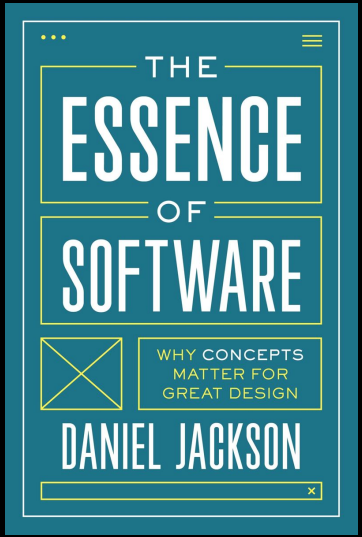
Cálculo de Programas

Class TO1

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

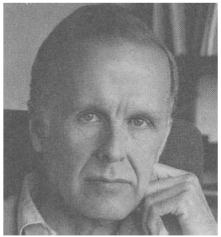
... small number
... well defined
... easy to understand

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 ...

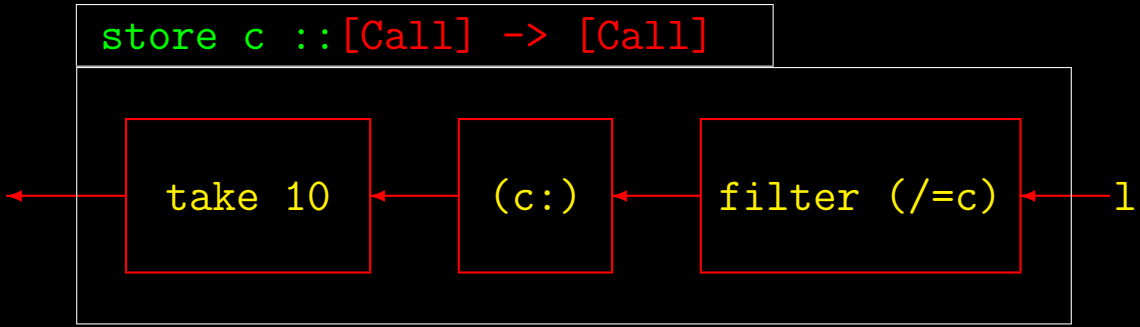
```
public void store10(string phoneNumber)
{
  System.Collections.ArrayList auxList =
      new System.Collections.ArrayList();
  auxList.Add(phoneNumber);
  auxList.AddRange(
      this.filteratmost9(phoneNumber) );
  this.callList = auxList;
}
```

+ filteratmost9 (next slide)

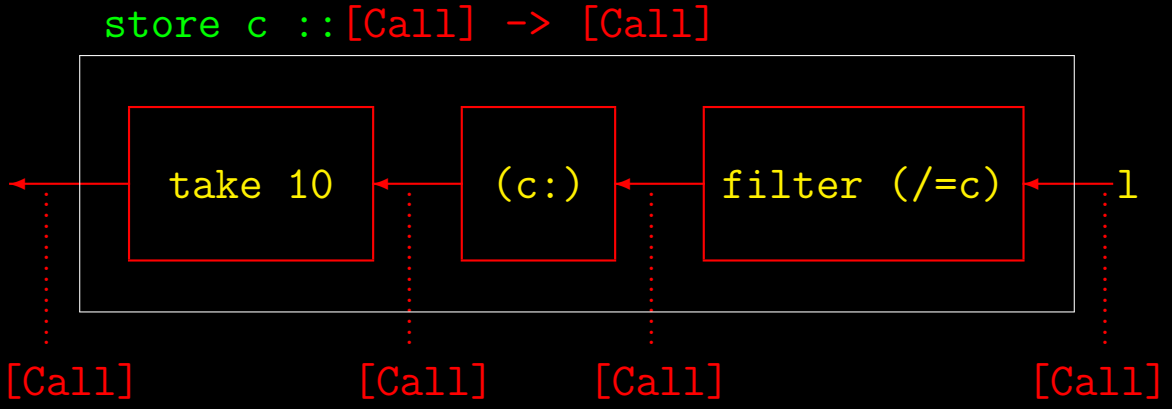
Compare with ...

```
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;
}
```

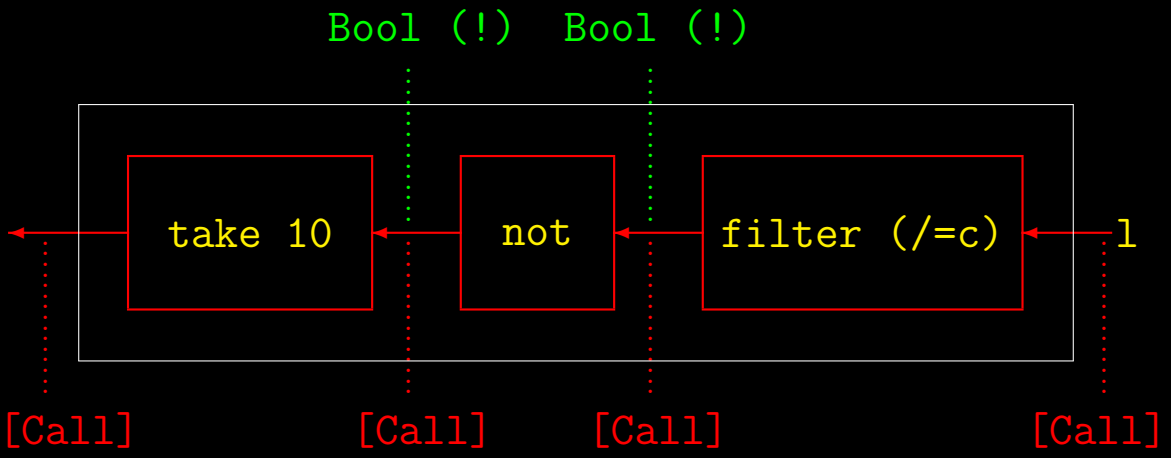
From a mobile phone manufacturer



From a mobile phone manufacturer

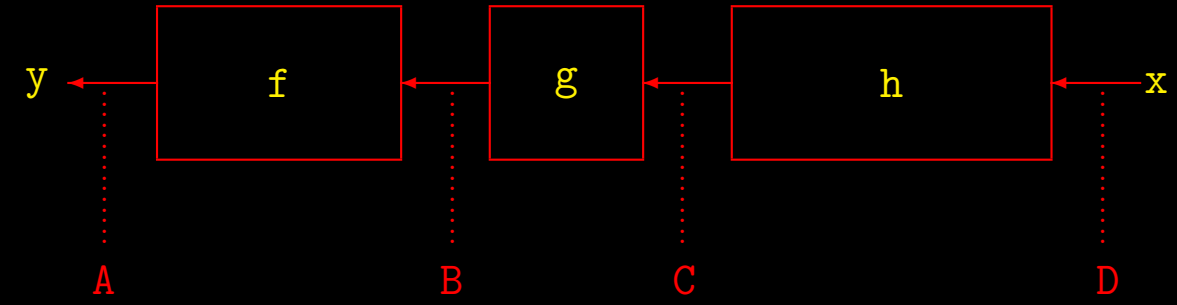


Uups!



