?

$$h \cdot (p \rightarrow f, g) = ?$$

# Between conditional and composition

Any “chemistry”?

$$h \cdot (p \rightarrow f, g) = ?$$

$$(p \rightarrow f, g) \cdot h = ?$$

“Calculemus”

$$h \cdot (p \rightarrow f, g)$$

# “Calculus”

$$h \cdot (p \rightarrow f, g)$$

$$\Leftrightarrow \left\{ \text{definition of conditional} \right\}$$

$$h \cdot [f, g] \cdot p?$$

# “Calculus”

$$h \cdot (p \rightarrow f, g)$$

$$\Leftrightarrow \left\{ \text{definition of conditional} \right\}$$

$$h \cdot [f, g] \cdot p?$$

$$\Leftrightarrow \left\{ \text{+-fusion} \right\}$$

$$[h \cdot f, h \cdot g] \cdot p?$$

# “Calculemus”

$$h \cdot (p \rightarrow f, g)$$

$$\Leftrightarrow \left\{ \text{definition of conditional} \right\}$$

$$h \cdot [f, g] \cdot p?$$

$$\Leftrightarrow \left\{ \text{+-fusion} \right\}$$

$$[h \cdot f, h \cdot g] \cdot p?$$

$$\Leftrightarrow \left\{ \text{definition of conditional} \right\}$$

$$p \rightarrow (h \cdot f), (h \cdot g)$$

## Conditional — 1st fusion law

Altogether:

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

## Conditional — 1st fusion law

Altogether:

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

What about running **h before** the conditional?

$$(p \rightarrow f, g) \cdot h = ?$$



## Conditional — 2nd fusion law

Intuitively, one has:

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

## Conditional — 2nd fusion law

Intuitively, one has:

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

Intuitively?

## Conditional — 2nd fusion law

Intuitively, one has:

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

Intuitively?

**Proof ?**

## Conditional — 2nd fusion law

Intuitively, one has:

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

Intuitively?

**Proof** ? Later on...

(We need to know something else about  $p$ ?)

# John McCarthy (1927-2011)

**Turing Award**

*Founding father of **Artificial Intelligence***

PhD in mathematics (Princeton, 1951)

Inventor of **LISP**

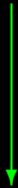


# Exchange law



# Exchange law

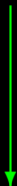
$A + B$



$\alpha$

# Exchange law

$A + B$

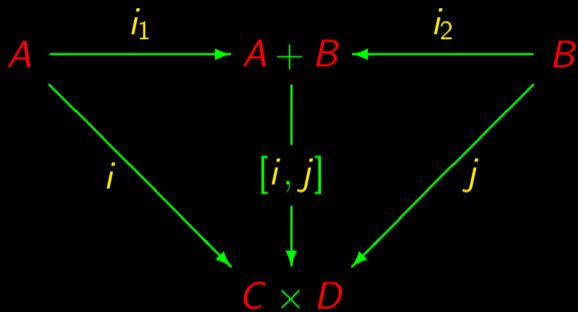


$\alpha$

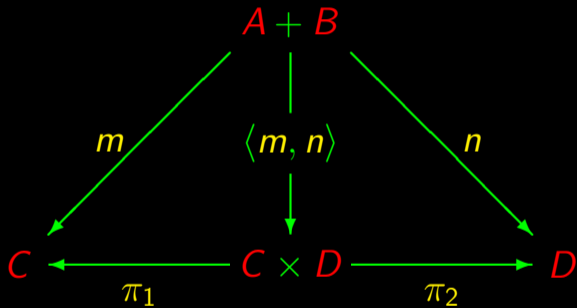
$C \times D$



# Exchange law



# Exchange law

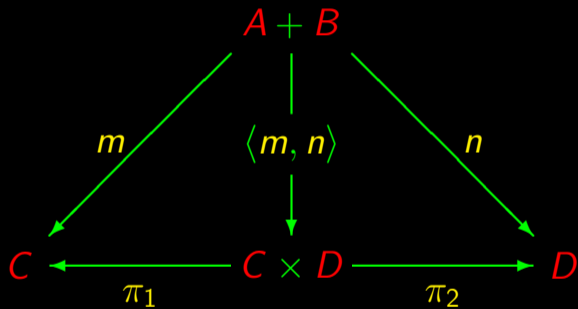


# Exchange law

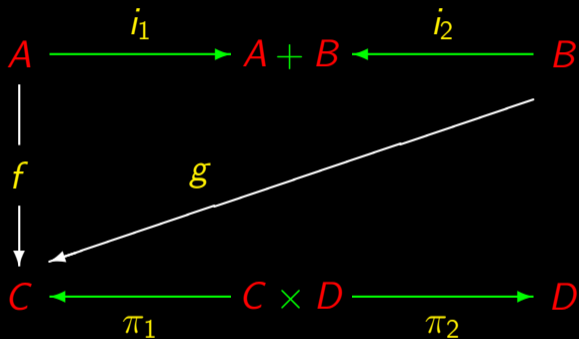
$$\langle m, n \rangle = [i, j]$$

One equation, four unknowns

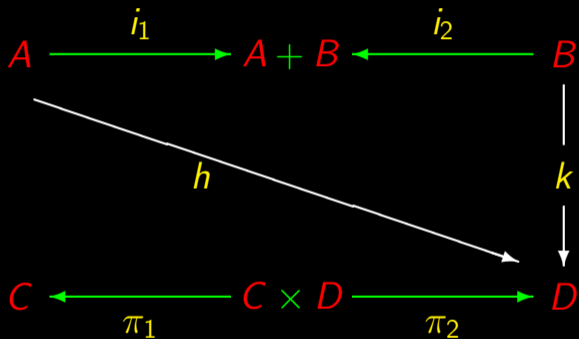
# Exchange law



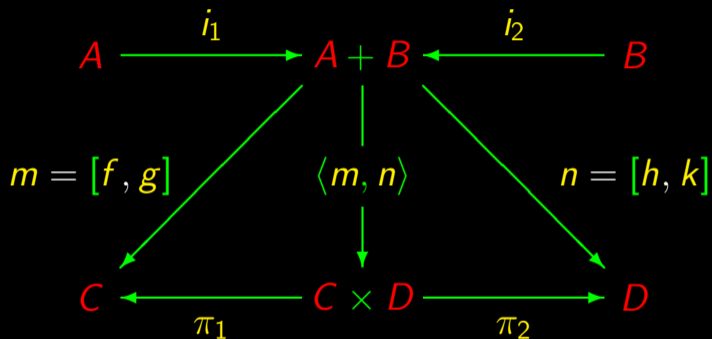
# Exchange law



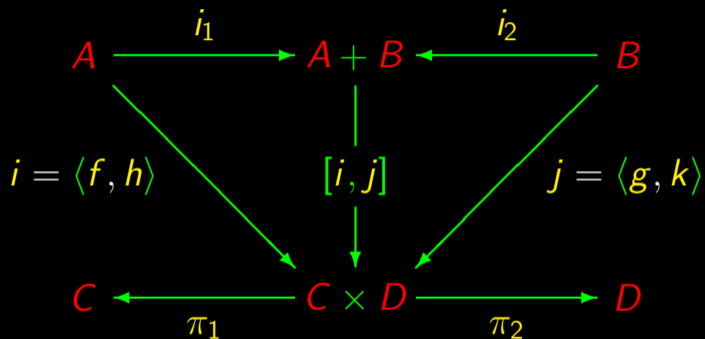
# Exchange law



# Exchange law



# Exchange law





# Summing up

Exchange law:

$$\langle [f, g], [h, k] \rangle = [\langle f, h \rangle, \langle g, k \rangle]$$

“I have a functional **split** but need an **alternative**...”

... and vice-versa

# Cálculo de Programas

Class T05: 20-Mar

# Isomorphisms

$$A \times (B \times C) \cong (A \times B) \times C$$



$$A \times B \cong B \times A$$



# Isomorphisms

$$A \times (B \times C) \cong (A \times B) \times C$$



$$A \times B \cong B \times A$$



$$A + (B + C) \cong (A + B) + C$$

$$A + B \cong B + A$$

# Isomorphisms

$$A + A \cong 2 \times A$$



# Isomorphisms

$$A + A \cong 2 \times A$$



$$A \times A \cong A^2 \quad ?$$

# Isomorphisms

$$A + A \cong 2 \times A$$



$$A \times A \cong A^2 \quad ?$$

$$A \times 1 \cong A \quad ?$$

# Isomorphisms

$$A + A \cong 2 \times A$$



$$A \times A \cong A^2 \quad ?$$

$$A \times 1 \cong A \quad ?$$

$$A \times 0 \cong 0 \quad ?$$



# Isomorphisms

$$A + A \cong 2 \times A$$



$$A \times A \cong A^2 \quad ?$$

$$A \times 1 \cong A \quad ?$$

$$A \times 0 \cong 0 \quad ?$$

$$A + 0 \cong A \quad ?$$

# Isomorphisms

$$A + A \cong 2 \times A$$



$$A \times A \cong A^2 \quad ?$$

$$A \times 1 \cong A \quad ?$$

$$A \times 0 \cong 0 \quad ?$$

$$A + 0 \cong A \quad ?$$

$$A \times (B + C) \cong A \times B + A \times C \quad ?$$

# Distributivity

$$A \times (B + C) \cong A \times B + A \times C$$

The diagram illustrates the distributivity property. It features two expressions:  $A \times (B + C)$  on the left and  $A \times B + A \times C$  on the right. A central symbol  $\cong$  indicates that these two expressions are isomorphic. Two curved arrows connect the expressions: the top arrow points from left to right and is labeled *distr*, while the bottom arrow points from right to left and is labeled *undistr*.

# Distributivity

$$A \times (B + C) \cong A \times B + A \times C$$

*distr*

*undistr*

$$(B + C) \times A \cong B \times A + C \times A$$

*distl*

*undistl*

# Distributivity

$$A \times (B + C) \stackrel{\text{distr}}{\cong} A \times B + A \times C$$

The diagram illustrates the distributivity property. It shows two expressions:  $A \times (B + C)$  on the left and  $A \times B + A \times C$  on the right. A double-headed arrow connects them, with the label *distr* above the arrow and *undistr* below the arrow. The expressions are separated by an equivalence symbol  $\cong$ . The variables  $A$ ,  $B$ , and  $C$  are color-coded:  $A$  is green,  $B$  is red, and  $C$  is green.

# Distributivity

$$\begin{array}{ccc} & \xrightarrow{\text{distr}} & \\ A \times (B + C) & \cong & A \times B + A \times C \\ & \xleftarrow{\text{undistr}} & \end{array}$$

$$\text{undistr} = [\langle f, g \rangle, \langle h, k \rangle]$$

# Distributivity

$$\begin{array}{ccc} & \xrightarrow{\text{distr}} & \\ A \times (B + C) & \cong & A \times B + A \times C \\ & \xleftarrow{\text{undistr}} & \end{array}$$

$$\text{undistr} = [\langle f, g \rangle, \langle h, k \rangle]$$

$$A \xleftarrow{f} A \times B$$

# Distributivity

$$\begin{array}{ccc} & \xrightarrow{\text{distr}} & \\ A \times (B + C) & \cong & A \times B + A \times C \\ & \xleftarrow{\text{undistr}} & \end{array}$$

$$\text{undistr} = [\langle f, g \rangle, \langle h, k \rangle]$$

$$A \xleftarrow{f} A \times B$$

$$B + C \xleftarrow{g} A \times B$$



# Distributivity

$$\begin{array}{ccc} & \xrightarrow{\text{distr}} & \\ A \times (B + C) & \cong & A \times B + A \times C \\ & \xleftarrow{\text{undistr}} & \end{array}$$

$$\text{undistr} = [\langle f, g \rangle, \langle h, k \rangle]$$

$$A \xleftarrow{f} A \times B$$

$$A \xleftarrow{h} A \times C$$

$$B + C \xleftarrow{g} A \times B$$

# Distributivity

$$\begin{array}{ccc} & \xrightarrow{\text{distr}} & \\ A \times (B + C) & \cong & A \times B + A \times C \\ & \xleftarrow{\text{undistr}} & \end{array}$$

$$\text{undistr} = [\langle f, g \rangle, \langle h, k \rangle]$$

$$A \xleftarrow{f} A \times B$$

$$B + C \xleftarrow{g} A \times B$$

$$A \xleftarrow{h} A \times C$$

$$B + C \xleftarrow{k} A \times C$$

# Distributivity

$$\text{undistr} = [\langle f, g \rangle, \langle h, k \rangle]$$

$$A \xleftarrow{f} A \times B$$

$$B + C \xleftarrow{g} A \times B$$

$$A \xleftarrow{h} A \times C$$

$$B + C \xleftarrow{k} A \times C$$

# Distributivity

$$\text{undistr} = [\langle f, g \rangle, \langle h, k \rangle]$$

$$A \xleftarrow{f} A \times B$$

$$B + C \xleftarrow{g} A \times B$$

$$A \xleftarrow{\pi_1} A \times B$$

$$A \xleftarrow{h} A \times C$$

$$B + C \xleftarrow{k} A \times C$$

# Distributivity

$$\text{undistr} = [\langle f, g \rangle, \langle h, k \rangle]$$

$$A \xleftarrow{f} A \times B$$

$$B + C \xleftarrow{g} A \times B$$

$$A \xleftarrow{\pi_1} A \times B$$

$$B + C \xleftarrow{i_1 \cdot \pi_2} A \times B$$

$$A \xleftarrow{h} A \times C$$

$$B + C \xleftarrow{k} A \times C$$

# Distributivity

$$\text{undistr} = [\langle f, g \rangle, \langle h, k \rangle]$$

$$A \xleftarrow{f} A \times B$$

$$B + C \xleftarrow{g} A \times B$$

$$A \xleftarrow{\pi_1} A \times B$$

$$B + C \xleftarrow{j_1 \cdot \pi_2} A \times B$$

$$A \xleftarrow{h} A \times C$$

$$B + C \xleftarrow{k} A \times C$$

$$A \xleftarrow{\pi_1} A \times C$$

# Distributivity

$$\text{undistr} = [\langle f, g \rangle, \langle h, k \rangle]$$

$$A \xleftarrow{f} A \times B$$

$$A \xleftarrow{h} A \times C$$

$$B + C \xleftarrow{g} A \times B$$

$$B + C \xleftarrow{k} A \times C$$

$$A \xleftarrow{\pi_1} A \times B$$

$$A \xleftarrow{\pi_1} A \times C$$

$$B + C \xleftarrow{i_1 \cdot \pi_2} A \times B$$

$$B + C \xleftarrow{i_2 \cdot \pi_2} A \times C$$

$$\text{undistr} = [\langle \pi_1, i_1 \cdot \pi_2 \rangle, \langle \pi_1, i_2 \cdot \pi_2 \rangle]$$

# Distributivity

$$\text{undistr} = [\langle \pi_1, i_1 \cdot \pi_2 \rangle, \langle \pi_1, i_2 \cdot \pi_2 \rangle]$$

$$\begin{array}{ccc} & \xrightarrow{\text{distr}} & \\ A \times (B + C) & \cong & A \times B + A \times C \\ & \xleftarrow{\text{undistr}} & \end{array}$$



# Distributivity

$$\mathit{undistr} = [\langle \pi_1, i_1 \cdot \pi_2 \rangle, \langle \pi_1, i_2 \cdot \pi_2 \rangle]$$

$$\begin{array}{ccc} & \xrightarrow{\mathit{distr}} & \\ A \times (B + C) & \cong & A \times B + A \times C \\ & \xleftarrow{\mathit{undistr}} & \end{array}$$

By definition of  $f \times g$

$$\mathit{undistr} = [id \times i_1, id \times i_2]$$

# The type $1$

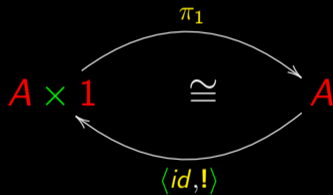
In Haskell:

$() :: ()$

Then

$1 = \{()\}$

Isomorphism



# The type 1

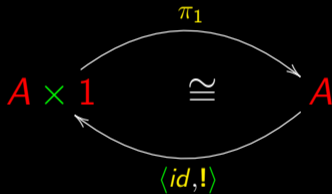
In Haskell:

`() :: ()`

Then

`1 = { () }`

Isomorphism



... "!" ?

# The type 1

**Fact** functions involving type 1  
are necessarily **constant**

# The type 1

**Fact** functions involving type 1  
are necessarily **constant**

”Bang”:

$$A \xrightarrow{!} 1$$

$$! = \underline{()}$$

# The type 1

**Fact** functions involving type 1  
are necessarily **constant**

"Bang":

$$A \xrightarrow{!} 1$$

$$! = \underline{()}$$

"Points":

$$1 \xrightarrow{a} A$$

$$\underline{a} () = a$$

(provided  $a \in A$ )

The type 0

$$0 = \{\}$$

## The type 0

$$0 = \{\}$$

Immediate in e.g. Haskell:

```
data Zero
```



# The type 0

$$0 = \{\}$$

Immediate in e.g. Haskell:

```
data Zero
```

0 is not inhabited:  $\neg (a \in 0)$   
for all  $a$

# The type 0

$$0 = \{\}$$

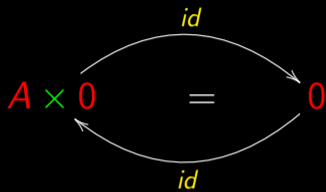
Immediate in e.g. Haskell:

```
data Zero
```

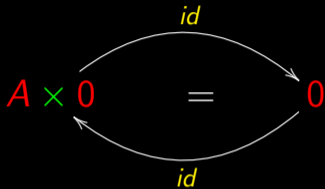
0 is not inhabited:  $\neg (a \in 0)$   
for all  $a$

**Fact** impossible  $f : A \rightarrow 0$  whenever  $A \neq 0$

The type  $0$



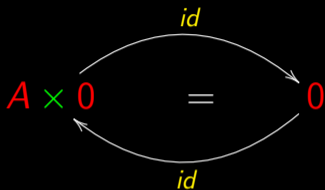
# The type 0



In fact

$$A \times 0 = \{(a, b) \mid a \in A \wedge b \in 0\}$$

# The type 0

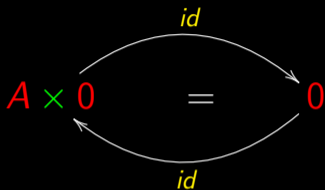


In fact

$$A \times 0 = \{(a, b) \mid a \in A \wedge b \in 0\}$$

$$A \times 0 = \{(a, b) \mid a \in A \wedge F\}$$

# The type 0



In fact

$$A \times 0 = \{(a, b) \mid a \in A \wedge b \in 0\}$$

$$A \times 0 = \{(a, b) \mid a \in A \wedge F\}$$

$$A \times 0 = \{\}$$

The type  $1 + A$

$1 + A$

The type  $1 + A$

$$1 \xrightarrow{i_1} 1 + A \xleftarrow{i_2} A$$

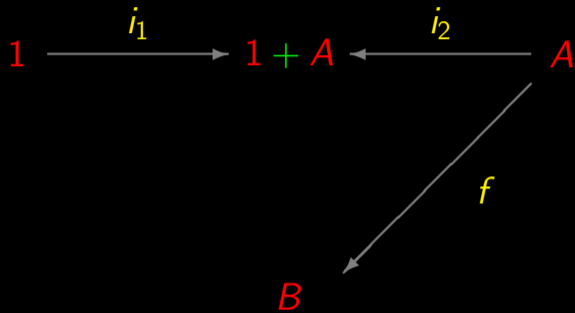


The type  $1 + A$

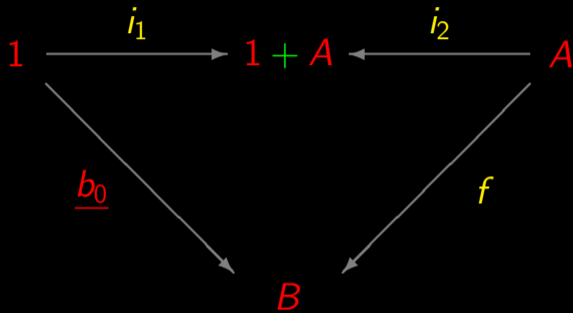
$$1 \xrightarrow{i_1} 1 + A \xleftarrow{i_2} A$$

$B$

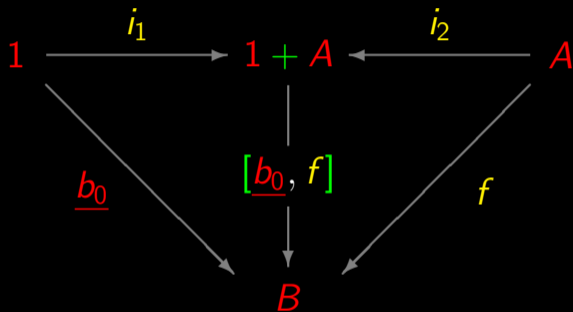
The type  $1 + A$



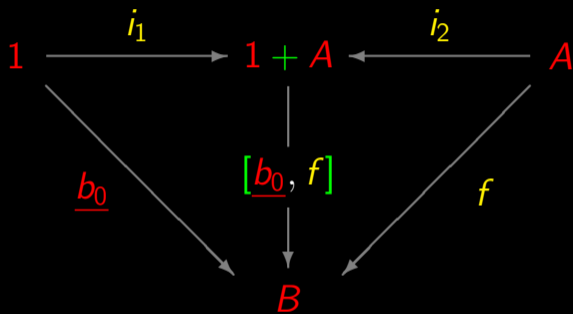
# The type $1 + A$



# The type $1 + A$



# The type $1 + A$



$1 + A$

"pointer to  $A$ "

More isomorphisms!

# More isomorphisms!

Back to

*Retrieve the address of a civil servant, knowing that she/he can be identified either by a citizen card number (CC) or a fiscal number (NIF).*

# More isomorphisms!

Back to

*Retrieve the address of a civil servant, knowing that she/he can be identified either by a citizen card number (**CC**) or a fiscal number (**NIF**).*

In Haskell:

```
data Iden = CC Int | NIF Int
```



## More isomorphisms!

*Obter a morada of a funcionário público, knowing that este se can identificar através d its number of the cartão of cidadão (**CC**) ou d its number of identificação fiscal (**NIF**).*

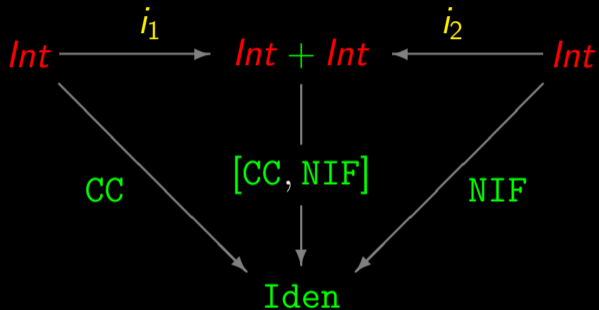
In Haskell (Portugal):

```
data Iden = CC Int | NIF Int
```

In Haskell (Germany):

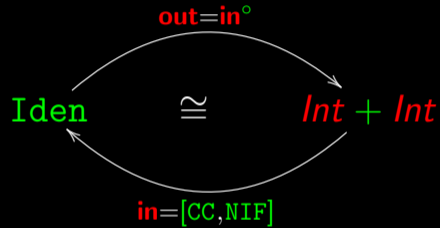
```
data Iden = IDK Int | SZN Int
```

# More isomorphisms!

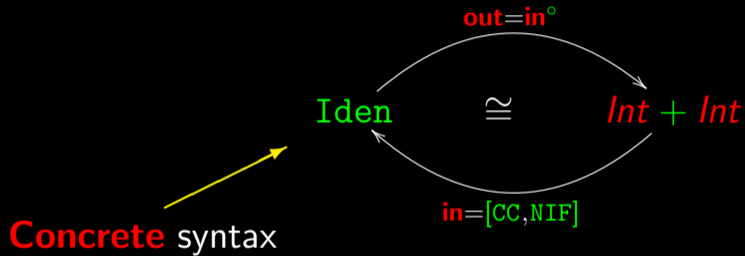


```
data Iden = CC Int | NIF Int
```

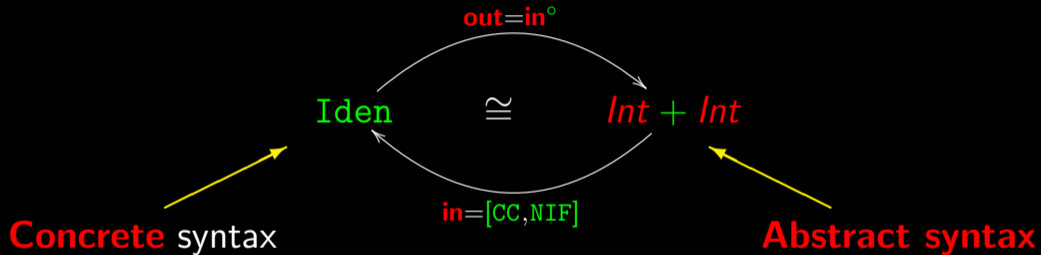
# More isomorphisms!



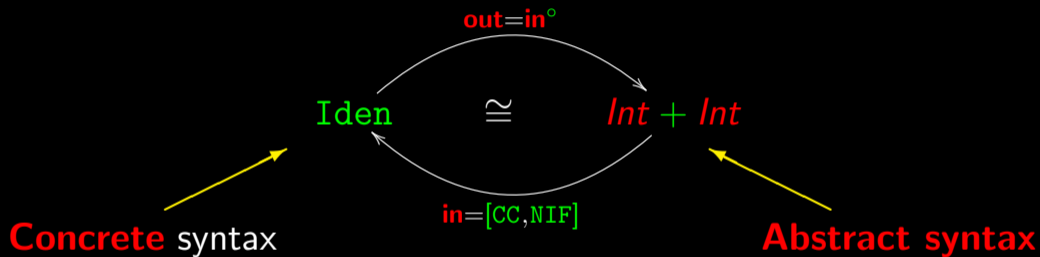
# More isomorphisms!



# More isomorphisms!

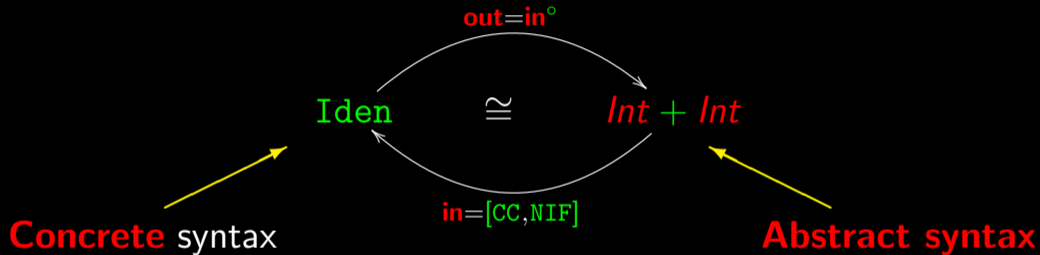


# More isomorphisms!



**in** ("get in")

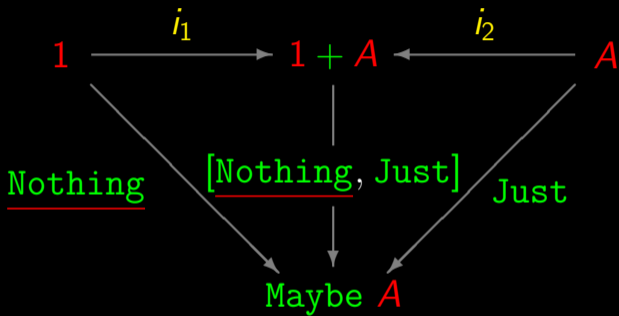
# More isomorphisms!



