

Cálculo de Programas

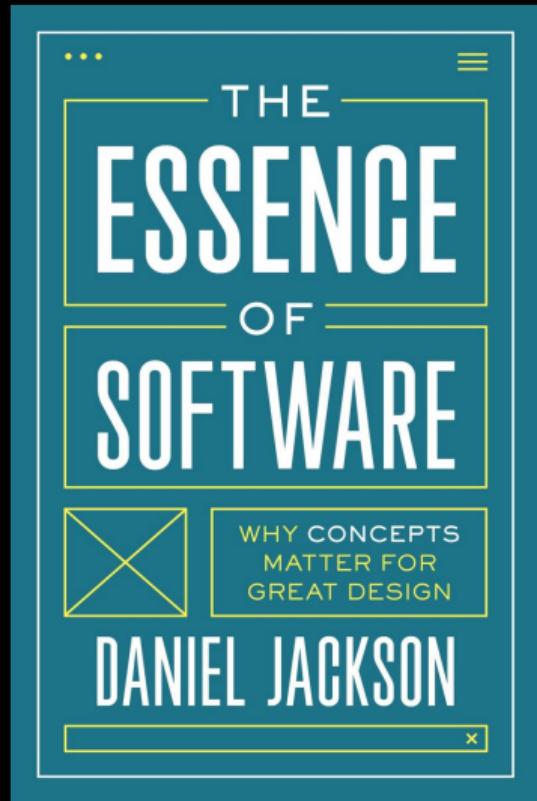
Class T01

J.N. Oliveira



Part I

"(...) 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



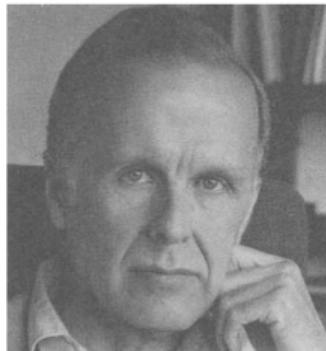
Urgently needed in software design



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 ...

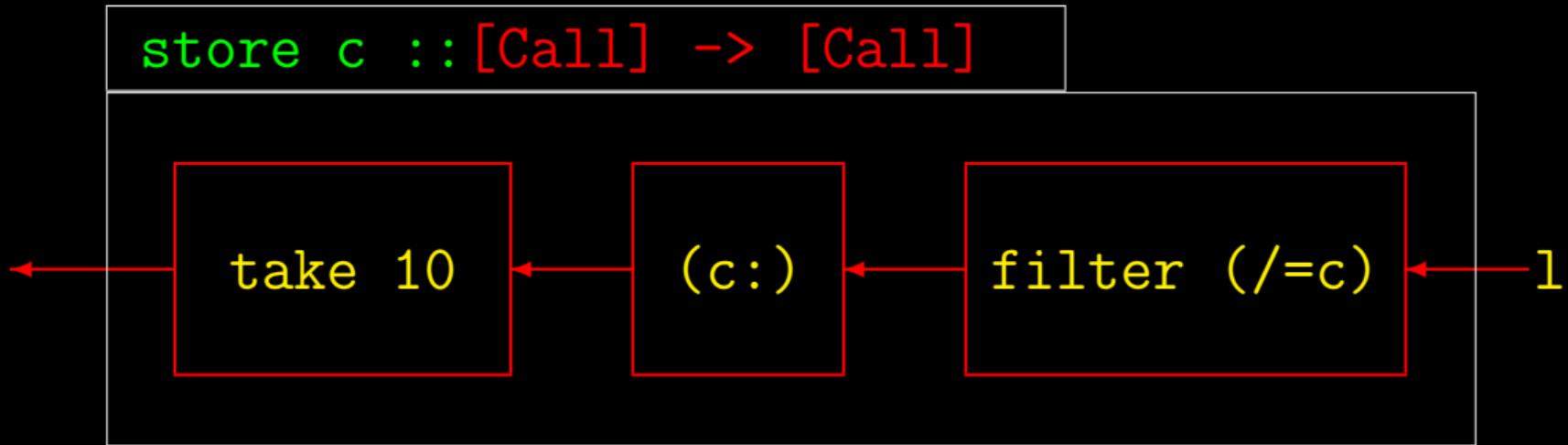
```
public void store10(string phoneNumber)
{
    System.Collections.ArrayList auxList =
        new System.Collections.ArrayList();
    auxList.Add(phoneNumber);
    auxList.AddRange(
        this.filteratmost9(phoneNumber) );
    this.callList = auxList;
}
```

+ filteratmost9 (next slide)

Compare with ...