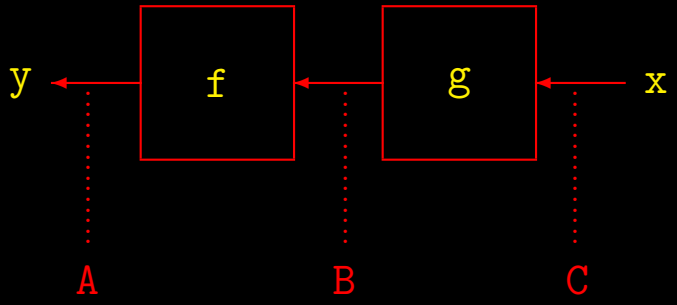
In general

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



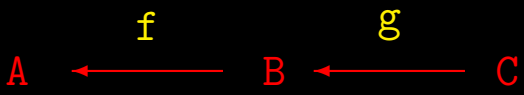
In general

```
y = f(g x)
```



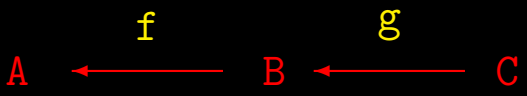
Simplification

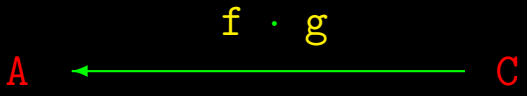
```
y = f(g x)
```



Composition

```
y = f(g x)
```






```
y = (f · 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

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

i.e.

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

Composition

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

i.e.

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

the same as

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

Composition

$$(f \cdot g) \cdot h = f \cdot (g \cdot h)$$
$$(a + b) + c = a + (b + c)$$

Composition

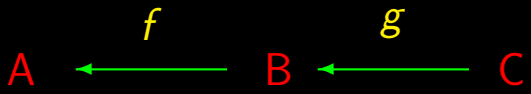
$$(f \cdot g) \cdot h = f \cdot (g \cdot h)$$
$$(a + b) + c = a + (b + c)$$

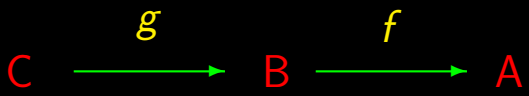
$$a + 0 = 0 + a = a$$

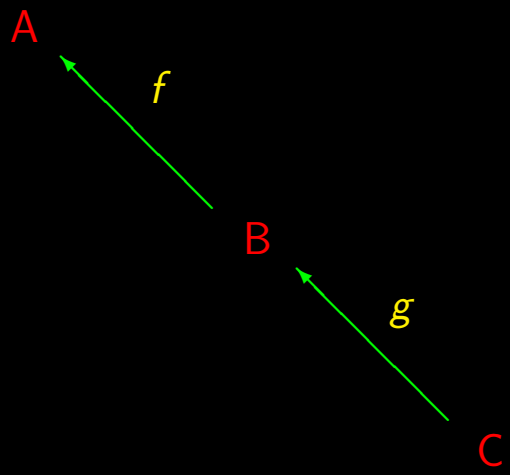
Composition

$$(f \cdot g) \cdot h = f \cdot (g \cdot h)$$
$$(a + b) + c = a + (b + c)$$

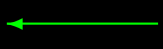
$$a + 0 = 0 + a = a$$
$$f \cdot ? = ? \cdot f = f$$


f
g
A
B
C


g
f
C
B
A


A
f
B
g
C

A

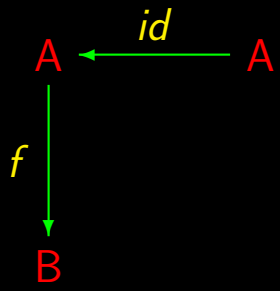

A


A
A

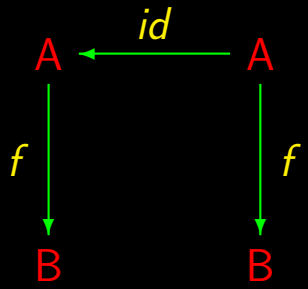

id
A
A

$$id\ a = a$$

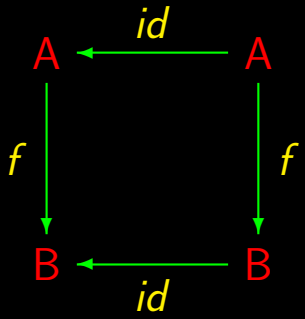
Identity



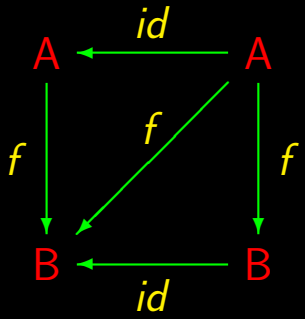
Identity



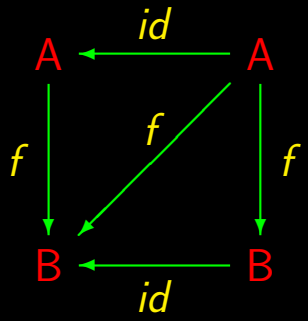
Identity



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
f
B
and
A
g
C

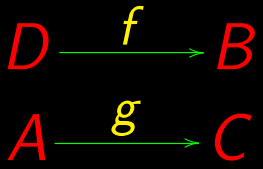
$C \xrightarrow{f} B$ and $A \xrightarrow{g} C$

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

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

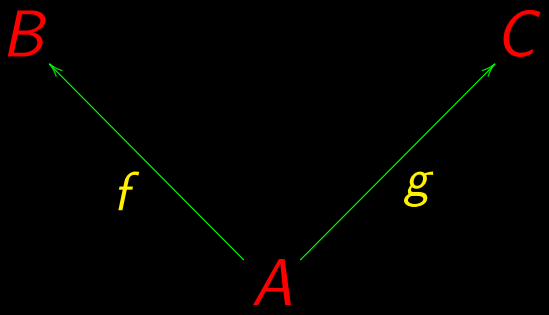
Composition: $A \xrightarrow{f \cdot g} B$

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

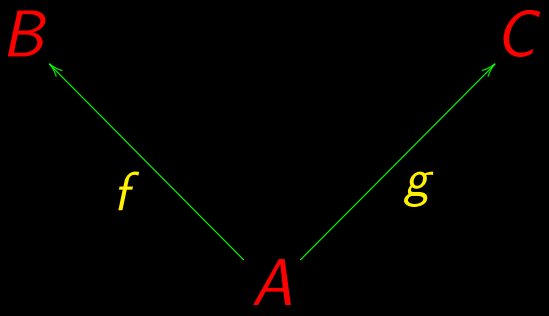

D
f
B
A
g
C

?

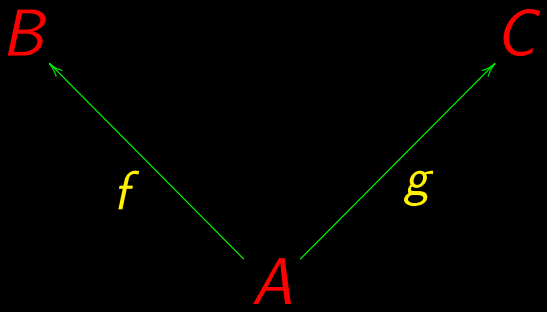
$D = A$?



$D = A$?



$f\ a \ldots g\ a$



$(f\ a, g\ a)$

Cartesian product

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

Cartesian product

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

$$f\ a \in B$$

Cartesian product

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

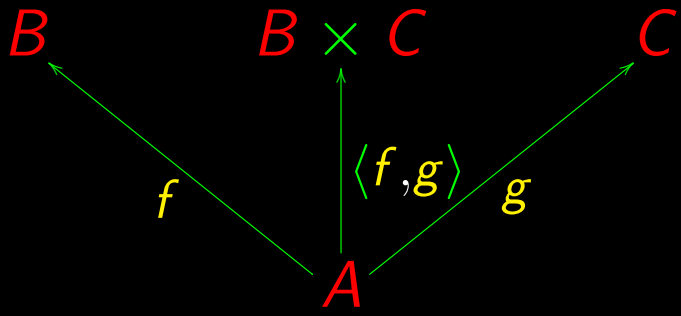
$$f\ a \in B$$
$$g\ a \in C$$

Cartesian product

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

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

“Split”



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

Product

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

Product

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

$$\pi_1 : A \times B \to A$$
$$\pi_1\,(a, b) = a$$

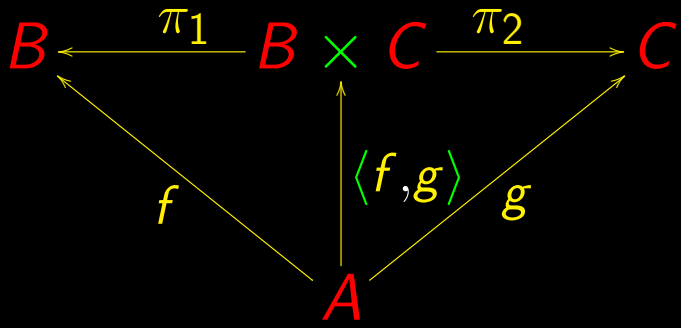
Product

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

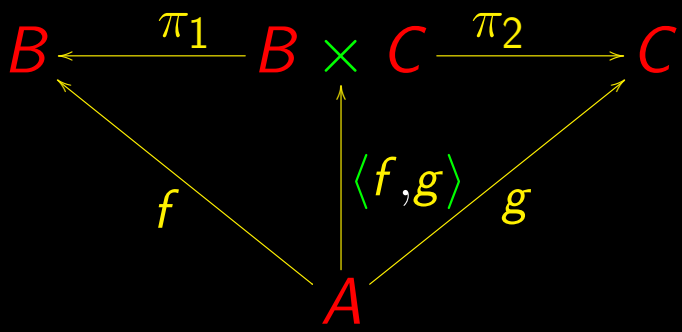
$$\pi_1 : A \times B \to A$$
$$\pi_1\,(a, b) = a$$

$$\pi_2 : A \times B \to B$$
$$\pi_2\,(a, b) = b$$

Product

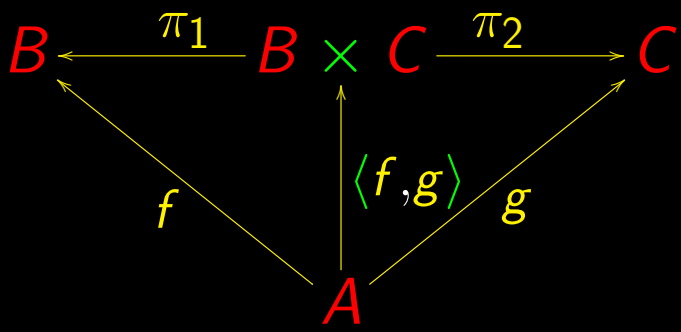


Product



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

Product



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

Product

$$\langle f, g \rangle$$

Product

$$\langle f, g \rangle$$

f and g in paralelo
f “split” g

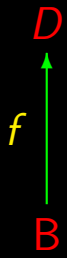
Product

$$\langle f, g \rangle$$

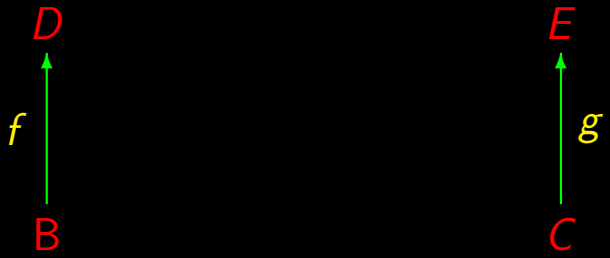
f and g in paralelo
f "split" g

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

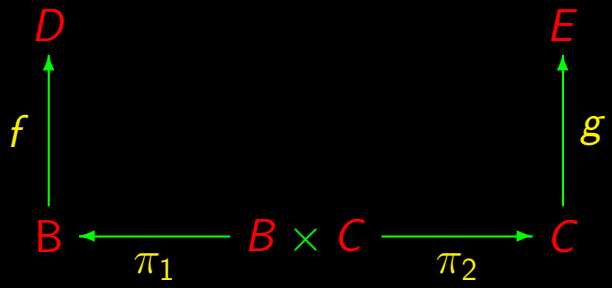
Product



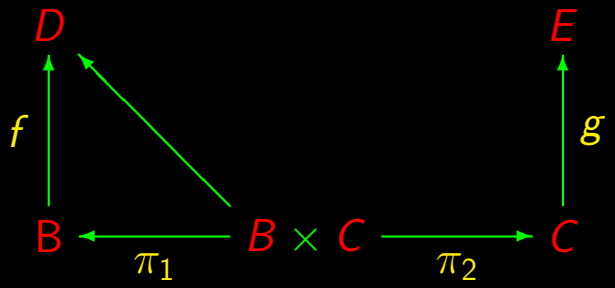
Product



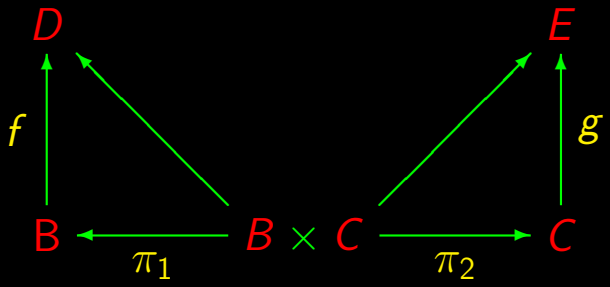
Product



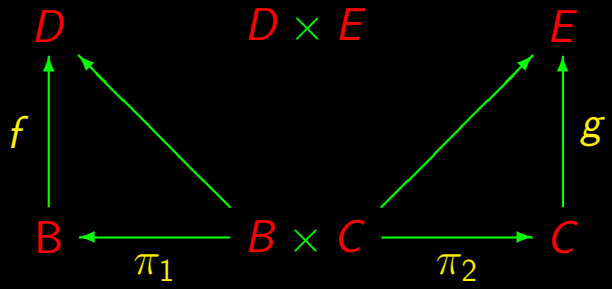
Product



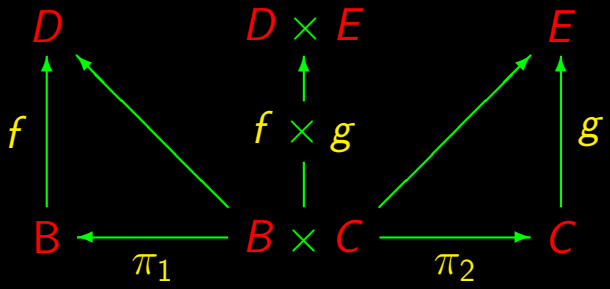
Product



Product

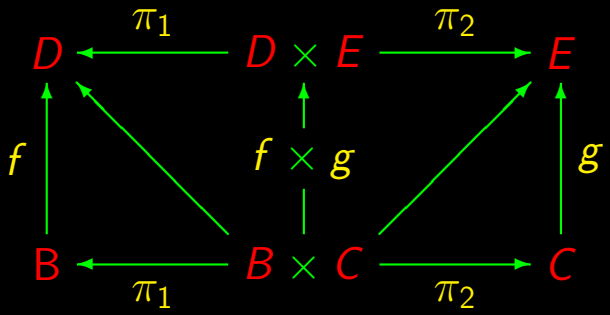


Product



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

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

$f \cdot g$ Sequential composition

$\langle f, g \rangle$

Summing up

$f \cdot g$	Sequential composition
$\langle f, g \rangle$	Parallel composition

Summing up

$f \cdot g$	Sequential composition
$\langle f, g \rangle$	Parallel composition (synchronous)

Summing up

$f \cdot g$	Sequential composition
$\langle f, g \rangle$	Parallel composition (synchronous)
$f \times g$	

Summing up

$f \cdot g$	Sequential composition
$\langle f, g \rangle$	Parallel composition (synchronous)
$f \times g$	Parallel composition

Summing up

$f \cdot g$	Sequential composition
$\langle f, g \rangle$	Parallel composition (synchronous)
$f \times g$	Parallel composition (asynchronous)

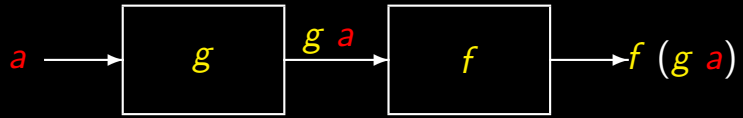
Summing up

$f \cdot g$	Sequential composition
$\langle f, g \rangle$	Parallel composition (synchronous)
$f \times g$	Parallel composition (asynchronous)

Compositional programming

Summing up

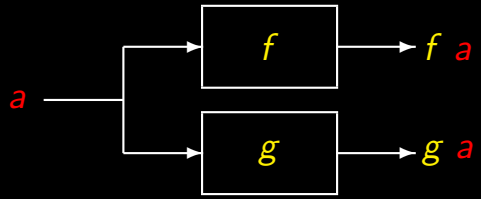
$$(f \cdot g)\ a = f\ (g\ a) \tag{2.6}$$



Function **composition**

Summing up

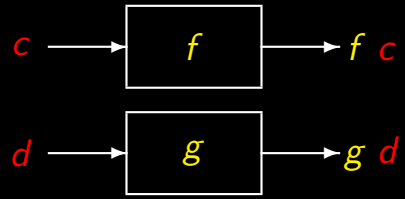
$$\langle f, g \rangle\ a = (f\ a, g\ a) \tag{2.20}$$



Functional “splits”

Summing up

$$f \times g = \langle f \cdot \pi_1, g \cdot \pi_2 \rangle \tag{2.24}$$

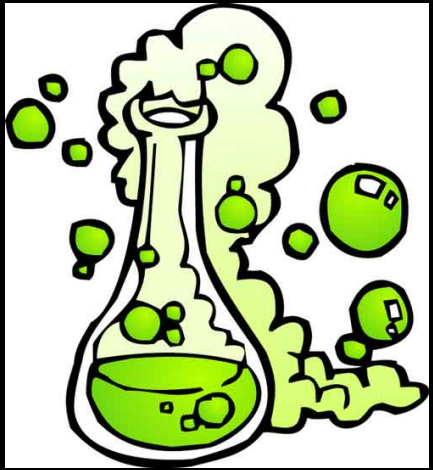


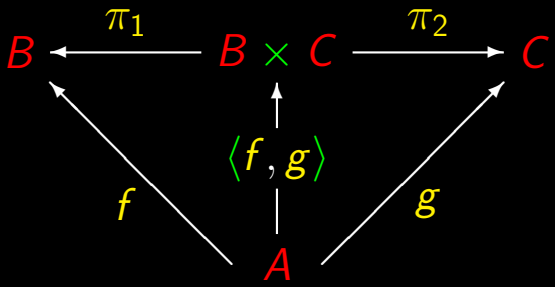
Functional **products**

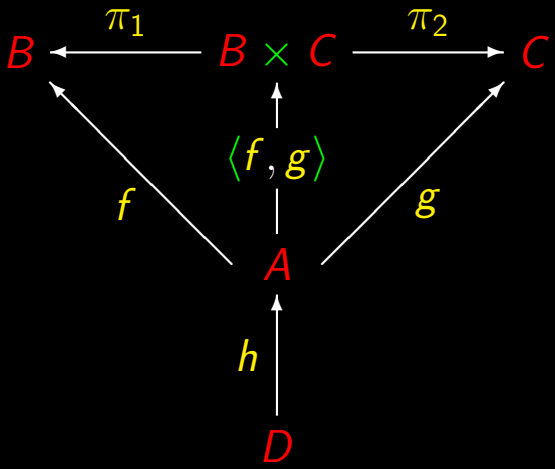
$f \cdot g$	Sequential composition
$\langle f, g \rangle$	Parallel composition (synchronous)
$f \times g$	Parallel composition (asynchronous)

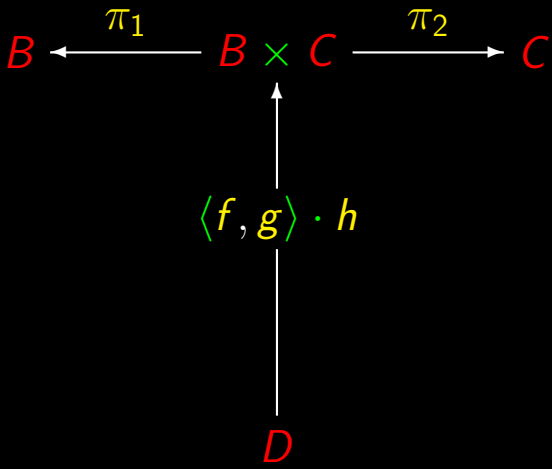
Compositional programming

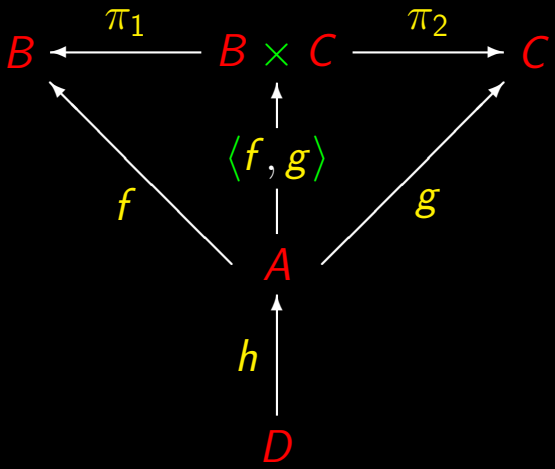
Calculus?

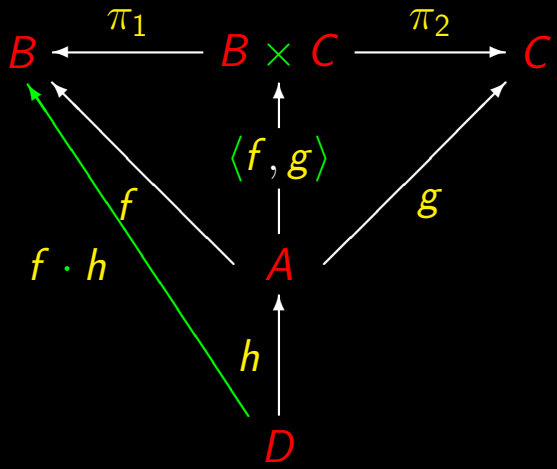


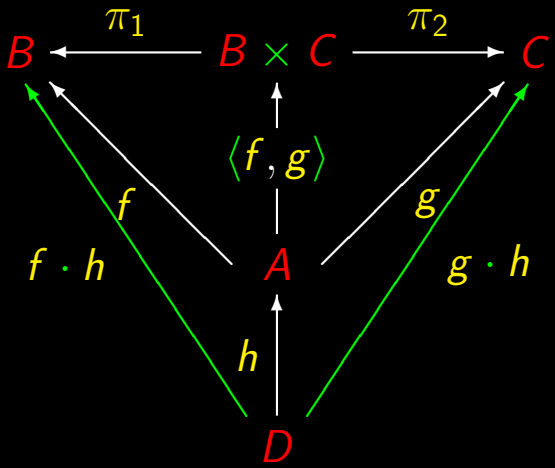

π_1
π_2
B
$B \times C$
C
$\langle f, g \rangle$
f
g
A

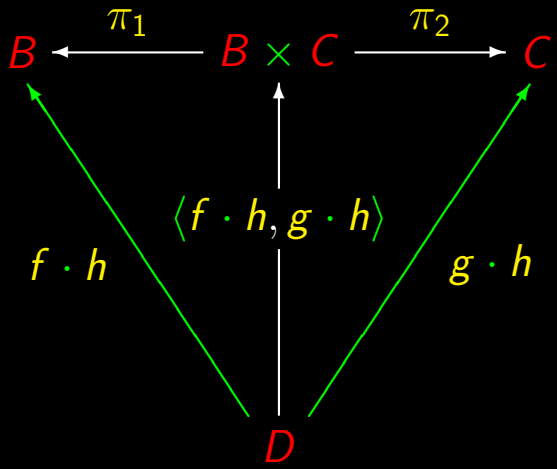

π_1
π_2
B
$B \times C$
C
$\langle f, g \rangle$
f
g
A
h
D

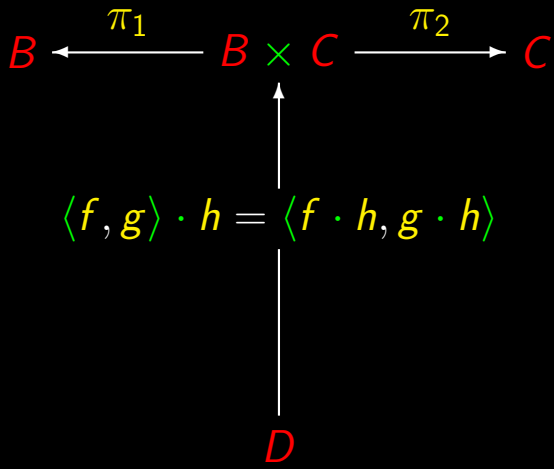

π_1
π_2
B
$B \times C$
C
$\langle f, g \rangle \cdot h$
D


π_1
π_2
B
$B \times C$
C
$\langle f, g \rangle$
f
g
A
h
D


π_1
B
$B \times C$
π_2
C
$\langle f, g \rangle$
f
g
$f \cdot h$
A
h
D

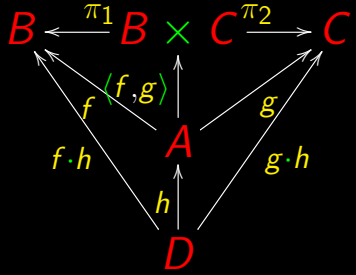

π_1
π_2
B
$B \times C$
C
$\langle f, g \rangle$
f
g
$f \cdot h$
A
$g \cdot h$
h
D

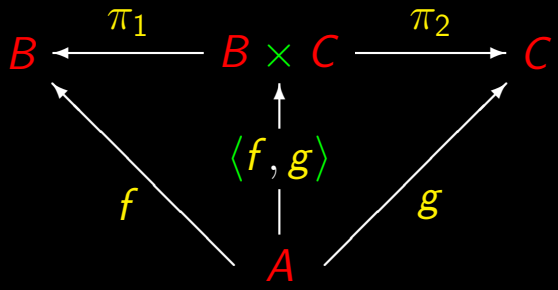

π_1
π_2
B
$B \times C$
C
$\langle f \cdot h, g \cdot h \rangle$
$f \cdot h$
$g \cdot h$
D

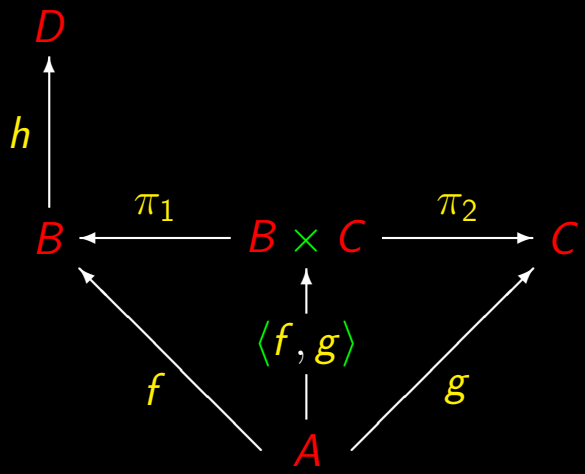

π_1
π_2
B
$B \times C$
C
$\langle f, g \rangle \cdot h = \langle f \cdot h, g \cdot h \rangle$
D

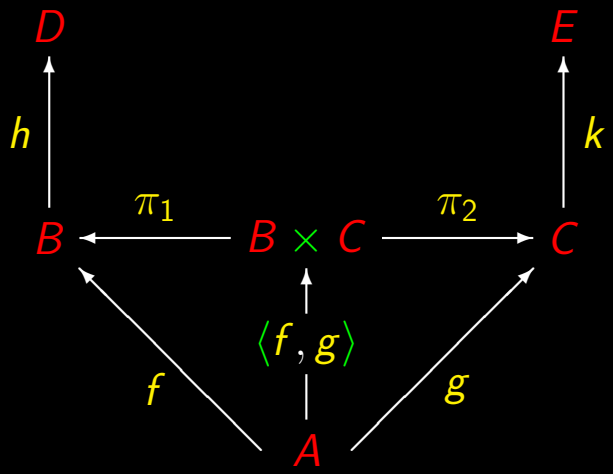
$\times$-Fusion

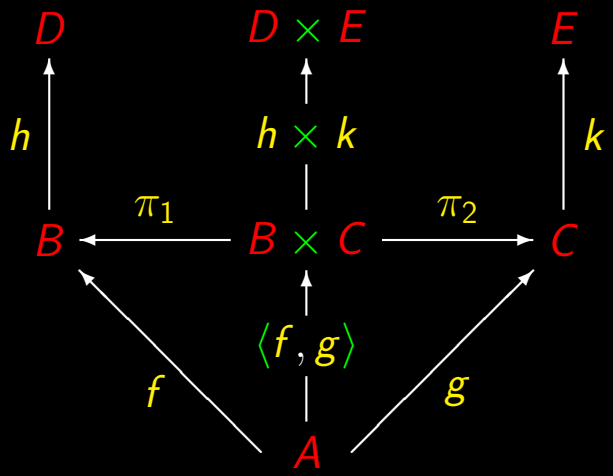
$$\langle f, g \rangle \cdot h = \langle f \cdot h, g \cdot h \rangle \tag{2.26}$$

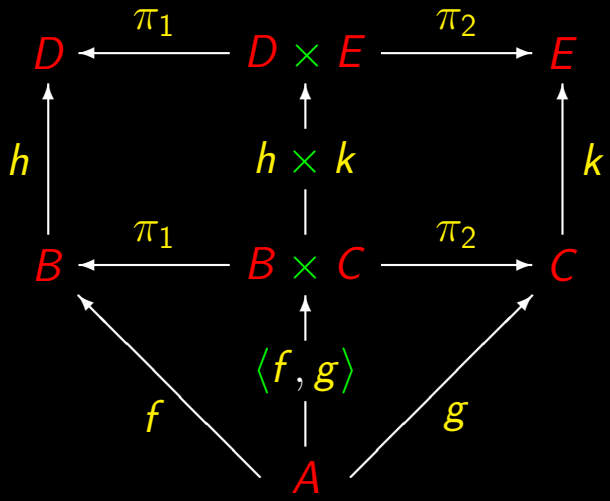


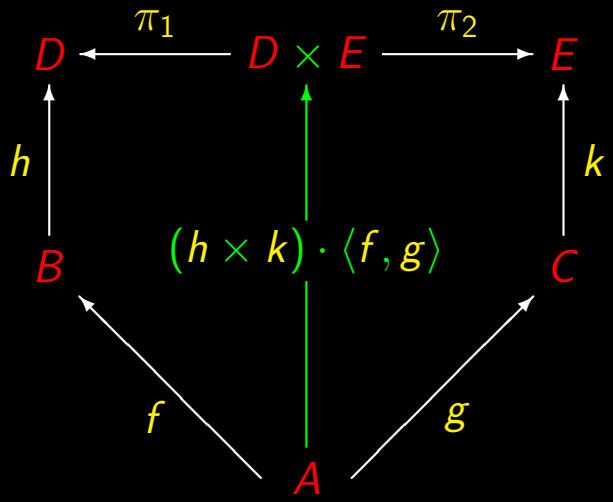

π_1
π_2
B
$B \times C$
C
$\langle f, g \rangle$
f
g
A

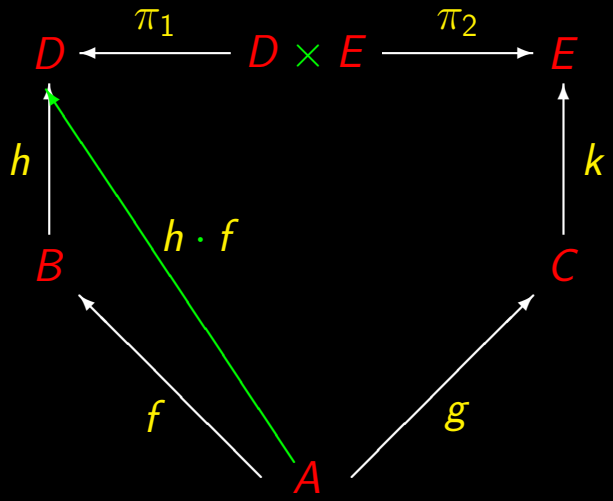

D
h
B
π1
B × C
π2
C
⟨f, g⟩
f
g
A

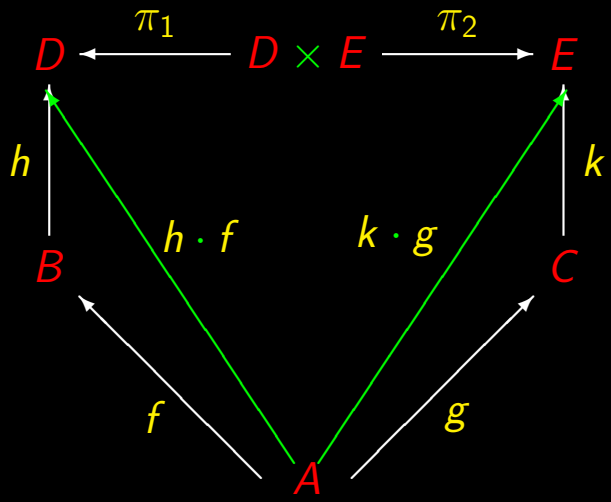

D
E
h
k
π1
π2
B
B × C
C
⟨f, g⟩
f
g
A

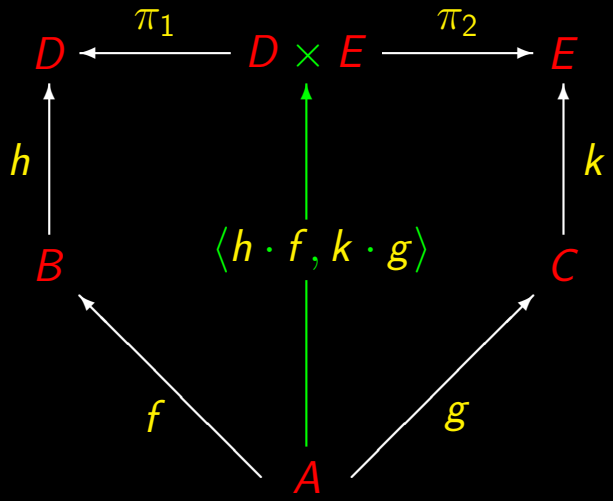

D
D × E
E
h
h × k
k
π1
π2
B
B × C
C
⟨f, g⟩
f
g
A

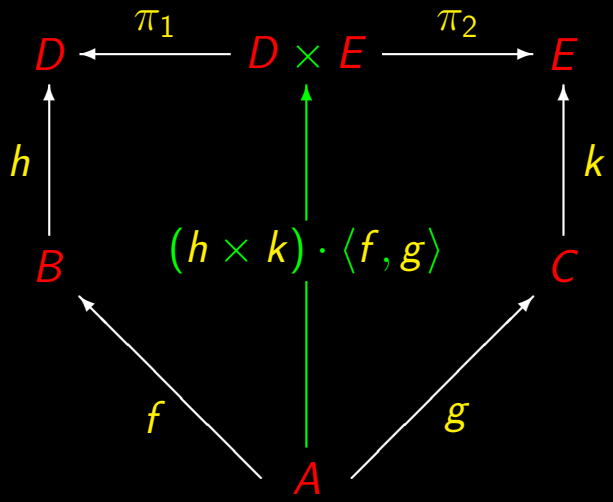

π_1
π_2
D
$D \times E$
E
h
$h \times k$
k
π_1
π_2
B
$B \times C$
C
$\langle f, g \rangle$
f
g
A


π_1
π_2
D
$D \times E$
E
h
k
$(h \times k) \cdot \langle f, g \rangle$
B
C
f
g
A


π_1
π_2
D
$D \times E$
E
h
k
$h \cdot f$
B
C
f
g
A

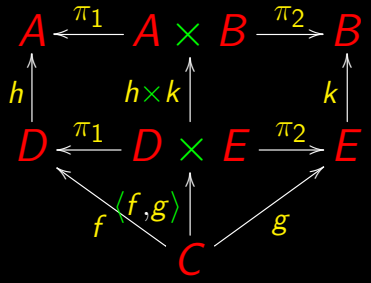

π_1
π_2
D
$D \times E$
E
h
k
$h \cdot f$
$k \cdot g$
B
C
f
g
A

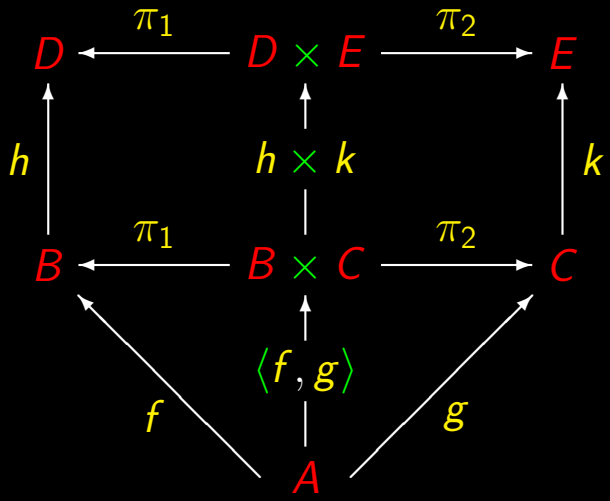

π_1
π_2
D
$D \times E$
E
h
k
$\langle h \cdot f, k \cdot g \rangle$
B
C
f
g
A

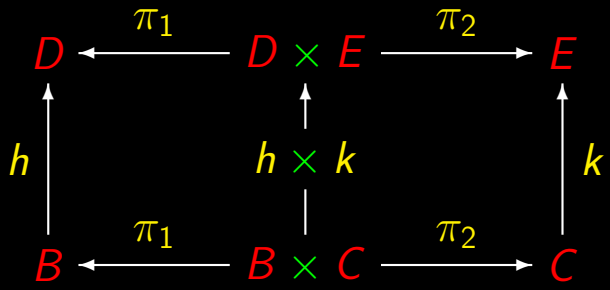

π_1
π_2
D
$D \times E$
E
h
k
$(h \times k) \cdot \langle f, g \rangle$
B
C
f
g
A

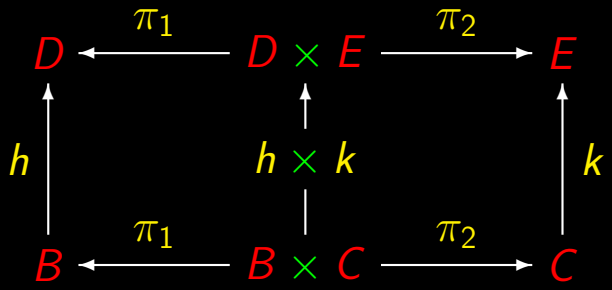
$\times$-Absorption

$$(h \times k) \cdot \langle f, g \rangle = \langle h \cdot f, k \cdot g \rangle \tag{2.27}$$

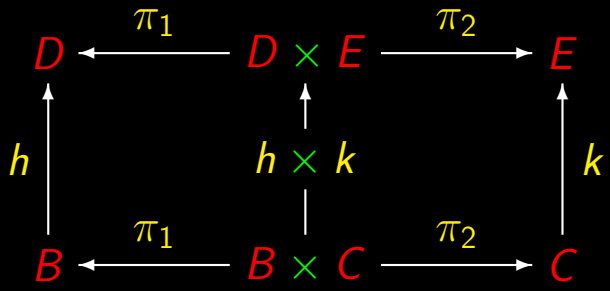



π_1
π_2
D
$D \times E$
E
h
$h \times k$
k
π_1
π_2
B
$B \times C$
C
$\langle f, g \rangle$
f
g
A


π_1
π_2
D
$D \times E$
E
h
$h \times k$
k
π_1
π_2
B
$B \times C$
C



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



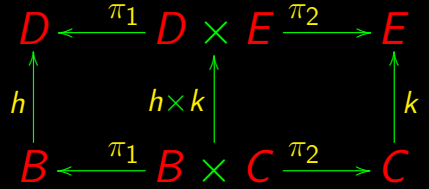
$$\pi_1 \cdot (h \times k) = h \cdot \pi_1$$