**in** (“get in”)

**out** (“get out”)

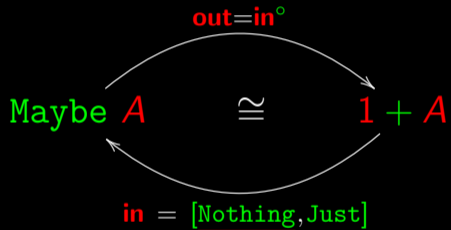
# Maybe



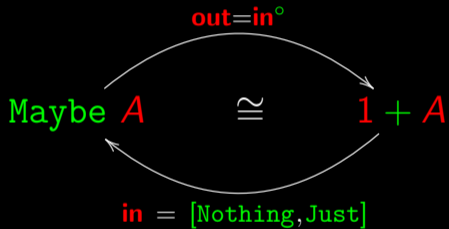
```
data Maybe a = Nothing | Just a
```



# Maybe



# Maybe



$$\text{out} \cdot \text{in} = \text{id}$$

# Maybe

$$\mathbf{out} \cdot \mathbf{in} = id$$

# Maybe

$$\text{out} \cdot \text{in} = id$$

$$\Leftrightarrow \{ \text{in} = [\underline{\text{Nothing}}, \text{Just}] \}$$

$$\text{out} \cdot [\underline{\text{Nothing}}, \text{Just}] = id$$

# Maybe

$$\mathbf{out} \cdot \mathbf{in} = id$$

$$\Leftrightarrow \{ \mathbf{in} = [\underline{\mathbf{Nothing}}, \mathbf{Just}] \}$$

$$\mathbf{out} \cdot [\underline{\mathbf{Nothing}}, \mathbf{Just}] = id$$

$$\Leftrightarrow \{ \text{+-fusion} \}$$

$$[\mathbf{out} \cdot \underline{\mathbf{Nothing}}, \mathbf{out} \cdot \mathbf{Just}] = id$$

# Maybe

$$\mathbf{out} \cdot \mathbf{in} = id$$

$$\Leftrightarrow \{ \mathbf{in} = [\underline{\mathbf{Nothing}}, \mathbf{Just}] \}$$

$$\mathbf{out} \cdot [\underline{\mathbf{Nothing}}, \mathbf{Just}] = id$$

$$\Leftrightarrow \{ \text{+-fusion} \}$$

$$[\mathbf{out} \cdot \underline{\mathbf{Nothing}}, \mathbf{out} \cdot \mathbf{Just}] = id$$

$$\Leftrightarrow \{ \text{universal-+; constant function} \}$$

$$\left\{ \begin{array}{l} \underline{\mathbf{out} \cdot \mathbf{Nothing}} = i_1 \\ \mathbf{out} \cdot \mathbf{Just} = i_2 \end{array} \right.$$

# Maybe

$\Leftrightarrow$  { variables ; Nothing : 1  $\rightarrow$  Maybe A }

{ out Nothing () =  $i_1$  ()  
(out · Just) a =  $i_2$  a }

# Maybe

$\Leftrightarrow$  { variables ; Nothing :  $1 \rightarrow \text{Maybe } A$  }

$$\left\{ \begin{array}{l} \text{out Nothing } () = i_1 () \\ (\text{out} \cdot \text{Just}) a = i_2 a \end{array} \right.$$

$\Leftrightarrow$  { composition (*pointwise*) }

$$\left\{ \begin{array}{l} \text{out Nothing} = i_1 () \\ \text{out (Just } a) = i_2 a \end{array} \right.$$



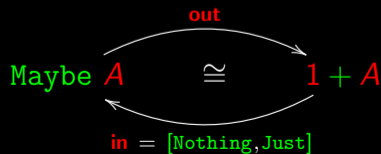
# Maybe

$\Leftrightarrow$  { variables ; Nothing :  $1 \rightarrow \text{Maybe } A$  }

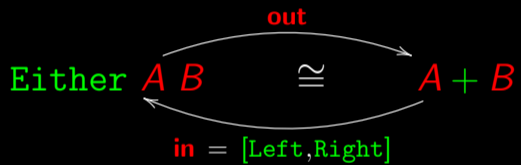
$$\left\{ \begin{array}{l} \text{out Nothing } () = i_1 () \\ (\text{out} \cdot \text{Just}) a = i_2 a \end{array} \right.$$

$\Leftrightarrow$  { composition (*pointwise*) }

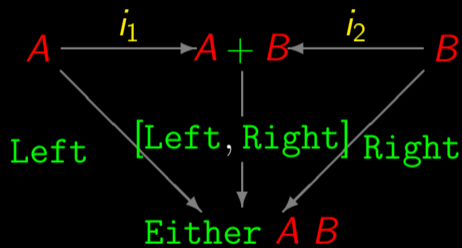
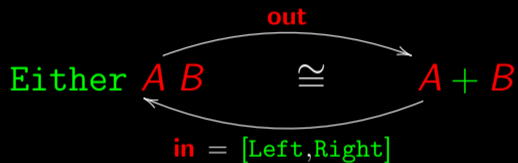
$$\left\{ \begin{array}{l} \text{out Nothing} = i_1 () \\ \text{out (Just } a) = i_2 a \end{array} \right.$$



# Either



# Either



`[f, g]`

`either f g`

# Polymorphism (and why it matters!)

# Polymorphism

*reverse* :: [a] → [a]

# Polymorphism

*reverse* :: [ *a* ] → [ *a* ]

*reverse* :: [ *Int* ] → [ *Int* ]

# Polymorphism

*reverse* :: [a] → [a]

*reverse* :: [Int] → [Int]

*reverse* :: String → String

# Polymorphism

*reverse* :: [a] → [a]

*reverse* :: [Int] → [Int]

*reverse* :: String → String

...



# Polymorphism

*reverse* :: [ *a* ] → [ *a* ]

*reverse* :: [ *Int* ] → [ *Int* ]

*reverse* :: *String* → *String*

...

“From the **type** of a **polymorphic function** we can derive a **theorem** that it satisfies. (...) How useful are the theorems so generated? Only time and experience will tell (...)” [Philip Wadler 1989]

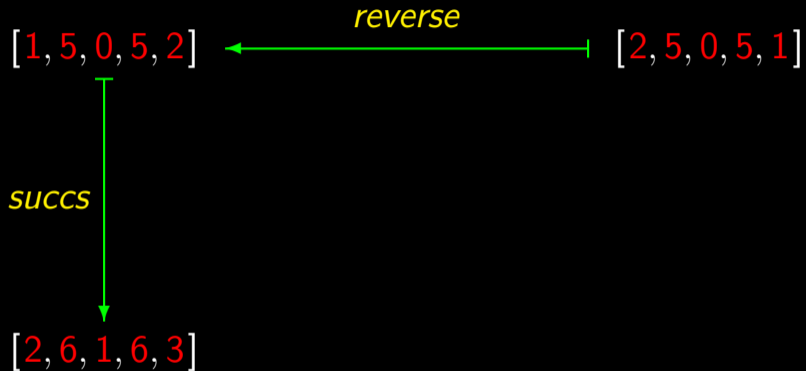
# Natural properties

[2, 5, 0, 5, 1]

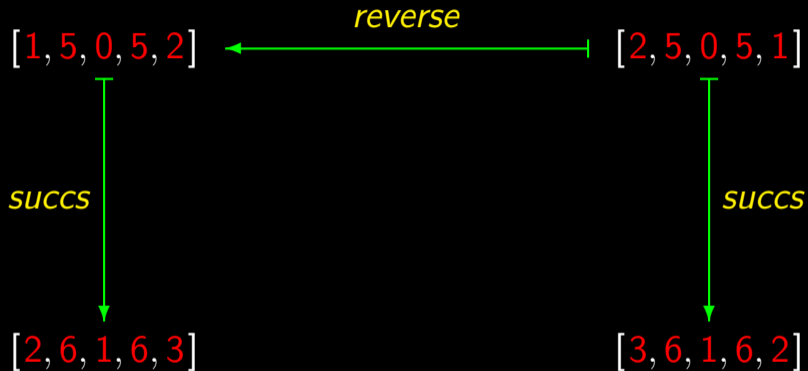
# Natural properties



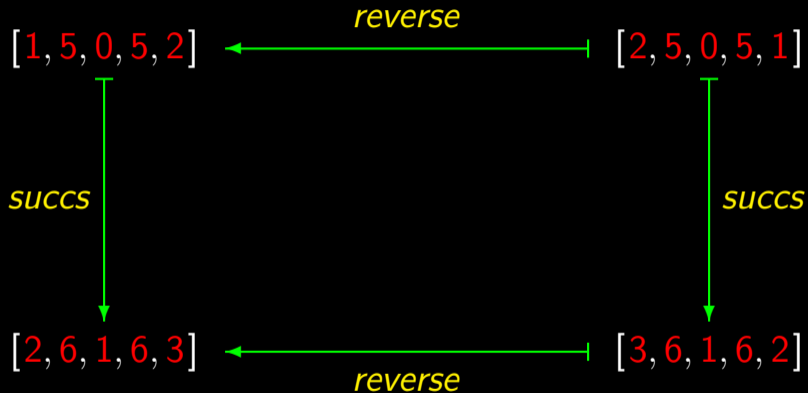
# Natural properties



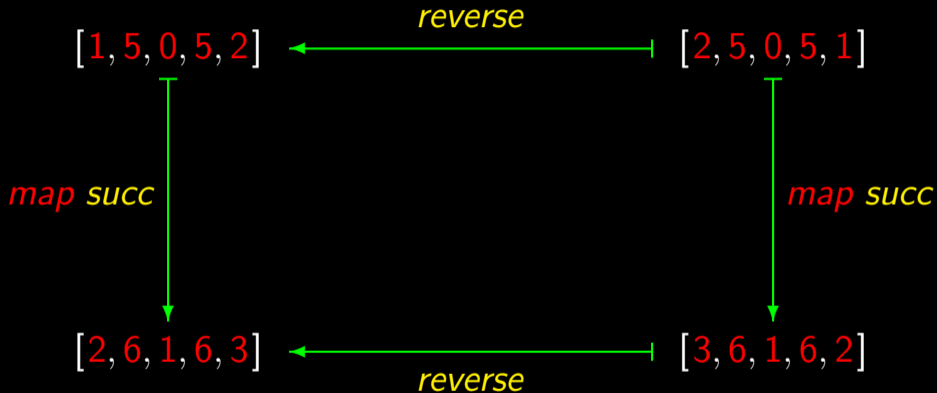
# Natural properties



# Natural properties



# Natural properties

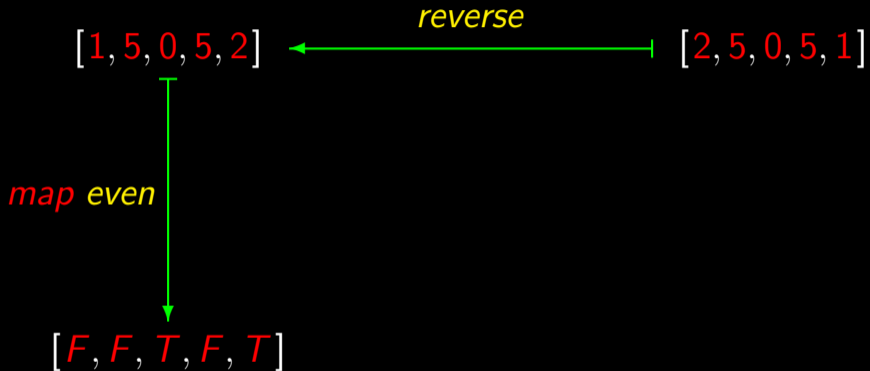


# Natural properties

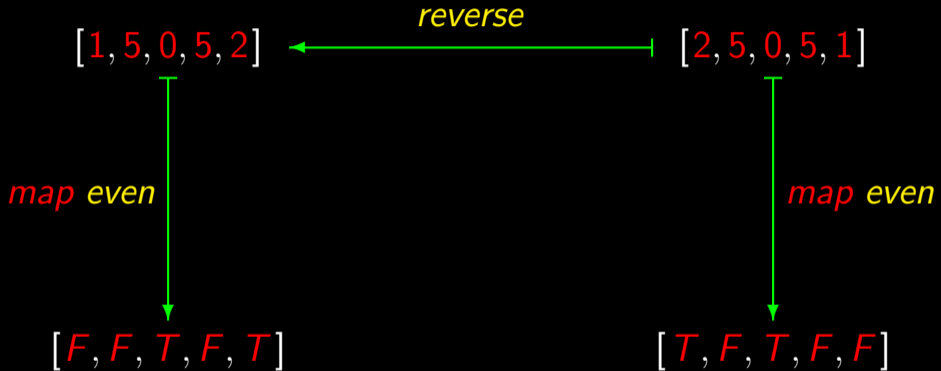




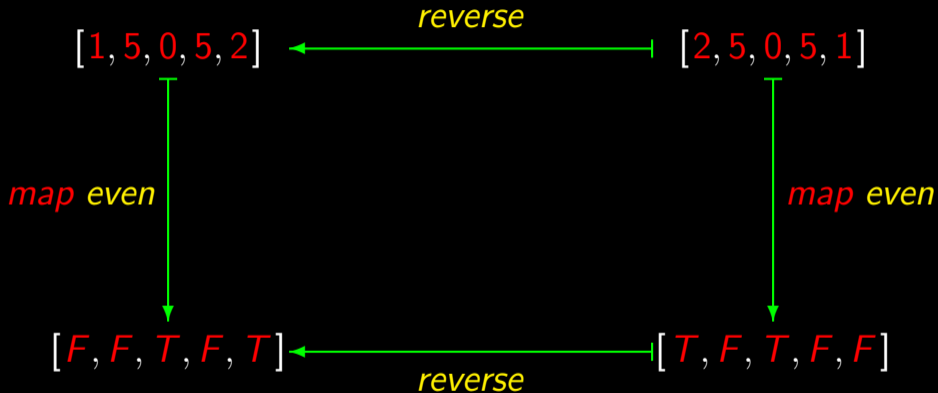
# Natural properties



# Natural properties



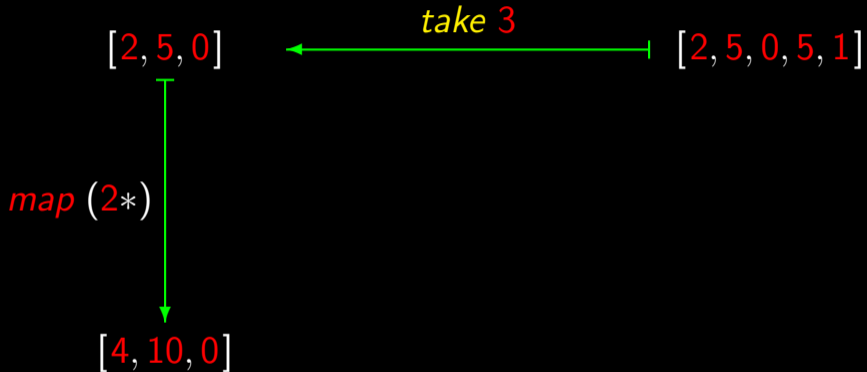
# Natural properties



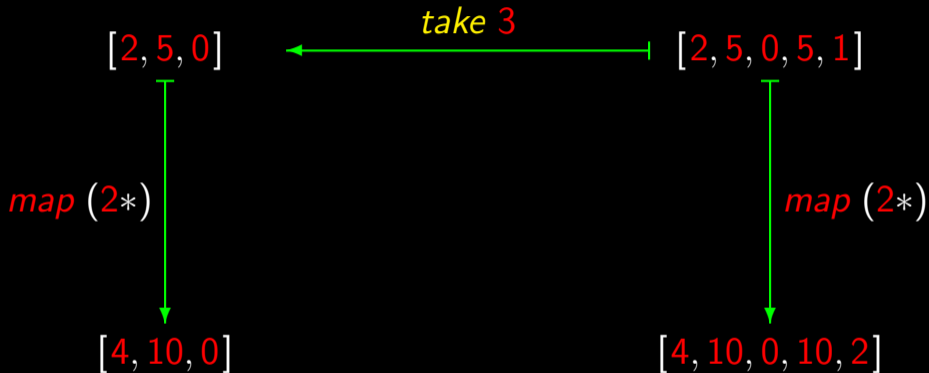
# Natural properties



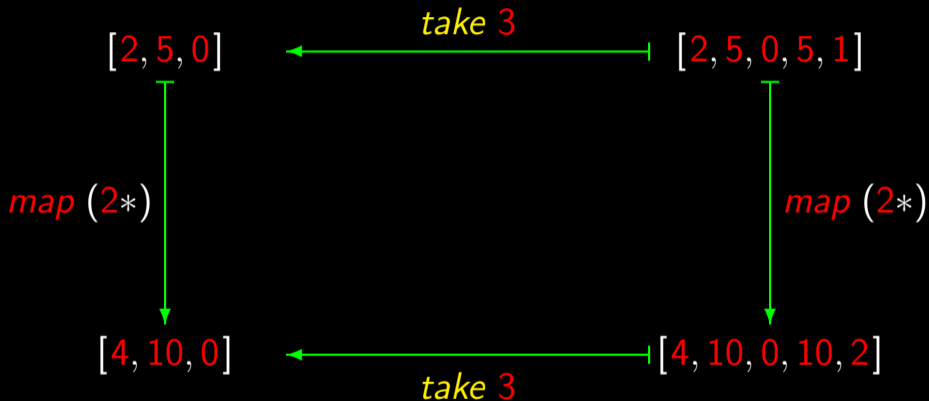
# Natural properties



# Natural properties



# Natural properties



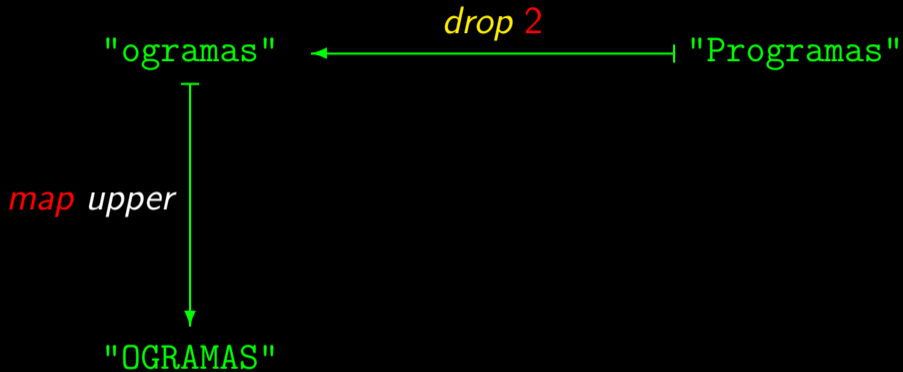
# Natural properties

"ogramas" ← *drop 2* "Programas"

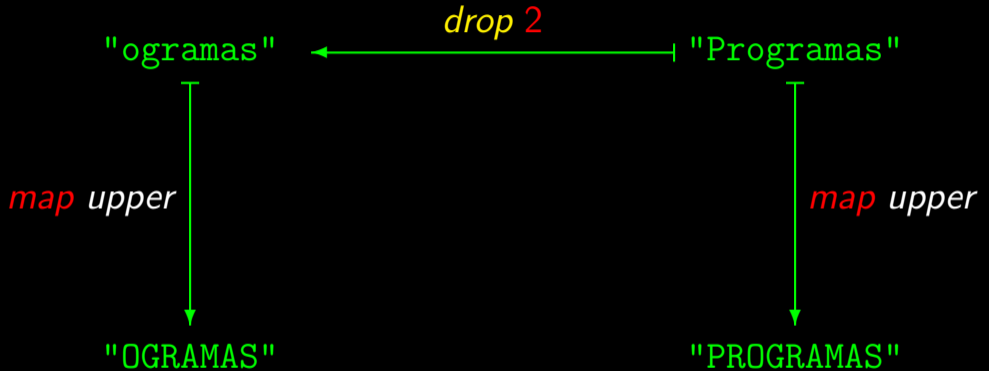
A diagram illustrating the 'drop 2' operation. It shows the string "ogramas" on the left and "Programas" on the right. A green arrow points from the right side of "Programas" to the left side of "ogramas". Above the arrow, the text "drop 2" is written in a yellow font. The string "ogramas" is written in a green font, and "Programas" is written in a black font.



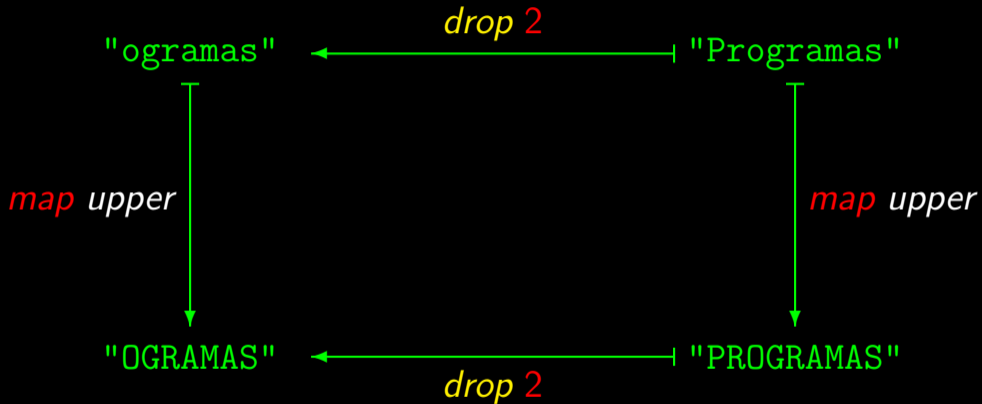
# Natural properties



# Natural properties



# Natural properties



Back to...

$$\begin{array}{ccc} & \xrightarrow{\text{assocr}} & \\ (A \times B) \times C & \cong & A \times (B \times C) \\ & \xleftarrow{\text{assocl}} & \end{array}$$

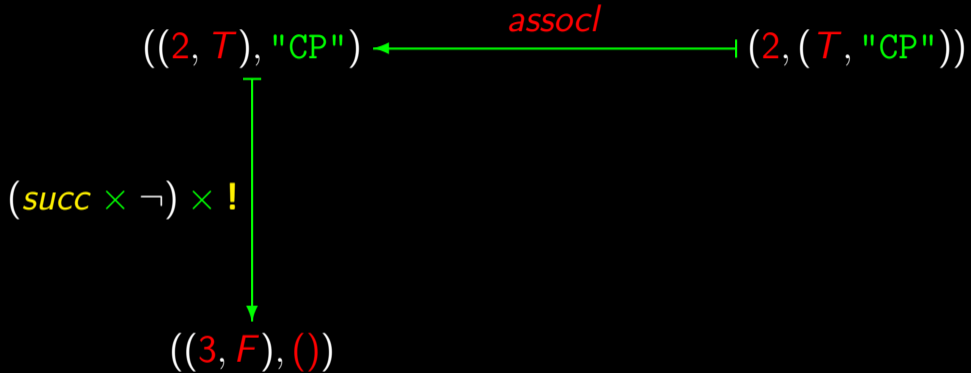
$$\text{assocr} \cdot \text{assocl} = \text{id}$$