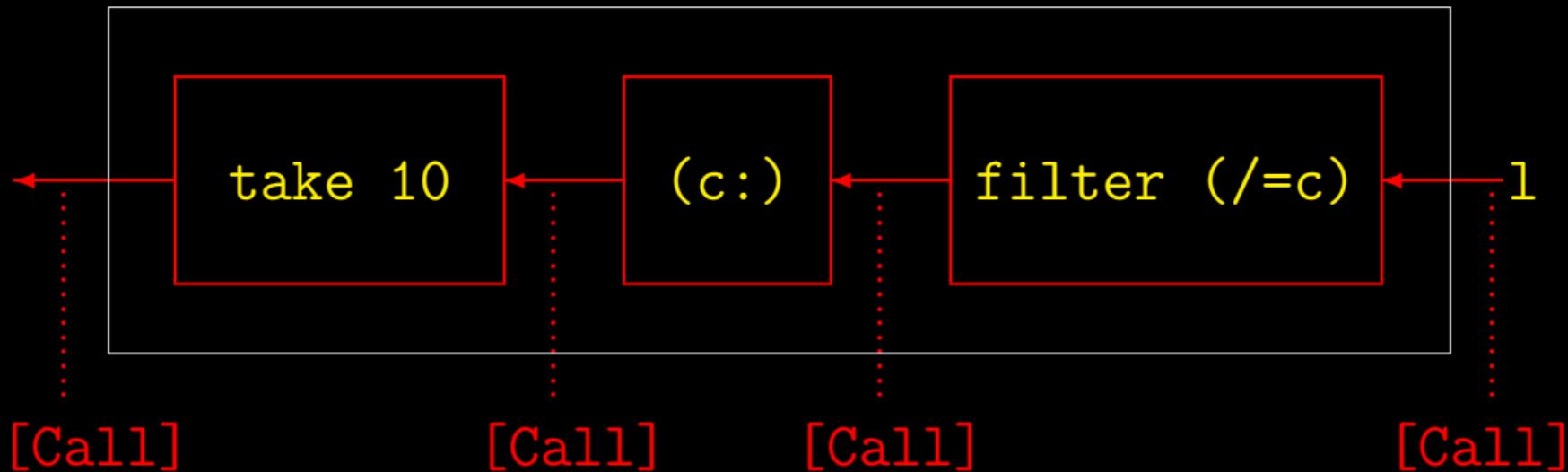
```
public System.Collections.ArrayList filteratmost9(string n)
{
    System.Collections.ArrayList retList =
        new System.Collections.ArrayList();
    int i=0, m=0;
    while((i < this.callList.Count) && (m < 9))
    {
        if ((string)this.callList[i] != n)
        {
            retList.Add(this.callList[i]);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

From a mobile phone manufacturer



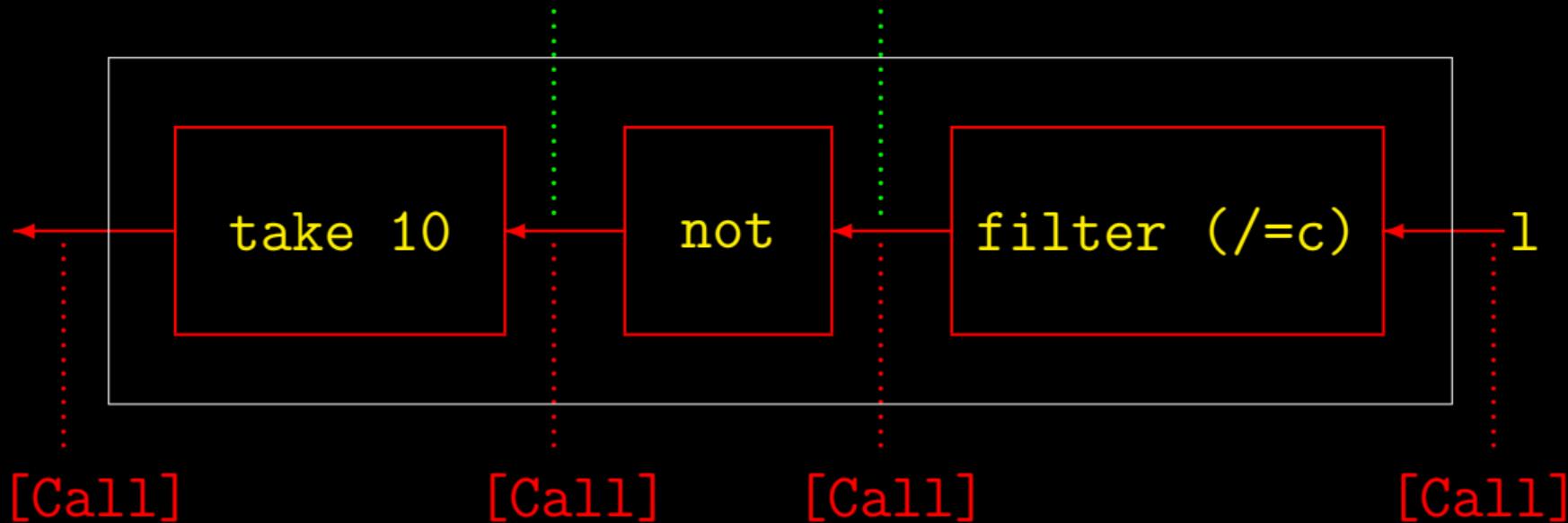
From a mobile phone manufacturer

store $c :: [\text{Call}] \rightarrow [\text{Call}]$



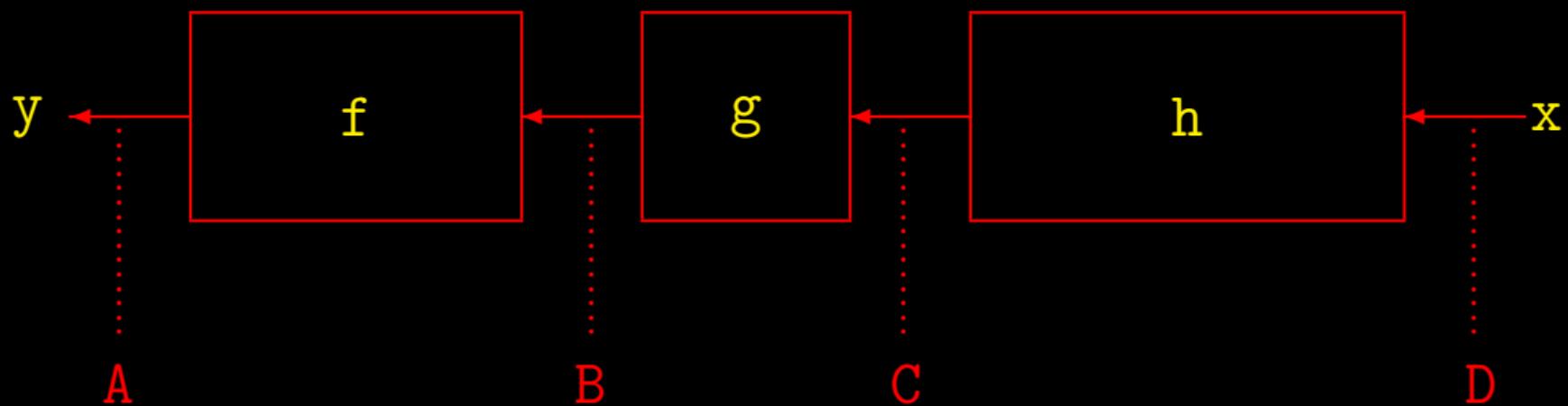
Ups!

Bool (!) Bool (!)



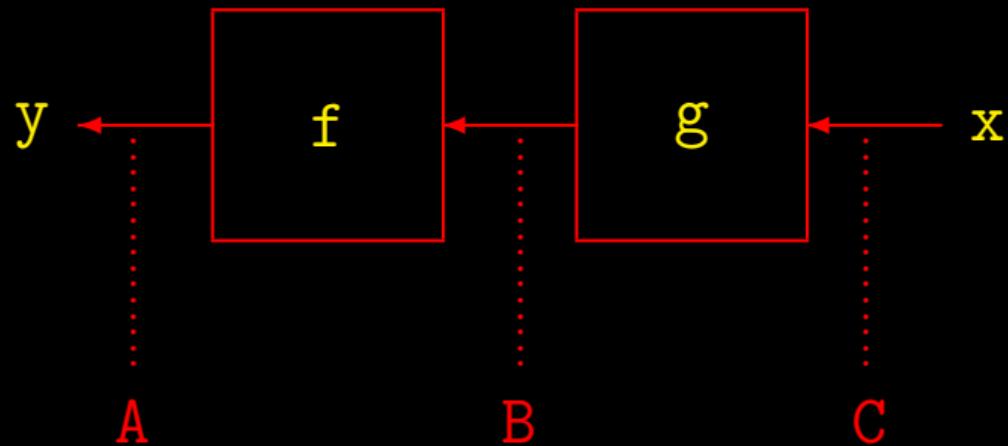
In general

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



In general

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



Simplification

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



Composition

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



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

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

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

Composition

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

i.e.

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

Composition

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

i.e.

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

the same as

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

Composition

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

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

Composition

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

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

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

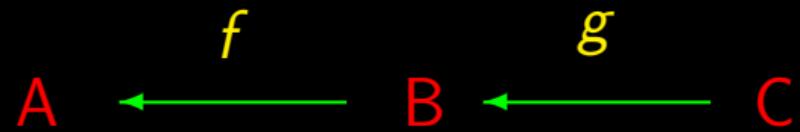
Composition

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

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

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

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



$$\begin{array}{ccccc} & & g & & \\ C & \xrightarrow{\hspace{2cm}} & B & \xrightarrow{\hspace{2cm}} & A \end{array}$$

A

f

B

g

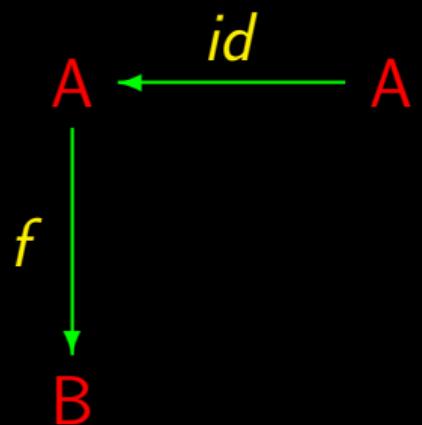
C

$$A \xleftarrow{id} A$$

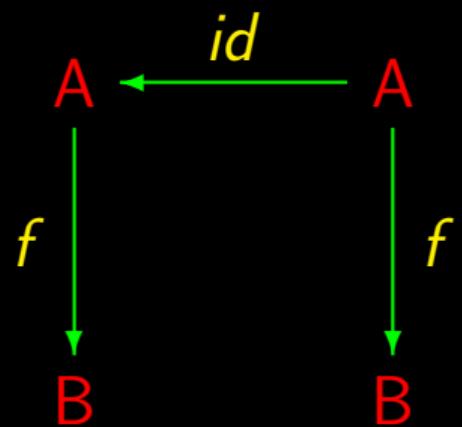
$$A \xleftarrow{id} A$$

$$id\ a = a$$

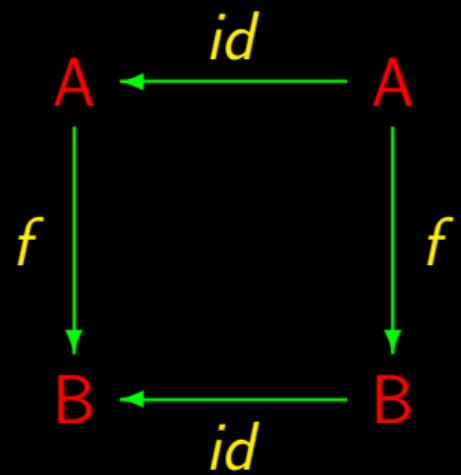
Identity



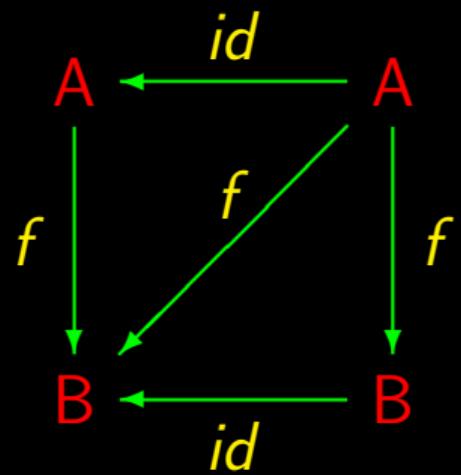
Identity



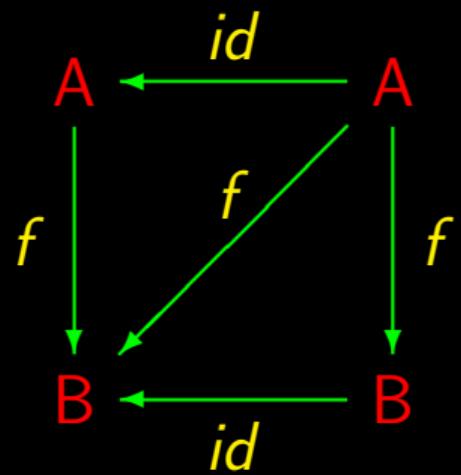
Identity



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*”:

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

$$f\cdot g$$

$f \cdot g$

$f \times g$?

$$f \cdot g$$

$$f \times g ?$$

$$f + g ?$$

Cálculo de Programas

Class T02

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

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

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

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

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

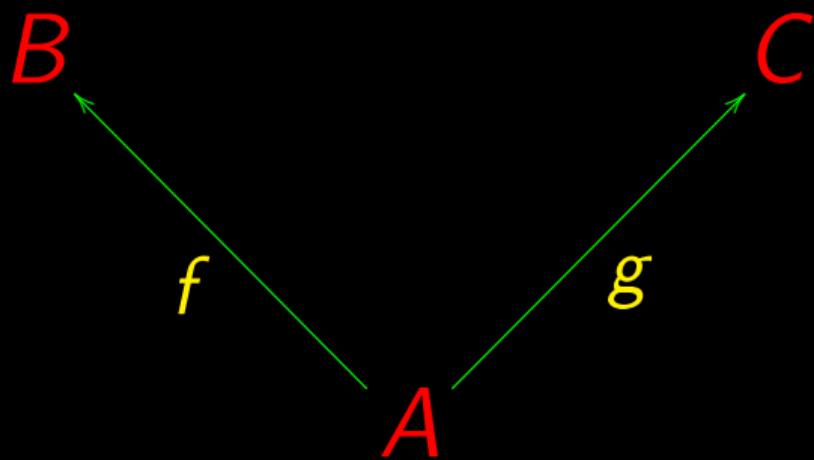
$$B \xleftarrow{f} C \quad \text{e} \quad C \xleftarrow{g} A$$

$$D \xrightarrow{f} B$$

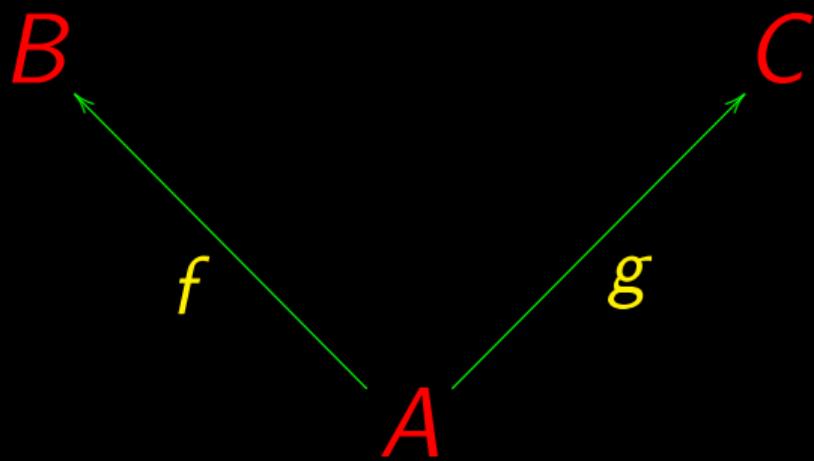
$$A \xrightarrow{g} C$$

?

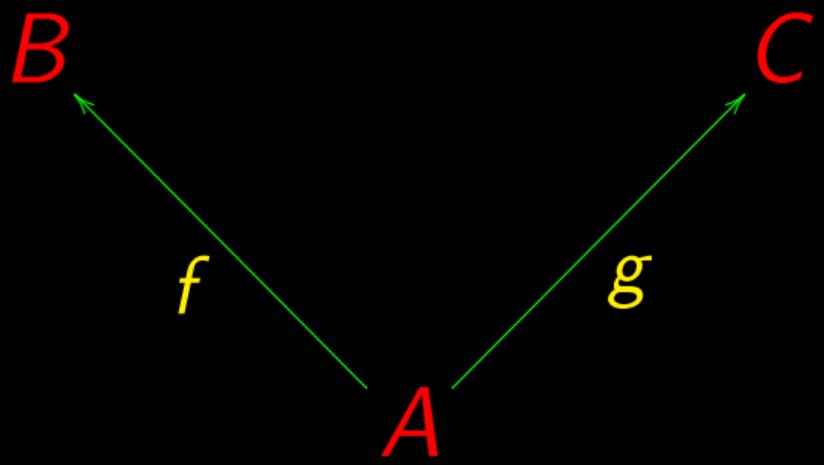
$D = A ?$



$D = A ?$



$f \ a \dots g \ a$



$(f \ a, g \ a)$

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

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

$$f \ a \in B$$

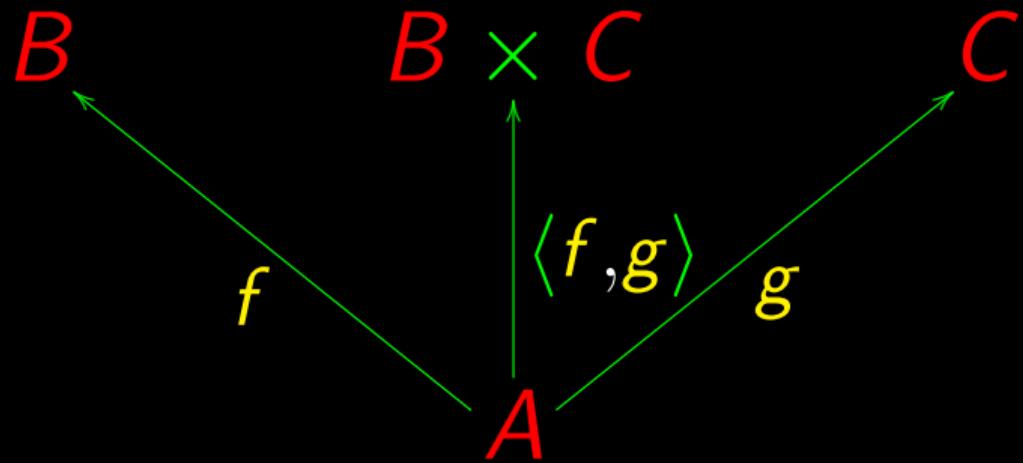
$$g \ a \in C$$

Cartesian product

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

$$\frac{f\ a \in B}{g\ a \in C} \quad \frac{}{(f\ a, g\ a) \in B \times C}$$

“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

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

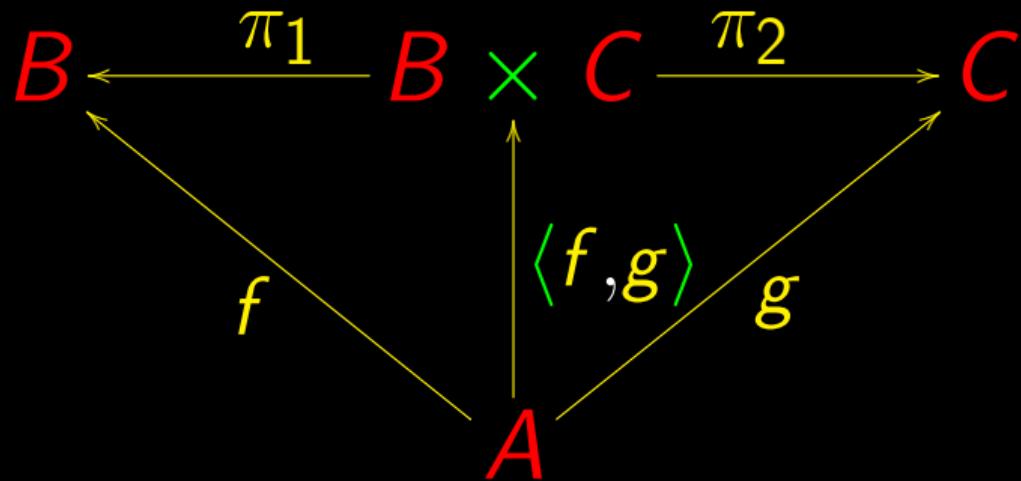
$$\pi_1 : A \times B \rightarrow A$$

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

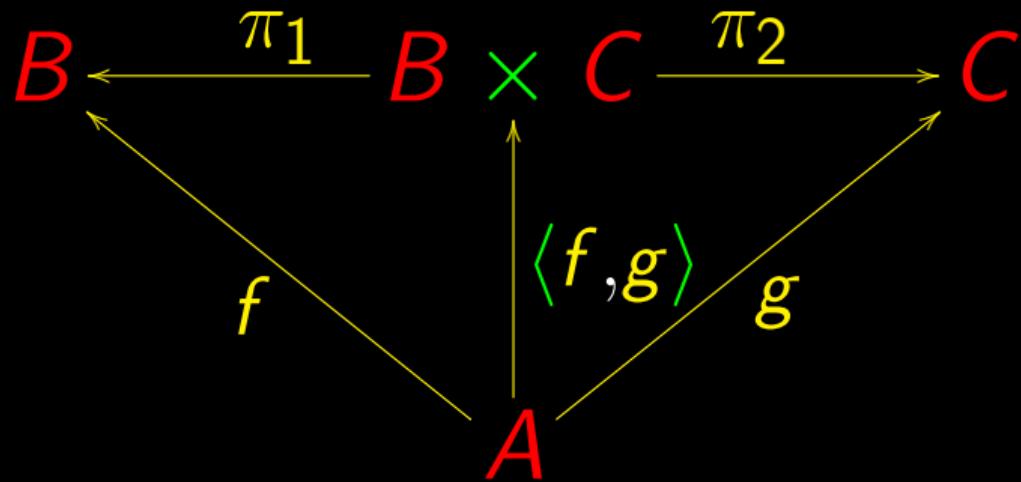
$$\pi_2 : A \times B \rightarrow B$$

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

Product

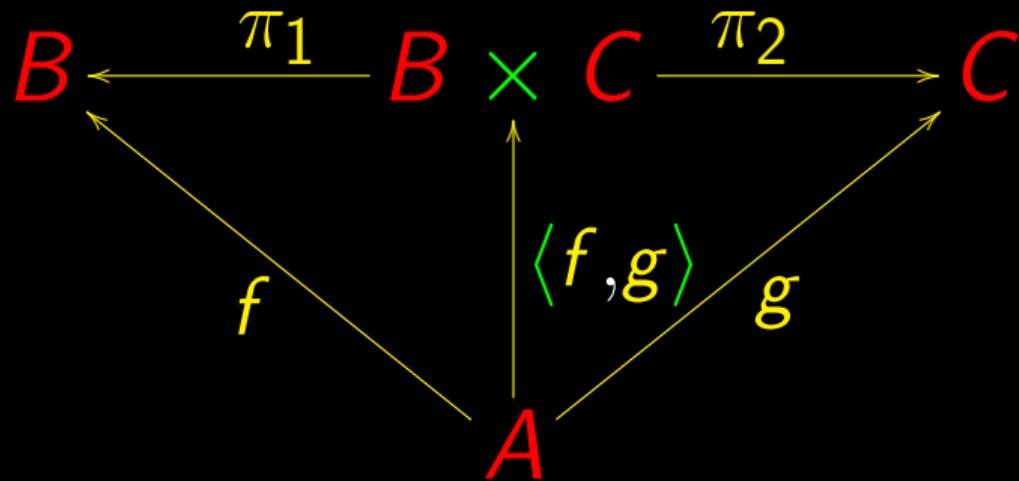


Product



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

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 paralelo

f “split” g

Product

$$\langle f, g \rangle$$

f and g in paralelo

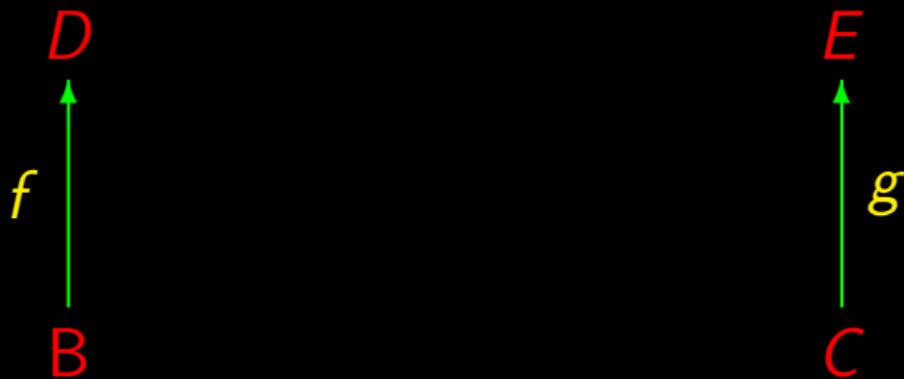
f “split” g

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

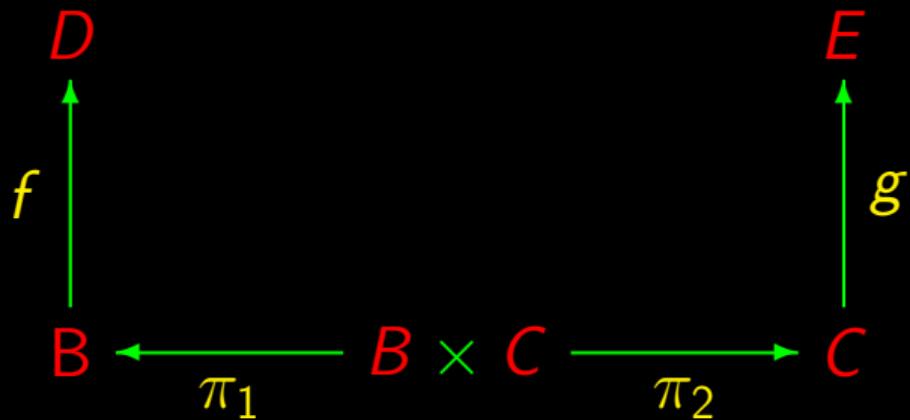
Product

$$D \xrightarrow{f} B$$

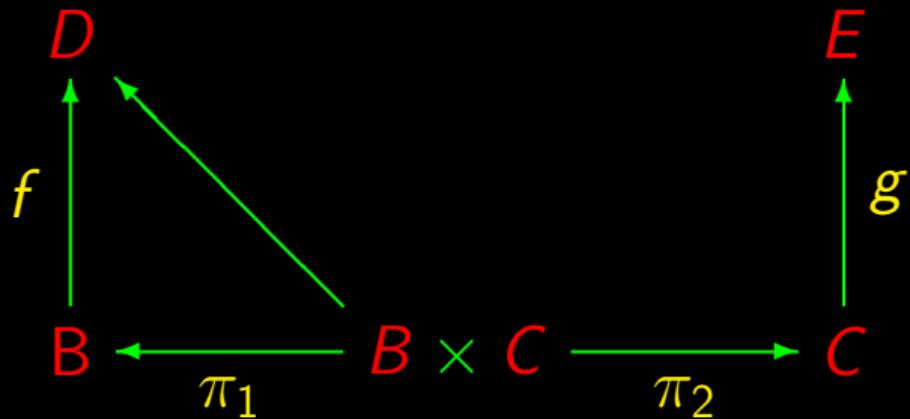
Product



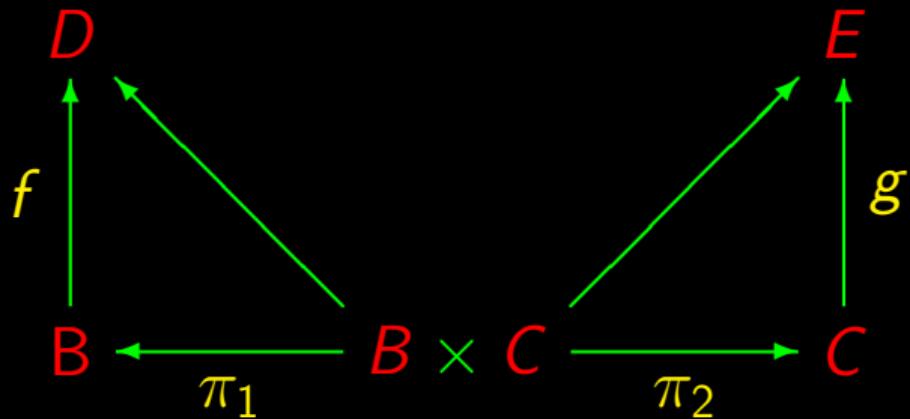
Product



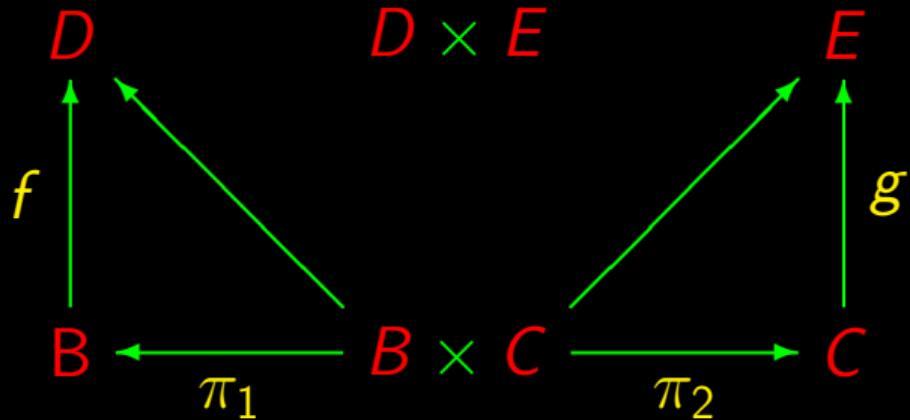
Product



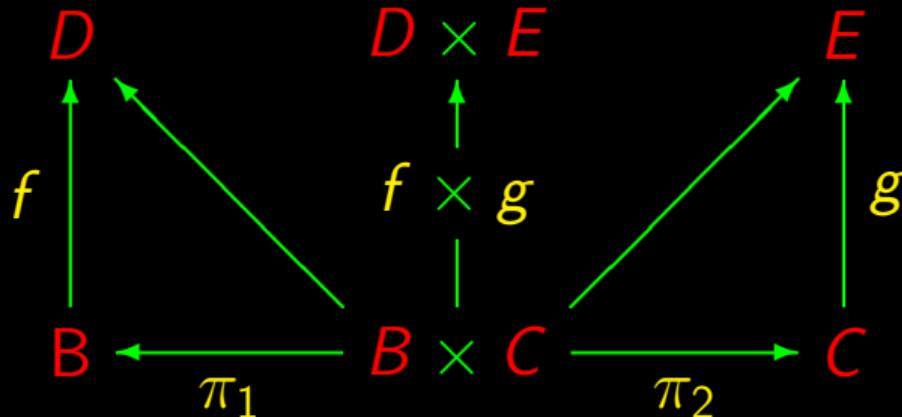
Product



Product

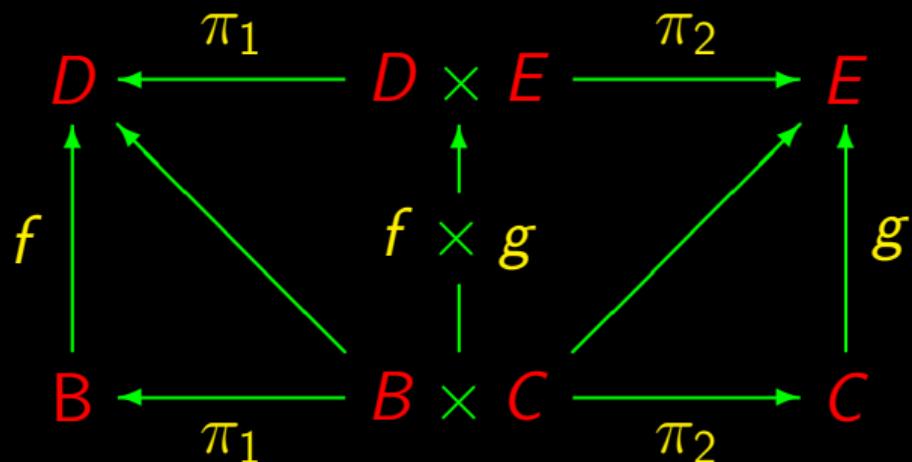


Product



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

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

$f \cdot g$

$\langle f, g \rangle$

$f \times g$

Sequential composition

Parallel composition (**synchronous**)

Parallel composition (**asynchronous**)

Summing up

$f \cdot g$

$\langle f, g \rangle$

$f \times g$

Sequential composition

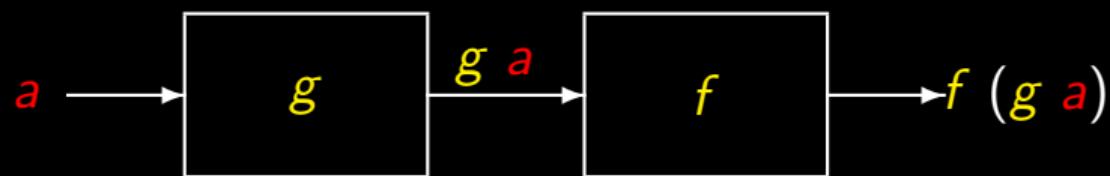
Parallel composition (synchronous)

Parallel composition (asynchronous)

Compositional programming

Summing up

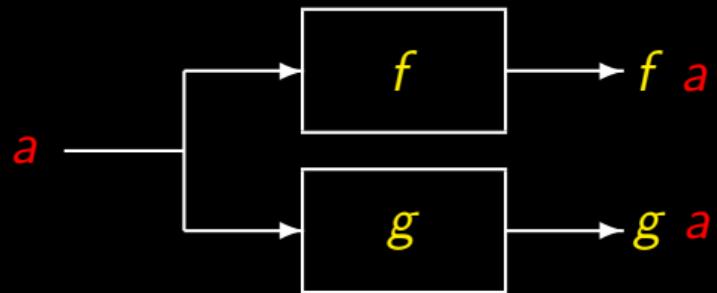
$$(f \cdot g) a = f (g a) \quad (2.6)$$



Function **composition**

Summing up

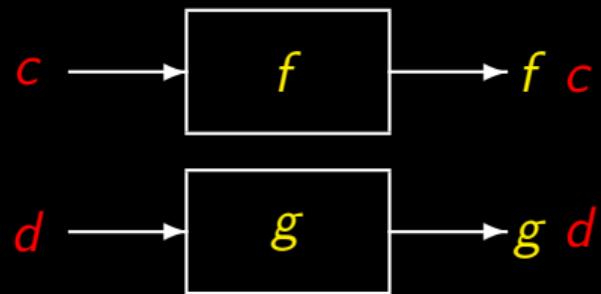
$$\langle f, g \rangle a = (f a, g a) \quad (2.20)$$



Functional “splits”

Summing up

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



Functional **products**

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

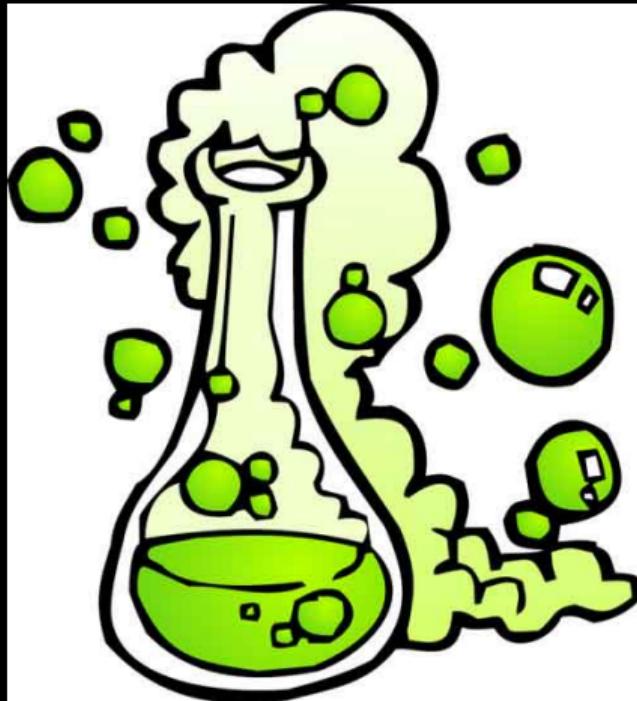
Sequential composition

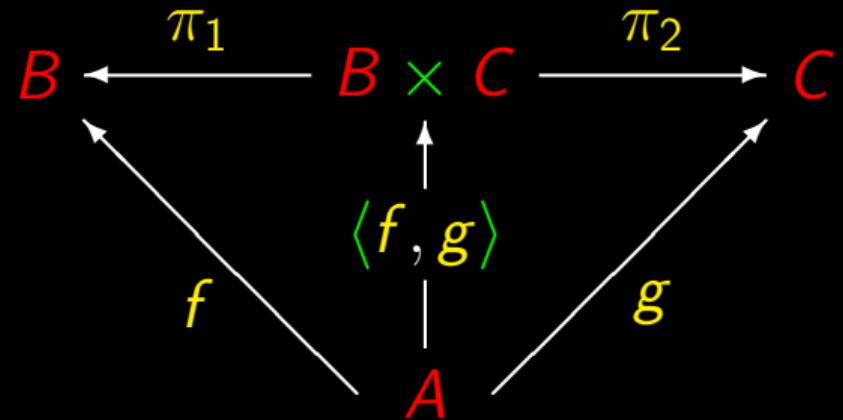
Parallel composition (synchronous)

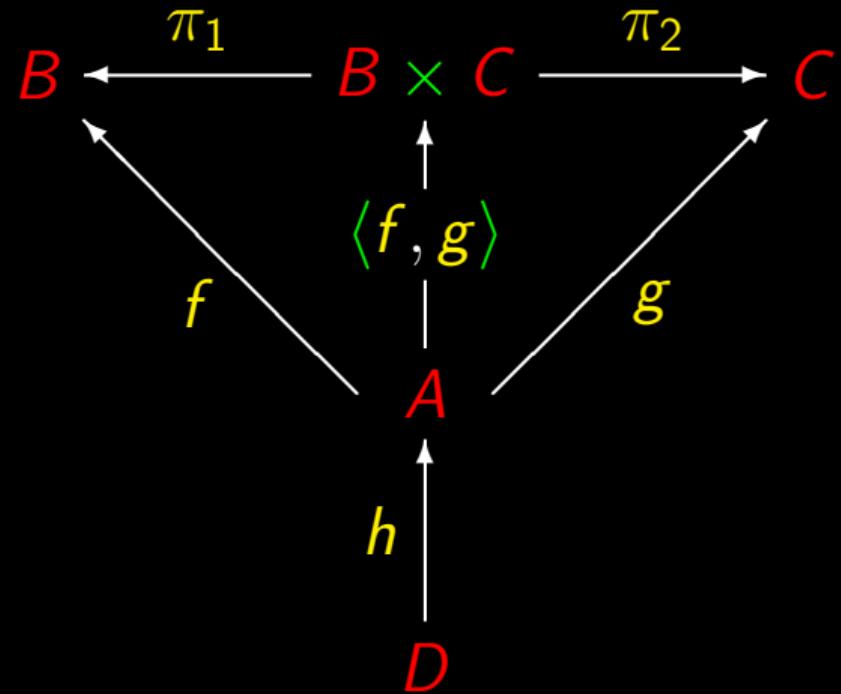
Parallel composition (asynchronous)

Compositional programming

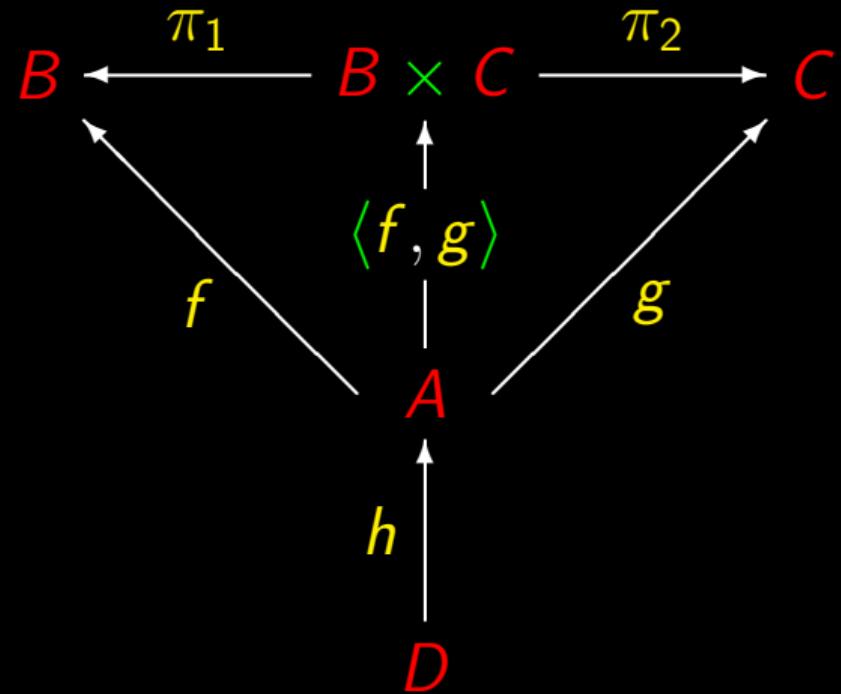
Calculus?

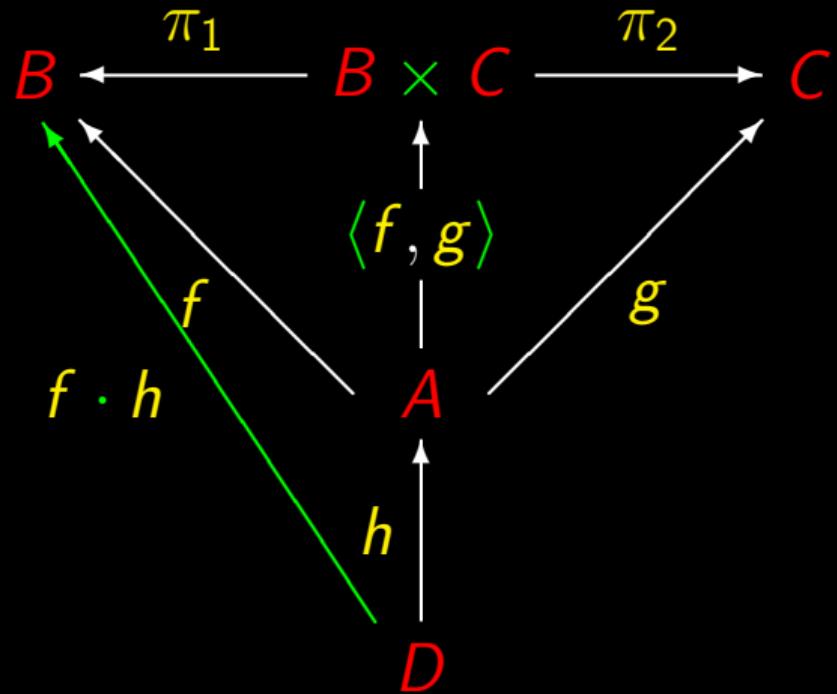


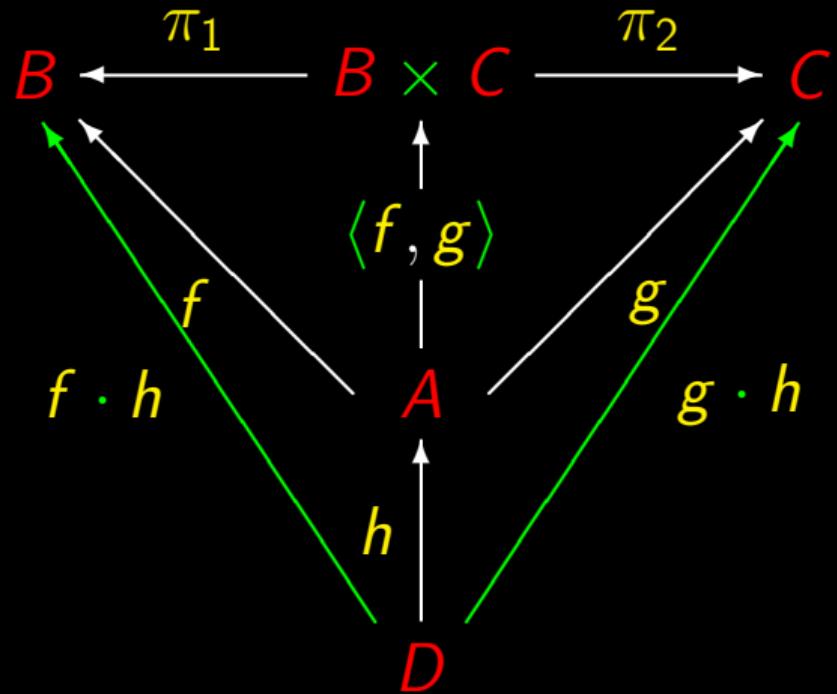


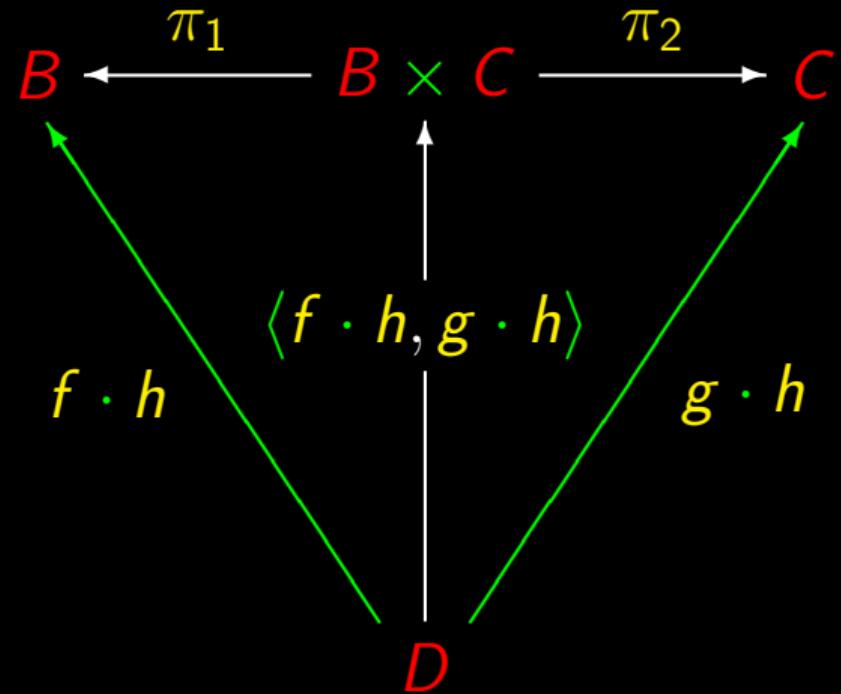


$$\begin{array}{ccccc} & & \pi_1 & & \\ & B & \xleftarrow{\hspace{1cm}} & B \times C & \xrightarrow{\hspace{1cm}} C \\ & & \uparrow & & \\ & & \langle f, g \rangle \cdot h & & \\ & & \downarrow & & \\ & & D & & \end{array}$$







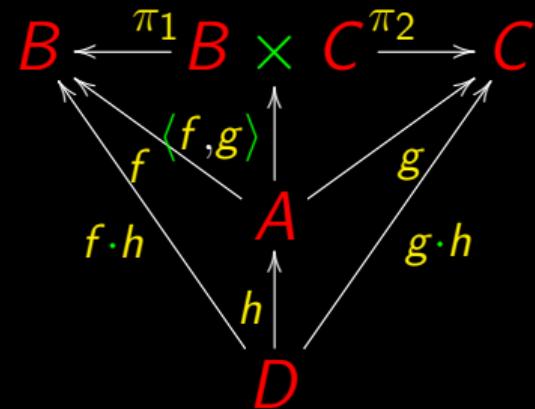


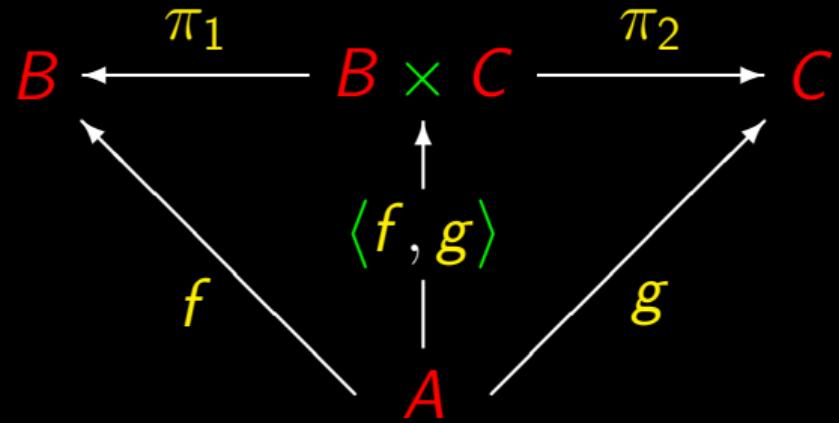
$$B \xleftarrow{\pi_1} B \times C \xrightarrow{\pi_2} C$$

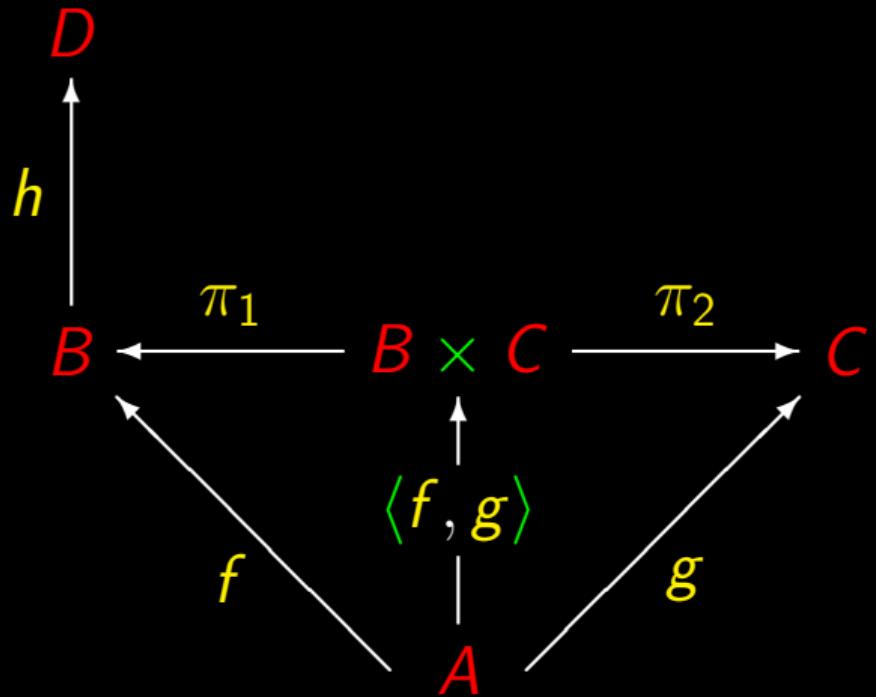
$$\begin{array}{c} \langle f, g \rangle \cdot h = \langle f \cdot h, g \cdot h \rangle \\ \downarrow \\ D \end{array}$$

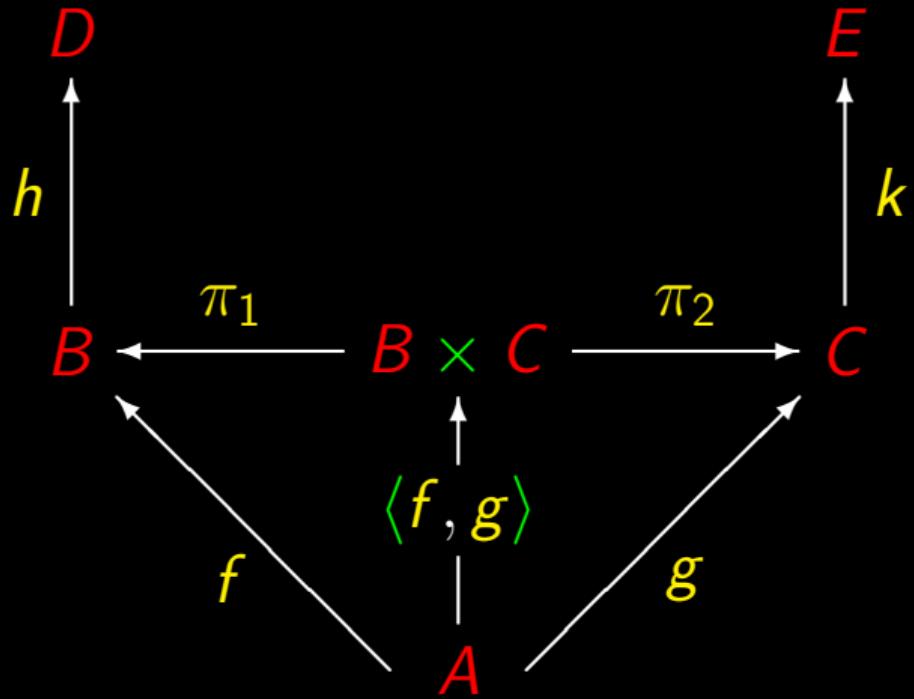
\times -Fusion

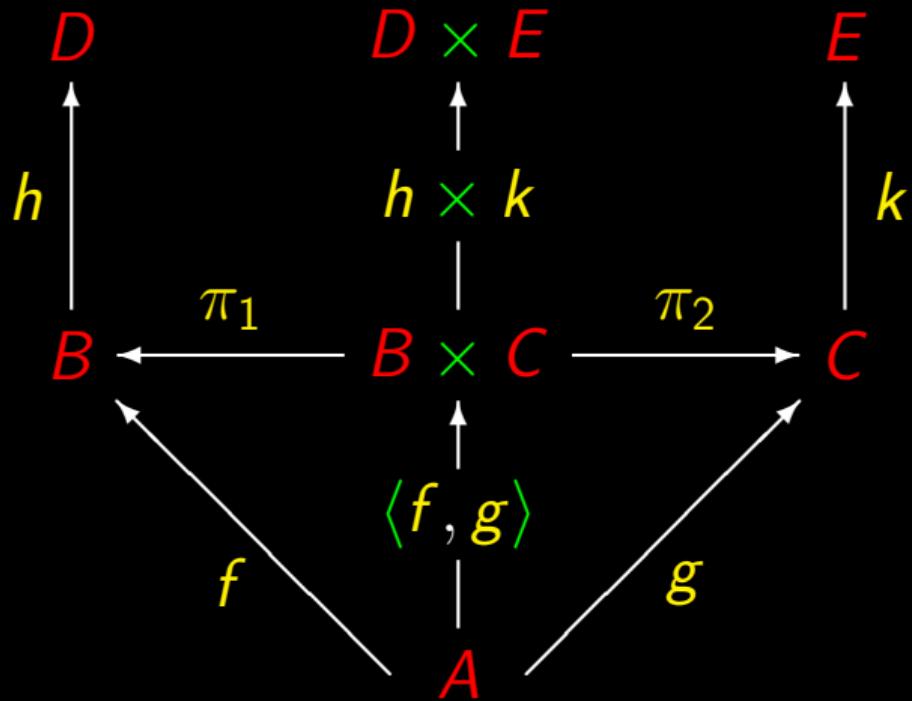
$$\langle f, g \rangle \cdot h = \langle f \cdot h, g \cdot h \rangle \quad (2.26)$$

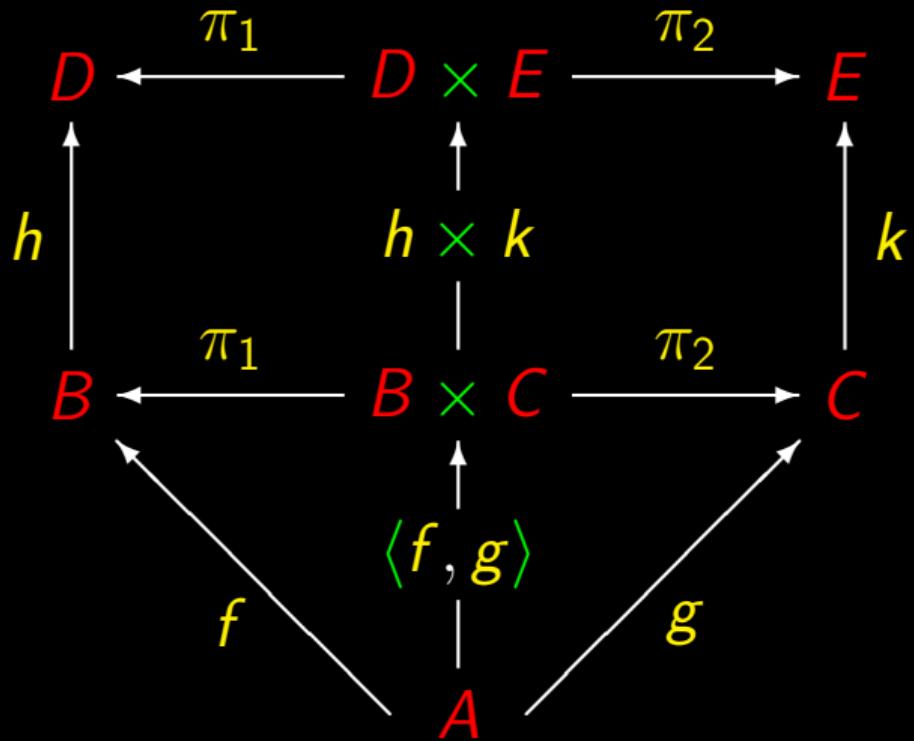


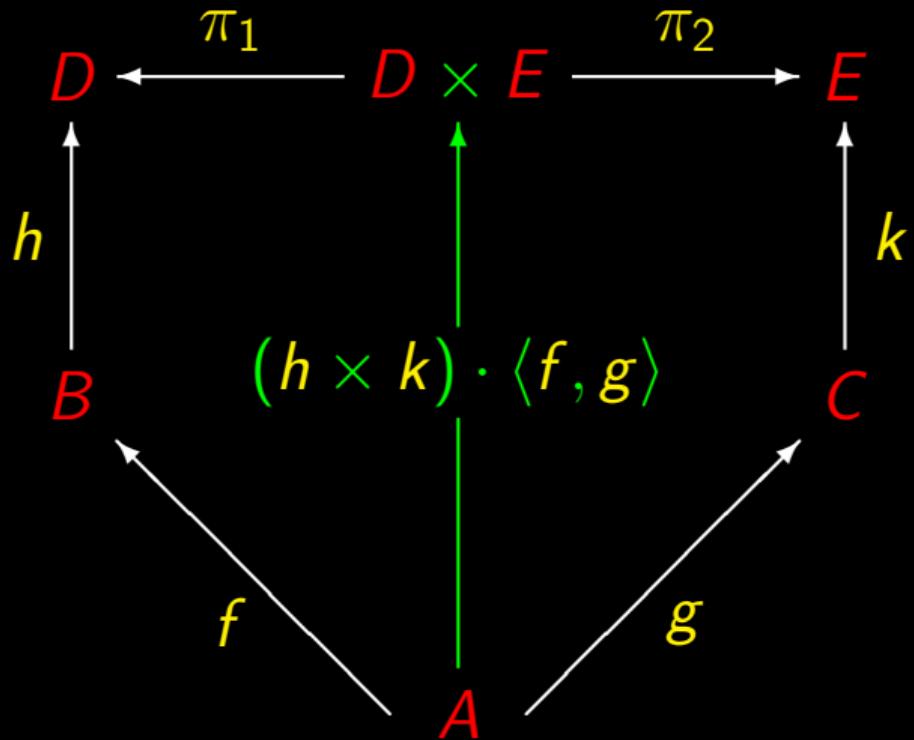


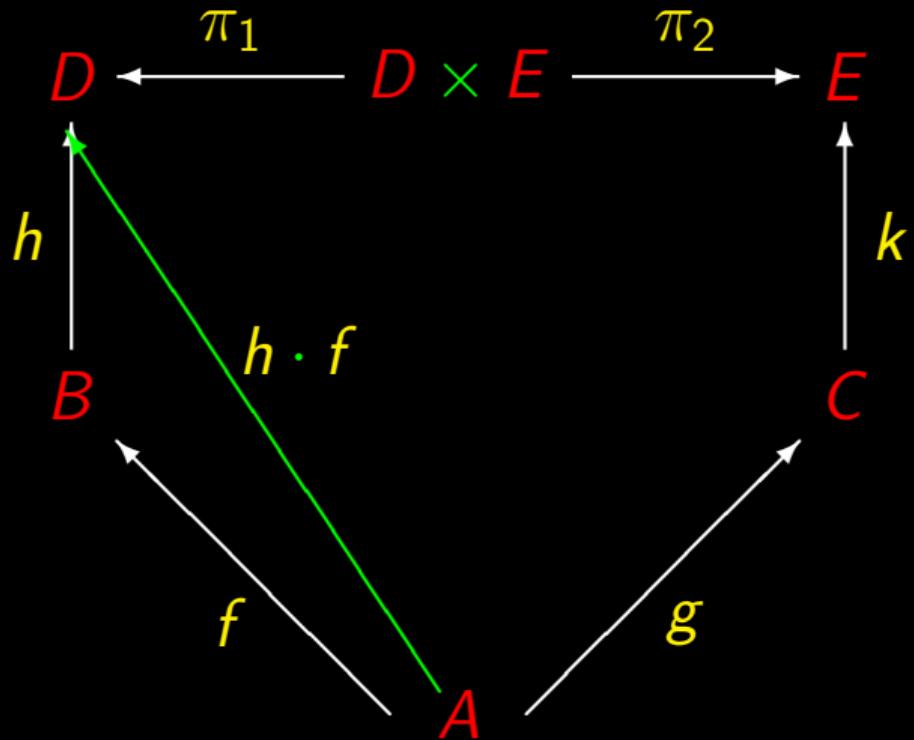


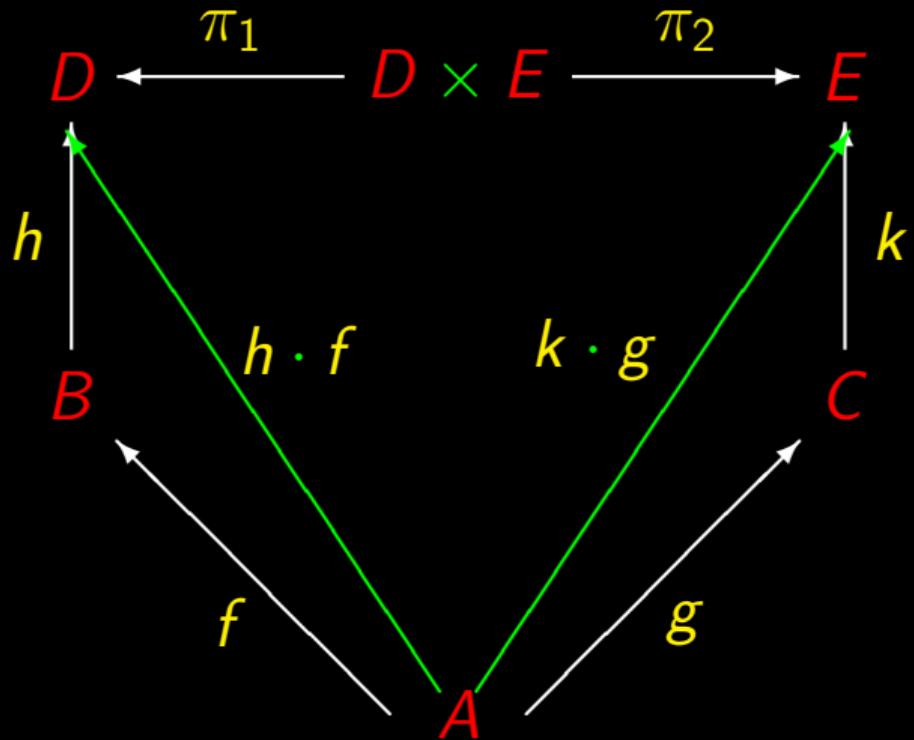


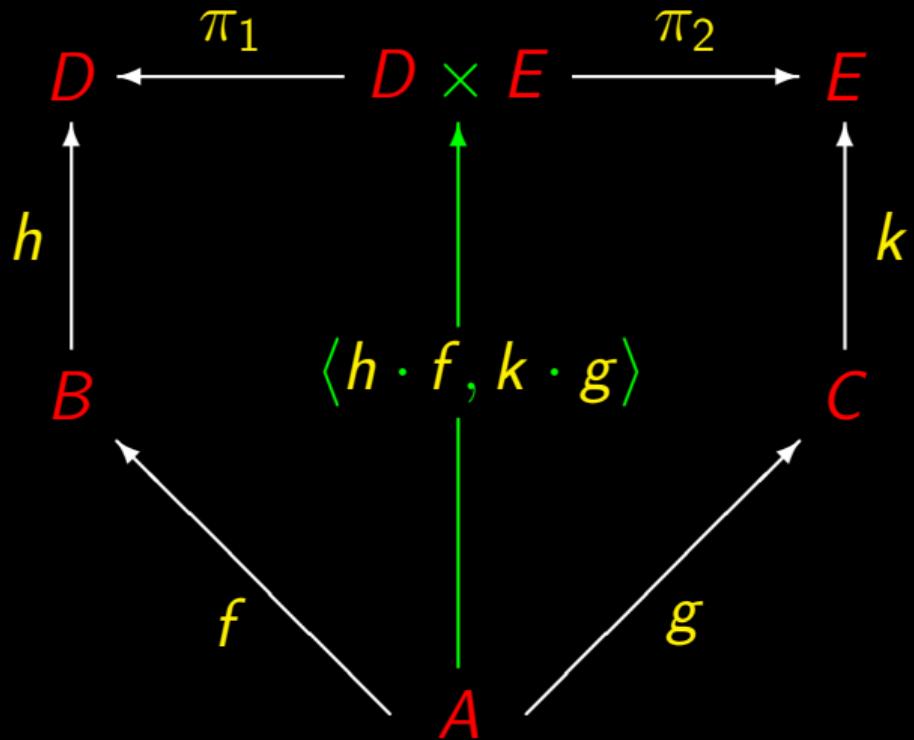


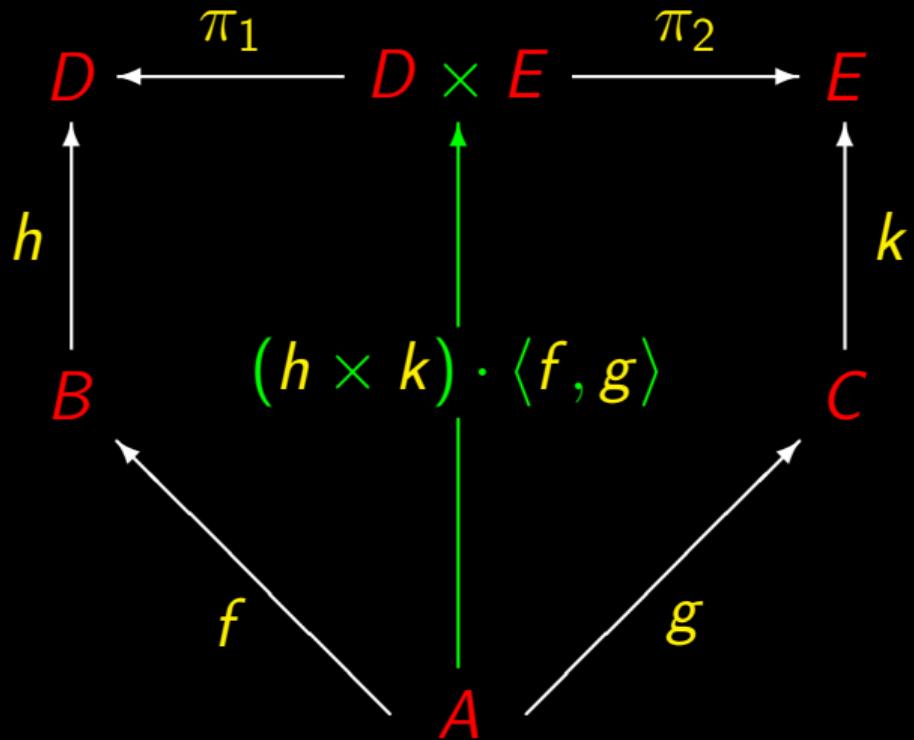






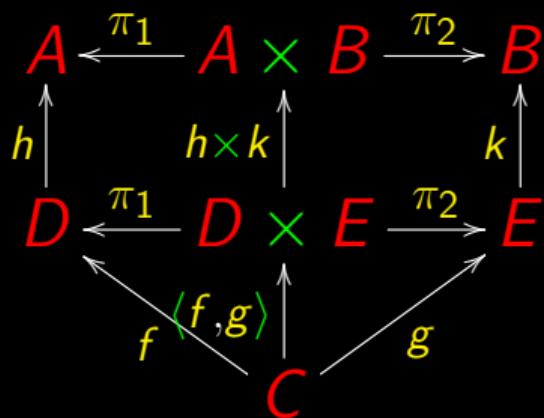


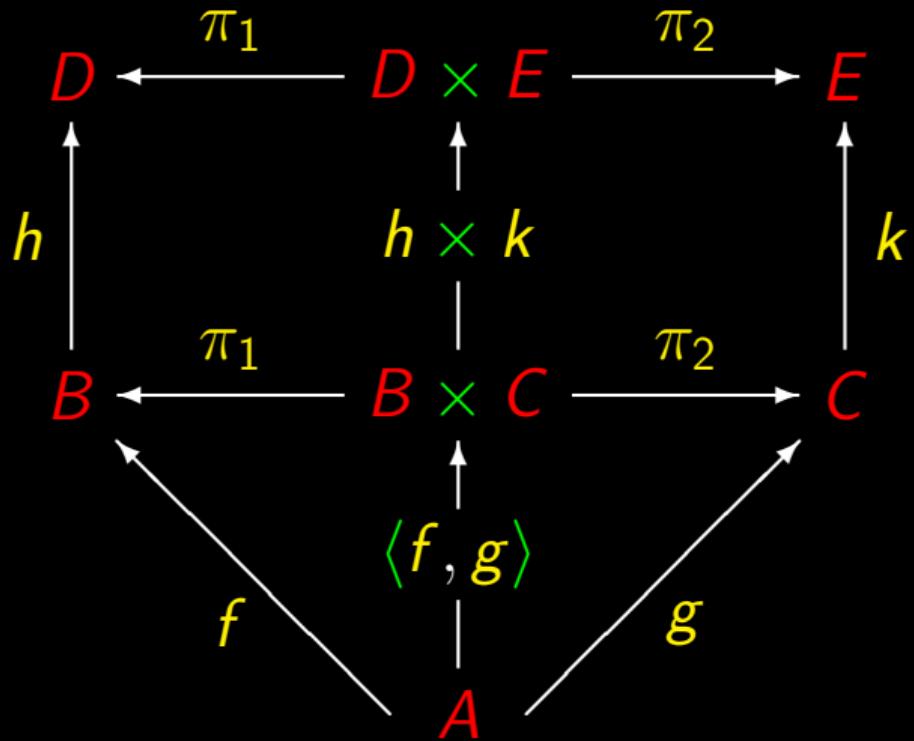


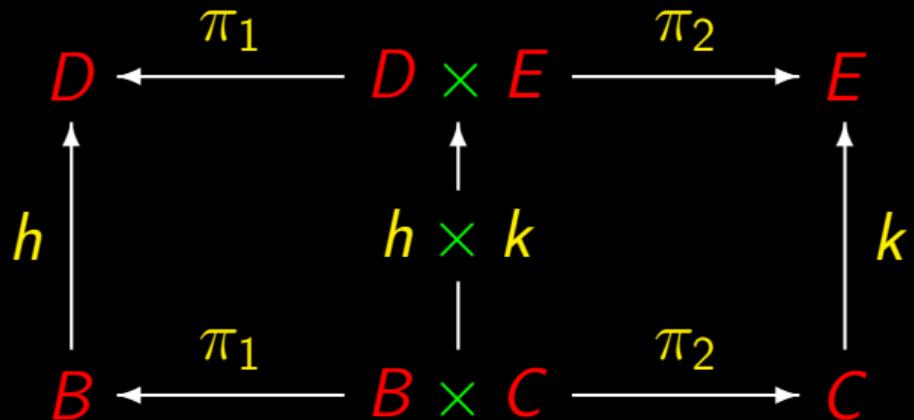


\times -Absorption

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

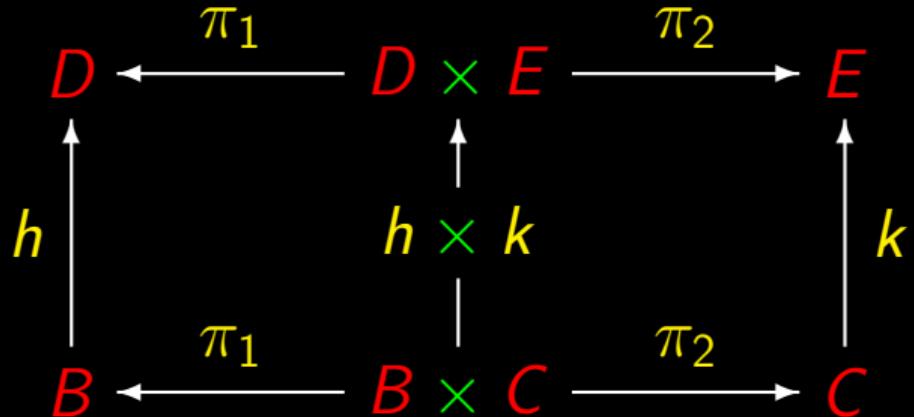






$$\begin{array}{ccccc}
 & \pi_1 & & \pi_2 & \\
 D & \xleftarrow{\hspace{1cm}} & D \times E & \xrightarrow{\hspace{1cm}} & E \\
 h \uparrow & & h \times k & & k \uparrow \\
 B & \xleftarrow{\hspace{1cm}} & B \times C & \xrightarrow{\hspace{1cm}} & C
 \end{array}$$

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



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

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

Natural- π_1 , natural- π_2

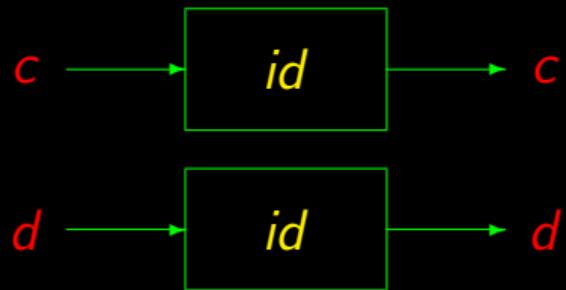
$$\pi_1 \cdot (h \times k) = h \cdot \pi_1 \quad (2.28)$$

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

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

Functor-*id*-×

$$id \times id = id \quad (2.31)$$



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

\times -Reflexion

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

Two laws still missing

x-Reflexion

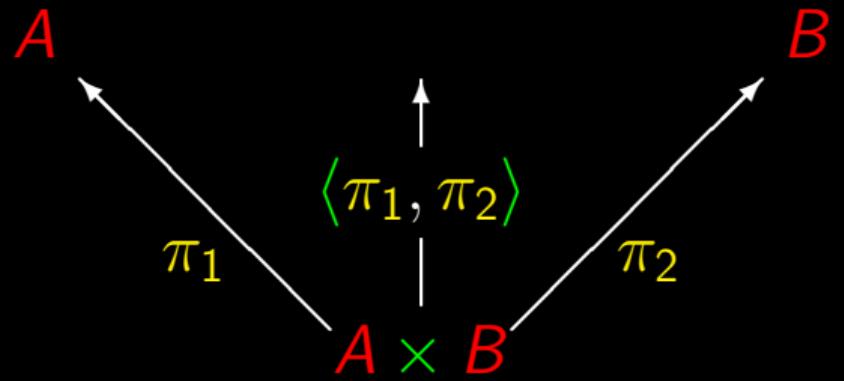
$$\langle \pi_1, \pi_2 \rangle = id \quad (2.32)$$

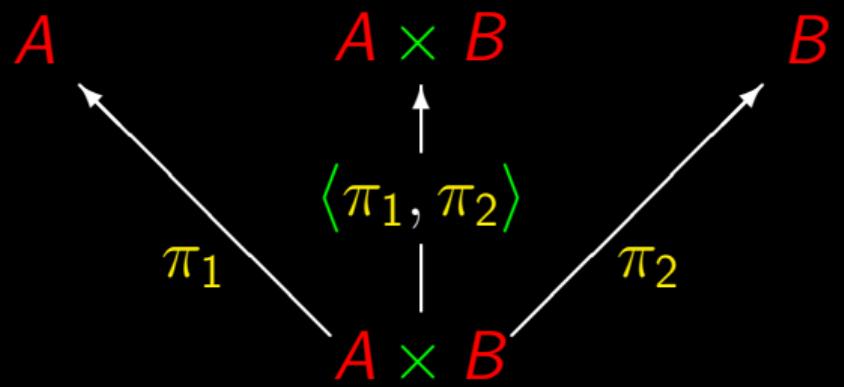
x-Eq

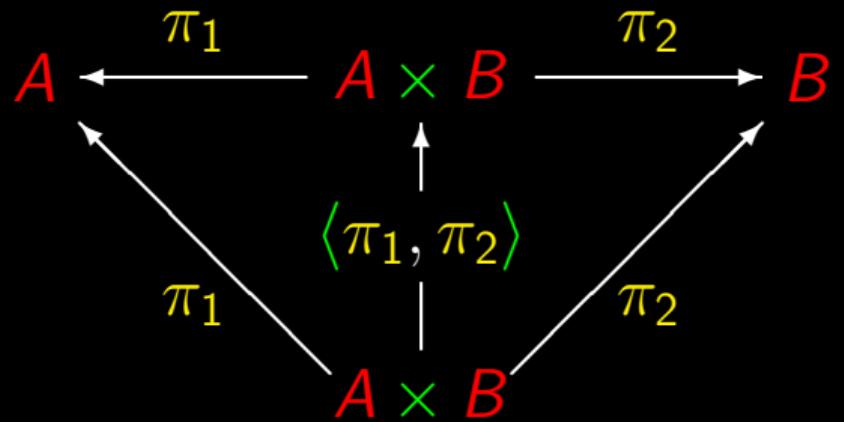
$$\langle i, j \rangle = \langle f, g \rangle \Leftrightarrow \begin{cases} i = f \\ j = g \end{cases} \quad (2.64)$$

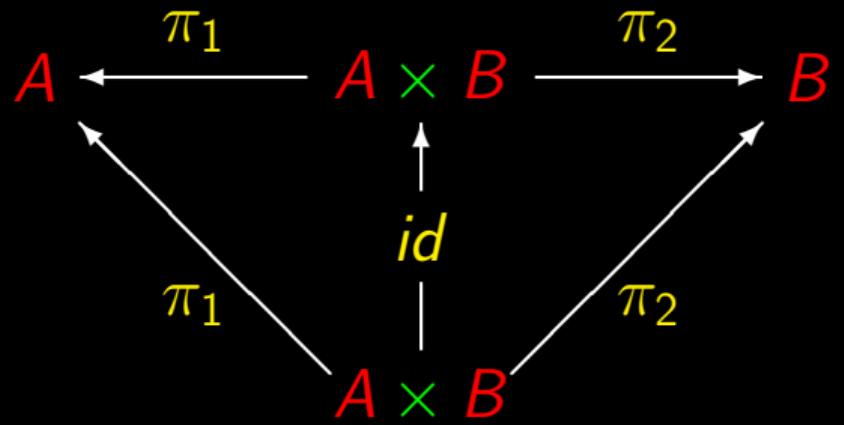
$$\begin{array}{c} \uparrow \\ \langle \pi_1, \pi_2 \rangle \\ \downarrow \end{array}$$

$$\begin{array}{ccc} A & & \langle \pi_1, \pi_2 \rangle \\ \swarrow & & \uparrow \\ \pi_1 & & | \\ & & A \times B \end{array}$$









Finally the more important...

Recall **X-cancellation**:

$$\begin{array}{ccccc} & & B \times C & & \\ & \swarrow & \uparrow \langle f, g \rangle & \searrow & \\ B & \xleftarrow{\pi_1} & & \xrightarrow{\pi_2} & C \\ f \quad & & & & g \\ & \searrow & \downarrow & \swarrow & \\ & & A & & \end{array}$$

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

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

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

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

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

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

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

\times -Universal

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

\times -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”

\times -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!

$$\left\{ \begin{array}{l} x = 2y \\ z = \frac{y}{3} \\ x + y + z = 10 \end{array} \right. \Leftrightarrow \left\{ \begin{array}{l} x = 6 \\ z = 1 \\ y = 3 \end{array} \right.$$

Equations!

Problem

Solve the equation

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

for f and g .

Equations!

Calculation

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!

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

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:

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

\times -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

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

$$\Leftrightarrow \quad \left\{ \begin{array}{l} \text{---} \\ \text{---} \end{array} \right. \text{---} \left. \begin{array}{l} \text{---} \\ \text{---} \end{array} \right\} \text{---} \left. \begin{array}{l} \text{---} \\ \text{---} \end{array} \right\}$$

\times -universal

$$\left\{ \begin{array}{l} \pi_1 \cdot \langle h, k \rangle = f \\ \pi_2 \cdot \langle h, k \rangle = g \end{array} \right.$$

Calculation

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

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

$$\left\{ \begin{array}{l} \pi_1 \cdot \langle h, k \rangle = f \\ \pi_2 \cdot \langle h, k \rangle = g \end{array} \right.$$

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

$$\left\{ \begin{array}{l} h = f \\ k = g \end{array} \right.$$

Calculation

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

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

$$\left\{ \begin{array}{l} \pi_1 \cdot \langle h, k \rangle = f \\ \pi_2 \cdot \langle h, k \rangle = g \end{array} \right.$$

x-Eq !



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

$$\left\{ \begin{array}{l} h = f \\ k = g \end{array} \right.$$

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

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

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

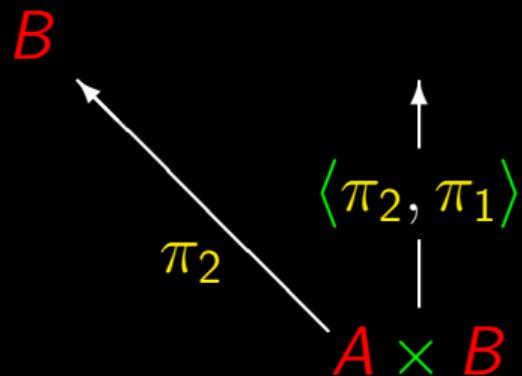
$$\langle \pi_1, \pi_2 \rangle = id$$

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

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

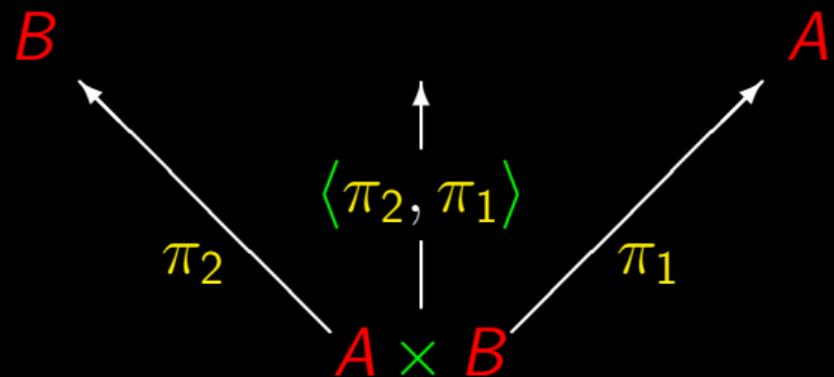
$$\langle \pi_1, \pi_2 \rangle = id$$

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



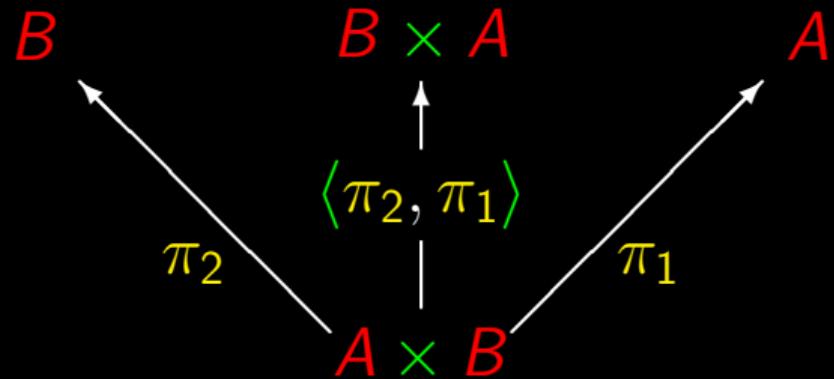
$$\langle \pi_1, \pi_2 \rangle = id$$

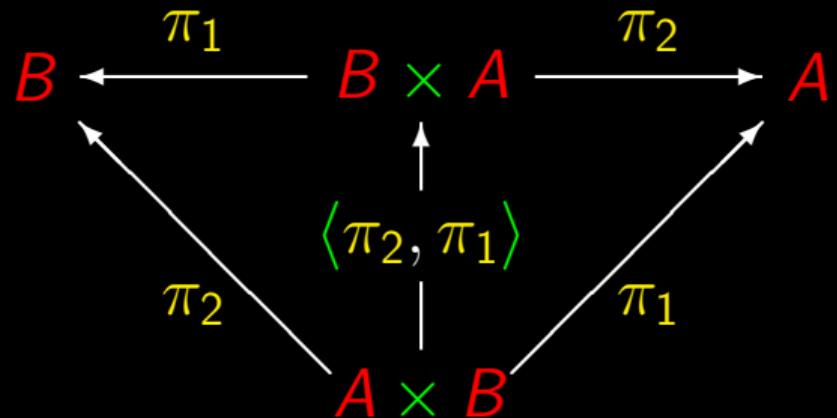
$$\langle \pi_2, \pi_1 \rangle ?$$



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

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



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

Problem

Solve

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

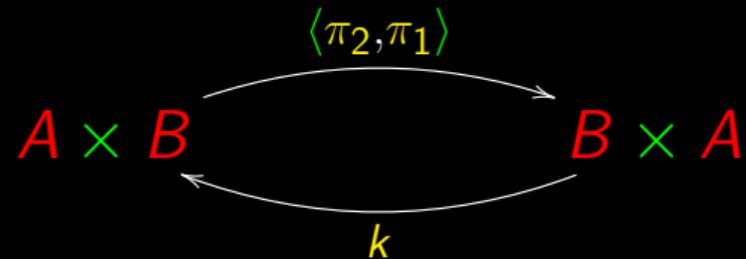
for k

Problem

Solve

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

for k



Calculation

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

Calculation

$$\langle \pi_2, \pi_1 \rangle \cdot k = id$$
$$\Leftrightarrow \quad \left\{ \begin{array}{c} \text{---} \\ \text{x-fusion} \end{array} \right\}$$
$$\langle \pi_2 \cdot k, \pi_1 \cdot k \rangle = id$$

Calculation

$$\begin{aligned} & \langle \pi_2, \pi_1 \rangle \cdot k = id \\ \Leftrightarrow & \quad \left\{ \begin{array}{l} \text{x-fusion} \end{array} \right\} \\ & \langle \pi_2 \cdot k, \pi_1 \cdot k \rangle = id \\ \Leftrightarrow & \quad \left\{ \begin{array}{l} \text{x-universal} \end{array} \right\} \\ & \left\{ \begin{array}{l} \pi_2 \cdot k = \pi_1 \\ \pi_1 \cdot k = \pi_2 \end{array} \right. \end{aligned}$$

Calculation

$$\begin{aligned} & \langle \pi_2, \pi_1 \rangle \cdot k = id \\ \Leftrightarrow & \left\{ \begin{array}{l} \text{---} \\ \text{x-fusion} \\ \text{---} \end{array} \right\} & \Leftrightarrow & \left\{ \begin{array}{l} \text{---} \\ \text{trivial} \\ \text{---} \end{array} \right\} \\ & \langle \pi_2 \cdot k, \pi_1 \cdot k \rangle = id \\ \Leftrightarrow & \left\{ \begin{array}{l} \text{---} \\ \text{x-universal} \\ \text{---} \end{array} \right\} & & \left\{ \begin{array}{l} \pi_1 \cdot k = \pi_2 \\ \pi_2 \cdot k = \pi_1 \end{array} \right. \\ & \left\{ \begin{array}{l} \pi_2 \cdot k = \pi_1 \\ \pi_1 \cdot k = \pi_2 \end{array} \right. \end{aligned}$$

Calculation

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

\Leftrightarrow { x-fusion }

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

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

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

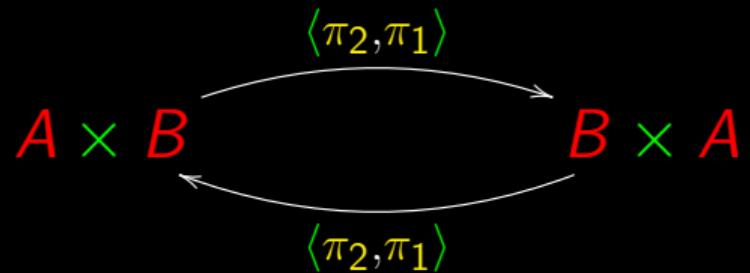
$\Leftrightarrow \{ \text{ trivial } \}$

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

\Leftrightarrow { x-universal }

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

Swap



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

Swap

$$\begin{array}{ccc} & \langle \pi_2, \pi_1 \rangle & \\ A \times B & \swarrow & \searrow \\ & \langle \pi_2, \pi_1 \rangle & \end{array}$$

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

$$swap \cdot swap = id$$

Até agora

$f \cdot g$

Sequential composition

Até agora

$f \cdot g$

$\langle f, g \rangle$

Sequential composition

Parallel composition

Até agora

$f \cdot g$

$\langle f, g \rangle$

Sequential composition

Parallel composition

Associativity

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

Até agora

$f \cdot g$

$\langle f, g \rangle$

Sequential composition

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

$f \cdot g$

$\langle f, g \rangle$

Sequential composition

Parallel composition

Associativity

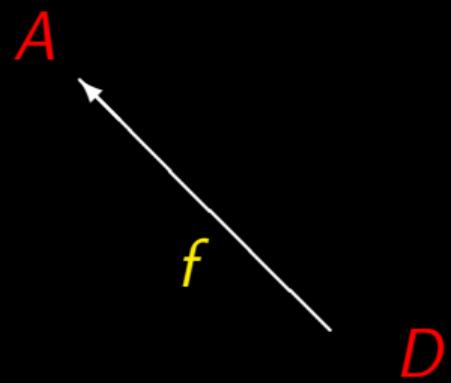
$$(f \cdot g) \cdot h = f \cdot (g \cdot h)$$

Associativity?

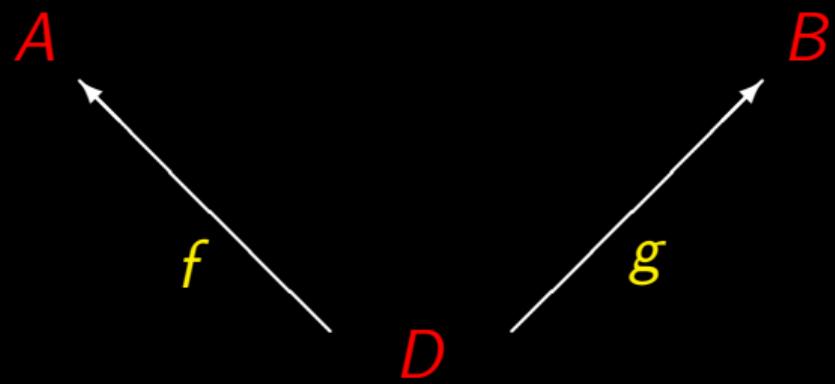
$$\langle\langle f, g \rangle, h \rangle = \langle f, \langle g, h \rangle \rangle ?$$

No! but...

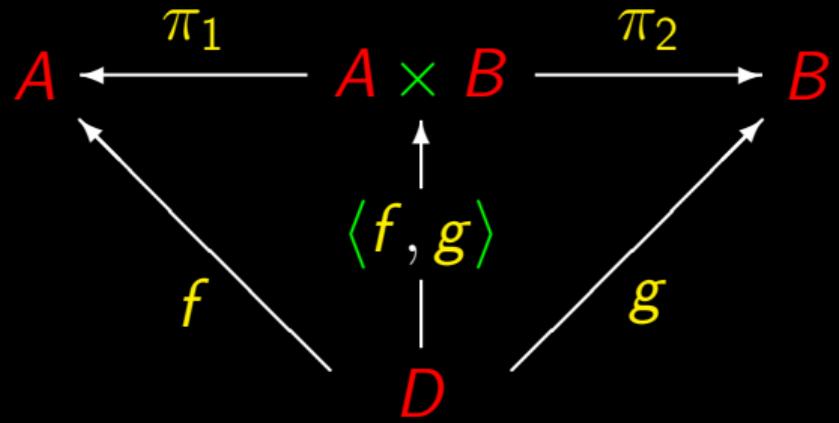
$$\langle\langle f, g \rangle, h \rangle$$



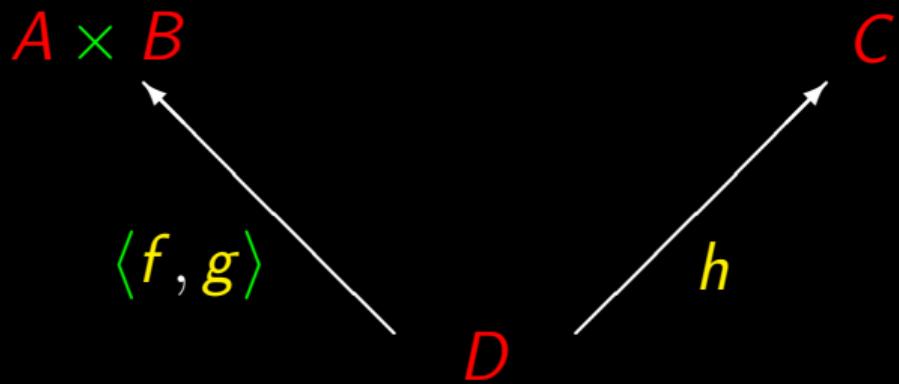
$$\langle \langle f, g \rangle, h \rangle$$



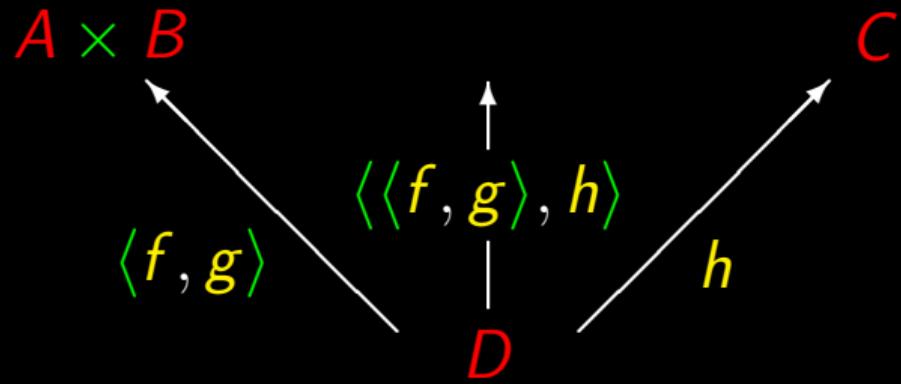
$$\langle \langle f, g \rangle, h \rangle$$



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



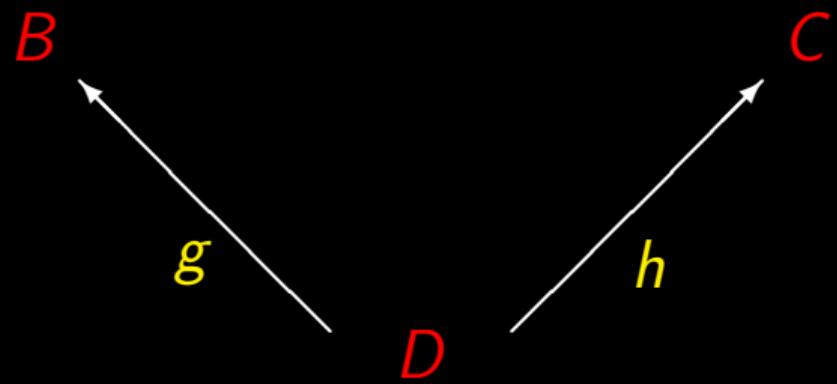
$$\langle \langle f, g \rangle, h \rangle$$



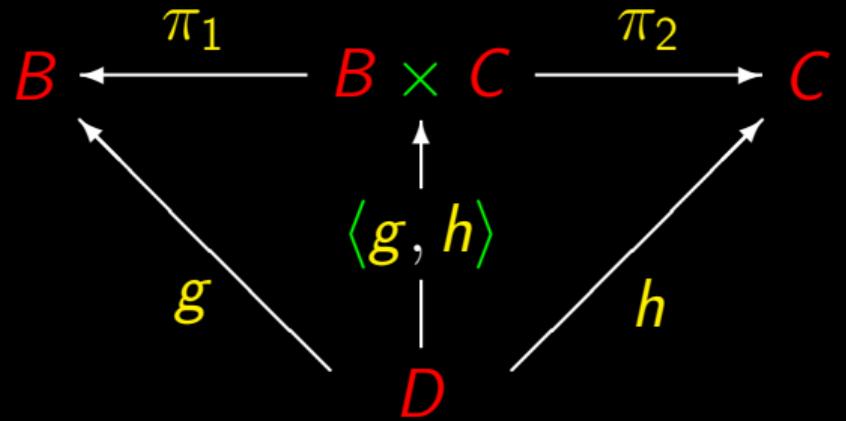
$$\langle \langle f, g \rangle, h \rangle$$

$$\begin{array}{ccccc} A \times B & \xleftarrow{\pi_1} & (A \times B) \times C & \xrightarrow{\pi_2} & C \\ \swarrow & & \uparrow & & \searrow \\ \langle f, g \rangle & & \langle \langle f, g \rangle, h \rangle & & h \\ & & | & & \\ & & D & & \end{array}$$

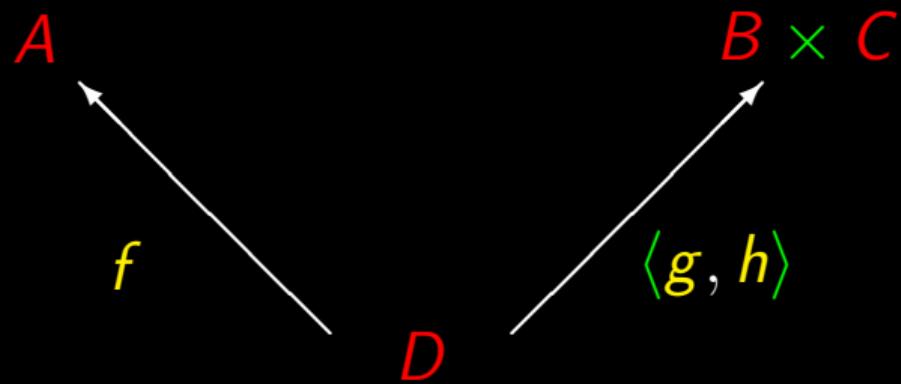
$$\langle f, \langle g, h \rangle \rangle$$



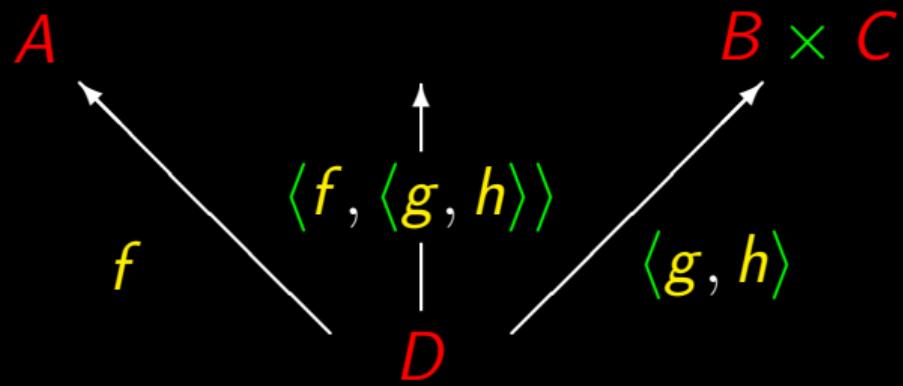
$$\langle f, \langle g, h \rangle \rangle$$



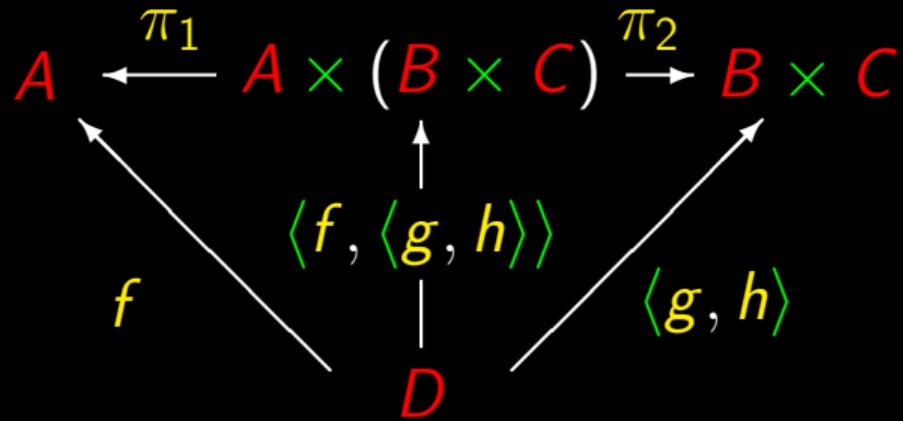
$$\langle f, \langle g, h \rangle \rangle$$



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



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



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

$$\begin{array}{ccccc} A \times B & \xleftarrow{\pi_1} & (A \times B) \times C & \xrightarrow{\pi_2} & C \\ \swarrow & & \uparrow & & \searrow \\ \langle f, g \rangle & & \langle \langle f, g \rangle, h \rangle & & h \\ & & | & & \\ & & D & & \end{array}$$

$$\begin{array}{ccc} & k & \\ (A \times B) \times C & \swarrow & \searrow A \times (B \times C) \\ \langle \langle f, g \rangle, h \rangle & & \langle f, \langle g, h \rangle \rangle \\ & D & \end{array}$$

$$\begin{array}{ccc} & k & \\ (A \times B) \times C & \swarrow & \searrow A \times (B \times C) \\ \langle \langle f, g \rangle, h \rangle & & \langle f, \langle g, h \rangle \rangle \\ & D & \end{array}$$

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

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

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

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

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

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



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

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

$$\begin{aligned} & \langle\langle f, g \rangle, h \rangle = id \\ \Leftrightarrow & \quad \left\{ \begin{array}{c} \text{ } \\ \text{ } \end{array} \right. \begin{array}{l} \text{ } \\ \text{ } \end{array} \left. \begin{array}{c} \text{ } \\ \text{ } \end{array} \right\} \\ & \left\{ \begin{array}{l} \pi_1 = \langle f, g \rangle \\ \pi_2 = h \end{array} \right. \end{aligned}$$

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

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

$$\left\{ \begin{array}{l} \pi_1 = \langle f, g \rangle \\ \pi_2 = h \end{array} \right.$$

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

$$\left\{ \begin{array}{l} \pi_1 \cdot \pi_1 = f \\ \pi_2 \cdot \pi_1 = g \\ \pi_2 = h \end{array} \right.$$

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...

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

Slighth improvement...

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

Slighth improvement...

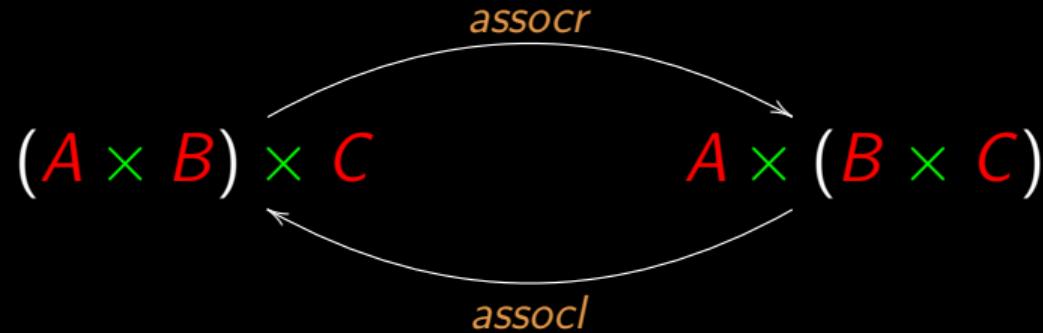
$$\begin{aligned} k &= \langle \pi_1 \cdot \pi_1, \langle \pi_2 \cdot \pi_1, \pi_2 \rangle \rangle \\ \Leftrightarrow &\quad \left\{ \begin{array}{l} \pi_2 = id \cdot \pi_2 \end{array} \right\} \\ k &= \langle \pi_1 \cdot \pi_1, \langle \pi_2 \cdot \pi_1, id \cdot \pi_2 \rangle \rangle \\ \Leftrightarrow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

$$\begin{array}{ccc} & \langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle & \\ (A \times B) \times C & \swarrow & \searrow \\ & D & \\ & \langle \langle f, g \rangle, h \rangle & \langle f, \langle g, h \rangle \rangle \end{array}$$

$$\begin{array}{ccc} & \langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle & \\ (A \times B) \times C & \swarrow & \searrow \\ & j & \\ & D & \end{array}$$

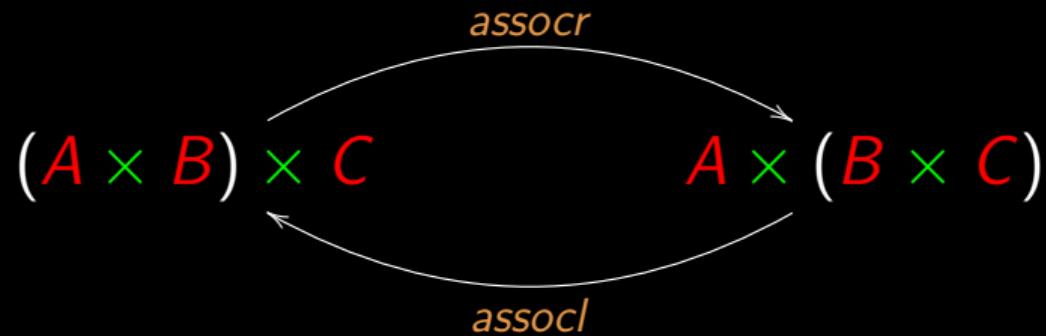
A commutative diagram illustrating a natural transformation between two ways of multiplying three objects. The top row consists of two objects: $(A \times B) \times C$ on the left and $A \times (B \times C)$ on the right. Above this row is a green bracketed expression $\langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle$, which represents a morphism from the left object to the right. The bottom row consists of two objects: $\langle \langle f, g \rangle, h \rangle$ on the left and $\langle f, \langle g, h \rangle \rangle$ on the right. Below this row is a red bracketed expression D , which represents a morphism from the left object to the right. A curved arrow labeled j connects the two rows, indicating a commutative relationship between the two multiplication operations.

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

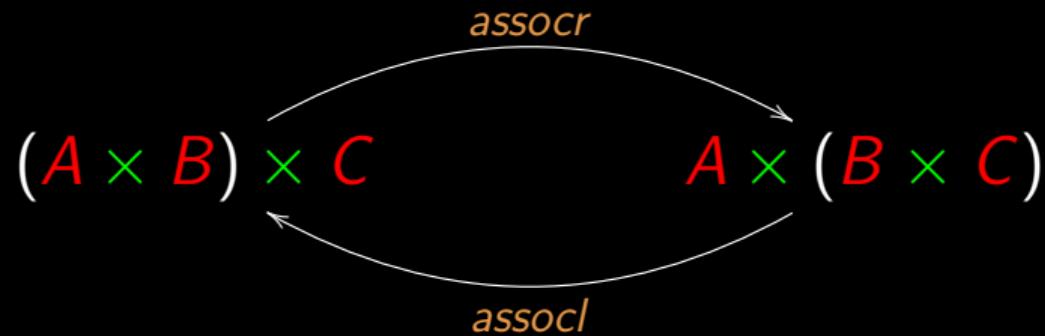


$$assocr = \langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle$$

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



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



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

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

Isomorphism

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

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

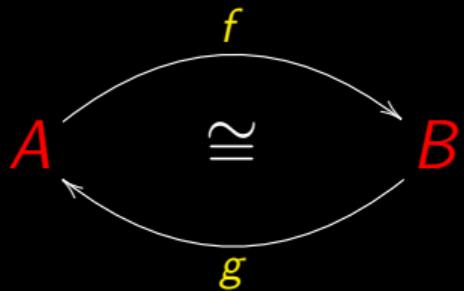
$$\text{assocl} \cdot \text{assocr} = \text{id}$$

Isomorphism

$$\begin{array}{ccc} & \text{swap} & \\ A \times B & \cong & B \times A \\ & \text{swap} & \end{array}$$

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

Isomorphism



$$f \cdot g = id$$

$$g \cdot f = id$$

Isomorphism

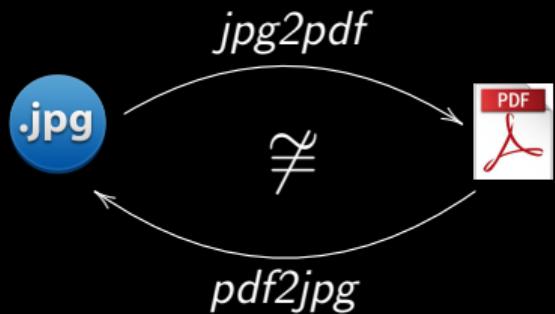
$\underbrace{iso}_{\text{the same}} (\iota\sigma o)$

Isomorphism

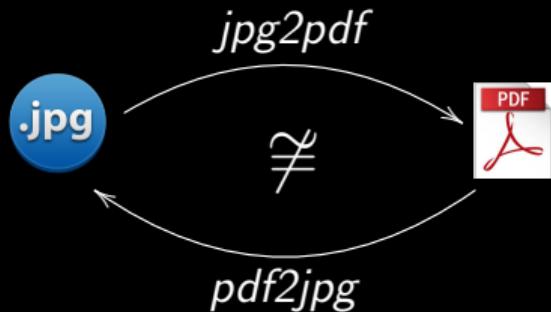
$\underbrace{\text{iso}}_{\text{the same}} (\iota\sigma o) + \underbrace{\text{morphism}}_{\text{shape}} (\mu\text{the}\rho\phi\iota\sigma\mu o\zeta)$

“Similar shape”

Practical problem!



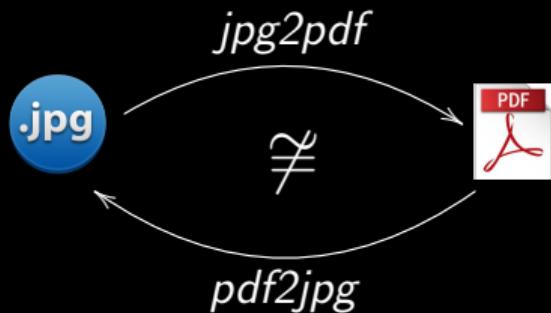
Practical problem!



$$jpg2pdf \cdot pdf2jpg \neq id$$

$$pdf2jpg \cdot jpg2pdf \neq id$$

Practical problem!



$$jpg2pdf \cdot pdf2jpg \neq id$$

$$pdf2jpg \cdot jpg2pdf \neq id$$

Format conversion

Need

$$\begin{array}{ccc} A & & \\ \downarrow f & & \\ B & & \end{array}$$

Format conversion

Need



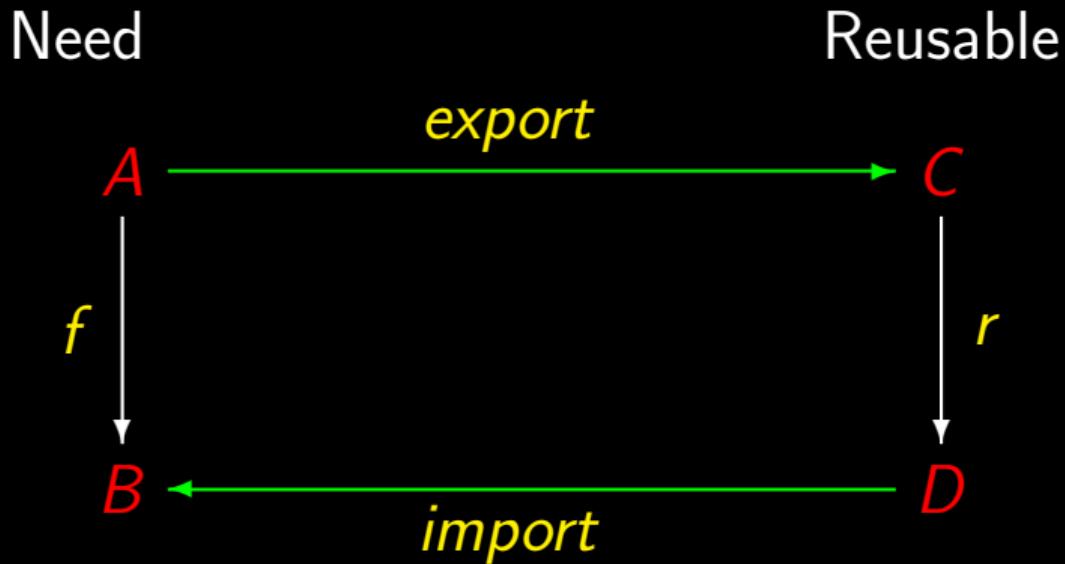
Reusable



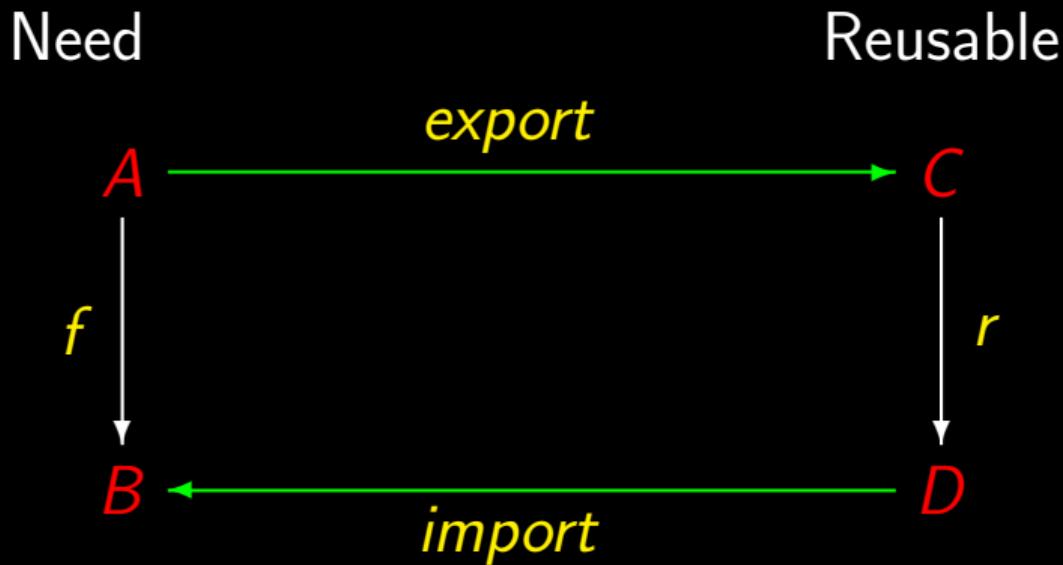
Format conversion



Format conversion

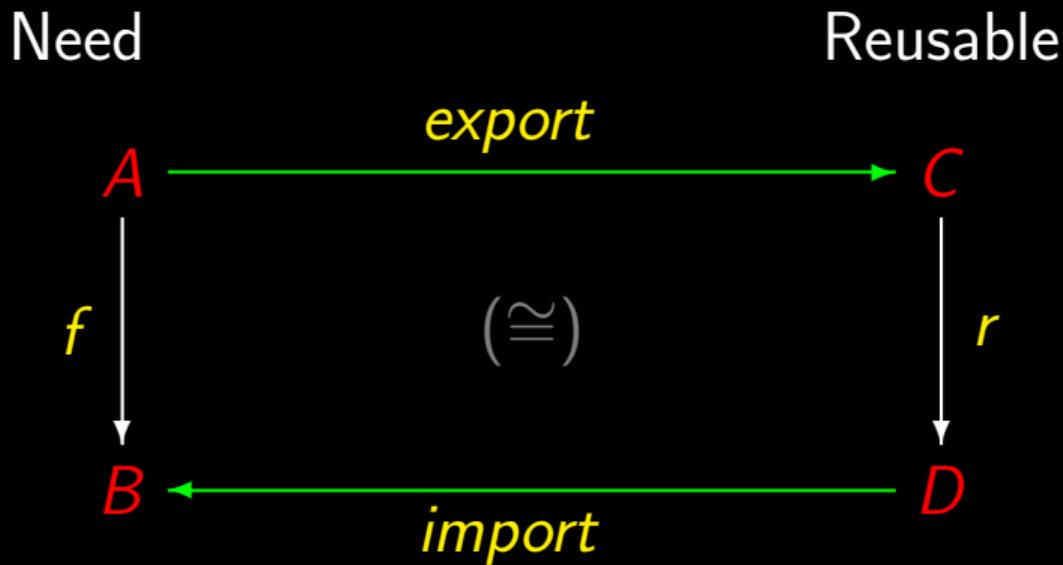


Format conversion



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

Format conversion



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

By the way — *swap*

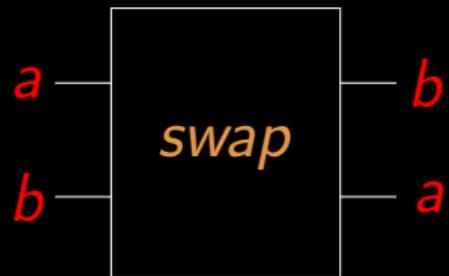
$$\begin{array}{ccc} & \text{swap} & \\ A \times B & \cong & B \times A \\ \text{swap} & & \end{array}$$

Isomorphisms are **reversible** computations

By the way — *swap*

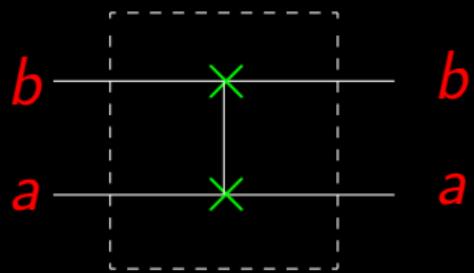


By the way — *swap*



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

By the way — swap



Cálculo de Programas

Class T03

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 → Address

Iden = CC ∪ 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

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

starting 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,

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

upon further defining:

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

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

Disjoint union

Altogether:

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