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

Natural-π_1, natural-π_2

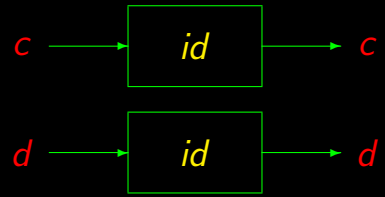
$$\pi_1 \cdot (h \times k) \;=\; h \cdot \pi_1 \tag{2.28}$$

$$\pi_2 \cdot (h \times k) = k \cdot \pi_2 \tag{2.29}$$



Functor-id-$\times$

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



Product of two identities is an identity.

$\times$-Functor

$$(f \times h) \cdot (g \times k) = (f \cdot g) \times (h \cdot k) \tag{2.30}$$

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

Two laws still missing

×-Reflexion $\langle \pi_1, \pi_2 \rangle = id$ (2.32)

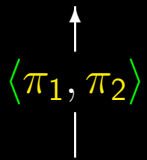
Two laws still missing

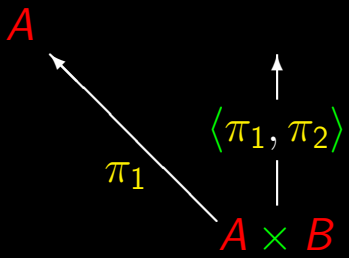
×-Reflexion

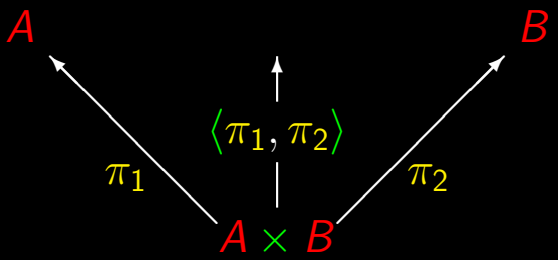
$$\langle \pi_1, \pi_2 \rangle = id \tag{2.32}$$

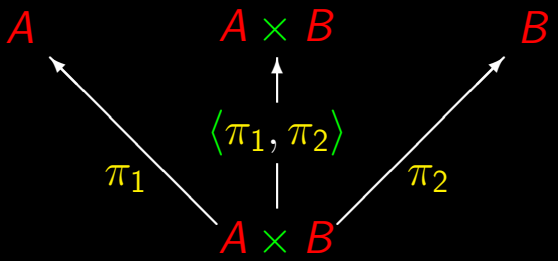
×-Eq

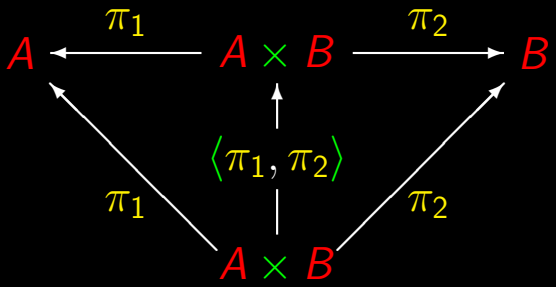
$$\langle i, j \rangle = \langle f, g \rangle \Leftrightarrow \begin{cases} i = f \\ j = g \end{cases} \tag{2.64}$$

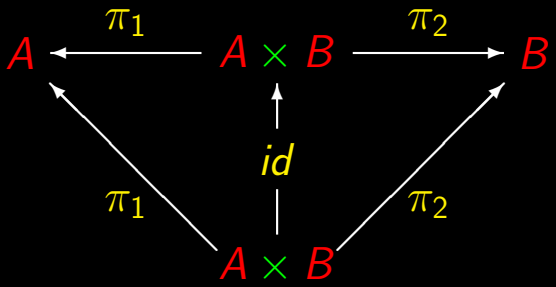

$\langle \pi_1, \pi_2 \rangle$


A
π_1
$\langle \pi_1, \pi_2 \rangle$
$A \times B$


A
B
⟨π1, π2⟩
π1
π2
A × B

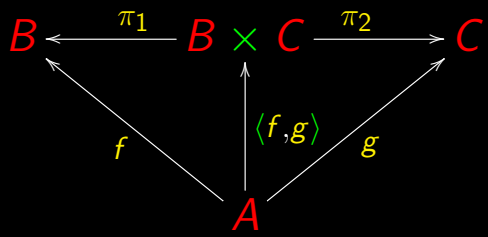

A
$A \times B$
B
$\langle \pi_1, \pi_2 \rangle$
π_1
π_2
$A \times B$


A
π_1
$A \times B$
π_2
B
$\langle \pi_1, \pi_2 \rangle$
π_1
π_2
$A \times B$


π_1
π_2
A
$A \times B$
B
id
π_1
π_2
$A \times B$

Finally the more important...

Recall $\times$-**cancellation**:



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

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

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

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

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

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

$\times$-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

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

"Such a solution, $k = \langle f, g \rangle$, is unique"

Equations!

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

Equations!

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

Equations!

Problem

Solve the equation

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

for f *and* g.

Equations!

Calculation

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

Equations!

Calculation

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

Equations!

Calculation

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

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

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

Equations!

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

Equations!

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

Substituting:

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

Equations!

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

Substituting:

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

$\times$-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

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

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

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

Calculation

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

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

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

$\Leftrightarrow$ $\{$ $\times$-cancellation $\}$

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

Calculation

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

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

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

$\Leftrightarrow$ $\{$ $\times$-cancellation $\}$

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

$\times$-Eq !

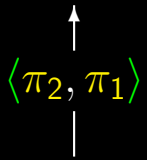
$$\langle \pi_1, \pi_2 \rangle = id$$

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

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

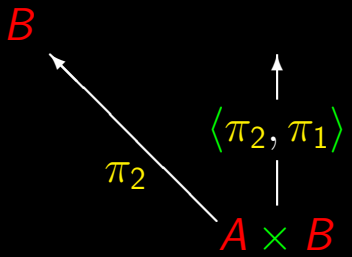
$\langle \pi_1, \pi_2 \rangle = id$

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



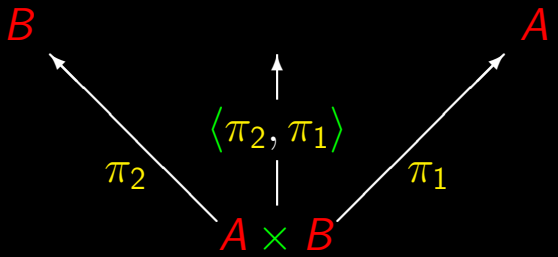
$\langle \pi_1, \pi_2 \rangle = id$

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



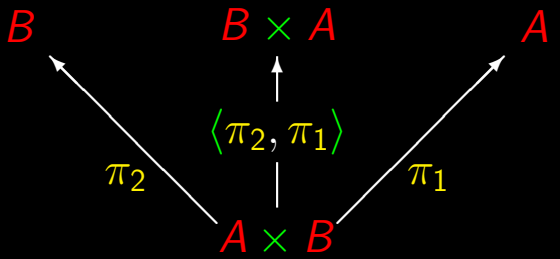
$\langle \pi_1, \pi_2 \rangle = id$

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



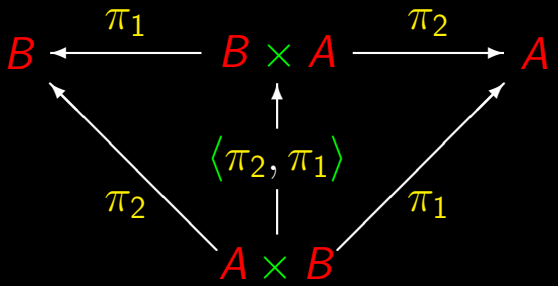
$\langle \pi_1, \pi_2 \rangle = id$

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



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

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



Problem

Solve

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

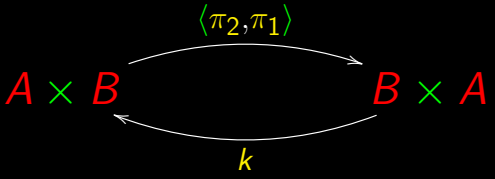
for k

Problem

Solve

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

for k



Calculation

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

Calculation

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

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

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

Calculation

$$
\begin{aligned}
& \langle \pi_2, \pi_1 \rangle \cdot k = id \\
\Leftrightarrow & \quad \{ \ \times\text{-fusion} \ \} \\
& \langle \pi_2 \cdot k, \pi_1 \cdot k \rangle = id \\
\Leftrightarrow & \quad \{ \ \times\text{-universal} \ \} \\
& \begin{cases} \pi_2 \cdot k = \pi_1 \\ \pi_1 \cdot k = \pi_2 \end{cases}
\end{aligned}
$$

Calculation

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

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

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

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

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

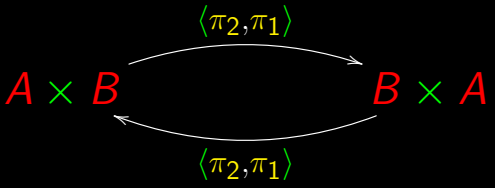
$$\Leftrightarrow \quad \{ \ \text{trivial} \ \}$$

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

Calculation

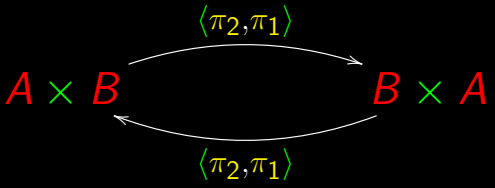
$$
\begin{aligned}
& \langle \pi_2, \pi_1 \rangle \cdot k = id \\
\Leftrightarrow & \quad \{ \ \times\text{-fusion} \ \} \\
& \langle \pi_2 \cdot k, \pi_1 \cdot k \rangle = id \\
\Leftrightarrow & \quad \{ \ \times\text{-universal} \ \} \\
& \begin{cases} \pi_2 \cdot k = \pi_1 \\ \pi_1 \cdot k = \pi_2 \end{cases}
\end{aligned}
\qquad
\begin{aligned}
\Leftrightarrow & \quad \{ \ \text{trivial} \ \} \\
& \begin{cases} \pi_1 \cdot k = \pi_2 \\ \pi_2 \cdot k = \pi_1 \end{cases} \\
\Leftrightarrow & \quad \{ \ \times\text{-universal} \ \} \\
& k = \langle \pi_2, \pi_1 \rangle
\end{aligned}
$$

Swap



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

Swap



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

$$swap \cdot swap = id$$

Até agora

$f \cdot g$ Sequential composition

Até agora

$f \cdot g$ Sequential composition

$\langle f, g \rangle$ Parallel composition

Até agora

$f \cdot g$ Sequential composition

$\langle f, g \rangle$ Parallel composition

Associativity

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

Até agora

$f \cdot g$ Sequential composition

$\langle f, g \rangle$ Parallel composition

Associativity

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

Associativity?

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

Até agora

$f \cdot g$ Sequential composition

$\langle f, g \rangle$ Parallel composition

Associativity

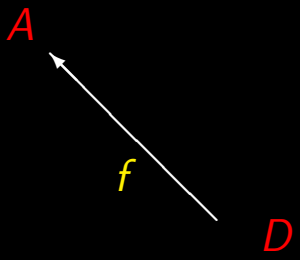
$$(f \cdot g) \cdot h = f \cdot (g \cdot h)$$

Associativity?

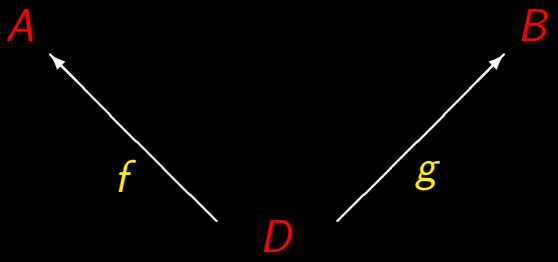
$$\langle \langle f, g \rangle, h \rangle = \langle f, \langle g, h \rangle \rangle ?$$

No! but...

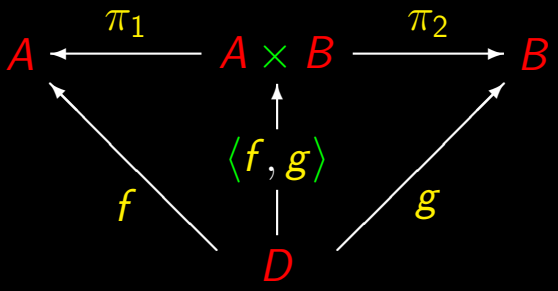
$\langle\langle f, g\rangle, h\rangle$



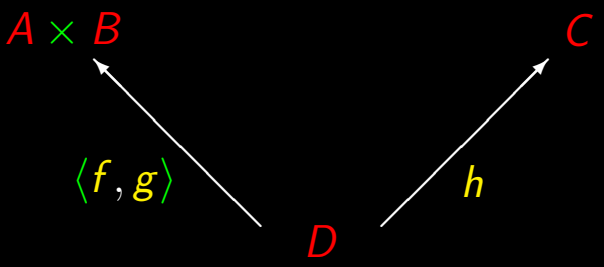
$\langle \langle f, g \rangle, h \rangle$



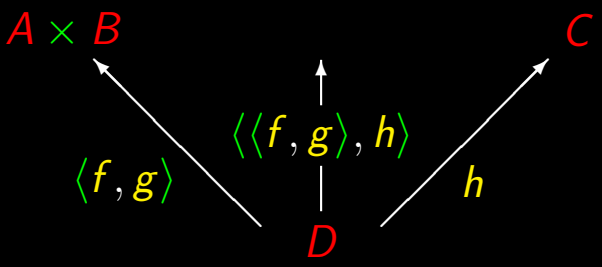
$\langle\langle f, g\rangle, h\rangle$



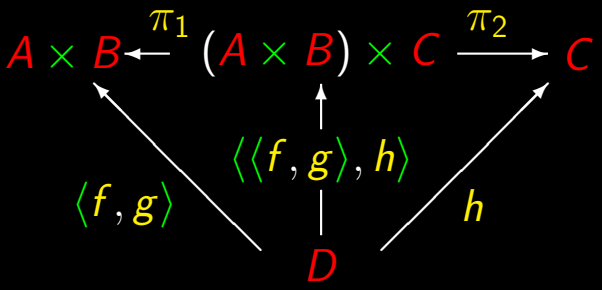
$\langle\langle f, g\rangle, h\rangle$



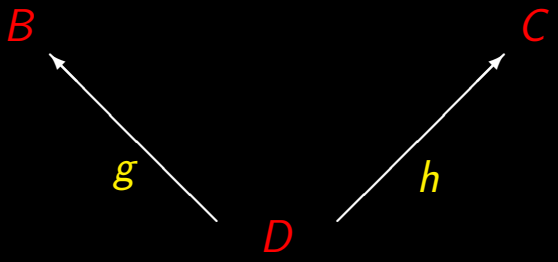
$\langle\langle f, g\rangle, h\rangle$



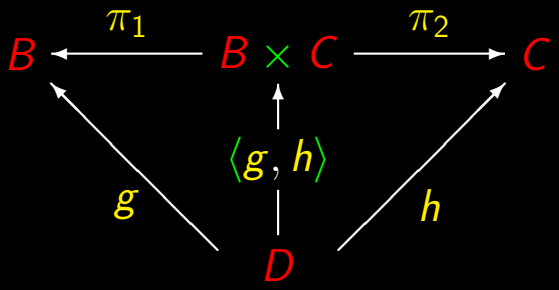
$\langle\langle f, g\rangle, h\rangle$



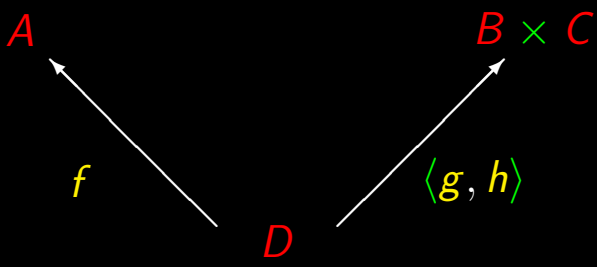
$\langle f, \langle g, h \rangle \rangle$



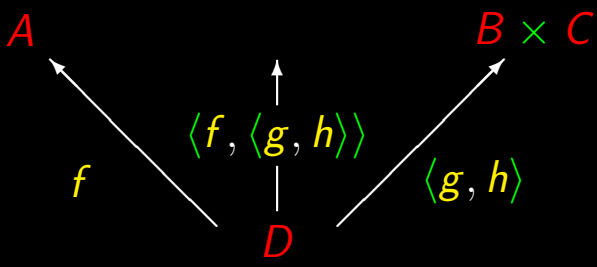
$\langle f, \langle g, h \rangle \rangle$



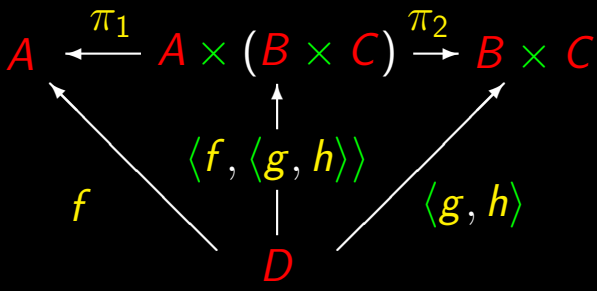