$$\text{assocl} \cdot \text{assocr} = \text{id}$$

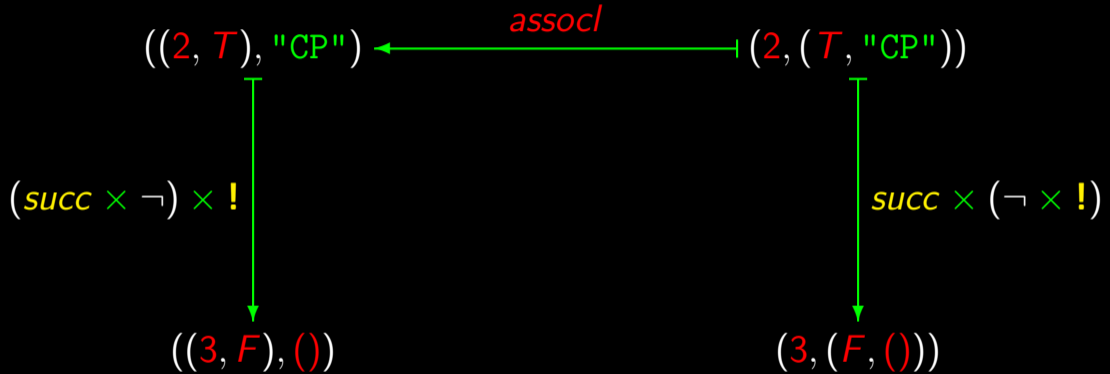
# “Natural” property

$$((2, T), \text{"CP"}) \xleftarrow{\text{assocl}} (2, (T, \text{"CP"}))$$

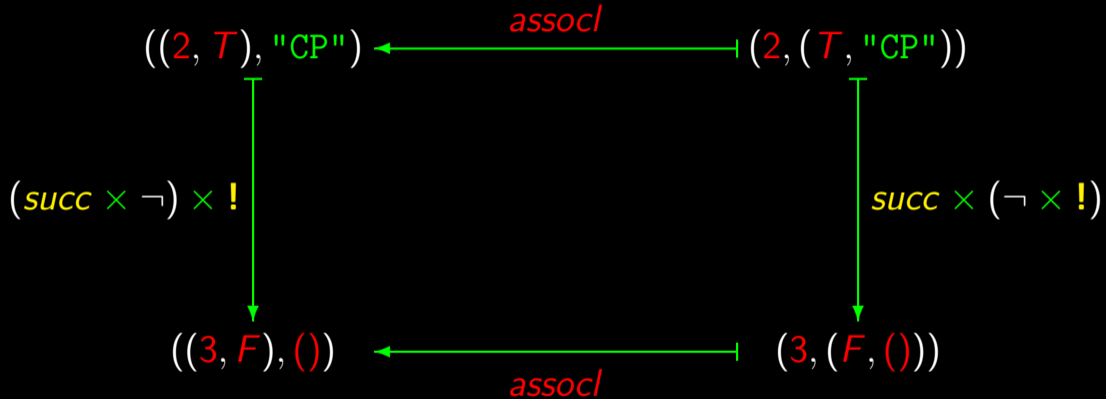
# “Natural” property



# “Natural” property



# “Natural” property

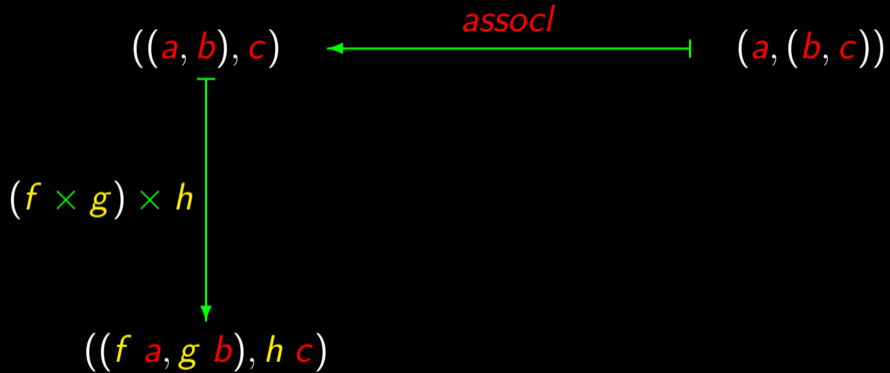




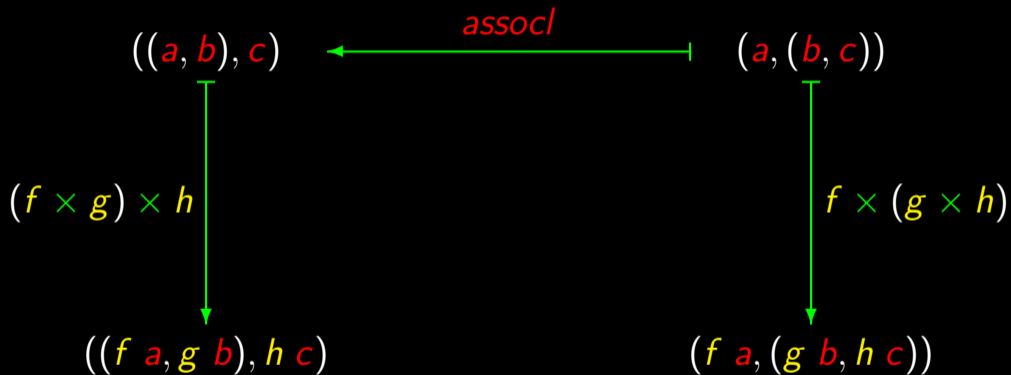
# “Natural” property

$$((a, b), c) \xleftarrow{\text{assocl}} (a, (b, c))$$

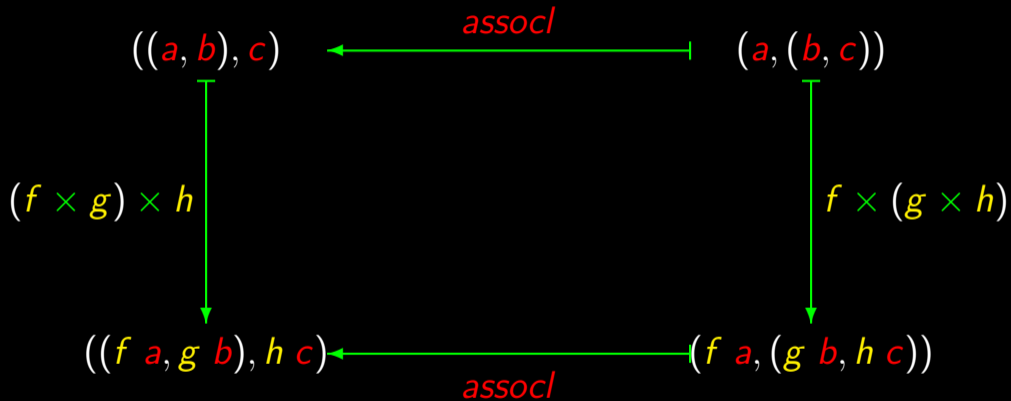
# “Natural” property



# “Natural” property



# “Natural” property



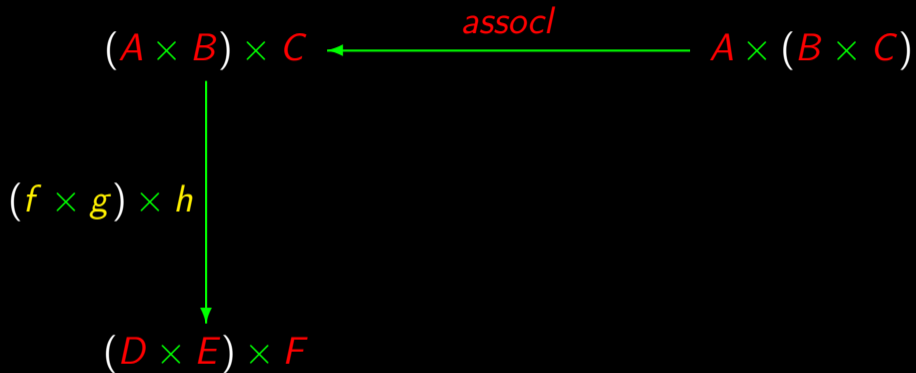
# Natural properties

$$A \times (B \times C)$$

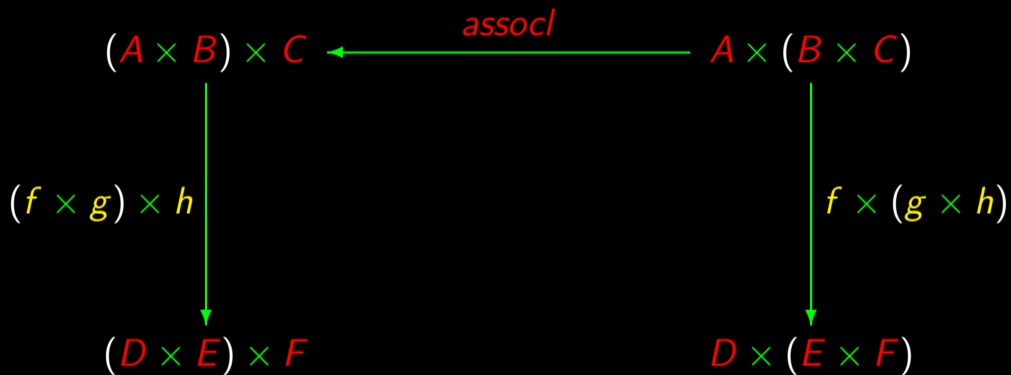
# Natural properties

$$(A \times B) \times C \xleftarrow{\text{assocl}} A \times (B \times C)$$

# Natural properties

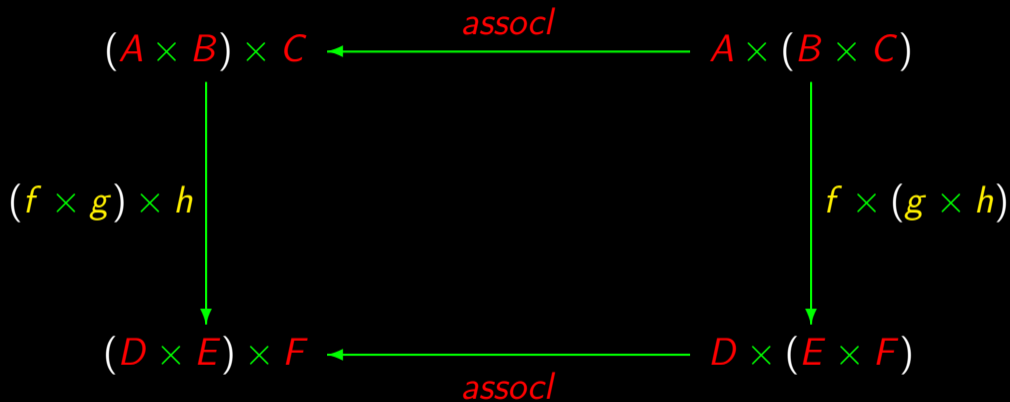


# Natural properties



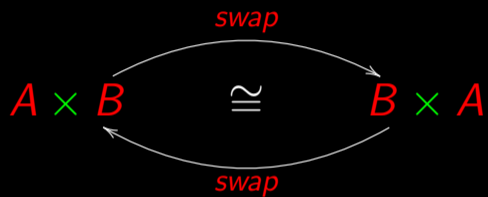


# Natural properties



$$((f \times g) \times h) \cdot \text{assocl} = \text{assocl} \cdot (f \times (g \times h))$$

Back to...



$$\text{swap} \cdot \text{swap} = \text{id}$$

# Practical rule

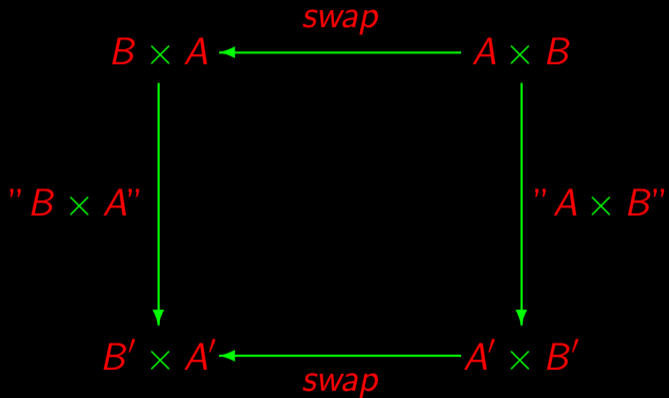
$$B \times A \xleftarrow{\text{swap}} A \times B$$

# Practical rule

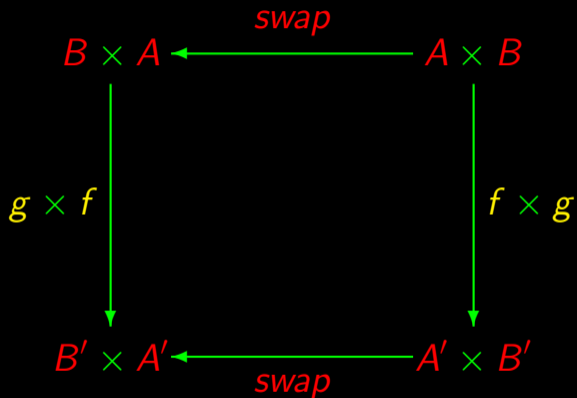
$$B \times A \xleftarrow{\text{swap}} A \times B$$

$$B' \times A' \xleftarrow{\text{swap}} A' \times B'$$

# Practical rule



## Practical rule



$$(g \times f) \cdot \text{swap} = \text{swap} \cdot (f \times g)$$

## Practical rule

In the vertical arrows,

- ▶ Substitute  $A := f$ ,  $B := g$ , etc

# Practical rule

In the vertical arrows,

- ▶ Substitute  $A := f$ ,  $B := g$ , etc
- ▶ In case of concrete types, replace by  $id$ , e.g.  $2 := id$



## Practical rule

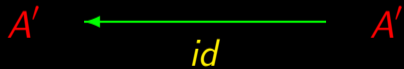
In the vertical arrows,

- ▶ Substitute  $A := f$ ,  $B := g$ , etc
- ▶ In case of concrete types, replace by *id*, e.g.  $2 := id$
- ▶ Remove the quotes.

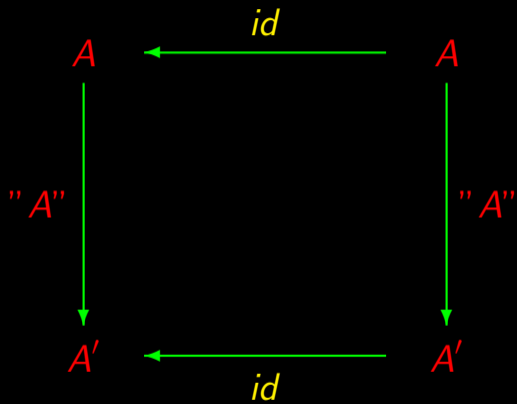
Simplest case...



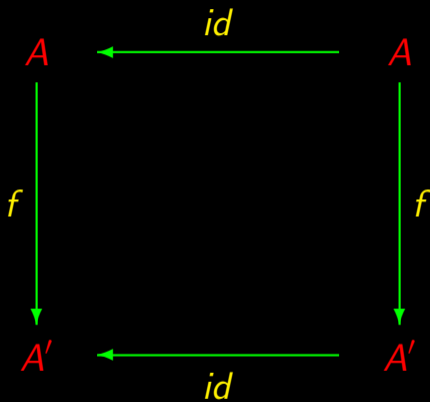
Simplest case...



Simplest case...



Natural-*id*



$$f \cdot id = id \cdot f$$

Natural- $\pi_1$

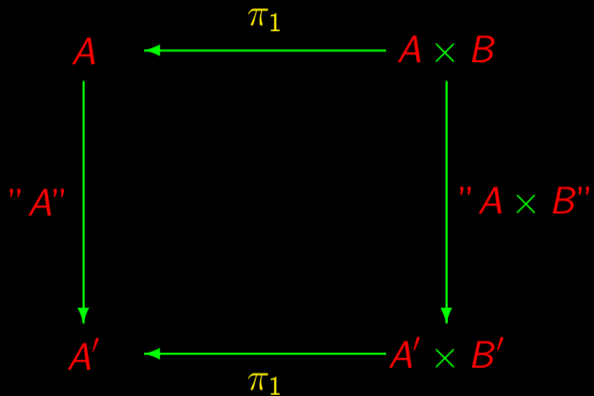
$$A \xleftarrow{\pi_1} A \times B$$

# Natural- $\pi_1$

$$A \xleftarrow{\pi_1} A \times B$$

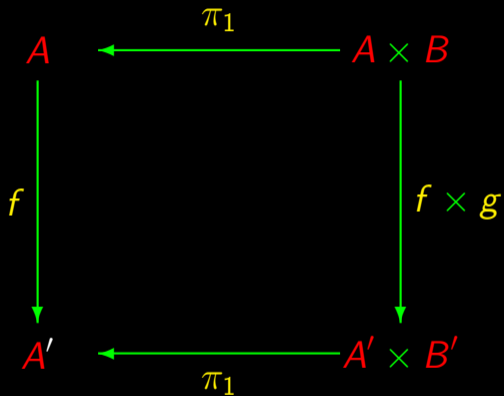
$$A' \xleftarrow{\pi_1} A' \times B'$$

# Natural- $\pi_1$



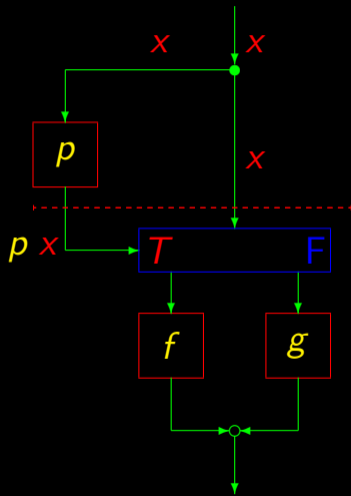
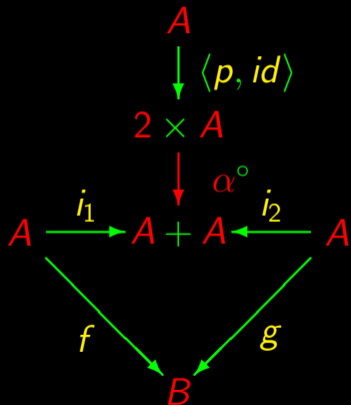


# Natural- $\pi_1$ 😊



$$(f) \cdot \pi_1 = \pi_1 \cdot (f \times g)$$

# Back to "if-then-else"



## 2<sup>a</sup> fusion law of conditionals

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

## 2<sup>a</sup> fusion law of conditionals

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

follows from

$$p? \cdot f = (f + f) \cdot (p \cdot f)?$$

## 2<sup>a</sup> fusion law of conditionals

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

follows from

$$p? \cdot f = (f + f) \cdot (p \cdot f)?$$

So we are left with the **proof** of this thence...

Let us do it now

Prove

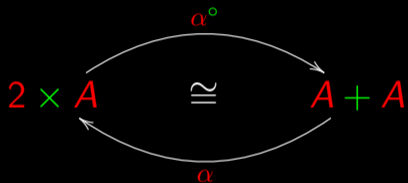
$$p? \cdot f = (f + f) \cdot (p \cdot f)?$$

knowing:

$$A \xrightarrow{\langle p, id \rangle} 2 \times A \xrightarrow{\alpha^{\circ}} A + A$$

$p?$

# Recall



$$\alpha = [\langle \underline{T}, id \rangle, \langle \underline{F}, id \rangle]$$

$$\alpha^\circ = \dots ?$$

Let us try it

$$p? \cdot f$$



Let us try it

$$\begin{aligned} & p? \cdot f \\ = & \left\{ \text{substitution } p? = \alpha^\circ \cdot \langle p, id \rangle \right\} \\ & \alpha^\circ \cdot \langle p, id \rangle \cdot f \end{aligned}$$

Let us try it

$$p? \cdot f$$

$$= \left\{ \text{substitution } p? = \alpha^\circ \cdot \langle p, id \rangle \right\}$$

$$\alpha^\circ \cdot \langle p, id \rangle \cdot f$$

$$= \left\{ \times\text{-fusion} \right\}$$

$$\alpha^\circ \cdot \langle p \cdot f, id \cdot f \rangle$$

# Let us try it

$$p? \cdot f$$

$$= \left\{ \text{substitution } p? = \alpha^\circ \cdot \langle p, id \rangle \right\}$$

$$\alpha^\circ \cdot \langle p, id \rangle \cdot f$$

$$= \left\{ \text{x-fusion} \right\}$$

$$\alpha^\circ \cdot \langle p \cdot f, id \cdot f \rangle$$

$$= \left\{ \text{natural-}id \text{ twice} \right\}$$

$$\alpha^\circ \cdot \langle id \cdot p \cdot f, f \cdot id \rangle$$

# Let us try it

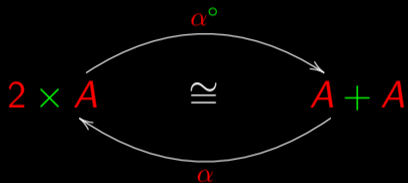
$$\begin{aligned} & p? \cdot f \\ = & \left\{ \text{substitution } p? = \alpha^\circ \cdot \langle p, id \rangle \right\} = \left\{ \times\text{-absorption} \right\} \\ & \alpha^\circ \cdot \langle p, id \rangle \cdot f \qquad \alpha^\circ \cdot (id \times f) \cdot \langle p \cdot f, id \rangle \\ = & \left\{ \times\text{-fusion} \right\} \\ & \alpha^\circ \cdot \langle p \cdot f, id \cdot f \rangle \\ = & \left\{ \text{natural-}id \text{ twice} \right\} \\ & \alpha^\circ \cdot \langle id \cdot p \cdot f, f \cdot id \rangle \end{aligned}$$

# Let us try it

$$p? \cdot f$$

$$\begin{aligned} &= \left\{ \text{substitution } p? = \alpha^\circ \cdot \langle p, id \rangle \right\} &= \left\{ \times\text{-absorption} \right\} \\ &\alpha^\circ \cdot \langle p, id \rangle \cdot f &\alpha^\circ \cdot (id \times f) \cdot \langle p \cdot f, id \rangle \\ &= \left\{ \times\text{-fusion} \right\} &= \left\{ ?? \right\} \\ &\alpha^\circ \cdot \langle p \cdot f, id \cdot f \rangle &?? \\ &= \left\{ \text{natural-}id \text{ twice} \right\} &= \left\{ ?? \right\} \\ &\alpha^\circ \cdot \langle id \cdot p \cdot f, f \cdot id \rangle &(f + f) \cdot (p \cdot f)? \end{aligned}$$

# Recall



$$\alpha = [\langle \underline{T}, id \rangle, \langle \underline{F}, id \rangle]$$

$$\alpha^\circ = \dots ?$$

Natural- $\alpha^\circ$

$$A + A \xleftarrow{\alpha^\circ} 2 \times A$$

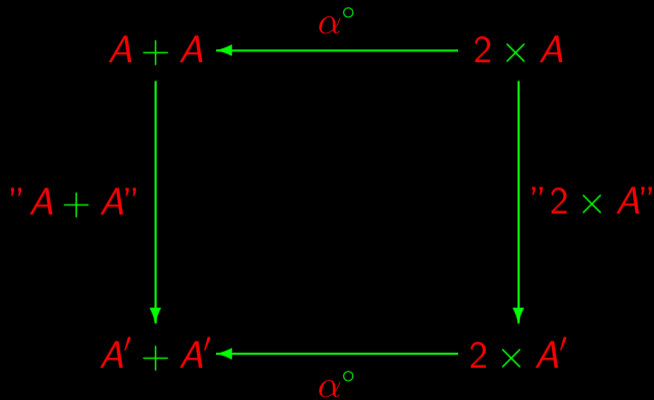
# Natural- $\alpha^\circ$

$$A + A \xleftarrow{\alpha^\circ} 2 \times A$$

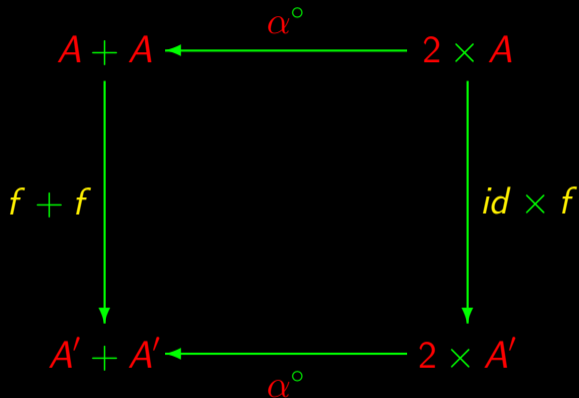
$$A' + A' \xleftarrow{\alpha^\circ} 2 \times A'$$



# Natural- $\alpha^\circ$



# Natural- $\alpha^\circ$



$$(f + f) \cdot \alpha^\circ = \alpha^\circ \cdot (id \times f)$$

... and this finishes the proof!

$$\begin{aligned} & p? \cdot f &= & \{ \text{x-absorption} \} \\ = & \{ p? = \alpha^\circ \cdot \langle p, id \rangle \} & \alpha^\circ \cdot (id \times f) \cdot \langle p \cdot f, id \rangle \\ & \alpha^\circ \cdot \langle p, id \rangle \cdot f \\ = & \{ \text{x-fusion} \} \\ & \alpha^\circ \cdot \langle p \cdot f, id \cdot f \rangle \\ = & \{ \text{natural-}id \text{ twice} \} \\ & \alpha^\circ \cdot \langle id \cdot p \cdot f, f \cdot id \rangle \end{aligned}$$

... and this finishes the proof!

$$\begin{aligned} & p? \cdot f & = & \{ \times\text{-absorption} \} \\ = & \{ p? = \alpha^\circ \cdot \langle p, id \rangle \} & \alpha^\circ \cdot (id \times f) \cdot \langle p \cdot f, id \rangle \\ & \alpha^\circ \cdot \langle p, id \rangle \cdot f & = & \{ \text{free thef } \alpha^\circ \} \\ = & \{ \times\text{-fusion} \} & (f + f) \cdot \alpha^\circ \cdot \langle p \cdot f, id \rangle \\ & \alpha^\circ \cdot \langle p \cdot f, id \cdot f \rangle \\ = & \{ \text{natural-}id \text{ twice} \} \\ & \alpha^\circ \cdot \langle id \cdot p \cdot f, f \cdot id \rangle \end{aligned}$$

... and this finishes the proof!

$$\begin{aligned} & p? \cdot f & = & \{ \text{x-absorption} \} \\ = & \{ p? = \alpha^\circ \cdot \langle p, id \rangle \} & \alpha^\circ \cdot (id \times f) \cdot \langle p \cdot f, id \rangle \\ & \alpha^\circ \cdot \langle p, id \rangle \cdot f & = & \{ \text{free thef } \alpha^\circ \} \\ = & \{ \text{x-fusion} \} & (f + f) \cdot \alpha^\circ \cdot \langle p \cdot f, id \rangle \\ & \alpha^\circ \cdot \langle p \cdot f, id \cdot f \rangle & = & \{ p? = \alpha^\circ \cdot \langle p, id \rangle \} \\ = & \{ \text{natural-id twice} \} & (f + f) \cdot (p \cdot f)? \\ & \alpha^\circ \cdot \langle id \cdot p \cdot f, f \cdot id \rangle \end{aligned}$$



# Cálculo de Programas

Class T06

# Notation variants

2 + 3

**infix** notation

operator between

## Notation variants

2 + 3

**infix** notation

operator between

+ 2 3

**prefix** notation

operator before



## Notation variants

2 + 3

**infix** notation

operator between

+ 2 3

**prefix** notation

operator before

2 3 +

**postfix** notation

operator after

## Notation variants

$2 + 3$	<b>infix</b> notation	operator between
$+ 2 3$	<b>prefix</b> notation	operator before
$2 3 +$	<b>postfix</b> notation	operator after
$+ (2, 3)$	<b>prefix</b> “uncurried”	grouped inputs

# Notation variants

$f\ a\ b$

**prefix** notation “curried”

separated inputs

# Notation variants

$f\ a\ b$       **prefix** notation “curried”      separated inputs

$f\ (a,\ b)$       **prefix** “uncurried”      grouped inputs

# “Curried”? “uncurried”

Terminology in honour of mathematician **Haskell Curry** (1900-1982) who developed the theory behind **functional programming** (1936).



# Example

$$\pi_1 : A \times B \rightarrow A$$

# Example

$$\pi_1 : A \times B \rightarrow A$$

$$\pi_1 (a, b) = a$$

# Example

$$\pi_1 : A \times B \rightarrow A$$

$$\pi_1 (a, b) = a$$

(notation: prefix “uncurried”)



## Exponentials

```
Prelude> p1(a,b)=a
```

```
Prelude> p1' a b = a
```

```
Prelude> :t p1
```

```
p1 :: (a, b) -> a
```

```
Prelude> :t p1'
```

```
p1' :: a -> b -> a
```

```
Prelude>
```

# In Haskell

$$\pi_1' = \text{curry } \pi_1$$

# In Haskell

$\pi_1' = \text{curry } \pi_1$

$\pi_1 = \text{uncurry } \pi_1'$

# In Haskell

$$\pi_1' = \text{curry } \pi_1$$
$$\pi_1 = \text{uncurry } \pi_1'$$

## Important:

- ▶ `curry` (e `uncurry` ) accept functions as **arguments**
- ▶ `curry` (e `uncurry` ) yield functions as **outputs**

# In Haskell

$$\pi_1' = \text{curry } \pi_1$$

$$\pi_1 = \text{uncurry } \pi_1'$$

## Important:

- ▶ `curry` (e `uncurry` ) accept functions as **arguments**
- ▶ `curry` (e `uncurry` ) yield functions as **outputs**

They are said to be

**higher order** functions.

# Curry | uncurry

`curry` ::  $((a, b) \rightarrow c) \rightarrow (a \rightarrow b \rightarrow c)$

`uncurry` ::  $(a \rightarrow b \rightarrow c) \rightarrow ((a, b) \rightarrow c)$

# Curry | uncurry

`curry` ::  $((a, b) \rightarrow c) \rightarrow (a \rightarrow b \rightarrow c)$

`uncurry` ::  $(a \rightarrow b \rightarrow c) \rightarrow ((a, b) \rightarrow c)$

# Curry | uncurry

curry ::  $((a, b) \rightarrow c) \rightarrow (a \rightarrow b \rightarrow c)$

uncurry ::  $(a \rightarrow b \rightarrow c) \rightarrow ((a, b) \rightarrow c)$



# Curry | uncurry

curry ::  $((a, b) \rightarrow c) \rightarrow (a \rightarrow b \rightarrow c)$

uncurry ::  $(a \rightarrow b \rightarrow c) \rightarrow ((a, b) \rightarrow c)$

each other inverses?