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

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

Disjoint union

Altogether:

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

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

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

$$\begin{array}{c} A + B \\ | \\ m \\ \downarrow \\ C \end{array}$$

Disjoint union

Altogether:

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

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

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

$$\begin{array}{ccc} A & \xrightarrow{i_1} & A + B \\ & & | \\ & & m \\ & & \downarrow \\ & & C \end{array}$$

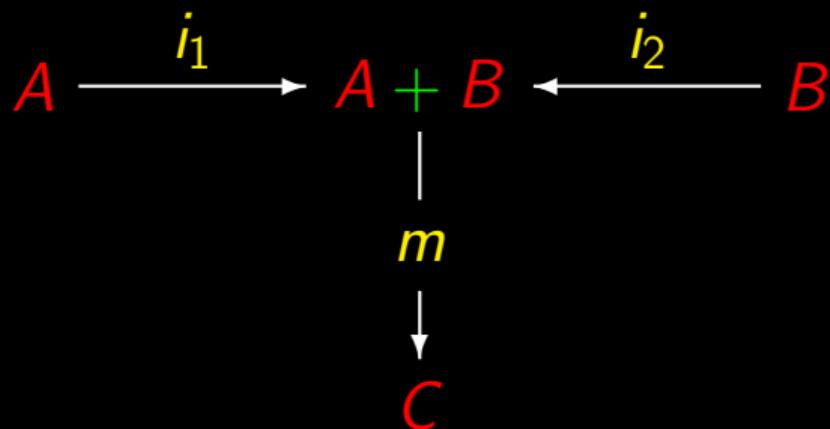
Disjoint union

Altogether:

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

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

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



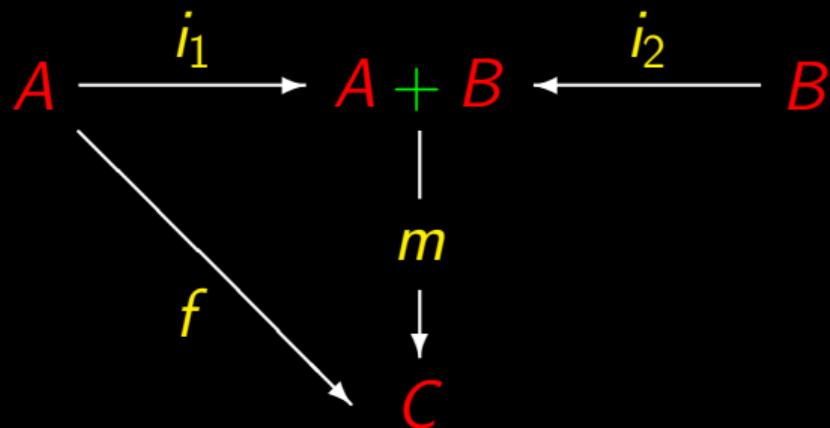
Disjoint union

Altogether:

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

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

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



$$f = m \cdot i_1$$

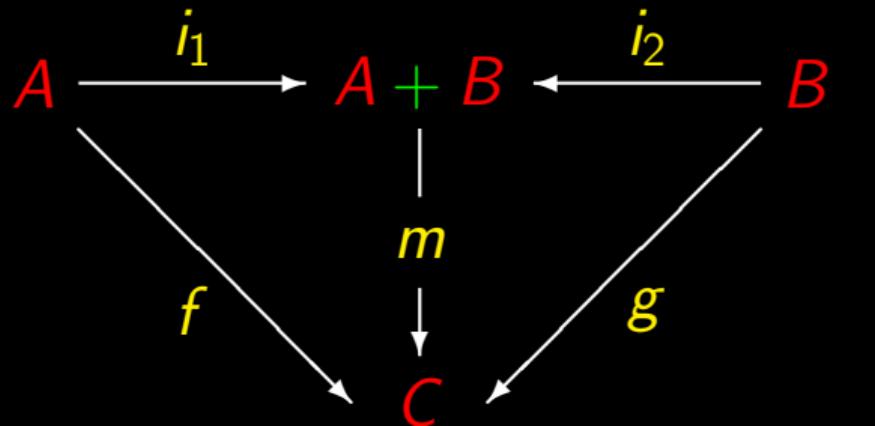
Disjoint union

Altogether:

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

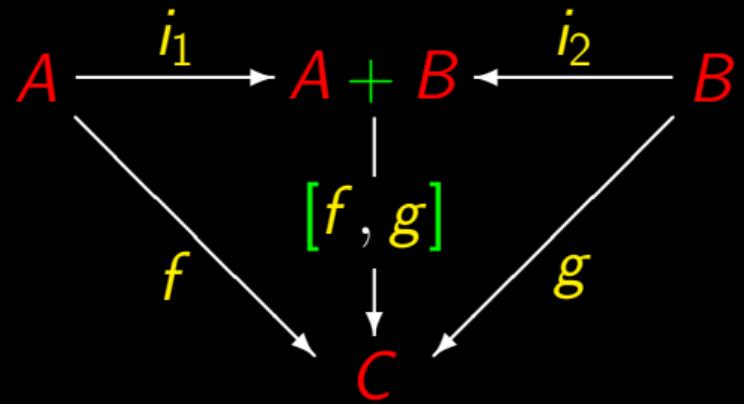
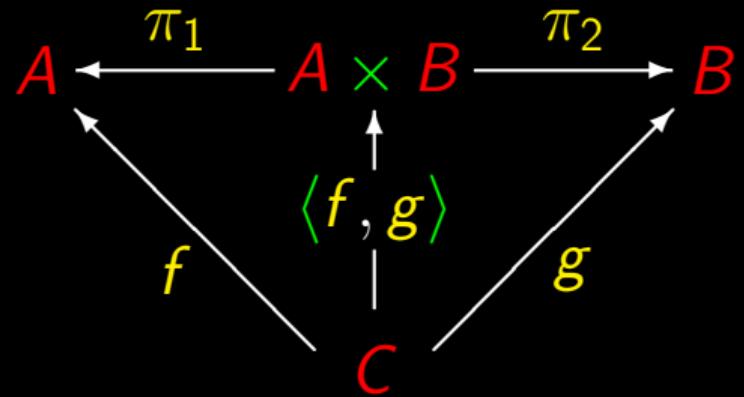
$$i_1 : A \rightarrow A + B$$

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



$$f = m \cdot i_1$$

$$g = m \cdot i_2$$



+ -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

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

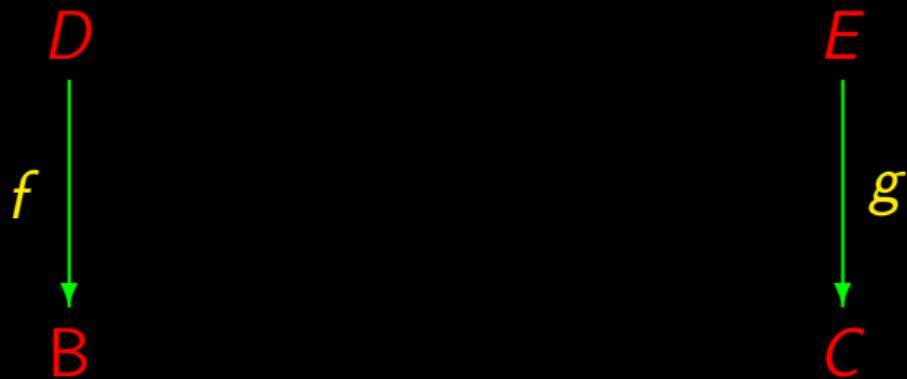
$$[f, g] x = \begin{cases} x = i_1 a \Rightarrow f a \\ x = i_2 b \Rightarrow g b \end{cases}$$

$$f + g \quad ?$$

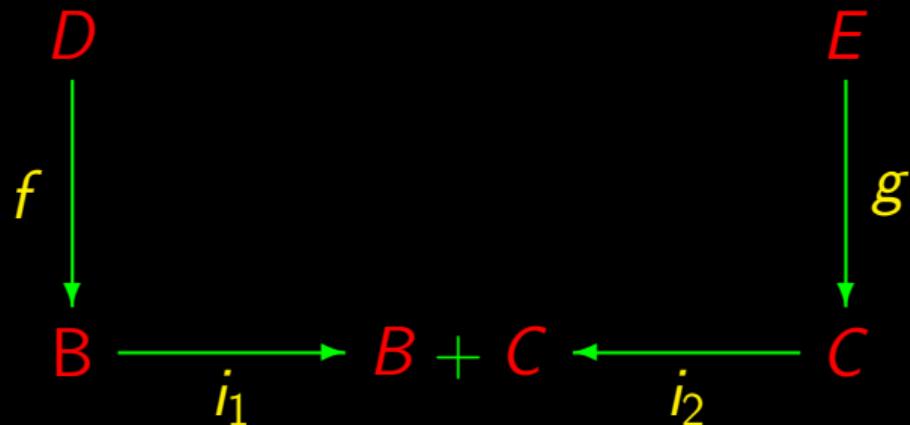
Sum of two functions

$$D \xrightarrow{f} B$$

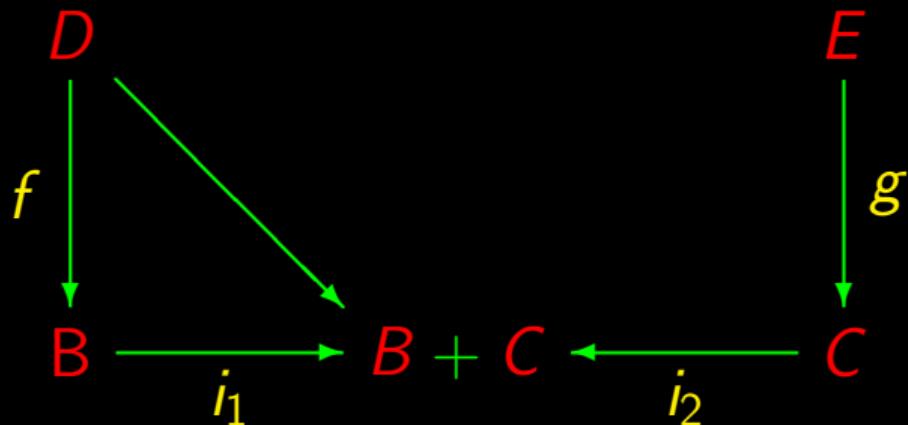
Sum of two functions



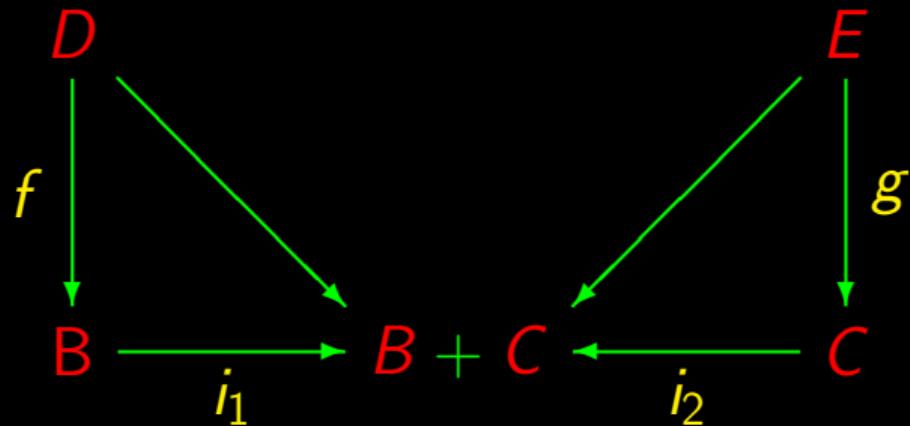
Sum of two functions



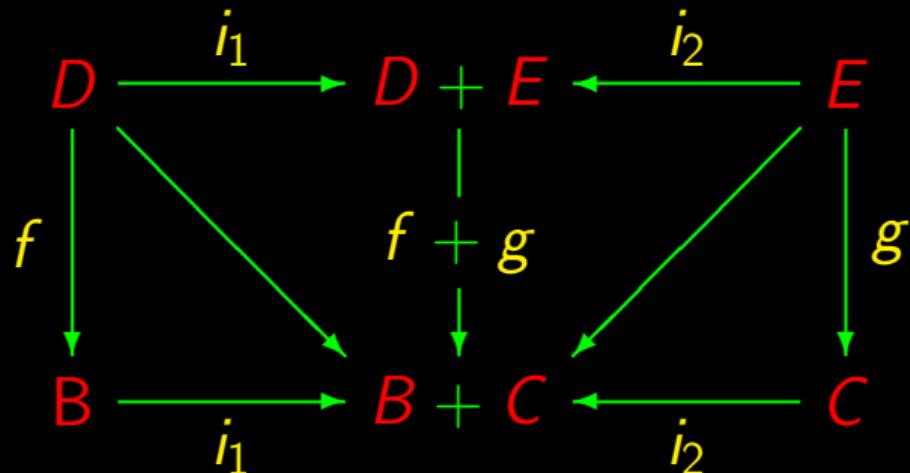
Sum of two functions



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

Coproduct laws

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

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

Coproduct laws

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

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

+**-Functor-id** $id + id = id$ (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

$f \cdot g$

$\langle f, g \rangle$

$f \times g$

$[f, g]$

$f + g$

Sequential composition

Parallel composition

Product composition

Alternative composition

Coproduct (function sum)

Summary

$f \cdot g$

$\langle f, g \rangle$

$f \times g$

$[f, g]$

$f + g$

Sequential composition

Parallel composition

Product composition

Alternative composition

Coproduct (function sum)

Compositional programming



What about...

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

What about...

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

...?

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

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...

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

...?

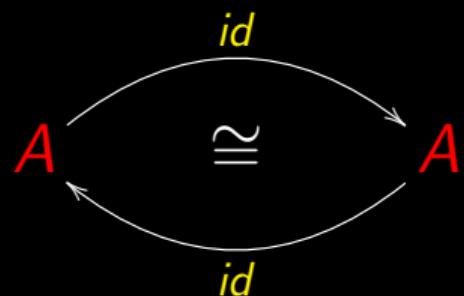
$$A \times (B + C) \stackrel{?}{\cong} A \times B + A \times C$$

Moreover, any “0” such that

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

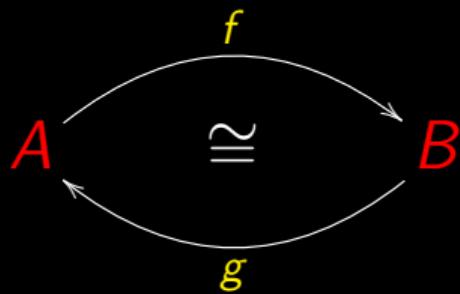
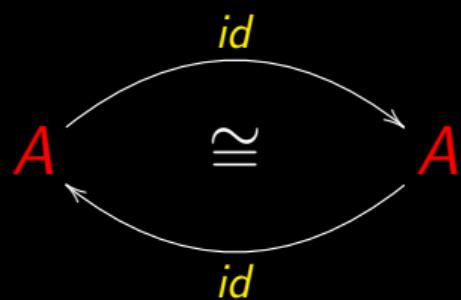
Recall

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



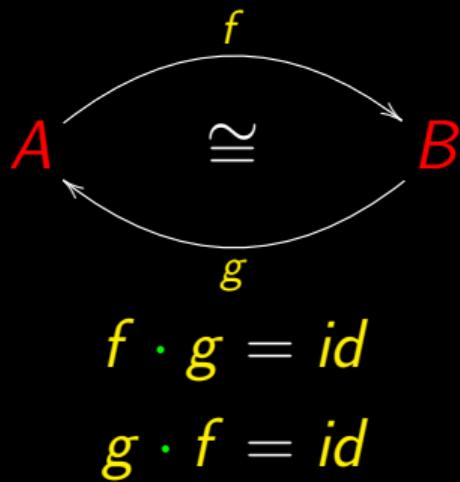
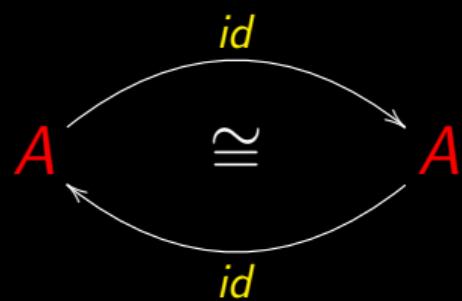
Recall

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



Recall

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



Recall

$$\left\{ \begin{array}{l} \textcolor{red}{swap} : A \times B \rightarrow B \times A \\ \textcolor{brown}{swap}(a, b) = (\textcolor{red}{b}, a) \end{array} \right.$$

Recall

$$\left\{ \begin{array}{l} \textit{swap} : A \times B \rightarrow B \times A \\ \textit{swap} (a, b) = (b, a) \end{array} \right.$$

$$\begin{array}{ccc} A \times B & \xrightarrow{\quad \cong \quad} & B \times A \\ \swarrow \textit{swap} & & \searrow \textit{swap} \end{array}$$

Recall

$$\left\{ \begin{array}{l} \text{swap} : A \times B \rightarrow B \times A \\ \text{swap}(a, b) = (b, a) \end{array} \right.$$

$$A \times B \xrightarrow{\quad \cong \quad} B \times A$$

swap swap

$$A \xrightarrow{\quad ? \quad} A \times B$$

π_1

Recall

$$\left\{ \begin{array}{l} \text{swap} : A \times B \rightarrow B \times A \\ \text{swap}(a, b) = (b, a) \end{array} \right.$$

$$A \times B \xrightarrow{\quad \cong \quad} B \times A$$

swap
swap

$$A \xrightarrow{\quad ? \quad} A \times B$$

? π_1

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

Recall

$$\left\{ \begin{array}{l} \text{swap} : A \times B \rightarrow B \times A \\ \text{swap}(a, b) = (b, a) \end{array} \right.$$

$$A \times B \xrightarrow{\quad \cong \quad} B \times A$$

swap
swap

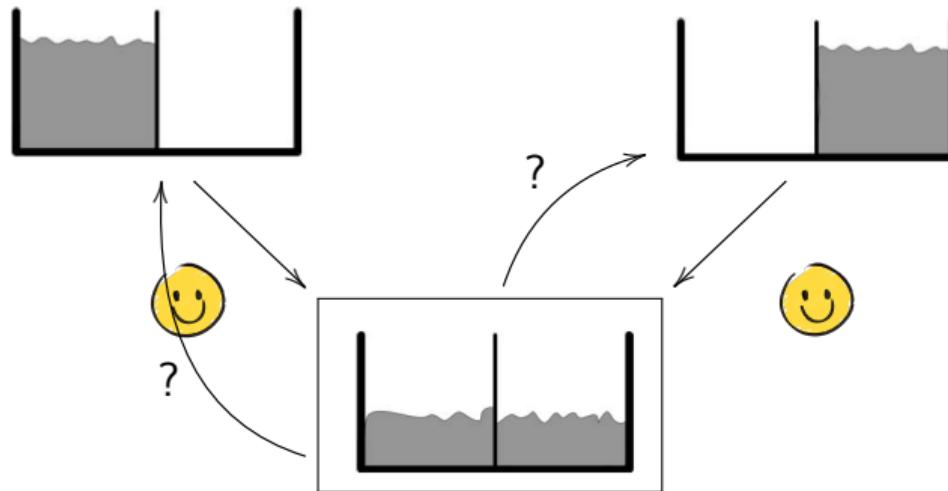
$$A \xrightarrow{\quad ? \quad} A \times B$$

?
 π_1

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

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

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 $\pi_1\dots$

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

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 $\pi_1\dots$

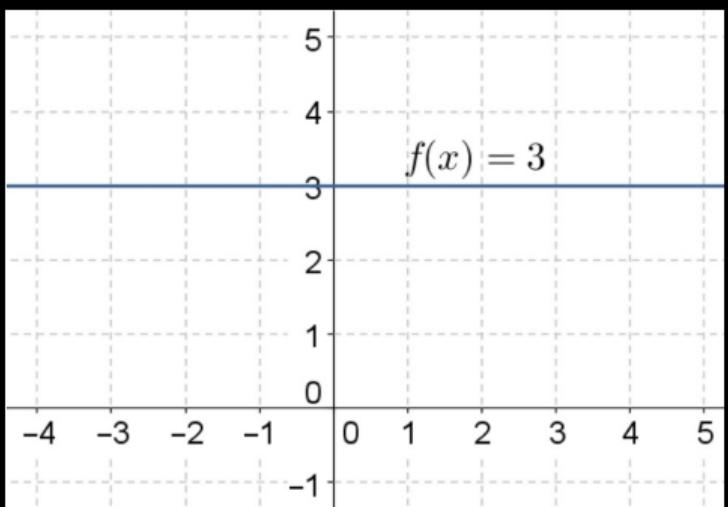
$$\begin{cases} \text{zero } x = 0 \\ \text{one } x = 1 \end{cases}$$

one $10 = 1$

one "string" = 1

zero "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

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

$$b_0 \in B$$

$$f(x) = 3$$

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

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

Constant functions

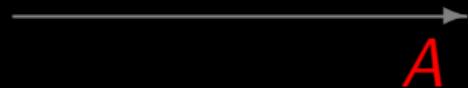
$$3 \in \mathbb{N}_0$$

$$b_0 \in B$$

$$f(x) = 3$$

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

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



Constant functions

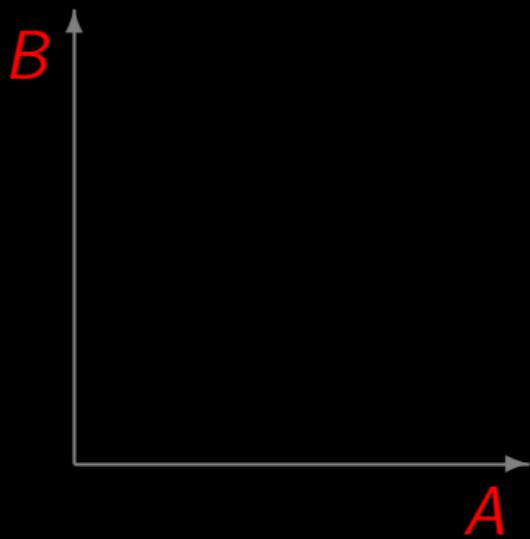
$$3 \in \mathbb{N}_0$$

$$b_0 \in B$$

$$f(x) = 3$$

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

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



Constant functions

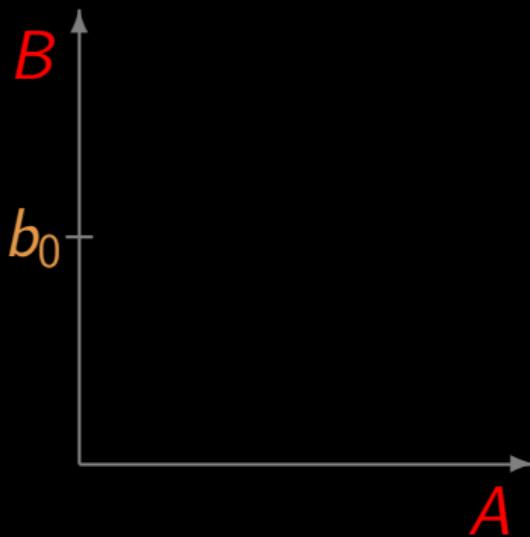
$$3 \in \mathbb{N}_0$$

$$b_0 \in B$$

$$f(x) = 3$$

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

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



Constant functions

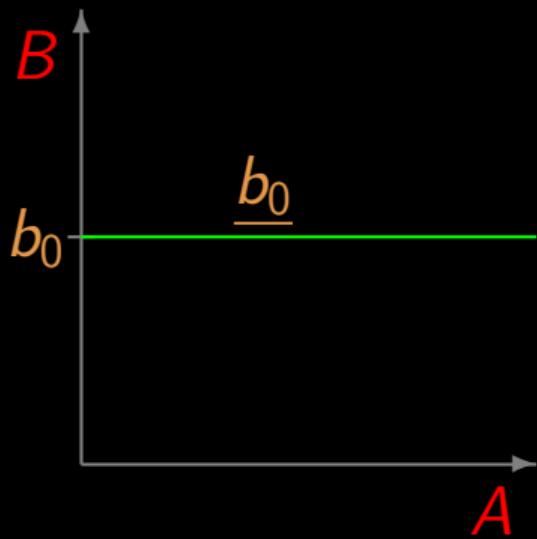
$$3 \in \mathbb{N}_0$$

$$b_0 \in B$$

$$f(x) = 3$$

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

$$\begin{cases} \frac{b_0}{b_0} : A \rightarrow B \\ \underline{b_0} a = b_0 \end{cases}$$



Constant functions

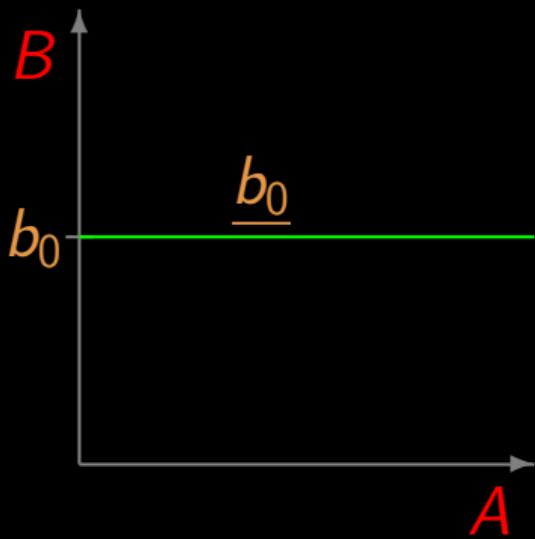
$$3 \in \mathbb{N}_0$$

$$b_0 \in B$$

$$f \ x = 3$$

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

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



(Haskell: `const b0`)

Properties

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

Properties

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

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

Properties

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

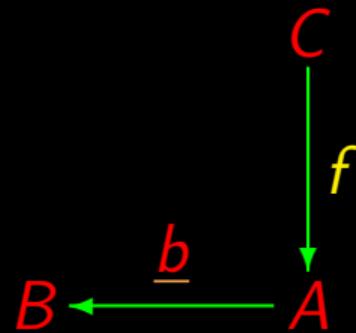
$$B \xleftarrow{\underline{b}} A$$

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

Properties

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

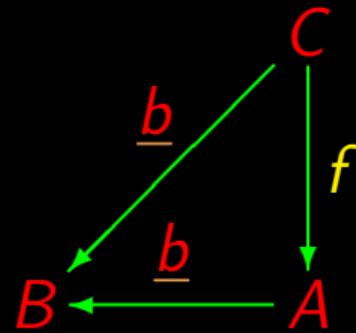
$$g \cdot \underline{b} = g \underline{b}$$



Properties

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

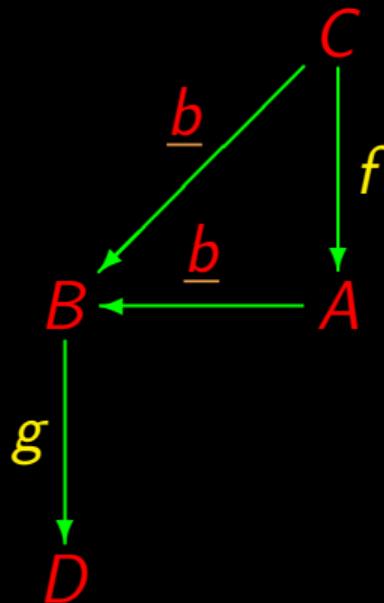
$$g \cdot \underline{b} = g \underline{b}$$



Properties

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

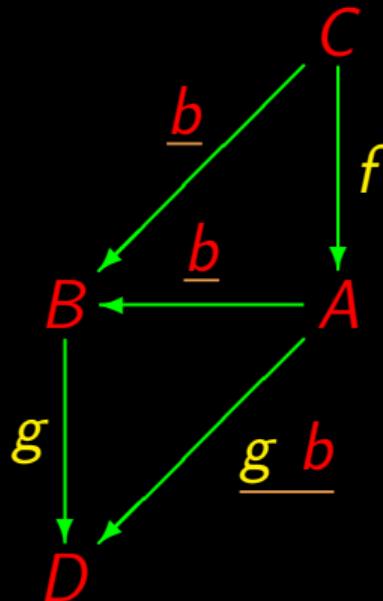
$$g \cdot \underline{b} = \underline{g} \underline{b}$$



Properties

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

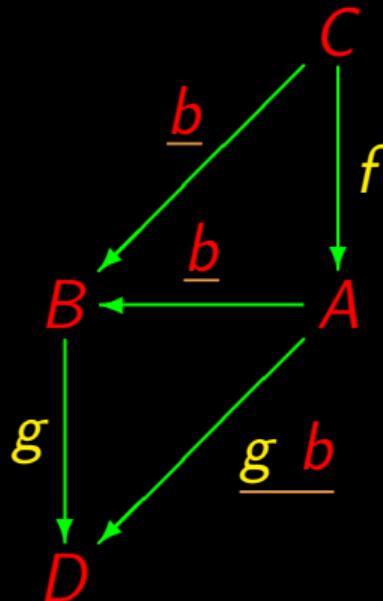
$$g \cdot \underline{b} = \underline{g \ b}$$



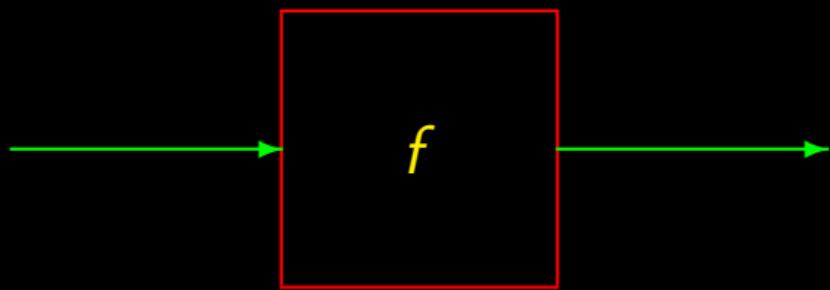
Properties

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

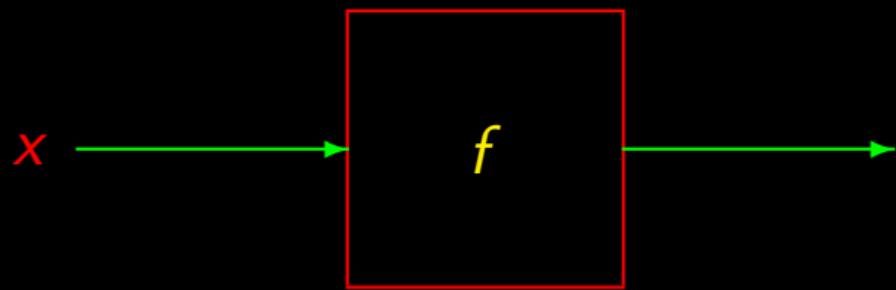
$$g \cdot \underline{b} = \underline{g \ b}$$



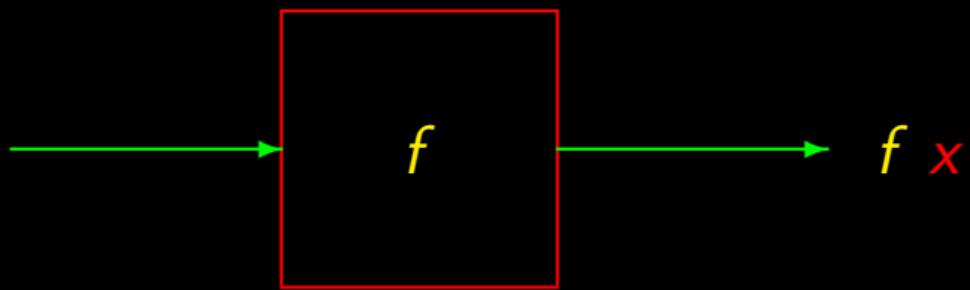
Data flow



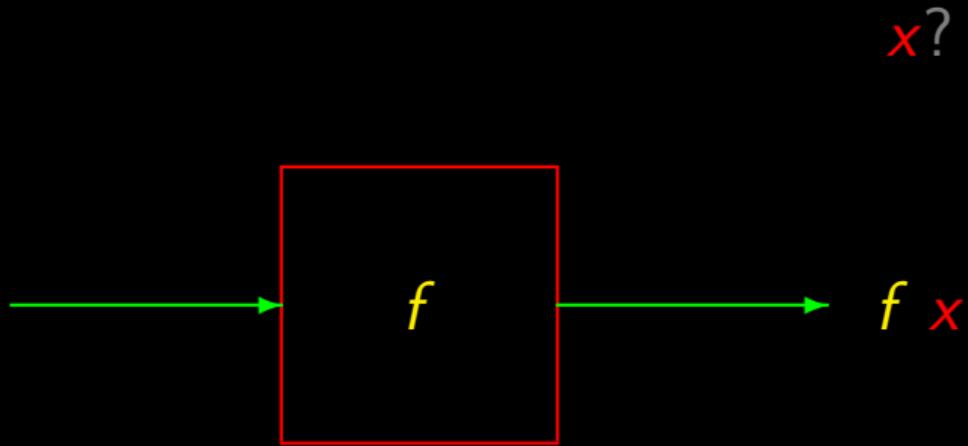
Data flow



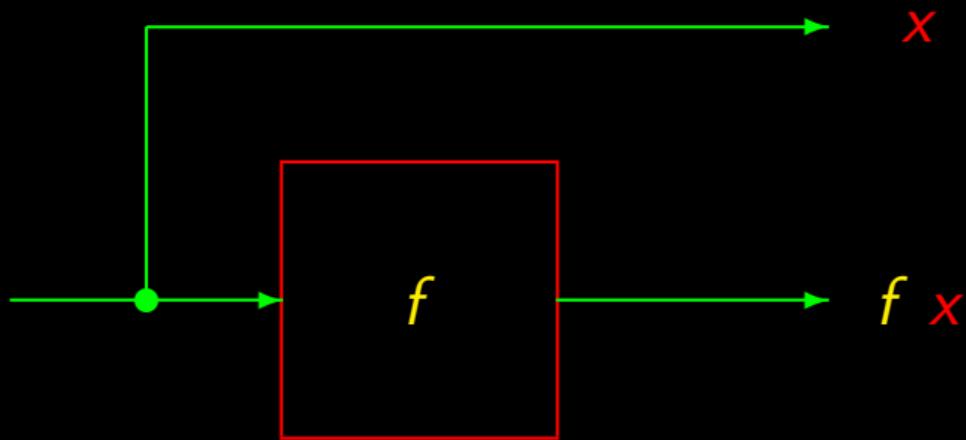
Data flow



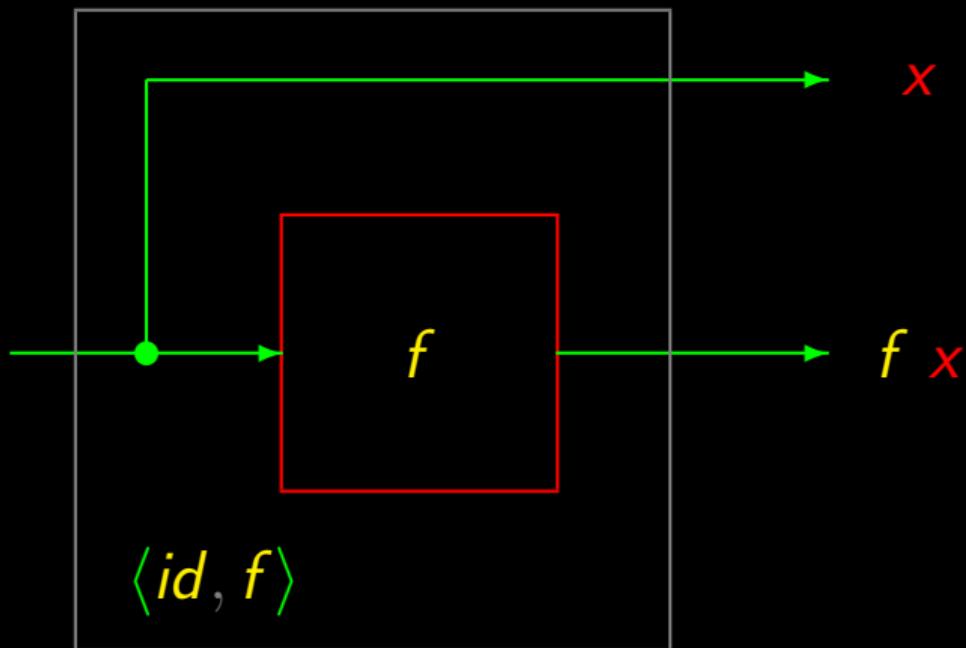
Data flow



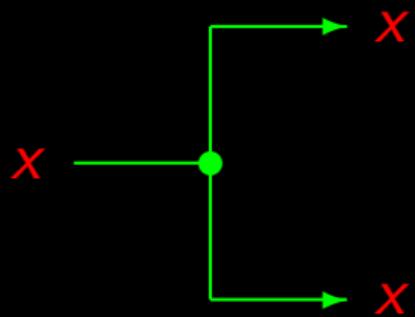
Data flow



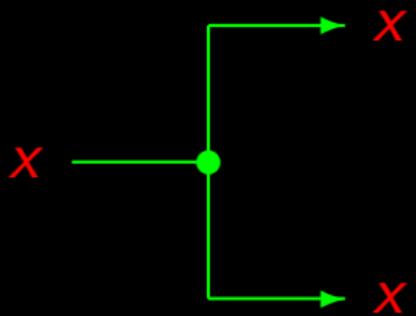
Data flow



Data duplication

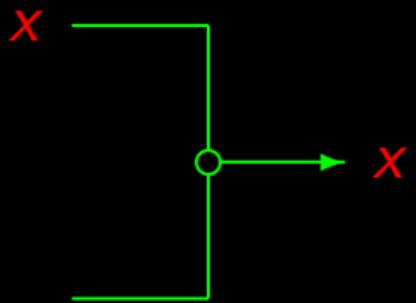


Data duplication



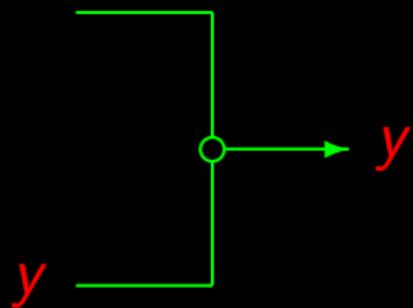
$$dup = \langle id, id \rangle$$

Data joins



join = [id, id]

Data joins



join = [id, id]

Cálculo de Programas

Class T04

if-then-else

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

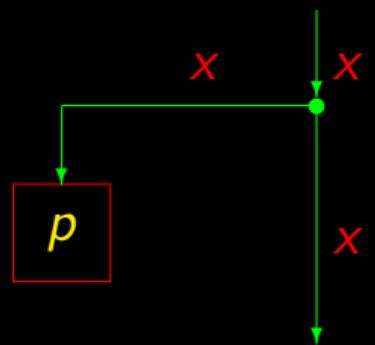
if-then-else

↓
 x

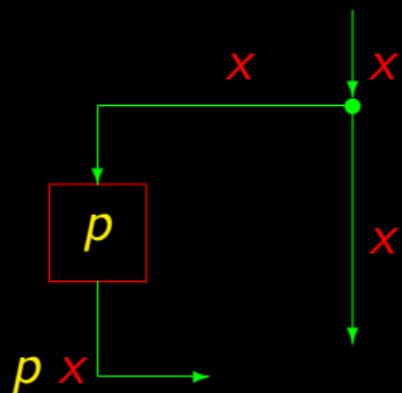
p

f g

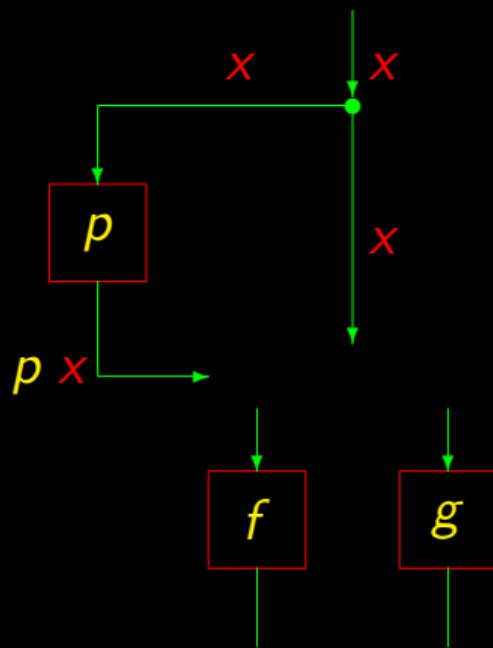
if-then-else



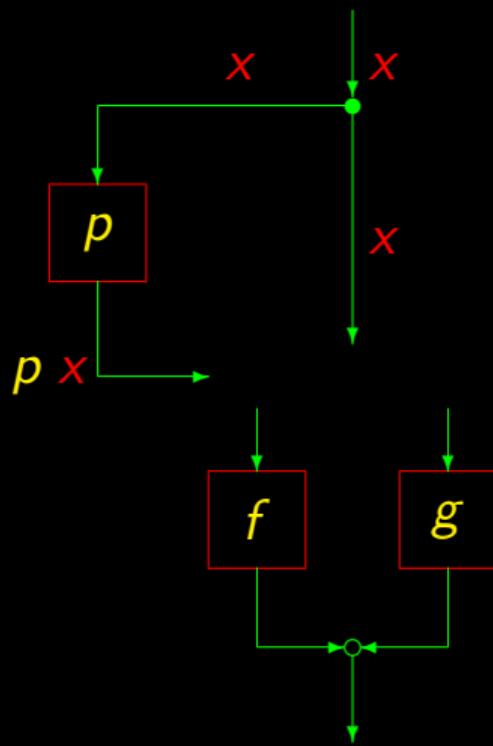
if-then-else



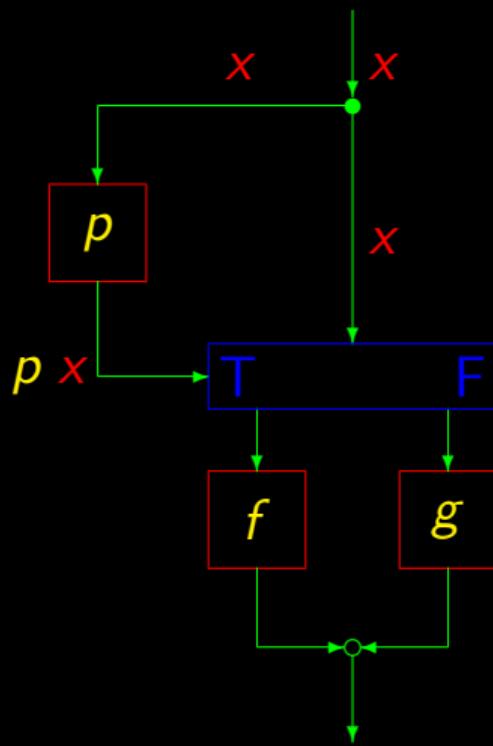
if-then-else



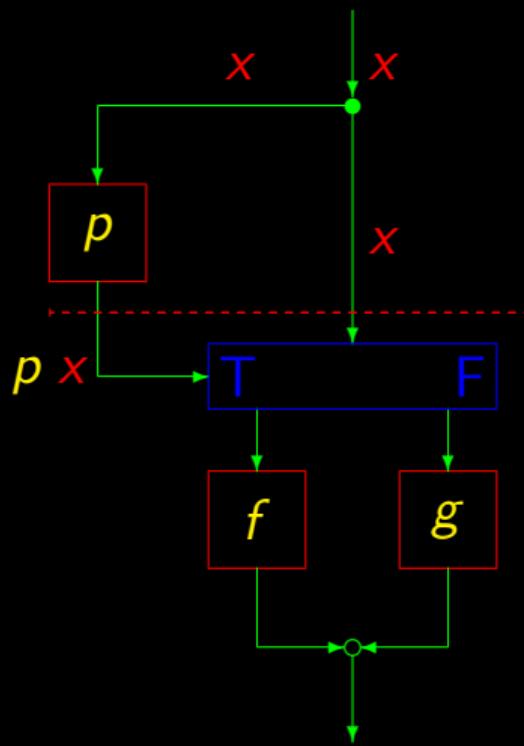
if-then-else



if-then-else



if-then-else



“Point-free”

if-then-else

$$x \in A$$

if-then-else

$$x \in A$$

$$(p\textcolor{red}{x}, x) \in \textcolor{red}{B} \times A$$

if-then-else

$$x \in A$$

$$(p\ x, x) \in \mathbf{B} \times A$$

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

if-then-else

$\textcolor{red}{B}$ in Haskell:

$x \in A$

data Bool = False | True

$(\textcolor{blue}{p} \textcolor{red}{x}, x) \in \textcolor{red}{B} \times A$

$\textcolor{red}{B} = \{ \textcolor{violet}{F}, \textcolor{violet}{T} \}$

if-then-else

$x \in A$

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

$B = \{F, T\}$

B in Haskell:

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

T represented by `True`

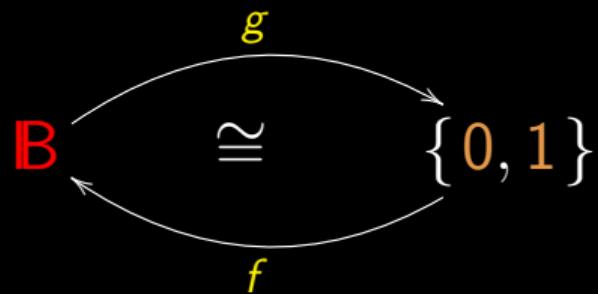
F represented by `False`

The type 2

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

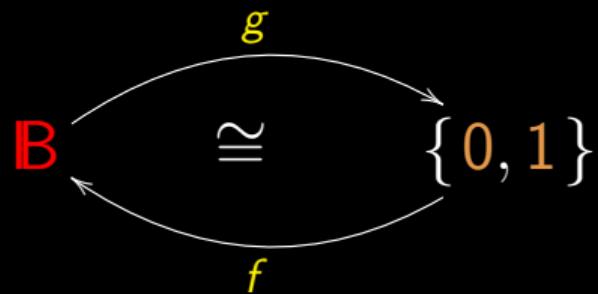
The type 2

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



The type 2

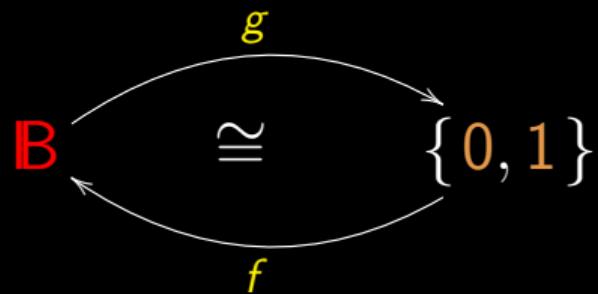
$$\mathbb{B} = \{F, T\} \cong \{\text{False}, \text{True}\} \cong \{0, 1\} \cong \dots \cong 2$$



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

The type 2

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



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

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

$$a + a = 2 \ a$$

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

$$\textcolor{red}{a} + \textcolor{red}{a} = 2 \textcolor{red}{a}$$

$$A + A \xrightarrow{\textcolor{blue}{join}} A$$

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

$$a + a = 2 \ a$$

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

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

Is it the case that...?

$$\begin{array}{ccc} 2 \times A & \xrightleftharpoons[\text{f}]{\text{g}} & A + A \end{array}$$



$$\begin{array}{ccc} 2 \times A & \cong & A + A \\ \textcolor{red}{2} \times \textcolor{green}{A} & \cong & \textcolor{red}{A} + \textcolor{green}{A} \end{array}$$

A commutative diagram showing the isomorphism between $2 \times A$ and $A + A$. The top row consists of red text. The bottom row consists of green text. The isomorphism is labeled with \cong in the center. Two curved arrows connect the terms: one from $2 \times A$ to $A + A$ labeled g , and another from $A + A$ to $2 \times A$ labeled f .

$$\begin{aligned} f &= [\langle \underline{T}, id \rangle, \langle \underline{F}, id \rangle] \\ g &= \dots (?) \end{aligned}$$

Notation

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

The diagram illustrates the isomorphism between the Cartesian product of two sets \$A\$ and the disjoint union of two sets \$A\$. It features two curved arrows connecting the terms. The top arrow is labeled \$\alpha^\circ\$ and the bottom arrow is labeled \$\alpha\$.

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

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

Notation

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

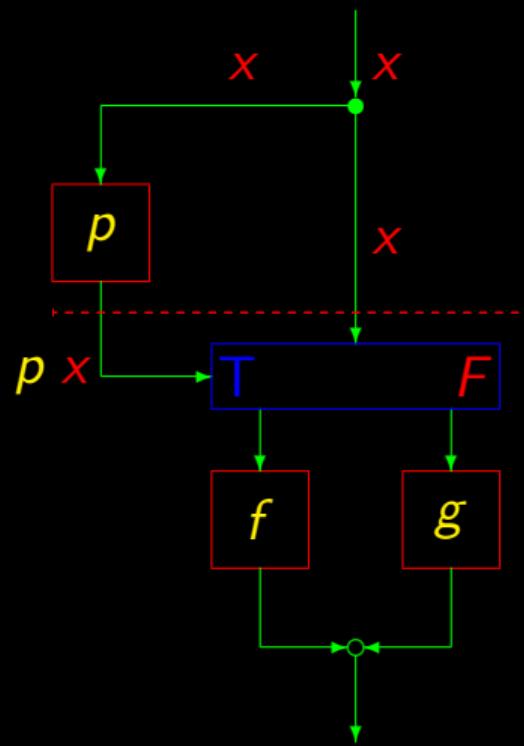
The diagram illustrates a commutative relationship between two sets. On the left is the set $2 \times A$, where the first 2 is green and the A is red. On the right is the set $A + A$, where the A 's are red. Two curved arrows connect these sets. The top arrow points from $2 \times A$ to $A + A$ and is labeled α° above it. The bottom arrow points from $A + A$ back to $2 \times A$ and is labeled α below it.

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

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

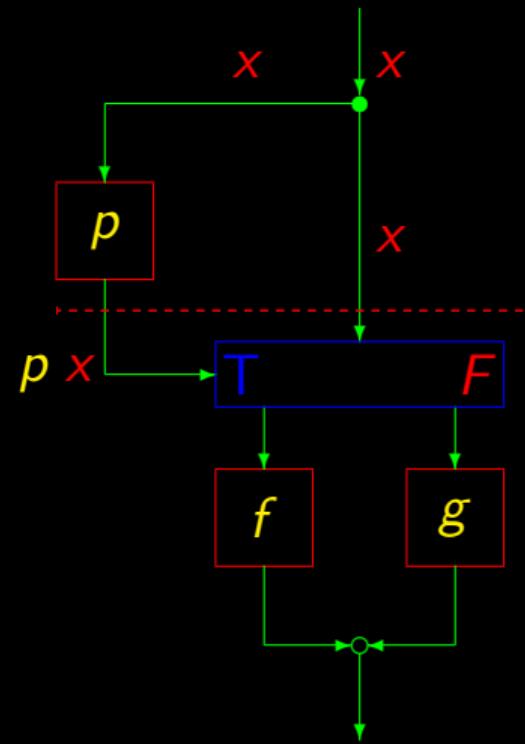
α determines α°

if-then-else



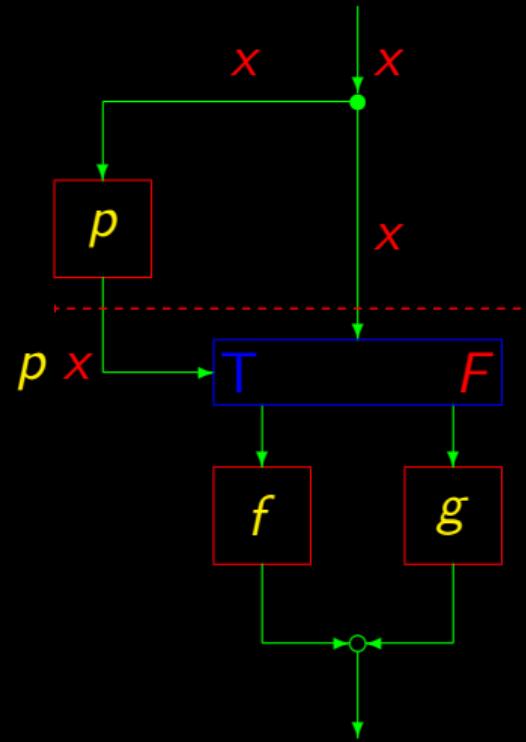
if-then-else

$$\begin{array}{c} A \\ \downarrow \\ \langle p, id \rangle \\ 2 \times A \end{array}$$



if-then-else

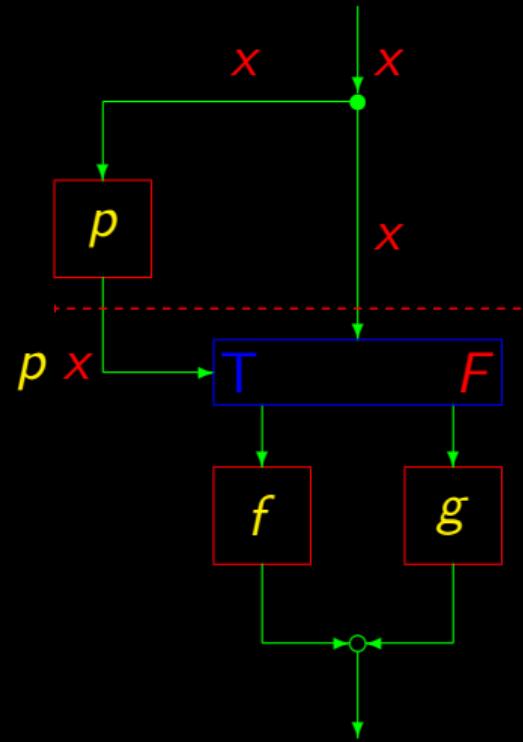
$$\begin{array}{c} A \\ \downarrow \langle p, id \rangle \\ 2 \times A \\ \\ A \\ f \downarrow \\ B \\ \\ A \\ \downarrow g \\ B \end{array}$$



if-then-else

$$\begin{array}{c} A \\ \downarrow \\ \langle p, id \rangle \\ 2 \times A \end{array}$$

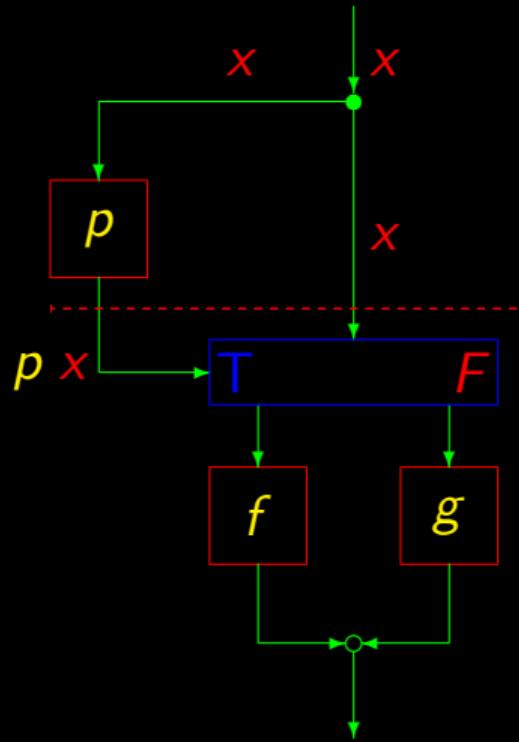
$$\begin{array}{ccccc} A & \xrightarrow{i_1} & A + A & \xleftarrow{i_2} & A \\ f \downarrow & & f + g & & \downarrow g \\ B & \xrightarrow{i_1} & B + B & \xleftarrow{i_2} & B \end{array}$$



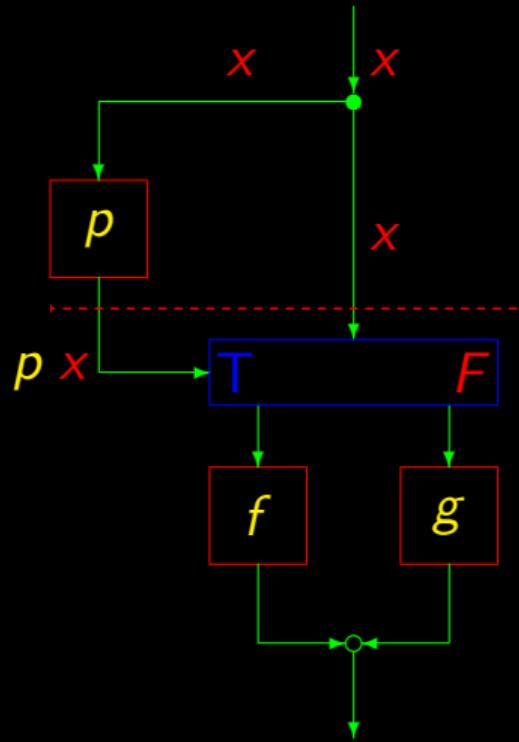
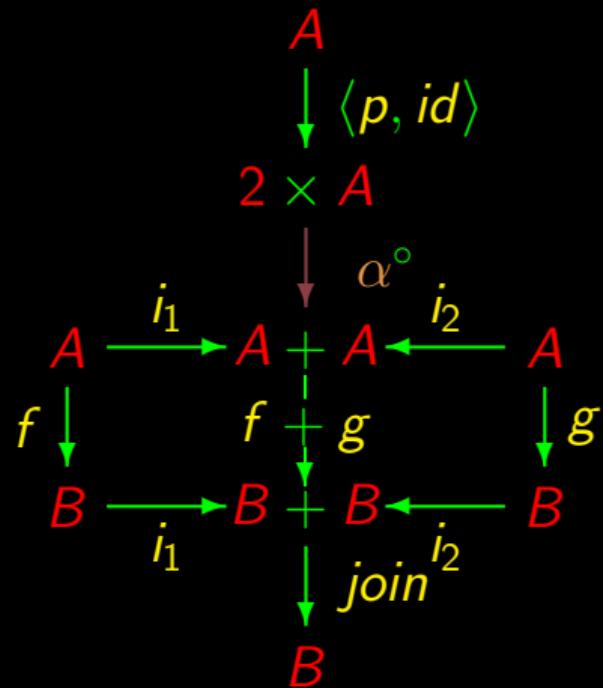
if-then-else

$$\begin{array}{c} A \\ \downarrow \langle p, id \rangle \\ 2 \times A \end{array}$$

$$\begin{array}{ccccc} A & \xrightarrow{i_1} & A + A & \xleftarrow{i_2} & A \\ f \downarrow & & f + g & & \downarrow g \\ B & \xrightarrow{i_1} & B + B & \xleftarrow{i_2} & B \\ & & \downarrow join & & \end{array}$$



if-then-else



McCarthy Conditional $p \rightarrow f , g$

From the diagram:

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

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:

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

$p?$

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

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

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

wherfrom

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

Pointwise

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

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

wherfrom

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

Case analysis?

Pointwise

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

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

wherfrom

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

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”

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

“Calculemus”

$$\begin{aligned} h \cdot (p \rightarrow f , g) \\ \Leftrightarrow & \quad \left\{ \begin{array}{l} \text{definition of conditional} \end{array} \right\} \\ h \cdot [f, g] \cdot p? \end{aligned}$$

“Calculemus”

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

$$\Leftrightarrow \quad \left\{ \begin{array}{l} \text{definition of conditional} \end{array} \right\}$$

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

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

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

“Calculemus”

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

$$\Leftrightarrow \quad \left\{ \begin{array}{l} \text{definition of conditional} \end{array} \right\}$$

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

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

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

$$\Leftrightarrow \quad \left\{ \begin{array}{l} \text{definition of conditional} \end{array} \right\}$$

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

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



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

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



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

Isomorphisms

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



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

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

Isomorphisms

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



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

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

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

Isomorphisms

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



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

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

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

$$A + 0 \cong A ?$$

Isomorphisms

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



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

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

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

$$A + 0 \cong A ?$$

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

Distributivity

$$A \times (B + C) \underset{\approx}{\sim} A \times B + A \times C$$

distr *undistr*

Distributivity

$$A \times (B + C) \underset{\approx}{\equiv} A \times B + A \times C$$

distr

undistr

$$(B + C) \times A \underset{\approx}{\equiv} B \times A + C \times A$$

distl

undistl

Exchange law



Exchange law

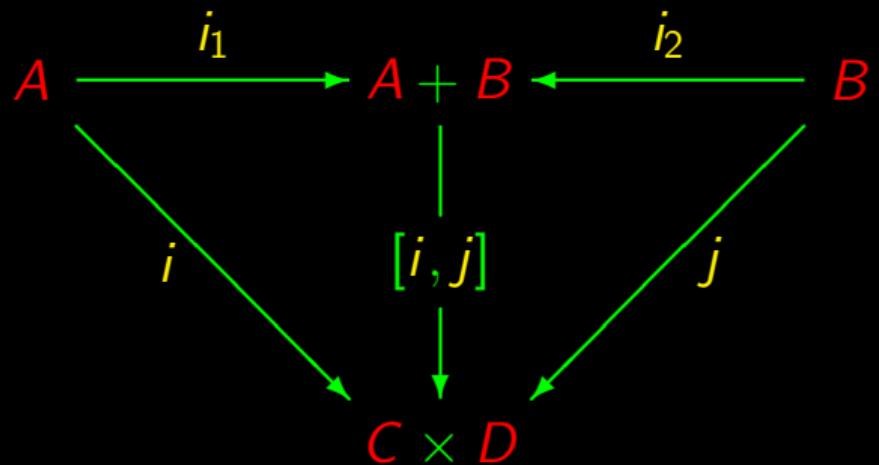
$$A + B$$

$$\downarrow \alpha$$

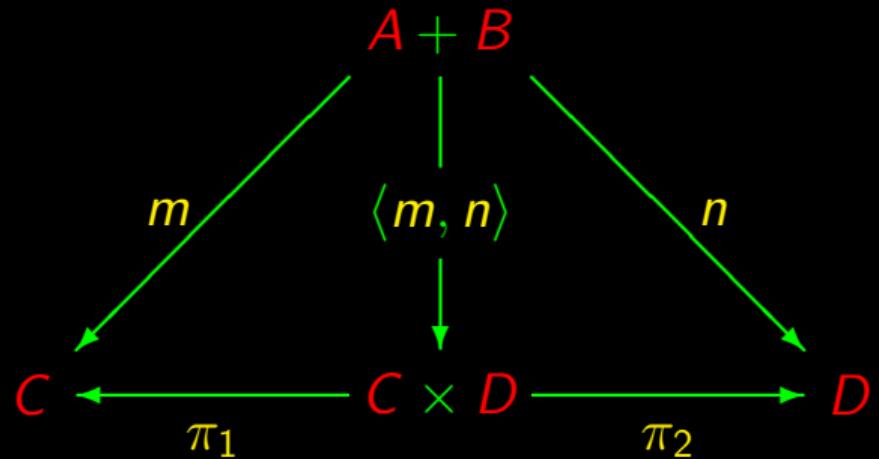
Exchange law

$$\begin{array}{ccc} A + B & & \\ \downarrow \alpha & & \\ C \times D & & \end{array}$$

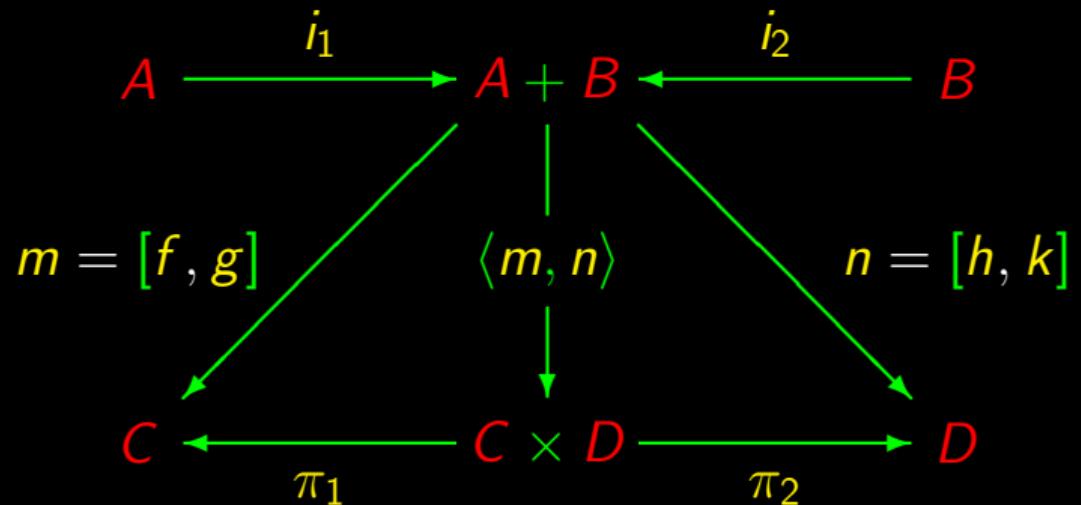
Exchange law



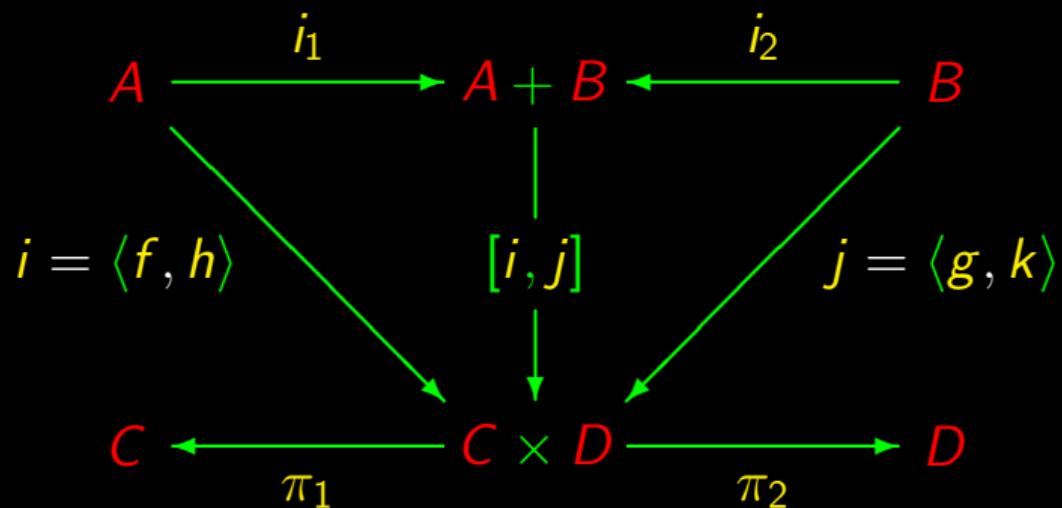
Exchange law



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

Distributivity

$$A \times (B + C) \underset{\approx}{\sim} A \times B + A \times C$$

distr

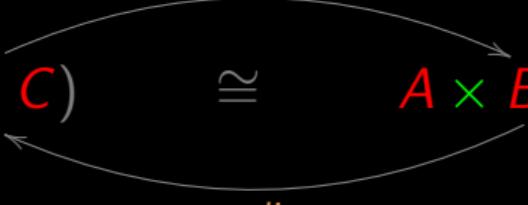
undistr

The diagram illustrates the distributive law between product and sum. It features two expressions: $A \times (B + C)$ on the left and $A \times B + A \times C$ on the right, separated by a symbol \approx . Two curved arrows connect the expressions: one arrow points from the left expression to the right, labeled *distr* above; another arrow points from the right expression back to the left, labeled *undistr* below.

Distributivity

$$A \times (B + C) \underset{\text{undistr}}{\approx} A \times B + A \times C$$

distr

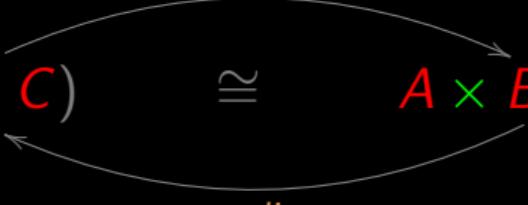


$$\text{undistr} = [\langle f, g \rangle, \langle h, k \rangle]$$

Distributivity

$$A \times (B + C) \underset{\text{undistr}}{\approx} A \times B + A \times C$$

distr



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

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

Distributivity

$$A \times (B + C) \underset{\text{undistr}}{\approx} A \times B + A \times C$$

distr



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

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

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

Distributivity

$$A \times (B + C) \underset{\approx}{\sim} A \times B + A \times C$$

distr *undistr*

The diagram illustrates the distributive law between product and sum. It shows two expressions connected by an equivalence symbol (\approx). Above the equivalence symbol is the word "distr" and above the left expression is "undistr".

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

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

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

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

Distributivity

$$A \times (B + C) \underset{\approx}{\sim} A \times B + A \times C$$

distr *undistr*



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

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

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

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

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

Distributivity

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

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

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

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

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

Distributivity

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

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

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

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

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

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

Distributivity

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

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

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

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

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

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

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

Distributivity

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

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

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

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

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

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

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

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

Distributivity

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

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

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

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

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

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

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

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

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

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

Distributivity

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

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

Distributivity

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

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

By definition of $f \times g$

$$undistr = [id \times i_1, id \times i_2]$$

The type 1

In Haskell:

$$() :: ()$$

Then

$$1 = \{()\}$$

Isomorphism

$$\begin{array}{ccc} A \times 1 & \xrightarrow{\quad \cong \quad} & A \\ \swarrow \pi_1 & & \searrow \langle id, ! \rangle \end{array}$$

The type 1

In Haskell:

`() :: ()`

Then

`1 = { () }`

Isomorphism

$$\begin{array}{ccc} A \times 1 & \xrightarrow{\quad \cong \quad} & A \\ \swarrow \pi_1 & & \searrow \langle id, ! \rangle \end{array}$$

... "!" ?

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":

$$1 \xrightarrow{\underline{a}} A$$

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

(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

$$0 = \{ \}$$

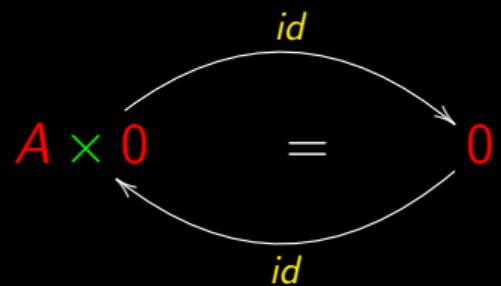
Immediate in e.g. Haskell:

```
data Zero
```

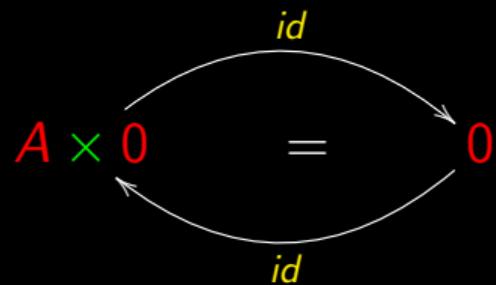
0 is not inhabited: $\neg (a \in 0)$
for all a

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

The type 0



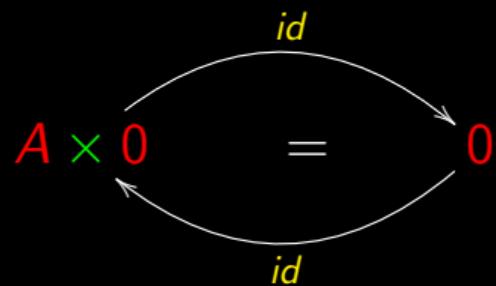
The type 0



In fact

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

The type 0

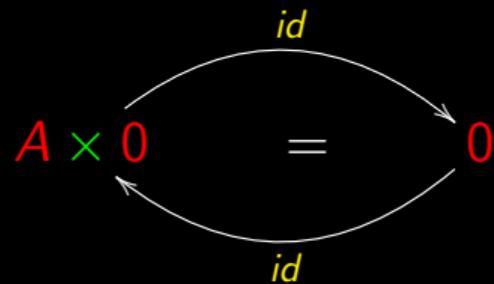


In fact

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

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

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 $1 + A$

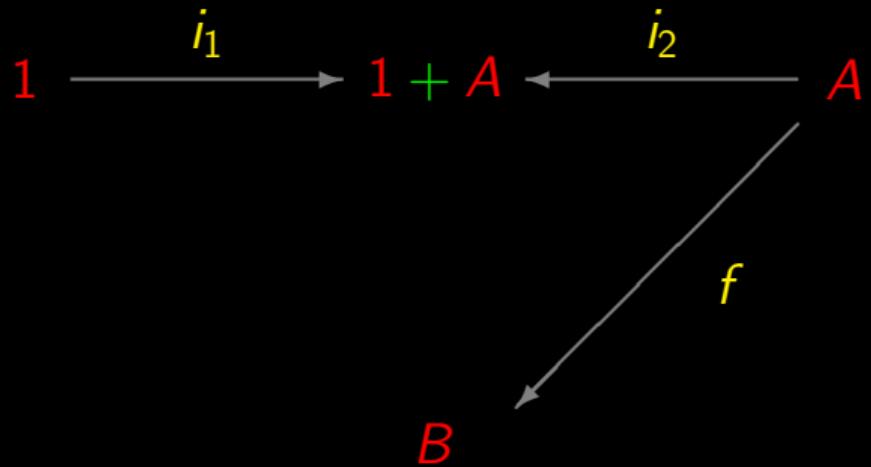
$$1 \xrightarrow{i_1} 1 + A \xleftarrow{i_2} A$$

The type $1 + A$

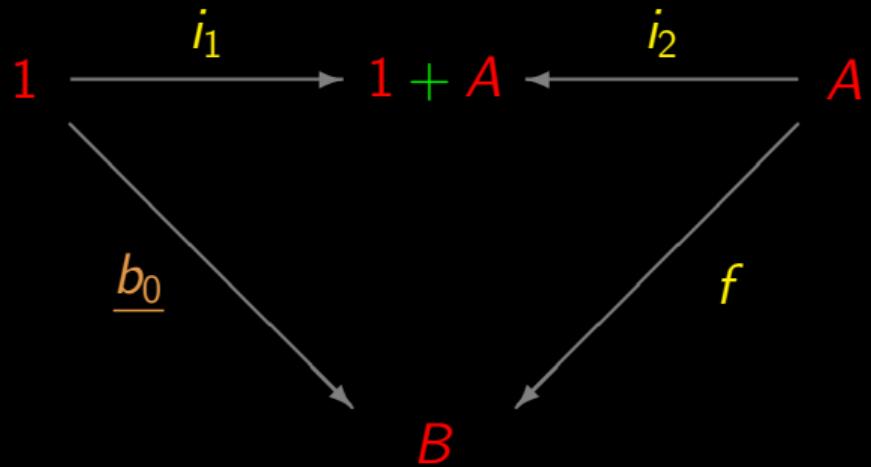
$$1 \xrightarrow{i_1} 1 + A \xleftarrow{i_2} A$$

B

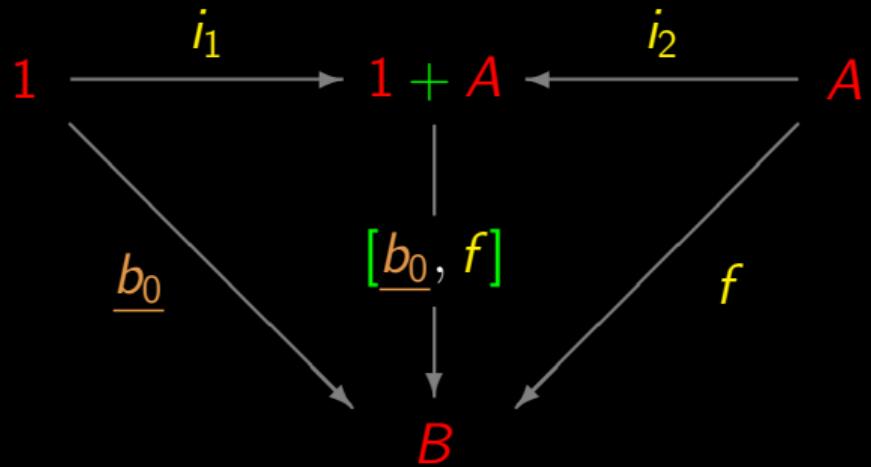
The type $1 + A$



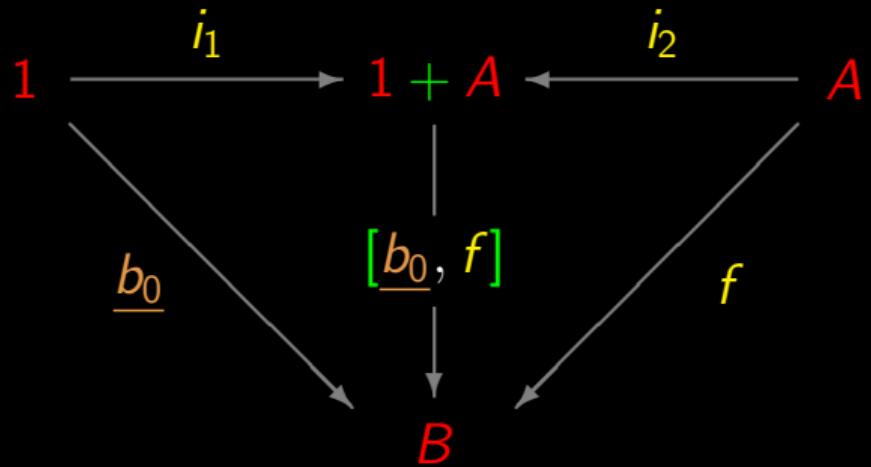
The type $1 + A$



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).

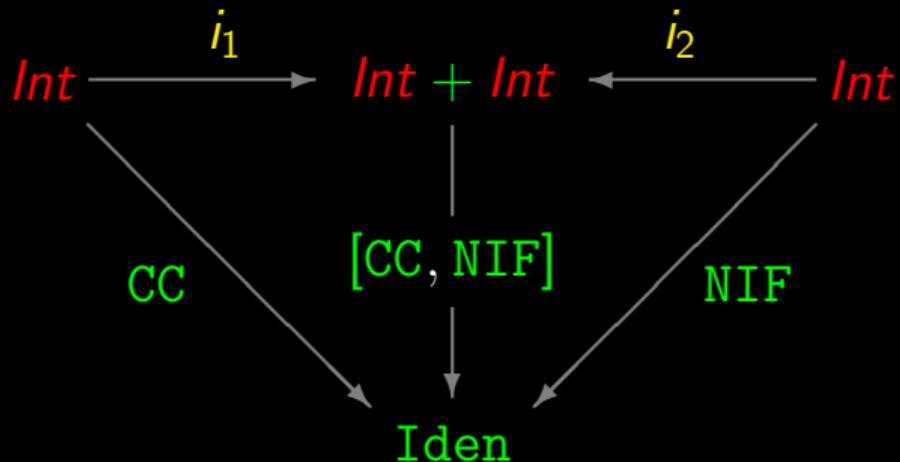
In Haskell (Portugal):

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

In Haskell (Germany):

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

More isomorphisms!



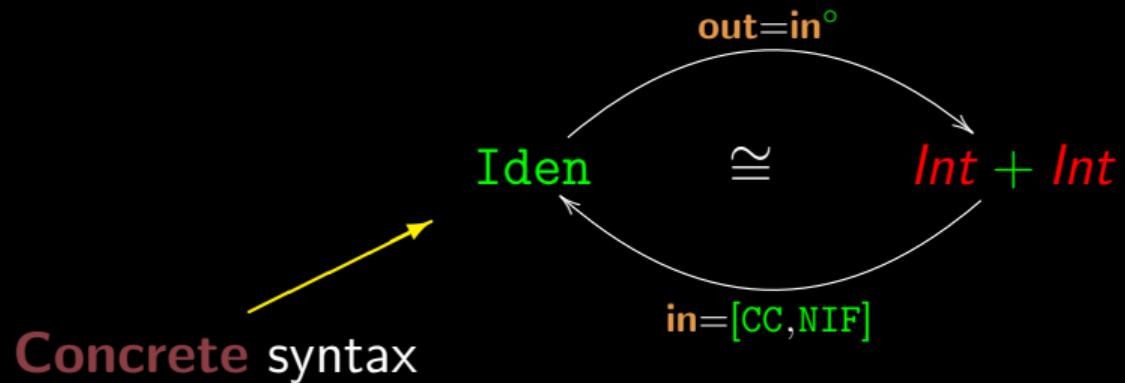
```
data Iden = CC Int | NIF Int
```

More isomorphisms!

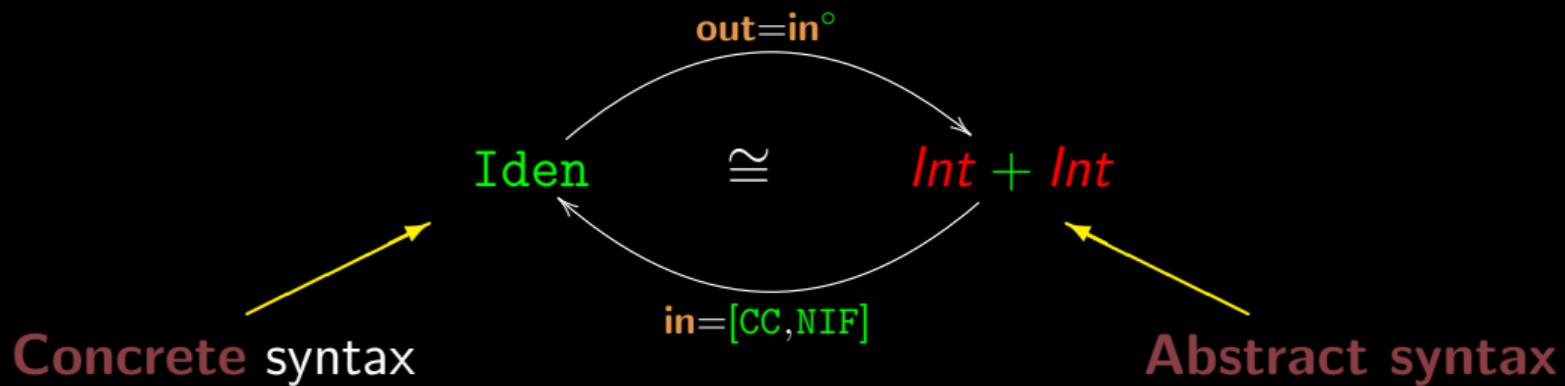
$$\text{Iden} \underset{\cong}{\sim} \textcolor{red}{Int} + \textcolor{green}{Int}$$

$\text{out} = \text{in}^\circ$
 $\text{in} = [\text{CC}, \text{NIF}]$

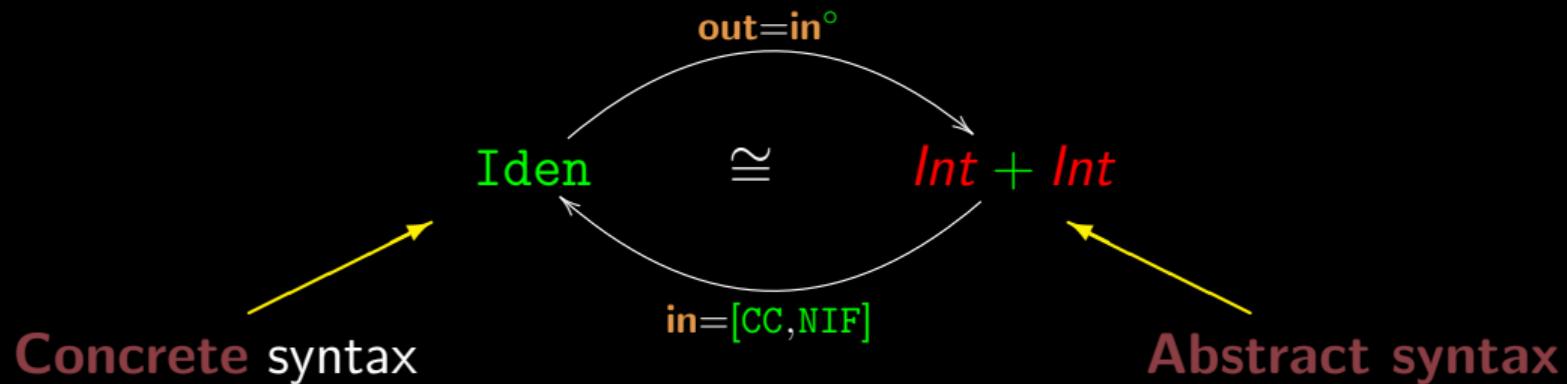
More isomorphisms!



More isomorphisms!

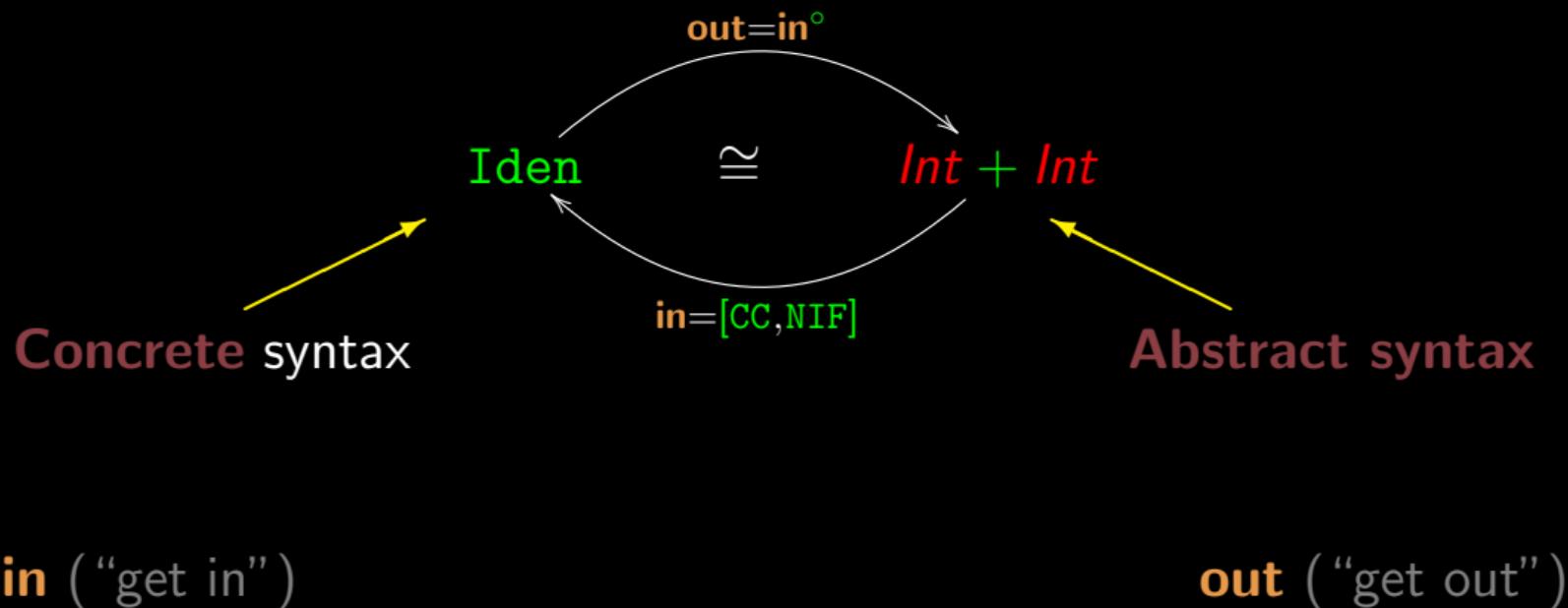


More isomorphisms!

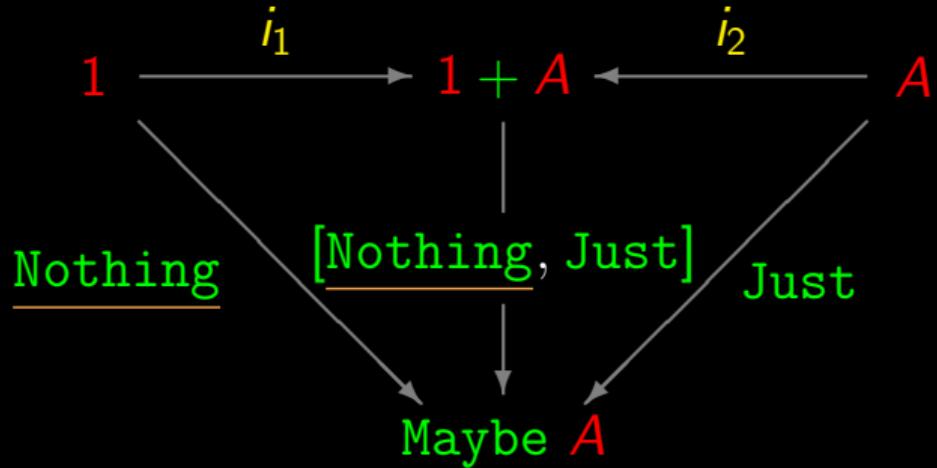


in (“get in”)

More isomorphisms!



Maybe



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

Maybe

$$\text{Maybe } A \cong 1 + A$$

$\text{out} = \text{in}^\circ$

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

Maybe

$$\text{Maybe } A \cong 1 + A$$

$\text{out} = \text{in}^\circ$

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

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

Maybe

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

Maybe

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

$$\Leftrightarrow \quad \left\{ \begin{array}{l} \mathbf{in} = [\underline{\text{Nothing}}, \text{Just}] \end{array} \right\}$$

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

Maybe

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

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

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

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

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

Maybe

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

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

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

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

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

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

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

Maybe

$$\Leftrightarrow \quad \left\{ \begin{array}{l} \text{variables ; } \underline{\text{Nothing}} : \textcolor{red}{1} \rightarrow \text{Maybe } A \\ \left\{ \begin{array}{l} \textcolor{blue}{\text{out Nothing}} () = \textcolor{blue}{i_1} () \\ \textcolor{blue}{(\text{out} \cdot \text{Just})} a = \textcolor{blue}{i_2} a \end{array} \right. \end{array} \right\}$$

Maybe

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

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

$$\Leftrightarrow \quad \{ \text{ composition (pointwise) } \}$$

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

Maybe

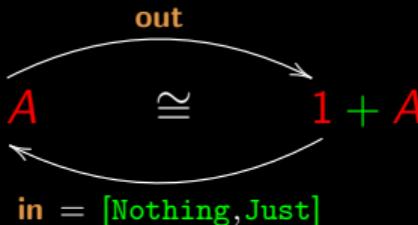
$$\Leftrightarrow \{ \text{ variables ; } \underline{\text{Nothing}} : 1 \rightarrow \text{Maybe } A \ }$$

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

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

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

$$\text{Maybe } A \cong 1 + A$$



Either

$$\text{Either } A \ B \quad \cong \quad A + B$$

out **in** = [Left,Right]

Either

$$\text{Either } A \ B \quad \cong \quad A + B$$

in = [Left,Right]

out

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

Left [Left, Right] Right

Either A B

[f, g]

either $f \ g$

Cálculo de Programas

Class T05

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]

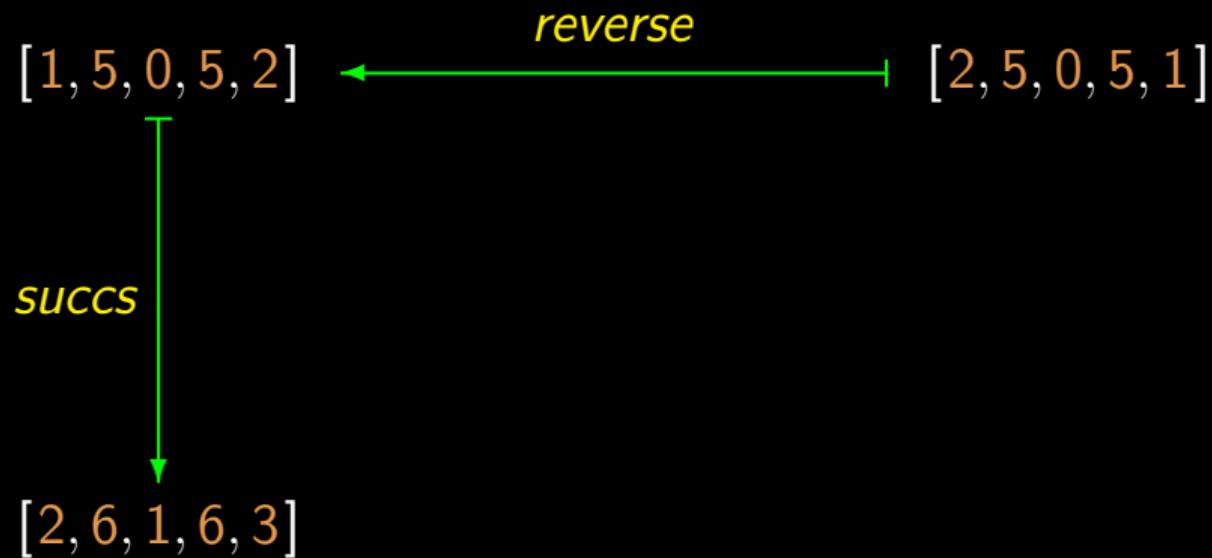
Natural properties

[2, 5, 0, 5, 1]

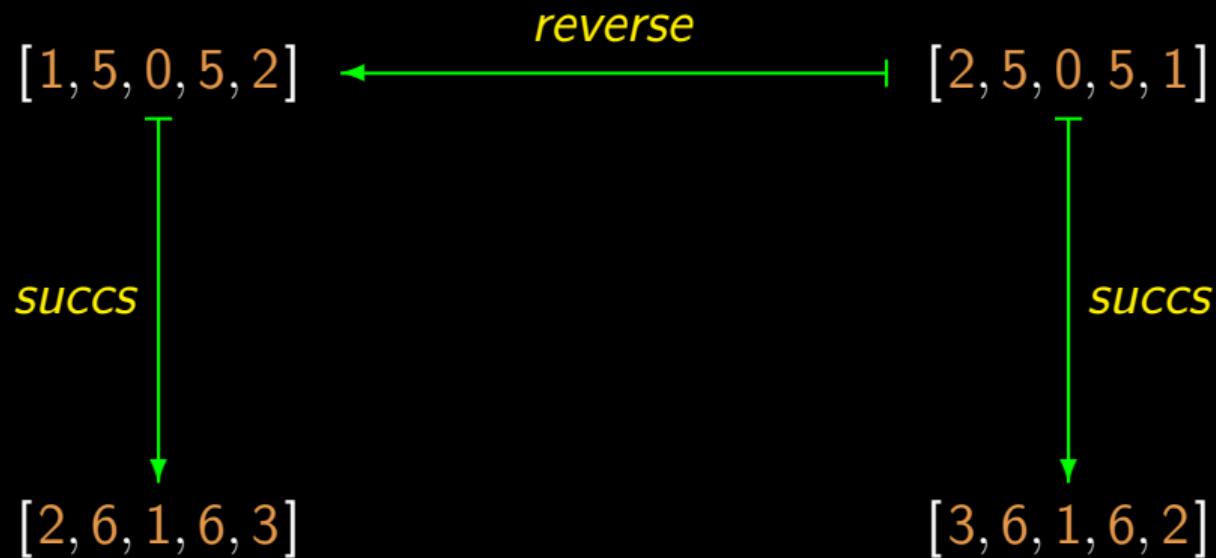
Natural properties



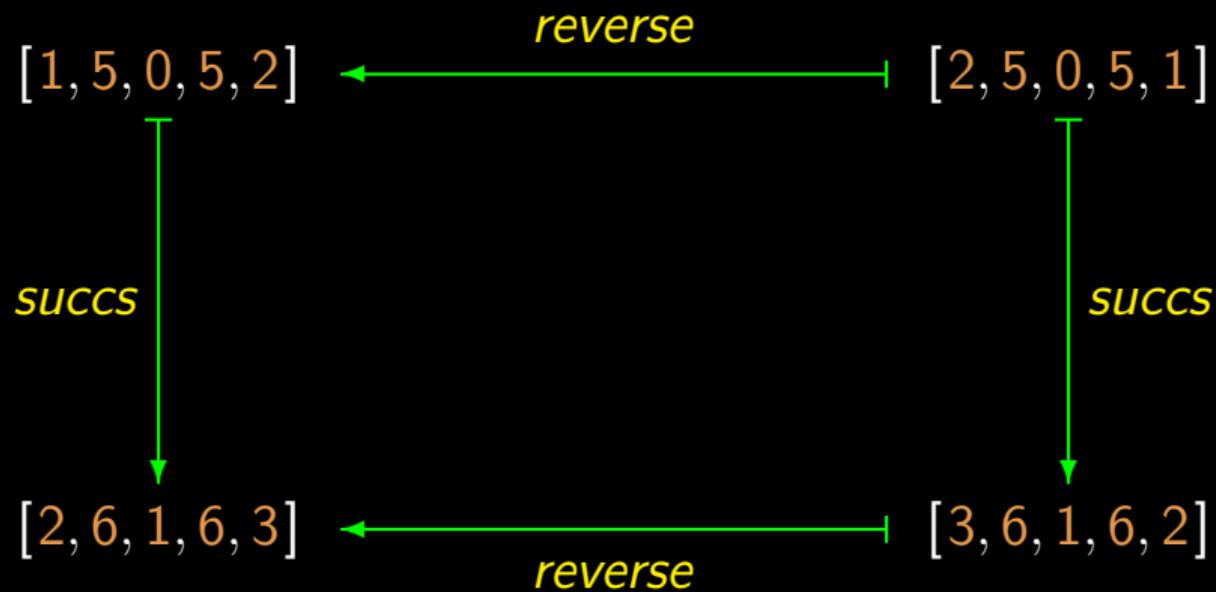
Natural properties



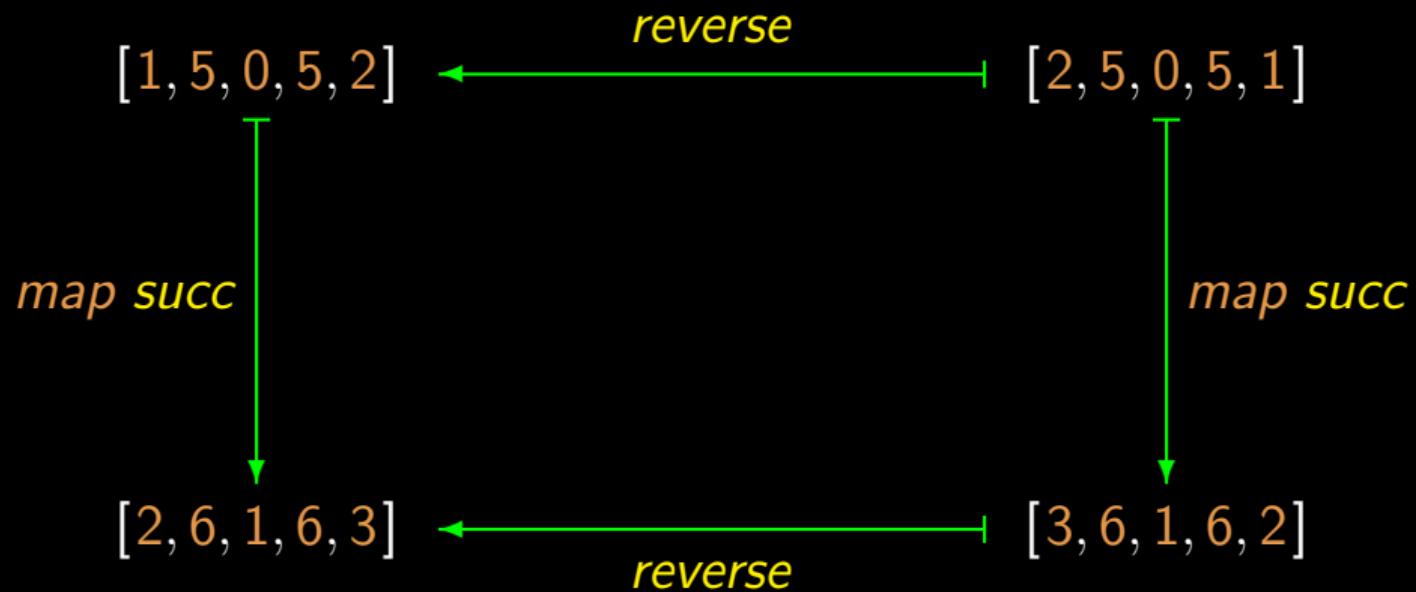
Natural properties



Natural properties



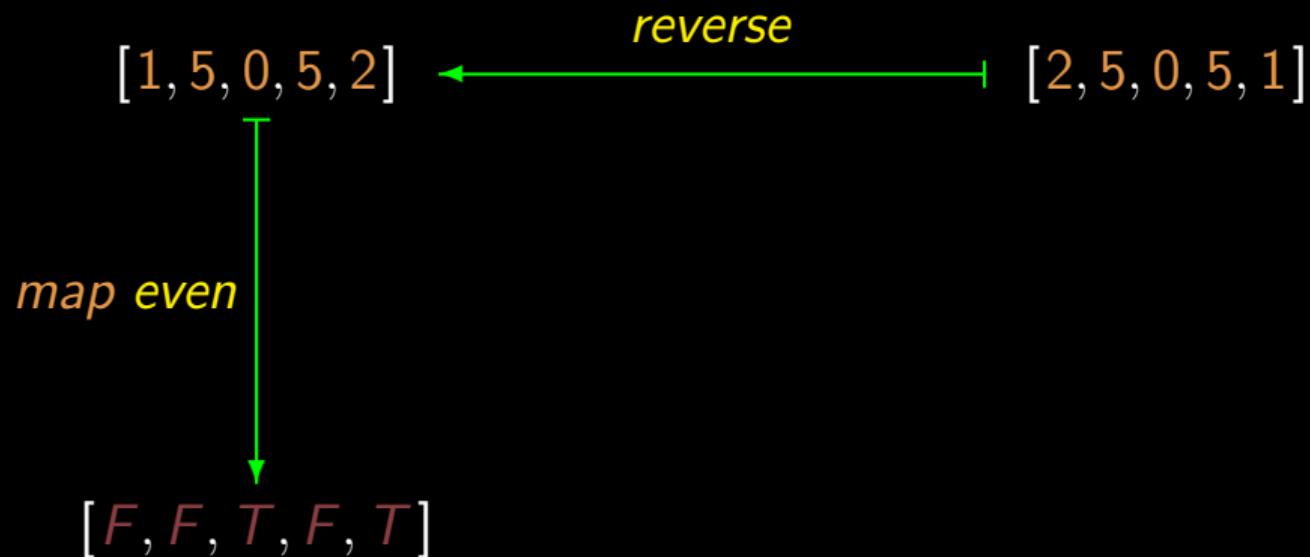
Natural properties



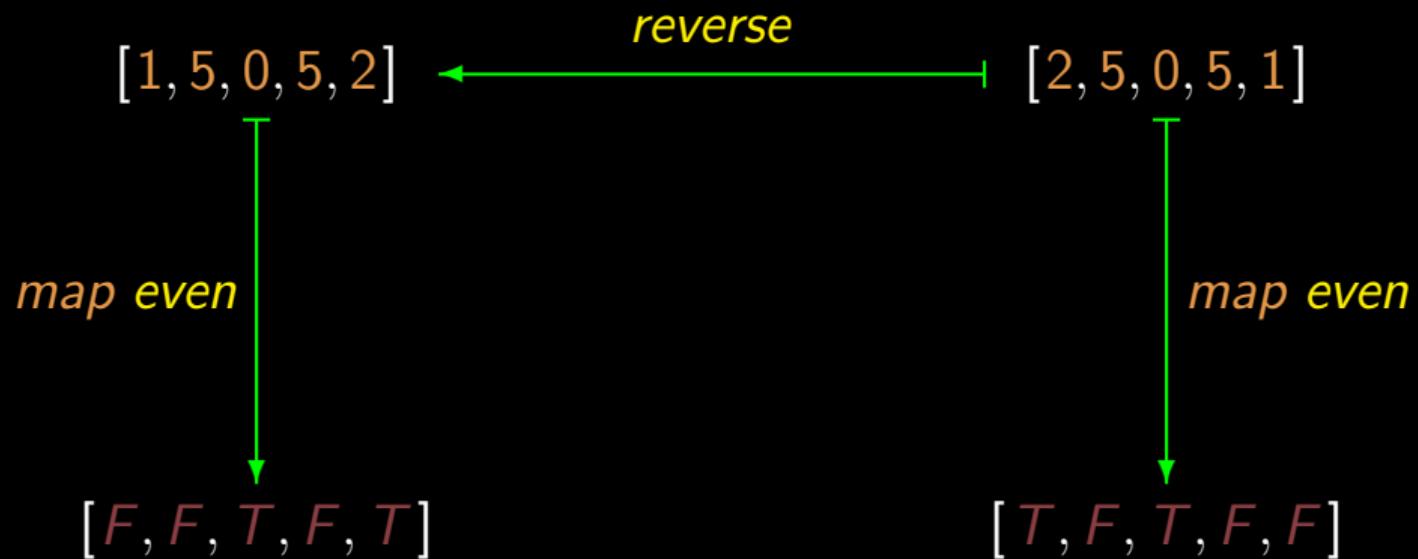
Natural properties



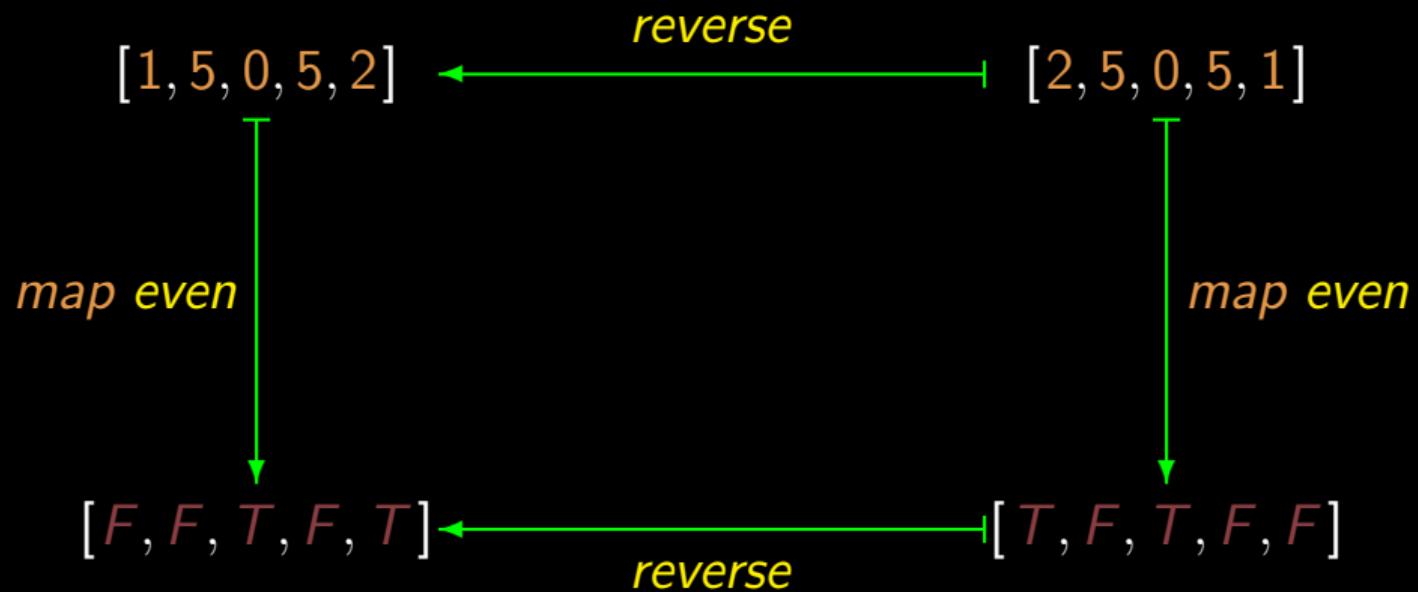
Natural properties



Natural properties



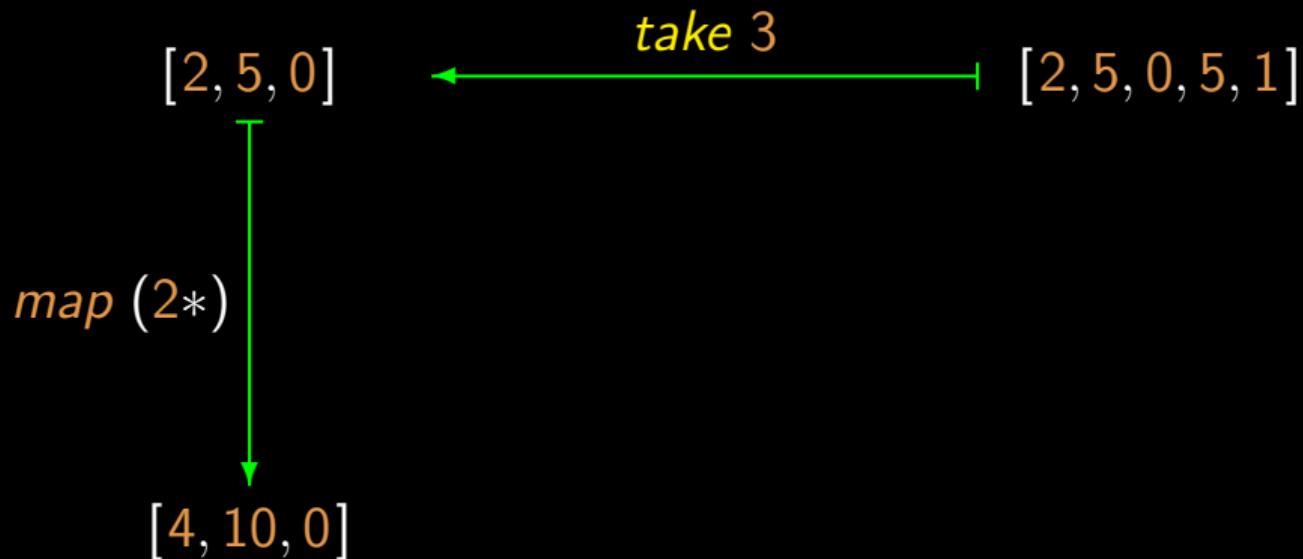
Natural properties



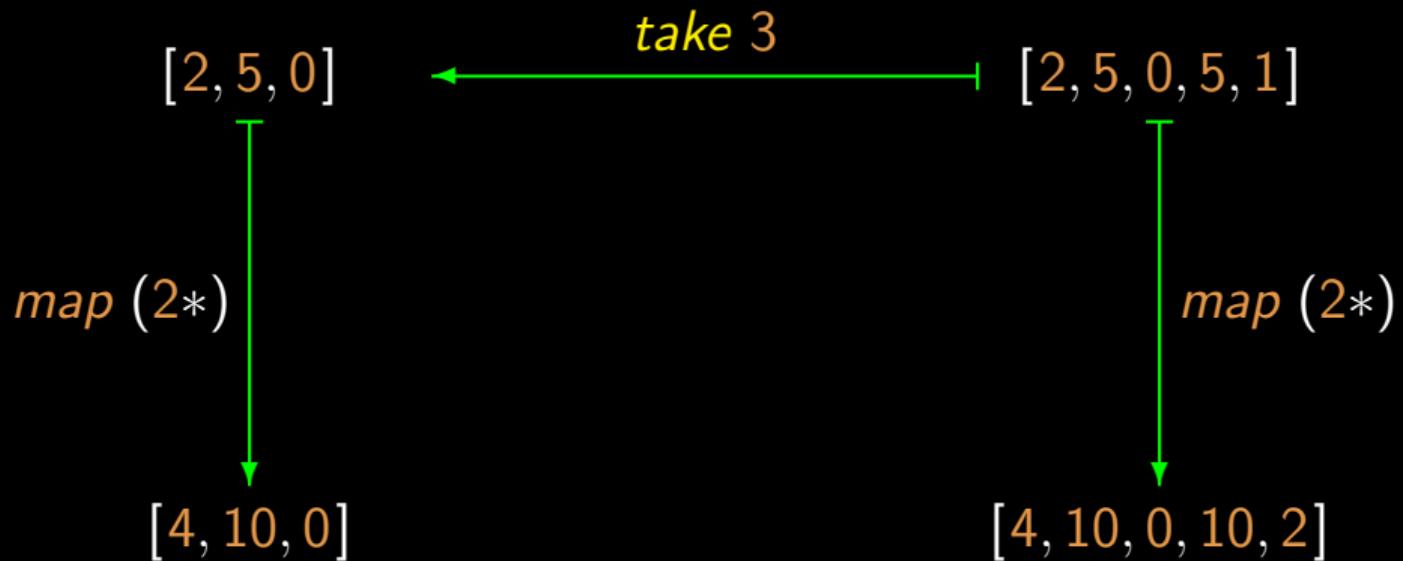
Natural properties



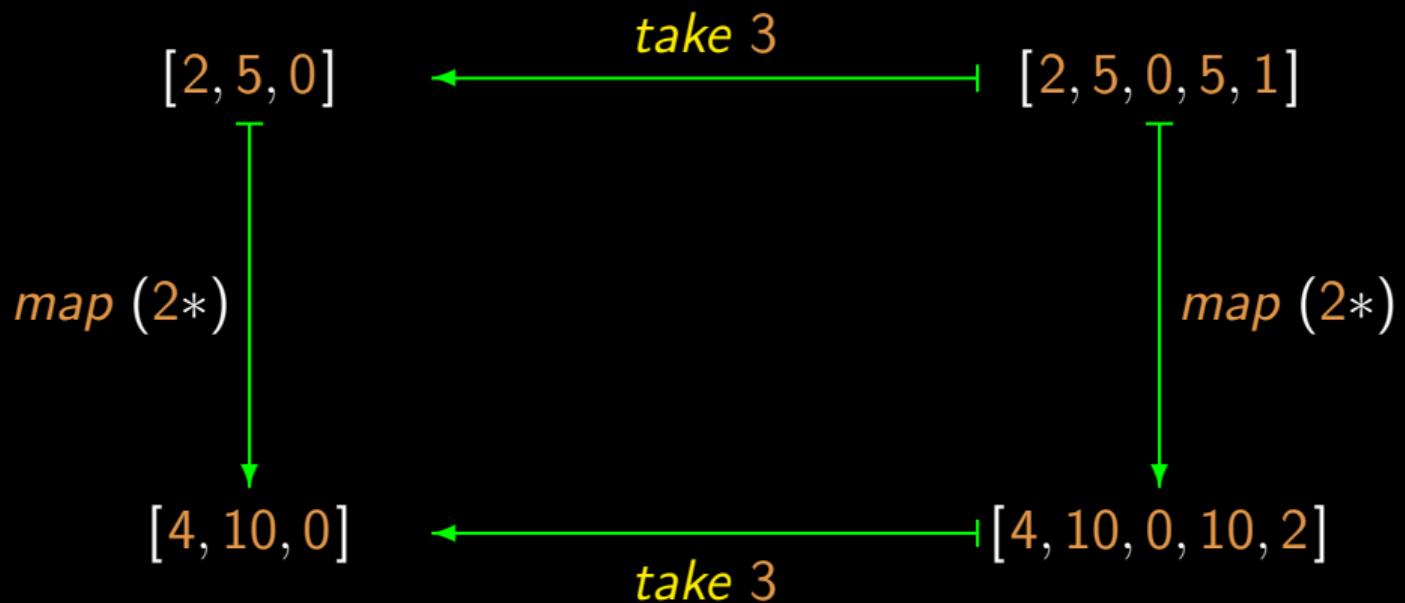
Natural properties



Natural properties



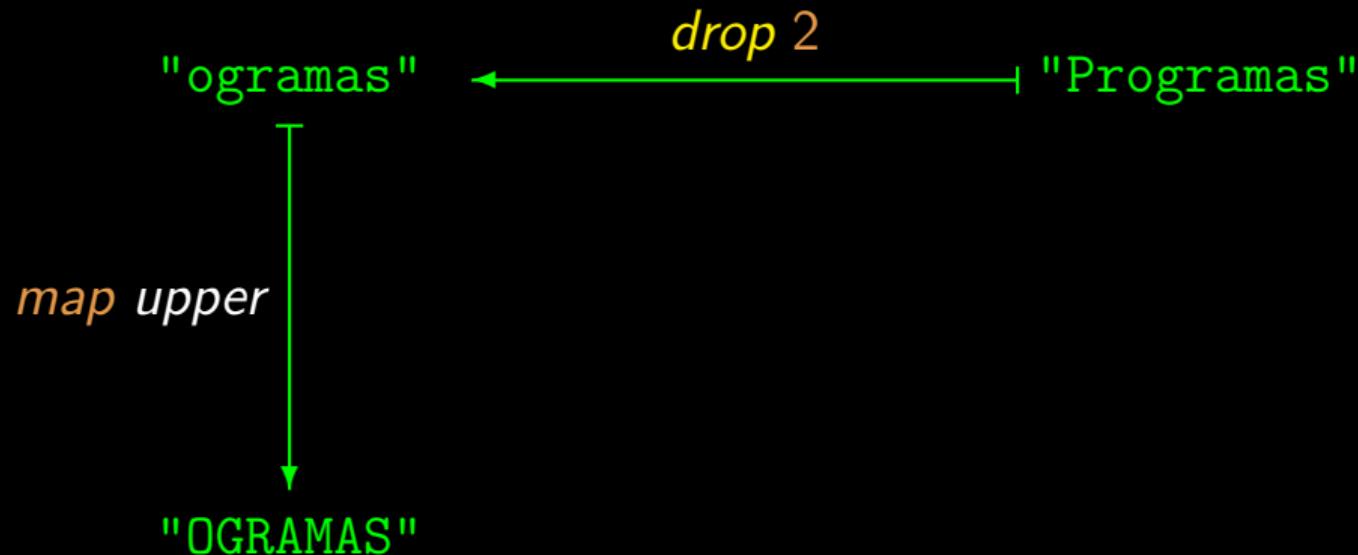
Natural properties



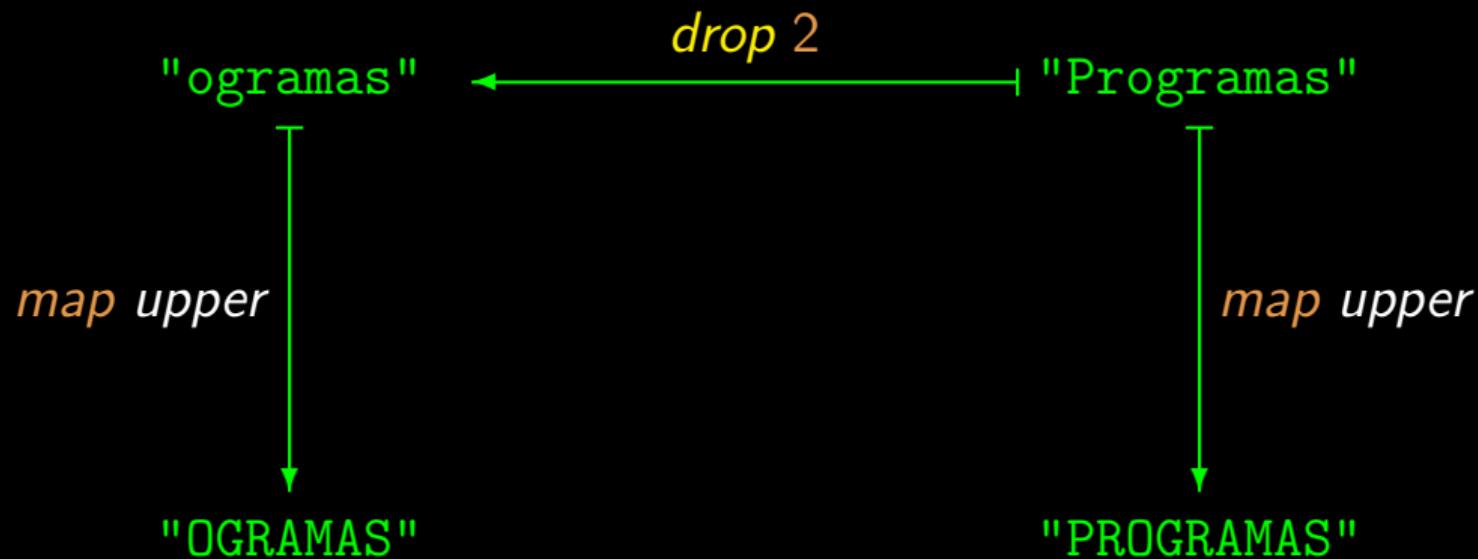
Natural properties



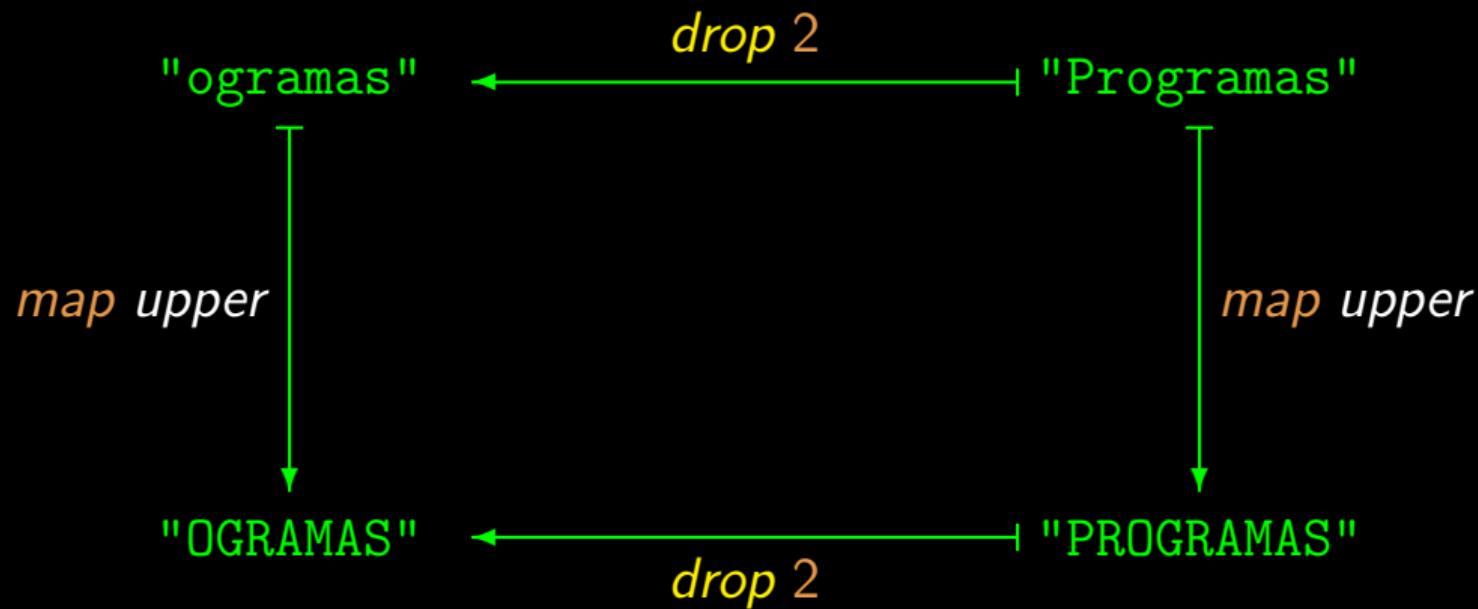
Natural properties



Natural properties



Natural properties



Back to...

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

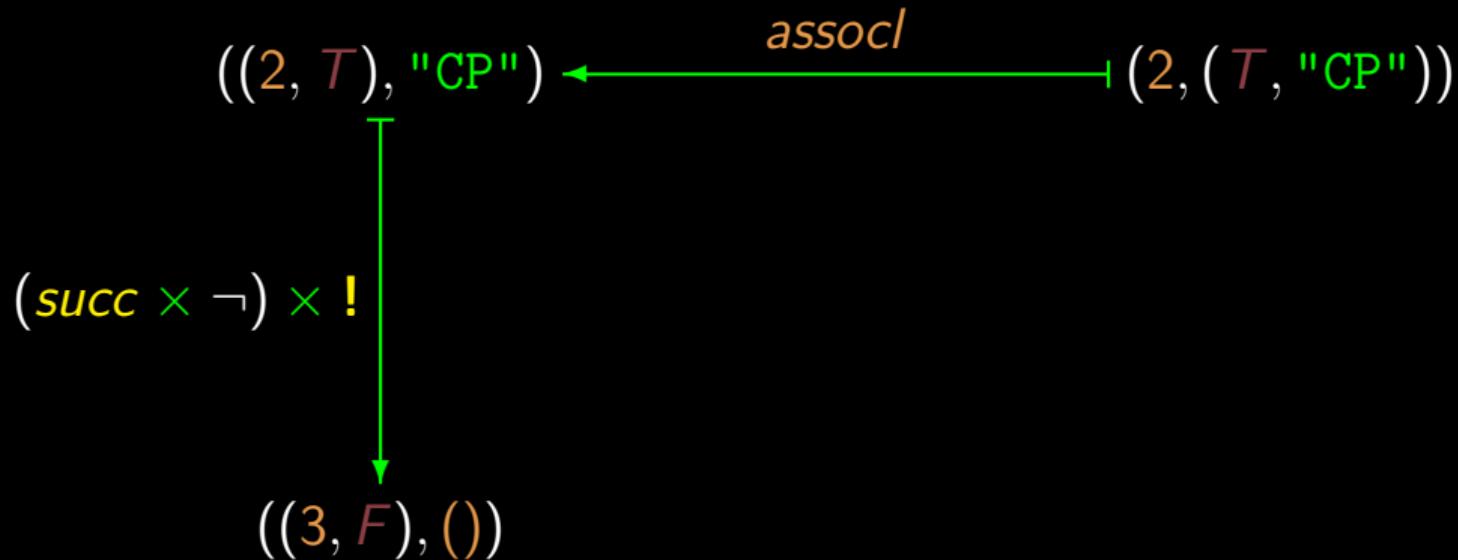
$$\text{assocr} \cdot \text{assocl} = \text{id}$$

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

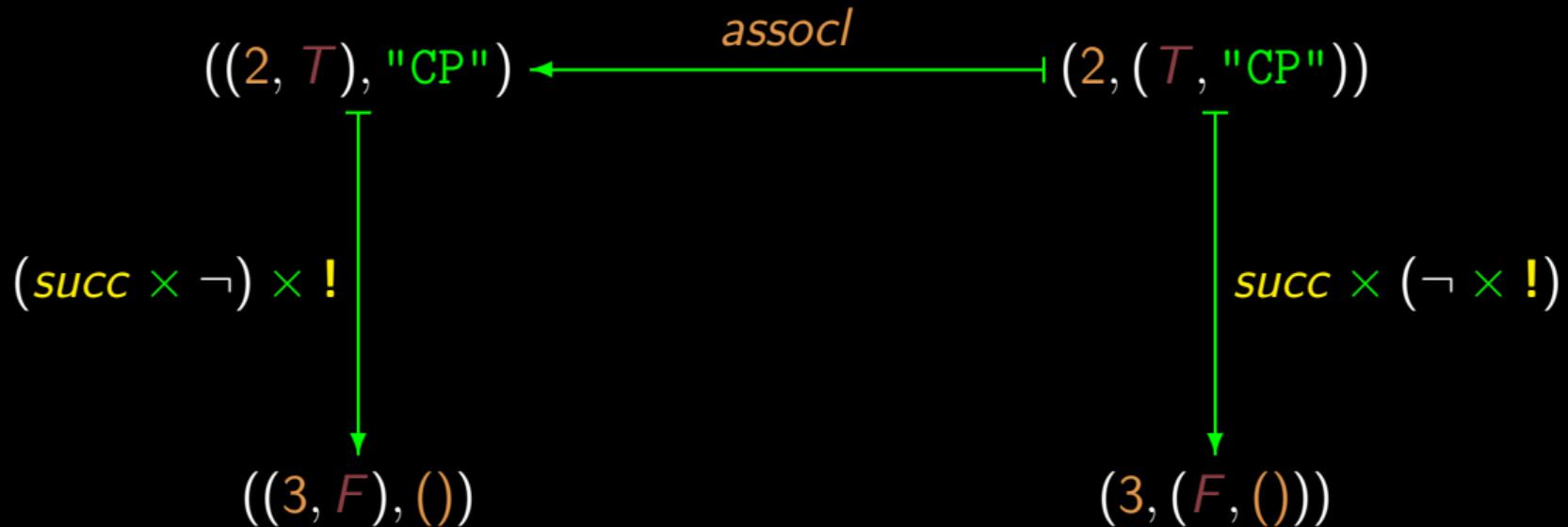
“Natural” property

$$((2, T), "CP") \xleftarrow{assoc!} (2, (T, "CP"))$$

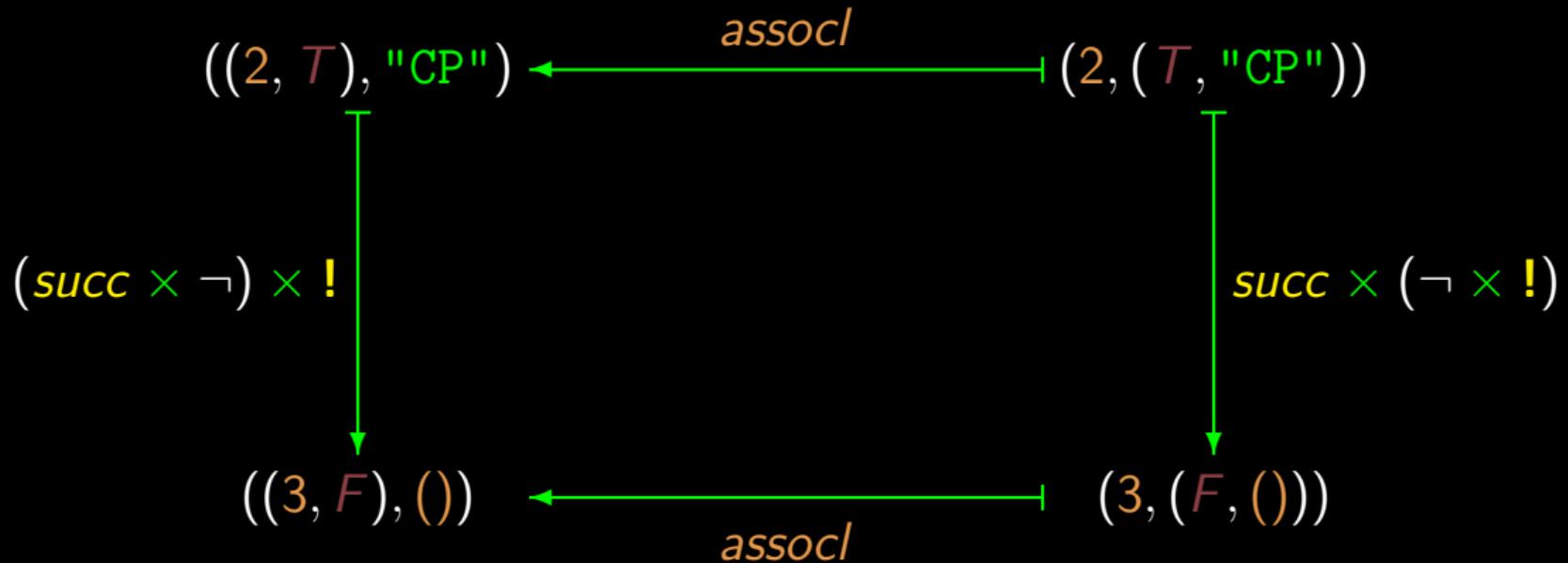
“Natural” property



“Natural” property



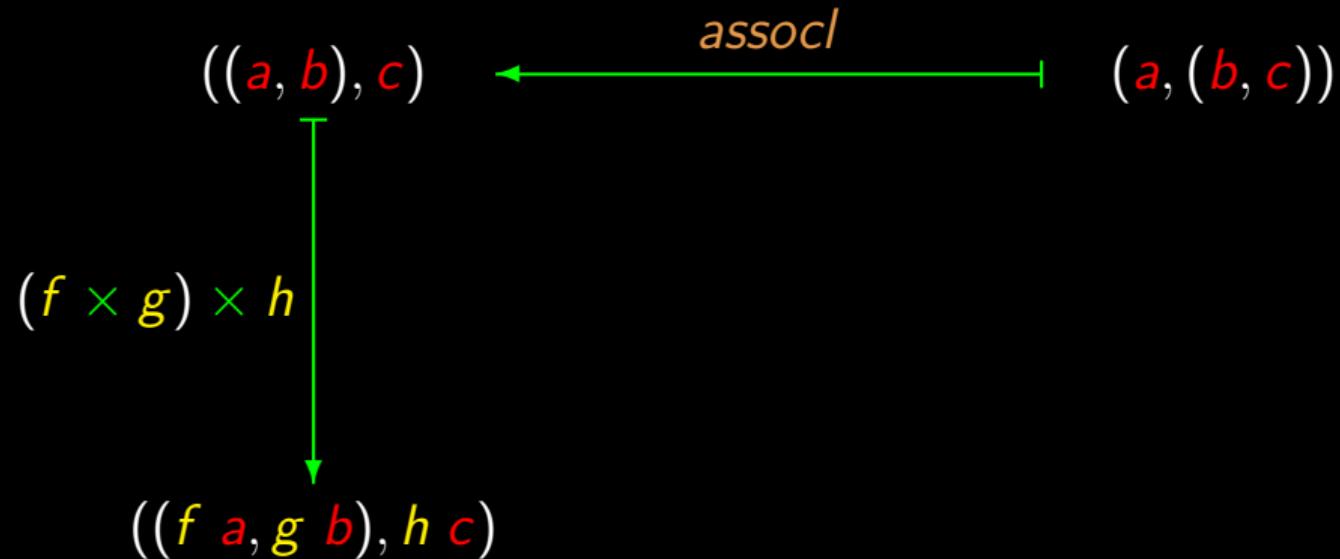
“Natural” property



“Natural” property

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

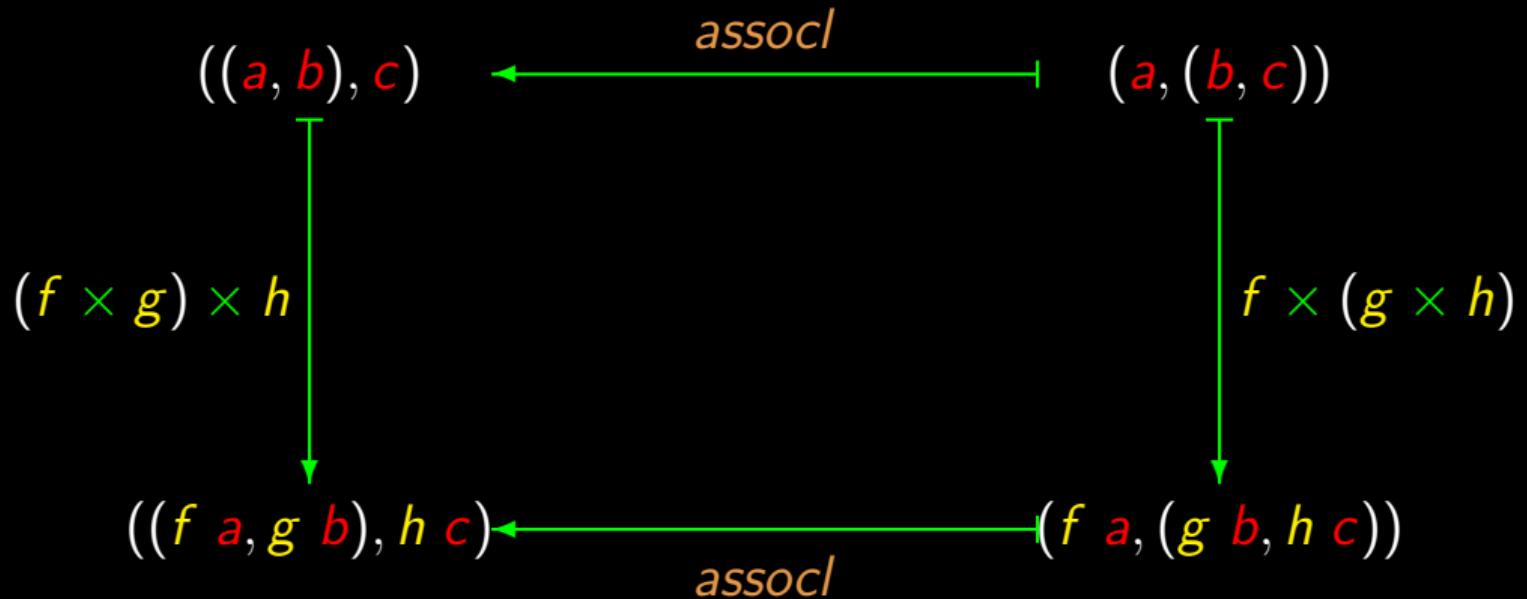
“Natural” property



“Natural” property

$$\begin{array}{ccc} ((a, b), c) & \xleftarrow{\textit{assocl}} & (a, (b, c)) \\ \downarrow & & \downarrow \\ ((f \times g) \times h) & & f \times (g \times h) \\ \downarrow & & \downarrow \\ ((f \ a, g \ b), h \ c) & & (f \ a, (g \ b, h \ c)) \end{array}$$

“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

$$\begin{array}{ccc} (A \times B) \times C & \xleftarrow{\text{assocl}} & A \times (B \times C) \\ \downarrow & & \\ (f \times g) \times h & & \\ \downarrow & & \\ (D \times E) \times F & & \end{array}$$

Natural properties

$$\begin{array}{ccc} (A \times B) \times C & \xleftarrow{\text{assocl}} & A \times (B \times C) \\ \downarrow & & \downarrow \\ (f \times g) \times h & & f \times (g \times h) \\ \downarrow & & \downarrow \\ (D \times E) \times F & & D \times (E \times F) \end{array}$$

Natural properties

$$\begin{array}{ccc} (A \times B) \times C & \xleftarrow{\textit{assocl}} & A \times (B \times C) \\ \downarrow & & \downarrow \\ (f \times g) \times h & & f \times (g \times h) \\ \downarrow & & \downarrow \\ (D \times E) \times F & \xleftarrow{\textit{assocl}} & D \times (E \times F) \end{array}$$

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

Back to...

$$\begin{array}{ccc} & \text{swap} & \\ A \times B & \cong & B \times A \\ & \text{swap} & \end{array}$$

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

Practical rule

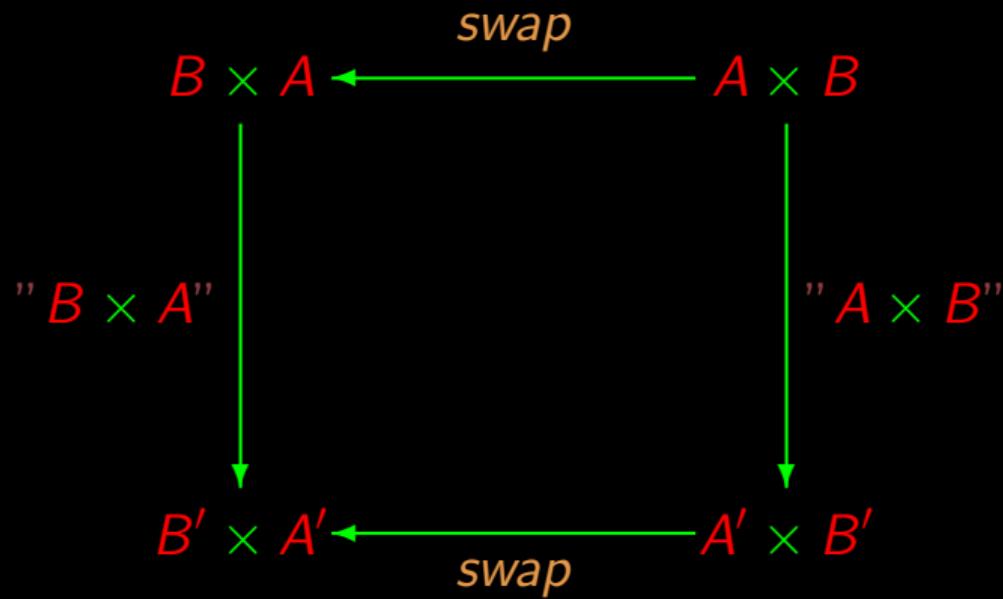
$$B \times A \xleftarrow{\text{swap}} A \times B$$

Practical rule

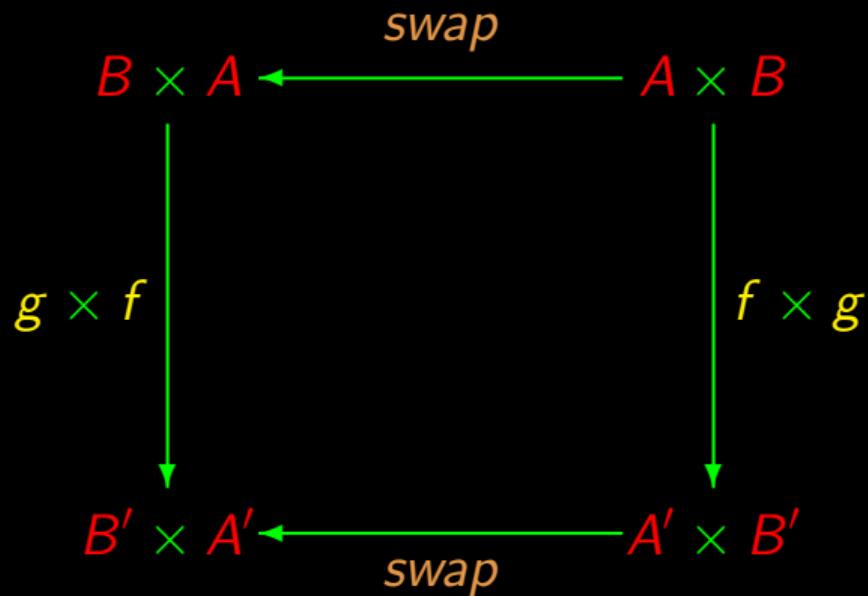
$$B \times A \xleftarrow{\text{swap}} A \times B$$

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

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. $2 := \text{id}$

Practical rule

In the vertical arrows,

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

Simplest case...

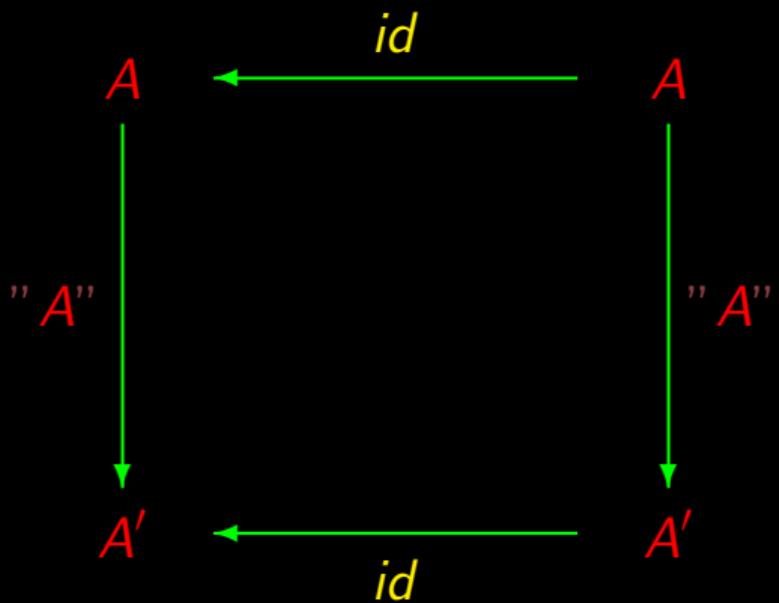


Simplest case...

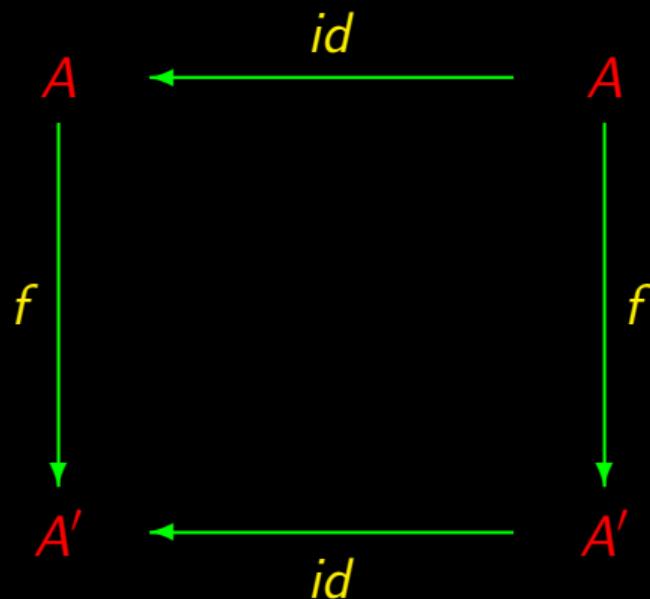
$$A \xleftarrow{id} A$$

$$A' \xleftarrow{id} A'$$

Simplest case...



Natural-*id*



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

Natural- π_1

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

Natural- π_1

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

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

Natural- π_1

$$\begin{array}{ccc} A & \xleftarrow{\pi_1} & A \times B \\ \downarrow "A" & & \downarrow "A \times B" \\ A' & \xleftarrow{\pi_1} & A' \times B' \end{array}$$

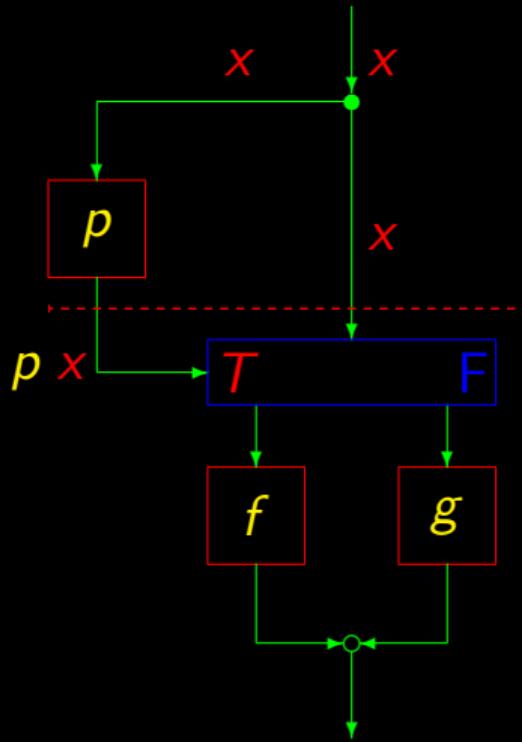
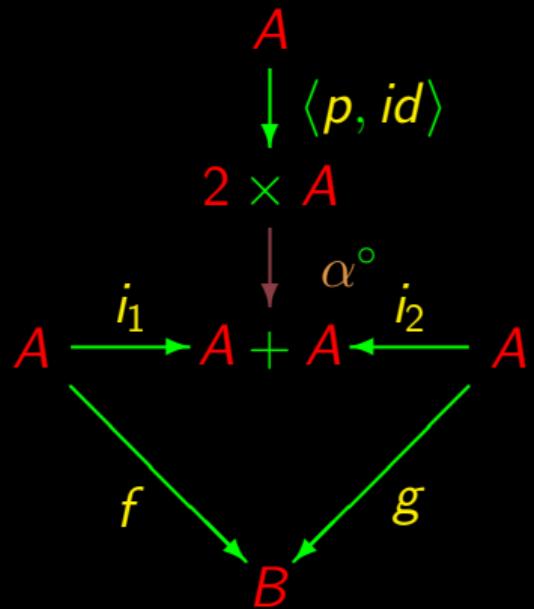
Natural- π_1



$$\begin{array}{ccc} A & \xleftarrow{\pi_1} & A \times B \\ f \downarrow & & \downarrow f \times g \\ A' & \xleftarrow{\pi_1} & A' \times B' \end{array}$$

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

Back to "if-then-else"



2^a fusion law of conditionals

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

2^a 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^a 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 thene...

Let us do it now

Prove

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

knowing:

$$\begin{array}{ccccc} A & \xrightarrow{\langle p, id \rangle} & 2 \times A & \xrightarrow{\alpha^\circ} & A + A \\ & \searrow p? & \swarrow & & \end{array}$$

Recall

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

The diagram illustrates a commutative diagram with two sets of curved arrows. The top set of arrows, labeled α° in green, consists of two curved arrows: one from $2 \times A$ to $A + A$, and another from $A + A$ back to $2 \times A$. The bottom set of arrows, labeled α in brown, also consists of two curved arrows: one from $A + A$ to $2 \times A$, and another from $2 \times A$ back to $A + A$. The entire diagram is enclosed in a black circle.

$$\begin{aligned}\alpha &= [\langle \underline{T}, id \rangle, \langle \underline{F}, id \rangle] \\ \alpha^\circ &= \dots ?\end{aligned}$$

Let us try it

$p?$ · f

Let us try it

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

Let us try it

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

Let us try it

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

Let us try it

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

Let us try it

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

Recall

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

The diagram illustrates a categorical equivalence between the product of a set with itself and its disjoint union. It features two curved arrows: one pointing from left to right labeled α° , and another pointing from right to left labeled α . Between these arrows is a symbol \cong .

$$\begin{aligned}\alpha &= [\langle \underline{T}, id \rangle, \langle \underline{F}, id \rangle] \\ \alpha^\circ &= \dots ?\end{aligned}$$

Natural- α°

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

Natural- α°

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

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

Natural- α°

$$\begin{array}{ccc} A + A & \xleftarrow{\alpha^\circ} & 2 \times A \\ \downarrow "A+A" & & \downarrow "2 \times A" \\ A' + A' & \xleftarrow{\alpha^\circ} & 2 \times A' \end{array}$$

Natural- α°

$$\begin{array}{ccc} A + A & \xleftarrow{\alpha^\circ} & 2 \times A \\ f + f \downarrow & & \downarrow id \times f \\ A' + A' & \xleftarrow{\alpha^\circ} & 2 \times A' \end{array}$$

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

... and this finishes the proof!

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

... and this finishes the proof!

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

... and this finishes the proof!

$$\begin{aligned} p? \cdot f &= \left\{ \begin{array}{l} \text{ } \\ \text{ } \end{array} \right. \quad \left. \begin{array}{l} \text{ } \\ \text{ } \end{array} \right\} \quad \text{ } & \quad \left\{ \begin{array}{l} \text{ } \\ \text{ } \end{array} \right. \quad \left. \begin{array}{l} \text{ } \\ \text{ } \end{array} \right\} \\ = \quad \left\{ \begin{array}{l} p? = \alpha^\circ \cdot \langle p, id \rangle \\ \text{ } \end{array} \right. \quad \left. \begin{array}{l} \text{ } \\ \text{ } \end{array} \right\} &= \alpha^\circ \cdot (id \times f) \cdot \langle p \cdot f, id \rangle \\ \alpha^\circ \cdot \langle p, id \rangle \cdot f &= \left\{ \begin{array}{l} \text{ } \\ \text{ } \end{array} \right. \quad \left. \begin{array}{l} \text{ } \\ \text{ } \end{array} \right\} \\ = \quad \left\{ \begin{array}{l} \text{ } \\ \text{ } \end{array} \right. \quad \left. \begin{array}{l} \text{ } \\ \text{ } \end{array} \right\} &= (f + f) \cdot \alpha^\circ \cdot \langle p \cdot f, id \rangle \\ \alpha^\circ \cdot \langle p \cdot f, id \cdot f \rangle &= \left\{ \begin{array}{l} \text{ } \\ \text{ } \end{array} \right. \quad \left. \begin{array}{l} \text{ } \\ \text{ } \end{array} \right\} \\ = \quad \left\{ \begin{array}{l} \text{ } \\ \text{ } \end{array} \right. \quad \left. \begin{array}{l} \text{ } \\ \text{ } \end{array} \right\} &= (f + f) \cdot (p \cdot f)? \\ \alpha^\circ \cdot \langle id \cdot p \cdot f, f \cdot id \rangle & \end{aligned}$$



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

$2 + 3$

infix notation

operator between

$+ 2 3$

prefix notation

operator before

$2 3 +$

postfix notation

operator after

$+ (2, 3)$

prefix “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

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 g

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 $g\ (a, b)$

Curry | uncurry

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

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

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

uncurry $g\ (a, b) =$

Curry | uncurry

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

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

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

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

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

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

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

Curry | uncurry

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

□

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

curry (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

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

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

Curry | uncurry

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

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

Curry | uncurry

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

□

Isomorphism

$$\begin{array}{ccc} (a, b) \rightarrow c & \cong & a \rightarrow b \rightarrow c \\ \text{curry} \curvearrowright & & \curvearrowleft \text{uncurry} \end{array}$$

The type B^A

The type B^A

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

NB: functions that map A to B

Isomorphism

$$A \times B \xrightarrow{\text{curry}} A \rightarrow (B \rightarrow C) \quad \approx \quad A \rightarrow (B \rightarrow C) \xleftarrow{\text{uncurry}}$$

Isomorphism

$$\begin{array}{ccc} A \times B \rightarrow C & \cong & A \rightarrow C^B \\ \text{curry} \nearrow \searrow \text{uncurry} & & \end{array}$$

Isomorphism

$$\begin{array}{ccc} & \text{curry} & \\ C^{A \times B} & \cong & (C^B)^A \\ & \text{uncurry} & \end{array}$$

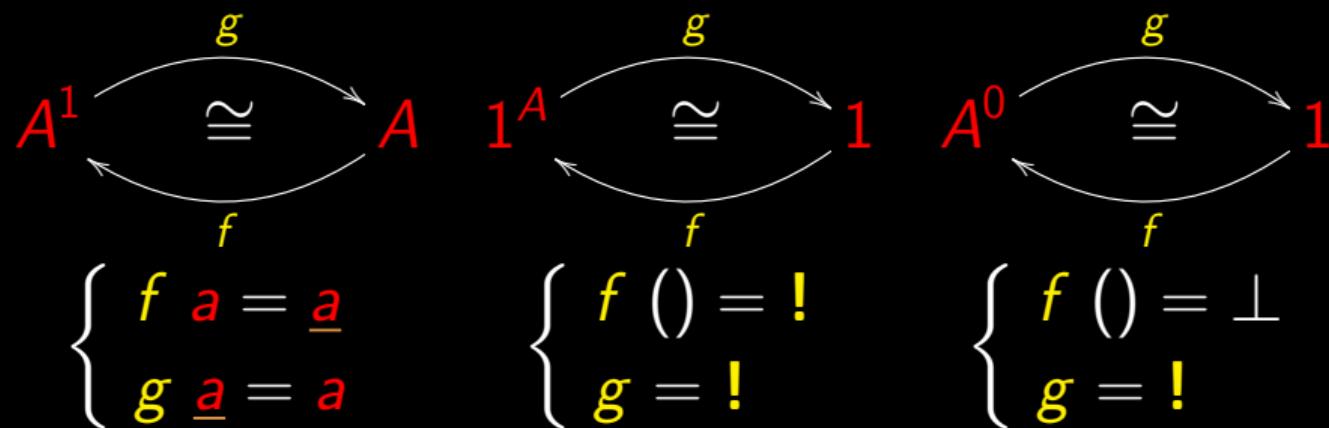
Cf. arithmetic exponentials: $c^{a b} = (c^b)^a$

More isomorphisms

$$a^1 = a$$

$$1^a = 1$$

$$a^0 = 1$$



Curry | uncurry

$$\begin{array}{ccc} A \times B \rightarrow C & \cong & A \rightarrow C^B \\ \text{curry} \nearrow \searrow \text{uncurry} & & \end{array}$$

Curry | uncurry

$$f : A \times B \rightarrow C \quad \approx \quad A \rightarrow C^B$$

curry uncurry

The diagram illustrates the Curry-Howard correspondence. It shows two equivalent representations of a function f . On the left, f is a function from the product type $A \times B$ to the type C , represented as $A \times B \rightarrow C$. On the right, f is a function from the domain type A to the function type C^B , represented as $A \rightarrow C^B$. These two types are connected by a horizontal double-headed arrow labeled with the symbol \approx , indicating they are isomorphic. Above this arrow, the word "curry" is written above a curved arrow pointing from the left side to the right side. Below the arrow, the word "uncurry" is written below a curved arrow pointing from the right side to the left side.

Curry | uncurry

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

$$\begin{array}{ccc} A \times B \xrightarrow{\text{curry}} & \cong & A \xrightarrow{\text{curry}} C^B \\ \text{uncurry} & \swarrow & \uparrow \end{array}$$

Curry | uncurry

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

$$\begin{array}{ccc} A \times B \xrightarrow{\text{curry}} & \cong & A \xrightarrow{\text{curry}} C^B \\ \text{uncurry} & \swarrow & \uparrow \end{array}$$

k

Curry | uncurry

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

$$\begin{array}{ccc} A \times B \xrightarrow{\text{curry}} & \cong & A \xrightarrow{\text{curry}} C^B \\ \text{uncurry} & \swarrow & \uparrow \\ & & \end{array}$$