$\langle f, \langle g, h \rangle \rangle$



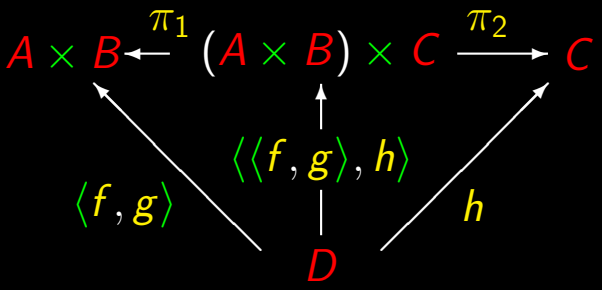
$\langle f, \langle g, h \rangle \rangle$

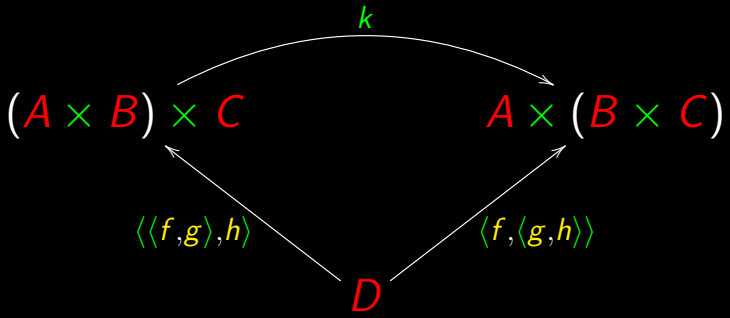


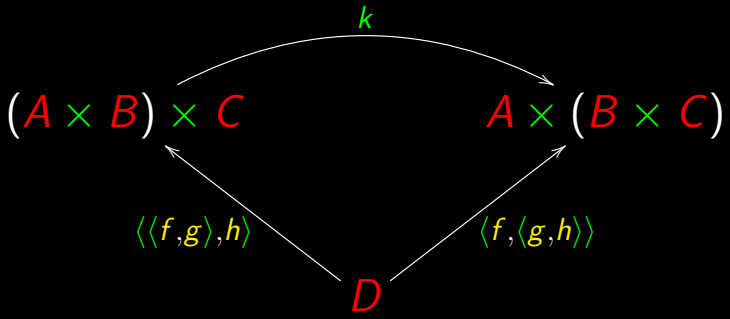
$\langle f, \langle g, h \rangle \rangle$



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




k
(A × B) × C
A × (B × C)
⟨⟨f,g⟩,h⟩
⟨f,⟨g,h⟩⟩
D



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Substitute solutions

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

Substitute solutions

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

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

Substitute solutions

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

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

Sligth improvement...

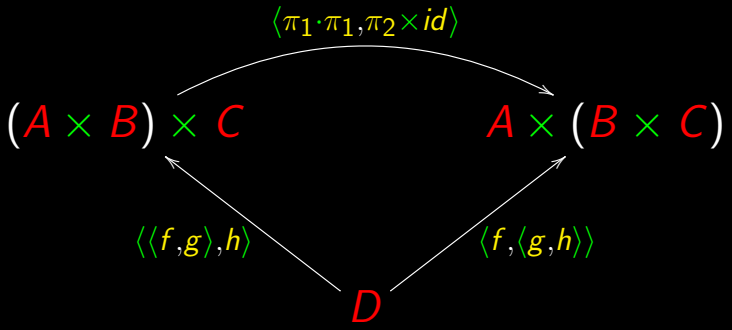
$$k = \langle \pi_1 \cdot \pi_1, \langle \pi_2 \cdot \pi_1, \pi_2 \rangle \rangle$$

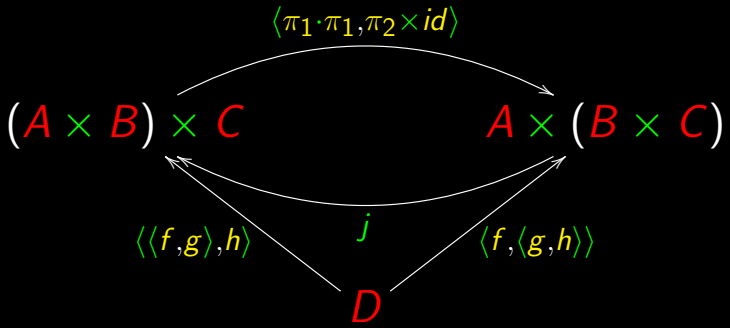
Sligth improvement...

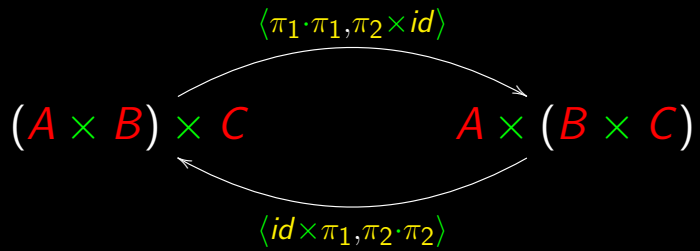
$$
\begin{aligned}
& k = \langle \pi_1 \cdot \pi_1, \langle \pi_2 \cdot \pi_1, \pi_2 \rangle \rangle \\
\Leftrightarrow \quad & \{\ \pi_2 = id \cdot \pi_2\ \} \\
& k = \langle \pi_1 \cdot \pi_1, \langle \pi_2 \cdot \pi_1, id \cdot \pi_2 \rangle \rangle
\end{aligned}
$$

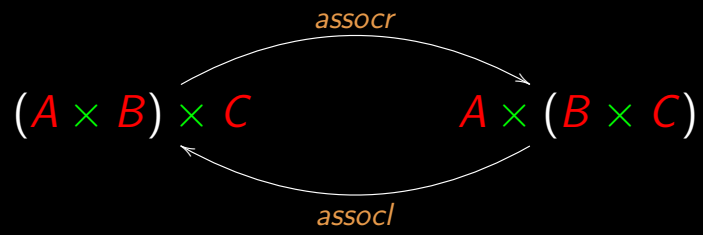
Sligth improvement...

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

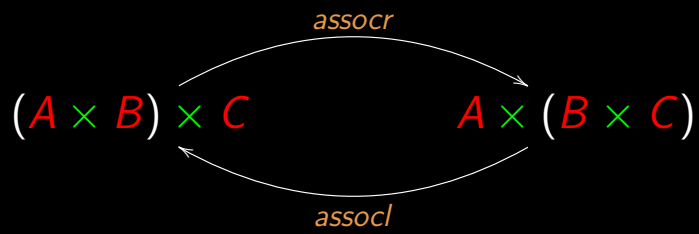

$\langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle$
$(A \times B) \times C$
$A \times (B \times C)$
$\langle \langle f, g \rangle, h \rangle$
$\langle f, \langle g, h \rangle \rangle$
D


$\langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle$
$(A \times B) \times C$
$A \times (B \times C)$
j
$\langle \langle f, g \rangle, h \rangle$
$\langle f, \langle g, h \rangle \rangle$
D

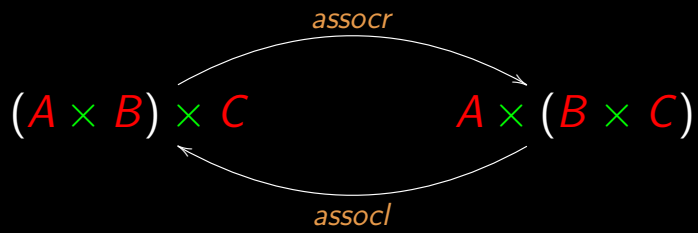

$\langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle$
$(A \times B) \times C$
$A \times (B \times C)$
$\langle id \times \pi_1, \pi_2 \cdot \pi_2 \rangle$



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



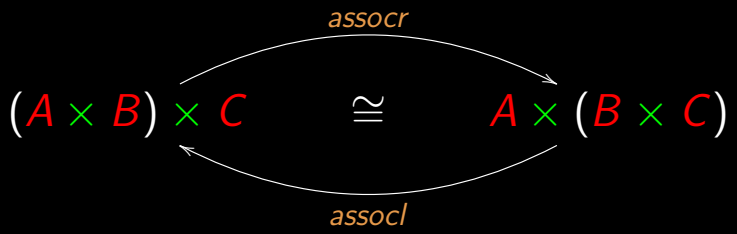
$$assocr \cdot assocl = id$$



$$assocr \cdot assocl = id$$

$$assocl \cdot assocr = id$$

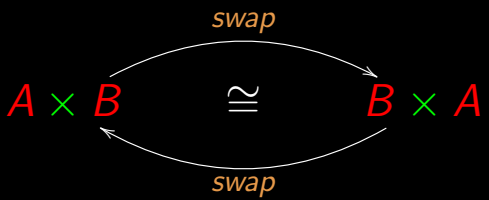
Isomorphism



$$assocr \cdot assocl = id$$

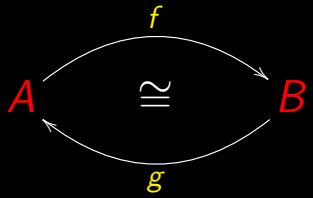
$$assocl \cdot assocr = id$$

Isomorphism



$$swap \cdot swap = id$$

Isomorphism



$$f \cdot g = id$$

$$g \cdot f = id$$

Isomorphism

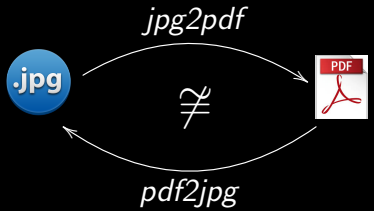
$\underbrace{iso}_{\text{the same}}$ $(\iota\sigma o)$

Isomorphism

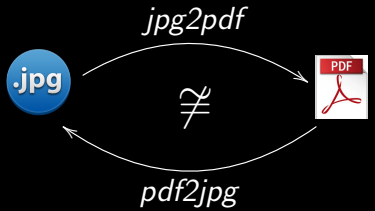
$$\underbrace{iso}_{\text{the same}}\ (\iota\sigma o) + \underbrace{morphism}_{\text{shape}}\ (\mu\text{the}\rho\phi\iota\sigma\mu o\zeta)$$

"Similar shape"

Practical problem!

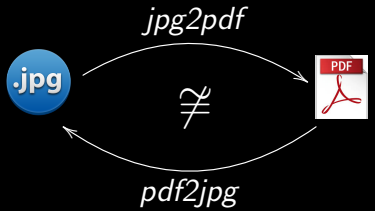


Practical problem!



$$jpg2pdf \cdot pdf2jpg \neq id$$
$$pdf2jpg \cdot jpg2pdf \neq id$$

Practical problem!



$$jpg2pdf \cdot pdf2jpg \neq id$$
$$pdf2jpg \cdot jpg2pdf \neq id$$

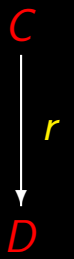
Format conversion

Need

Format conversion

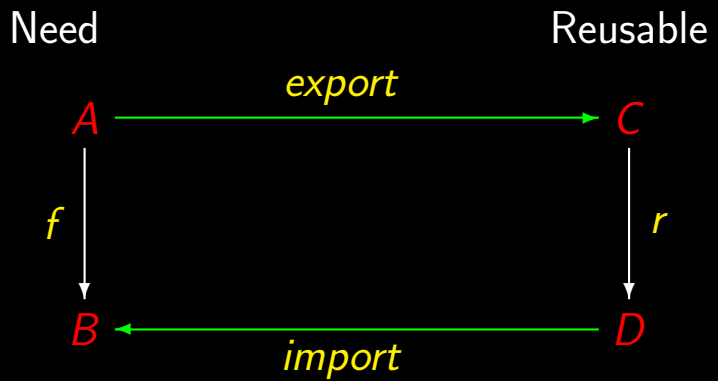
Need

Reusable

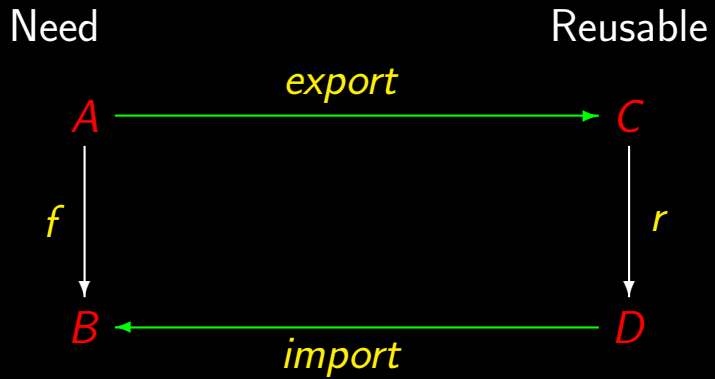


Format conversion

Format conversion

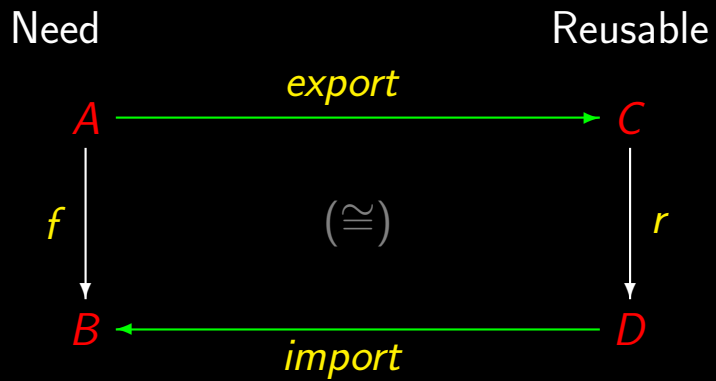


Format conversion



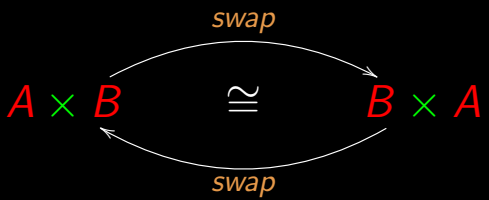
$$f = import \cdot r \cdot export$$

Format conversion



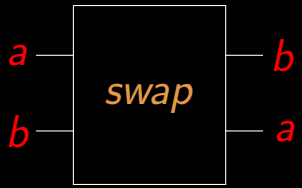
$$f = import \cdot r \cdot export$$

By the way — *swap*

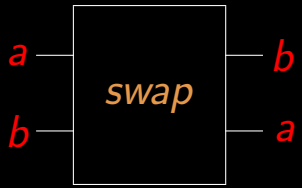


Isomorphisms are **reversible** computations

By the way — *swap*

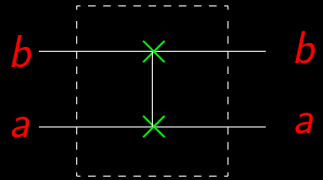


By the way — *swap*



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

By the way — *swap*



Cálculo de Programas

Class 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).

$$address : Iden \rightarrow Address$$
$$Iden = CC \cup NIF$$

Problem!

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

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

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

In general

We need to fix

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

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

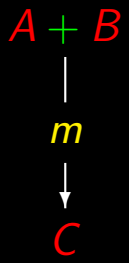
Altogether:

$$m : A + B \to C$$
$$i_1 : A \to A + B$$
$$i_2 : B \to A + B$$

Disjoint union

Altogether:

$$m : A + B \to C$$
$$i_1 : A \to A + B$$
$$i_2 : B \to A + B$$



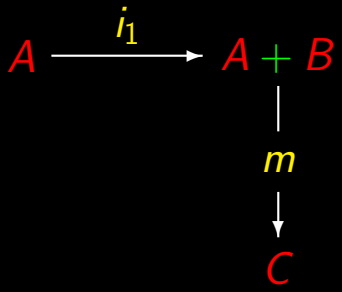
Disjoint union

Altogether:

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

$$i_1 : A \to A + B$$

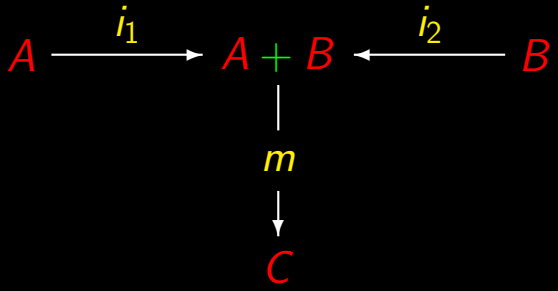
$$i_2 : B \to A + B$$



Disjoint union

Altogether:

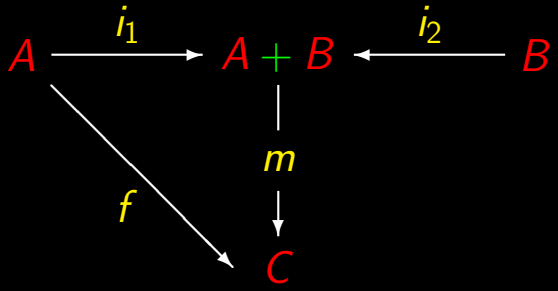
$$m : A + B \to C$$
$$i_1 : A \to A + B$$
$$i_2 : B \to A + B$$



Disjoint union

Altogether:

$$m : A + B \to C$$
$$i_1 : A \to A + B$$
$$i_2 : B \to A + B$$

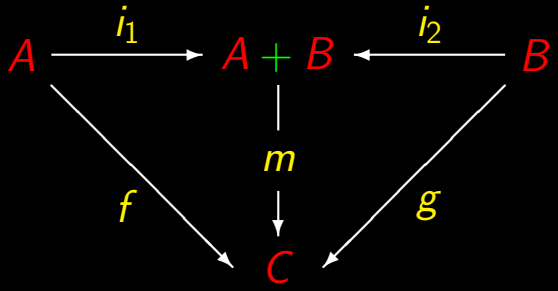


$$f = m \cdot i_1$$

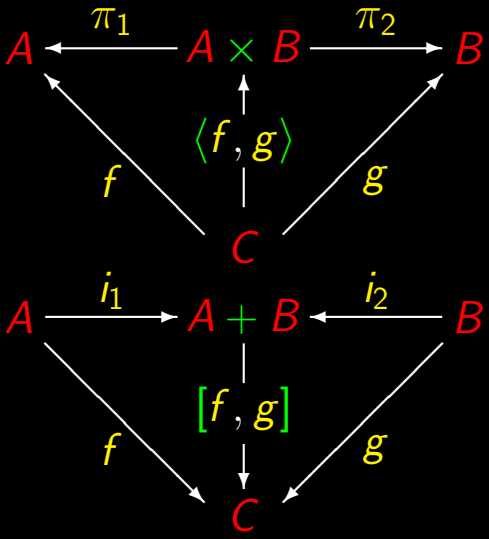
Disjoint union

Altogether:

$$m : A + B \to C$$
$$i_1 : A \to A + B$$
$$i_2 : B \to A + B$$



$$f = m \cdot i_1 \qquad g = m \cdot i_2$$


π_1
A
$A \times B$
π_2
B
$\langle f, g \rangle$
f
g
C
i_1
A
$A + B$
i_2
B
$[f, g]$
f
g
C

$+$-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

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

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

“Alternating” functions

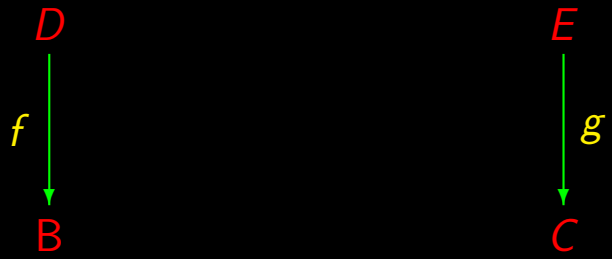
$$[f, g] : A + B \to C$$

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

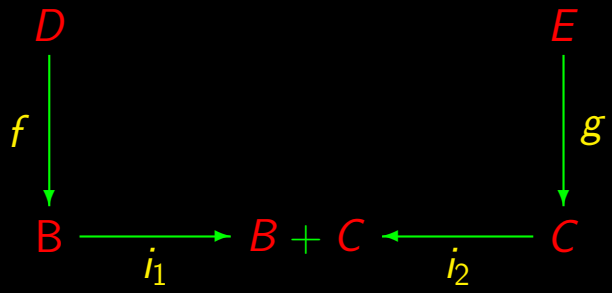
$f + g \quad ?$

Sum of two functions

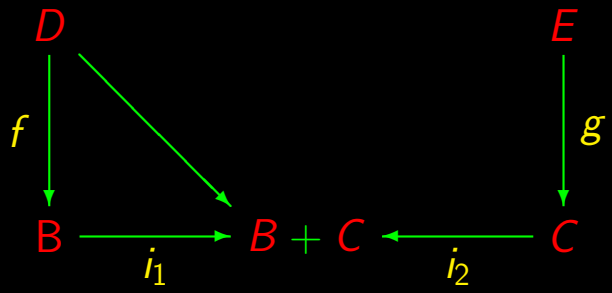
Sum of two functions



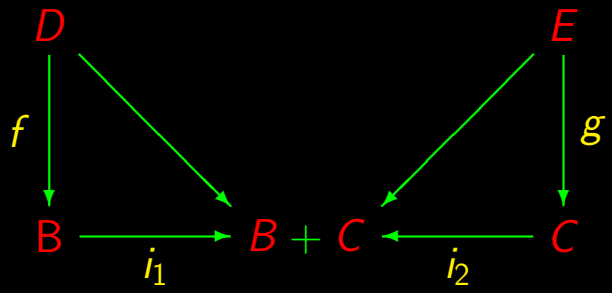
Sum of two functions



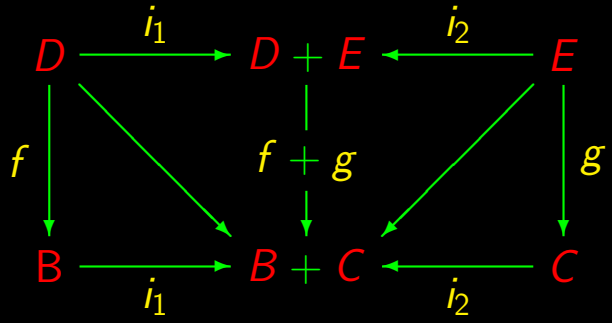
Sum of two functions



Sum of two functions



Sum of two functions



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

Coproduct laws

"Just reverse the arrows", cf.

$+$-Absorption $\qquad [h, k] \cdot (f + g) = [h \cdot f, k \cdot g] \qquad (2.43)$

Coproduct laws

"Just reverse the arrows", cf.

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

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

Coproduct laws

"Just reverse the arrows", cf.

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

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

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

Coproduct laws

$+$-Equality

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

Coproduct laws

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

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

Coproduct laws

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

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

$+$-Functor-id $$id + id = id \quad (2.45)$$

and so on

All these laws can be found in the reference sheet

(WWW $\rightarrow$ **Material**)

Summary

$f \cdot g$	Sequential composition
$\langle f, g \rangle$	Parallel composition
$f \times g$	Product composition

Summary

$f \cdot g$	Sequential composition
$\langle f, g \rangle$	Parallel composition
$f \times g$	Product composition
$[f, g]$	Alternative composition

Summary

$f \cdot g$	Sequential composition
$\langle f, g \rangle$	Parallel composition
$f \times g$	Product composition
$[f, g]$	Alternative composition
$f + g$	Coproduct (function sum)

Summary

$f \cdot g$	Sequential composition
$\langle f, g \rangle$	Parallel composition
$f \times g$	Product composition
$[f, g]$	Alternative composition
$f + g$	Coproduct (function sum)

Compositional programming 

What about...

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

What about...

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

...?

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

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...

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

...?

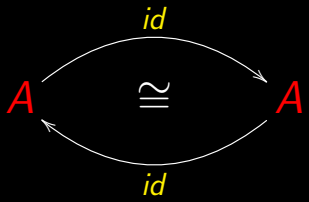
$$A \times (B + C) \stackrel{?}{\cong} A \times B + A \times C$$

Moreover, any "0" such that

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

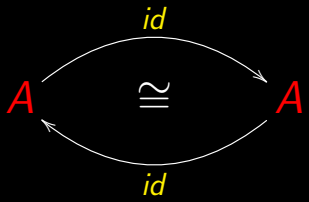
Recall

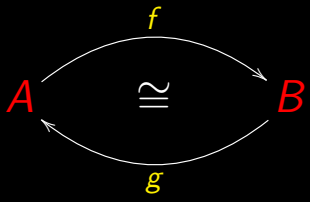