# Curry | uncurry

`curry` ::  $((a, b) \rightarrow c) \rightarrow a \rightarrow b \rightarrow c$

# Curry | uncurry

`curry` ::  $((a, b) \rightarrow c) \rightarrow a \rightarrow b \rightarrow c$

`curry`

## Curry | uncurry

`curry` ::  $((a, b) \rightarrow c) \rightarrow a \rightarrow b \rightarrow c$

`curry` *f*

## Curry | uncurry

`curry` ::  $((a, b) \rightarrow c) \rightarrow a \rightarrow b \rightarrow c$

`curry`  $f$   $a$

## Curry | uncurry

`curry` ::  $((a, b) \rightarrow c) \rightarrow a \rightarrow b \rightarrow c$

`curry` *f a b*

## Curry | uncurry

`curry` ::  $((a, b) \rightarrow c) \rightarrow a \rightarrow b \rightarrow c$

`curry`  $f$   $a$   $b$  =

## Curry | uncurry

`curry` ::  $((a, b) \rightarrow c) \rightarrow a \rightarrow b \rightarrow c$

`curry f a b = f (a, b)`



## Curry | uncurry

`curry` ::  $((a, b) \rightarrow c) \rightarrow a \rightarrow b \rightarrow c$

`curry f a b = f (a, b)`

`uncurry` ::  $(a \rightarrow b \rightarrow c) \rightarrow (a, b) \rightarrow c$

# Curry | uncurry

`curry` ::  $((a, b) \rightarrow c) \rightarrow a \rightarrow b \rightarrow c$

`curry f a b = f (a, b)`

`uncurry` ::  $(a \rightarrow b \rightarrow c) \rightarrow (a, b) \rightarrow c$

`uncurry`

## Curry | uncurry

$\text{curry} :: ((a, b) \rightarrow c) \rightarrow a \rightarrow b \rightarrow c$

$\text{curry } f \ a \ b = f \ (a, b)$

$\text{uncurry} :: (a \rightarrow b \rightarrow c) \rightarrow (a, b) \rightarrow c$

$\text{uncurry } g$

## Curry | uncurry

$\text{curry} :: ((a, b) \rightarrow c) \rightarrow a \rightarrow b \rightarrow c$

$\text{curry } f \ a \ b = f \ (a, b)$

$\text{uncurry} :: (a \rightarrow b \rightarrow c) \rightarrow (a, b) \rightarrow c$

$\text{uncurry } g \ (a, b)$

## Curry | uncurry

$\text{curry} :: ((a, b) \rightarrow c) \rightarrow a \rightarrow b \rightarrow c$

$\text{curry } f \ a \ b = f \ (a, b)$

$\text{uncurry} :: (a \rightarrow b \rightarrow c) \rightarrow (a, b) \rightarrow c$

$\text{uncurry } g \ (a, b) =$

## Curry | uncurry

$\text{curry} :: ((a, b) \rightarrow c) \rightarrow a \rightarrow b \rightarrow c$

$\text{curry } f \ a \ b = f \ (a, b)$

$\text{uncurry} :: (a \rightarrow b \rightarrow c) \rightarrow (a, b) \rightarrow c$

$\text{uncurry } g \ (a, b) = g \ a \ b$

## Curry | uncurry

$$\begin{cases} \text{curry} (\text{uncurry } f) = f? \\ \text{uncurry} (\text{curry } f) = f? \end{cases}$$

Curry | uncurry

$$\text{uncurry } g \ (a, b) = g \ a \ b$$



## Curry | uncurry

$$\text{uncurry } (\text{curry } f) (a, b) = \text{curry } f a b$$

$$\text{curry } f a b = f (a, b)$$

## Curry | uncurry

$$\text{uncurry} (\text{curry } f) (a, b) = f (a, b)$$

$$f \ x = g \ x \Leftrightarrow f = g$$

## Curry | uncurry

$$\text{uncurry} (\text{curry } f) = f$$

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

Curry | uncurry

$$(\text{uncurry} \cdot \text{curry}) f = f$$

$$\text{id } x = x$$

# Curry | uncurry

$$(\text{uncurry} \cdot \text{curry}) f = \text{id } f$$

$$f x = g x \Leftrightarrow f = g$$

Curry | uncurry

$$\text{uncurry} \cdot \text{curry} = \text{id}$$



Curry | uncurry

`curry (uncurry f) = f ?`

## Curry | uncurry

$$\text{curry } f \ a \ b = f \ (a, b)$$



## Curry | uncurry

$$\text{curry } (\text{uncurry } g) \ a \ b = \text{uncurry } g \ (a, b)$$

$$\text{uncurry } g \ (a, b) = g \ a \ b$$

## Curry | uncurry

$$\text{curry} (\text{uncurry } g) a b = g a b$$

$$f x = g x \Leftrightarrow f = g$$

## Curry | uncurry

$$\text{curry} (\text{uncurry } g) a = g a$$

$$f x = g x \Leftrightarrow f = g$$

## Curry | uncurry

$$\text{curry} (\text{uncurry } g) = g$$

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

Curry | uncurry

$$(\text{curry} \cdot \text{uncurry}) g = g$$

$$id \ x = x$$

## Curry | uncurry

$$(\text{curry} \cdot \text{uncurry}) g = \text{id } g$$

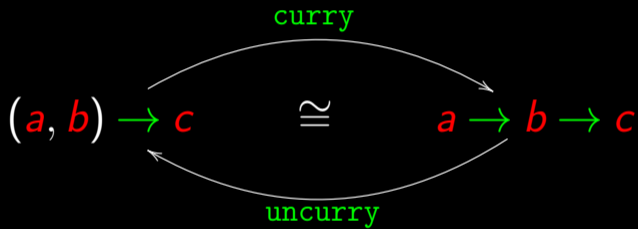
$$f x = g x \Leftrightarrow f = g$$

Curry | uncurry

$$\text{curry} \cdot \text{uncurry} = \text{id}$$



# Isomorphism





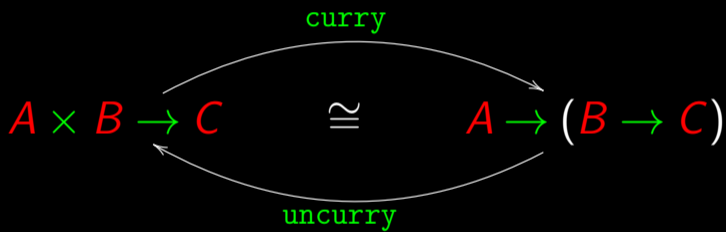
The type  $B^A$

The type  $B^A$

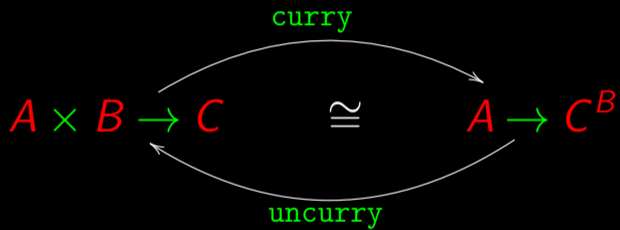
$$B^A = \{f \mid f : A \rightarrow B\}$$

NB: functions that map  $A$  to  $B$

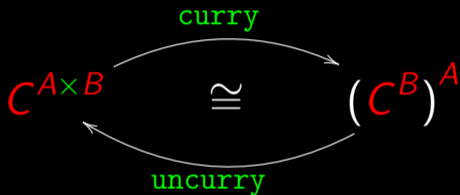
# Isomorphism



# Isomorphism



# Isomorphism



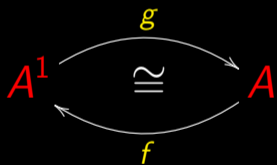
Cf. arithmetic exponentials:  $c^{a b} = (c^b)^a$

# More isomorphisms

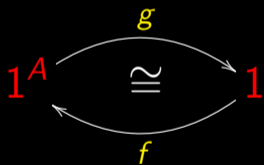
$$a^1 = a$$

$$1^a = 1$$

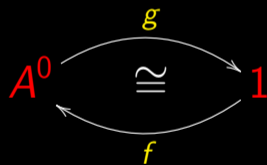
$$a^0 = 1$$



$$\begin{cases} f \ a = \underline{a} \\ g \ \underline{a} = a \end{cases}$$

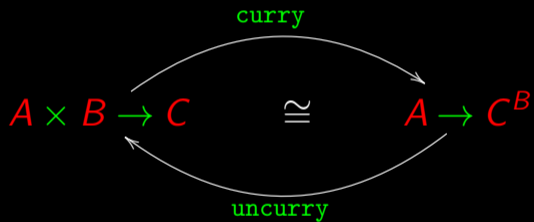


$$\begin{cases} f \ () = ! \\ g = ! \end{cases}$$



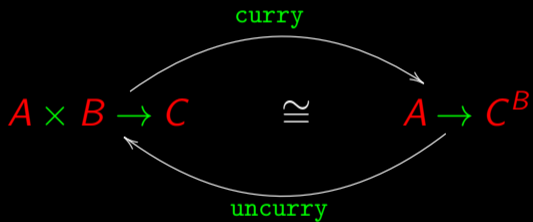
$$\begin{cases} f \ () = \perp \\ g = ! \end{cases}$$

# Curry | uncurry



# Curry | uncurry

$f$





# Curry | uncurry

$$f \xrightarrow{\text{curry}} k$$

$$A \times B \rightarrow C \quad \cong \quad A \rightarrow C^B$$

Diagram illustrating the relationship between the curried and uncurried forms of a function:

- The left side represents the uncurried function:  $A \times B \rightarrow C$ .
- The right side represents the curried function:  $A \rightarrow C^B$ .
- The two forms are shown to be isomorphic ( $\cong$ ).
- The top arrow is labeled "curry", indicating the transformation from the uncurried form to the curried form.
- The bottom arrow is labeled "uncurry", indicating the transformation from the curried form back to the uncurried form.

# Curry | uncurry

$$f \xrightarrow{\text{curry}} k$$

$$A \times B \rightarrow C \xrightleftharpoons[\text{uncurry}]{\text{curry}} A \rightarrow C^B$$

$k$

# Curry | uncurry

$$f \xrightarrow{\text{curry}} k$$

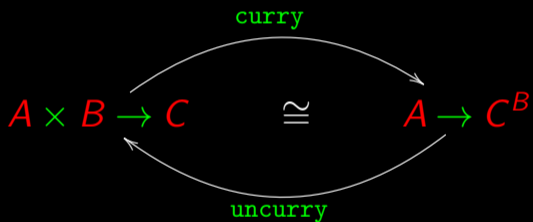
$$A \times B \rightarrow C \quad \cong \quad A \rightarrow C^B$$

curry (top arrow), uncurry (bottom arrow)

$$f \xleftarrow{\text{uncurry}} k$$

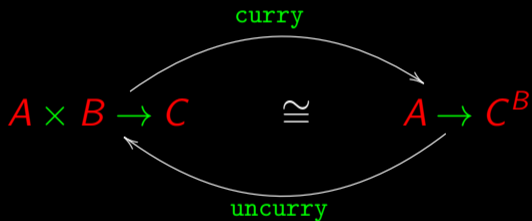
# Curry | uncurry

$$k = \text{curry } f$$



$$\text{uncurry } k = f$$

# Curry | uncurry



$$k = \text{curry } f \iff \text{uncurry } k = f$$

# Curry | uncurry

$$k = \text{curry } f$$

$\Leftrightarrow$  { isomorphism (last slide) }

$$\text{uncurry } k = f$$

# Curry | uncurry

$$k = \text{curry } f$$

$\Leftrightarrow$  { isomorphism (last slide) }

$$\text{uncurry } k = f$$

$\Leftrightarrow$  { add variables }

$$\text{uncurry } k(a, b) = f(a, b)$$

# Curry | uncurry

$$(k\ a)\ b = f\ (a,\ b)$$

$$k = \text{curry}\ f$$

$\Leftrightarrow$  { isomorphism (last slide) }

$$\text{uncurry}\ k = f$$

$\Leftrightarrow$  { add variables }

$$\text{uncurry}\ k\ (a,\ b) = f\ (a,\ b)$$

$\Leftrightarrow$  { definition of **uncurry**  $k$  }

$$k\ a\ b = f\ (a,\ b)$$



# Curry | uncurry

$$k = \text{curry } f$$

$$\Leftrightarrow \{ \text{isomorphism (last slide)} \}$$

$$\text{uncurry } k = f$$

$$\Leftrightarrow \{ \text{add variables} \}$$

$$\text{uncurry } k (a, b) = f (a, b)$$

$$\Leftrightarrow \{ \text{definition of uncurry } k \}$$

$$k a b = f (a, b)$$

$$(k a) b = f (a, b)$$

$$\Leftrightarrow \{ \text{introduce } ap (f, x) = f x \}$$

$$ap (k a, b) = f (a, b)$$

# Curry | uncurry

$$k = \text{curry } f$$
$$\Leftrightarrow \{ \text{isomorphism (last slide)} \}$$

$$\text{uncurry } k = f$$

$$\Leftrightarrow \{ \text{add variables} \}$$

$$\text{uncurry } k (a, b) = f (a, b)$$

$$\Leftrightarrow \{ \text{definition of uncurry } k \}$$

$$k a b = f (a, b)$$

$$(k a) b = f (a, b)$$

$$\Leftrightarrow \{ \text{introduce } ap (f, x) = f x \}$$

$$ap (k a, b) = f (a, b)$$

$$\Leftrightarrow \{ \text{add } id \}$$

$$ap (k a, id b) = f (a, b)$$

# Curry | uncurry

$$k = \text{curry } f$$

$$\Leftrightarrow \{ \text{isomorphism (last slide)} \}$$

$$\text{uncurry } k = f$$

$$\Leftrightarrow \{ \text{add variables} \}$$

$$\text{uncurry } k (a, b) = f (a, b)$$

$$\Leftrightarrow \{ \text{definition of uncurry } k \}$$

$$k a b = f (a, b)$$

$$(k a) b = f (a, b)$$

$$\Leftrightarrow \{ \text{introduce } ap (f, x) = f x \}$$

$$ap (k a, b) = f (a, b)$$

$$\Leftrightarrow \{ \text{add } id \}$$

$$ap (k a, id b) = f (a, b)$$

$$\Leftrightarrow \{ \text{composition and } \times \}$$

$$(ap \cdot (k \times id)) (a, b) = f (a, b)$$

# Curry | uncurry

$k = \text{curry } f$

$\Leftrightarrow$  { isomorphism (last slide) }

$\text{uncurry } k = f$

$\Leftrightarrow$  { add variables }

$\text{uncurry } k (a, b) = f (a, b)$

$\Leftrightarrow$  { definition of  $\text{uncurry } k$  }

$k a b = f (a, b)$

$(k a) b = f (a, b)$

$\Leftrightarrow$  { introduce  $ap (f, x) = f x$  }

$ap (k a, b) = f (a, b)$

$\Leftrightarrow$  { add  $id$  }

$ap (k a, id b) = f (a, b)$

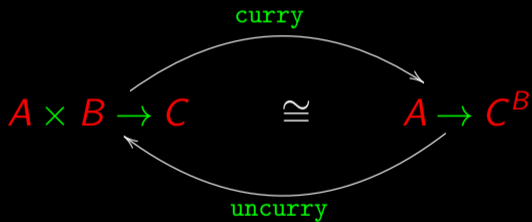
$\Leftrightarrow$  { composition and  $\times$  }

$(ap \cdot (k \times id)) (a, b) = f (a, b)$

$\Leftrightarrow$  { drop variables }

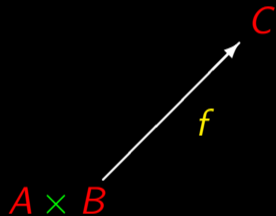
$ap \cdot (k \times id) = f$

# Universal property

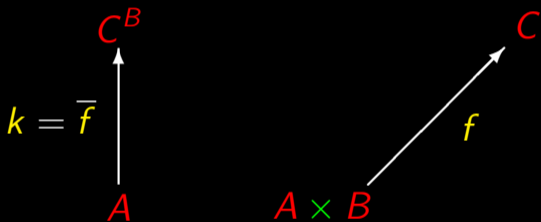


$$k = \text{curry } f \iff \text{ap} \cdot (k \times \text{id}) = f$$

# Diagrams

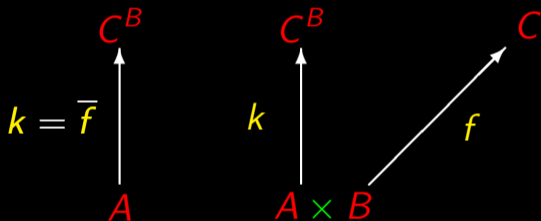


# Diagrams



$\bar{f}$  abbreviates `curry f`

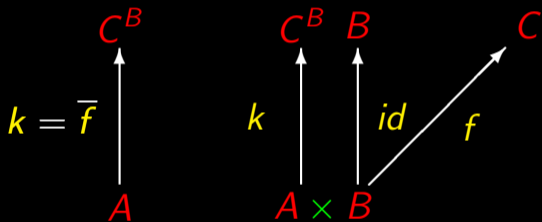
# Diagrams



$\bar{f}$  abbreviates `curry f`

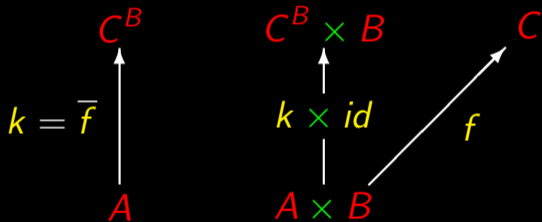


# Diagrams

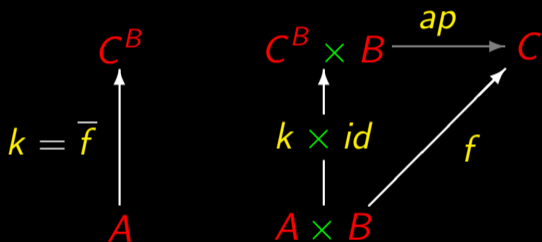


$\bar{f}$  abbreviates `curry f`

# Universal property

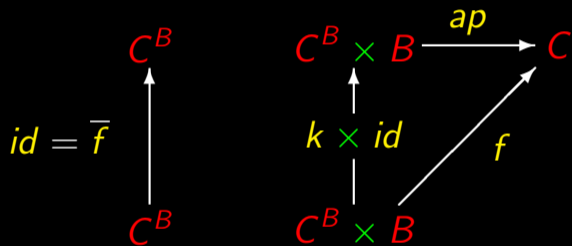


# Universal property



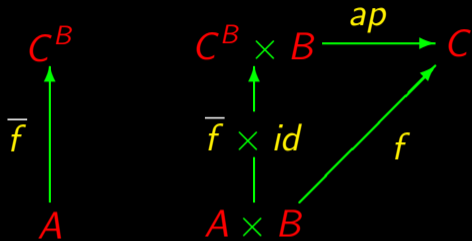
$$k = \bar{f} \Leftrightarrow ap \cdot (k \times id) = f$$

# Reflexion ( $k := id$ )

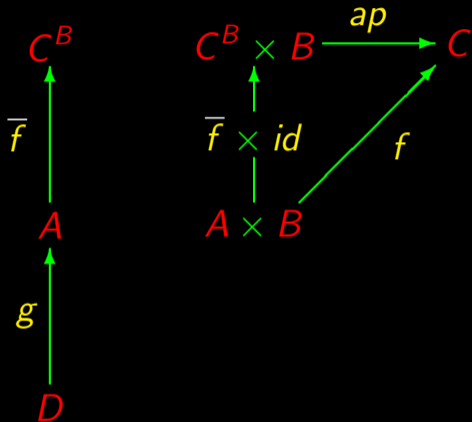


$$id = \bar{f} \Leftrightarrow ap = f$$

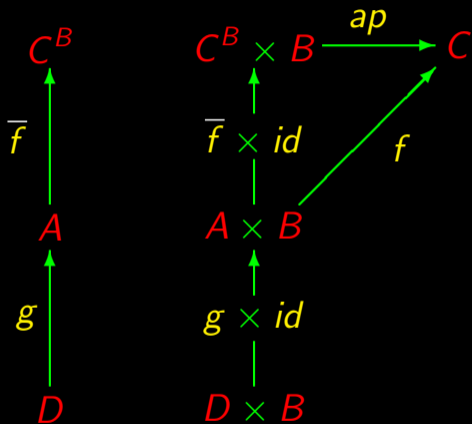
# Fusion



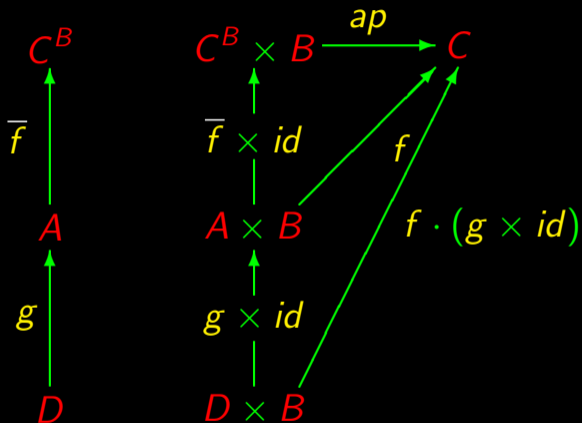
# Fusion



# Fusion

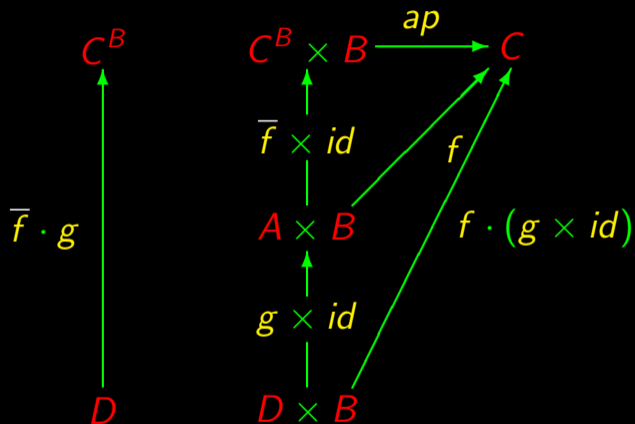


# Fusion

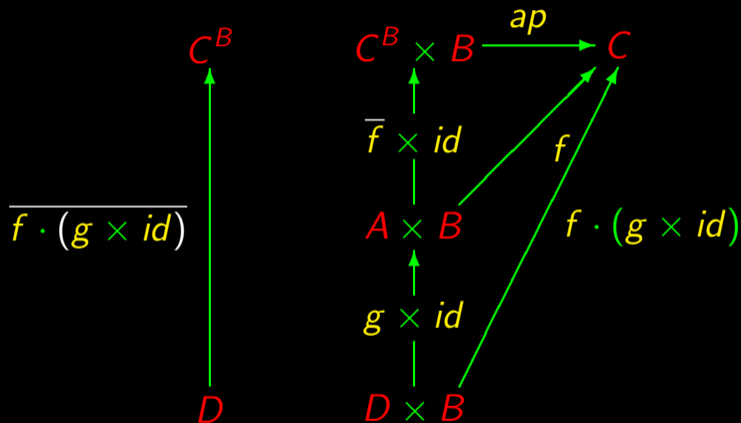




# Fusion



# Fusion



$$\overline{f} \cdot g = \overline{f \cdot (g \times id)}$$

# Curry | uncurry

$$f \xrightarrow{\text{curry}} k$$

$$A \times B \rightarrow C \xrightleftharpoons[\text{uncurry}]{\text{curry}} A \rightarrow C^B$$

$$f \xleftarrow{\text{uncurry}} k$$

# Curry | uncurry

$$f \dashv \xrightarrow{\text{curry}} k$$

Abbreviations:

$$\text{curry } f = \bar{f}$$

$$A \times B \rightarrow C \cong A \rightarrow C^B$$

Diagram illustrating the relationship between the two forms of a function:

- The left side is  $A \times B \rightarrow C$ .
- The right side is  $A \rightarrow C^B$ .
- The two forms are related by an isomorphism  $\cong$ .
- The top arrow is labeled **curry**.
- The bottom arrow is labeled **uncurry**.

$$f \dashv \xleftarrow{\text{uncurry}} k$$

# Curry | uncurry

$$f \dashv \xrightarrow{\text{curry}} k$$

$$A \times B \rightarrow C \xrightleftharpoons[\text{uncurry}]{\text{curry}} A \rightarrow C^B$$

$$f \xleftarrow{\text{uncurry}} k \dashv$$

Abbreviations:

$$\text{curry } f = \bar{f}$$

$$\text{uncurry } f = \hat{f}$$

# Curry | uncurry

$$f \xrightarrow{\text{curry}} k$$

$$A \times B \rightarrow C \xrightleftharpoons[\text{uncurry}]{\text{curry}} A \rightarrow C^B$$

$$f \xleftarrow{\text{uncurry}} k$$

Abbreviations:

$$\text{curry } f = \bar{f}$$

$$\text{uncurry } f = \hat{f}$$

Isomorphisms:

$$f = g \Leftrightarrow \bar{f} = \bar{g}$$

$$k = h \Leftrightarrow \hat{k} = \hat{h}$$

# Summary

$f \cdot g$

sequential

# Summary

$f \cdot g$

sequential

$\langle f, g \rangle$



# Summary

$$f \cdot g$$

sequential

$$\langle f, g \rangle, f \times g$$

parallel

# Summary

$$f \cdot g$$

sequential

$$\langle f, g \rangle, f \times g$$

parallel

$$[f, g]$$

# Summary

$$f \cdot g$$

sequential

$$\langle f, g \rangle, f \times g$$

parallel

$$[f, g], f + g$$

# Summary

$f \cdot g$

sequential

$\langle f, g \rangle, f \times g$

parallel

$[f, g], f + g, p \rightarrow f, g$

alternation

# Summary

$f \cdot g$

sequential

$\langle f, g \rangle, f \times g$

parallel

$[f, g], f + g, p \rightarrow f, g$

alternation

$\bar{f}$

# Summary

$f \cdot g$

sequential

$\langle f, g \rangle, f \times g$

parallel

$[f, g], f + g, p \rightarrow f, g$

alternation

$\bar{f}, \text{exp } f$

higher order

# Summary

$f \cdot g$

sequential

$\langle f, g \rangle, f \times g$

parallel

$[f, g], f + g, p \rightarrow f, g$

alternation

$\bar{f}, \text{exp } f$

higher order

**Compositional programming** 😊

# Summary

$A \times B$

records (“structs”)



# Summary

$A \times B$

records (“structs”)

$A + B$

variant records (“unions”)

# Summary

$A \times B$

records (“structs”)

$A + B$

variant records (“unions”)

$1 + A$

“pointers”

# Summary

$A \times B$

records (“structs”)

$A + B$

variant records (“unions”)

$1 + A$

“pointers”

$B^A$

“arrays”, “streams”

# Summary

$A \times B$

records (“structs”)

$A + B$

variant records (“unions”)

$1 + A$

“pointers”

$B^A$

“arrays”, “streams”

**data structuring** 😊

# Summary

$A \times B$

records (“structs”)

$A + B$

variant records (“unions”)

$1 + A$

“pointers”

$B^A$

“arrays”, “streams”

**data structuring** 😊

$1 + A \times X^2$

polynomial structures

# Part II

Where do programs come from?