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

Cálculo de Programas

Class T06

Curry | uncurry

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

$$\begin{array}{ccc} A \times B \xrightarrow{\text{curry}} & \cong & A \xrightarrow{\text{curry}} C^B \\ \text{uncurry} & \swarrow & \uparrow \\ & & \end{array}$$

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

Curry | uncurry

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

$$\begin{array}{ccc} & \text{curry} & \\ A \times B \rightarrow C & \cong & A \rightarrow C^B \\ & \text{uncurry} & \end{array}$$

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

Curry | uncurry

$$\begin{array}{ccc} & \text{curry} & \\ A \times B \rightarrow C & \cong & A \rightarrow C^B \\ & \text{uncurry} & \end{array}$$

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

Curry | uncurry

$k = \text{curry } f$

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

$\text{uncurry } k = f$

Curry | uncurry

$k = \text{curry } f$

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

$\text{uncurry } k = f$

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

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

Curry | uncurry

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

$k = \text{curry } f$

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

$\text{uncurry } k = f$

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

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

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

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

Curry | uncurry

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

$k = \text{curry } f$

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

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

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

$\text{uncurry } k = f$

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

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

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

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

Curry | uncurry

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

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

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

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

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

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

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

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

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

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

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

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

Curry | uncurry

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

$k = \text{curry } f$

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

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

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

$\text{uncurry } k = f$

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

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

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

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

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

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

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

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

Curry | uncurry

$$(k \ a) \ b = f \ (a, b)$$
$$k = \text{curry } f \Leftrightarrow \{ \text{ introduce } ap \ (f, x) = f \ x \ }$$
$$\Leftrightarrow \{ \text{ isomorphism (last slide)} \} \qquad ap \ (k \ a, b) = f \ (a, b)$$
$$\text{uncurry } k = f \Leftrightarrow \{ \text{ add id} \}$$
$$\Leftrightarrow \{ \text{ add variables} \} \qquad ap \ (k \ a, id \ b) = f \ (a, b)$$
$$\text{uncurry } k \ (a, b) = f \ (a, b) \Leftrightarrow \{ \text{ composition and } \times \}$$
$$\Leftrightarrow \{ \text{ definition of uncurry } k \} \qquad (ap \cdot (k \times id)) \ (a, b) = f \ (a, b)$$
$$k \ a \ b = f \ (a, b) \Leftrightarrow \{ \text{ drop variables} \}$$
$$ap \cdot (k \times id) = f$$

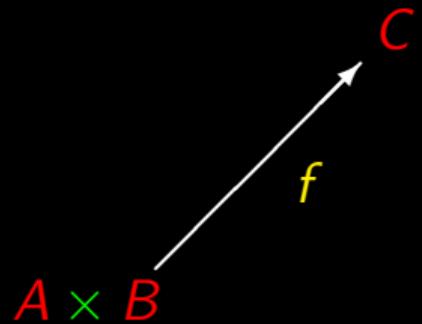
Universal property



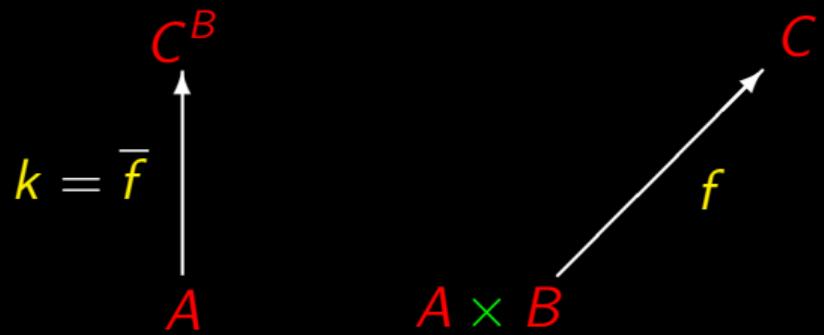
$$\begin{array}{ccc} A \times B \rightarrow C & \cong & A \rightarrow C^B \\ \text{curry} \curvearrowright & & \curvearrowleft \text{uncurry} \end{array}$$

$$k = \text{curry } f \Leftrightarrow ap \cdot (k \times id)$$

Diagrams

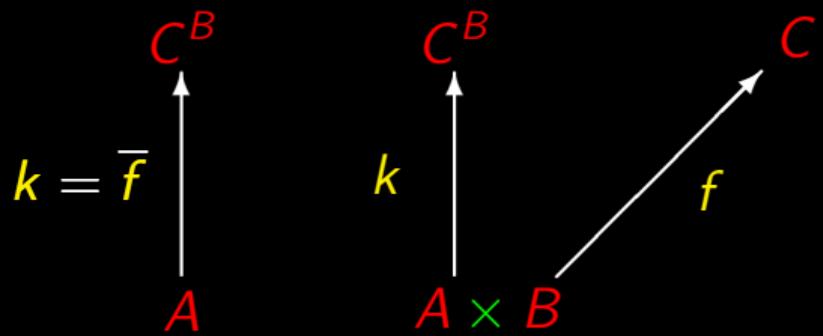


Diagrams



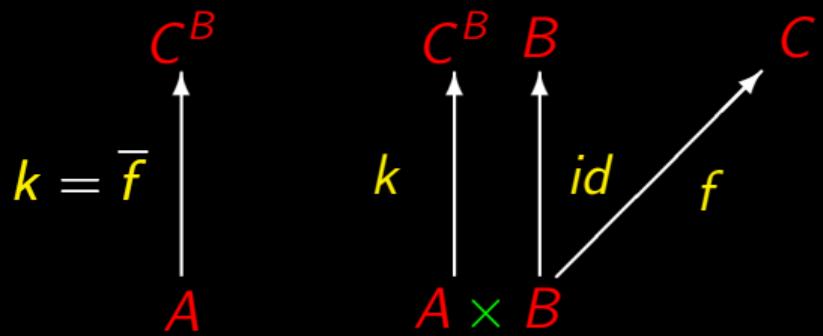
\bar{f} abbreviates curry f

Diagrams



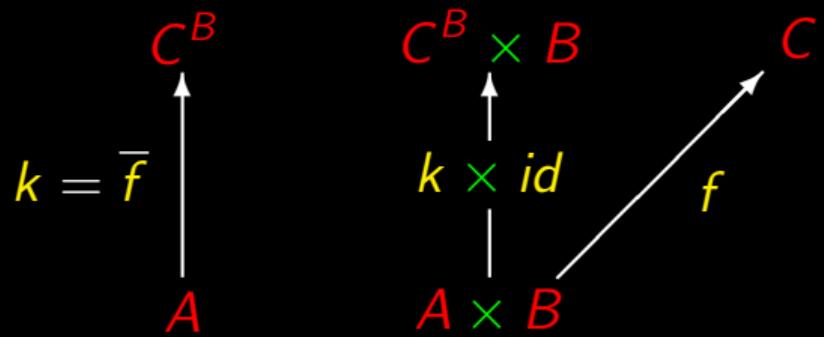
\bar{f} abbreviates curry f

Diagrams



\bar{f} abbreviates *curry* f

Universal property



Universal property

$$\begin{array}{ccc} C^B & & \\ \uparrow k = \bar{f} & & \\ A & & \\ & & \\ C^B \times B & \xrightarrow{\text{ap}} & C \\ \uparrow k \times id & & \swarrow f \\ A \times B & & \end{array}$$

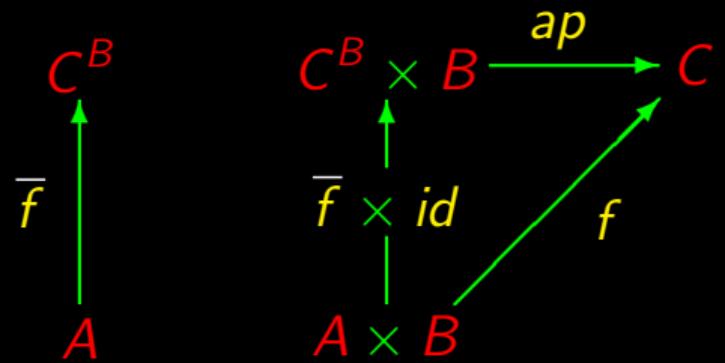
$$k = \bar{f} \Leftrightarrow \text{ap} \cdot (k \times \text{id}) \ k = f$$

Reflexion ($k := id$)

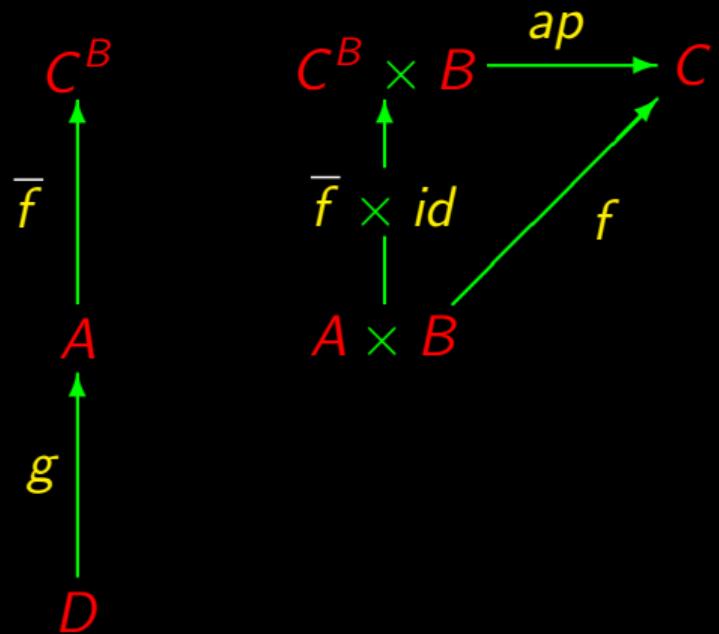
$$\begin{array}{ccc} C^B & & \\ \uparrow & & \\ id = \bar{f} & & \\ & & \\ C^B \times B & \xrightarrow{ap} & C \\ \uparrow k \times id & & \swarrow f \\ C^B \times B & & \end{array}$$

$$id = \bar{f} \Leftrightarrow ap = f$$

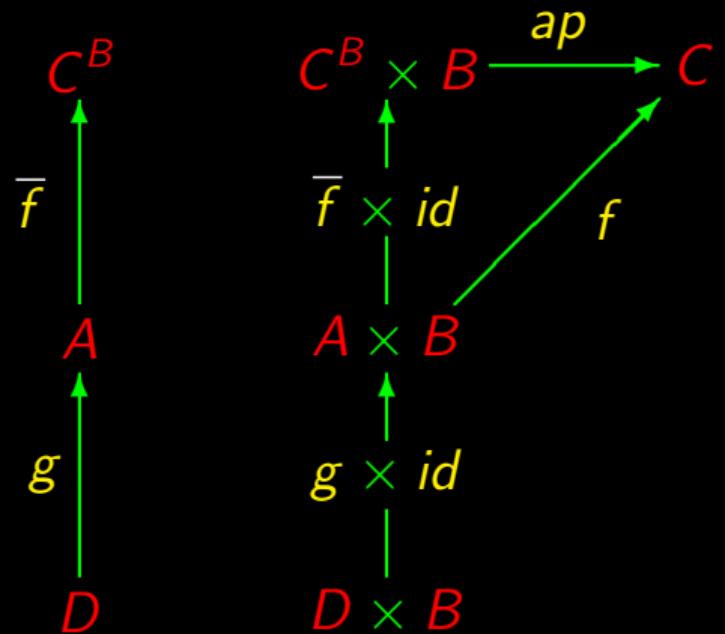
Fusion



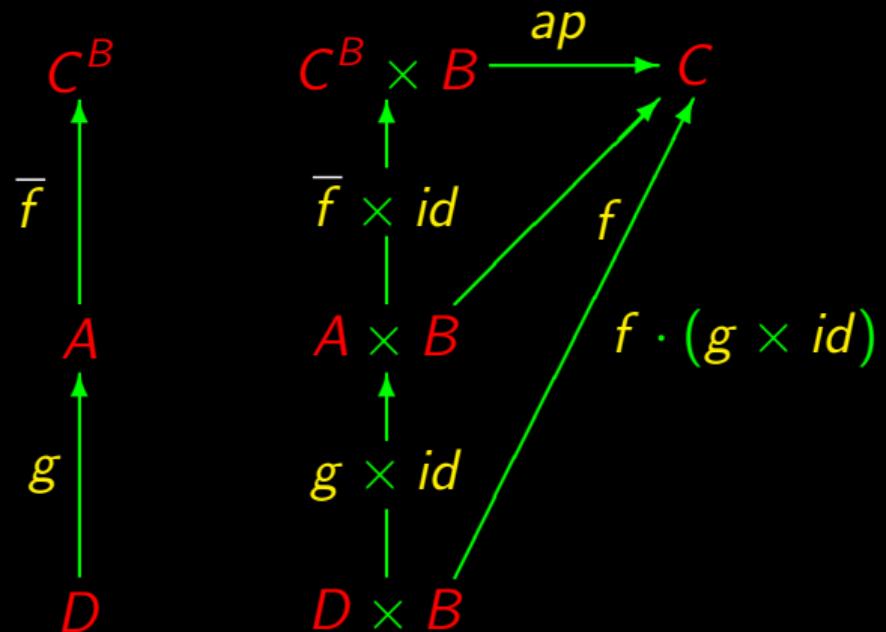
Fusion



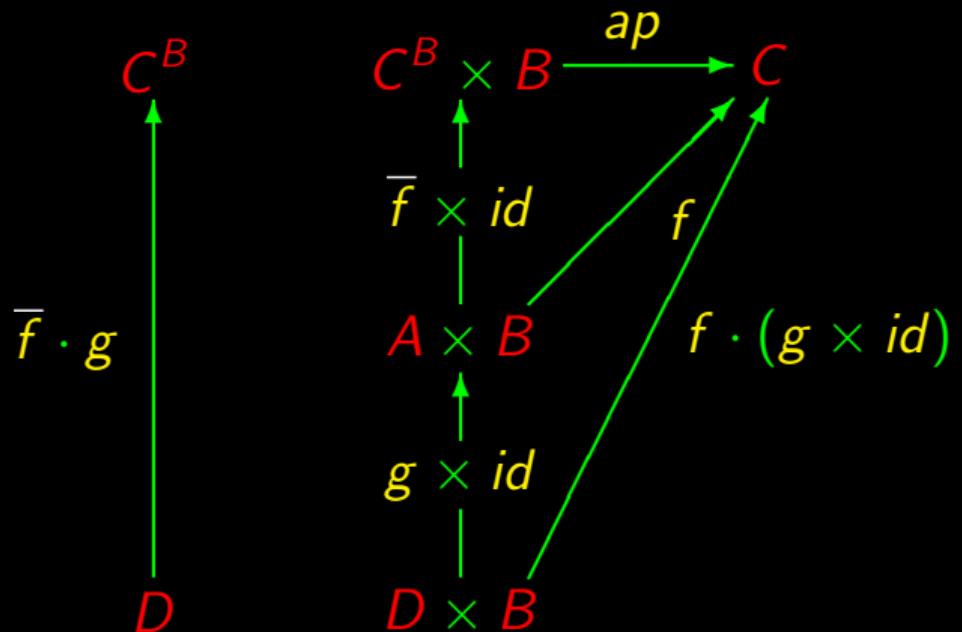
Fusion



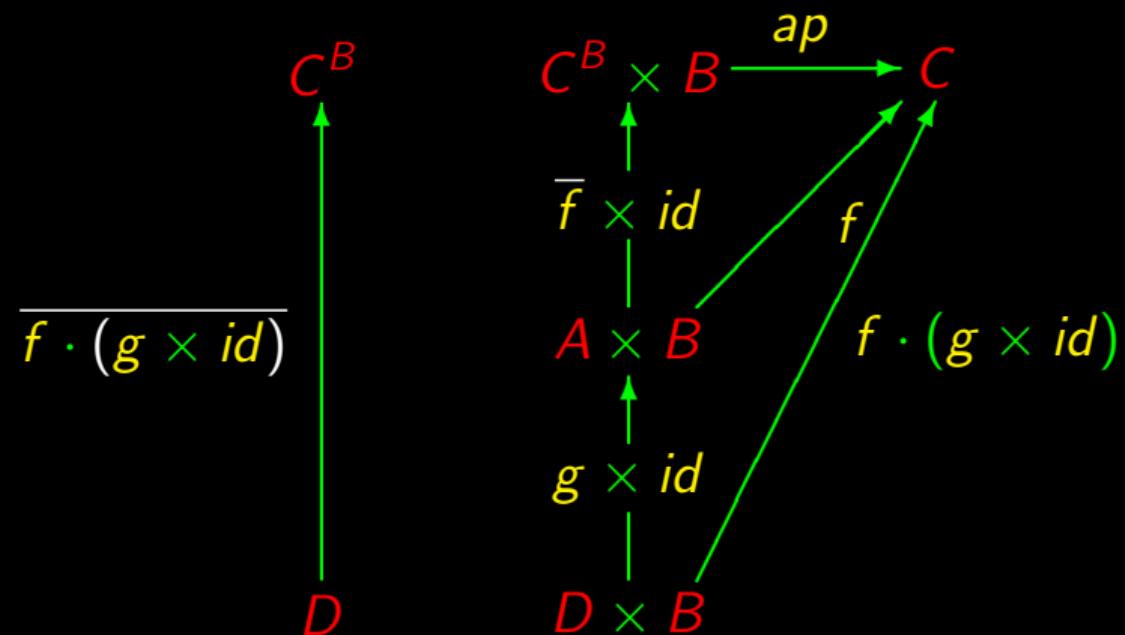
Fusion



Fusion



Fusion

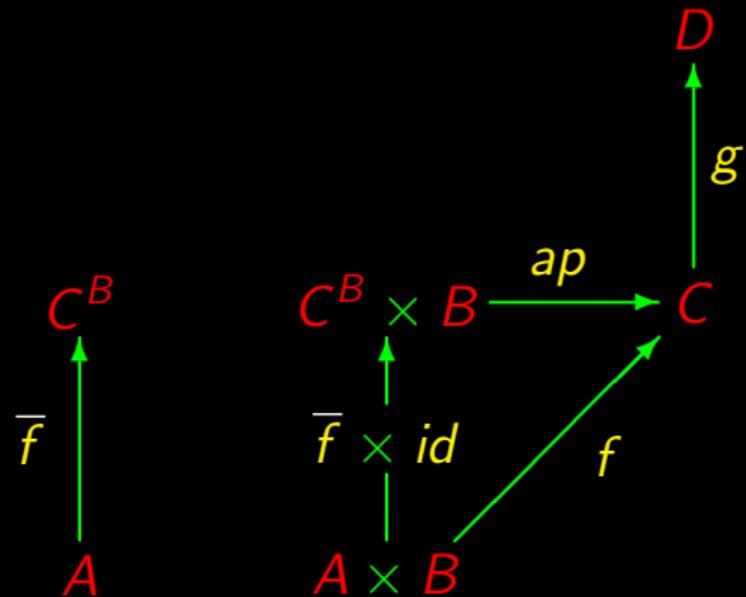


$$\bar{f} \cdot g = \overline{f \cdot (g \times id)}$$

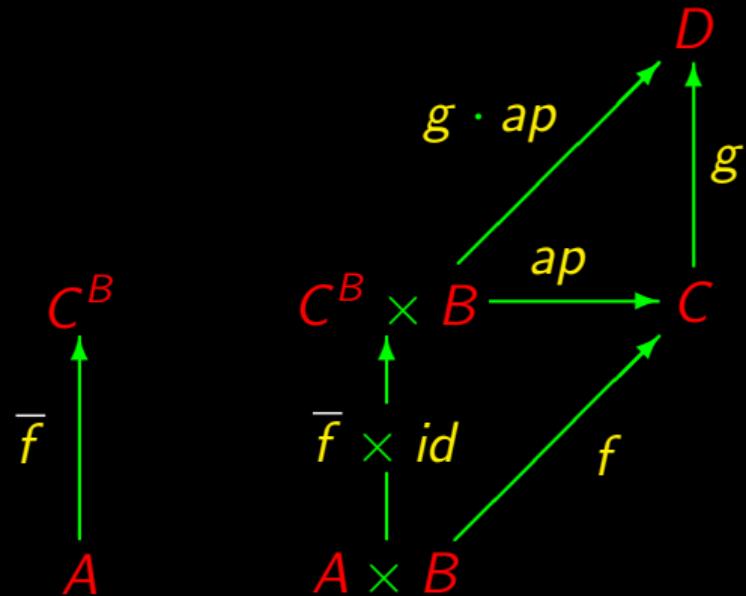
Absorption (exponentials)

$$\begin{array}{ccc} C^B & & \\ \uparrow \bar{f} & & \\ A & & \\ & & \\ C^B \times B & \xrightarrow{ap} & C \\ \uparrow \bar{f} \times id & & \\ A \times B & & \end{array}$$

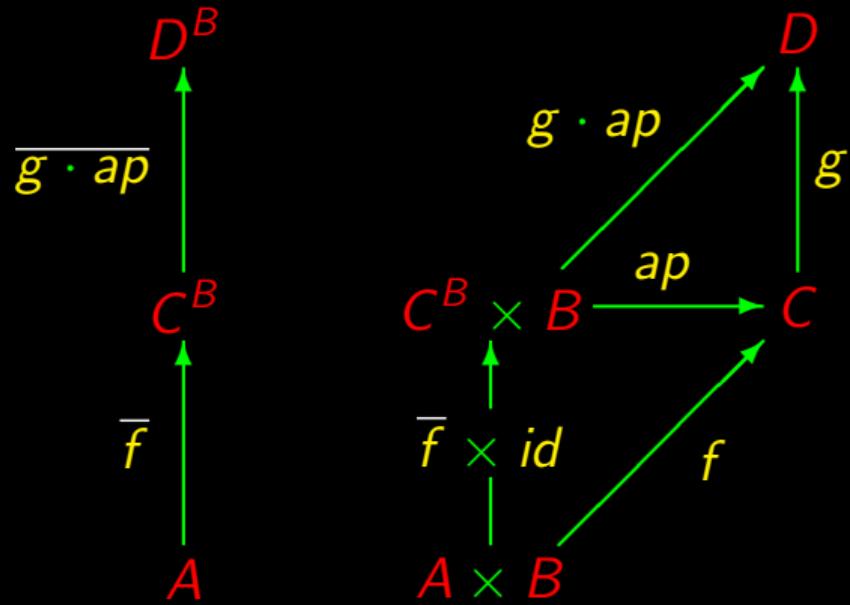
Absorption (exponentials)



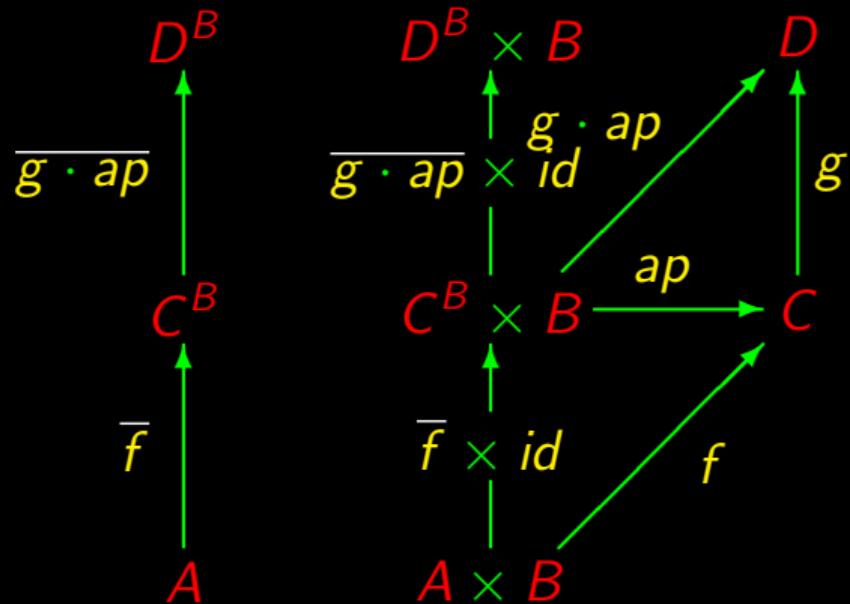
Absorption (exponentials)



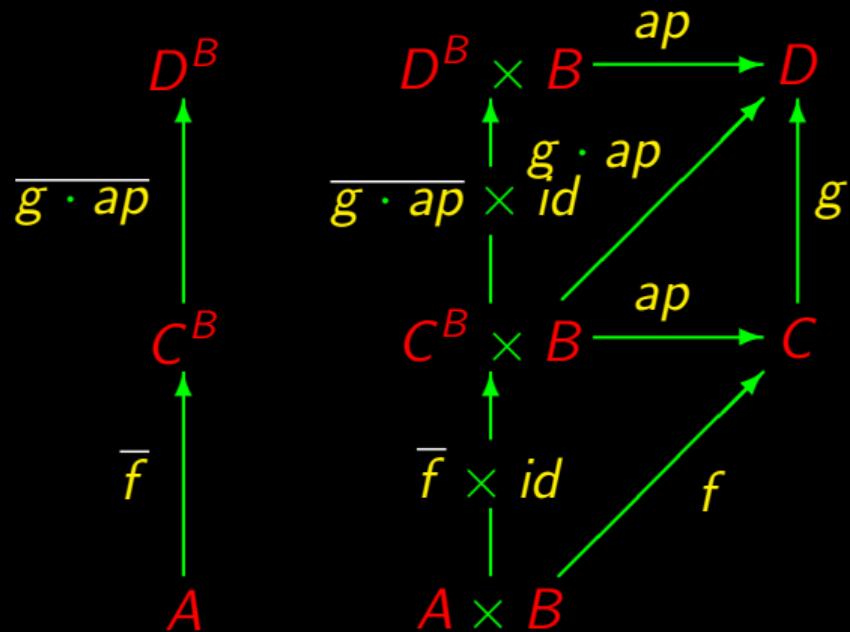
Absorption (exponentials)



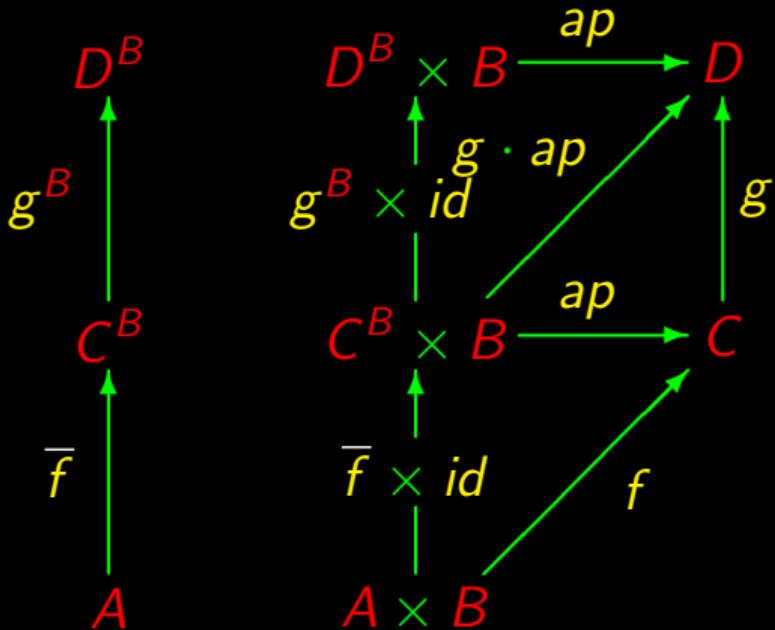
Absorption (exponentials)



Absorption (exponentials)

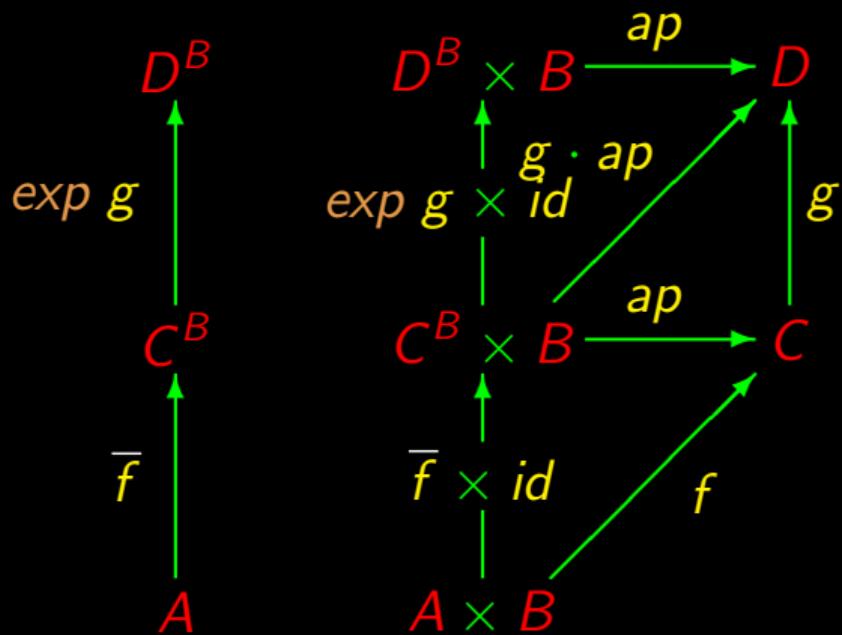


Absorption



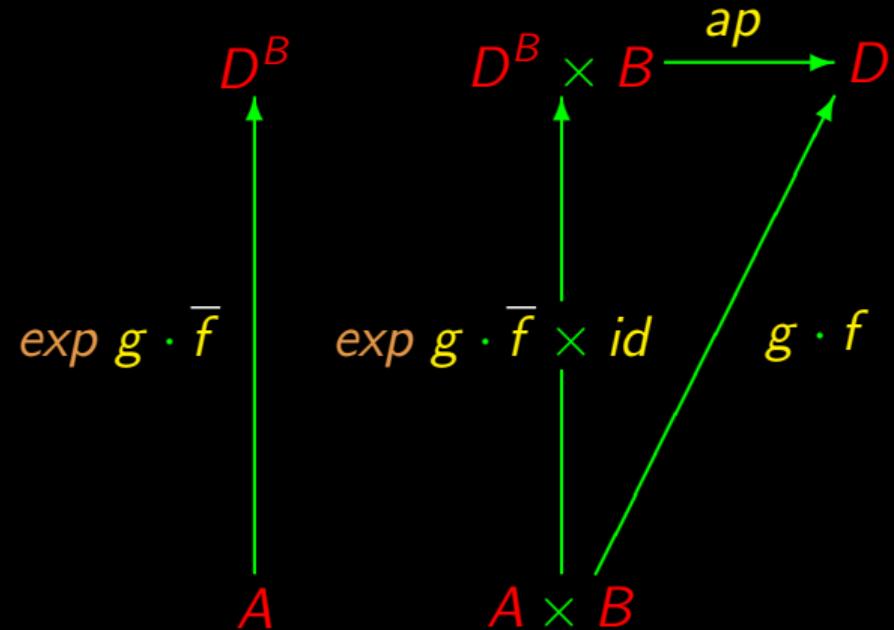
$$g^B = \overline{g \cdot ap}$$

Absorption

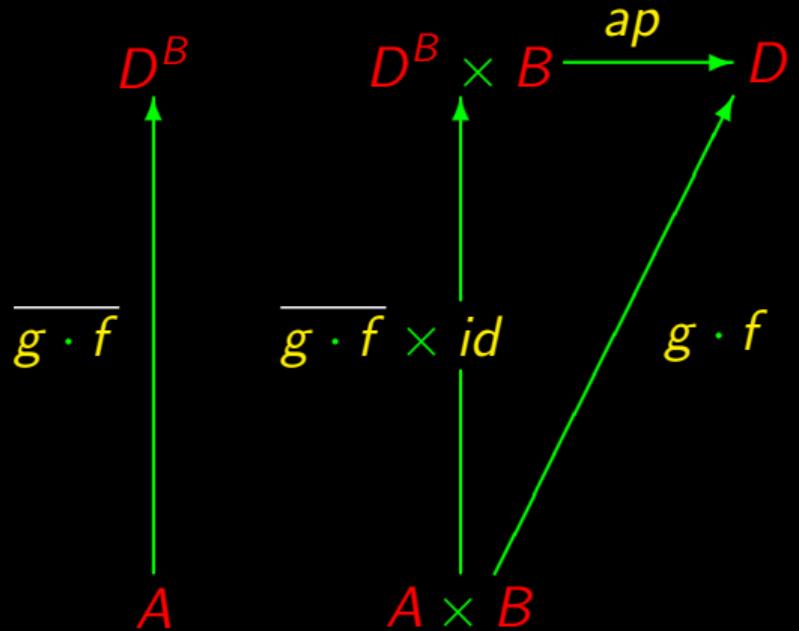


$$\exp g = \overline{g \cdot ap}$$

Absorption



Absorption



$$\overline{g \cdot f} = \exp g \cdot \overline{f}$$

Curry | uncurry

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

$$\begin{array}{ccc} & \text{curry} & \\ A \times B \rightarrow C & \approx & A \rightarrow C^B \\ & \text{uncurry} & \end{array}$$

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

Curry | uncurry

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

Abbreviations:

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

$$A \times B \xrightarrow{\text{curry}} C \underset{\cong}{\sim} A \rightarrow C^B$$

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

Curry | uncurry

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

$$\begin{array}{ccc} A \times B & \xrightarrow{\text{curry}} & A \rightarrow C^B \\ \text{uncurry} & \approx & \end{array}$$

Abbreviations:

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

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

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

Curry | uncurry

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

$$A \times B \xrightarrow{\text{curry}} C \underset{\cong}{\sim} A \rightarrow C^B$$

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

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

\overline{f}

Summary

$f \cdot g$

sequential

$\langle f, g \rangle, f \times g$

parallel

$[f, g], f + g, p \rightarrow f, g$

alternation

$\bar{f}, \exp f$

higher order

Summary

$f \cdot g$

sequential

$\langle f, g \rangle, f \times g$

parallel

$[f, g], f + g, p \rightarrow f, g$

alternation

$\bar{f}, \exp f$

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?

$$\left\{ \begin{array}{l} \textcolor{red}{a} \times 0 = 0 \\ \textcolor{red}{a} \times (\textcolor{red}{b} + 1) = \textcolor{red}{a} \times b + a \end{array} \right.$$

Where do programs come from?

$$\left\{ \begin{array}{l} \textcolor{red}{a} \times 0 = 0 \\ \textcolor{red}{a} \times (\textcolor{blue}{b} + 1) = \textcolor{red}{a} \times \textcolor{blue}{b} + \textcolor{red}{a} \end{array} \right.$$

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?

$$\left\{ \begin{array}{l} (\textcolor{red}{a} \times) \ 0 = 0 \\ (\textcolor{red}{a} \times) (\textcolor{red}{b} + 1) = \textcolor{red}{a} + (\textcolor{red}{a} \times) \ b \end{array} \right.$$

What **is** a program?

$$\left\{ \begin{array}{l} (\textcolor{red}{a} \times) \ 0 = 0 \\ (\textcolor{red}{a} \times) (\textcolor{red}{b} + 1) = (\textcolor{red}{a}+) ((\textcolor{red}{a} \times) \ b) \end{array} \right.$$

What **is** a program?

$$\left\{ \begin{array}{l} (\textcolor{red}{a}\times) (\underline{0} \textcolor{red}{x}) = \underline{0} \textcolor{red}{x} \\ (\textcolor{red}{a}\times) (\textcolor{red}{b} + 1) = (\textcolor{red}{a}+) ((\textcolor{red}{a}\times) \textcolor{red}{b}) \end{array} \right.$$

What **is** a program?

$$\left\{ \begin{array}{l} (\textcolor{red}{a} \times) (\underline{0} \textcolor{red}{x}) = \underline{0} \textcolor{red}{x} \\ (\textcolor{red}{a} \times) (\textcolor{blue}{succ} \textcolor{red}{b}) = (\textcolor{red}{a}+) ((\textcolor{green}{a} \times) \textcolor{red}{b}) \end{array} \right.$$

What **is** a program?

$$\left\{ \begin{array}{l} ((a\times) \cdot \underline{0})\ x = \underline{0}\ x \\ (a\times) (\textit{succ } b) = (a+) ((a\times) b) \end{array} \right.$$

What **is** a program?

$$\left\{ \begin{array}{l} ((a\times) \cdot \underline{0})\ x = \underline{0}\ x \\ ((a\times) \cdot succ)\ b = (a+) ((a\times) b) \end{array} \right.$$

What **is** a program?

$$\left\{ \begin{array}{l} ((a\times) \cdot \underline{0})\ x = \underline{0}\ x \\ ((a\times) \cdot succ)\ b = ((a+) \cdot (a\times))\ b \end{array} \right.$$

What **is** a program?

$$\left\{ \begin{array}{l} (\textcolor{red}{a} \times) \cdot \underline{0} = \underline{0} \\ ((\textcolor{red}{a} \times) \cdot \textit{succ}) \ b = ((\textcolor{red}{a}+) \cdot (\textcolor{green}{a} \times)) \ b \end{array} \right.$$

What **is** a program?

$$\left\{ \begin{array}{l} (\textcolor{red}{a}\times) \cdot \underline{0} = \underline{0} \\ (\textcolor{red}{a}\times) \cdot succ = (\textcolor{red}{a}+) \cdot (\textcolor{red}{a}\times) \end{array} \right.$$

What **is** a program?

$$\left\{ \begin{array}{l} (\textcolor{red}{a}\times) \cdot \underline{0} = \underline{0} \\ (\textcolor{red}{a}\times) \cdot succ = (\textcolor{red}{a}+) \cdot (\textcolor{red}{a}\times) \end{array} \right.$$

What **is** a program?

$$[(\textcolor{red}{a} \times) \cdot \underline{0}, (\textcolor{red}{a} \times) \cdot \textit{succ}] = [\underline{0}, (\textcolor{red}{a} +) \cdot (\textcolor{red}{a} \times)]$$

What **is** a program?

$$(a \times) \cdot [\underline{0}, \text{succ}] = [\underline{0}, (a+) \cdot (a \times)]$$

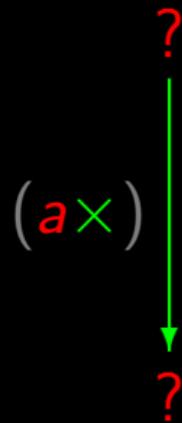
What **is** a program?

$$(a \times) \cdot [\underline{0}, \textit{succ}] = [\underline{0} \cdot \textit{id}, (a+) \cdot (a \times)]$$

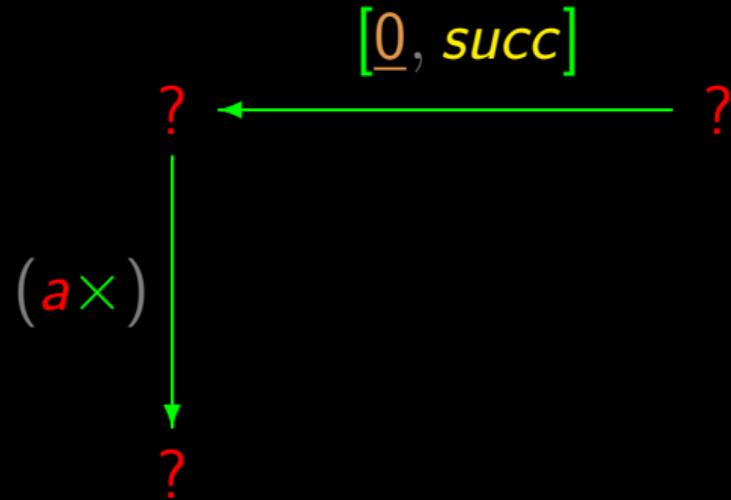
What **is** a program?

$$(a \times) \cdot [\underline{0}, \text{succ}] = [\underline{0}, (a +)] \cdot (id + (a \times))$$

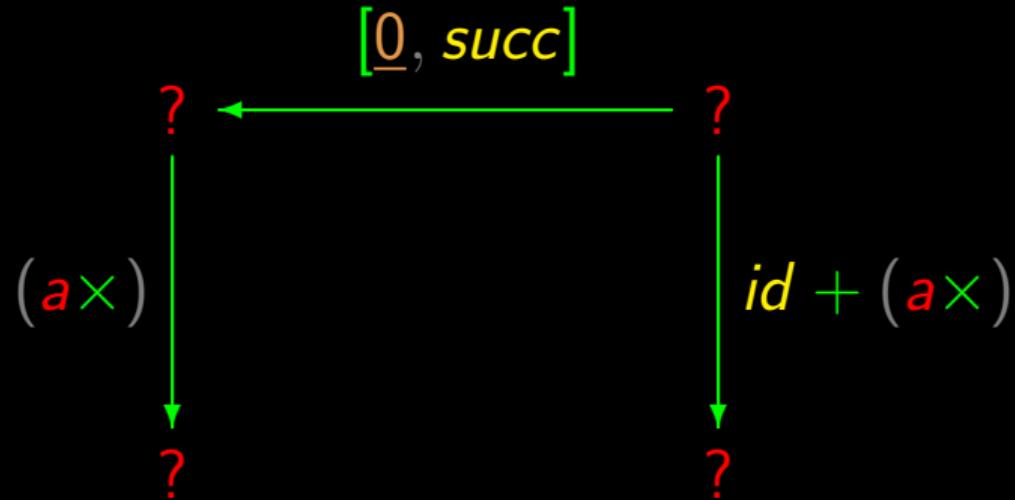
$$(\textcolor{red}{a} \times) \cdot [\underline{0}, \textcolor{blue}{succ}] = [\underline{0}, (\textcolor{red}{a} +)] \cdot (\textcolor{blue}{id} + (\textcolor{red}{a} \times))$$



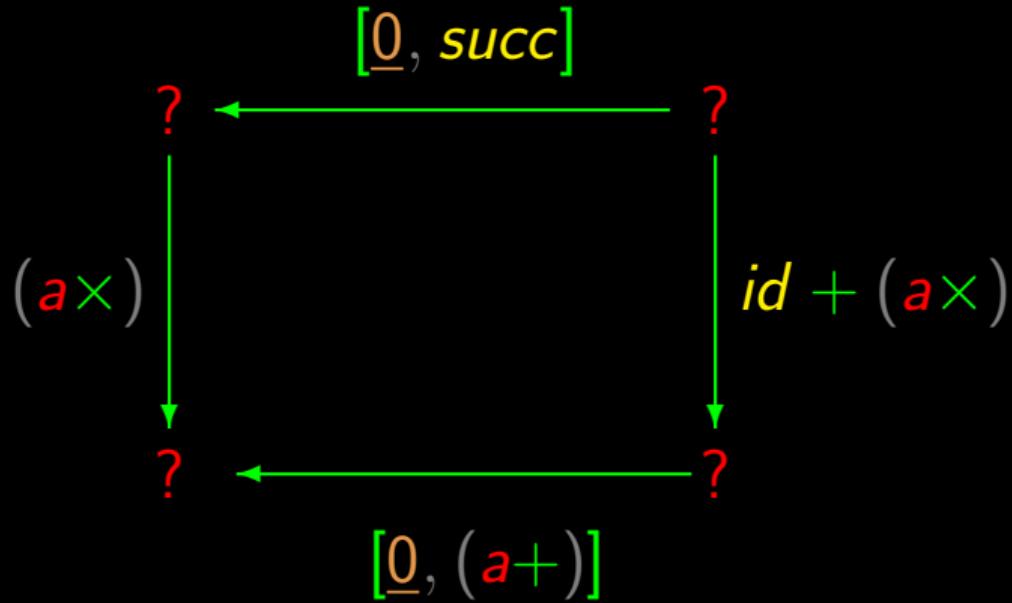
$$(\textcolor{red}{a}\times) \cdot [\underline{0}, \textcolor{blue}{succ}] = [\underline{0}, (\textcolor{red}{a}+)] \cdot (id + (\textcolor{red}{a}\times))$$



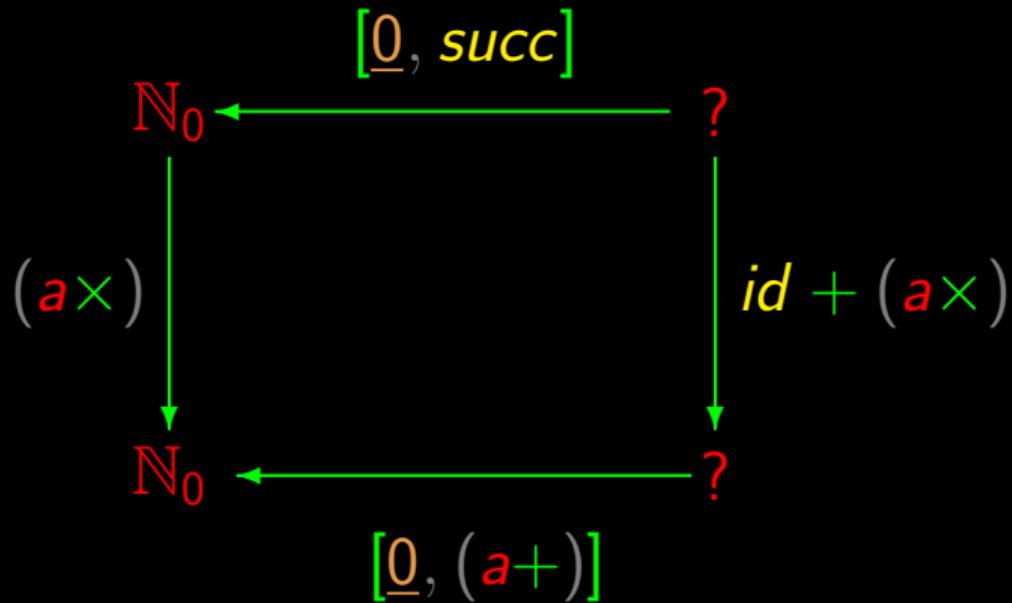
$$(\textcolor{red}{a} \times) \cdot [\underline{0}, \textcolor{blue}{succ}] = [\underline{0}, (\textcolor{red}{a} +)] \cdot (\textcolor{blue}{id} + (\textcolor{red}{a} \times))$$



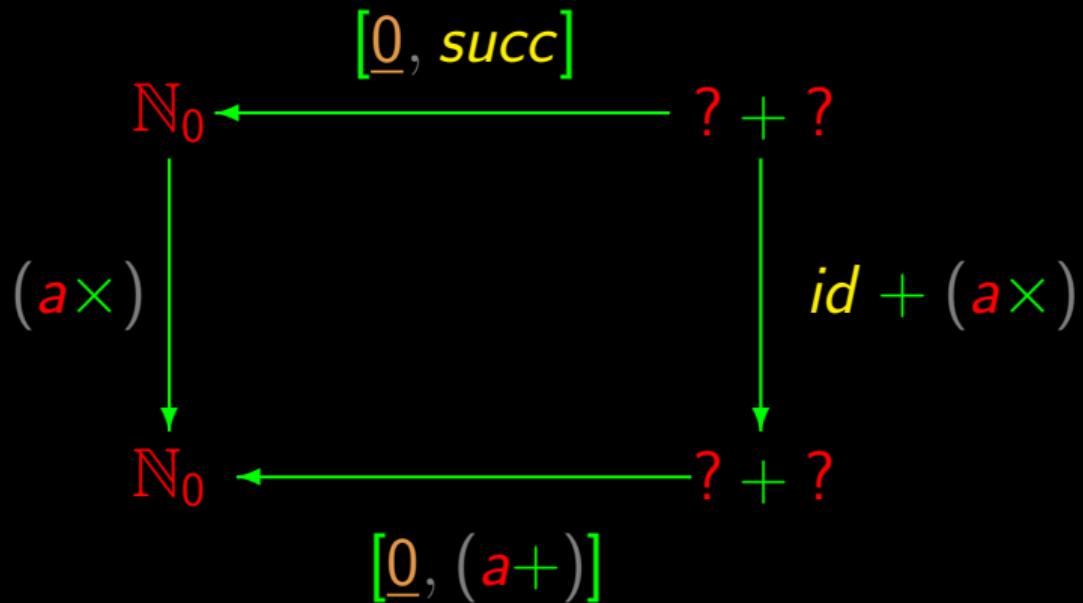
$$(\textcolor{red}{a} \times) \cdot [\underline{0}, \textcolor{blue}{succ}] = [\underline{0}, (\textcolor{red}{a} +)] \cdot (\textcolor{violet}{id} + (\textcolor{red}{a} \times))$$



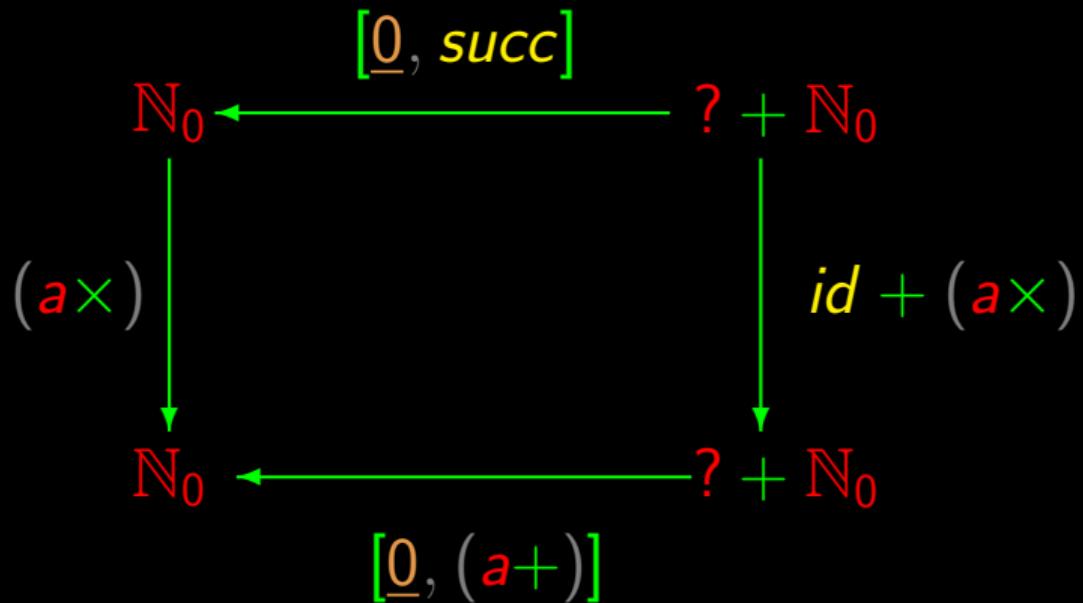
A program is a “rectangle”



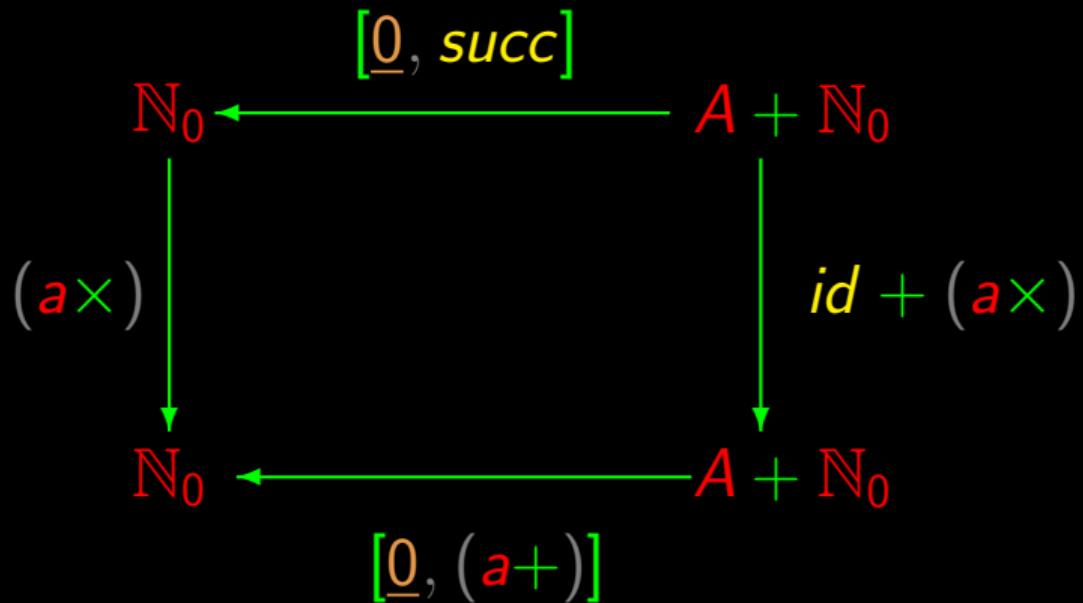
A program is a “rectangle”



A program is a “rectangle”



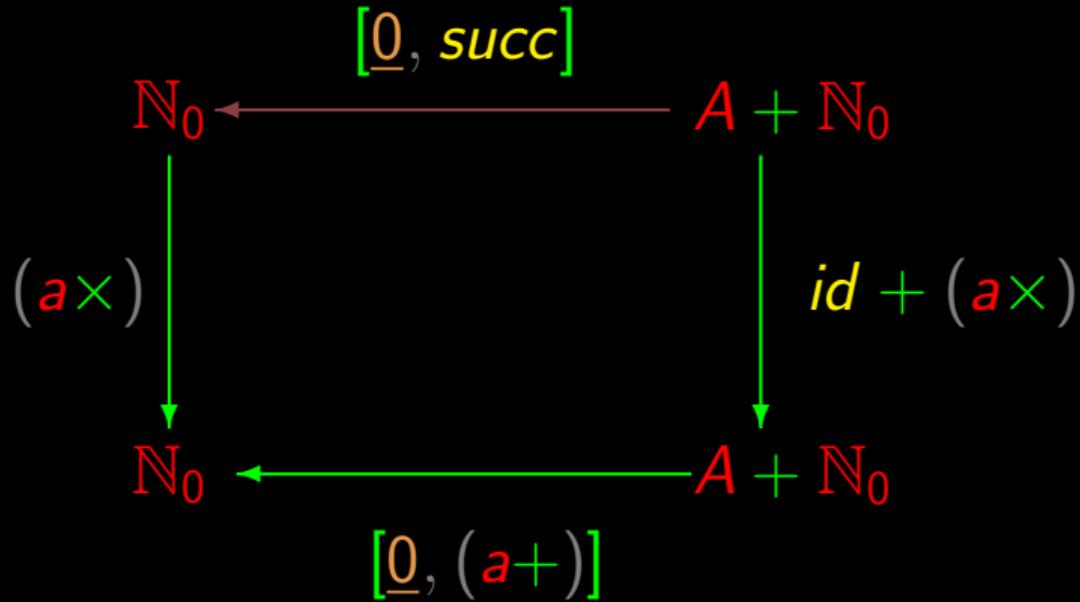
A program is a “rectangle”



Does $[\underline{0}, \text{succ}]^\circ$ exist?

$$\begin{array}{ccc} & [\underline{0}, \text{succ}]^\circ & \\ \mathbb{N}_0 & \xrightarrow{\hspace{3cm}} & A + \mathbb{N}_0 \\ \downarrow (\textcolor{red}{a}\times) & & \downarrow \text{id} + (\textcolor{red}{a}\times) \\ \mathbb{N}_0 & \xleftarrow{\hspace{3cm}} & A + \mathbb{N}_0 \\ & [\underline{0}, (\textcolor{red}{a}+)] & \end{array}$$

Does $[\underline{0}, \text{succ}]^\circ$ exist?



Does $[\underline{0}, \text{succ}]^\circ$ exist?

$$\begin{array}{ccc} & [\underline{0}, \text{succ}]^\circ & \\ \mathbb{N}_0 & \xrightarrow{\hspace{3cm}} & A + \mathbb{N}_0 \\ \downarrow (\textcolor{red}{a}\times) & & \downarrow \text{id} + (\textcolor{red}{a}\times) \\ \mathbb{N}_0 & \xleftarrow{\hspace{3cm}} & A + \mathbb{N}_0 \\ & [\underline{0}, (\textcolor{red}{a}+)] & \end{array}$$

Does $[\underline{0}, \text{succ}]^\circ$ exist?

$$\begin{array}{ccc} & [\underline{0}, \text{succ}]^\circ & \\ \mathbb{N}_0 & \xrightarrow{\hspace{3cm}} & A + \mathbb{N}_0 \\ \downarrow (\textcolor{red}{a}\times) & & \downarrow \text{id} + (\textcolor{red}{a}\times) \\ \mathbb{N}_0 & \xleftarrow{\hspace{3cm}} & A + \mathbb{N}_0 \\ & [\underline{0}, (\textcolor{red}{a}+)] & \end{array}$$

$$(\textcolor{red}{a}\times) = [\underline{0}, (\textcolor{red}{a}+)] \cdot (\text{id} + (\textcolor{red}{a}\times)) \cdot [\underline{0}, \text{succ}]^\circ(?)$$

Does $[\underline{0}, \text{succ}]^\circ$ exist?

Conjecture: $\mathbf{out} : \mathbb{N}_0 \rightarrow A + \mathbb{N}_0$ exists and is $[\underline{0}, \text{succ}]^\circ$

$$\mathbf{out} \cdot [\underline{0}, \text{succ}] = id$$

Does $[\underline{0}, \text{succ}]^\circ$ exist?

Conjecture: $\mathbf{out} : \mathbb{N}_0 \rightarrow A + \mathbb{N}_0$ exists and is $[\underline{0}, \text{succ}]^\circ$

$$\begin{aligned}\mathbf{out} \cdot [\underline{0}, \text{succ}] &= id \\ \Leftrightarrow \quad \quad \quad \{ \quad \text{+-fusion} \quad \} \\ [\mathbf{out} \cdot \underline{0}, \mathbf{out} \cdot \text{succ}] &= id\end{aligned}$$

Does $[\underline{0}, \text{succ}]^\circ$ exist?

Conjecture: $\mathbf{out} : \mathbb{N}_0 \rightarrow A + \mathbb{N}_0$ exists and is $[\underline{0}, \text{succ}]^\circ$

$$\mathbf{out} \cdot [\underline{0}, \text{succ}] = id$$

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

$$[\mathbf{out} \cdot \underline{0}, \mathbf{out} \cdot \text{succ}] = id$$

$$\Leftrightarrow \quad \left\{ \begin{array}{c} \text{universal-+} \end{array} \right\}$$

$$\left\{ \begin{array}{l} \mathbf{out} \cdot \underline{0} = i_1 \\ \mathbf{out} \cdot \text{succ} = i_2 \end{array} \right.$$

Does $[\underline{0}, \text{succ}]^\circ$ exist?

$$\left\{ \begin{array}{l} \mathbf{out} \cdot \underline{0} = i_1 \\ \mathbf{out} \cdot \text{succ} = i_2 \end{array} \right.$$

Does $[\underline{0}, \text{succ}]^\circ$ exist?

$$\left\{ \begin{array}{l} \textbf{out} \cdot \underline{0} = i_1 \\ \textbf{out} \cdot \text{succ} = i_2 \end{array} \right.$$

$\Leftrightarrow \quad \left\{ \begin{array}{l} \text{introduce variables} \end{array} \right\}$

$$\left\{ \begin{array}{l} \textbf{out} (\underline{0} \ x) = i_1 \ x \\ \textbf{out} (\text{succ } x) = i_2 \ x \end{array} \right.$$

Does $[\underline{0}, \text{succ}]^\circ$ exist?

$$\begin{cases} \mathbf{out} \cdot \underline{0} = i_1 \\ \mathbf{out} \cdot \text{succ} = i_2 \end{cases}$$

$\Leftrightarrow \quad \left\{ \begin{array}{l} \text{introduce variables} \end{array} \right\}$

$$\begin{cases} \mathbf{out} (\underline{0} \ x) = i_1 \ x \\ \mathbf{out} (\text{succ } x) = i_2 \ x \end{cases}$$

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

$$\begin{cases} \mathbf{out} 0 = i_1 \ x \\ \mathbf{out} (x + 1) = i_2 \ x \end{cases}$$

Does $[\underline{0}, \text{succ}]^\circ$ exist?

$$\begin{cases} \mathbf{out} \cdot \underline{0} = i_1 \\ \mathbf{out} \cdot \text{succ} = i_2 \end{cases}$$

$\Leftrightarrow \quad \left\{ \begin{array}{l} \text{introduce variables} \end{array} \right\}$

$$\begin{cases} \mathbf{out} (\underline{0} \ x) = i_1 \ x \\ \mathbf{out} (\text{succ } x) = i_2 \ x \end{cases}$$

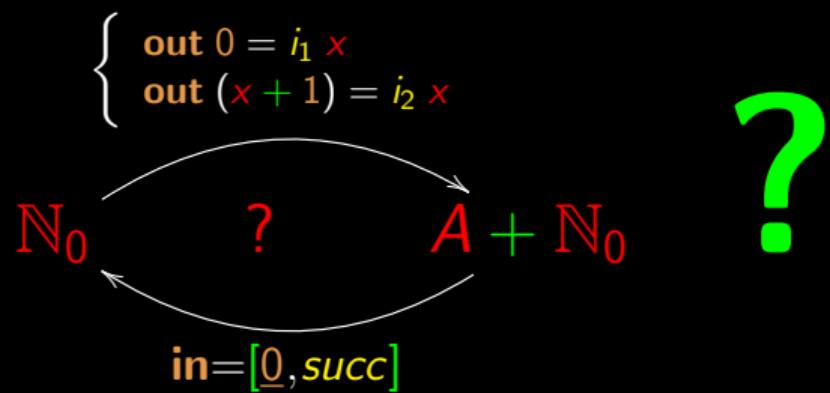
$$\begin{cases} \mathbf{out} 0 = i_1 \ x \\ \mathbf{out} (x + 1) = i_2 \ x \end{cases}$$

?

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

$$\begin{cases} \mathbf{out} 0 = i_1 \ x \\ \mathbf{out} (x + 1) = i_2 \ x \end{cases}$$