$$\begin{cases} id : A \to A \\ id\ a = a \end{cases}$$



Recall

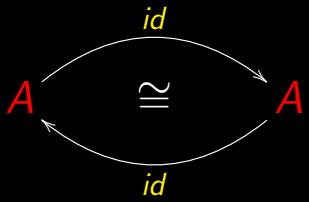
$$\begin{cases} id : A \to A \\ id\ a = a \end{cases}$$

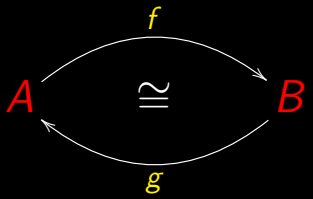




Recall

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





$$f \cdot g = id$$

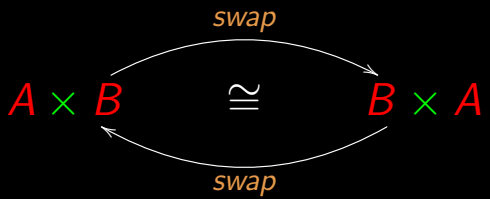
$$g \cdot f = id$$

Recall

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

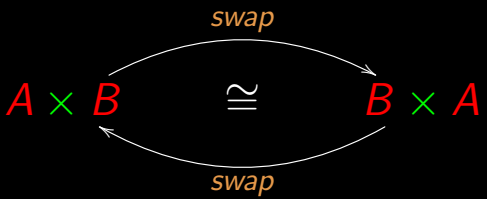
Recall

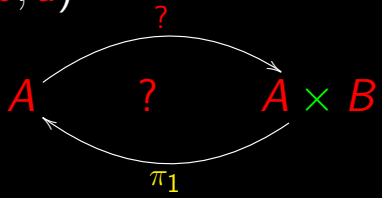
$$\begin{cases} swap : A \times B \to B \times A \\ swap\ (a, b) = (b, a) \end{cases}$$



Recall

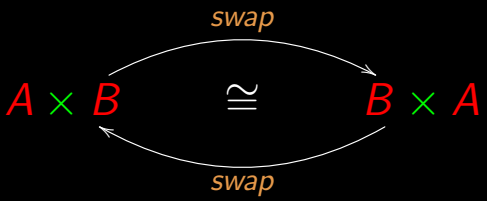
$$\begin{cases} swap : A \times B \to B \times A \\ swap\ (a, b) = (b, a) \end{cases}$$

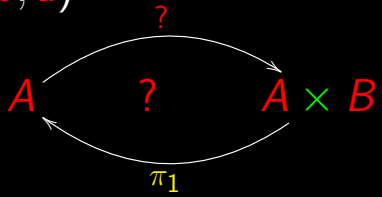




Recall

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

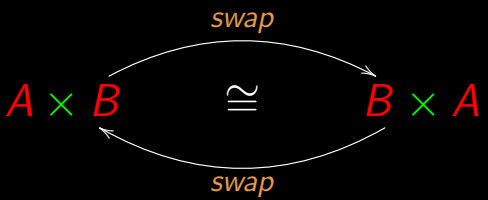


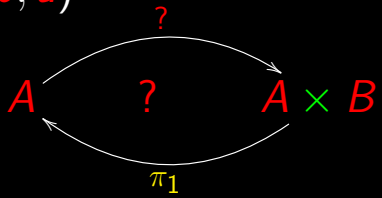


$$\pi_1 \cdot \langle id, h \rangle = id$$

Recall

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

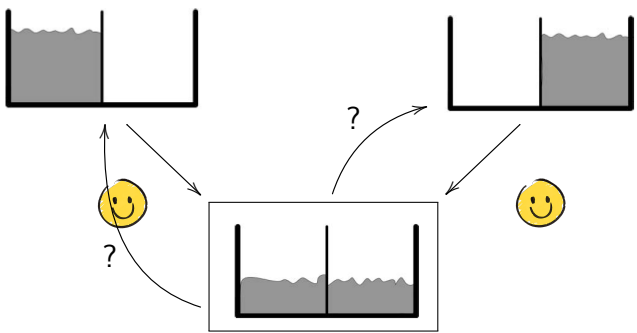




$$\pi_1 \cdot \langle id, h \rangle = id$$

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

Irreversibility…



Rolf Landauer (1927–99)

“Information is physical”

Even worse than π_1...

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

Even worse than π_1...

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

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

Even worse than π_1...

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

$$\mathit{one}\ 10 = 1$$
$$\mathit{one}\ \texttt{"string"} = 1$$

Even worse than π_1...

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

$$\mathit{one}\ 10 = 1$$
$$\mathit{one}\ \texttt{"string"} = 1$$
$$\mathit{zero}\ \texttt{"string"} = 0$$

Even worse than π_1...

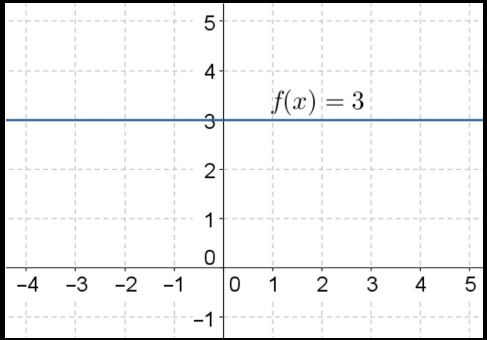
$$\begin{cases} \mathit{zero}\ x = 0 \\ \mathit{one}\ x = 1 \end{cases}$$

$\mathit{one}\ 10 = 1$

$\mathit{one}\ \texttt{"string"} = 1$

$\mathit{zero}\ \texttt{"string"} = 0$

$f\ x = 3$



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

$3 \in \mathbb{N}_0$

$f\ x = 3$

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

$b_0 \in B$

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

Constant functions

$3 \in \mathbb{N}_0$

$f\ x = 3$

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

$b_0 \in B$

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

Constant functions

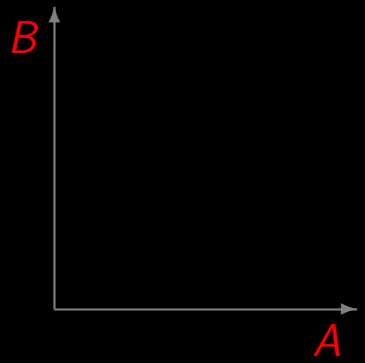
$3 \in \mathbb{N}_0$

$f\ x = 3$

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

$b_0 \in B$

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



Constant functions

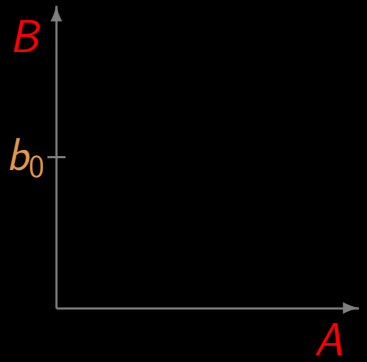
$3 \in \mathbb{N}_0$

$f\ x = 3$

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

$b_0 \in B$

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



Constant functions

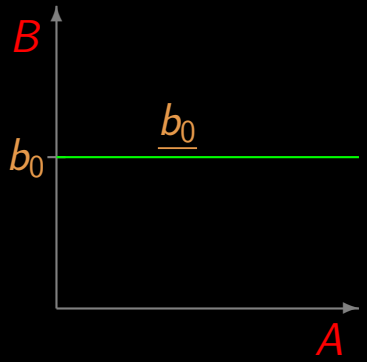
$3 \in \mathbb{N}_0$

$f\ x = 3$

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

$b_0 \in B$

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



Constant functions

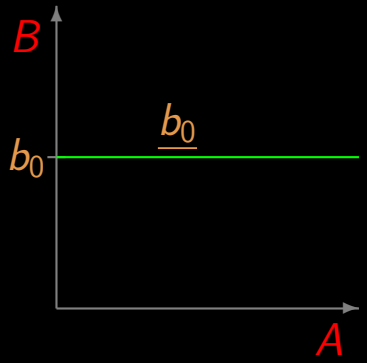
$3 \in \mathbb{N}_0$

$f\ x = 3$

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

$b_0 \in B$

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



(**Haskell**: `const` b_0)

Properties

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

Properties

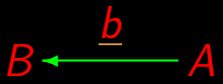
$$\underline{b} \cdot f = \underline{b}$$

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

Properties

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

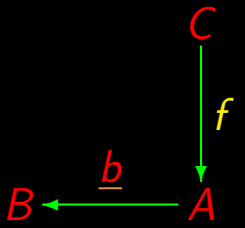
$$g \cdot \underline{b} = \underline{g\ b}$$



Properties

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

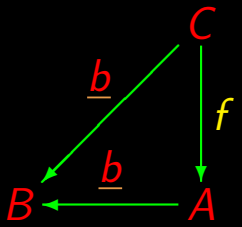
$$g \cdot \underline{b} = \underline{g\ b}$$



Properties

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

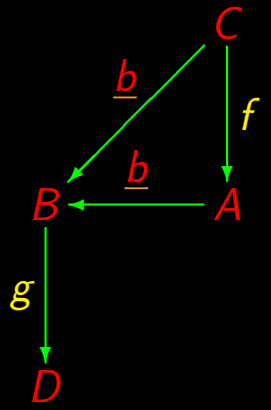
$$g \cdot \underline{b} = \underline{g\ b}$$



Properties

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

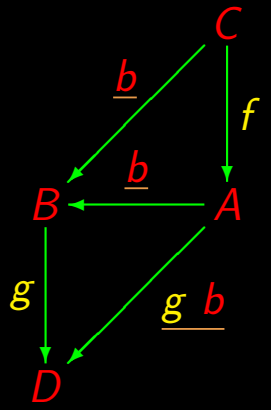
$$g \cdot \underline{b} = \underline{g\ b}$$



Properties

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

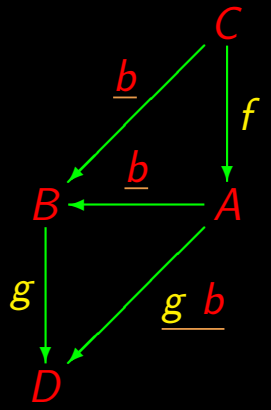
$$g \cdot \underline{b} = \underline{g\ b}$$



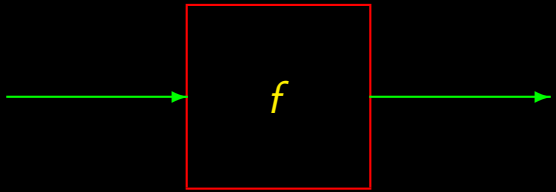
Properties

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

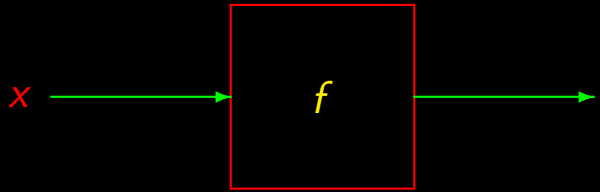
$$g \cdot \underline{b} = \underline{g\ b}$$



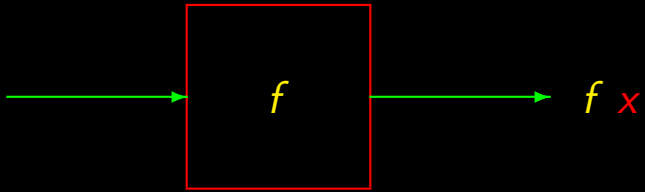
Data flow



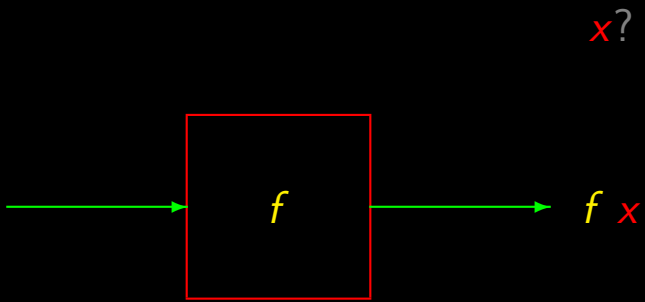
Data flow



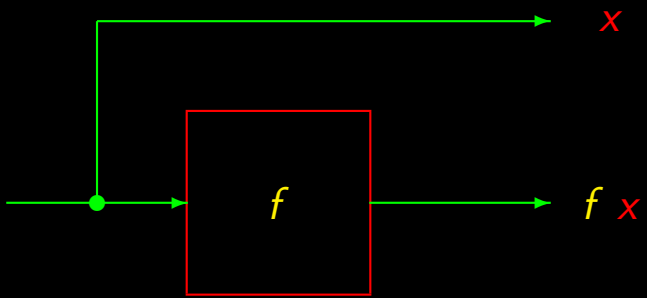
Data flow



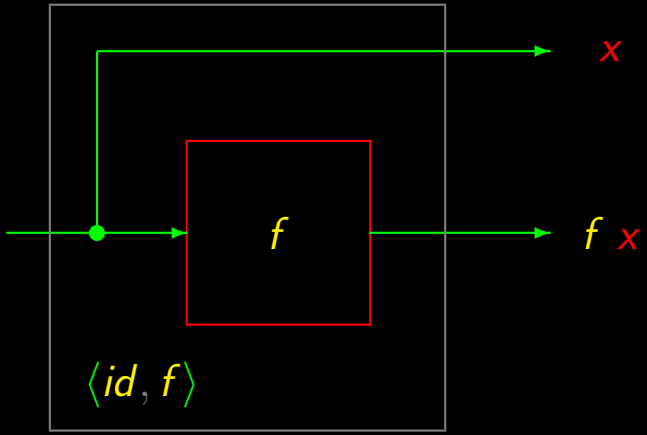
Data flow



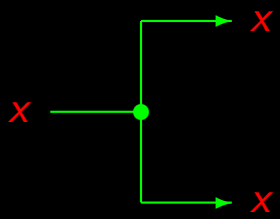
Data flow



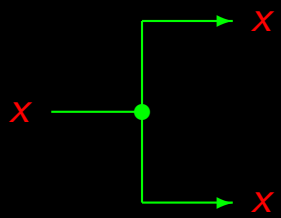
Data flow



Data duplication

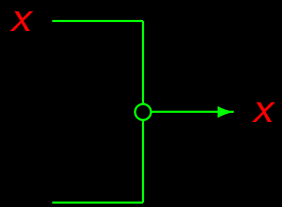


Data duplication



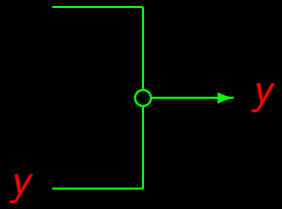
$$dup = \langle id, id \rangle$$

Data joins



$$join = [id, id]$$

Data joins



$$join = [id, id]$$

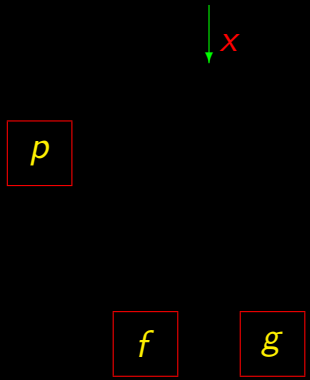
Cálculo de Programas

Class T04

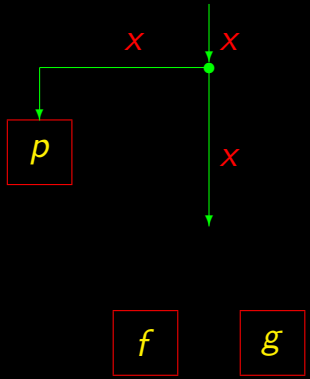
if-then-else

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

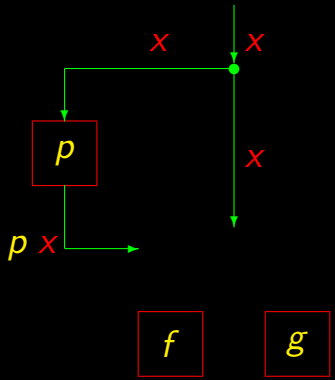
if-then-else



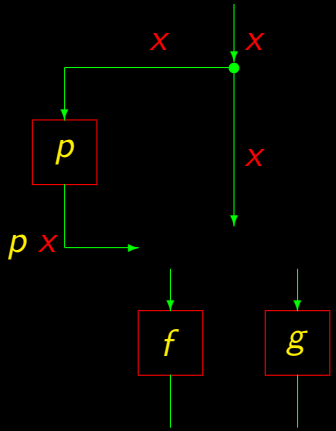
if-then-else



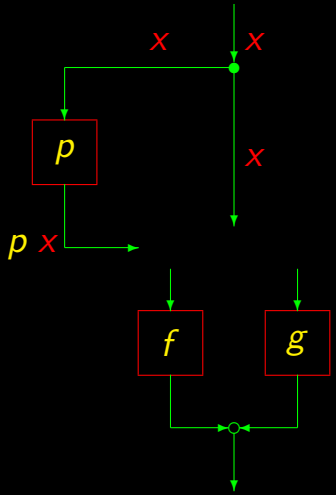
if-then-else



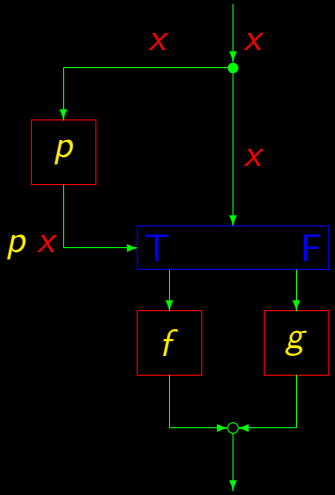
if-then-else



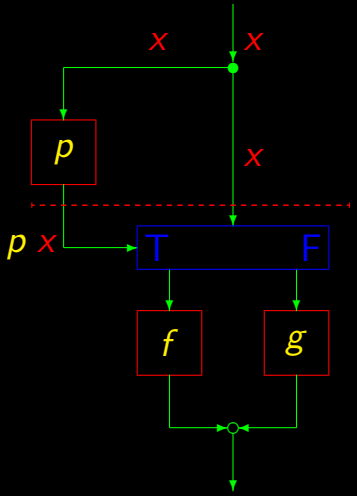
if-then-else



if-then-else



if-then-else



“Point-free”

if-then-else

$x \in A$

if-then-else

$x \in A$

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

if-then-else

$x \in A$

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

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

if-then-else

$x \in A$

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

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

$\mathbb{B}$ in Haskell:

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

if-then-else

$x \in A$

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

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

$\mathbb{B}$ in Haskell:

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

T represented by `True`

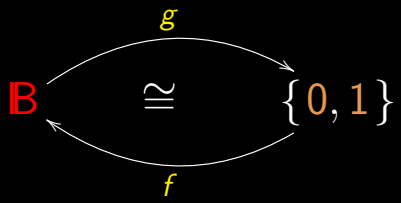
F represented by `False`

The type 2

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

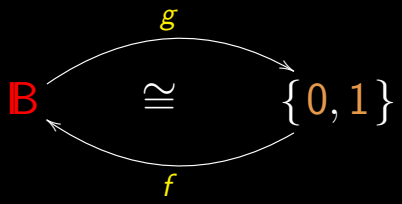
The type 2

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



The type 2

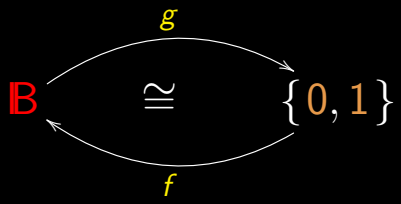
$$\mathbb{B} = \{F, T\} \cong \{\texttt{False}, \texttt{True}\} \cong \{0, 1\} \cong \cdots \cong 2$$



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

The type 2

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



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

$$a + a = 2\,a$$

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

$$a + a = 2\ a$$

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

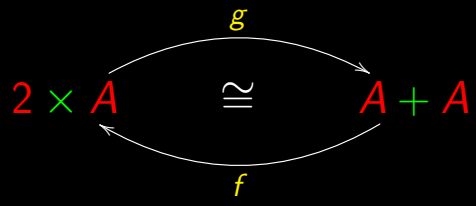
$$A \xrightarrow{\langle p, id \rangle} 2 \times A$$

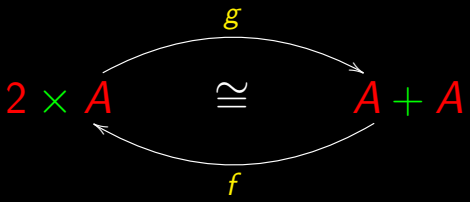
$a + a = 2\ a$

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

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

Is it the case that...?

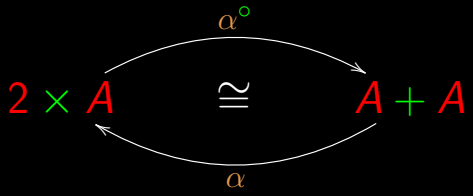




$$f = [\langle \underline{T}, id \rangle, \langle \underline{F}, id \rangle]$$

$$g = \cdots (?)$$

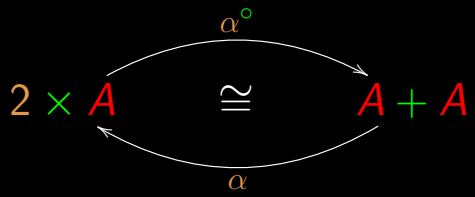
Notation



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

$$\alpha^\circ = \cdots$$

Notation

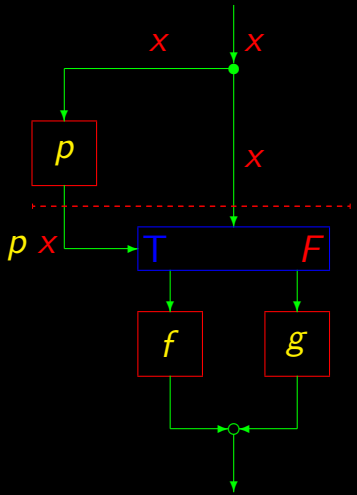


$$\alpha \cdot \alpha^\circ = id$$

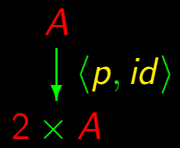
$$\alpha^\circ \cdot \alpha = id$$

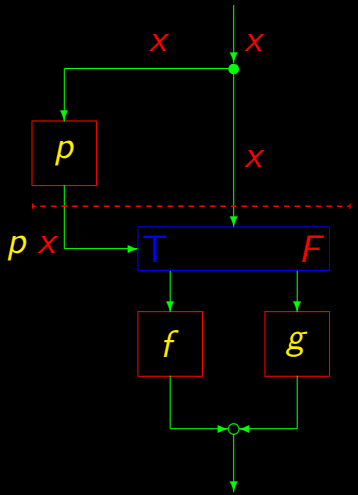
α determines α°

if-then-else



if-then-else

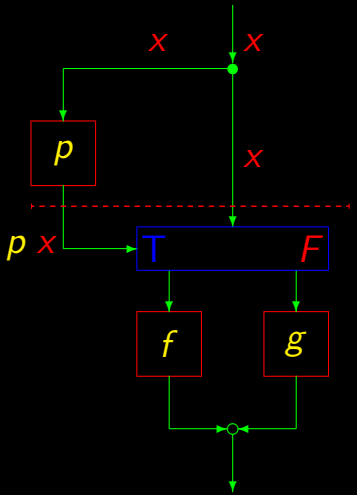




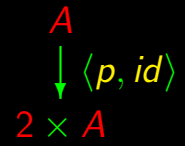
if-then-else

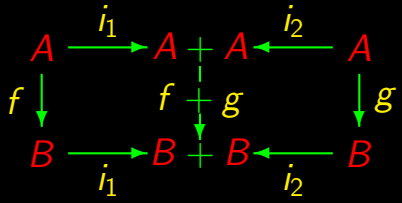
$$A \xrightarrow{\langle p, id \rangle} 2 \times A$$

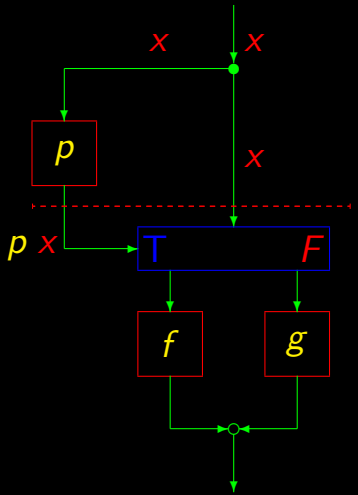
$$A \xrightarrow{f} B \qquad A \xrightarrow{g} B$$



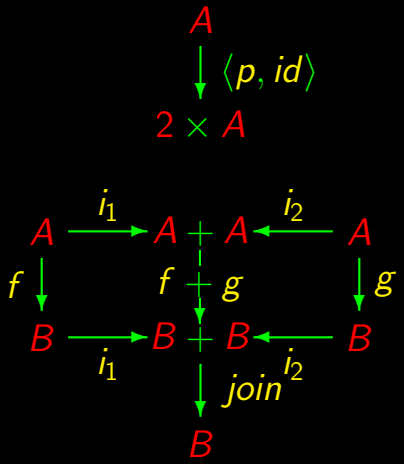