What **is** a program?

Where do programs come from?

The **algorithms** we know were born of what?



# Where do programs come from?

The **algorithms** we know were born of what?

For instance, **quicksort**?

(See [video 05b](#), t=3.25)

# Where do programs come from?

$$\left\{ \begin{array}{l} a \times 0 = 0 \\ a \times 1 = a \\ a \times (b + c) = a \times b + a \times c \end{array} \right.$$

# Where do programs come from?

$$\left\{ \begin{array}{l} a \times 0 = 0 \\ a \times 1 = a \\ a \times (b + 1) = a \times b + a \times 1 \end{array} \right.$$

# Where do programs come from?

$$\left\{ \begin{array}{l} a \times 0 = 0 \\ a \times 1 = a \\ a \times (b + 1) = a \times b + a \end{array} \right.$$

# Where do programs come from?

$$\left\{ \begin{array}{l} a \times 0 = 0 \\ a \times 1 = a \\ a \times (0 + 1) = a \times 0 + a \end{array} \right.$$

# Where do programs come from?

$$\left\{ \begin{array}{l} a \times 0 = 0 \\ a \times 1 = a \\ a \times 1 = a \times 0 + a \end{array} \right.$$

# Where do programs come from?

$$\left\{ \begin{array}{l} a \times 0 = 0 \\ a \times 1 = a \\ a \times 1 = 0 + a \end{array} \right.$$

# Where do programs come from?

$$\left\{ \begin{array}{l} a \times 0 = 0 \\ a \times 1 = a \\ a \times 1 = a \end{array} \right.$$



# Where do programs come from?

$$\left\{ \begin{array}{l} a \times 0 = 0 \\ a \times 1 = a \\ a \times (b + 1) = a \times b + a \end{array} \right.$$

# Where do programs come from?

$$\left\{ \begin{array}{l} a \times 0 = 0 \\ a \times (b + 1) = a \times b + a \end{array} \right.$$

# Where do programs come from?

$$\begin{cases} a \times 0 = 0 \\ a \times (b + 1) = a \times b + a \end{cases}$$

What **is** a program?

$$\left\{ \begin{array}{l} a \times 0 = 0 \\ a \times (b + 1) = a \times b + a \end{array} \right.$$

What **is** a program?

$$\left\{ \begin{array}{l} a \times 0 = 0 \\ a \times (b + 1) = a + a \times b \end{array} \right.$$

What **is** a program?

$$\left\{ \begin{array}{l} (a \times) 0 = 0 \\ (a \times) (b + 1) = a + (a \times) b \end{array} \right.$$

What **is** a program?

$$\left\{ \begin{array}{l} (a \times) 0 = 0 \\ (a \times) (b + 1) = (a +) ((a \times) b) \end{array} \right.$$

What **is** a program?

$$\left\{ \begin{array}{l} (a \times) (\underline{0} x) = \underline{0} x \\ (a \times) (b + 1) = (a +) ((a \times) b) \end{array} \right.$$



What **is** a program?

$$\left\{ \begin{array}{l} (a \times) (\underline{0} x) = \underline{0} x \\ (a \times) (\text{succ } b) = (a+) ((a \times) b) \end{array} \right.$$

What **is** a program?

$$\left\{ \begin{array}{l} ((a \times) \cdot \underline{0}) x = \underline{0} x \\ (a \times) (\text{succ } b) = (a+) ((a \times) b) \end{array} \right.$$

What **is** a program?

$$\left\{ \begin{array}{l} ((a \times) \cdot \underline{0}) x = \underline{0} x \\ ((a \times) \cdot \text{succ}) b = (a+) ((a \times) b) \end{array} \right.$$

What **is** a program?

$$\left\{ \begin{array}{l} ((a \times) \cdot \underline{0}) x = \underline{0} x \\ ((a \times) \cdot \text{succ}) b = ((a +) \cdot (a \times)) b \end{array} \right.$$

What **is** a program?

$$\left\{ \begin{array}{l} (a \times) \cdot \underline{0} = \underline{0} \\ ((a \times) \cdot \text{succ}) b = ((a +) \cdot (a \times)) b \end{array} \right.$$

What **is** a program?

$$\left\{ \begin{array}{l} (a \times) \cdot \underline{0} = \underline{0} \\ (a \times) \cdot \text{succ} = (a+) \cdot (a \times) \end{array} \right.$$

What **is** a program?

$$\left\{ \begin{array}{l} (a \times) \cdot \underline{0} = \underline{0} \\ (a \times) \cdot \text{succ} = (a+) \cdot (a \times) \end{array} \right.$$

What **is** a program?

$$[(a \times) \cdot \underline{0}, (a \times) \cdot \text{succ}] = [\underline{0}, (a +) \cdot (a \times)]$$



What **is** a program?

$$(a \times) \cdot [\underline{0}, succ] = [\underline{0}, (a+) \cdot (a \times)]$$

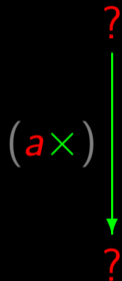
What **is** a program?

$$(a \times) \cdot [\underline{0}, succ] = [\underline{0} \cdot id, (a+) \cdot (a \times)]$$

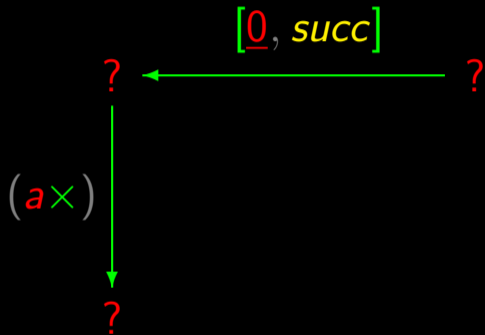
What **is** a program?

$$(a \times) \cdot [\underline{0}, succ] = [\underline{0}, (a+)] \cdot (id + (a \times))$$

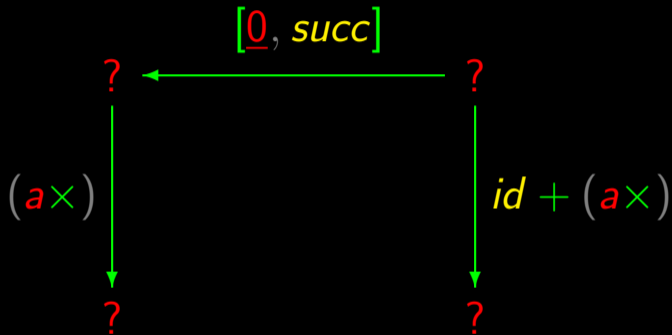
$$(a \times) \cdot [\underline{0}, succ] = [\underline{0}, (a+)] \cdot (id + (a \times))$$



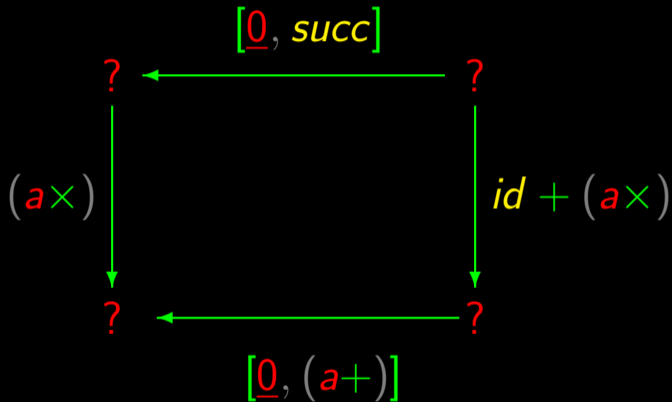
$$(a \times) \cdot [\underline{0}, succ] = [\underline{0}, (a+)] \cdot (id + (a \times))$$



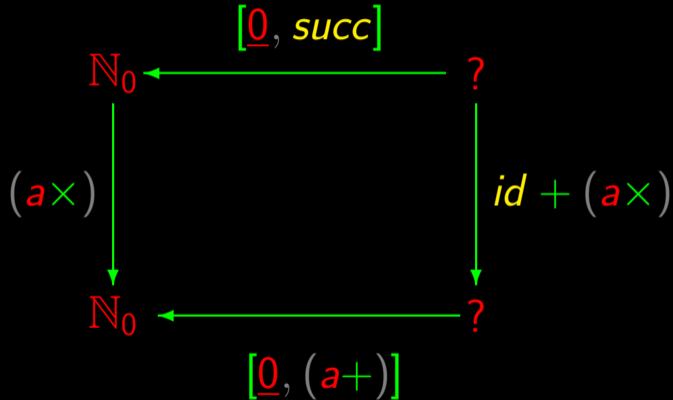
$$(a \times) \cdot [\underline{0}, succ] = [\underline{0}, (a+)] \cdot (id + (a \times))$$



$$(a \times) \cdot [\underline{0}, succ] = [\underline{0}, (a+)] \cdot (id + (a \times))$$

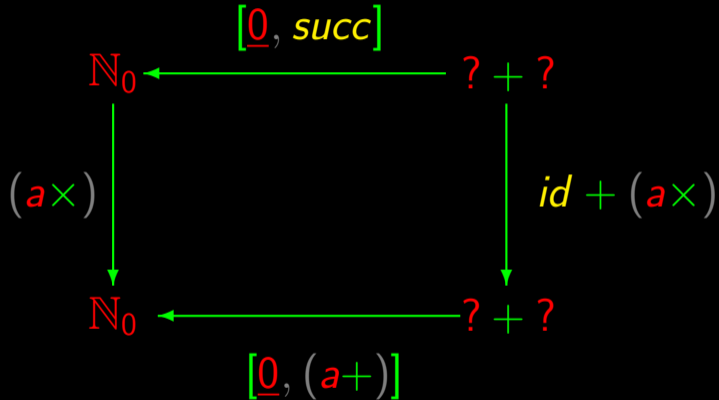


A program is a “rectangle”

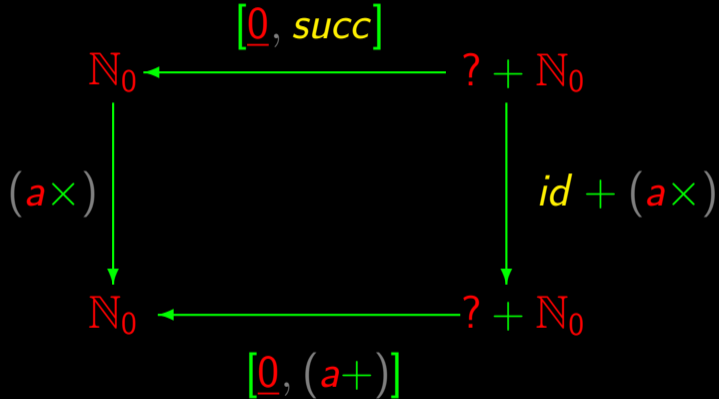




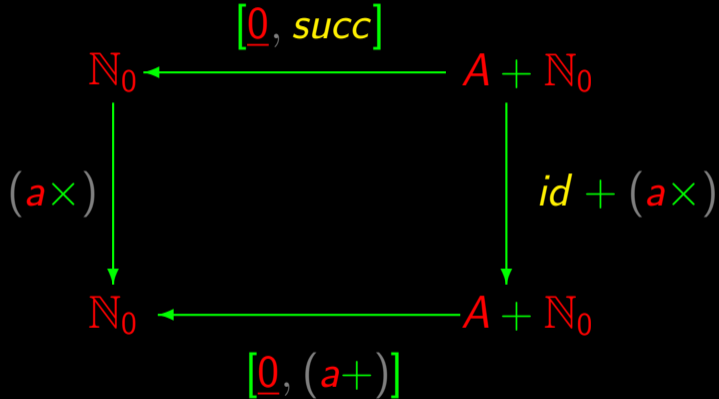
A program is a “rectangle”



# A program is a “rectangle”



A program is a “rectangle”



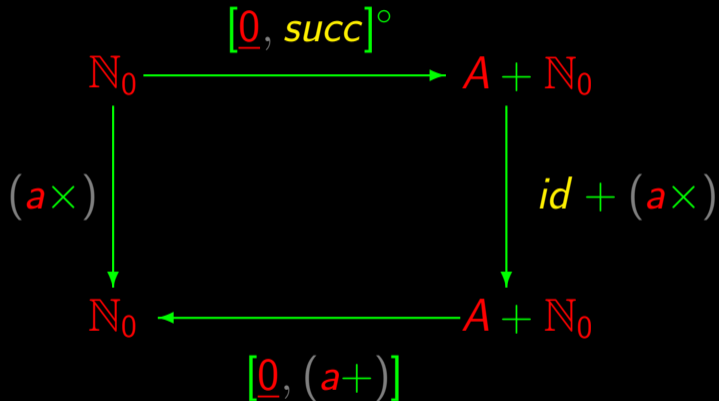
Does  $[\underline{0}, \text{succ}]^\circ$  exist?

$$\begin{array}{ccc} \mathbb{N}_0 & \xrightarrow{[\underline{0}, \text{succ}]^\circ} & A + \mathbb{N}_0 \\ \downarrow (a \times) & & \downarrow \text{id} + (a \times) \\ \mathbb{N}_0 & \xleftarrow{[\underline{0}, (a+)]} & A + \mathbb{N}_0 \end{array}$$

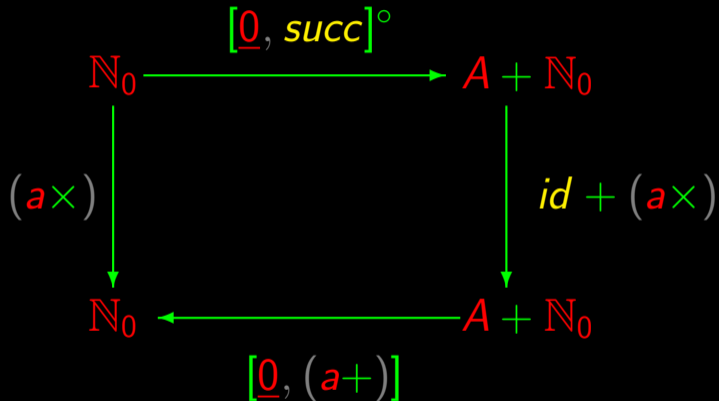
Does  $[\underline{0}, \text{succ}]^\circ$  exist?

$$\begin{array}{ccc} N_0 & \xleftarrow{[\underline{0}, \text{succ}]} & A + N_0 \\ \downarrow (a \times) & & \downarrow \text{id} + (a \times) \\ N_0 & \xleftarrow{[\underline{0}, (a+)]} & A + N_0 \end{array}$$

Does  $[\underline{0}, \text{succ}]^\circ$  exist?



Does  $[\underline{0}, \text{succ}]^\circ$  exist?



$$(a \times) = [\underline{0}, (a+)] \cdot (id + (a \times)) \cdot [\underline{0}, \text{succ}]^\circ(?)$$

Does  $[\underline{0}, succ]^\circ$  exist?

Conjecture:  $out : \mathbb{N}_0 \rightarrow A + \mathbb{N}_0$  exists and is  $[\underline{0}, succ]^\circ$

$$out \cdot [\underline{0}, succ] = id$$



Does  $[\underline{0}, succ]^\circ$  exist?

Conjecture:  $out : \mathbb{N}_0 \rightarrow A + \mathbb{N}_0$  exists and is  $[\underline{0}, succ]^\circ$

$$out \cdot [\underline{0}, succ] = id$$

$$\Leftrightarrow \{ \text{+-fusion} \}$$

$$[out \cdot \underline{0}, out \cdot succ] = id$$

Does  $[\underline{0}, succ]^\circ$  exist?

Conjecture:  $out : \mathbb{N}_0 \rightarrow A + \mathbb{N}_0$  exists and is  $[\underline{0}, succ]^\circ$

$$out \cdot [\underline{0}, succ] = id$$

$$\Leftrightarrow \{ \text{+-fusion} \}$$

$$[out \cdot \underline{0}, out \cdot succ] = id$$

$$\Leftrightarrow \{ \text{universal-+} \}$$

$$\begin{cases} out \cdot \underline{0} = i_1 \\ out \cdot succ = i_2 \end{cases}$$

Does  $[\underline{0}, succ]^\circ$  exist?

$$\begin{cases} \mathbf{out} \cdot \underline{0} = i_1 \\ \mathbf{out} \cdot succ = i_2 \end{cases}$$

Does  $[\underline{0}, succ]^\circ$  exist?

$$\begin{cases} \mathbf{out} \cdot \underline{0} = i_1 \\ \mathbf{out} \cdot succ = i_2 \end{cases}$$

$\Leftrightarrow$  { introduce variables }

$$\begin{cases} \mathbf{out} (\underline{0} \ x) = i_1 \ x \\ \mathbf{out} (succ \ x) = i_2 \ x \end{cases}$$

Does  $[\underline{0}, succ]^\circ$  exist?

$$\begin{cases} \mathbf{out} \cdot \underline{0} = i_1 \\ \mathbf{out} \cdot succ = i_2 \end{cases}$$

$$\Leftrightarrow \quad \{ \text{introduce variables} \}$$

$$\begin{cases} \mathbf{out} (\underline{0} \ x) = i_1 \ x \\ \mathbf{out} (succ \ x) = i_2 \ x \end{cases}$$

$$\Leftrightarrow \quad \{ \text{simplificar} \}$$

$$\begin{cases} \mathbf{out} \ 0 = i_1 \ x \\ \mathbf{out} (x + 1) = i_2 \ x \end{cases}$$

Does  $[\underline{0}, succ]^\circ$  exist?

$$\begin{cases} \text{out} \cdot \underline{0} = i_1 \\ \text{out} \cdot succ = i_2 \end{cases}$$

$\Leftrightarrow$  { introduce variables }

$$\begin{cases} \text{out} (\underline{0} x) = i_1 x \\ \text{out} (succ x) = i_2 x \end{cases}$$

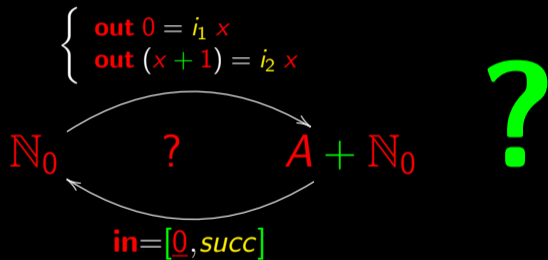
$$\begin{cases} \text{out } 0 = i_1 x \\ \text{out} (x + 1) = i_2 x \end{cases}$$

?

$\Leftrightarrow$  { simplificar }

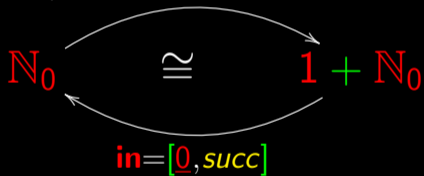
$$\begin{cases} \text{out } 0 = i_1 x \\ \text{out} (x + 1) = i_2 x \end{cases}$$

# Problem



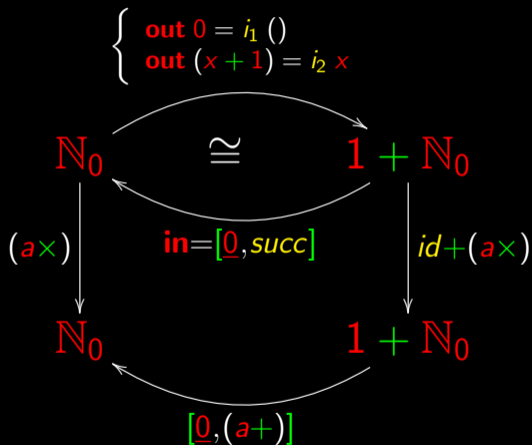
# Solution

$$\begin{cases} \text{out } 0 = i_1 () \\ \text{out } (x + 1) = i_2 x \end{cases} \quad \text{😊}$$

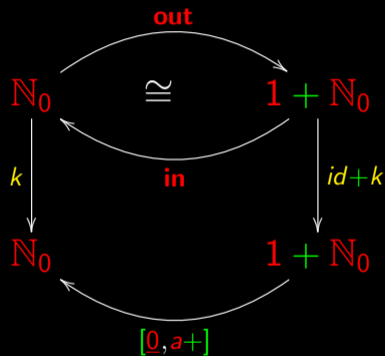




# Solution



# Diagrama

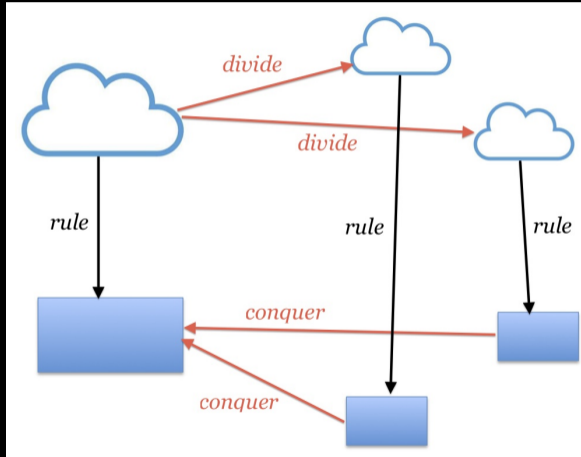


$$(a \times) = k \text{ where } k = [0, (a+)] \cdot (id + k) \cdot \text{out}$$

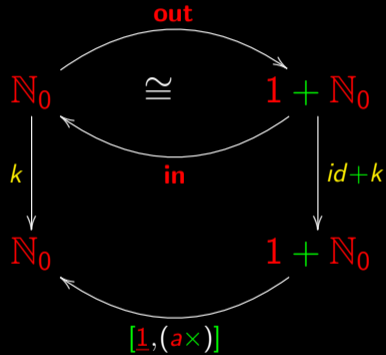
# Cálculo de Programas

Class T07

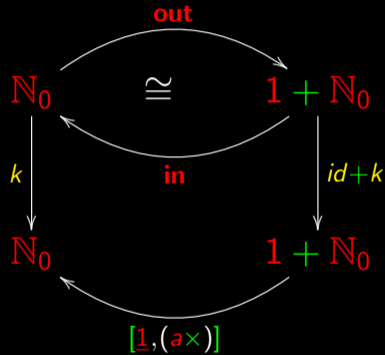
# Anticipation of what is to come...