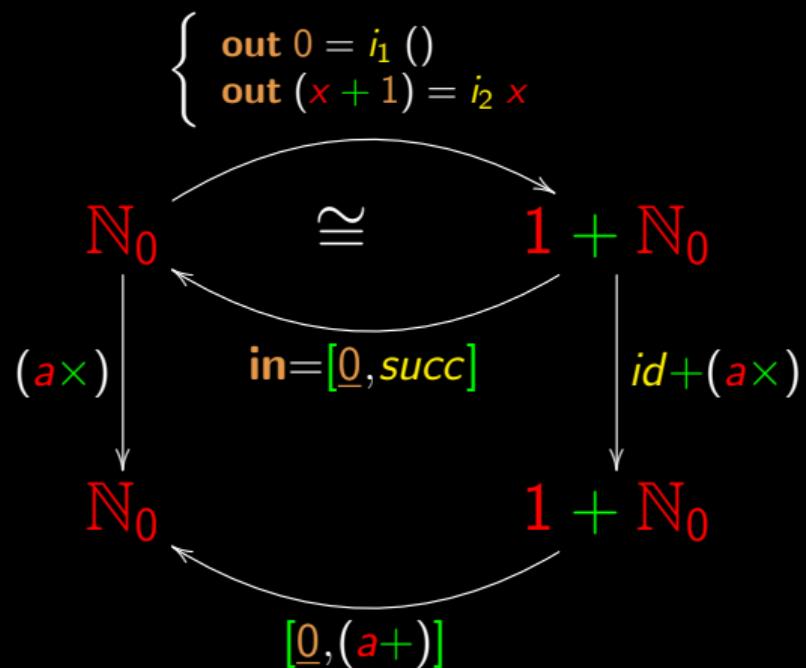
Problem



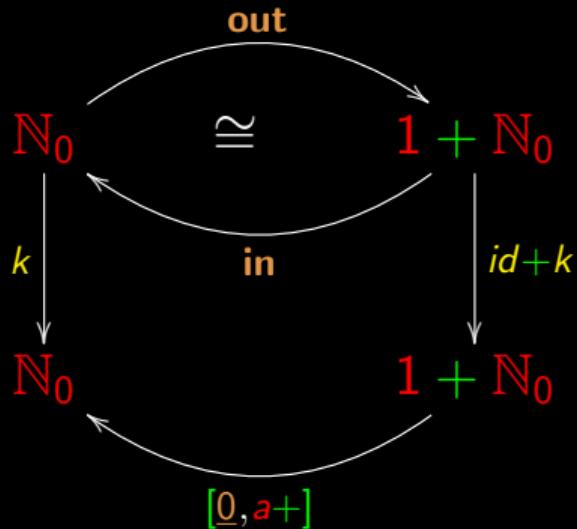
Solution

$$\left\{ \begin{array}{l} \text{out } 0 = i_1 () \\ \text{out } (x + 1) = i_2 x \end{array} \right. \quad \smiley$$
$$\mathbb{N}_0 \xrightarrow{\cong} 1 + \mathbb{N}_0$$
$$\text{in} = [0, \text{succ}]$$

Solution

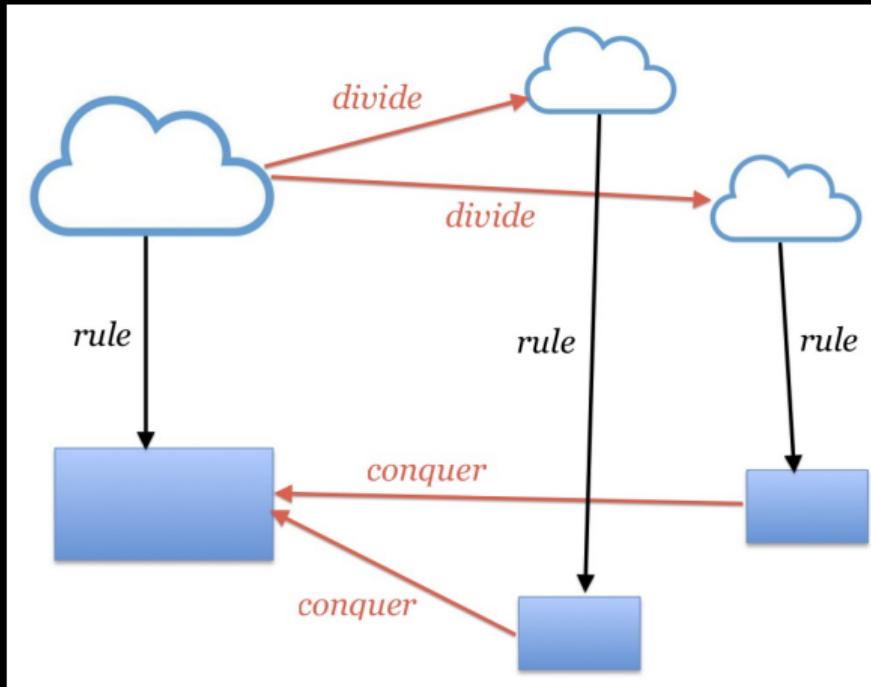


Diagrama

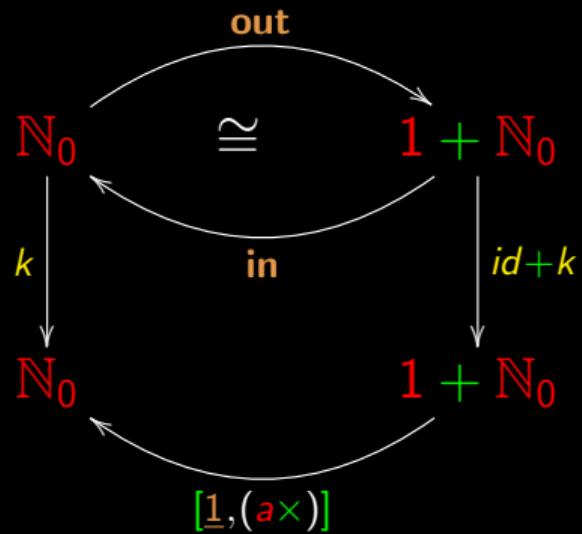


$$(a+) = k \text{ where } k = [0, (a+)] \cdot (id + k) \cdot \text{out}$$

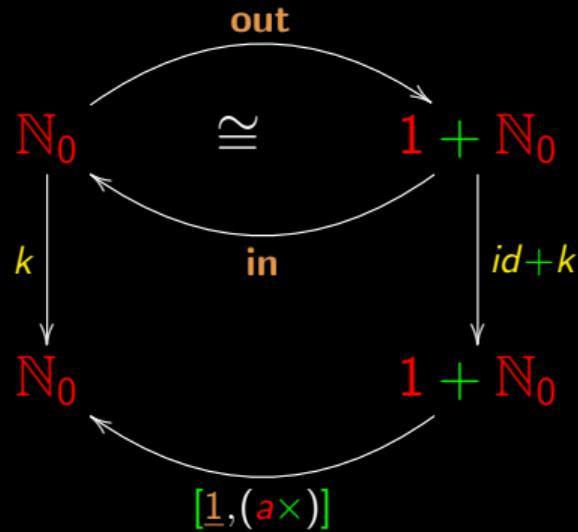
Anticipation of what is to come...



Variations in diagram



Variations in diagram

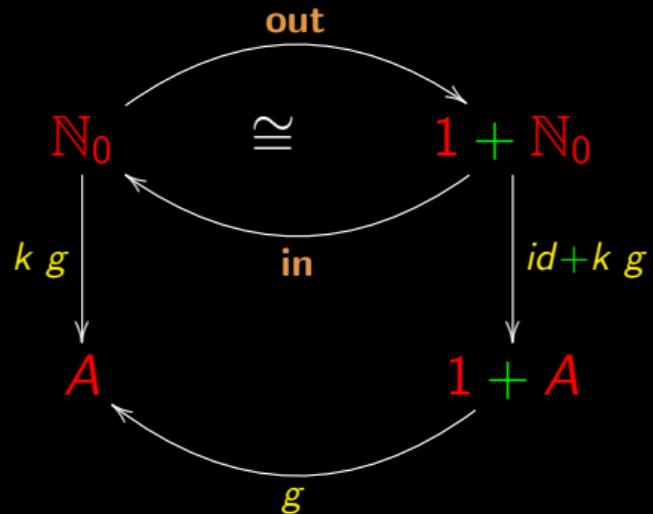


$$a^b = k \text{ where } k = [1, (a \times)] \cdot (id + k) \cdot \text{out}$$

Generalizing the diagram

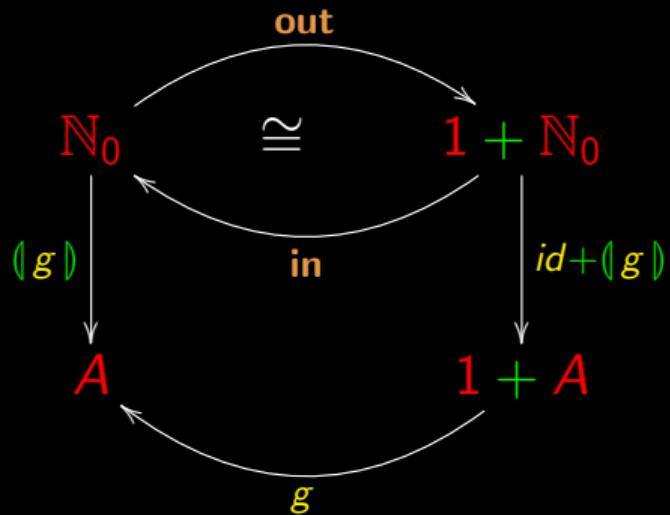
$$\begin{cases} a^0 = 1 \\ a^{b+1} = a \times a^b \end{cases}$$

Generalizing the diagram



$$k g = g \cdot (id + k g) \cdot \text{out}$$

Generalizing the diagram



$$\langle g \rangle = g \cdot (id + \langle g \rangle) \cdot \text{out}$$

Catamorphism

$\langle\!\rangle g \rangle$

$\underbrace{cata}_{downwards} (\kappa\alpha\tau\alpha)$

Catamorphism

$\langle\!\langle g \rangle\!\rangle$

$\underbrace{cata}_{downwards} (\kappa\alpha\tau\alpha) + \underbrace{morphism}_{\text{transformation}} (\mu\text{the}\rho\phi\iota\sigma\mu\circ\zeta)$

“Downwards transformation”

Definition

$$\begin{array}{ccc} \mathbb{N}_0 & \xrightarrow{\text{out}} & 1 + \mathbb{N}_0 \\ (\text{g}) \downarrow & & \downarrow id + (\text{g}) \\ A & \xleftarrow{g} & 1 + A \end{array}$$

$$(\text{g}) = g \cdot (id + (\text{g})) \cdot \text{out}$$

Property

$$\begin{array}{ccc} \mathbb{N}_0 & \xleftarrow{\text{in}} & 1 + \mathbb{N}_0 \\ \Downarrow \langle g \rangle & & \Downarrow id + \langle g \rangle \\ A & \xleftarrow{g} & 1 + A \end{array}$$

$$\langle g \rangle \cdot \text{in} = g \cdot (id + \langle g \rangle)$$

Property

$$\begin{array}{ccc} \mathbb{N}_0 & \xleftarrow{\text{in}} & 1 + \mathbb{N}_0 \\ \Downarrow \langle g \rangle & & \Downarrow id + \langle g \rangle \\ A & \xleftarrow{g} & 1 + A \end{array}$$

$$k = \langle g \rangle \Rightarrow k \cdot \text{in} = g \cdot (id + k)$$

Uniqueness

$$\begin{array}{ccc} \mathbb{N}_0 & \xleftarrow{\text{in}} & 1 + \mathbb{N}_0 \\ \Downarrow \langle g \rangle & & \Downarrow id + \langle g \rangle \\ A & \xleftarrow{g} & 1 + A \end{array}$$

$$k = \langle g \rangle \Leftarrow k \cdot \text{in} = g \cdot (id + k)$$

Universal property

$$\begin{array}{ccc} \mathbb{N}_0 & \xleftarrow{\text{in}} & 1 + \mathbb{N}_0 \\ k \downarrow & & \downarrow id+k \\ A & \xleftarrow{g} & 1 + A \end{array}$$

$$k = \langle g \rangle \Leftrightarrow k \cdot \text{in} = g \cdot (id + k)$$

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:

$$k = \langle\! g \!\rangle \Leftrightarrow k \cdot \mathbf{in} = g \cdot (id + k)$$

Cata-universal

$$\begin{array}{ccc} \mathbb{N}_0 & \xleftarrow{\text{in}} & 1 + \mathbb{N}_0 \\ k \downarrow & & \downarrow id+k \\ A & \xleftarrow{g} & 1 + A \end{array}$$

$$k = \langle g \rangle \Leftrightarrow k \cdot \text{in} = g \cdot (id + k)$$

Cata-cancellation

$$k = \langle\!g\!\rangle \Leftrightarrow k \cdot \mathbf{in} = g \cdot (id + k)$$

Cata-cancellation

$$\langle\!\langle g \rangle\!\rangle = \langle\!\langle g \rangle\!\rangle \Leftrightarrow \langle\!\langle g \rangle\!\rangle \cdot \mathbf{in} = g \cdot (id + \langle\!\langle g \rangle\!\rangle)$$

Cata-cancellation

$$\langle\!\langle g \rangle\!\rangle \cdot \mathbf{in} = g \cdot (id + \langle\!\langle g \rangle\!\rangle)$$

Cata-definition

$$(\textcolor{blue}{g}) = g \cdot (\textcolor{red}{id} + (\textcolor{blue}{g})) \cdot \textcolor{brown}{out}$$

Cata-definition

$$\begin{array}{ccc} \mathbb{N}_0 & \xrightarrow{\text{out}} & 1 + \mathbb{N}_0 \\ \Downarrow \langle\langle g \rangle\rangle & & \Downarrow id + \langle\langle g \rangle\rangle \\ A & \xleftarrow{g} & 1 + A \end{array}$$

$$\langle\langle g \rangle\rangle = g \cdot (id + \langle\langle g \rangle\rangle) \cdot \text{out}$$

Cata-universal

$$\begin{array}{ccc} \mathbb{N}_0 & \xleftarrow{\text{in}=[\underline{0}, \text{succ}]} & 1 + \mathbb{N}_0 \\ k \downarrow & & \downarrow id+k \\ A & \xleftarrow{g} & 1 + A \end{array}$$

$$k = \langle g \rangle \Leftrightarrow k \cdot \text{in} = g \cdot (id + k)$$

Cata-reflexion

$$k = \langle\! \langle g \rangle\! \rangle \Leftrightarrow k \cdot \mathbf{in} = g \cdot (id + k)$$

Cata-reflexion

$$id = (\!(g)\!) \Leftrightarrow id \cdot \mathbf{in} = g \cdot (id + id)$$

Cata-reflexion

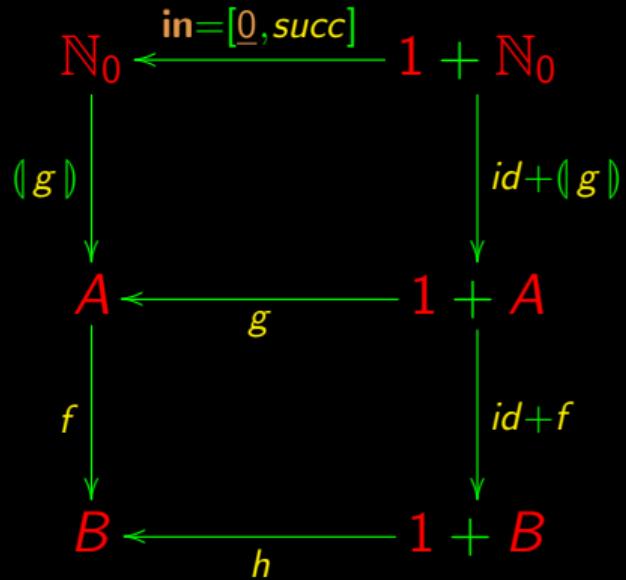
$$id = (\!(\, g \,) \!) \Leftrightarrow \mathbf{in} = g$$

Cata-reflexion

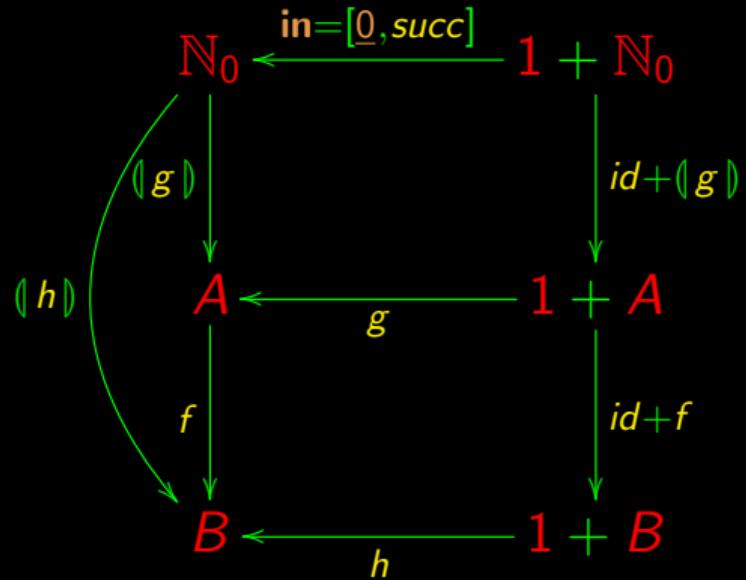
$$\begin{array}{ccc} \mathbb{N}_0 & \xleftarrow{\text{in}=[\underline{0}, \text{succ}]} & 1 + \mathbb{N}_0 \\ id = (\text{in}) \downarrow & & \downarrow id + (\text{in}) \\ \mathbb{N}_0 & \xleftarrow{\text{in}} & 1 + \mathbb{N}_0 \end{array}$$

$$(\text{in}) = id$$

Cata-fusion



Cata-fusion



Cata-fusion

$$f \cdot (\textcolor{red}{g}) = (\textcolor{blue}{h})$$

Cata-fusion

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

$\Leftrightarrow \{ \text{ Cata-universal for } k = f \cdot (\text{ } g \text{ }) \}$

$$f \cdot (\text{ } g \text{ }) \cdot \mathbf{in} = h \cdot (id + f \cdot (\text{ } g \text{ }))$$

Cata-fusion

$$f \cdot (\textcolor{blue}{g}) = (\textcolor{red}{h})$$

$$\Leftrightarrow \quad \left\{ \begin{array}{l} \text{Cata-universal for } k = f \cdot (\textcolor{blue}{g}) \end{array} \right\}$$

$$f \cdot (\textcolor{blue}{g}) \cdot \mathbf{in} = h \cdot (id + f \cdot (\textcolor{blue}{g}))$$

$$\Leftrightarrow \quad \left\{ \begin{array}{l} \text{Cata-cancellation} \end{array} \right\}$$

$$f \cdot g \cdot (id + (\textcolor{blue}{g})) = h \cdot (id + f \cdot (\textcolor{blue}{g}))$$

Cata-fusion

$$f \cdot g \cdot (id + \langle\!\langle g \rangle\!\rangle) = h \cdot (id \cdot id + f \cdot \langle\!\langle g \rangle\!\rangle)$$

Cata-fusion

$$\begin{aligned} f \cdot g \cdot (id + \langle\!\langle g \rangle\!\rangle) &= h \cdot (id \cdot id + f \cdot \langle\!\langle g \rangle\!\rangle) \\ \Leftrightarrow \quad \left\{ \begin{array}{l} \text{functor-+} \end{array} \right\} \\ f \cdot g \cdot (id + \langle\!\langle g \rangle\!\rangle) &= h \cdot (id + f) \cdot (id + \langle\!\langle g \rangle\!\rangle) \end{aligned}$$

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

$$f \cdot g \cdot (id + \langle\!\langle g \rangle\!\rangle) = h \cdot (id \cdot id + f \cdot \langle\!\langle g \rangle\!\rangle)$$

\Leftarrow $\{$ functor- $+$ $\}$

$$f \cdot g \cdot (id + \langle\!\langle g \rangle\!\rangle) = h \cdot (id + f) \cdot (id + \langle\!\langle g \rangle\!\rangle)$$

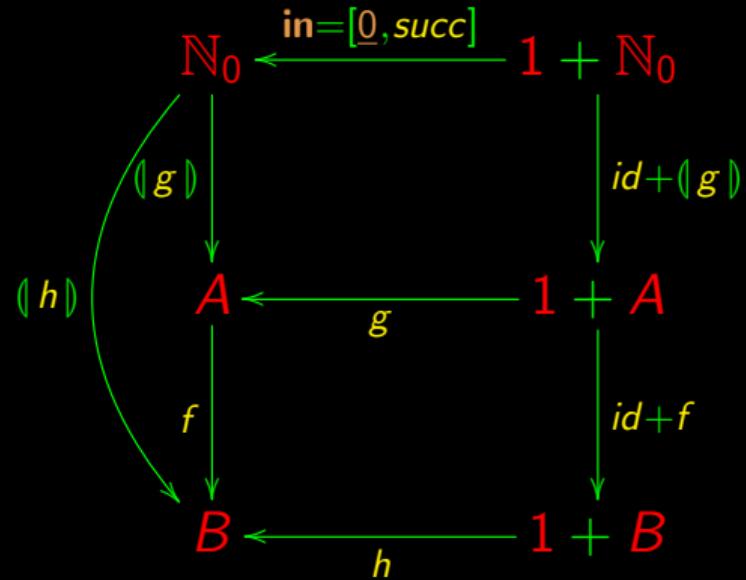
\Leftarrow $\{$ Leibniz $\}$

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

Summing up:

$$f \cdot \langle\!\langle g \rangle\!\rangle = \langle\!\langle h \rangle\!\rangle \Leftarrow f \cdot g = h \cdot (id + f)$$

Cata-fusion



What **is** a program?

What **is** a program?

Programs are “rectangles”



What **is** a program?

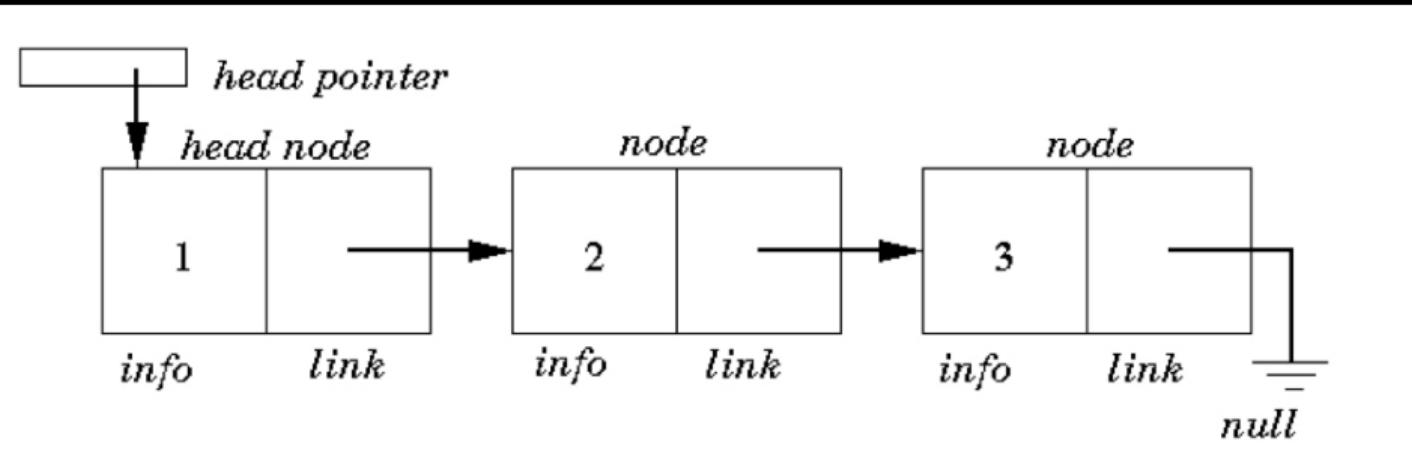
Programs are “rectangles” 

But there is a long way to the general case...

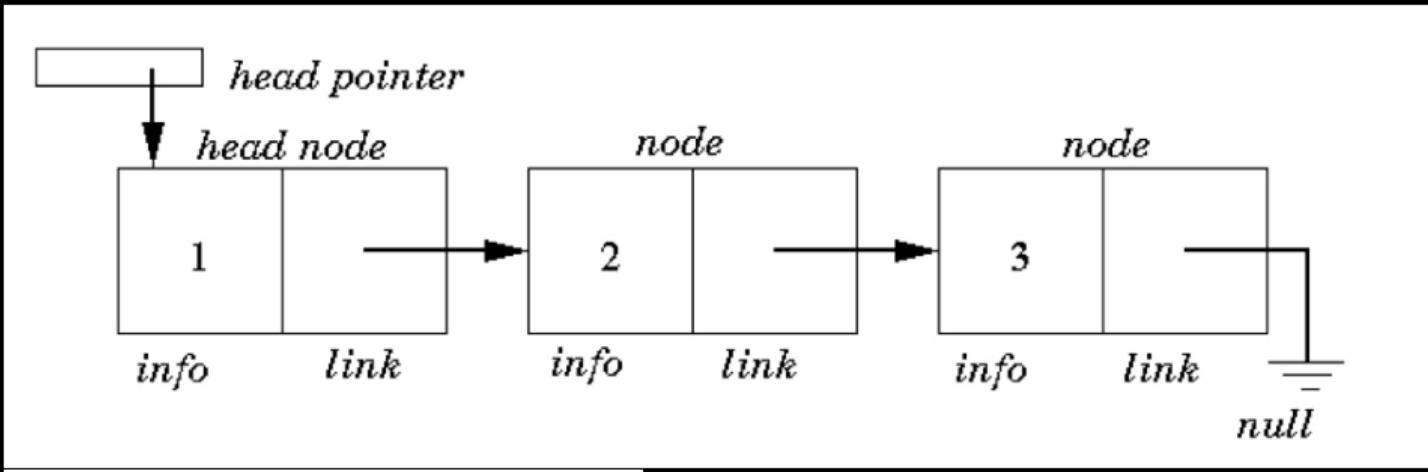
Cálculo de Programas

Class T07

Linked list



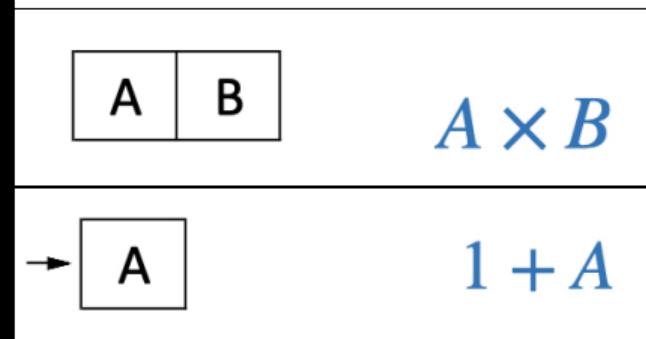
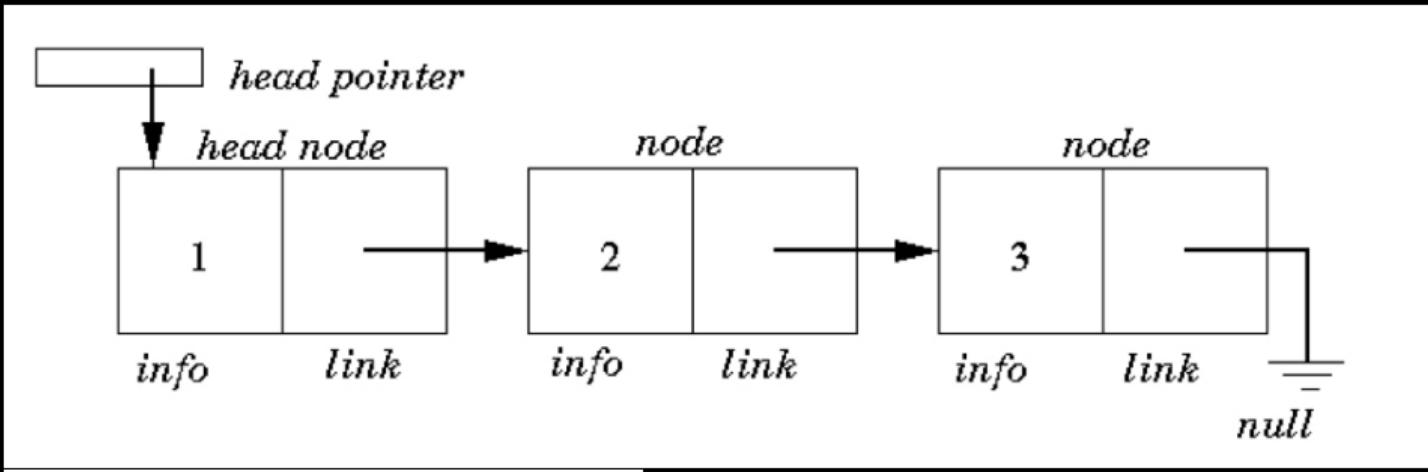
Linked list



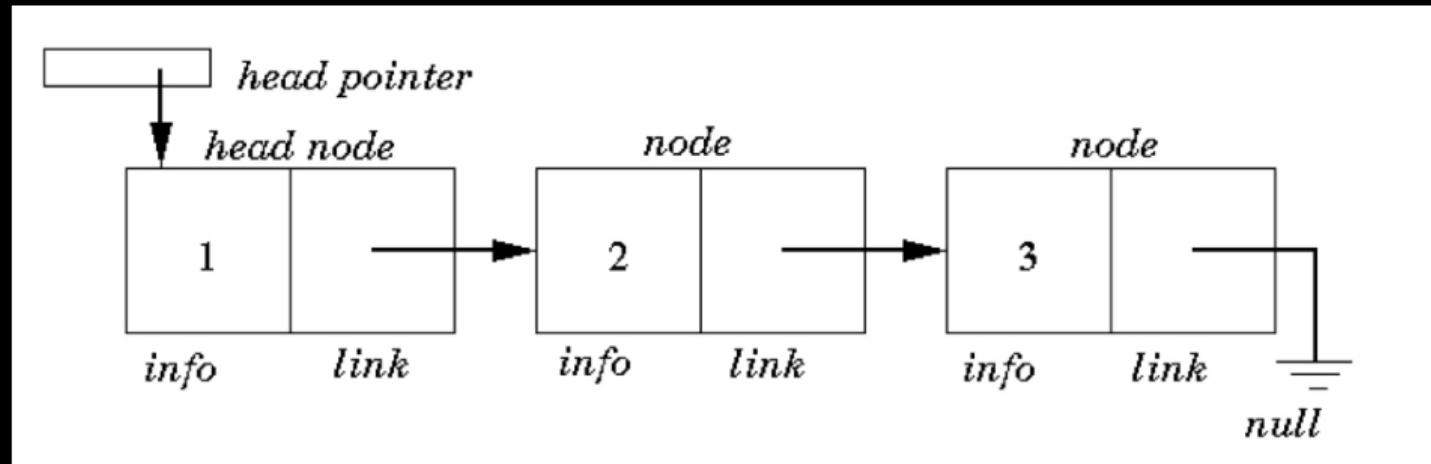
A	B
---	---

$$A \times B$$

Linked list



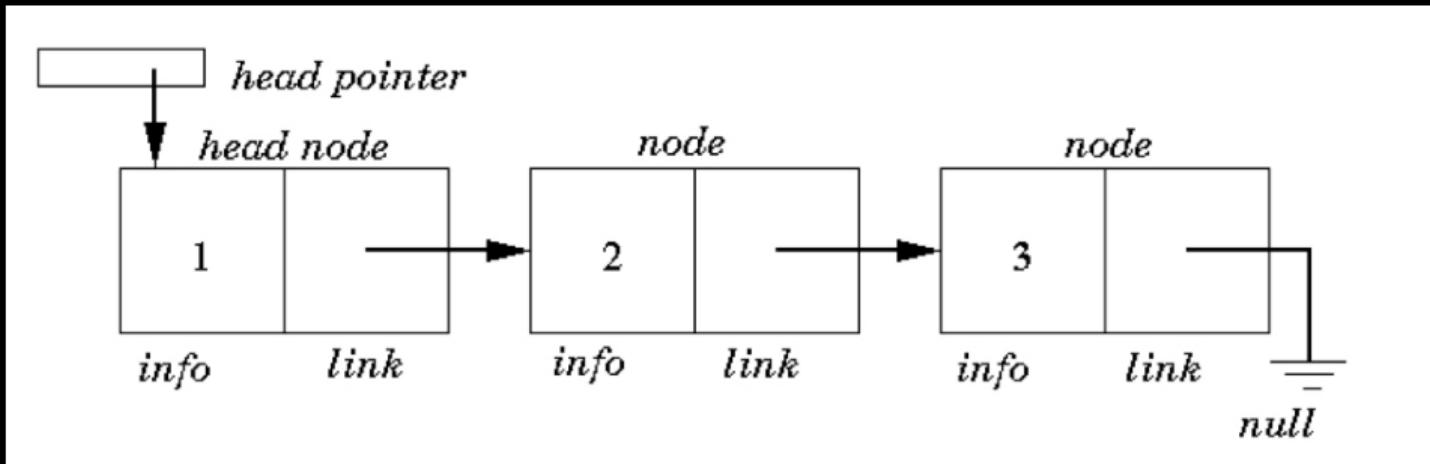
Linked list



“pointer”

$$L = 1 + N$$

Linked list



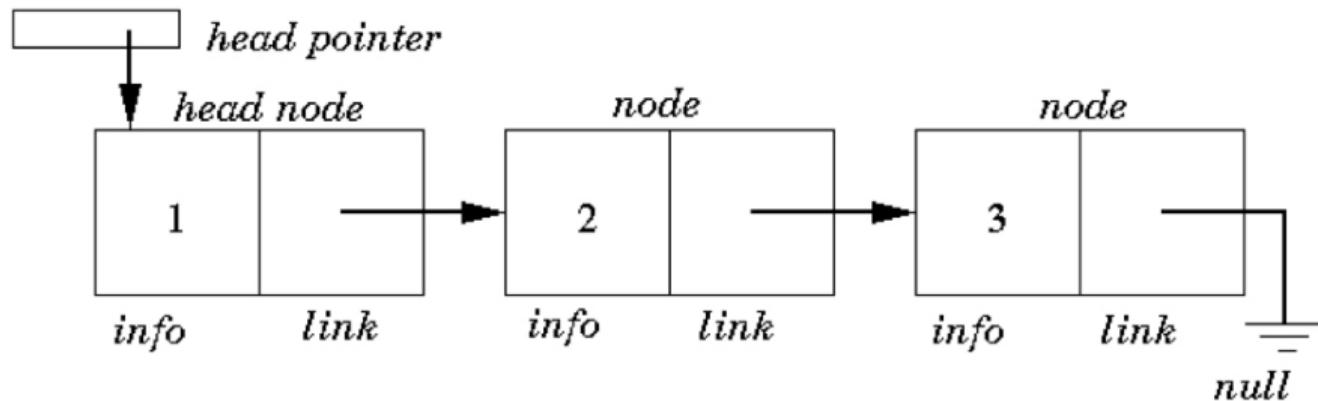
“pointer”

$$L = 1 + N$$

“node”

$$N = \mathbb{N}_0 \times L$$

Linked list

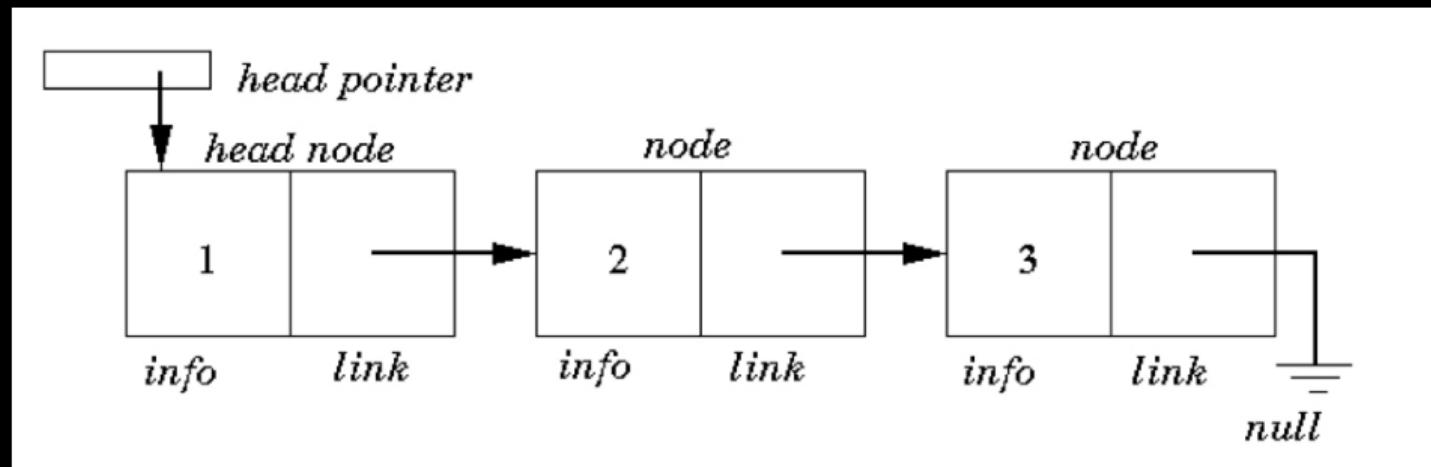


Substitution

$$\begin{cases} L = 1 + N \\ N = \mathbb{N}_0 \times L \end{cases}$$

$$L = 1 + \mathbb{N}_0 \times L$$

Linked list



In fact

$$L \cong 1 + \text{N}_0 \times L$$

Linked list

$$L \cong 1 + \mathbb{N}_0 \times L$$

Linked list

$$L \cong 1 + \mathbb{N}_0 \times L$$

$$\begin{array}{ccc} & \text{out} & \\ L & \approx & 1 + \mathbb{N}_0 \times L \\ & \text{in} & \end{array}$$

Linked list

$$L \cong 1 + \mathbb{N}_0 \times L$$

$$\begin{array}{ccc} L & \begin{matrix} \xrightarrow{\text{out}} \\ \cong \\ \xleftarrow{\text{in}} \end{matrix} & 1 + \mathbb{N}_0 \times L \end{array}$$

$$\mathbf{in} = [nil, cons]$$

Linked list

$$L \cong 1 + \mathbb{N}_0 \times L$$

$$\begin{array}{ccc} L & \xrightarrow{\cong} & 1 + \mathbb{N}_0 \times L \\ \text{in} \swarrow \curvearrowright \quad \curvearrowright \text{out} \end{array}$$

$$\text{in} = [\text{nil}, \text{cons}] \quad \left\{ \begin{array}{l} \text{nil} _ = [] \\ \text{cons } (a, l) = a : l \end{array} \right.$$

Linked list

$$L \cong 1 + \mathbb{N}_0 \times L$$

$$\begin{array}{ccc} L & \xrightarrow{\text{out}} & 1 + \mathbb{N}_0 \times L \\ & \xleftarrow{\text{in}} & \end{array}$$

$$\mathbf{in} = [nil, cons] \quad \begin{cases} nil _ = [] \\ cons (a, l) = a : l \end{cases}$$

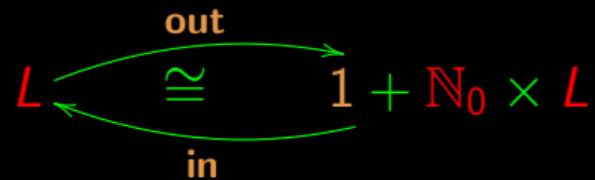
$$\mathbf{out} \cdot \mathbf{in} = id$$

Linked list

$$L \underset{\cong}{\sim} 1 + \mathbb{N}_0 \times L$$

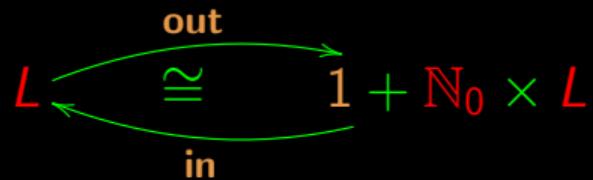
$$\begin{cases} \text{out} \cdot \text{in} = id \\ \text{in} = [\text{nil}, \text{cons}] \end{cases}$$

Linked list

$$L \underset{\cong}{\sim} 1 + \mathbb{N}_0 \times L$$


$$\begin{cases} \text{out} \cdot \text{in} = id \\ \text{in} = [\text{nil}, \text{cons}] \end{cases} \Rightarrow$$

Linked list

$$L \underset{\cong}{\sim} 1 + \mathbb{N}_0 \times L$$


$$\begin{cases} \text{out} \cdot \text{in} = id \\ \text{in} = [\text{nil}, \text{cons}] \end{cases} \Rightarrow \begin{cases} \text{out} [] = i_1 () \\ \text{out} (a : x) = i_2 (a, x) \end{cases}$$

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** $x \textcolor{red}{+} y$ such that

$$[] \textcolor{green}{+} x = x$$

$$[a] \textcolor{green}{+} x = a : x$$

Length of a list

Recall list **concatenation** $x \textcolor{red}{+} y$ such that

$$[] \textcolor{green}{+} x = x$$

$$[a] \textcolor{green}{+} x = a : x$$

Properties:

Length of a list

Recall list **concatenation** $x \textcolor{green}{+} y$ such that

$$[] \textcolor{green}{+} x = x$$

$$[a] \textcolor{green}{+} x = a : x$$

Properties:

$$\textcolor{blue}{length} [] = 0$$

Length of a list

Recall list **concatenation** $x \textcolor{green}{+} y$ such that

$$[] \textcolor{green}{+} x = x$$

$$[a] \textcolor{green}{+} x = a : x$$

Properties:

$$\textit{length} [] = 0$$

$$\textit{length} [a] = 1$$

Length of a list

Recall list **concatenation** $x \textcolor{green}{+} y$ such that

$$[] \textcolor{green}{+} x = x$$

$$[a] \textcolor{green}{+} x = a : x$$

Properties:

$$\textit{length} [] = 0$$

$$\textit{length} [a] = 1$$

$$\textit{length} (x \textcolor{green}{+} y) = \textit{length} x + \textit{length} y$$

Length of a list

$$\text{length} [] = 0$$

$$\text{length} [a] = 1$$

$$\text{length} ([a] ++ y) = \text{length} [a] + \text{length} y$$

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

$$\text{length} \cdot \text{nil} = \underline{0}$$

$$\text{length} \cdot \text{cons} = \text{succ} \cdot \text{length} \cdot \pi_2$$

$$\text{length} \cdot [\text{nil}, \text{cons}] = [\underline{0}, \text{succ} \cdot \pi_2] \cdot (\text{id} + \text{id} \times \text{length})$$

$$\begin{array}{ccc} L & \xrightarrow{\quad \cong \quad} & 1 + \mathbb{N}_0 \times L \\ \downarrow \text{length} & \nearrow \text{in} & \downarrow \text{id} + \text{id} \times \text{length} \\ \mathbb{N}_0 & & 1 + \mathbb{N}_0 \times \mathbb{N}_0 \\ & \searrow [\underline{0}, \text{succ} \cdot \pi_2] & \end{array}$$

$$\left\{ \begin{array}{l} \text{in} = [\text{nil}, \text{cons}] \\ \text{out} = \text{in}^\circ \end{array} \right.$$

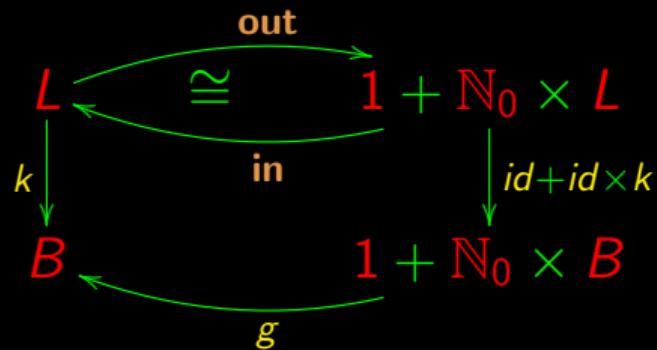
List sum

$$\begin{array}{ccc} L & \xrightarrow{\approx} & 1 + \mathbb{N}_0 \times L \\ \text{sum} \downarrow & \text{in} \curvearrowleft & \downarrow id + id \times sum \\ \mathbb{N}_0 & \xleftarrow{[0, add]} & 1 + \mathbb{N}_0 \times \mathbb{N}_0 \end{array}$$

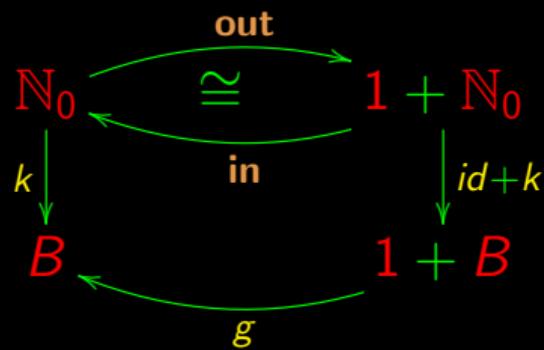
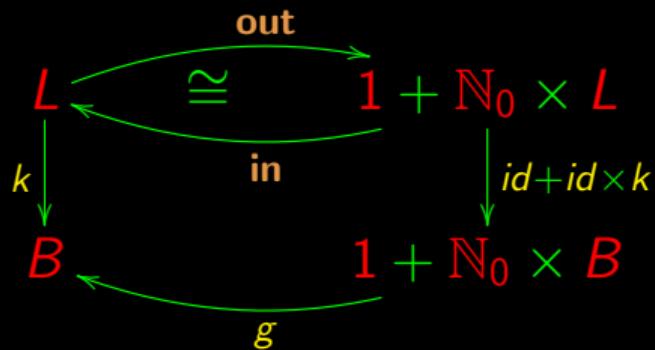
$$\left\{ \begin{array}{l} \text{in} = [nil, cons] \\ \text{out} = \text{in}^\circ \end{array} \right.$$

$$sum \cdot \text{in} = [0, add] \cdot (id + id \times sum)$$

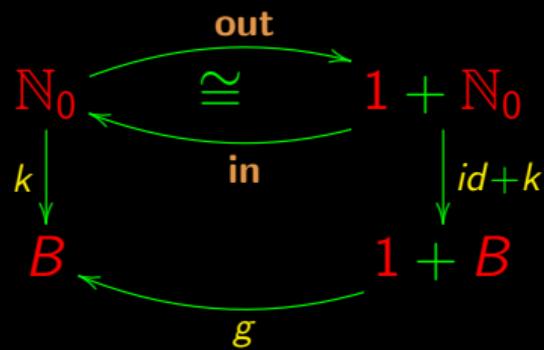
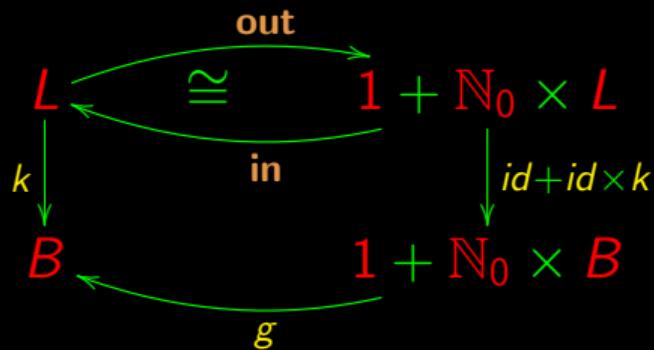
Generalizing



Generalizing

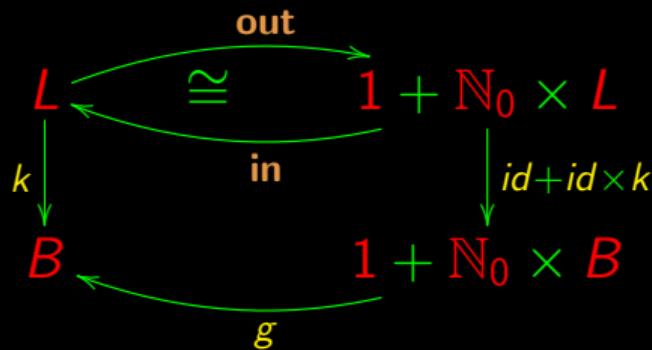


Generalizing

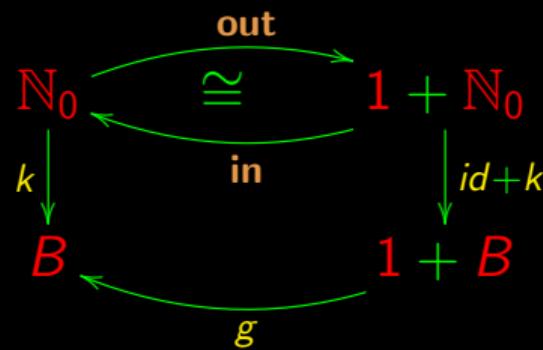


$$k \cdot \mathbf{in} = g \cdot \underbrace{id + id \times k}_{\mathbf{F} k}$$

Generalizing

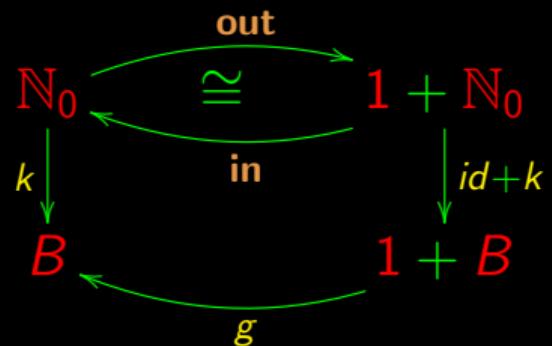


$$k \cdot \mathbf{in} = g \cdot \underbrace{id + id \times k}_{\mathbf{F} k}$$

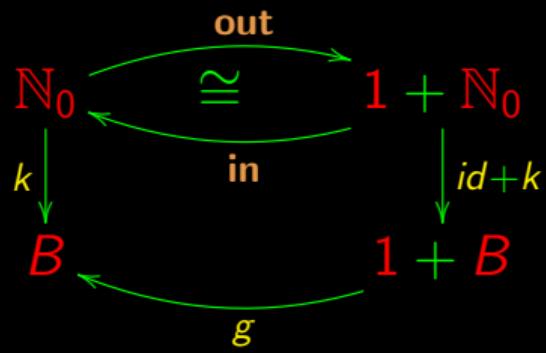


$$k \cdot \mathbf{in} = g \cdot \underbrace{id + k}_{\mathbf{F} k}$$

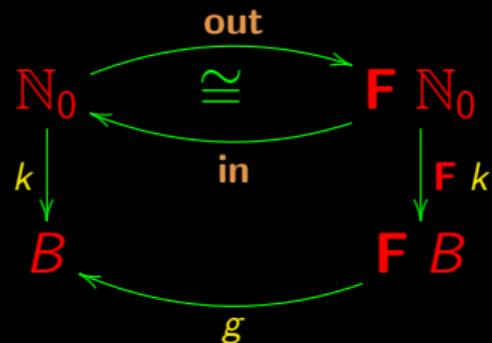
Abstraction (\mathbb{N}_0)



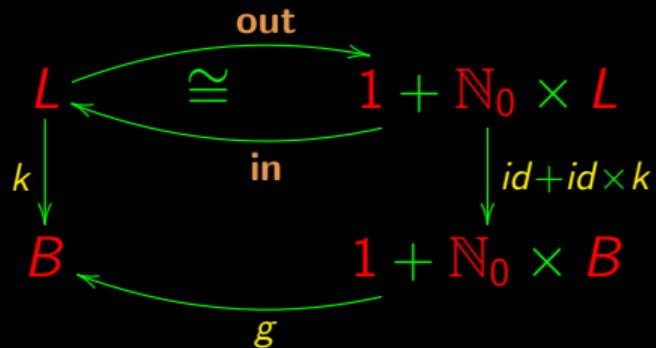
Abstraction (\mathbb{N}_0)



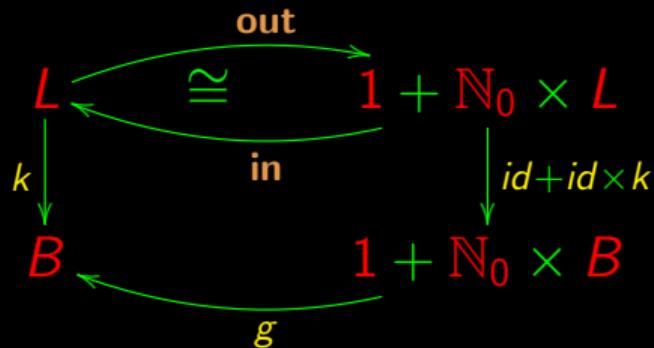
$$\begin{cases} \mathbf{F} k = id + k \\ \mathbf{F} X = 1 + X \end{cases}$$



Abstraction (lists)

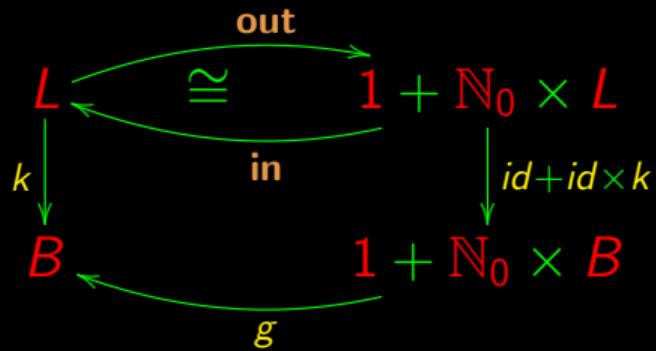


Abstraction (lists)

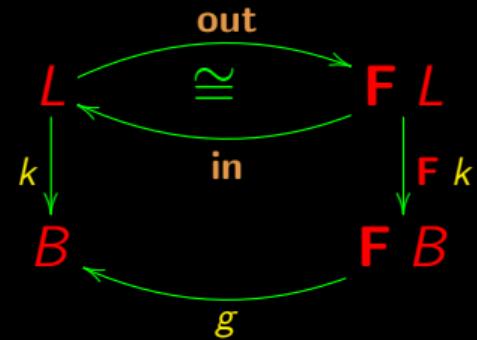


$$\begin{cases} \text{F } k = id + id \times k \\ \text{F } X = 1 + \mathbb{N}_0 \times X \end{cases}$$

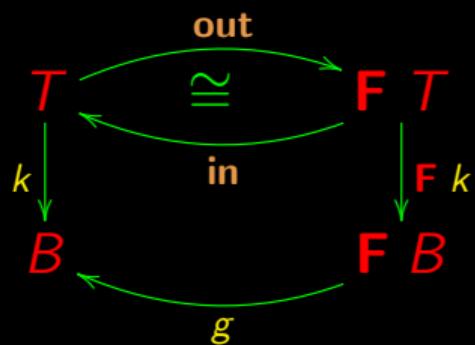
Abstraction (lists)



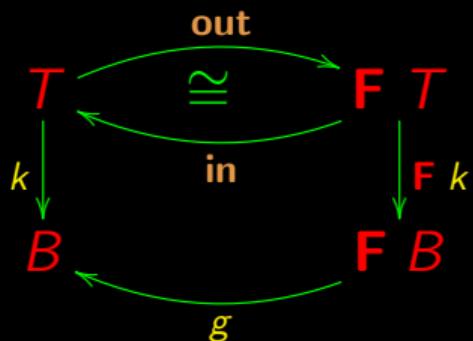
$$\begin{cases} \mathbf{F} k = id + id \times k \\ \mathbf{F} X = 1 + \mathbb{N}_0 \times X \end{cases}$$



Abstraction

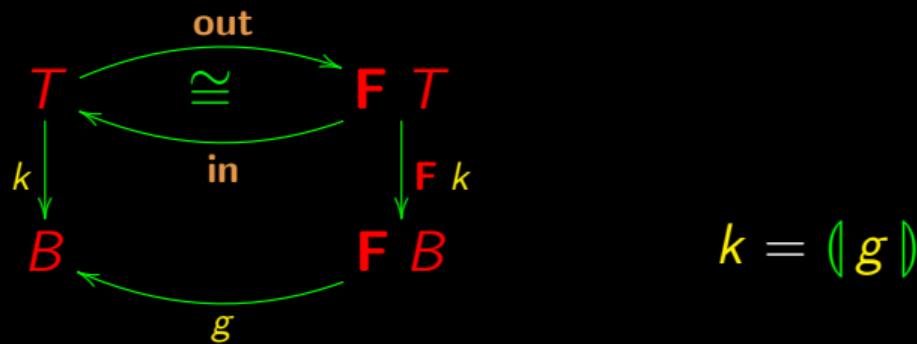


Abstraction



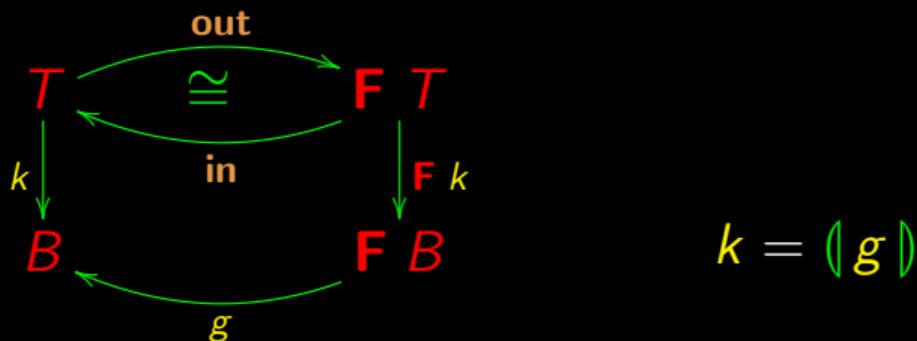
$$k \cdot \text{in} = g \cdot F k$$

Abstraction



$$k \cdot \mathbf{in} = g \cdot \mathbf{F} k$$

Abstraction

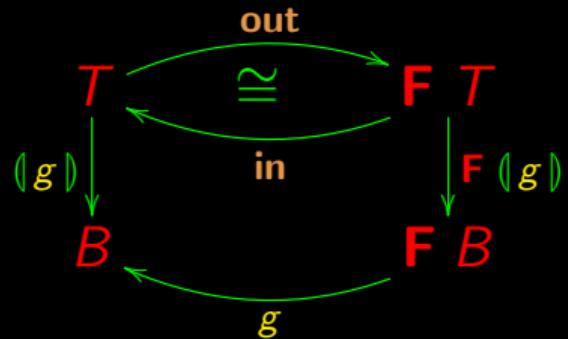


$$k \cdot \mathbf{in} = g \cdot \mathbf{F} k$$

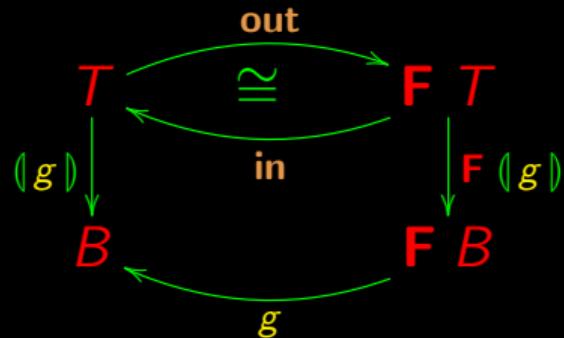
Read $k = \langle g \rangle$ as before:

k is the *catamorphism* of g .

Catamorphism



Catamorphism



Universal property:

$$k = (g) \Leftrightarrow k \cdot \text{in} = g \cdot \mathbf{F} k$$

Summary

Natural numbers: $\left\{ \begin{array}{l} T = \mathbb{N}_0 \\ \text{in} = [\underline{0}, \text{succ}] \\ \quad \left\{ \begin{array}{l} \mathbf{F} X = 1 + X \\ \mathbf{F} f = id + f \end{array} \right. \end{array} \right.$

Summary

Natural numbers: $\left\{ \begin{array}{l} T = \mathbb{N}_0 \\ \text{in} = [\underline{0}, \text{succ}] \\ \quad \left\{ \begin{array}{l} \mathbf{F} X = 1 + X \\ \mathbf{F} f = id + f \end{array} \right. \end{array} \right.$ for b $i = (\underline{i}, b)$

Summary

Natural numbers: $\left\{ \begin{array}{l} T = \mathbb{N}_0 \\ \mathbf{in} = [\underline{0}, \text{succ}] \\ \quad \left\{ \begin{array}{l} \mathbf{F} X = 1 + X \\ \mathbf{F} f = id + f \end{array} \right. \end{array} \right.$ for $b \ i = (\underline{i}, b)$

Lists: $\left\{ \begin{array}{l} T = L \\ \mathbf{in} = [nil, cons] \\ \quad \left\{ \begin{array}{l} \mathbf{F} X = 1 + \mathbb{N}_0 \times X \\ \mathbf{F} f = id + id \times f \end{array} \right. \end{array} \right.$

Summary

Natural numbers:
$$\left\{ \begin{array}{l} T = \mathbb{N}_0 \\ \mathbf{in} = [\underline{0}, \text{succ}] \\ \quad \left\{ \begin{array}{l} \mathbf{F} X = 1 + X \\ \mathbf{F} f = id + f \end{array} \right. \end{array} \right.$$
 for b $i = (\underline{i}, b)$

Lists:
$$\left\{ \begin{array}{l} T = L \\ \mathbf{in} = [nil, cons] \\ \quad \left\{ \begin{array}{l} \mathbf{F} X = 1 + \mathbb{N}_0 \times X \\ \mathbf{F} f = id + id \times f \end{array} \right. \end{array} \right.$$
 foldr f $i = (\underline{i}, \widehat{f})$

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

$$k = \langle\langle g \rangle\rangle \Leftrightarrow k \cdot \mathbf{in} = g \cdot \mathbf{F} k$$

make $k := \langle\langle g \rangle\rangle$:

Cata-cancellation

In

$$k = \langle\langle g \rangle\rangle \Leftrightarrow k \cdot \mathbf{in} = g \cdot \mathbf{F} k$$

make $k := \langle\langle g \rangle\rangle$:

$$\langle\langle g \rangle\rangle = \langle\langle g \rangle\rangle \Leftrightarrow \langle\langle g \rangle\rangle \cdot \mathbf{in} = g \cdot \mathbf{F} \langle\langle g \rangle\rangle$$

Cata-cancellation

In

$$k = \langle\langle g \rangle\rangle \Leftrightarrow k \cdot \mathbf{in} = g \cdot \mathbf{F} k$$

make $k := \langle\langle g \rangle\rangle$:

$$\begin{aligned}\langle\langle g \rangle\rangle &= \langle\langle g \rangle\rangle \Leftrightarrow \langle\langle g \rangle\rangle \cdot \mathbf{in} = g \cdot \mathbf{F} \langle\langle g \rangle\rangle \\ &\Leftrightarrow \{ \ x = x \Leftrightarrow \text{true} \ } \\ \text{true} &\Leftrightarrow \langle\langle g \rangle\rangle \cdot \mathbf{in} = g \cdot \mathbf{F} \langle\langle g \rangle\rangle\end{aligned}$$

Cata-cancellation

In

$$k = \langle\langle g \rangle\rangle \Leftrightarrow k \cdot \mathbf{in} = g \cdot \mathbf{F} k$$

make $k := \langle\langle g \rangle\rangle$:

$$\langle\langle g \rangle\rangle = \langle\langle g \rangle\rangle \Leftrightarrow \langle\langle g \rangle\rangle \cdot \mathbf{in} = g \cdot \mathbf{F} \langle\langle g \rangle\rangle$$

$$\Leftrightarrow \{ x = x \Leftrightarrow \text{true} \}$$

$$\text{true} \Leftrightarrow \langle\langle g \rangle\rangle \cdot \mathbf{in} = g \cdot \mathbf{F} \langle\langle g \rangle\rangle$$

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

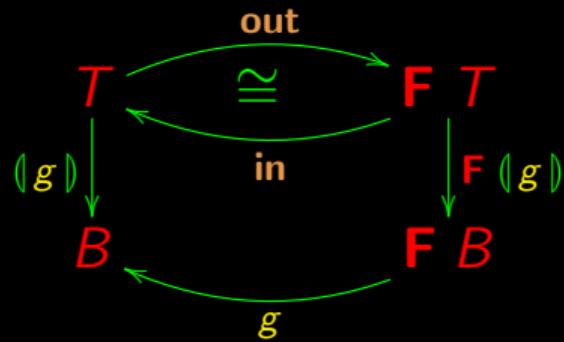
$$\langle\langle g \rangle\rangle \cdot \mathbf{in} = g \cdot \mathbf{F} \langle\langle g \rangle\rangle$$

Cata-cancellation

$$\begin{aligned} \langle\langle g \rangle\rangle \cdot \mathbf{in} &= g \cdot \mathbf{F} \langle\langle g \rangle\rangle \\ \Leftrightarrow \quad \left\{ \begin{array}{l} \text{isomorphism } \mathbf{in} / \mathbf{out} \end{array} \right\} \\ \langle\langle g \rangle\rangle &= g \cdot \mathbf{F} \langle\langle g \rangle\rangle \cdot \mathbf{out} \end{aligned}$$

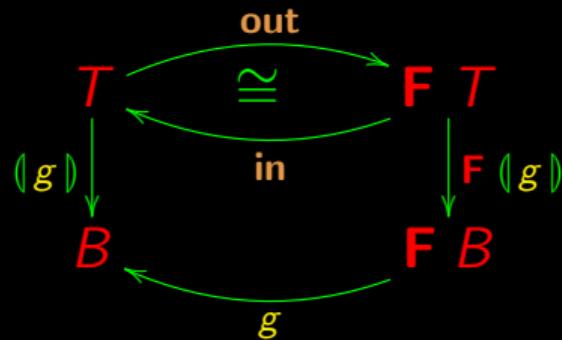
Cata-cancellation

$$\begin{aligned} \langle\langle g \rangle\rangle \cdot \mathbf{in} &= g \cdot \mathbf{F} \langle\langle g \rangle\rangle \\ \Leftrightarrow \quad \{ \text{ isomorphism } \mathbf{in} / \mathbf{out} \} \\ \langle\langle g \rangle\rangle &= g \cdot \mathbf{F} \langle\langle g \rangle\rangle \cdot \mathbf{out} \end{aligned}$$



Cata-cancellation

$$\begin{aligned} \langle\langle g \rangle\rangle \cdot \mathbf{in} &= g \cdot \mathbf{F} \langle\langle g \rangle\rangle \\ \Leftrightarrow \quad \{ \text{ isomorphism } \mathbf{in} / \mathbf{out} \} \\ \langle\langle g \rangle\rangle &= g \cdot \mathbf{F} \langle\langle g \rangle\rangle \cdot \mathbf{out} \end{aligned}$$



$$\langle\langle g \rangle\rangle = g \cdot \mathbf{F} \langle\langle g \rangle\rangle \cdot \mathbf{out}$$

executable definition

Functors

What does **F** mean, exactly?

$$\begin{array}{ccc} A & \xrightarrow{\quad} & \mathbf{F} A \\ f \downarrow & & \downarrow \mathbf{F} f \\ B & \xrightarrow{\quad} & \mathbf{F} B \end{array}$$

Functors

What does **F** mean, exactly?

$$\begin{array}{ccc} A & \cdots\cdots & \mathbf{F} A \\ f \downarrow & & \downarrow \mathbf{F} f \\ B & \cdots\cdots & \mathbf{F} B \end{array}$$

Recall **natural** (free)
properties

Functors

What does **F** mean, exactly?

For instance:

$$\begin{array}{ccc} A & \xrightarrow{\quad} & \mathbf{F} A \\ f \downarrow & & \downarrow \mathbf{F} f \\ B & \xrightarrow{\quad} & \mathbf{F} B \end{array}$$

$$\mathbf{F} X = 1 + \mathbb{N}_0 \times X$$

Recall **natural** (free)
properties

Functors

What does **F** mean, exactly?

For instance:

$$\begin{array}{ccc} A & \xrightarrow{\quad} & \mathbf{F} A \\ f \downarrow & & \downarrow \mathbf{F} f \\ B & \xrightarrow{\quad} & \mathbf{F} B \end{array}$$

$$\mathbf{F} X = 1 + \mathbb{N}_0 \times X$$

Substitute uppercase (**X**) by
lowercase *f* ($X \rightarrow f$):

Recall **natural** (free)
properties

Functors

What does **F** mean, exactly?

$$\begin{array}{ccc} A & \xrightarrow{\quad} & \mathbf{F} A \\ f \downarrow & & \downarrow \mathbf{F} f \\ B & \xrightarrow{\quad} & \mathbf{F} B \end{array}$$

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

$$\textcolor{red}{F} \ id = id$$

$$\textcolor{red}{F} (g \cdot f) = (\textcolor{red}{F} g) \cdot (\textcolor{blue}{F} \textcolor{yellow}{f})$$

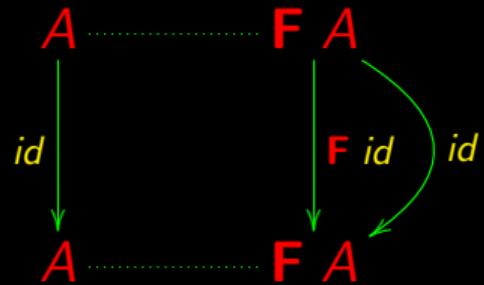
$\mathbf{F} \ id = id$

$$\begin{array}{ccc} A & \xrightarrow{\quad} & \mathbf{F} A \\ f \downarrow & & \downarrow \mathbf{F} f \\ B & \xrightarrow{\quad} & \mathbf{F} B \end{array}$$

$$\mathbf{F} \ id = id$$

$$\begin{array}{ccc} A & \xrightarrow{\hspace{2cm}} & \mathbf{F} A \\ id \downarrow & & \downarrow \mathbf{F} id \\ A & \xrightarrow{\hspace{2cm}} & \mathbf{F} A \end{array}$$

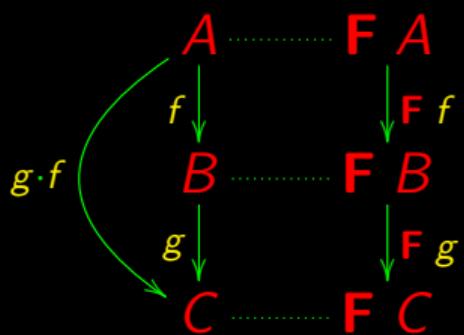
$\mathbf{F} \ id = id$



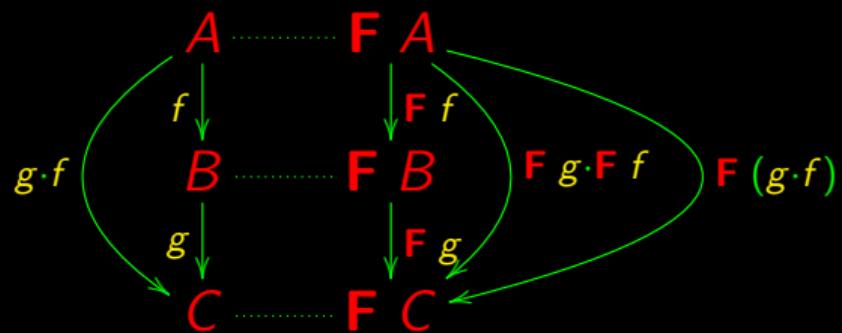
$$\mathbf{F}(g \cdot f) = (\mathbf{F} g) \cdot (\mathbf{F} f)$$

$$\begin{array}{ccc} A & \xrightarrow{\hspace{1cm}} & \mathbf{F} A \\ f \downarrow & & \downarrow \mathbf{F} f \\ B & \xrightarrow{\hspace{1cm}} & \mathbf{F} B \\ g \downarrow & & \downarrow \mathbf{F} g \\ C & \xrightarrow{\hspace{1cm}} & \mathbf{F} C \end{array}$$

$$\mathbf{F} (g \cdot f) = (\mathbf{F} g) \cdot (\mathbf{F} f)$$



$$\mathbf{F} (g \cdot f) = (\mathbf{F} g) \cdot (\mathbf{F} f)$$



Well known example

List functor:

$$\mathbf{F} f = f^*$$

Well known example

List functor:

$$\mathbf{F} f = f^* \text{ where } f^* x = [f a \mid a \leftarrow x] = \text{map } f x$$

$$\begin{array}{ccc} A & \overset{\text{.....}}{\longrightarrow} & A^* \\ f \downarrow & & \downarrow f^* \\ B & \overset{\text{.....}}{\longrightarrow} & B^* \end{array}$$

In fact

$$id^* x = [a \mid a \leftarrow x] = x$$

In fact

$$id^{\star} \ x = [a \mid a \leftarrow x] = x$$

$$(g^{\star} \cdot f^{\star}) \ x$$

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

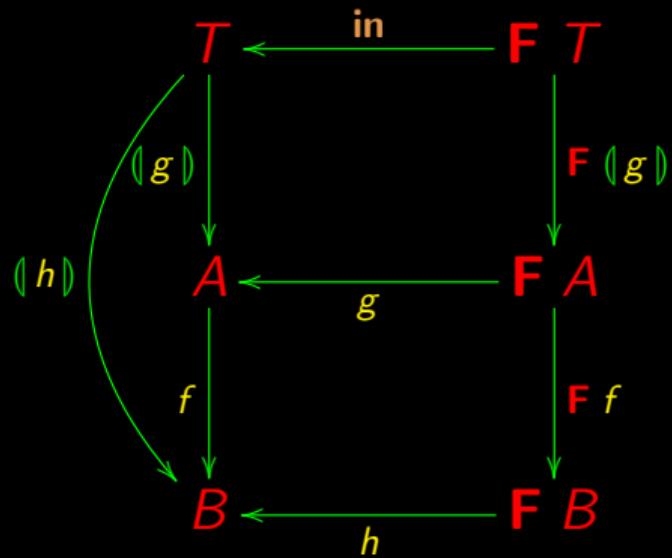
$$\mathbf{F} \ id = id$$

$$\mathbf{F} (g \cdot f) = (\mathbf{F} g) \cdot (\mathbf{F} f)$$

These properties are essential for reasoning about **catamorphisms**.

See example next.

Cata-fusion



Cata-fusion

$$f \cdot (\textcolor{red}{g}) = (\textcolor{blue}{h})$$

Cata-fusion

$$\begin{aligned} f \cdot (\textcolor{blue}{g}) &= (\textcolor{blue}{h}) \\ \Leftrightarrow \quad &\left\{ \begin{array}{l} \text{Cata-universal for } k = f \cdot (\textcolor{blue}{g}) \\ \end{array} \right\} \\ f \cdot (\textcolor{blue}{g}) \cdot \mathbf{in} &= h \cdot \mathbf{F}(f \cdot (\textcolor{blue}{g})) \end{aligned}$$

Cata-fusion

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

$$\Leftrightarrow \quad \left\{ \begin{array}{l} \text{Cata-universal for } k = f \cdot (\text{ } g \text{ }) \end{array} \right\}$$

$$f \cdot (\text{ } g \text{ }) \cdot \text{in} = h \cdot \text{F} (f \cdot (\text{ } g \text{ }))$$

$$\Leftrightarrow \quad \left\{ \begin{array}{l} \text{Cata-cancellation} \end{array} \right\}$$

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

Cata-fusion

$$f \cdot g \cdot \mathbf{F}(\!(g)\!) = h \cdot \mathbf{F}(f \cdot (\!(g)\!))$$

Cata-fusion

$$f \cdot g \cdot \mathbf{F}(\!(g)\!) = h \cdot \mathbf{F}(f \cdot (\!(g)\!))$$

$\Leftrightarrow \quad \left\{ \begin{array}{l} \text{functor-}\mathbf{F}(1) \end{array} \right\}$

$$f \cdot g \cdot \mathbf{F}(\!(g)\!) = h \cdot \mathbf{F} f \cdot \mathbf{F}(\!(g)\!)$$

Cata-fusion

$$f \cdot g \cdot \mathbf{F}(\!(g)\!) = h \cdot \mathbf{F}(f \cdot (\!(g)\!))$$

$$\Leftrightarrow \quad \left\{ \begin{array}{l} \text{functor-}\mathbf{F}(1) \end{array} \right\}$$

$$f \cdot g \cdot \mathbf{F}(\!(g)\!) = h \cdot \mathbf{F}f \cdot \mathbf{F}(\!(g)\!)$$

$$\Leftarrow \quad \left\{ \begin{array}{l} \text{Leibniz} \end{array} \right\}$$

$$f \cdot g = h \cdot (\mathbf{F}f)$$

Cata-fusion

$$f \cdot g \cdot \mathbf{F}(\!(g)\!) = h \cdot \mathbf{F}(f \cdot (\!(g)\!))$$

$$\Leftrightarrow \quad \left\{ \begin{array}{l} \text{functor-}\mathbf{F}(1) \end{array} \right\}$$

$$f \cdot g \cdot \mathbf{F}(\!(g)\!) = h \cdot \mathbf{F}f \cdot \mathbf{F}(\!(g)\!)$$

$$\Leftarrow \quad \left\{ \begin{array}{l} \text{Leibniz} \end{array} \right\}$$

$$f \cdot g = h \cdot (\mathbf{F}f)$$

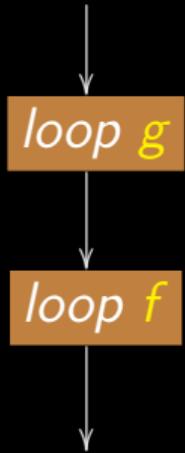
Summing up:

$$f \cdot (\!(g)\!) = (\!h\!) \quad \Leftarrow \quad f \cdot g = h \cdot (\mathbf{F}f)$$

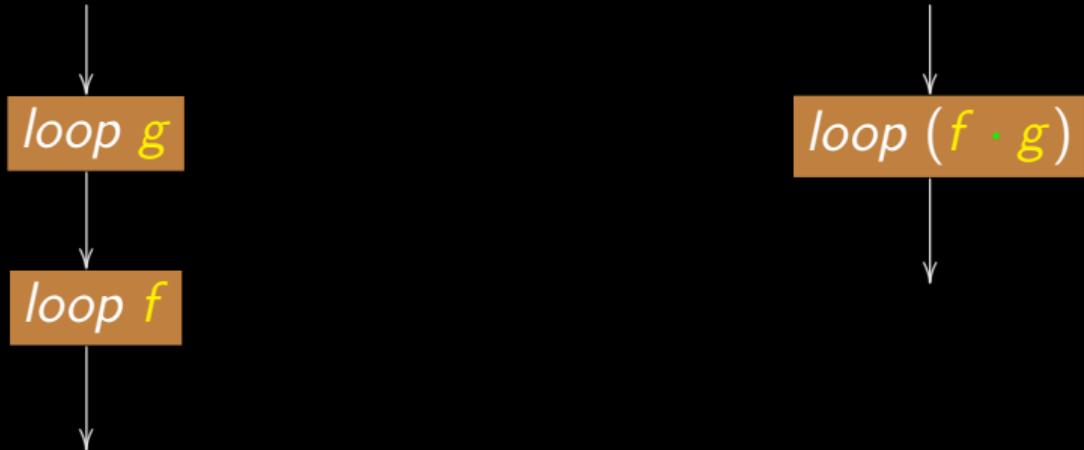
Cálculo de Programas

Class T08

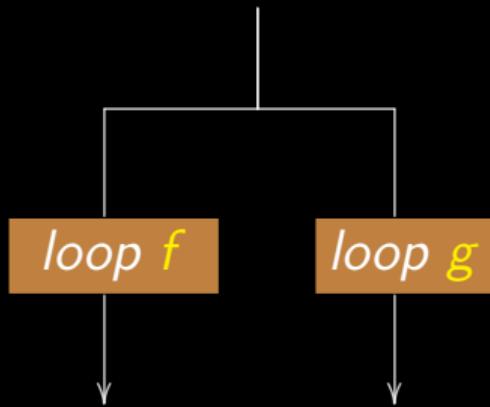
Fusion in programming — "sequential"



Fusion in programming — "sequential"



Fusion in programming — "parallel"

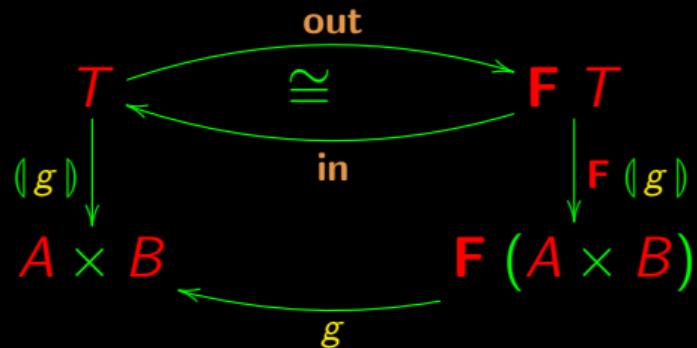


Fusion in programming — "parallel"

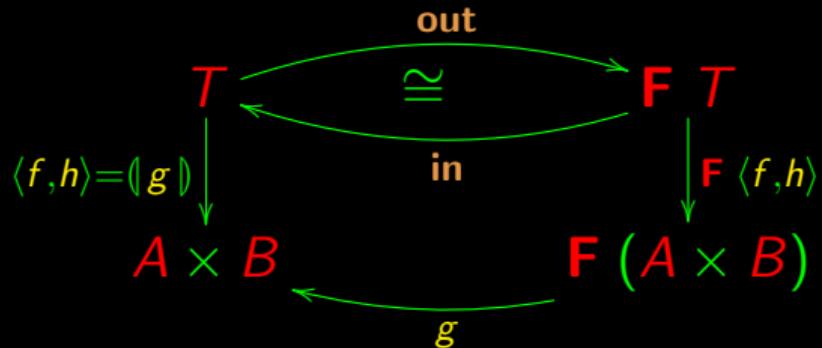


The pattern $\langle f, g \rangle = (\ h \)$

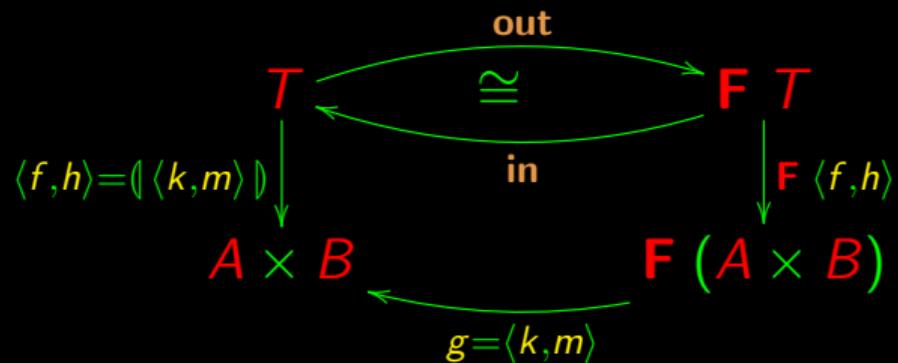
Catas that yield pairs:



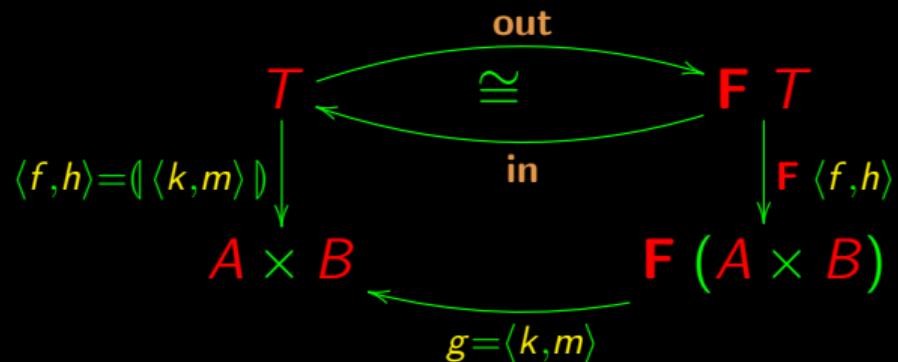
Catas that yield pairs



Mutual recursion



Mutual recursion



$$\langle f, h \rangle = \langle \langle k, m \rangle \rangle \Leftrightarrow \begin{cases} f \cdot \mathbf{in} = k \cdot \mathbf{F} \langle f, h \rangle \\ h \cdot \mathbf{in} = m \cdot \mathbf{F} \langle f, h \rangle \end{cases}$$

Mutual recursion

$$\langle f, h \rangle = (\langle k, m \rangle)$$

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

$$\langle f, h \rangle \cdot \mathbf{in} = \langle k, m \rangle \cdot \mathbf{F} \langle f, h \rangle$$

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

$$\langle f \cdot \mathbf{in}, h \cdot \mathbf{in} \rangle = \langle k \cdot \mathbf{F} \langle f, h \rangle, m \cdot \mathbf{F} \langle f, h \rangle \rangle$$

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

$$\begin{cases} f \cdot \mathbf{in} = k \cdot \mathbf{F} \langle f, h \rangle \\ h \cdot \mathbf{in} = m \cdot \mathbf{F} \langle f, h \rangle \end{cases}$$

Mutual recursion — case $T = \mathbb{N}_0$

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

$\Leftrightarrow \{ \text{ mutual recursion with } \mathbf{in} = [\underline{0}, \mathbf{succ}] \text{ etc } \}$

$$\begin{cases} f \cdot [\underline{0}, \mathbf{succ}] = h \cdot (id + \langle f, g \rangle) \\ g \cdot [\underline{0}, \mathbf{succ}] = k \cdot (id + \langle f, g \rangle) \end{cases}$$

$\Leftrightarrow \{ \text{ let } h = [h_1, h_2] \text{ and } k = [k_1, k_2] \}$

$$\begin{cases} f \cdot [\underline{0}, \mathbf{succ}] = [h_1, h_2] \cdot (id + \langle f, g \rangle) \\ g \cdot [\underline{0}, \mathbf{succ}] = [k_1, k_2] \cdot (id + \langle f, g \rangle) \end{cases}$$

Mutual recursion — case $T = \mathbb{N}_0$

Moving on:

$$\langle f, g \rangle = (\langle [h_1, h_2], [k_1, k_2] \rangle) \Leftrightarrow \left\{ \begin{array}{l} \text{coproduct laws} \\ \left\{ \begin{array}{l} f \cdot \underline{0} = h_1 \\ f \cdot \text{succ} = h_2 \cdot \langle f, g \rangle \\ g \cdot \underline{0} = k_1 \\ g \cdot \text{succ} = k_2 \cdot \langle f, g \rangle \end{array} \right. \end{array} \right.$$

Mutual recursion — case $T = \mathbb{N}_0$

Finally, add variables to righthand side, with $h_1 = \underline{a}$ and $k_1 = \underline{b}$:

$$\langle f, g \rangle = (\langle [\underline{a}, h_2], [\underline{b}, k_2] \rangle) \Leftrightarrow \left\{ \begin{array}{l} \left\{ \begin{array}{l} f 0 = \underline{a} \\ f (n + 1) = h_2 (f n, g n) \end{array} \right. \\ \left\{ \begin{array}{l} g 0 = \underline{b} \\ g (n + 1) = k_2 (f n, g n) \end{array} \right. \end{array} \right.$$

Example — Fibonacci

Clearly, the following function is not a catamorphism of \mathbb{N}_0 :

$$\text{fib } 0 = 1$$

$$\text{fib } 1 = 1$$

$$\text{fib } (n + 2) = \text{fib } (n + 1) + \text{fib } n$$

But it can be transformed into one using mutual recursion.

Example — Fibonacci

Let us define the auxiliary function:

$$f\ n = \text{fib}\ (n + 1)$$

Then:

$$f\ 0 = \text{fib}\ 1 = 1$$

$$f\ (n + 1) = \text{fib}\ (n + 2) = f\ n + \text{fib}\ n$$

Example — Fibonacci

Concerning *fib*:

$$\text{fib } 0 = 1$$

$$\text{fib } (n + 1) = f \ n$$

Summing up, we have the following mutual recursion:

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} f \ 0 = 1 \\ f \ (n + 1) = f \ n + \text{fib } n \end{array} \right. \\ \left\{ \begin{array}{l} \text{fib } 0 = 1 \\ \text{fib } (n + 1) = f \ n \end{array} \right. \end{array} \right.$$

Example — Fibonacci

By applying the mutual recursion law for $T = \mathbb{N}_0$:

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} f \ 0 = \underline{a} \\ f \ (n + 1) = h_2 \ (f \ n, fib \ n) \end{array} \right. \\ \left\{ \begin{array}{l} fib \ 0 = \underline{b} \\ fib \ (n + 1) = k_2 \ (f \ n, fib \ n) \end{array} \right. \end{array} \right. \Leftrightarrow \langle f, fib \rangle = (\langle [\underline{a}, h_2], [\underline{b}, k_2] \rangle)$$

Example — Fibonacci

Solutions:

$$a = 1$$

$$b = 1$$

$$h_2 = add$$

$$k_2 = \pi_1$$

Then:

$$\langle f, fib \rangle = () \langle [\underline{1}, add], [\underline{1}, \pi_1] \rangle ()$$

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 $i = \langle [i, b] \rangle$

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 $i = \langle [i, b] \rangle$

$$\langle f, fib \rangle = \langle \langle [\underline{1}, add], [\underline{1}, \pi_1] \rangle \rangle$$

\Leftrightarrow { exchange law }

$$\langle f, fib \rangle = \langle [\langle \underline{1}, \underline{1} \rangle, \langle add, \pi_1 \rangle] \rangle$$

\Leftrightarrow { recall exercise sheet 3 }

$$\langle f, fib \rangle = \langle [\underline{(1, 1)}, \langle add, \pi_1 \rangle] \rangle$$

\Leftrightarrow { definition of **for** b i }

$$\langle f, fib \rangle = \text{for } \langle add, \pi_1 \rangle (1, 1)$$

Example — Fibonacci

In summary:

$$fib = \pi_2 \cdot (\mathbf{for} \langle add, \pi_1 \rangle (1, 1))$$

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

$$\langle f, \text{fac} \rangle = (\langle [\underline{a}, h_2], [\underline{b}, k_2] \rangle) \Leftrightarrow \left\{ \begin{array}{l} \left\{ \begin{array}{l} f 0 = a \\ f (n + 1) = h_2 (f n, \text{fac } n) \end{array} \right. \\ \left\{ \begin{array}{l} \text{fac } 0 = b \\ \text{fac } (n + 1) = k_2 (f n, \text{fac } n) \end{array} \right. \end{array} \right.$$

$k_2 = \text{mul}$

$b = 1$

$f n = n + 1$

$a = 1$

(Still need h_2 ...)

Example — factorial

$$\begin{aligned} & f(n+1) \\ = & (n+1)+1 \\ = & 1 + (n+1) \\ = & 1 + f n \\ = & 1 + \pi_1(f n, fac n) \\ = & \underbrace{\text{succ} \cdot \pi_1}_{h_2}(f n, fac n) \end{aligned}$$

Then:

$$k_2 = \text{mul}$$

$$b = 1$$

$$h_2 = \text{succ} \cdot \pi_1$$

$$a = 1$$

Example — factorial becomes a for-loop

Finally:

$$\begin{aligned}& \langle f, \text{fac} \rangle \\&= \langle \langle [\underline{1}, \text{succ} \cdot \pi_1], [\underline{1}, \text{mul}] \rangle \rangle \\&= \langle [\underline{(1, 1)}, \langle \text{succ} \cdot \pi_1, \text{mul} \rangle] \rangle \\&= \text{for } \langle \text{succ} \cdot \pi_1, \text{mul} \rangle ((1, 1)) \\&= \text{for } g (1, 1) \text{ where } g (\textcolor{red}{x}, \textcolor{red}{y}) = (\textcolor{red}{x} + 1, \textcolor{red}{x} \times \textcolor{red}{y})\end{aligned}$$

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



(⟨-, -⟩)

⟨(-), (-)⟩

Banana-split



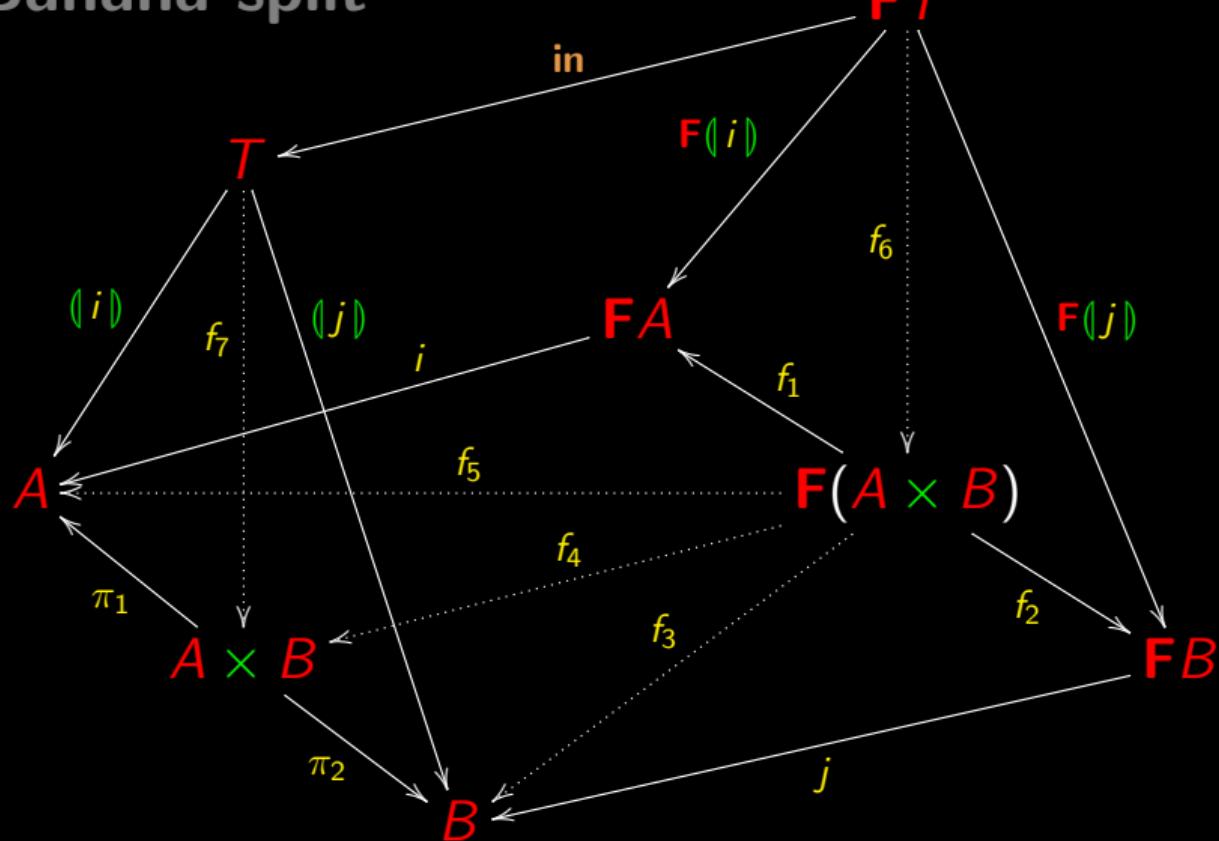
$(\langle -,- \rangle)$

$\langle (_), (_) \rangle$

$\langle (_i), (_j) \rangle$

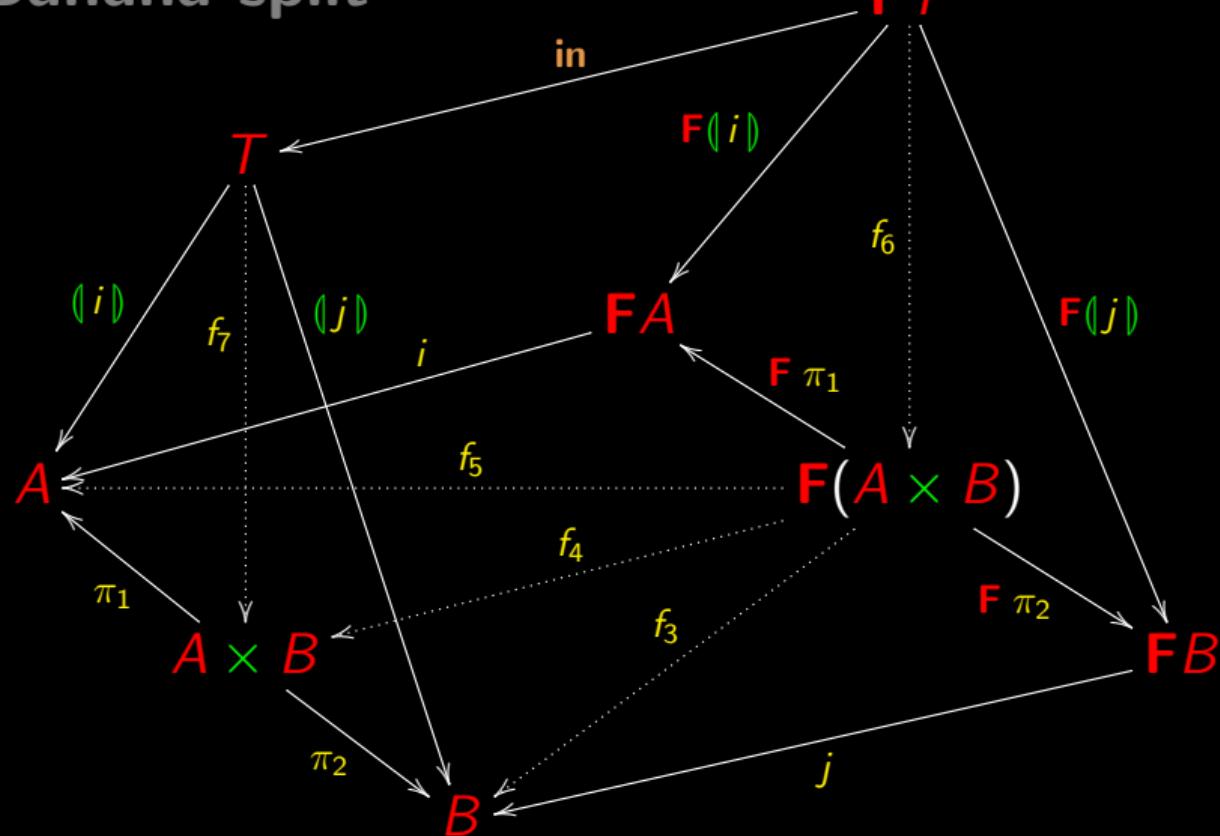
Banana-split

$\langle \langle i \rangle, \langle j \rangle \rangle$



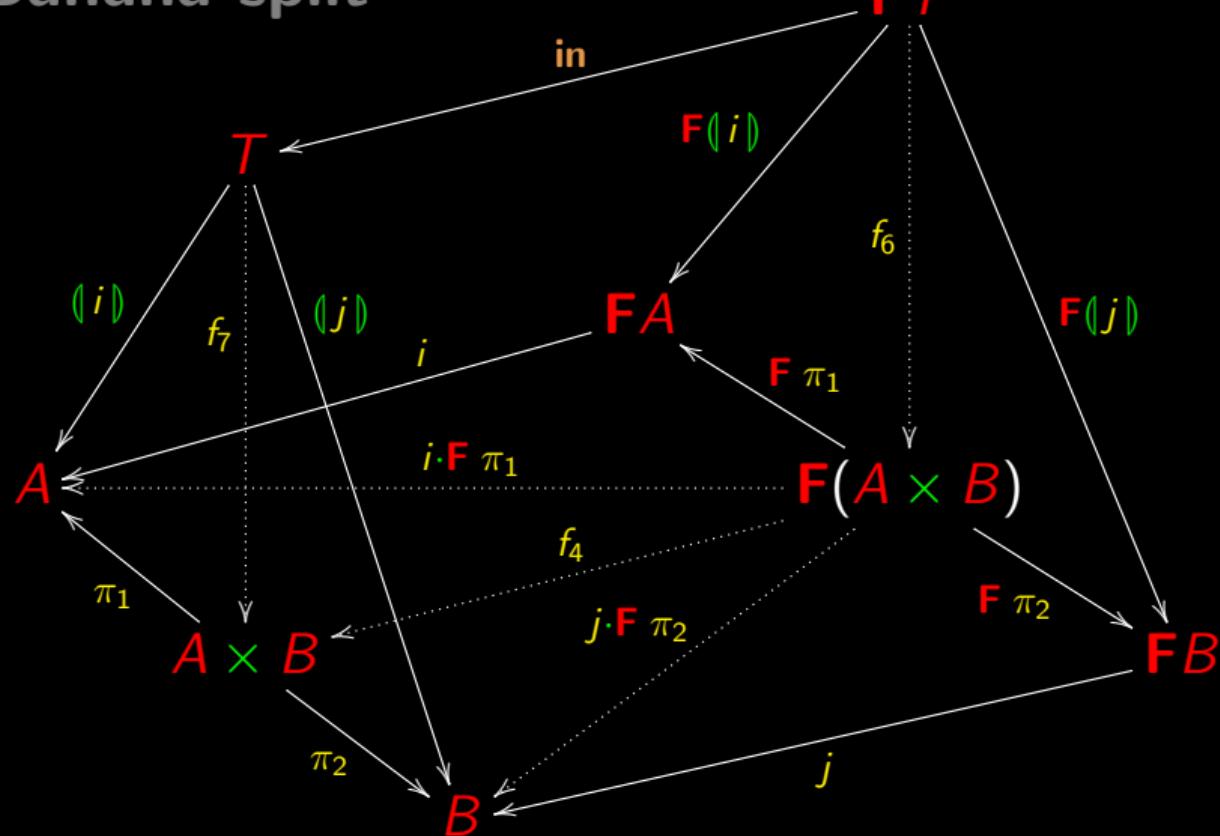
Banana-split

$\langle \langle i \rangle, \langle j \rangle \rangle$



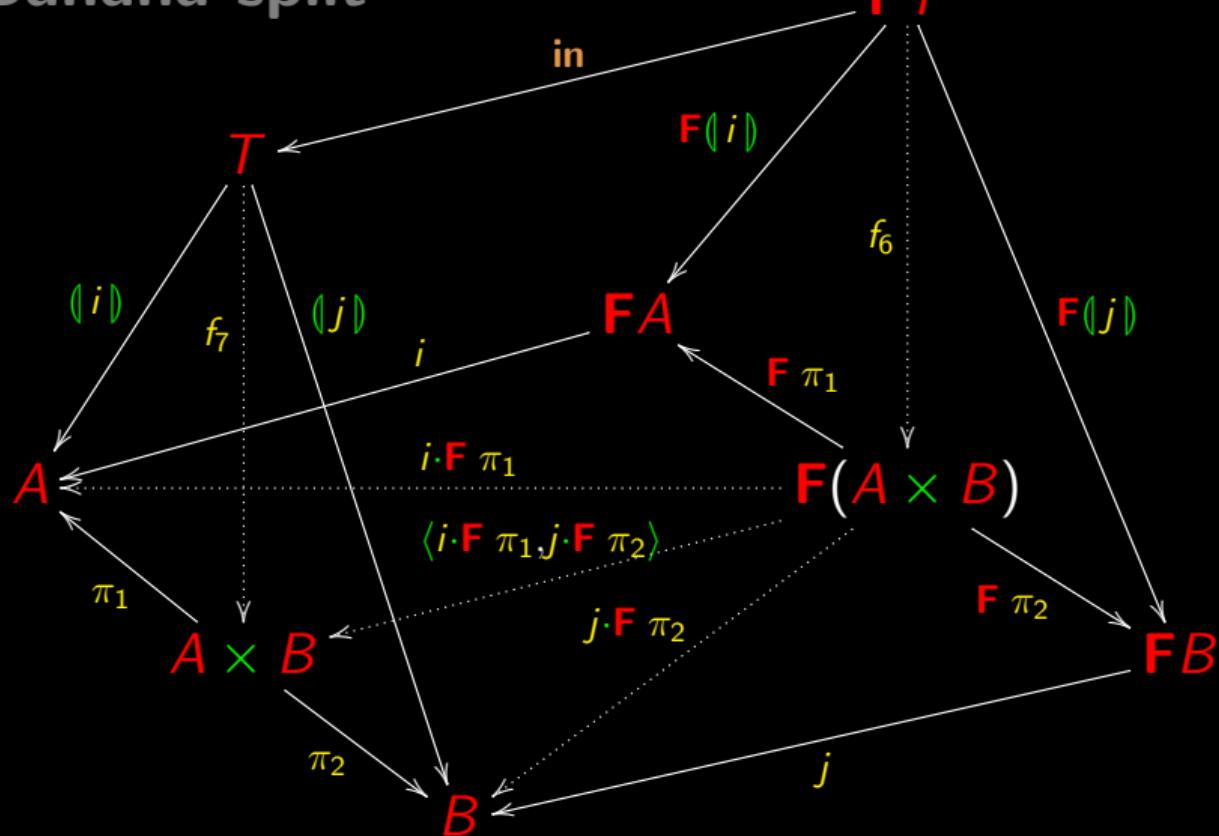
Banana-split

$\langle \langle i \rangle, \langle j \rangle \rangle$



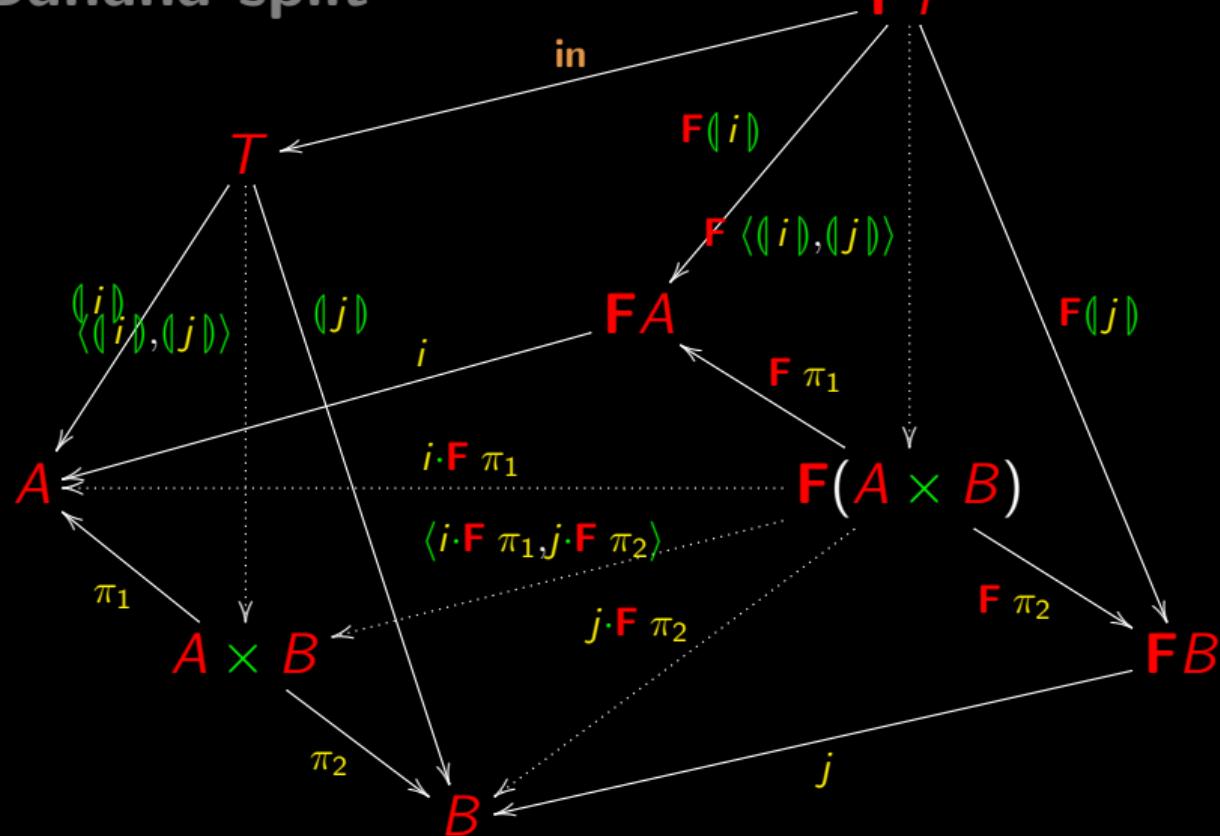
Banana-split

$\langle \langle i \rangle, \langle j \rangle \rangle$



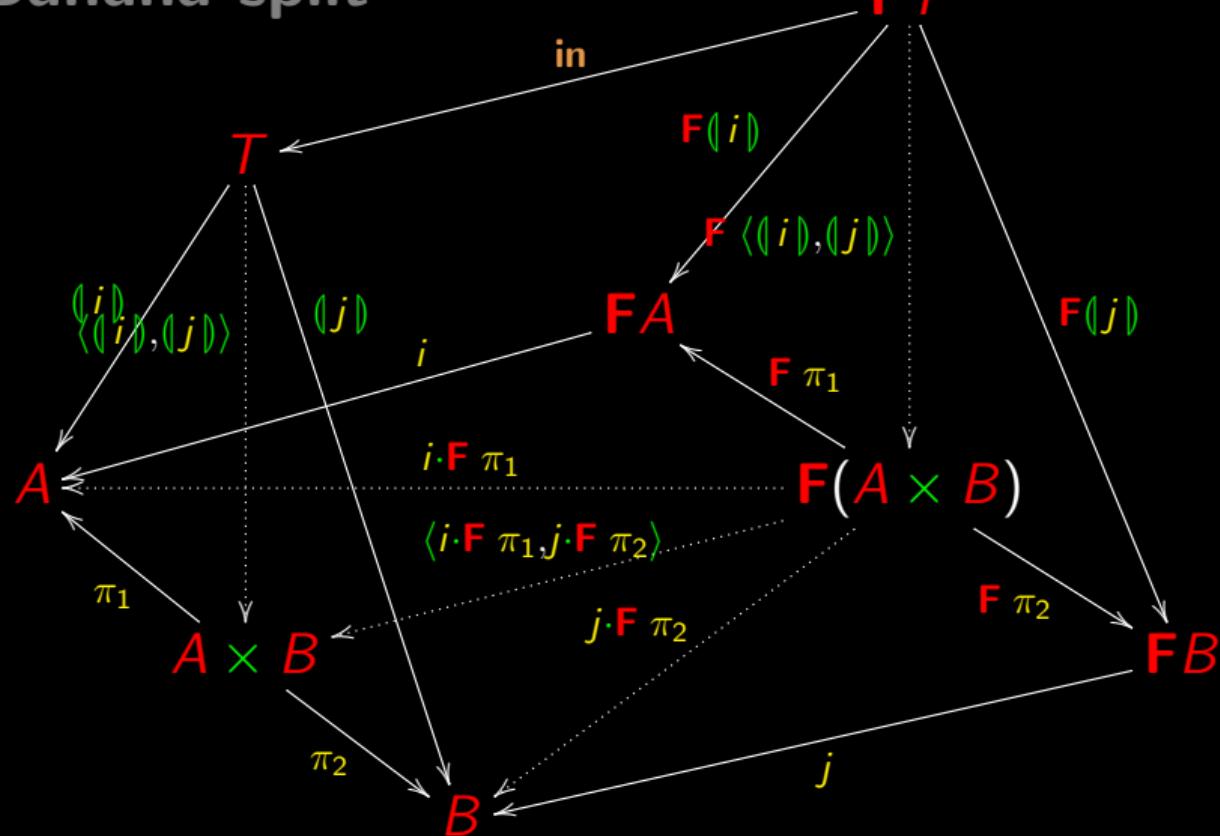
Banana-split

$\langle \langle i \rangle, \langle j \rangle \rangle$



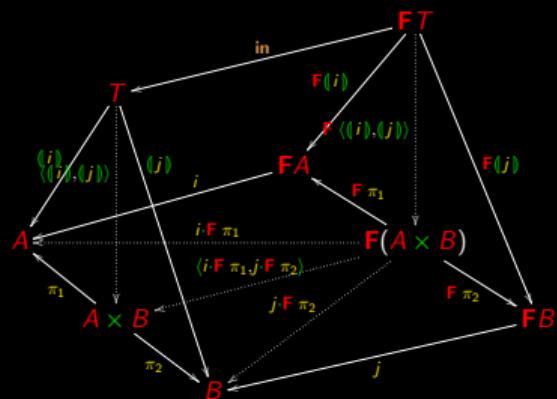
Banana-split

$\langle \langle i \rangle, \langle j \rangle \rangle$



Banana-split

$$\langle \langle i \rangle \rangle, \langle \langle j \rangle \rangle = \langle \langle i \cdot F \pi_1, j \cdot F \pi_2 \rangle \rangle$$



Example

Average of a (non empty) list:

$$\text{average} = (/) \cdot \langle \text{sum}, \text{length} \rangle$$

where

$$\text{sum} = () [0, \text{add}] ()$$

$$\text{length} = () [0, \text{succ} \cdot \pi_2] ()$$

Example

$$\text{average} = (/) \cdot \underbrace{\langle \text{sum}, \text{length} \rangle}_{\text{aux}}$$

By banana-split:

$$\text{aux} = (\langle f_5, f_3 \rangle) \text{ where}$$

$$f_5 = [0, \text{add}] \cdot (\text{id} + \text{id} \times \pi_1)$$

$$f_3 = [0, \text{succ} \times \pi_2] \cdot (\text{id} + \text{id} \times \pi_2)$$



Example

Finally, solve $\text{aux} = (\langle f_5, f_3 \rangle)$ and convert everything to pointwise notation:



average $l = x / y$ where

$(x, y) = \text{aux} l$

$\text{aux} [] = (0, 0)$

$\text{aux} (a : l) = (a + x, y + 1)$ where $(x, y) = \text{aux} l$

Example

Summing up,

average l = x / y where

(x, y) = aux l

aux [] = (0, 0)

aux (a : l) = (a + x, y + 1) where (x, y) = aux l

twice as fast as

average l = (sum l) / (length l) where

sum [] = 0

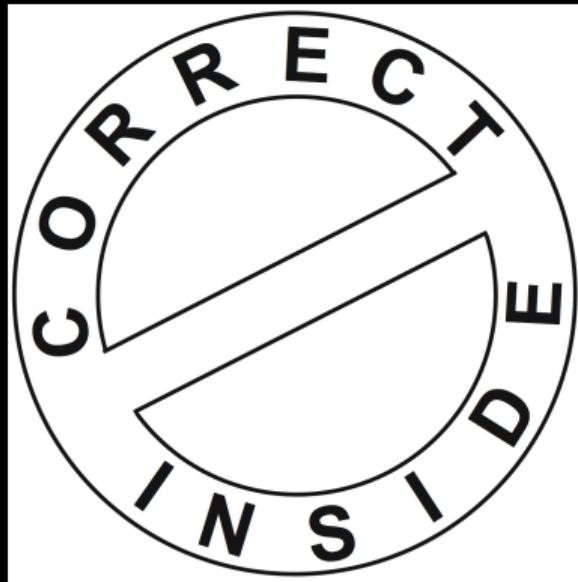
sum (h : t) = h + sum t

length [] = 0

length (h : t) = 1 + length t

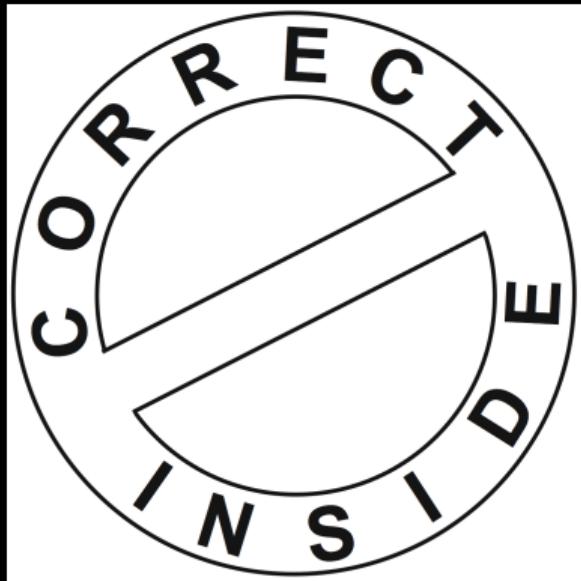
and **guaranteed** to be correct.

Program Calculation



Efficiency without sacrificing
correction.

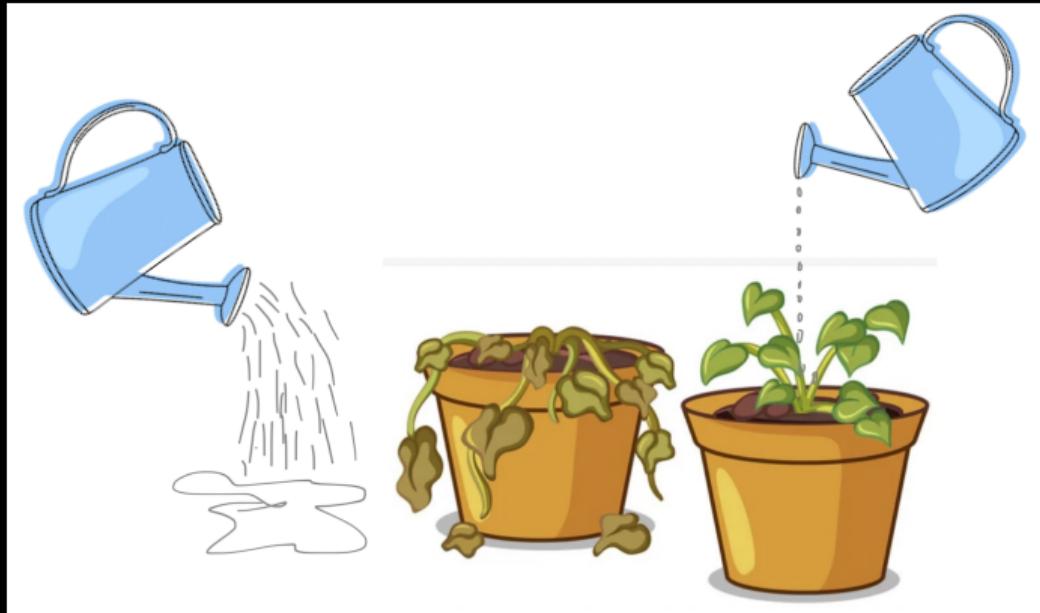
Program Calculation



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:

$$T \begin{array}{c} \xrightarrow{\text{out}} \\ \cong \\ \xleftarrow{\text{in}} \end{array} F T$$

Recap

Inductive type T and the functor \mathbf{F} that determines its pattern of recursion:

$$T \begin{array}{c} \xrightarrow{\text{out}} \\ \cong \\ \xleftarrow{\text{in}} \end{array} \mathbf{F} T$$

We have seen examples: $T = \mathbb{N}_0$, $T = A^*$ etc.

Recap

Inductive type T and the functor \mathbf{F} that determines its pattern of recursion:

$$T \begin{array}{c} \xrightarrow{\text{out}} \\ \cong \\ \xleftarrow{\text{in}} \end{array} \mathbf{F} T$$

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))
```