if-then-else

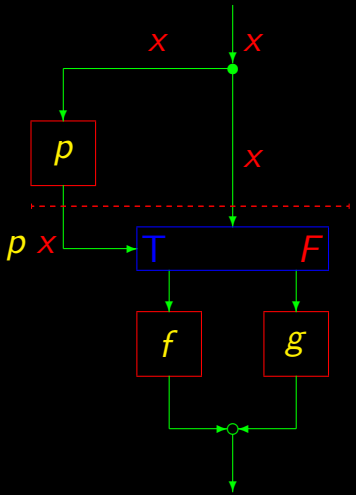




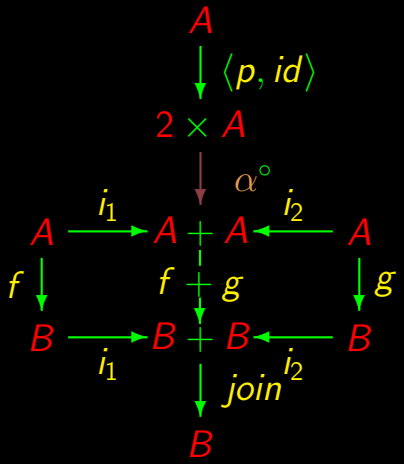


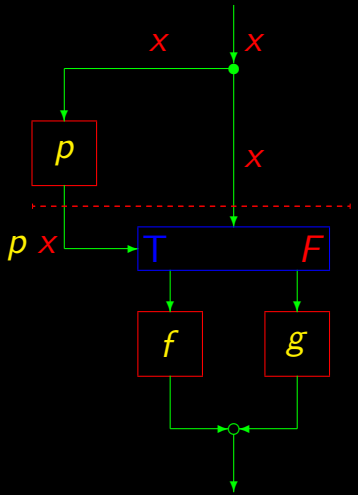
if-then-else





if-then-else





McCarthy Conditional $p \rightarrow f \ , g$

From the diagram:

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

McCarthy Conditional $p \to f \ , g$

From the diagram:

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

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

McCarthy Conditional $p \to f \ , g$

From the diagram:

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

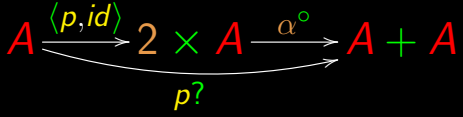
Since $$join \cdot (f + g) = [id, id] \cdot (f + g) = [f, g]$$

we have:

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

McCarthy Conditional $p \to f \ , g$

p-guard:



Conditional:

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

Pointwise

$$p? : A \to A + A$$

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

Pointwise

$$p? : A \to A + A$$

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

wherefrom

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

Pointwise

$$p? : A \to A + A$$

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

wherefrom

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

Case analysis?

Pointwise

$$p? : A \to A + A$$

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

wherefrom

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

Case analysis? No!

Between conditional and composition

Any "chemistry"?

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

Between conditional and composition

Any "chemistry"?

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

“Calculemus”

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

“Calculemus”

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

$$\Leftrightarrow \quad \{ \text{ definition of conditional } \}$$

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

“Calculemus”

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

$$\Leftrightarrow \quad \{ \text{ definition of conditional } \}$$

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

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

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

“Calculemus”

$$
\begin{array}{ll}
 & h \cdot (p \rightarrow f \ , g) \\
\Leftrightarrow & \quad \{ \text{ definition of conditional } \} \\
 & h \cdot [f , g] \cdot p? \\
\Leftrightarrow & \quad \{ \text{ } +\text{-fusion } \} \\
 & [h \cdot f , h \cdot g] \cdot p? \\
\Leftrightarrow & \quad \{ \text{ definition of conditional } \} \\
 & p \rightarrow (h \cdot f) \ , (h \cdot g)
\end{array}
$$

Conditional — 1st fusion law

Altogether:

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

Conditional — 1st fusion law

Altogether:

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

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

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

Conditional — 2nd fusion law

Intuitively, one has:

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

Conditional — 2nd fusion law

Intuitively, one has:

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

Intuitively?

Conditional — 2nd fusion law

Intuitively, one has:

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

Intuitively?

Proof ?

Conditional — 2nd fusion law

Intuitively, one has:

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

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

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

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

Isomorphisms

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

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

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

Isomorphisms

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

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

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

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

Isomorphisms

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

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

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

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

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

Isomorphisms

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

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

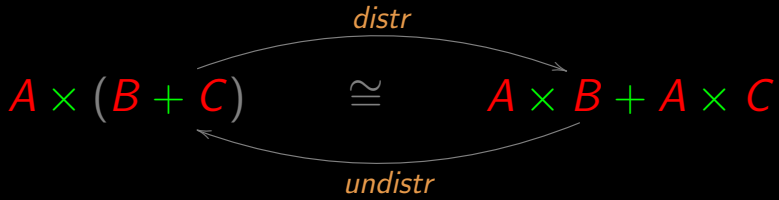
$$A \times 1 \cong A \quad ?$$

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

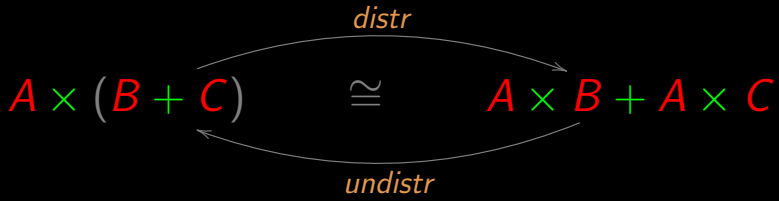
$$A + 0 \cong A \quad ?$$

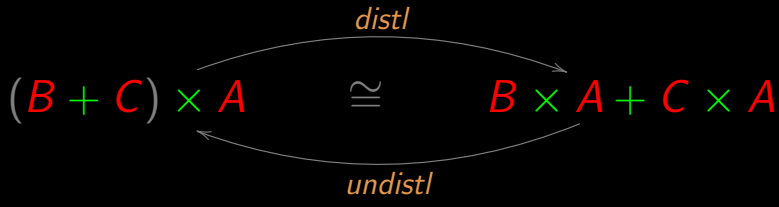
$$A \times (B + C) \cong A \times B + A \times C \quad ?$$

Distributivity

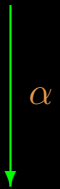


Distributivity

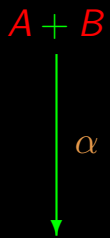




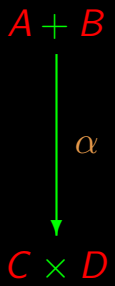
Exchange law



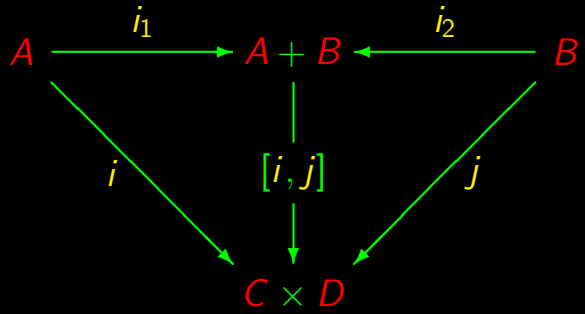
Exchange law



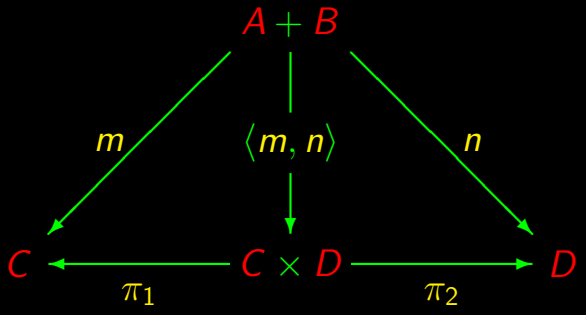
Exchange law



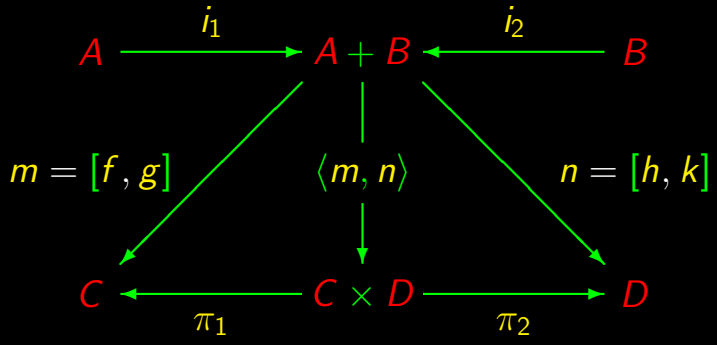
Exchange law



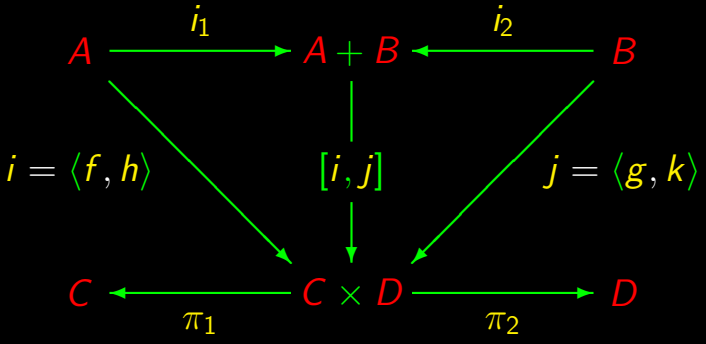
Exchange law



Exchange law



Exchange law



Summing up

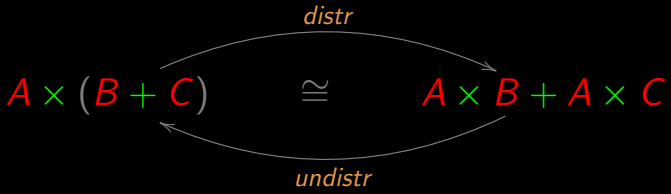
Exchange law:

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

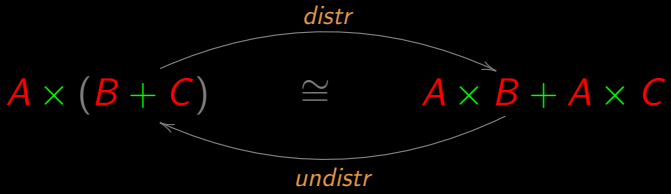
"I have a functional split but need an alternative..."

... and vice-versa

Distributivity

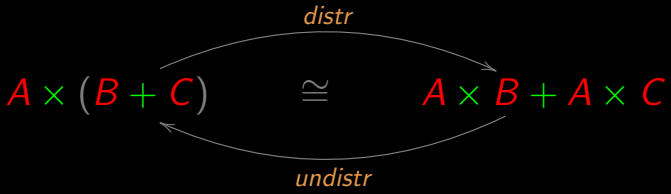


Distributivity



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

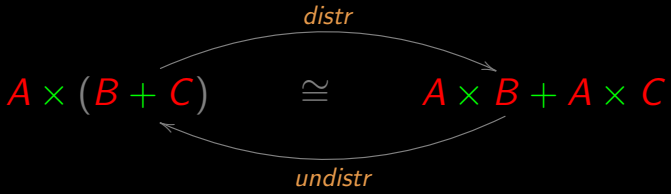
Distributivity



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

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

Distributivity

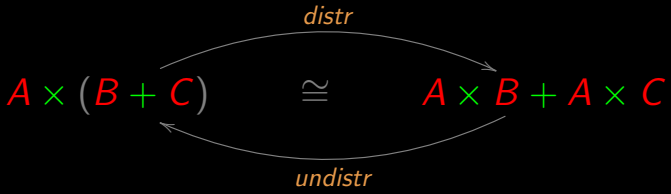


$$undistr = [\langle f, g \rangle, \langle h, k \rangle]$$

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

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

Distributivity

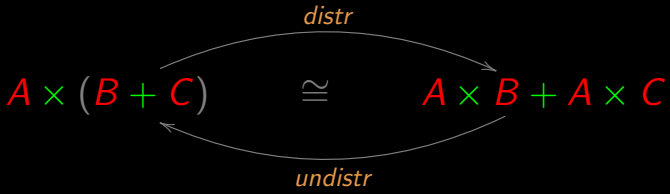


$$undistr = [\langle f, g \rangle, \langle h, k \rangle]$$

$$A \xleftarrow{f} A \times B \qquad A \xleftarrow{h} A \times C$$

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

Distributivity



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

$$A \xleftarrow{f} A \times B \qquad A \xleftarrow{h} A \times C$$

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

Distributivity

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

$$A \xleftarrow{f} A \times B \qquad A \xleftarrow{h} A \times C$$

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

Distributivity

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

$$A \xleftarrow{f} A \times B \qquad A \xleftarrow{h} A \times C$$

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

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

Distributivity

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

$$A \xleftarrow{f} A \times B \qquad A \xleftarrow{h} A \times C$$

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

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

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

Distributivity

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

$$A \xleftarrow{f} A \times B \qquad A \xleftarrow{h} A \times C$$

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

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

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

Distributivity

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

$$A \xleftarrow{f} A \times B \qquad A \xleftarrow{h} A \times C$$

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

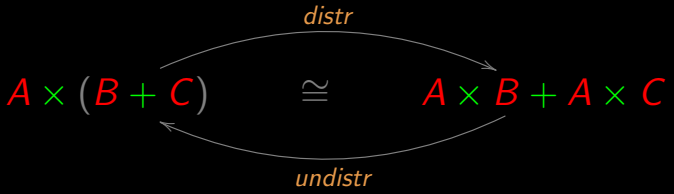
$$A \xleftarrow{\pi_1} A \times B \qquad A \xleftarrow{\pi_1} A \times C$$

$$B + C \xleftarrow{i_1 \cdot \pi_2} A \times B \qquad B + C \xleftarrow{i_2 \cdot \pi_2} A \times C$$

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

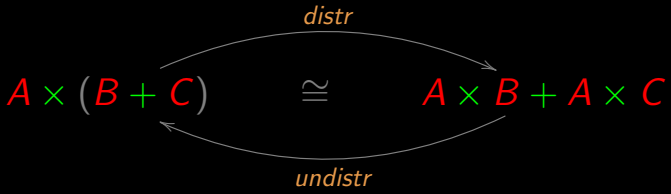
Distributivity

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



Distributivity

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



By definition of $f \times g$

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

The type 1

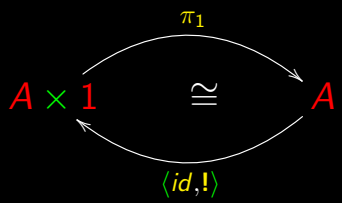
In Haskell:

```
() :: ()
```

Then

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

Isomorphism



The type 1

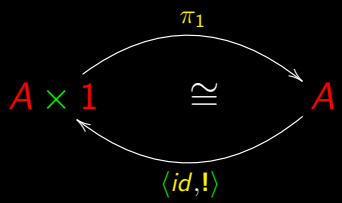
In Haskell:

```
() :: ()
```

Then

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

Isomorphism



... "**!**" ?

The type 1

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

The type 1

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

"Bang":

$$A \xrightarrow{!} 1$$

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

The type 1

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

"Bang":

$$A \xrightarrow{!} 1$$

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

"Points":

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

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

(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

$$0 = \{\,\}$$

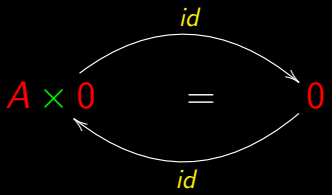
Immediate in e.g. Haskell:

```
data Zero
```

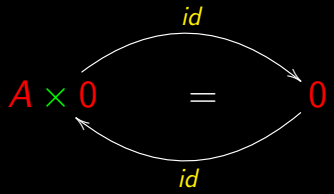
0 is not inhabited: $\neg\,(a \in 0)$ for all a

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

The type 0



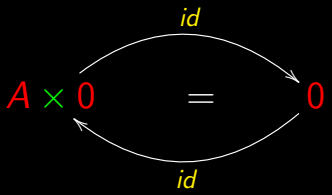
The type 0



In fact

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

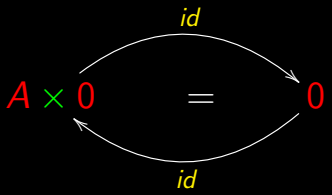
The type 0



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



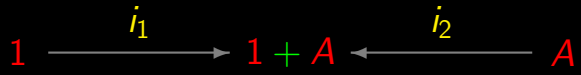
In fact

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

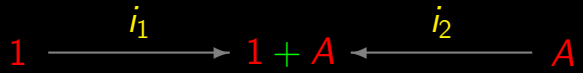
The type $1 + A$

$$1 + A$$

The type $1 + A$

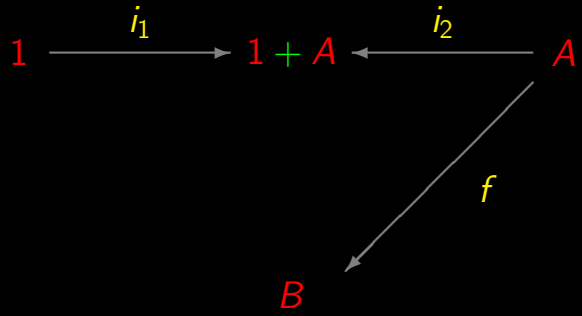


The type $1 + A$

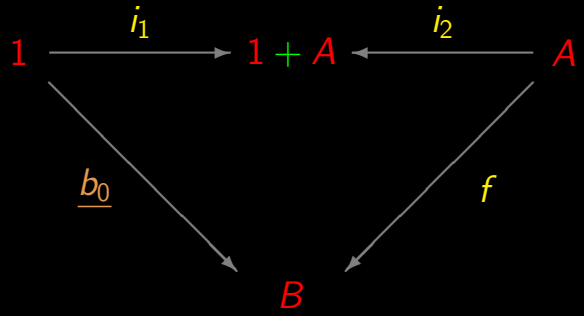


B

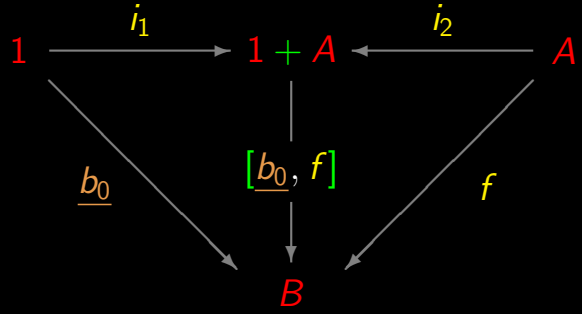
The type $1 + A$



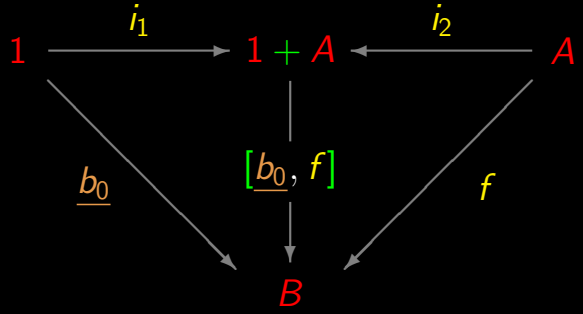
The type $1 + A$



The type $1 + A$



The type $1 + A$



$1 + A$

"pointer to A"

More isomorphisms!

More isomorphisms!

Back to

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

More isomorphisms!

Back to

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

In Haskell:

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

More isomorphisms!

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

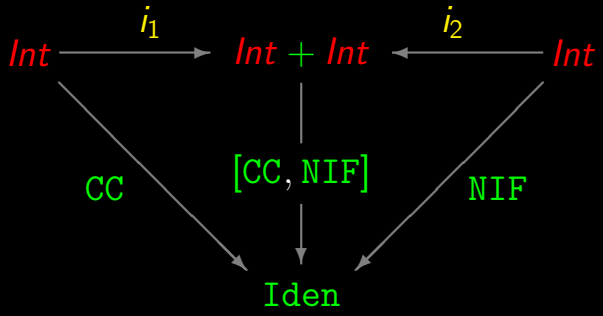
In Haskell (Portugal):

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

In Haskell (Germany):

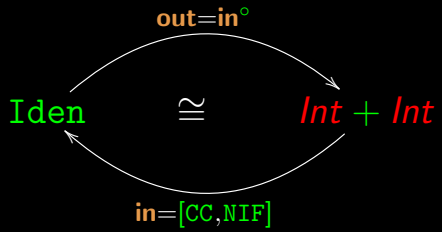
```
data Iden = IDK Int | SZN Int
```

More isomorphisms!

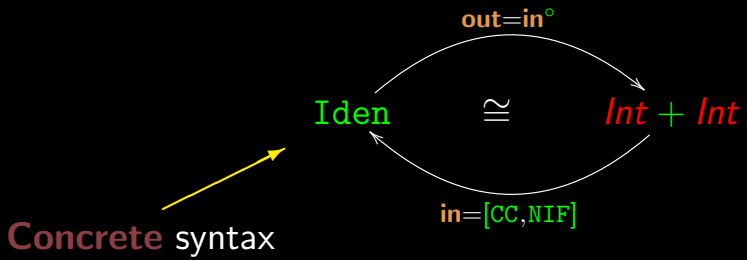