# Variations in diagram



# Variations in diagram

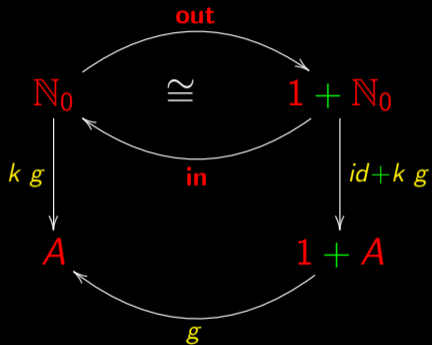


$$a^b = k \text{ where } k = [1, (a \times)] \cdot (id + k) \cdot out$$

# Generalizing the diagram

$$\begin{cases} a^0 = 1 \\ a^{b+1} = a \times a^b \end{cases}$$

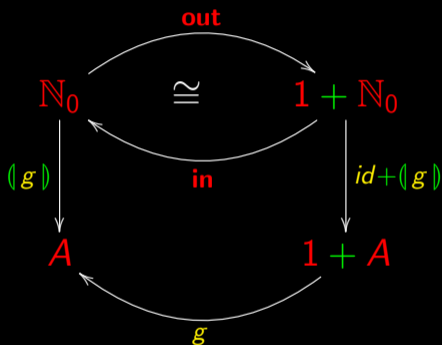
# Generalizing the diagram



$$kg = g \cdot (id + kg) \cdot \mathbf{out}$$



# Generalizing the diagram



$$(g) = g \cdot (id + (g)) \cdot \mathbf{out}$$

# Catamorphism

$(g)$

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

# Catamorphism

$(g)$

$\underbrace{\text{cata}}_{\text{downwards}} (\kappa\alpha\tau\alpha) + \underbrace{\text{morphism}}_{\text{transformation}} (\mu\omicron\rho\phi\iota\sigma\mu\omicron\zeta)$

“Downwards transformation”

# Definition

$$\begin{array}{ccc} N_0 & \xrightarrow{\text{out}} & 1 + N_0 \\ \downarrow (g) & & \downarrow id + (g) \\ A & \xleftarrow{g} & 1 + A \end{array}$$

$$(g) = g \cdot (id + (g)) \cdot \text{out}$$

# Property

$$\begin{array}{ccc} N_0 & \xleftarrow{\text{in}} & 1 + N_0 \\ \downarrow (g) & & \downarrow id + (g) \\ A & \xleftarrow{g} & 1 + A \end{array}$$

$$(g) \cdot \text{in} = g \cdot (id + (g))$$

# Property

$$\begin{array}{ccc} \mathbb{N}_0 & \xleftarrow{\text{in}} & 1 + \mathbb{N}_0 \\ \downarrow (g) & & \downarrow id + (g) \\ A & \xleftarrow{g} & 1 + A \end{array}$$

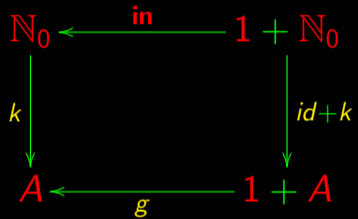
$$k = (g) \Rightarrow k \cdot \text{in} = g \cdot (id + k)$$

# Uniqueness

$$\begin{array}{ccc} \mathbb{N}_0 & \xleftarrow{\text{in}} & 1 + \mathbb{N}_0 \\ \downarrow (g) & & \downarrow id + (g) \\ A & \xleftarrow{g} & 1 + A \end{array}$$

$$k = (g) \Leftarrow k \cdot \text{in} = g \cdot (id + k)$$

# Universal property



$$k = \langle g \rangle \Leftrightarrow k \cdot \mathbf{in} = g \cdot (id + k)$$



Universal- $\times$ :

$$k = \langle f, g \rangle \Leftrightarrow \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases}$$

Universal- $\times$ :

$$k = \langle f, g \rangle \Leftrightarrow \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases}$$

Universal- $+$ :

$$k = [f, g] \Leftrightarrow \begin{cases} k \cdot i_1 = f \\ k \cdot i_2 = g \end{cases}$$

Universal- $\times$ :

$$k = \langle f, g \rangle \Leftrightarrow \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases}$$

Universal- $+$ :

$$k = [f, g] \Leftrightarrow \begin{cases} k \cdot i_1 = f \\ k \cdot i_2 = g \end{cases}$$

Universal-expo:

$$k = \bar{f} \Leftrightarrow ap \cdot (k \times id) = f$$

Universal- $\times$ :

$$k = \langle f, g \rangle \Leftrightarrow \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases}$$

Universal- $+$ :

$$k = [f, g] \Leftrightarrow \begin{cases} k \cdot i_1 = f \\ k \cdot i_2 = g \end{cases}$$

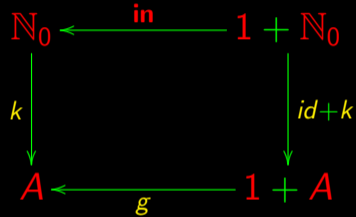
Universal-expo:

$$k = \bar{f} \Leftrightarrow ap \cdot (k \times id) = f$$

Cata-Universal:

$$k = \llbracket g \rrbracket \Leftrightarrow k \cdot \mathbf{in} = g \cdot (id + k)$$

# Cata-universal



$$k = \langle g \rangle \Leftrightarrow k \cdot \mathbf{in} = g \cdot (id + k)$$

# Cata-cancellation

$$k = \langle g \rangle \Leftrightarrow k \cdot \mathbf{in} = g \cdot (id + k)$$

# Cata-cancellation

$$\langle g \rangle = \langle g \rangle \Leftrightarrow \langle g \rangle \cdot \mathbf{in} = g \cdot (\mathit{id} + \langle g \rangle)$$

# Cata-cancellation

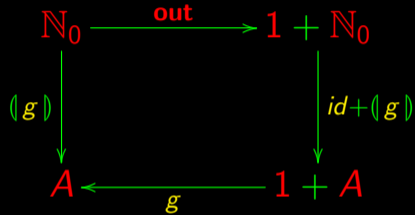
$$\langle g \rangle \cdot \mathbf{in} = g \cdot (id + \langle g \rangle)$$



# Cata-definition

$$\langle g \rangle = g \cdot (id + \langle g \rangle) \cdot \mathbf{out}$$

# Cata-definition



$$(g) = g \cdot (id + (g)) \cdot \mathbf{out}$$

# Cata-reflexion

$$k = \mathit{fold} \ g \ \Rightarrow k \cdot \mathit{in} = g \cdot (\mathit{id} + k)$$

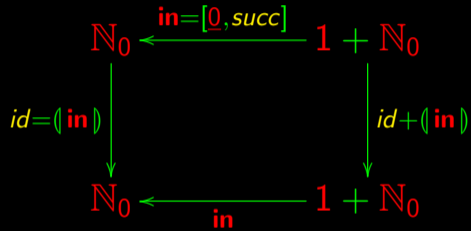
# Cata-reflexion

$$id = (g) \Leftrightarrow id \cdot \mathbf{in} = g \cdot (id + id)$$

# Cata-reflexion

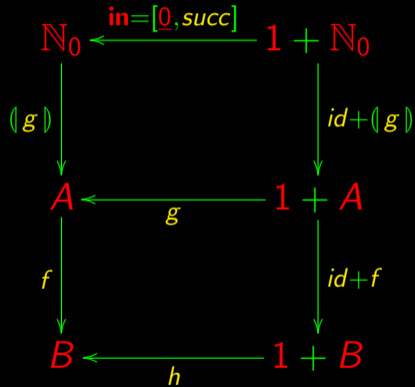
$$id = (g) \Leftrightarrow in = g$$

# Cata-reflexion

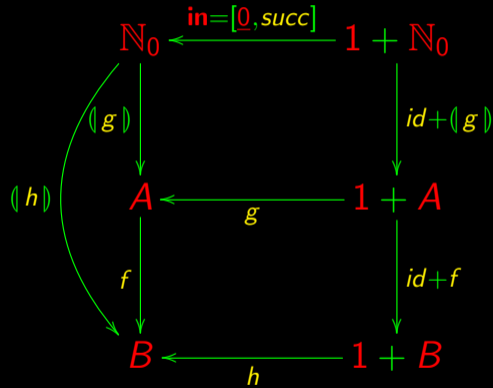


$$(\mathbf{in}) = id$$

# Cata-fusion



# Cata-fusion





# Cata-fusion

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

# Cata-fusion

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

$$\Leftrightarrow \{ \text{Cata-universal for } k = f \cdot (g) \}$$

$$f \cdot (g) \cdot \mathbf{in} = h \cdot (id + f \cdot (g))$$

# Cata-fusion

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

$$\Leftrightarrow \{ \text{Cata-universal for } k = f \cdot (g) \}$$

$$f \cdot (g) \cdot \mathbf{in} = h \cdot (id + f \cdot (g))$$

$$\Leftrightarrow \{ \text{Cata-cancellation} \}$$

$$f \cdot g \cdot (id + (g)) = h \cdot (id + f \cdot (g))$$

# Cata-fusion

$$f \cdot g \cdot (id + \langle g \rangle) = h \cdot (id \cdot id + f \cdot \langle g \rangle)$$

# Cata-fusion

$$f \cdot g \cdot (id + \langle\langle g \rangle\rangle) = h \cdot (id \cdot id + f \cdot \langle\langle g \rangle\rangle)$$

$$\Leftrightarrow \{ \text{functor-+} \}$$

$$f \cdot g \cdot (id + \langle\langle g \rangle\rangle) = h \cdot (id + f) \cdot (id + \langle\langle g \rangle\rangle)$$

# Cata-fusion

$$f \cdot g \cdot (id + \langle\langle g \rangle\rangle) = h \cdot (id \cdot id + f \cdot \langle\langle g \rangle\rangle)$$

$$\Leftrightarrow \{ \text{functor-+} \}$$

$$f \cdot g \cdot (id + \langle\langle g \rangle\rangle) = h \cdot (id + f) \cdot (id + \langle\langle g \rangle\rangle)$$

$$\Leftarrow \{ \text{Leibniz} \}$$

$$f \cdot g = h \cdot (id + f)$$

# Cata-fusion

$$f \cdot g \cdot (id + \llbracket g \rrbracket) = h \cdot (id \cdot id + f \cdot \llbracket g \rrbracket)$$

$$\Leftrightarrow \{ \text{functor-+} \}$$

$$f \cdot g \cdot (id + \llbracket g \rrbracket) = h \cdot (id + f) \cdot (id + \llbracket g \rrbracket)$$

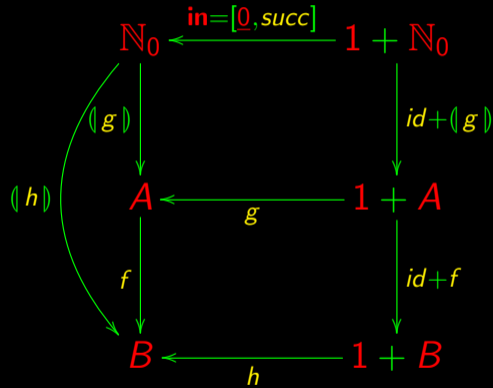
$$\Leftarrow \{ \text{Leibniz} \}$$

$$f \cdot g = h \cdot (id + f)$$

Summing up:

$$f \cdot \llbracket g \rrbracket = \llbracket h \rrbracket \Leftarrow f \cdot g = h \cdot (id + f)$$

# Cata-fusion





What **is** a program?

What **is** a program?

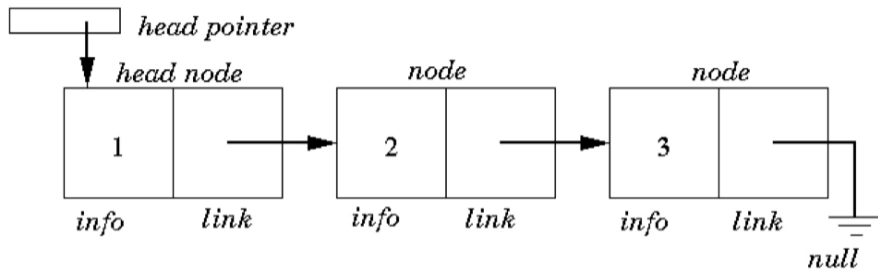
Programs are “rectangles” 😊

What **is** a program?

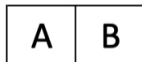
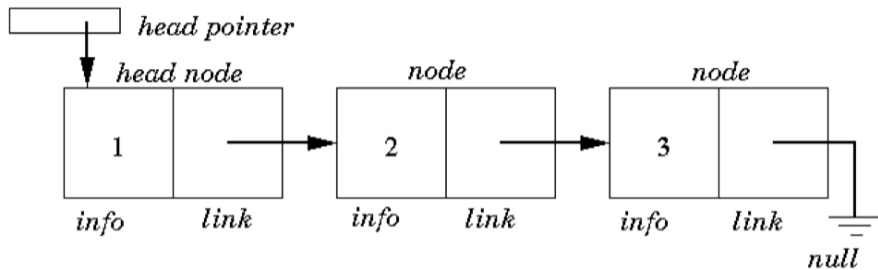
Programs are “rectangles” 😊

But there is a long way to the general case...

# Linked list

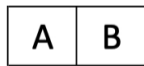
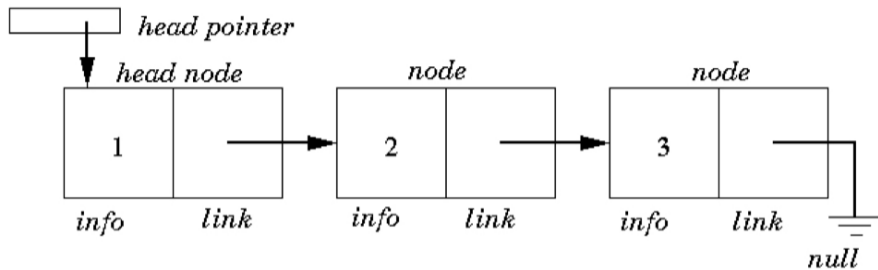


# Linked list

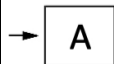


$$A \times B$$

# Linked list

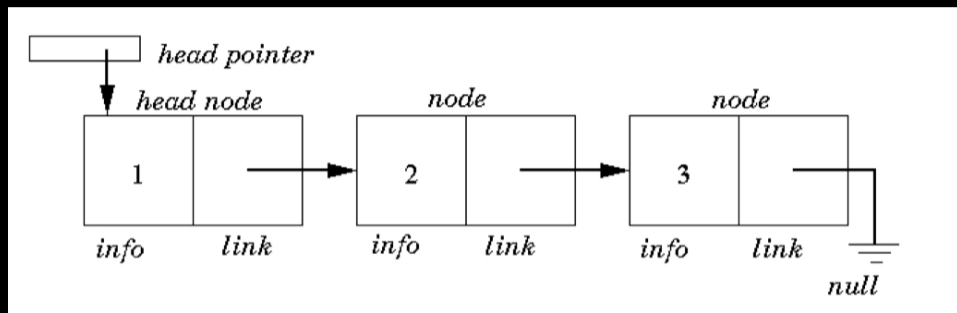


$$A \times B$$



$$1 + A$$

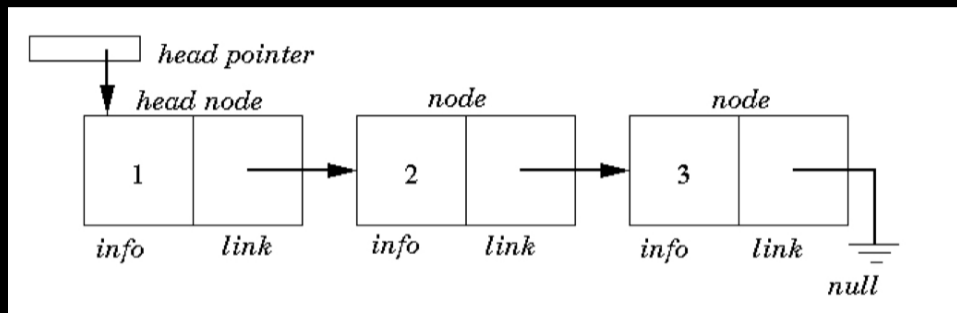
# Linked list



“pointer”

$$L = 1 + N$$

# Linked list



“pointer”

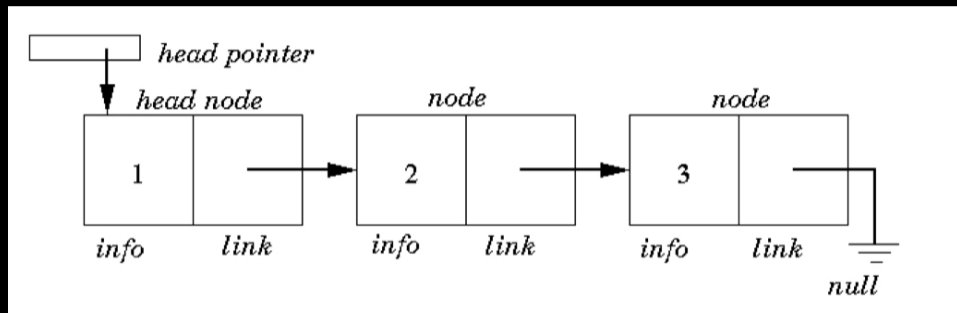
$$L = 1 + N$$

“node”

$$N = N_0 \times L$$



# Linked list

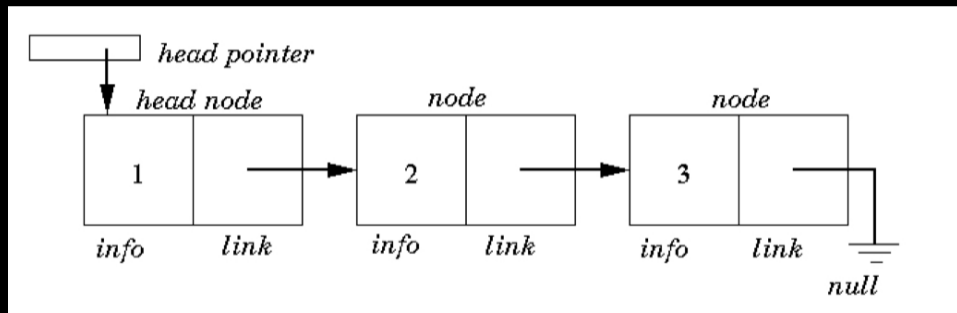


Substitution

$$\begin{cases} L = 1 + N \\ N = N_0 \times L \end{cases}$$

$$L = 1 + N_0 \times L$$

# Linked list



In fact

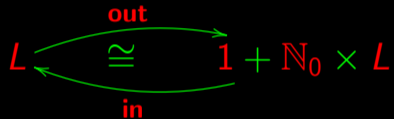
$$L \cong 1 + \mathbb{N}_0 \times L$$

# Linked list

$$L \cong 1 + \mathbb{N}_0 \times L$$

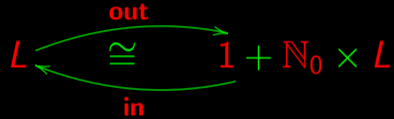
# Linked list

$$L \cong 1 + \mathbb{N}_0 \times L$$



# Linked list

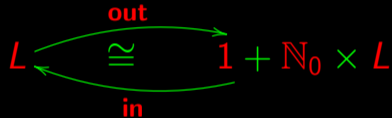
$$L \cong 1 + \mathbb{N}_0 \times L$$



$$\mathbf{in} = [\mathit{nil}, \mathit{cons}]$$

# Linked list

$$L \cong 1 + \mathbb{N}_0 \times L$$



$$\mathbf{in} = [nil, cons] \quad \begin{cases} nil \_ = [] \\ cons(a, l) = a : l \end{cases}$$

# Linked list

$$L \cong 1 + \mathbb{N}_0 \times L$$

$$\begin{array}{ccc} L & \xrightarrow{\text{out}} & 1 + \mathbb{N}_0 \times L \\ & \cong & \\ & \xleftarrow{\text{in}} & \end{array}$$

$$\mathbf{in} = [\mathit{nil}, \mathit{cons}] \quad \begin{cases} \mathit{nil} \_ = [] \\ \mathit{cons} (a, l) = a : l \end{cases}$$

$$\mathbf{out} \cdot \mathbf{in} = \mathit{id}$$

# Linked list

$$L \begin{array}{c} \xrightarrow{\text{out}} \\ \cong \\ \xleftarrow{\text{in}} \end{array} 1 + \mathbb{N}_0 \times L$$

$$\begin{cases} \text{out} \cdot \text{in} = id \\ \text{in} = [nil, cons] \end{cases}$$

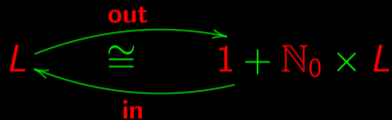


# Linked list

$$L \begin{array}{c} \xrightarrow{\text{out}} \\ \cong \\ \xleftarrow{\text{in}} \end{array} 1 + \mathbb{N}_0 \times L$$

$$\left\{ \begin{array}{l} \text{out} \cdot \text{in} = id \\ \text{in} = [nil, cons] \end{array} \right. \Rightarrow$$

# Linked list



$$\begin{cases} \text{out} \cdot \text{in} = id \\ \text{in} = [nil, cons] \end{cases} \Rightarrow \begin{cases} \text{out} [] = i_1 () \\ \text{out} (a : x) = i_2 (a, x) \end{cases}$$

## Length of a list

Recall list **concatenation**  $x ++ y$

## Length of a list

Recall list **concatenation**  $x ++ y$  such that

$$[] ++ x = x$$

## Length of a list

Recall list **concatenation**  $x ++ y$  such that

$$[] ++ x = x$$

$$[a] ++ x = a : x$$

## Length of a list

Recall list **concatenation**  $x ++ y$  such that

$$[] ++ x = x$$

$$[a] ++ x = a : x$$

Properties:

## Length of a list

Recall list **concatenation**  $x ++ y$  such that

$$[] ++ x = x$$

$$[a] ++ x = a : x$$

Properties:

$$\mathit{length} [] = 0$$

## Length of a list

Recall list **concatenation**  $x ++ y$  such that

$$[] ++ x = x$$

$$[a] ++ x = a : x$$

Properties:

$$\mathit{length} [] = 0$$

$$\mathit{length} [a] = 1$$



## Length of a list

Recall list **concatenation**  $x ++ y$  such that

$$[] ++ x = x$$

$$[a] ++ x = a : x$$

Properties:

$$\mathit{length} [] = 0$$

$$\mathit{length} [a] = 1$$

$$\mathit{length} (x ++ y) = \mathit{length} x + \mathit{length} y$$

# Length of a list

$$\mathit{length} [] = 0$$

$$\mathit{length} [a] = 1$$

$$\mathit{length} ([a] ++ y) = \mathit{length} [a] + \mathit{length} y$$

## Length of a list

$$\text{length } [] = 0$$

$$\text{length } [a] = 1$$

$$\text{length } ([a] ++ y) = \text{length } [a] + \text{length } y$$

$$\text{length } (a : y) = 1 + \text{length } y$$



## Length of a list

$$\text{length } [] = 0$$

$$\text{length } (a : y) = 1 + \text{length } y$$

# Length of a list

$$\text{length } [] = 0$$

$$\text{length } (a : y) = 1 + \text{length } y$$

$$(\text{length} \cdot \text{nil}) \_ = \underline{0} \_$$

$$(\text{length} \cdot \text{cons}) (a, y) = 1 + \text{length } (\pi_2 (a, y))$$

## Length of a list

$$\text{length } [] = 0$$

$$\text{length } (a : y) = 1 + \text{length } y$$

$$(\text{length} \cdot \text{nil}) \_ = \underline{0} \_$$

$$(\text{length} \cdot \text{cons}) (a, y) = 1 + \text{length } (\pi_2 (a, y))$$

$$\text{length} \cdot \text{nil} = \underline{0}$$

$$\text{length} \cdot \text{cons} = \text{succ} \cdot \text{length} \cdot \pi_2$$

## Length of a list

$$\text{length} \cdot \text{nil} = \underline{0}$$

$$\text{length} \cdot \text{cons} = \text{succ} \cdot \text{length} \cdot \pi_2$$



## Length of a list

$$\text{length} \cdot \text{nil} = \underline{0}$$

$$\text{length} \cdot \text{cons} = \text{succ} \cdot \text{length} \cdot \pi_2$$

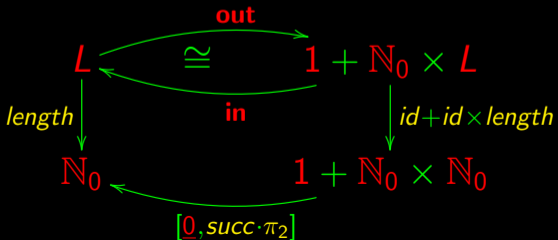
$$\text{length} \cdot [\text{nil}, \text{cons}] = [\underline{0}, \text{succ} \cdot \pi_2] \cdot (\text{id} + \text{id} \times \text{length})$$

# Length of a list

$$\text{length} \cdot \text{nil} = \underline{0}$$

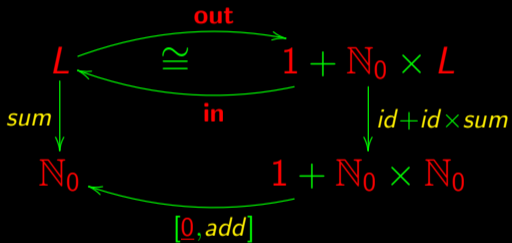
$$\text{length} \cdot \text{cons} = \text{succ} \cdot \text{length} \cdot \pi_2$$

$$\text{length} \cdot [\text{nil}, \text{cons}] = [\underline{0}, \text{succ} \cdot \pi_2] \cdot (\text{id} + \text{id} \times \text{length})$$



$$\left\{ \begin{array}{l} \text{in} = [\text{nil}, \text{cons}] \\ \text{out} = \text{in}^\circ \end{array} \right.$$

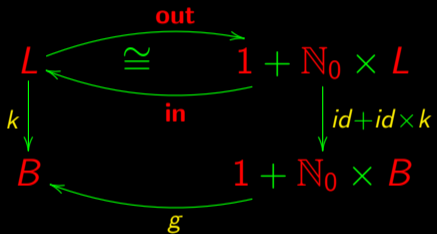
# List sum



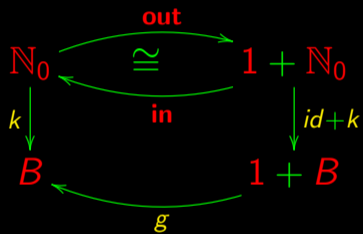
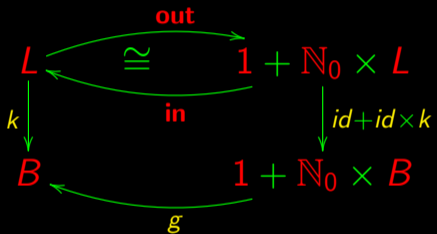
$$\begin{cases} \mathbf{in} = [\mathit{nil}, \mathit{cons}] \\ \mathbf{out} = \mathbf{in}^\circ \end{cases}$$

$$\text{sum} \cdot \mathbf{in} = [\underline{0}, \text{add}] \cdot (\text{id} + \text{id} \times \text{sum})$$

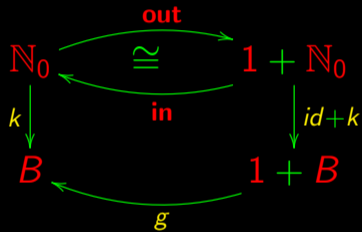
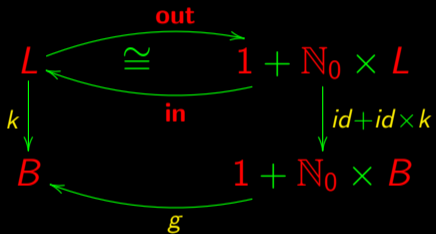
# Generalizing



# Generalizing

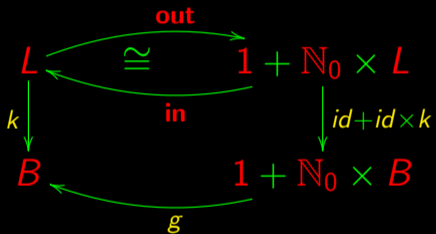


# Generalizing

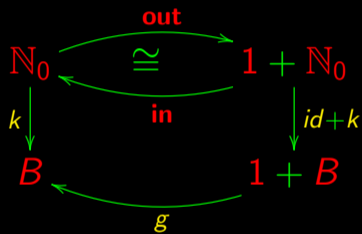


$$k \cdot in = g \cdot \underbrace{id + id \times k}_{F k}$$

# Generalizing

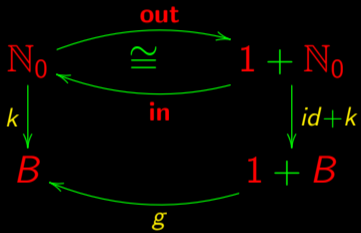


$$k \cdot in = g \cdot \underbrace{id + id \times k}_{F k}$$



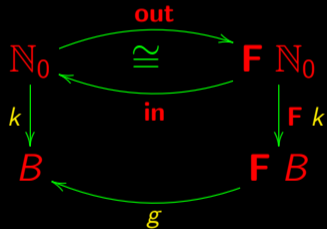
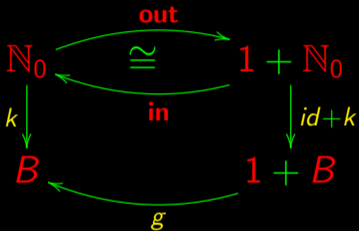
$$k \cdot in = g \cdot \underbrace{id + k}_{F k}$$

# Abstraction ( $\mathbb{N}_0$ )



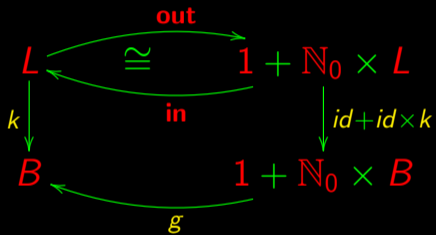


# Abstraction ( $\mathbb{N}_0$ )

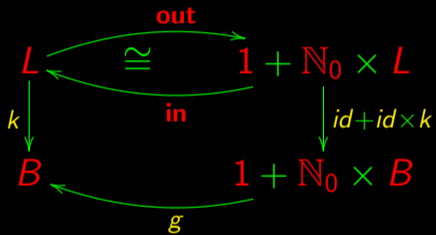


$$\begin{cases} \mathbf{F} k = id + k \\ \mathbf{F} X = 1 + X \end{cases}$$

# Abstraction (lists)

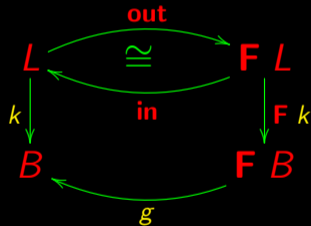
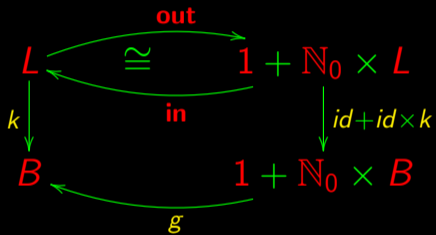


# Abstraction (lists)



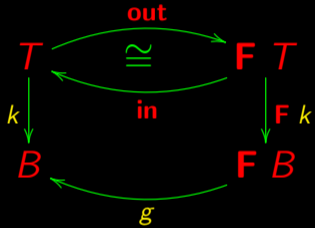
$$\begin{cases} \mathbf{F} k = id + id \times k \\ \mathbf{F} X = 1 + \mathbb{N}_0 \times X \end{cases}$$

# Abstraction (lists)

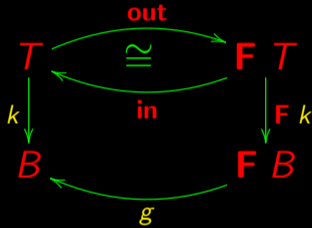


$$\begin{cases} \mathbf{F} k = id + id \times k \\ \mathbf{F} X = 1 + \mathbb{N}_0 \times X \end{cases}$$

# Abstraction

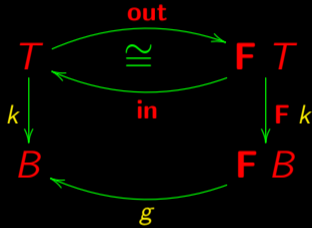


# Abstraction



$$k \cdot in = g \cdot F k$$

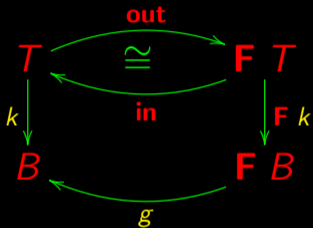
# Abstraction



$$k = \langle \langle g \rangle \rangle$$

$$k \cdot \text{in} = g \cdot F k$$

# Abstraction



$$k = \langle \langle g \rangle \rangle$$

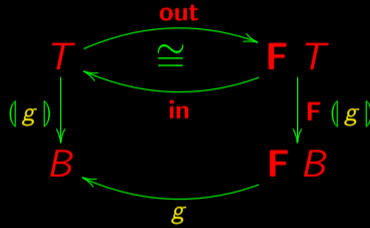
$$k \cdot \text{in} = g \cdot \mathbf{F} k$$

Read  $k = \langle \langle g \rangle \rangle$  as before:

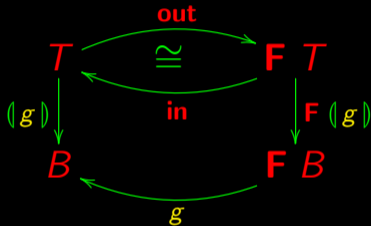
$k$  is the *catamorphism* of  $g$ .