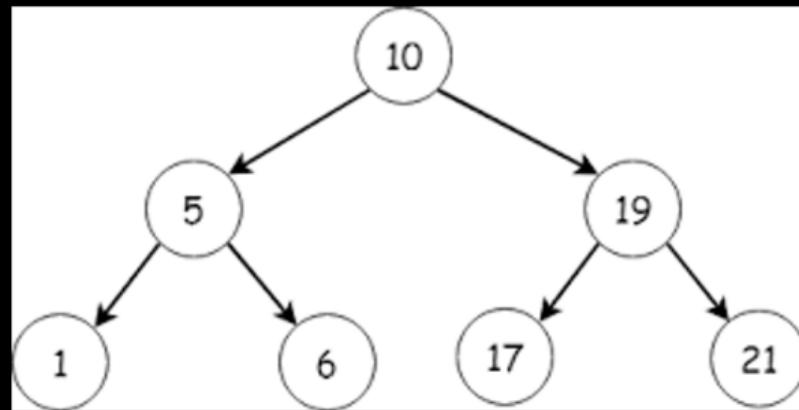
(A assumed pre-defined.)

Binary trees

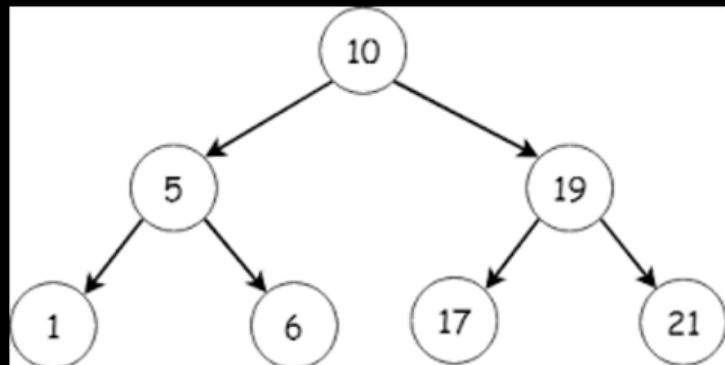
Trees with data of type *A* in their nodes, for example

```
data BTTree = Empty | Node (A, (BTTree, BTTree))
```

(*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 BTTree = Empty | Node(A, (BTTree, BTTree))  
  
: :t Node  
  
Node :: (A, (BTTree, BTTree)) -> BTTree  
  
: :t Empty  
  
Empty :: BTTree
```

Binary trees

```
: data BTTree = Empty | Node(A, (BTTree, BTTree))  
  
: :t Node  
  
Node :: (A, (BTTree, BTTree)) -> BTTree  
  
: :t Empty  
  
Empty :: BTTree
```

$$\text{Node} : A \times (BTTree \times BTTree) \rightarrow BTTree$$

$$\underline{\text{Empty}} : 1 \rightarrow BTTree$$

Binary trees

in = [Empty, Node]

Binary trees

in = [Empty, Node]

$$BTree \underset{\text{in}}{\underset{\cong}{\longleftrightarrow}} \underset{\text{F } BTree}{\overbrace{1 + A \times (BTree \times BTree)}}$$

data *BTree* = *Empty* | *Node* (*A*, (*BTree*, *BTree*))

Binary trees

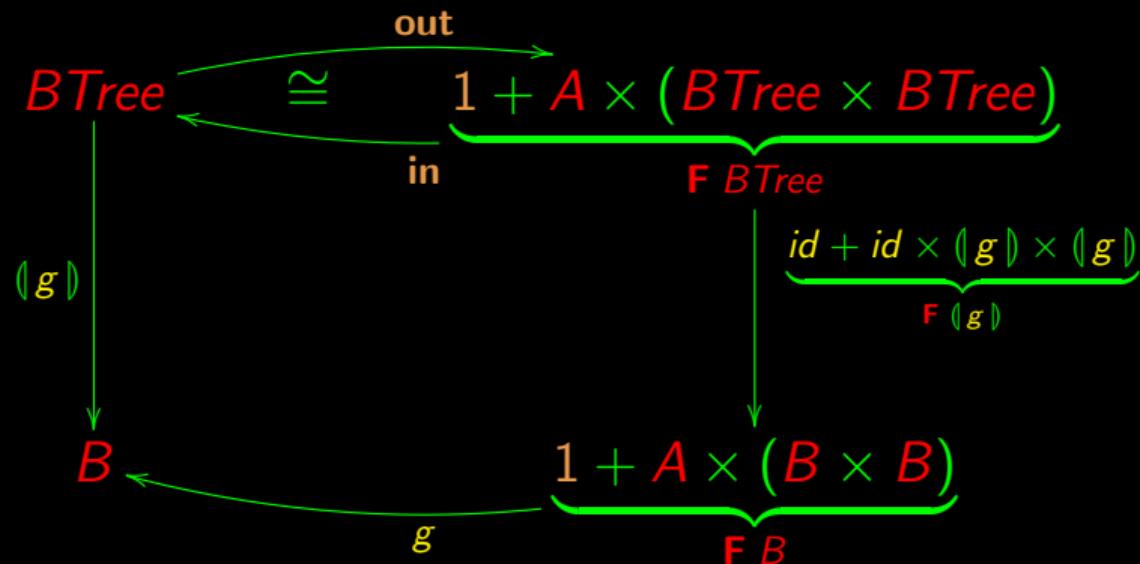
$$BTree \underset{\text{in}}{\approx} \underset{\text{out}}{\overset{\curvearrowright}{1 + A \times (\textcolor{red}{BTree} \times \textcolor{green}{BTree})}} \underset{\mathbf{F} BTree}{\curvearrowleft}$$

$$T = BTree$$

$$\begin{cases} \mathbf{F} X = 1 + A \times (X \times X) \\ \mathbf{F} f = id + id \times (f \times f) \end{cases}$$

$$\mathbf{in} = [\underline{Empty}, \textcolor{green}{Node}]$$

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

$$f : \textcolor{red}{BTree} \rightarrow \mathbb{N}_0$$

that computes the size of the tree (total number of nodes).

Catamorphisms of binary trees

Example: function

$$f : \textcolor{red}{BTree} \rightarrow \mathbb{N}_0$$

that computes the size of the tree (total number of nodes).

Enough caring about the “gene” g in

$$f = (\textcolor{green}{g})$$

where

$$\mathbb{N}_0 \xleftarrow{\textcolor{green}{g}} 1 + \textcolor{red}{A} \times (\mathbb{N}_0 \times \mathbb{N}_0)$$

Catamorphisms of binary trees

$$\mathbb{N}_0 \xleftarrow{g} 1 + A \times (\mathbb{N}_0 \times \mathbb{N}_0)$$

Catamorphisms of binary trees

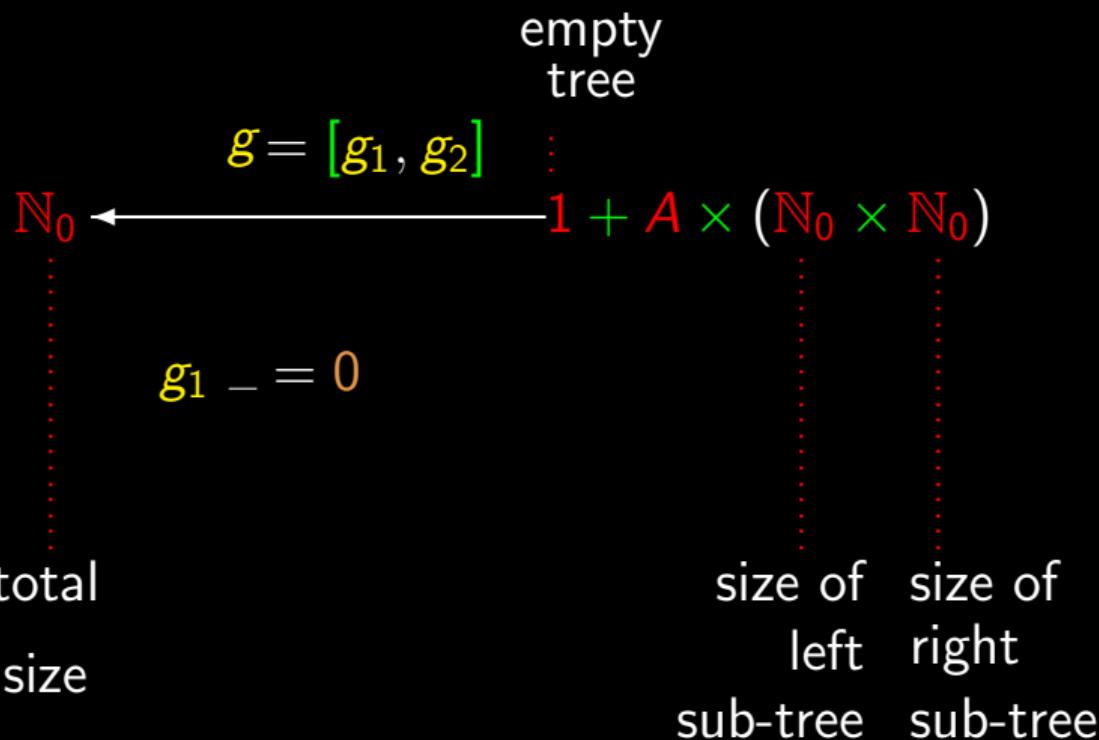
$$\begin{array}{ccc} g = [g_1, g_2] & & \\ \textcolor{red}{\mathbb{N}_0} & \xleftarrow{\hspace{1cm}} & \textcolor{red}{1 + A \times (\mathbb{N}_0 \times \mathbb{N}_0)} \end{array}$$

Catamorphisms of binary trees

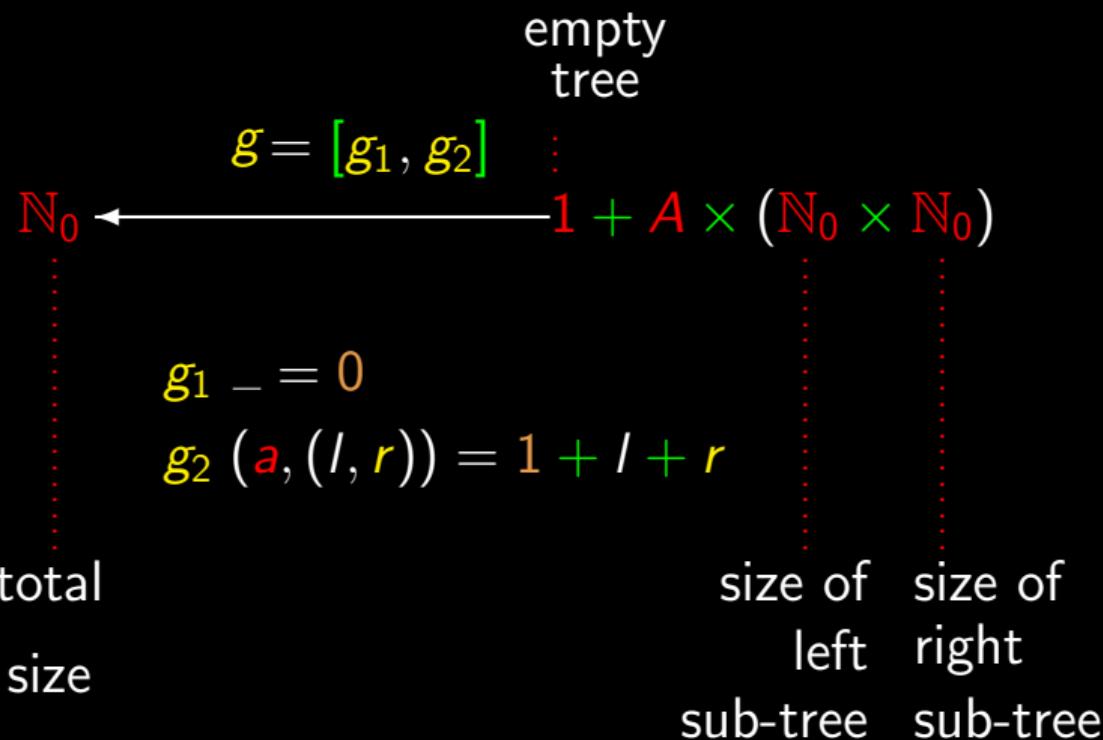
$$\begin{array}{ccc} & \text{empty} & \\ & \text{tree} & \\ g = [g_1, g_2] & : & \\ \mathbb{N}_0 & \xleftarrow{\hspace{1cm}} & 1 + A \times (\mathbb{N}_0 \times \mathbb{N}_0) \end{array}$$

$$g_1 - = 0$$

Catamorphisms of binary trees



Catamorphisms of binary trees



Catamorphisms of binary trees

Summing up,

$$f = \text{⟨} [g_1, g_2] \text{⟩ \text{ where}$$

$$g_1 _ = 0$$

$$g_2 (a, (l, r)) = 1 + l + r$$

Catamorphisms of binary trees

Summing up,

$$f = ([g_1, g_2]) \text{ where}$$

$$g_1 _ = 0$$

$$g_2 (a, (l, r)) = 1 + l + r$$

Haskell:

```
size = cataBTree(either g1 g2) where
  g1 \_ = 0
  g2(a, (l, r)) = 1 + l + r
```

Cálculo de Programas

Class T09

Inductive parametric types

Let us now show how recursive types become **parametric**.

Inductive parametric types

Let us now show how recursive types become **parametric**.

That is, they become **functors**.

Inductive parametric types

Let us now show how recursive types become **parametric**.

That is, they become **functors**.

It all amounts to generalize *map* to other recursive types.

Recall...

Functor \mathbf{F} :

$$\begin{array}{ccc} A & \xrightarrow{\quad} & \mathbf{F} A \\ f \downarrow & & \downarrow \mathbf{F} f \\ B & \xrightarrow{\quad} & \mathbf{F} B \end{array}$$

Recall...

Functor \mathbf{F} :

$$\begin{array}{ccc} A & \xrightarrow{\quad} & \mathbf{F} A \\ f \downarrow & & \downarrow \mathbf{F} f \\ B & \xrightarrow{\quad} & \mathbf{F} B \end{array}$$

$$\left\{ \begin{array}{l} \mathbf{F} id = id \\ \mathbf{F} (g \cdot f) = (\mathbf{F} g) \cdot (\mathbf{F} f) \end{array} \right.$$

map f

Functor $\mathbf{F} A = A^*$:

$$\begin{array}{ccc} A & \xrightarrow{\quad} & \mathbf{F} A \\ f \downarrow & & \downarrow f^* = \text{map } f \\ B & \xrightarrow{\quad} & B^* \end{array}$$

map f

Functor $\mathbf{F} A = A^*$:

$$\begin{array}{ccc} A & \xrightarrow{\quad \mathbf{F} \quad} & \mathbf{F} A \\ f \downarrow & & \downarrow f^* = \text{map } f \\ B & \xrightarrow{\quad} & B^* \end{array}$$
$$\left\{ \begin{array}{l} \text{map } id = id \\ \text{map } (g \cdot f) = (\text{map } g) \cdot (\text{map } f) \end{array} \right.$$

How do we **prove** this? **Generalize** this?

map f

From *specification*

$$\text{map } f \ [] = []$$

$$\text{map } f \ [a] = [f \ a]$$

$$\text{map } f \ (x + y) = \text{map } f \ x + \text{map } f \ y$$

easy to obtain (as before):

$$\text{map } f \ [] = []$$

$$\text{map } f \ (h : t) = f \ h : \text{map } f \ t$$

map f

Then *implementation*

map f [] = []

map f (h : t) = f h : map f t

map f

Then *implementation*

map f [] = []

map f (h : t) = f h : map f t

is

map f = ([nil, g₂]) where

g₂ (h, x) = f h : x

map f

Then *implementation*

$$\text{map } f \ [] = []$$

$$\text{map } f \ (h : t) = f \ h : \text{map } f \ t$$

is

$$\begin{aligned} \text{map } f &= (\text{nil}, g_2) \text{ where} \\ g_2 \ (h, x) &= f \ h : x \end{aligned}$$

the same thing as $\text{map } f = (\text{nil}, \text{cons}) \cdot (id + f \times id)$

map f

For instance, for *sq* $x = x^2$

map sq [1, 7, 8, 9, 3, 2] = [1, 49, 64, 81, 9, 4]

map f

For instance, for *sq* $x = x^2$

$$\text{map } \text{sq} [1, 7, 8, 9, 3, 2] = [1, 49, 64, 81, 9, 4]$$

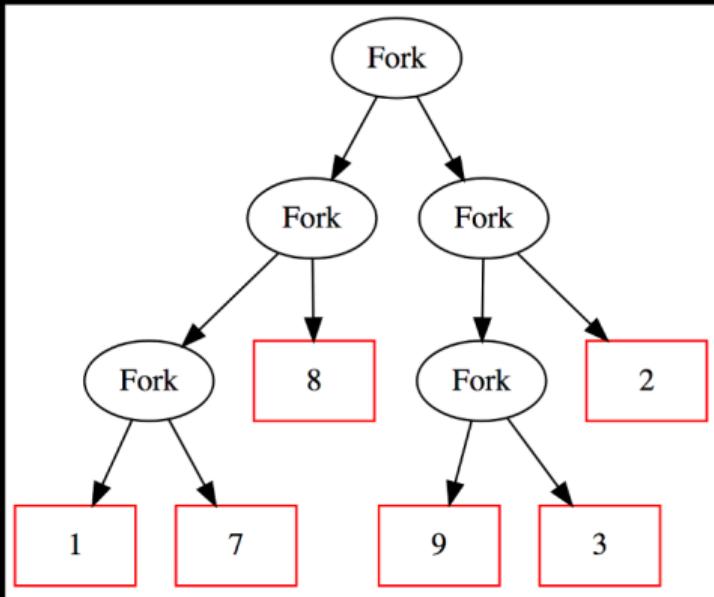
But still we haven't proved

$$\text{map } \text{id} = \text{id}$$

$$\text{map } (f \cdot g) = (\text{map } f) \cdot (\text{map } g)$$

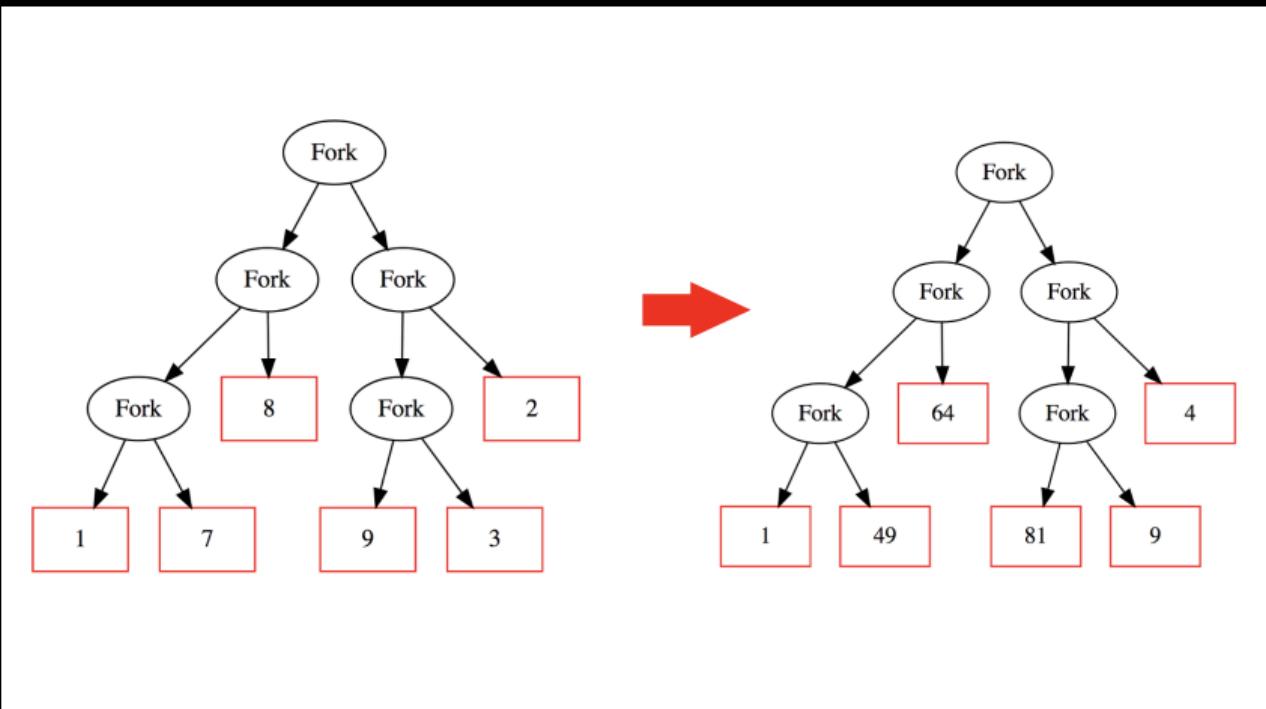
(Coming up soon.)

Inductive parametric types



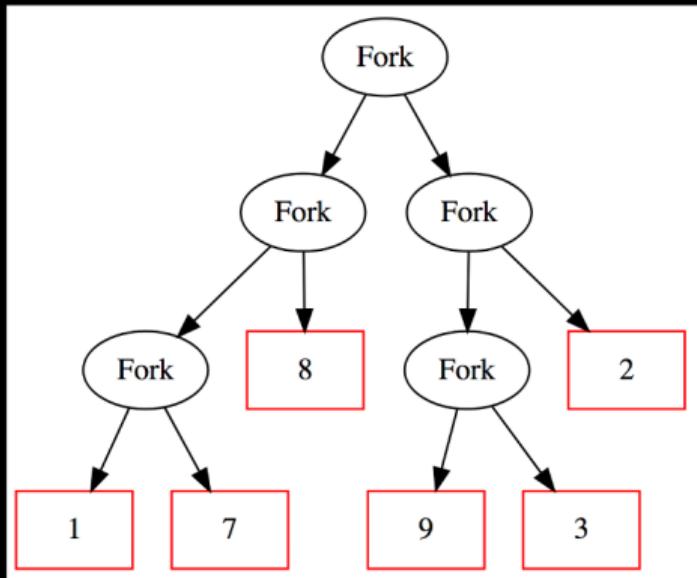
```
data LTree = Leaf Int | Fork (LTree, LTree)
```

Inductive parametric types



Inductive parametric types

$t = \text{Fork}(\text{Fork}(\text{Fork}(\text{Leaf}\ 1, \text{Leaf}\ 7), \text{Leaf}\ 8), \text{Fork}(\text{Fork}(\text{Leaf}\ 9, \text{Leaf}\ 3), \text{Leaf}\ 2))$



Inductive parametric types

$$k (\textcolor{red}{Leaf} \ a) = \textcolor{red}{Leaf} \ (a * a)$$
$$k (\textcolor{blue}{Fork} \ (l, r)) = \textcolor{blue}{Fork} \ (k \ l, k \ r)$$

$$\begin{aligned} k \ t &= \textcolor{blue}{Fork} \\ &(\textcolor{blue}{Fork} \\ &\quad (\textcolor{blue}{Fork} \ (\textcolor{brown}{Leaf} \ 1, \textcolor{brown}{Leaf} \ 49), \\ &\quad \textcolor{brown}{Leaf} \ 64), \\ &\textcolor{blue}{Fork} \\ &\quad (\textcolor{blue}{Fork} \ (\textcolor{brown}{Leaf} \ 81, \textcolor{brown}{Leaf} \ 9), \\ &\quad \textcolor{brown}{Leaf} \ 4)) \end{aligned}$$

Inductive parametric types

$$k (\textcolor{blue}{Leaf} \ a) = \textcolor{blue}{Leaf} \ (a * a)$$

$$k (\textcolor{blue}{Fork} \ (l, r)) = \textcolor{blue}{Fork} \ (k l, k r)$$

Inductive parametric types

$$k (\textcolor{blue}{Leaf} \ a) = \textcolor{blue}{Leaf} \ (a * a)$$

$$k (\textcolor{blue}{Fork} \ (l, r)) = \textcolor{blue}{Fork} \ (k l, k r)$$

By analogy with *map f*:

$$k f (\textcolor{blue}{Leaf} \ a) = \textcolor{blue}{Leaf} \ (f \ a)$$

$$k f (\textcolor{blue}{Fork} \ (l, r)) = \textcolor{blue}{Fork} \ (k f l, k f r)$$

Inductive parametric types

$$\text{map } id = id$$

$$\text{map } (f \cdot g) = (\text{map } f) \cdot (\text{map } g)$$

Inductive parametric types

$$\text{map } id = id$$

$$\text{map } (f \cdot g) = (\text{map } f) \cdot (\text{map } g)$$

$$k \text{ } id = id$$

$$k \text{ } (f \cdot g) = (k \text{ } f) \cdot (k \text{ } g)$$

?

Inductive parametric types

$$\left\{ \begin{array}{l} k f (\textcolor{red}{Leaf } a) = \textcolor{green}{Leaf } (f a) \\ k f (\textcolor{green}{Fork } (l, r)) = \textcolor{green}{Fork } (k f l, k f r) \end{array} \right.$$

Inductive parametric types

$$\left\{ \begin{array}{l} k f (\textcolor{teal}{Leaf } a) = \textcolor{teal}{Leaf } (f a) \\ k f (\textcolor{teal}{Fork } (l, r)) = \textcolor{teal}{Fork } (k f l, k f r) \end{array} \right. \Leftrightarrow \quad \left\{ \begin{array}{l} \text{go pointfree} \end{array} \right\}$$

$$\left\{ \begin{array}{l} k f \cdot \textcolor{teal}{Leaf } = \textcolor{teal}{Leaf } \cdot f \\ k f \cdot \textcolor{teal}{Fork } = \textcolor{teal}{Fork } \cdot (k f \times k f) \end{array} \right.$$

Inductive parametric types

$$\left\{ \begin{array}{l} k f (\textcolor{red}{Leaf } a) = \textcolor{green}{Leaf } (f a) \\ k f (\textcolor{red}{Fork } (l, r)) = \textcolor{green}{Fork } (k f l, k f r) \end{array} \right.$$

\Leftrightarrow { go pointfree }

$$\left\{ \begin{array}{l} k f \cdot \textcolor{red}{Leaf } = \textcolor{green}{Leaf } \cdot f \\ k f \cdot \textcolor{red}{Fork } = \textcolor{green}{Fork } \cdot (k f \times k f) \end{array} \right.$$

\Leftrightarrow { +Eq ; +fusion ; +absorption }

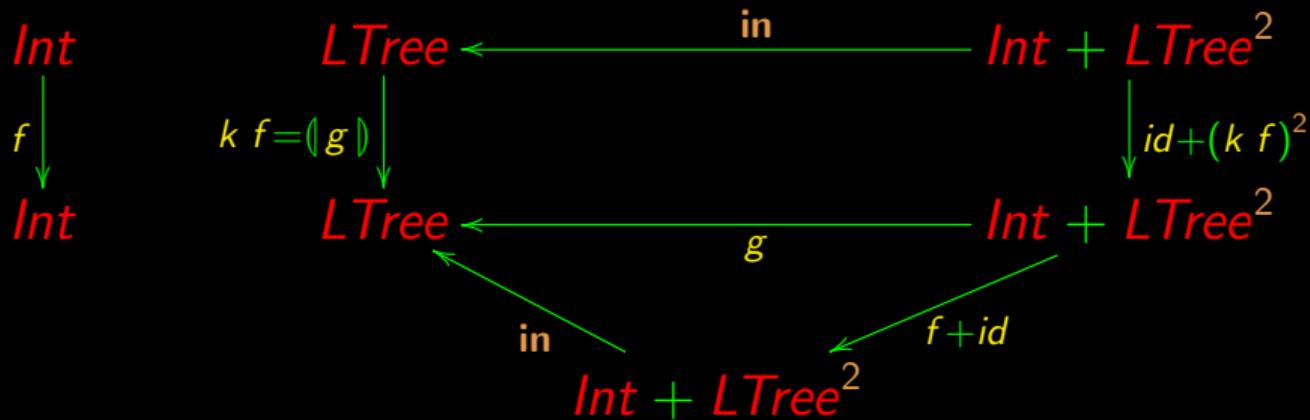
Inductive parametric types

$$k f \cdot [Leaf, Fork] = [Leaf, Fork] \cdot (f + k f \times k f)$$

Inductive parametric types

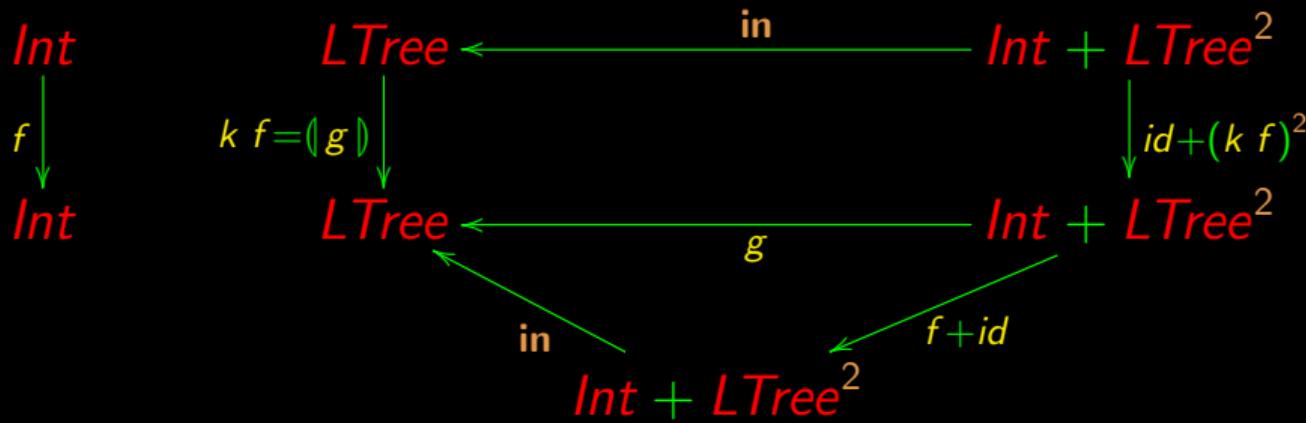
$$\begin{aligned} k f \cdot [Leaf, Fork] &= [Leaf, Fork] \cdot (f + k f \times k f) \\ \Leftrightarrow \quad \quad \quad \left\{ \begin{array}{l} [Leaf, Fork] = \mathbf{in} ; \text{functor-+} \end{array} \right\} \\ k f \cdot \mathbf{in} &= \mathbf{in} \cdot (f + id) \cdot (id + k f \times k f) \end{aligned}$$

Inductive parametric types



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

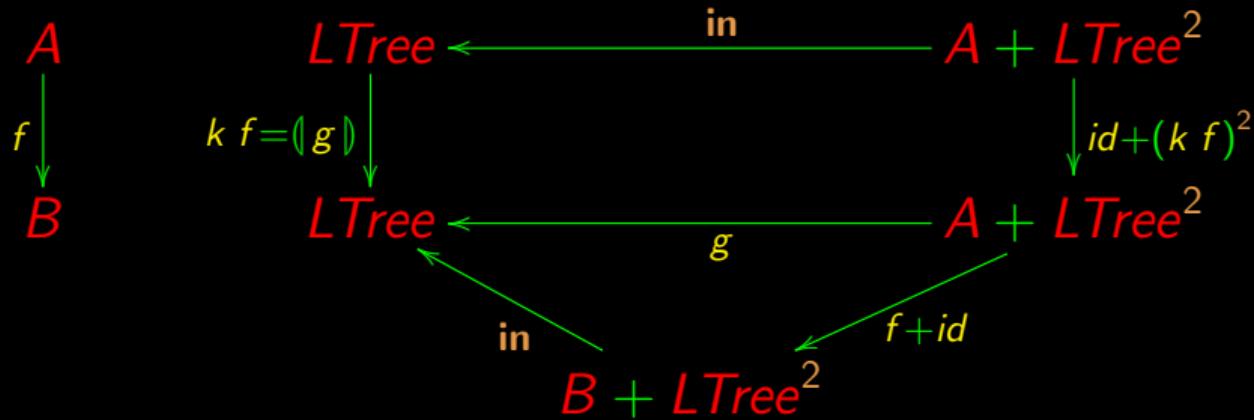
Inductive parametric types



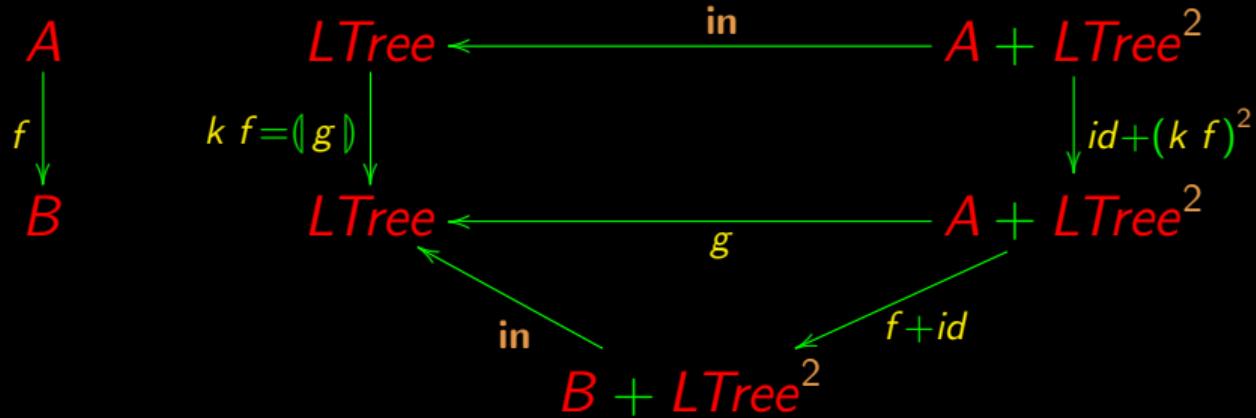
$$k f \cdot \text{in} = \text{in} \cdot (f + id) \cdot (id + k f \times k f)$$

NB: $L\text{Tree}^2$ abbreviates $L\text{Tree} \times L\text{Tree}$

Inductive parametric types

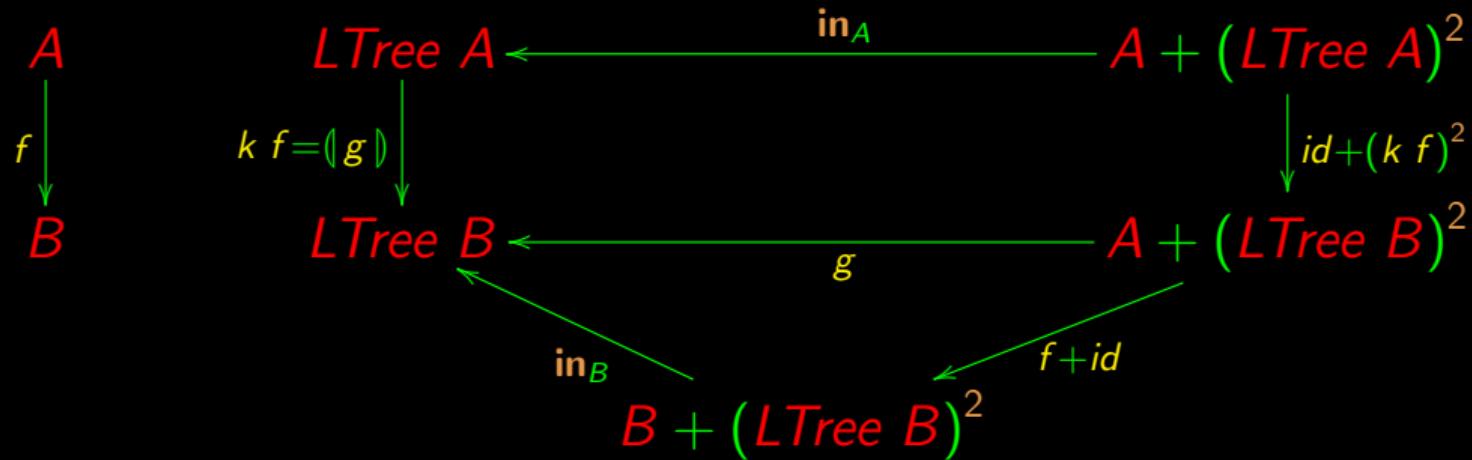


Inductive parametric types

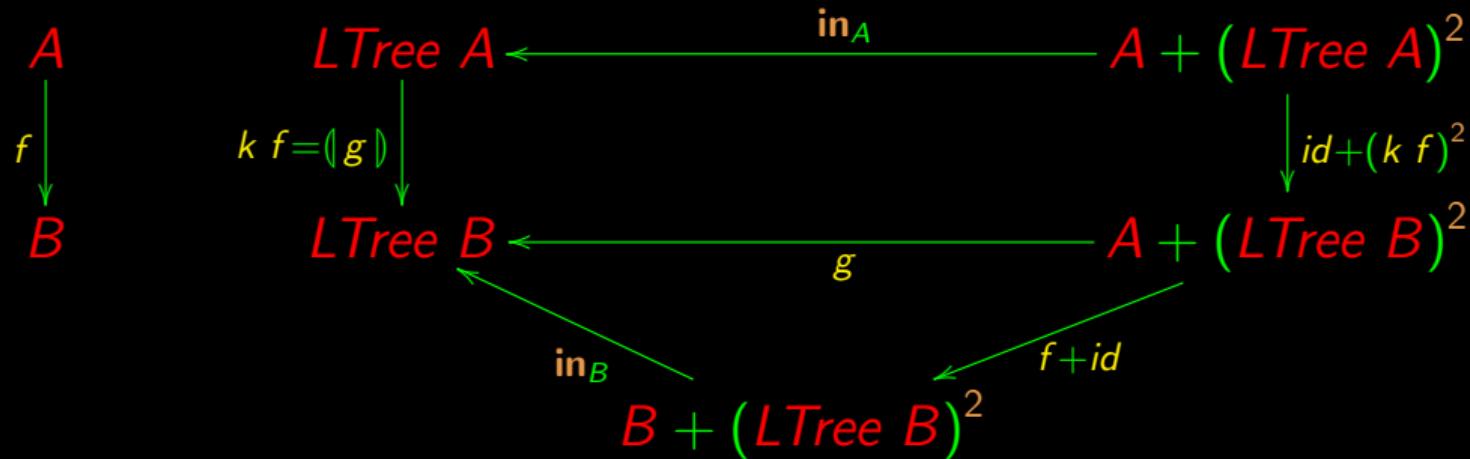


!!

Inductive parametric types



Inductive parametric types



$$k f = (\text{in} \cdot (f + id)) \circ$$

Follow-up: cp2223s12.pdf