# Catamorphism



# Catamorphism



Universal property:

$$k = (g) \iff k \cdot in = g \cdot F k$$

# Summary

$$\text{Natural numbers: } \left\{ \begin{array}{l} T = \mathbb{N}_0 \\ \text{in} = [\underline{0}, \text{succ}] \\ \left\{ \begin{array}{l} \mathbf{F} X = 1 + X \\ \mathbf{F} f = \text{id} + f \end{array} \right. \end{array} \right.$$

# Summary

$$\text{Natural numbers: } \left\{ \begin{array}{l} T = \mathbb{N}_0 \\ \text{in} = [0, \text{succ}] \\ \left\{ \begin{array}{l} \mathbf{F} X = 1 + X \\ \mathbf{F} f = \text{id} + f \end{array} \right. \end{array} \right. \quad \text{for } b \ i = ([i, b])$$

# Summary

$$\text{Natural numbers: } \left\{ \begin{array}{l} T = \mathbb{N}_0 \\ \mathbf{in} = [\underline{0}, \mathit{succ}] \\ \left\{ \begin{array}{l} \mathbf{F} X = 1 + X \\ \mathbf{F} f = \mathit{id} + f \end{array} \right. \end{array} \right. \quad \text{for } b \ i = ([\underline{i}, b])$$

$$\text{Lists: } \left\{ \begin{array}{l} T = L \\ \mathbf{in} = [\mathit{nil}, \mathit{cons}] \\ \left\{ \begin{array}{l} \mathbf{F} X = 1 + \mathbb{N}_0 \times X \\ \mathbf{F} f = \mathit{id} + \mathit{id} \times f \end{array} \right. \end{array} \right.$$

# Summary

$$\text{Natural numbers: } \left\{ \begin{array}{l} T = \mathbb{N}_0 \\ \mathbf{in} = [\underline{0}, \mathit{succ}] \\ \left\{ \begin{array}{l} \mathbf{F} X = 1 + X \\ \mathbf{F} f = \mathit{id} + f \end{array} \right. \end{array} \right. \quad \text{for } b \ i = ([\underline{i}, b])$$

$$\text{Lists: } \left\{ \begin{array}{l} T = L \\ \mathbf{in} = [\mathit{nil}, \mathit{cons}] \\ \left\{ \begin{array}{l} \mathbf{F} X = 1 + \mathbb{N}_0 \times X \\ \mathbf{F} f = \mathit{id} + \mathit{id} \times f \end{array} \right. \end{array} \right. \quad \text{foldr } f \ i = ([\underline{i}, \hat{f}])$$

## Examples already seen

Natural numbers:

$$(a \times) = ([\underline{0}, (a+)])$$

$$a^b = ([\underline{1}, (a \times)])^b$$

Lists:

$$sum = ([\underline{0}, add])$$

$$length = ([\underline{0}, succ \cdot \pi_2])$$

# Cata-cancellation

In

$$k = \langle g \rangle \Leftrightarrow k \cdot \mathbf{in} = g \cdot \mathbf{F} k$$

make  $k := \langle g \rangle$ :



# Cata-cancellation

In

$$k = \langle g \rangle \Leftrightarrow k \cdot \mathbf{in} = g \cdot \mathbf{F} k$$

make  $k := \langle g \rangle$ :

$$\langle g \rangle = \langle g \rangle \Leftrightarrow \langle g \rangle \cdot \mathbf{in} = g \cdot \mathbf{F} \langle g \rangle$$

# Cata-cancellation

In

$$k = (g) \Leftrightarrow k \cdot \mathbf{in} = g \cdot \mathbf{F} k$$

make  $k := (g)$ :

$$\begin{aligned} (g) = (g) &\Leftrightarrow (g) \cdot \mathbf{in} = g \cdot \mathbf{F} (g) \\ \Leftrightarrow &\{ x = x \Leftrightarrow \mathit{true} \} \\ \mathit{true} &\Leftrightarrow (g) \cdot \mathbf{in} = g \cdot \mathbf{F} (g) \end{aligned}$$

# Cata-cancellation

In

$$k = \langle g \rangle \Leftrightarrow k \cdot \mathbf{in} = g \cdot \mathbf{F} k$$

make  $k := \langle g \rangle$ :

$$\langle g \rangle = \langle g \rangle \Leftrightarrow \langle g \rangle \cdot \mathbf{in} = g \cdot \mathbf{F} \langle g \rangle$$

$$\Leftrightarrow \{ x = x \Leftrightarrow \text{true} \}$$

$$\text{true} \Leftrightarrow \langle g \rangle \cdot \mathbf{in} = g \cdot \mathbf{F} \langle g \rangle$$

$$\Leftrightarrow \{ \text{trivial} \}$$

$$\langle g \rangle \cdot \mathbf{in} = g \cdot \mathbf{F} \langle g \rangle$$

# Cata-cancellation

$$(g) \cdot \mathbf{in} = g \cdot \mathbf{F} (g)$$

$$\Leftrightarrow \{ \text{isomorphism } \mathbf{in} / \mathbf{out} \}$$

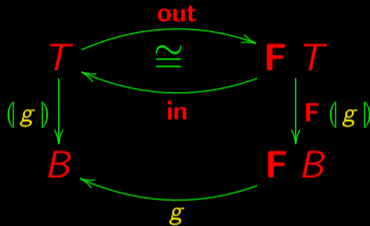
$$(g) = g \cdot \mathbf{F} (g) \cdot \mathbf{out}$$

# Cata-cancellation

$$(g) \cdot \mathbf{in} = g \cdot \mathbf{F} (g)$$

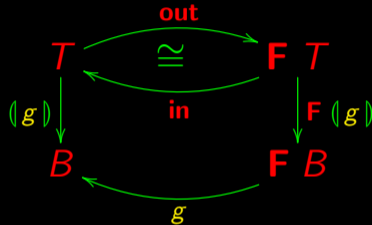
$$\Leftrightarrow \{ \text{isomorphism } \mathbf{in} / \mathbf{out} \}$$

$$(g) = g \cdot \mathbf{F} (g) \cdot \mathbf{out}$$



# Cata-cancellation

$$\begin{aligned} & \langle g \rangle \cdot \mathbf{in} = g \cdot \mathbf{F} \langle g \rangle \\ \Leftrightarrow & \quad \{ \text{isomorphism } \mathbf{in} / \mathbf{out} \} \\ & \langle g \rangle = g \cdot \mathbf{F} \langle g \rangle \cdot \mathbf{out} \end{aligned}$$



$$\langle g \rangle = g \cdot \mathbf{F} \langle g \rangle \cdot \mathbf{out}$$

executable definition

# Cálculo de Programas

Class T08

# Functors

What does **F** mean, exactly?

$$\begin{array}{ccc} A & \cdots & \mathbf{F} A \\ \downarrow f & & \downarrow \mathbf{F} f \\ B & \cdots & \mathbf{F} B \end{array}$$



# Functors

What does **F** mean, exactly?

$$\begin{array}{ccc} A & \cdots & \mathbf{F} A \\ \downarrow f & & \downarrow \mathbf{F} f \\ B & \cdots & \mathbf{F} B \end{array}$$

Recall **natural** (free)  
**properties**

# Functors

What does **F** mean, exactly?

$$\begin{array}{ccc} A & \xrightarrow{\quad} & \mathbf{F} A \\ \downarrow f & & \downarrow \mathbf{F} f \\ B & \xrightarrow{\quad} & \mathbf{F} B \end{array}$$

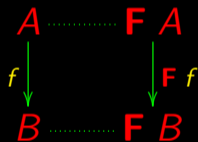
For instance:

$$\mathbf{F} X = 1 + \mathbb{N}_0 \times X$$

Recall **natural** (free)  
**properties**

# Functors

What does **F** mean, exactly?



Recall **natural** (free)  
**properties**

For instance:

$$F X = 1 + \mathbb{N}_0 \times X$$

Substitute uppercase ( $X$ ) by  
lowercase  $f$  ( $X \rightarrow f$ ):

# Functors

What does **F** mean, exactly?

$$\begin{array}{ccc} A & \cdots & \mathbf{F} A \\ \downarrow f & & \downarrow \mathbf{F} f \\ B & \cdots & \mathbf{F} B \end{array}$$

Recall **natural** (free)  
**properties**

For instance:

$$\mathbf{F} X = 1 + \mathbb{N}_0 \times X$$

Substitute uppercase ( $X$ ) by  
lowercase  $f$  ( $X \rightarrow f$ ):

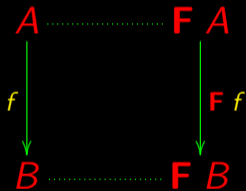
$$\mathbf{F} f = id + id \times f$$

# Properties

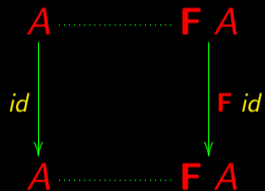
$$\mathbf{F} \textit{id} = \textit{id}$$

$$\mathbf{F} (g \cdot f) = (\mathbf{F} g) \cdot (\mathbf{F} f)$$

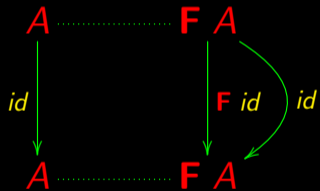
$$F id = id$$



$$F id = id$$



$$F id = id$$

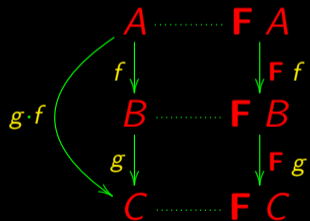




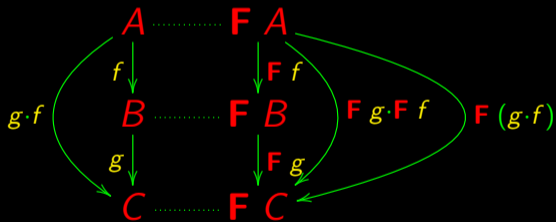
$$\mathbf{F} (g \cdot f) = (\mathbf{F} g) \cdot (\mathbf{F} f)$$

$$\begin{array}{ccc} A & \cdots & \mathbf{F} A \\ f \downarrow & & \downarrow \mathbf{F} f \\ B & \cdots & \mathbf{F} B \\ g \downarrow & & \downarrow \mathbf{F} g \\ C & \cdots & \mathbf{F} C \end{array}$$

$$\mathbf{F}(g \cdot f) = (\mathbf{F}g) \cdot (\mathbf{F}f)$$



$$F(g \cdot f) = (Fg) \cdot (Ff)$$



# Well known example

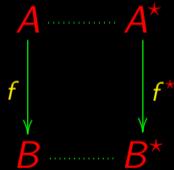
List functor:

$$\mathbf{F} f = f^*$$

# Well known example

List functor:

$$\mathbf{F} f = f^* \quad \text{where} \quad f^* x = [f a \mid a \leftarrow x] = \text{map } f x$$



In fact

$$id^* x = [a \mid a \leftarrow x] = x$$

In fact

$$id^* x = [a \mid a \leftarrow x] = x$$

$$(g^* \cdot f^*) x$$

In fact

$$id^* x = [a \mid a \leftarrow x] = x$$

$$(g^* \cdot f^*) x = [g \ b \mid b \leftarrow [f \ a \mid a \leftarrow x]]$$



In fact

$$id^* x = [a \mid a \leftarrow x] = x$$

$$(g^* \cdot f^*) x = [g b \mid b \leftarrow [f a \mid a \leftarrow x]] = (g \cdot f)^* x$$

# Properties

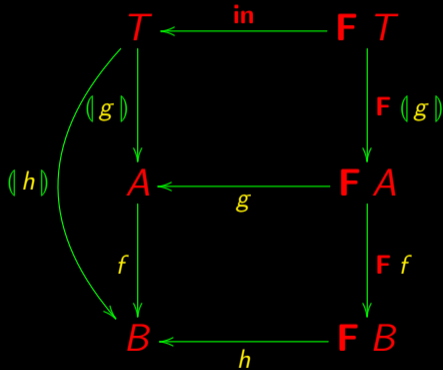
Recall

$$\begin{aligned} \mathbf{F} \, id &= id \\ \mathbf{F} (g \cdot f) &= (\mathbf{F} \, g) \cdot (\mathbf{F} \, f) \end{aligned}$$

These properties are essential for reasoning about **catamorphisms**.

See example next.

# Cata-fusion



# Cata-fusion

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

# Cata-fusion

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

$$\Leftrightarrow \left\{ \text{Cata-universal for } k = f \cdot (g) \right\}$$

$$f \cdot (g) \cdot \mathbf{in} = h \cdot \mathbf{F} (f \cdot (g))$$

# Cata-fusion

$$f \cdot \langle g \rangle = \langle h \rangle$$

$$\Leftrightarrow \left\{ \text{Cata-universal for } k = f \cdot \langle g \rangle \right\}$$

$$f \cdot \langle g \rangle \cdot \mathbf{in} = h \cdot \mathbf{F} (f \cdot \langle g \rangle)$$

$$\Leftrightarrow \left\{ \text{Cata-cancellation} \right\}$$

$$f \cdot g \cdot \mathbf{F} \langle g \rangle = h \cdot \mathbf{F} (f \cdot \langle g \rangle)$$

# Cata-fusion

$$f \cdot g \cdot \mathbf{F}(g) = h \cdot \mathbf{F}(f \cdot (g))$$

# Cata-fusion

$$f \cdot g \cdot \mathbf{F} (g) = h \cdot \mathbf{F} (f \cdot (g))$$

$$\Leftrightarrow \{ \text{functor-}\mathbf{F} (1) \}$$

$$f \cdot g \cdot \mathbf{F} (g) = h \cdot \mathbf{F} f \cdot \mathbf{F} (g)$$



# Cata-fusion

$$f \cdot g \cdot \mathbf{F} (g) = h \cdot \mathbf{F} (f \cdot (g))$$

$$\Leftrightarrow \{ \text{functor-}\mathbf{F} (1) \}$$

$$f \cdot g \cdot \mathbf{F} (g) = h \cdot \mathbf{F} f \cdot \mathbf{F} (g)$$

$$\Leftarrow \{ \text{Leibniz} \}$$

$$f \cdot g = h \cdot (\mathbf{F} f)$$

# Cata-fusion

$$f \cdot g \cdot \mathbf{F} \langle g \rangle = h \cdot \mathbf{F} (f \cdot \langle g \rangle)$$

$$\Leftrightarrow \{ \text{functor-}\mathbf{F} (1) \}$$

$$f \cdot g \cdot \mathbf{F} \langle g \rangle = h \cdot \mathbf{F} f \cdot \mathbf{F} \langle g \rangle$$

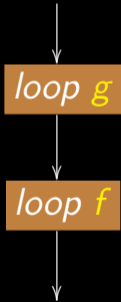
$$\Leftarrow \{ \text{Leibniz} \}$$

$$f \cdot g = h \cdot (\mathbf{F} f)$$

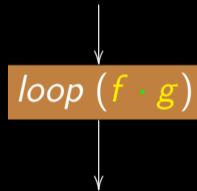
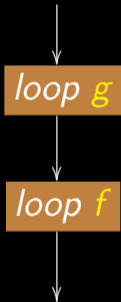
Summing up:

$$f \cdot \langle g \rangle = \langle h \rangle \Leftarrow f \cdot g = h \cdot (\mathbf{F} f)$$

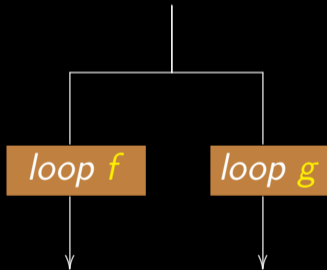
# Fusion in programming — "sequential"



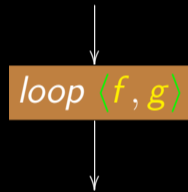
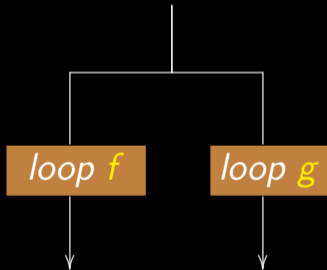
# Fusion in programming — "sequential"



# Fusion in programming — "parallel"

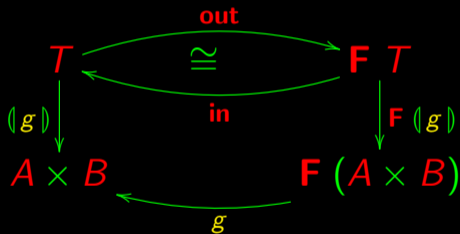


# Fusion in programming — "parallel"

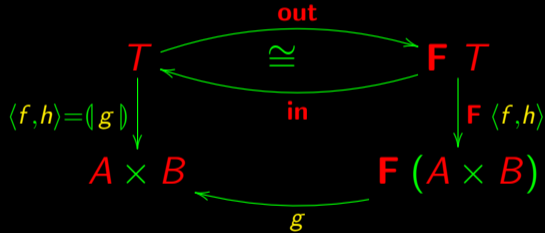


The pattern  $\langle f, g \rangle = \langle h \rangle$

Catas that yield pairs:

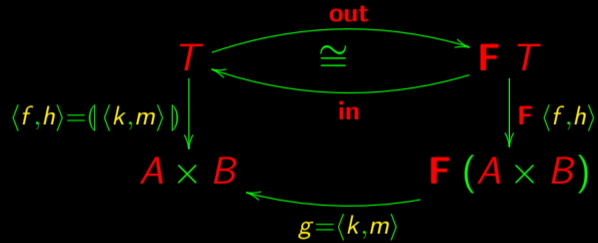


# Catas that yield pairs

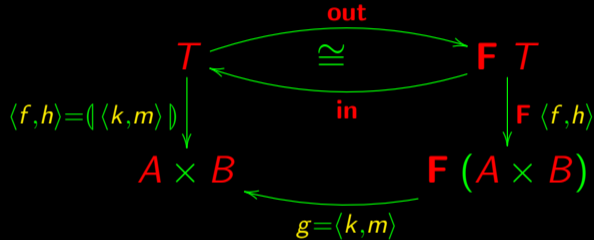




# Mutual recursion



# Mutual recursion



$$\langle f, h \rangle = (\langle k, m \rangle) \Leftrightarrow \begin{cases} f \cdot \mathbf{in} = k \cdot F \langle f, h \rangle \\ h \cdot \mathbf{in} = m \cdot F \langle f, h \rangle \end{cases}$$

# Mutual recursion

$$\langle f, h \rangle = \langle \langle k, m \rangle \rangle$$

$$\Leftrightarrow \{ \text{Cata-universal} \}$$

$$\langle f, h \rangle \cdot \mathbf{in} = \langle k, m \rangle \cdot \mathbf{F} \langle f, h \rangle$$

$$\Leftrightarrow \{ \times\text{-fusion} \}$$

$$\langle f \cdot \mathbf{in}, h \cdot \mathbf{in} \rangle = \langle k \cdot \mathbf{F} \langle f, h \rangle, m \cdot \mathbf{F} \langle f, h \rangle \rangle$$

$$\Leftrightarrow \{ \text{Eq-}\times \}$$

$$\begin{cases} f \cdot \mathbf{in} = k \cdot \mathbf{F} \langle f, h \rangle \\ h \cdot \mathbf{in} = m \cdot \mathbf{F} \langle f, h \rangle \end{cases}$$

# Mutual recursion — case $T = \mathbb{N}_0$

$$\langle f, g \rangle = (\langle h, k \rangle)$$

$$\Leftrightarrow \left\{ \text{mutual recursion with } \mathbf{in} = [\underline{0}, \mathit{succ}], \mathbf{F} f = \mathit{id} + f \text{ etc} \right\}$$

$$\begin{cases} f \cdot [\underline{0}, \mathit{succ}] = h \cdot (\mathit{id} + \langle f, g \rangle) \\ g \cdot [\underline{0}, \mathit{succ}] = k \cdot (\mathit{id} + \langle f, g \rangle) \end{cases}$$

$$\Leftrightarrow \left\{ \text{let } h = [h_1, h_2] \text{ and } k = [k_1, k_2] \right\}$$

$$\begin{cases} f \cdot [\underline{0}, \mathit{succ}] = [h_1, h_2] \cdot (\mathit{id} + \langle f, g \rangle) \\ g \cdot [\underline{0}, \mathit{succ}] = [k_1, k_2] \cdot (\mathit{id} + \langle f, g \rangle) \end{cases}$$

# Mutual recursion — case $T = \mathbb{N}_0$

Moving on:

$$\langle f, g \rangle = (\langle [h_1, h_2], [k_1, k_2] \rangle)$$

$$\Leftrightarrow \{ \text{coproduct laws} \}$$

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} f \cdot \underline{0} = h_1 \\ f \cdot \text{succ} = h_2 \cdot \langle f, g \rangle \end{array} \right. \\ \left\{ \begin{array}{l} g \cdot \underline{0} = k_1 \\ g \cdot \text{succ} = k_2 \cdot \langle f, g \rangle \end{array} \right. \end{array} \right.$$

## Mutual recursion — case $T = \mathbb{N}_0$

Finally, add variables to righthand side, with  $h_1 = \underline{a}$  and  $k_1 = \underline{b}$ :

$$\langle f, g \rangle = (\langle [\underline{a}, h_2], [\underline{b}, k_2] \rangle) \Leftrightarrow \begin{cases} \begin{cases} f\ 0 = a \\ f\ (n+1) = h_2\ (f\ n, g\ n) \end{cases} \\ \begin{cases} g\ 0 = b \\ g\ (n+1) = k_2\ (f\ n, g\ n) \end{cases} \end{cases}$$

## Example — Fibonacci

Clearly, the following function is not a catamorphism of  $\mathbb{N}_0$ :

$$\mathit{fib} \ 0 = 1$$

$$\mathit{fib} \ 1 = 1$$

$$\mathit{fib} \ (n + 2) = \mathit{fib} \ (n + 1) + \mathit{fib} \ n$$

But it can be transformed into one using mutual recursion.

## Example — Fibonacci

Let us define the auxiliary function:

$$f\ n = fib\ (n + 1)$$

Then:

$$f\ 0 = fib\ 1 = 1$$

$$f\ (n + 1) = fib\ (n + 2) = f\ n + fib\ n$$



## Example — Fibonacci

Concerning *fib*:

$$\mathit{fib} \ 0 = 1$$

$$\mathit{fib} \ (n + 1) = f \ n$$

Summing up, we have the following mutual recursion:

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} f \ 0 = 1 \\ f \ (n + 1) = f \ n + \mathit{fib} \ n \end{array} \right. \\ \left\{ \begin{array}{l} \mathit{fib} \ 0 = 1 \\ \mathit{fib} \ (n + 1) = f \ n \end{array} \right. \end{array} \right.$$

## Example — Fibonacci

By applying the mutual recursion law for  $T = \mathbb{N}_0$ :

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} f\ 0 = a \\ f\ (n + 1) = h_2\ (f\ n, fib\ n) \end{array} \right. \\ \left\{ \begin{array}{l} fib\ 0 = b \\ fib\ (n + 1) = k_2\ (f\ n, fib\ n) \end{array} \right. \end{array} \right. \Leftrightarrow \langle f, fib \rangle = \langle \langle [a, h_2], [b, k_2] \rangle \rangle$$

## Example — Fibonacci

Solutions:

$$a = 1$$

$$b = 1$$

$$h_2 = \text{add}$$

$$k_2 = \pi_1$$

Then:

$$\langle f, fib \rangle = ( \langle [\underline{1}, \text{add}], [\underline{1}, \pi_1] \rangle )$$

# Fibonacci becomes a for-loop

Given  $\langle f, fib \rangle$  (a  $\mathbb{N}_0$  **catamorphism**), we can convert it into a **for-loop** according to the definition:

**for**  $b$   $i = ([i, b])$

# Fibonacci becomes a for-loop

Given  $\langle f, fib \rangle$  (a  $\mathbb{N}_0$  catamorphism), we can convert it into a **for-loop** according to the definition:

**for**  $b$   $i = ([i, b])$

$$\langle f, fib \rangle = (\langle [\underline{1}, add], [\underline{1}, \pi_1] \rangle)$$

$$\Leftrightarrow \{ \text{exchange law} \}$$

$$\langle f, fib \rangle = ([\langle \underline{1}, \underline{1} \rangle, \langle add, \pi_1 \rangle])$$

$$\Leftrightarrow \{ \text{recall exercise sheet 3} \}$$

$$\langle f, fib \rangle = ([\underline{(1, 1)}, \langle add, \pi_1 \rangle])$$

$$\Leftrightarrow \{ \text{definition of for } b i \}$$

$$\langle f, fib \rangle = \text{for } \langle add, \pi_1 \rangle (1, 1)$$

## Example — Fibonacci

In summary:

$$fib = \pi_2 \cdot (\text{for } \langle add, \pi_1 \rangle (1, 1))$$

Cf. the same in C:

```
int fib(int n)
{
  int x=1; int y=1; int i;
  for (i=1; i<=n; i++) {int a=x; x=x+y; y=a;}
  return y;
};
```

## Example — factorial

$$fac\ 0 = 1$$

$$fac\ (n + 1) = \underbrace{(n + 1)}_{f\ n} \times fac\ n$$

Then:

$$\langle f, fac \rangle = (\langle [\underline{a}, h_2], [\underline{b}, k_2] \rangle) \Leftrightarrow \left\{ \begin{array}{l} \left\{ \begin{array}{l} f\ 0 = a \\ f\ (n + 1) = h_2 (f\ n, fac\ n) \end{array} \right. \\ \left\{ \begin{array}{l} fac\ 0 = b \\ fac\ (n + 1) = k_2 (f\ n, fac\ n) \end{array} \right. \end{array} \right.$$

## Example — factorial

$$\langle f, fac \rangle = (\langle [\underline{a}, h_2], [\underline{b}, k_2] \rangle) \Leftrightarrow \left\{ \begin{array}{l} \left\{ \begin{array}{l} f\ 0 = a \\ f\ (n + 1) = h_2(f\ n, fac\ n) \end{array} \right. \\ \left\{ \begin{array}{l} fac\ 0 = b \\ fac\ (n + 1) = k_2(f\ n, fac\ n) \end{array} \right. \end{array} \right.$$

$$k_2 = mul$$

$$b = 1$$

$$f\ n = n + 1$$

$$a = 1$$

(Still need  $h_2...$ )



## Example — factorial

$$\begin{aligned} & f(n+1) \\ = & (n+1) + 1 \\ = & 1 + (n+1) \\ = & 1 + f\ n \\ = & 1 + \pi_1(f\ n, fac\ n) \\ = & \underbrace{succ \cdot \pi_1}_{h_2}(f\ n, fac\ n) \end{aligned}$$

Then:

$$k_2 = mul$$

$$b = 1$$

$$h_2 = succ \cdot \pi_1$$

$$a = 1$$

## Example — factorial becomes a for-loop

Finally:

$$\begin{aligned} & \langle f, fac \rangle \\ = & \left( \left\langle [\underline{1}, succ \cdot \pi_1], [\underline{1}, mul] \right\rangle \right) \\ = & \left( \left[ \underline{(1, 1)}, \langle succ \cdot \pi_1, mul \rangle \right] \right) \\ = & \text{for } \langle succ \cdot \pi_1, mul \rangle ((1, 1)) \\ = & \text{for } g (1, 1) \text{ where } g (x, y) = (x + 1, x \times y) \end{aligned}$$

## For-loops (in C)

In general,  $k = \text{for } f \ i$  can be encoded in the syntax of C by writing:

```
int k(int n) {  
    int r=i;  
    int j;  
    for (j=1;j<n+1;j++) {r=f(r);}   
    return r;  
};
```

## Factorial as a for-loop (in C)

From

$$fac = \pi_2 (\text{for } g (1, 1)) \text{ where } g (x, y) = (x + 1, x \times y)$$

we thus obtain:

```
int fac(int n) {  
    int x=1; int y=1;  
    int j;  
    for (j=1; j<n+1; j++) {y=x*y;x=x+1;}  
    return y;  
};
```

# Banana-split



# Banana-split



( <-, -> )

# Banana-split



$( \langle -, - \rangle )$

$\langle ( - ), ( - ) \rangle$

# Banana-split



$( \langle \_ , \_ \rangle )$

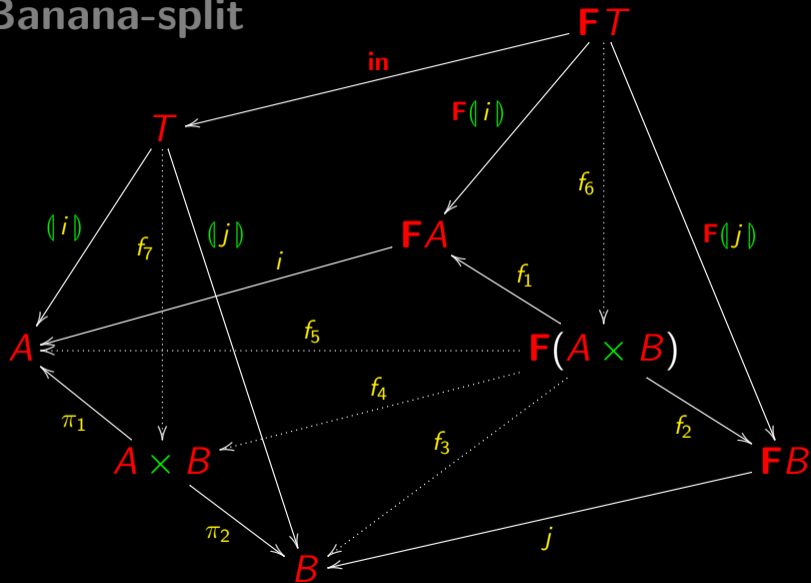
$\langle ( \_ ) , ( \_ ) \rangle$

$\langle ( i ) , ( j ) \rangle$



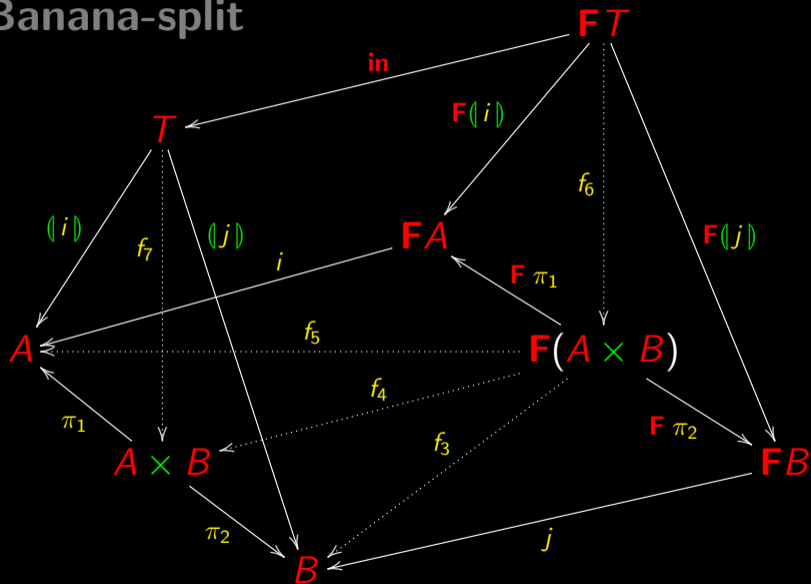
# Banana-split

$\langle (i), (j) \rangle$



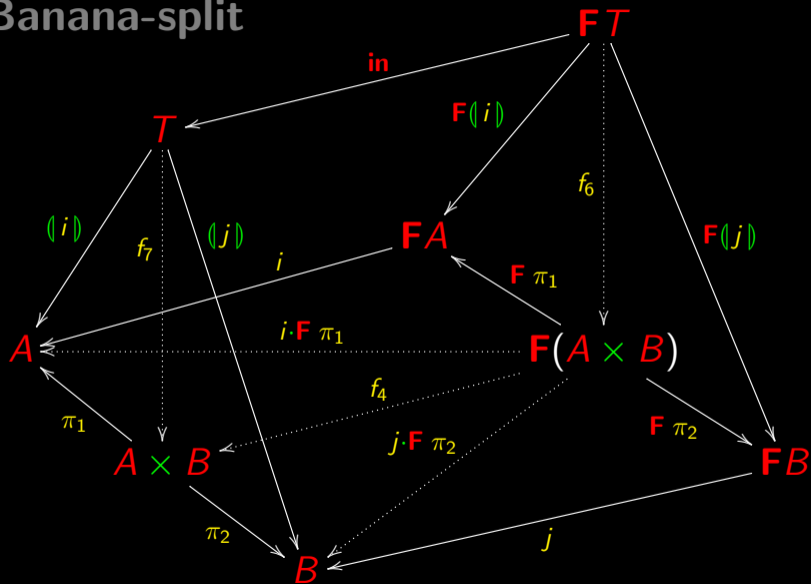
# Banana-split

$\langle (i), (j) \rangle$



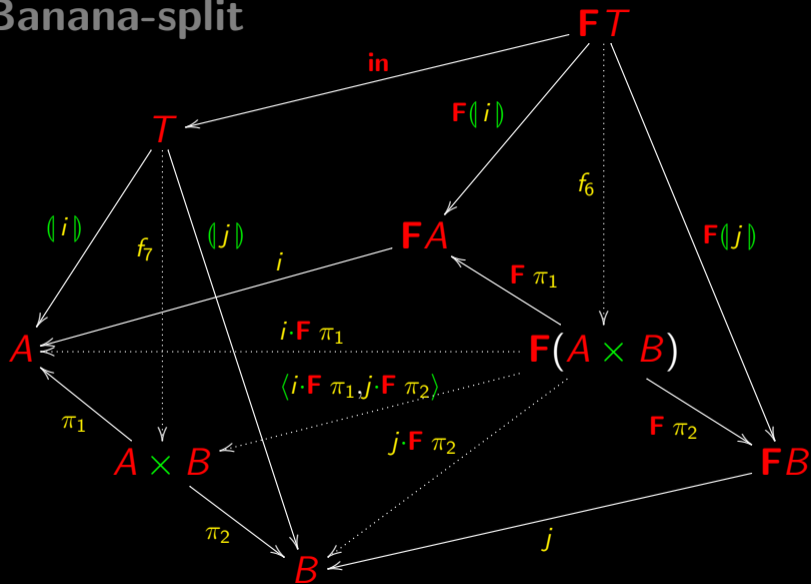
# Banana-split

$\langle (i), (j) \rangle$

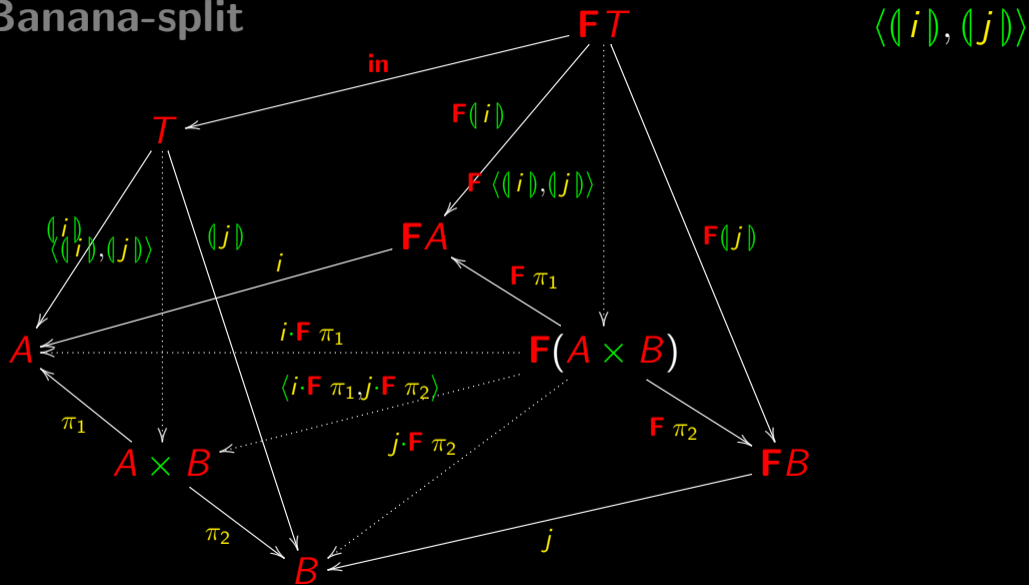


# Banana-split

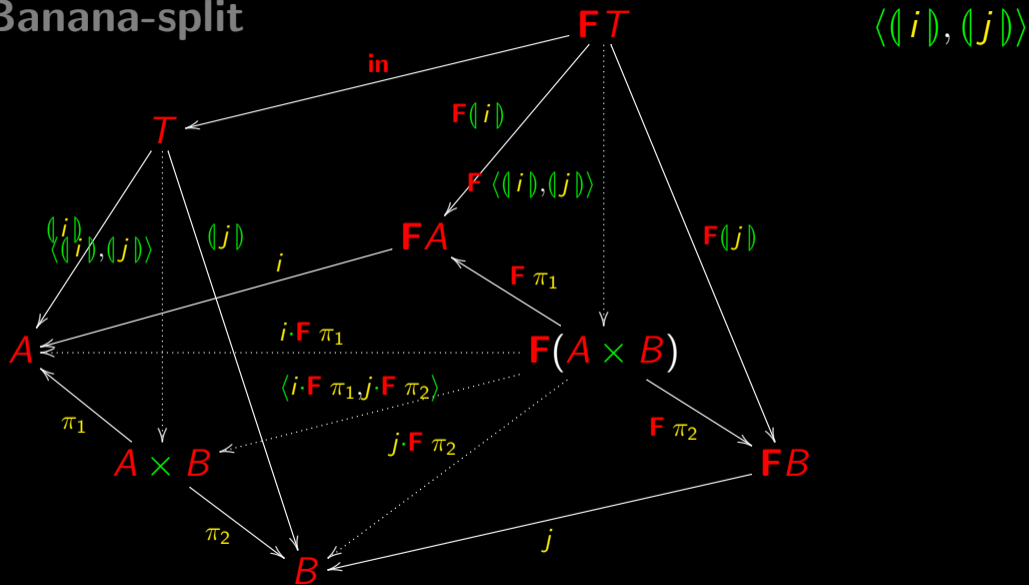
$\langle (i), (j) \rangle$



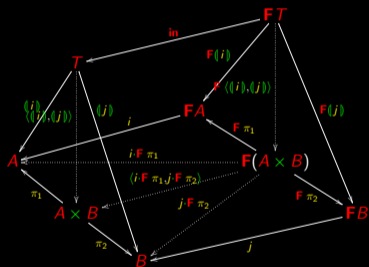
# Banana-split



# Banana-split



# Banana-split



$$\langle (i), (j) \rangle = \langle (i \cdot F\pi_1, j \cdot F\pi_2) \rangle$$

## Example

Average of a (non empty) list:

$$average = (/) \cdot \langle sum, length \rangle$$

where

$$sum = ([0, add])$$

$$length = ([0, succ \cdot \pi_2])$$



# Example

$$average = (/) \cdot \underbrace{\langle sum, length \rangle}_{aux}$$

By banana-split:

$aux = (\langle f_5, f_3 \rangle)$  where

$$f_5 = [0, add] \cdot (id + id \times \pi_1)$$

$$f_3 = [0, succ \cdot \pi_2] \cdot (id + id \times \pi_2)$$



# Example

Finally, solve  $aux = (\langle f_5, f_3 \rangle)$  and convert everything to pointwise notation:



$average\ l = x / y$  where

$(x, y) = aux\ l$

$aux\ [] = (0, 0)$

$aux\ (a : l) = (a + x, y + 1)$  where  $(x, y) = aux\ l$

# Example

Summing up,

*average*  $l = x / y$  where

$(x, y) = \text{aux } l$

$\text{aux } [] = (0, 0)$

$\text{aux } (a : l) = (a + x, y + 1)$  where  $(x, y) = \text{aux } l$

twice as fast as

*average*  $l = (\text{sum } l) / (\text{length } l)$  where

$\text{sum } [] = 0$

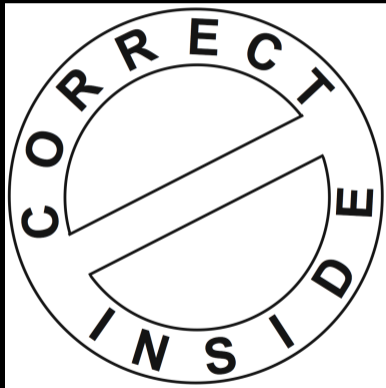
$\text{sum } (h : t) = h + \text{sum } t$

$\text{length } [] = 0$

$\text{length } (h : t) = 1 + \text{length } t$

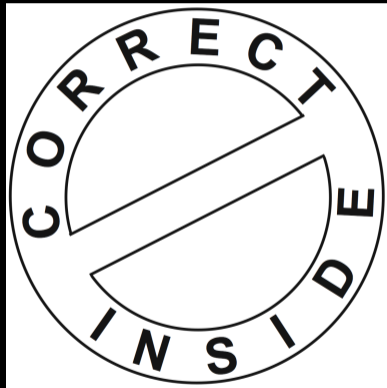
and **guaranteed** to be correct.

# Program Calculation



**Efficiency** without sacrificing  
**correction.**

# Program Calculation



**Efficiency** without sacrificing **correction**.

Who cares about an efficient program that gives wrong outputs?

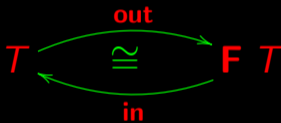
# Program Calculation



**Efficiency**  
should never  
sacrifice  
**correction!**

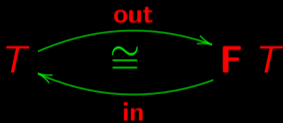
# Recap

Inductive type  $T$  and the functor  $F$  that determines its pattern of recursion:



# Recap

Inductive type  $T$  and the functor  $F$  that determines its pattern of recursion:

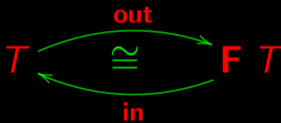


We have seen examples:  $T = \mathbb{N}_0$ ,  $T = A^*$  etc.



# Recap

Inductive type  $T$  and the functor  $F$  that determines its pattern of recursion:



We have seen examples:  $T = \mathbb{N}_0$ ,  $T = A^*$  etc.

Let us see some more.

## Binary trees

Trees with data of type  $A$  in their nodes, for example

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

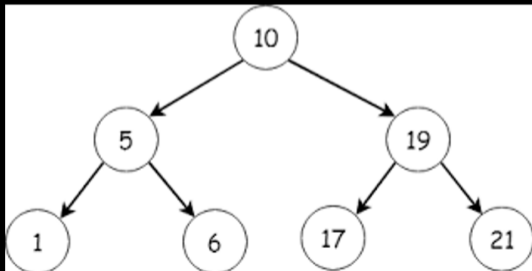
( $A$  assumed pre-defined.)

## Binary trees

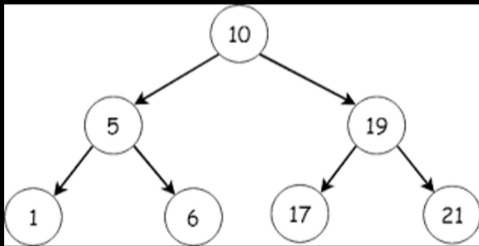
Trees with data of type  $A$  in their nodes, for example

$\text{data } BTree = \text{Empty} \mid \text{Node } (A, (BTree, BTree))$

( $A$  assumed pre-defined.) For instance



# Binary trees



```
Node (10,  
      (Node (5,  
            (Node (1,  
                  (Empty,Empty)),  
                  Node (6,  
                        (Empty,Empty))))),  
      Node (19,  
            (Node (17,  
                  (Empty,Empty)),  
                  Node (21,  
                        (Empty,Empty))))))
```

# Binary trees

```
: data BTree = Empty | Node(A, (BTree, BTree))
```

```
: :t Node
```

```
Node :: (A, (BTree, BTree)) -> BTree
```

```
: :t Empty
```

```
Empty :: BTree
```

# Binary trees

```
: data BTree = Empty | Node(A, (BTree, BTree))
```

```
: :t Node
```

```
Node :: (A, (BTree, BTree)) -> BTree
```

```
: :t Empty
```

```
Empty :: BTree
```

$Node : A \times (BTree \times BTree) \rightarrow BTree$

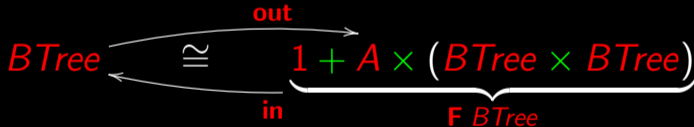
$\underline{Empty} : 1 \rightarrow BTree$

# Binary trees

**in** = [Empty, Node]

# Binary trees

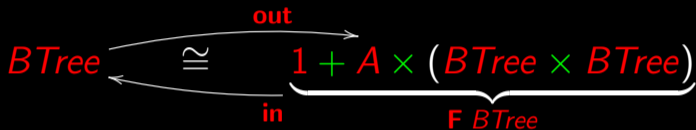
**in** = [Empty, Node]



**data** *BTree* = *Empty* | *Node* (*A*, (*BTree*, *BTree*))



# Binary trees

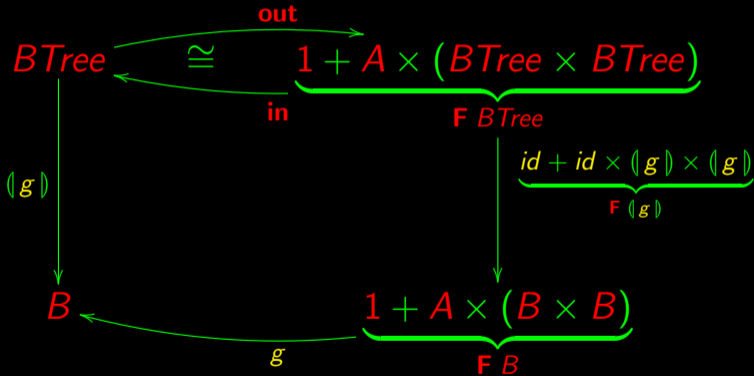


$$T = BTree$$

$$\begin{cases} \mathbf{F} X = 1 + A \times (X \times X) \\ \mathbf{F} f = id + id \times (f \times f) \end{cases}$$

$$\mathbf{in} = [\underline{Empty}, Node]$$

# Catamorphisms of binary trees



# Catamorphisms of binary trees

In Haskell:

---

```
cataBTree g = g . (recBTree (cataBTree g)) . outBTree
```

```
outBTree Empty           = Left ()  
outBTree (Node (a,(t1,t2))) = Right(a,(t1,t2))
```

```
recBTree f = id -|- id >< (f >< f)
```

```
inBTree = either (const Empty) Node
```

---

# Catamorphisms of binary trees

Example: function

$$f : BTree \rightarrow \mathbb{N}_0$$

that computes the size of the tree (total number of nodes).

# Catamorphisms of binary trees

Example: function

$$f : BTree \rightarrow \mathbb{N}_0$$

that computes the size of the tree (total number of nodes).

Enough caring about the “gene”  $g$  in

$$f = \llbracket g \rrbracket$$

where

$$\mathbb{N}_0 \xleftarrow{g} 1 + A \times (\mathbb{N}_0 \times \mathbb{N}_0)$$

# Catamorphisms of binary trees

$$N_0 \xleftarrow{g} 1 + A \times (N_0 \times N_0)$$

# Catamorphisms of binary trees

$$N_0 \longleftarrow 1 + A \times (N_0 \times N_0)$$

$g = [g_1, g_2]$

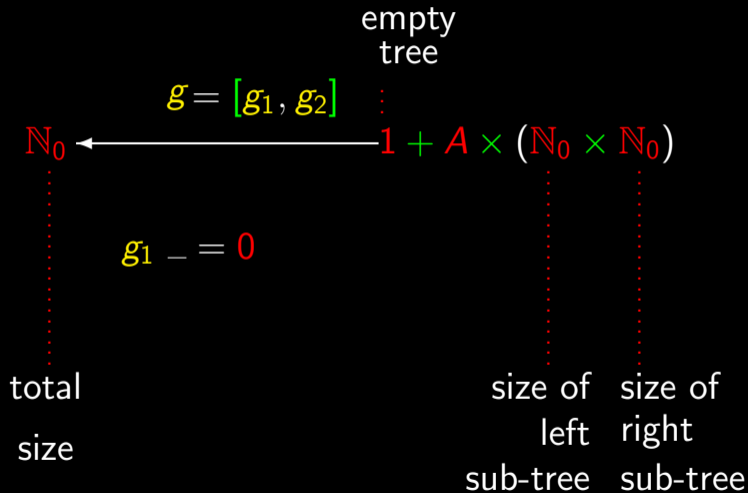
# Catamorphisms of binary trees

$$\begin{array}{ccc} & \text{empty} & \\ & \text{tree} & \\ & \vdots & \\ g = [g_1, g_2] & & \\ \leftarrow N_0 & \xleftarrow{\quad} & 1 + A \times (N_0 \times N_0) \end{array}$$

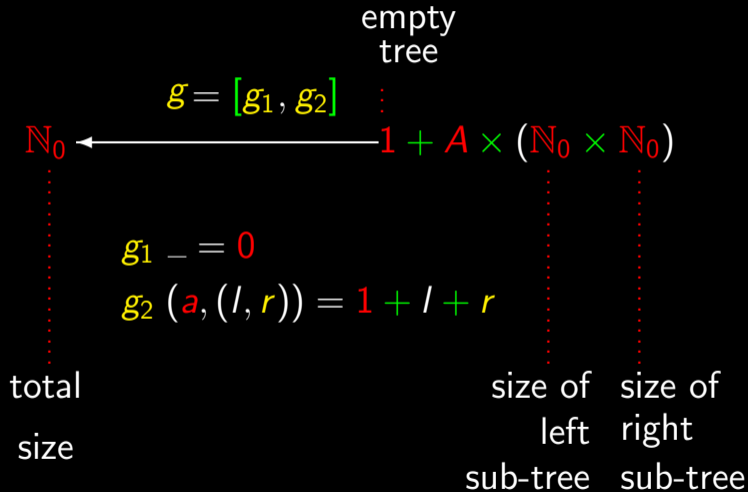
$$g_1 \_ = 0$$



# Catamorphisms of binary trees



# Catamorphisms of binary trees



# Catamorphisms of binary trees

Summing up,

$f = ([g_1, g_2])$  where

$$g_1 \_ = 0$$

$$g_2 (a, (l, r)) = 1 + l + r$$

---

# Catamorphisms of binary trees

Summing up,

$f = ([g_1, g_2])$  where

$$g_1 \_ = 0$$

$$g_2 (a, (l, r)) = 1 + l + r$$

---

Haskell:

```
size = cataBTree(either g1 g2) where
```

```
  g1 _ = 0
```

```
  g2(a, (l,r)) = 1 + l + r
```

# Cálculo de Programas

Class T09