```
data Iden = CC Int | NIF Int
```

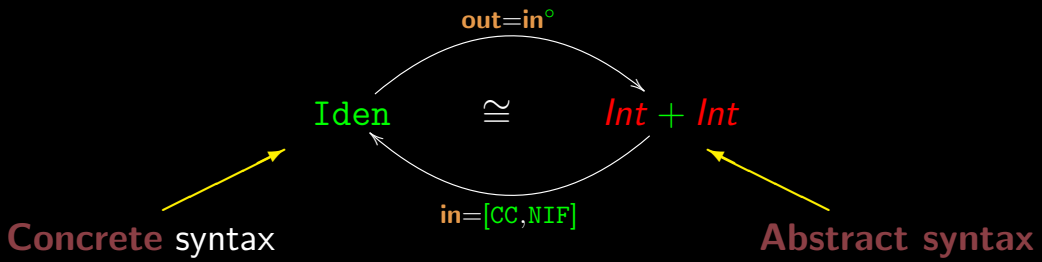
More isomorphisms!



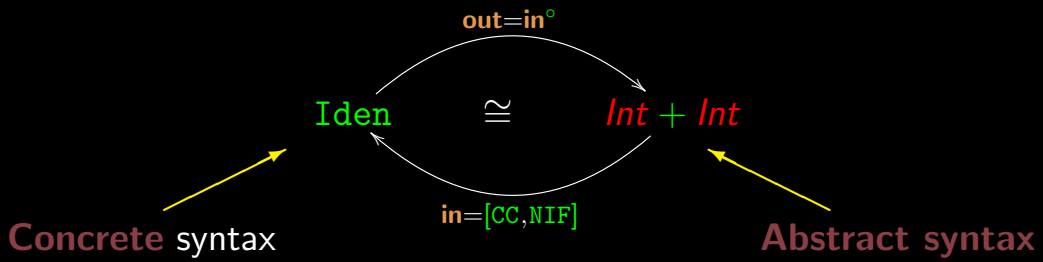
More isomorphisms!



More isomorphisms!

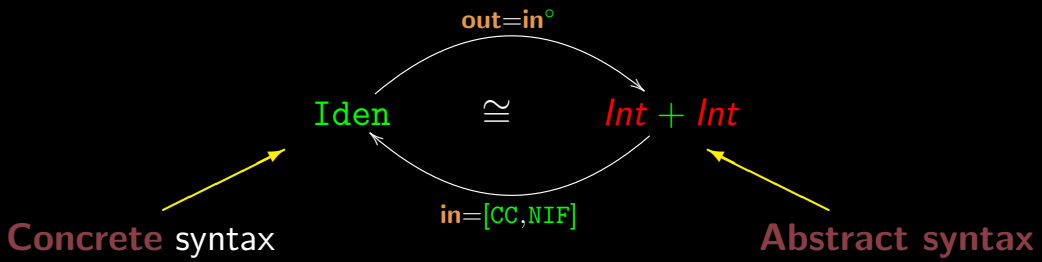


More isomorphisms!



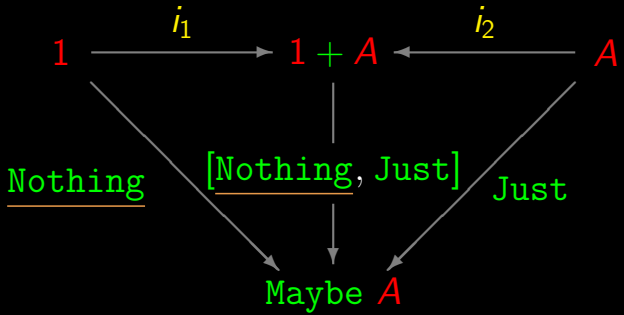
in (“get in”)

More isomorphisms!



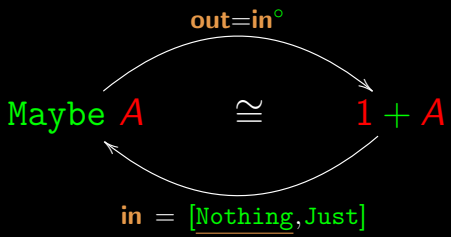
in (“get in”) **out** (“get out”)

Maybe

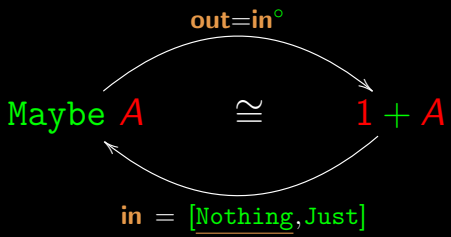



```
data Maybe a = Nothing | Just a
```

Maybe



Maybe



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

Maybe

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

Maybe

$$
\begin{aligned}
& \mathsf{out} \cdot \mathsf{in} = \mathit{id} \\
\Leftrightarrow \quad & \{\ \mathsf{in} = [\underline{\mathtt{Nothing}}, \mathtt{Just}]\ \} \\
& \mathsf{out} \cdot [\underline{\mathtt{Nothing}}, \mathtt{Just}] = \mathit{id}
\end{aligned}
$$

Maybe

$$
\begin{aligned}
& \mathsf{out} \cdot \mathsf{in} = \mathit{id} \\
\Leftrightarrow & \quad \{ \;\; \mathsf{in} = [\underline{\mathtt{Nothing}}, \mathtt{Just}] \;\; \} \\
& \mathsf{out} \cdot [\underline{\mathtt{Nothing}}, \mathtt{Just}] = \mathit{id} \\
\Leftrightarrow & \quad \{ \;\; +\text{-fusion} \;\; \} \\
& [\mathsf{out} \cdot \underline{\mathtt{Nothing}}, \mathsf{out} \cdot \mathtt{Just}] = \mathit{id}
\end{aligned}
$$

Maybe

$$
\begin{aligned}
& \mathsf{out} \cdot \mathsf{in} = id \\
\Leftrightarrow & \quad \{ \ \mathsf{in} = [\underline{\mathtt{Nothing}}, \mathtt{Just}] \ \} \\
& \mathsf{out} \cdot [\underline{\mathtt{Nothing}}, \mathtt{Just}] = id \\
\Leftrightarrow & \quad \{ \ + \text{-fusion} \ \} \\
& [\mathsf{out} \cdot \underline{\mathtt{Nothing}}, \mathsf{out} \cdot \mathtt{Just}] = id \\
\Leftrightarrow & \quad \{ \ \text{universal-}+\text{; constant function} \ \} \\
& \begin{cases} \underline{\mathsf{out}\ \mathtt{Nothing}} = i_1 \\ \mathsf{out} \cdot \mathtt{Just} = i_2 \end{cases}
\end{aligned}
$$

Maybe

$$\Leftrightarrow \qquad \{ \text{ variables } ; \underline{\texttt{Nothing}} : 1 \to \texttt{Maybe}\ A \ \}$$

$$\begin{cases} \underline{\mathbf{out}\ \texttt{Nothing}}\ () = i_1\ () \\ (\mathbf{out} \cdot \texttt{Just})\ a = i_2\ a \end{cases}$$

Maybe

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

$$\begin{cases} \underline{\mathbf{out}\ \texttt{Nothing}}\ () = i_1\ () \\ (\mathbf{out} \cdot \texttt{Just})\ a = i_2\ a \end{cases}$$

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

$$\begin{cases} \mathbf{out}\ \texttt{Nothing} = i_1\ () \\ \mathbf{out}\ (\texttt{Just}\ a) = i_2\ a \end{cases}$$

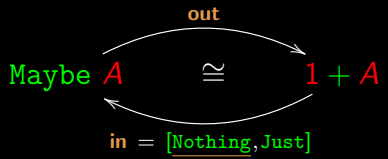
Maybe

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

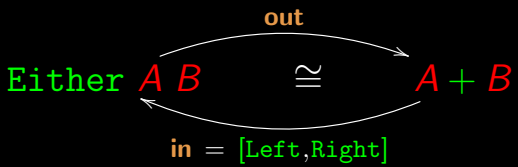
$$\begin{cases} \underline{\mathbf{out}\ \texttt{Nothing}}\ () = i_1\ () \\ \underline{(\mathbf{out} \cdot \texttt{Just})}\ a = i_2\ a \end{cases}$$

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

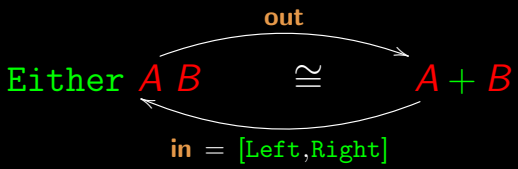
$$\begin{cases} \mathbf{out}\ \texttt{Nothing} = i_1\ () \\ \mathbf{out}\ (\texttt{Just}\ a) = i_2\ a \end{cases}$$

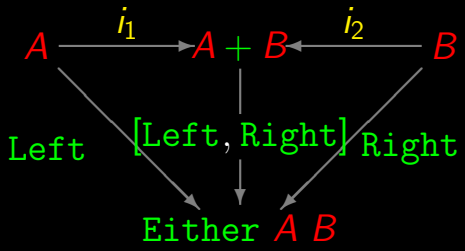


Either



Either





$[f, g]$

either f g

Cálculo de Programas

Class T05

Polymorphism

$$reverse :: [a] \to [a]$$

Polymorphism

$$reverse :: [a] \to [a]$$

$$reverse :: [Int] \to [Int]$$

Polymorphism

$$reverse :: [a] \to [a]$$

$$reverse :: [Int] \to [Int]$$
$$reverse :: String \to String$$

Polymorphism

$$reverse :: [a] \to [a]$$

$$reverse :: [Int] \to [Int]$$
$$reverse :: String \to String$$
$$\ldots$$

Polymorphism

$$reverse :: [a] \to [a]$$

$$reverse :: [Int] \to [Int]$$
$$reverse :: String \to String$$
...

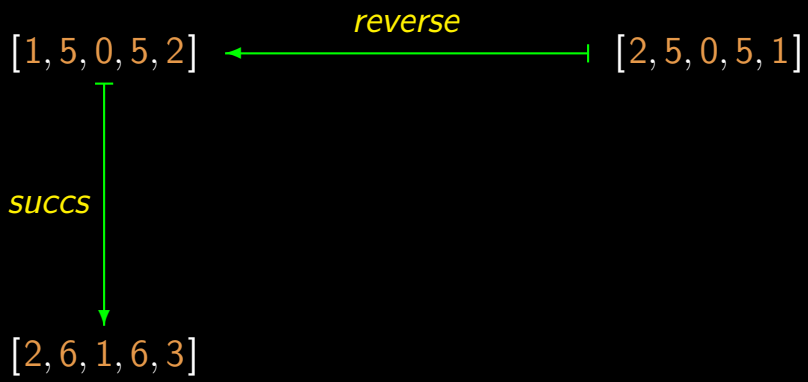
"From the **type** of a **polymorphic function** we can derive a **theorem** that it satisfies. (...) How useful are the theorems so generated? Only time and experience will tell (...)" [**Philip Wadler** 1989]

Natural properties

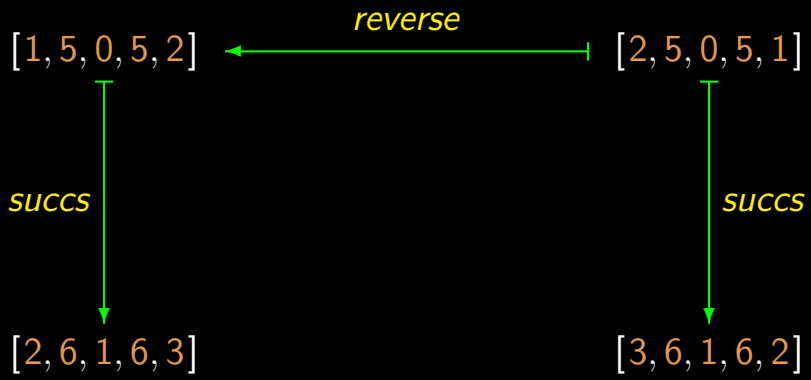
$[2, 5, 0, 5, 1]$

Natural properties

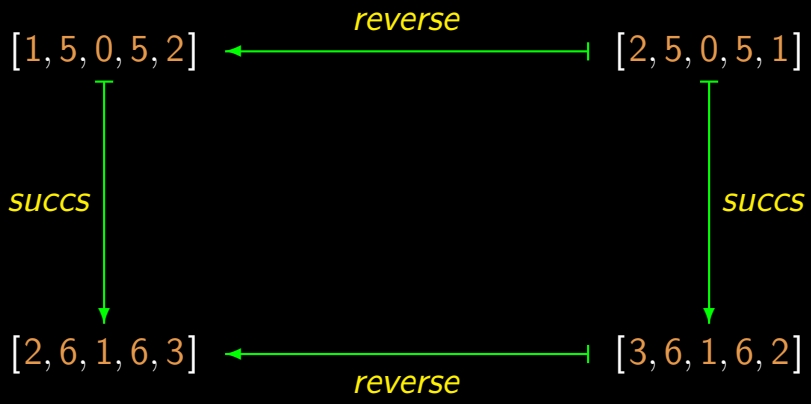
Natural properties



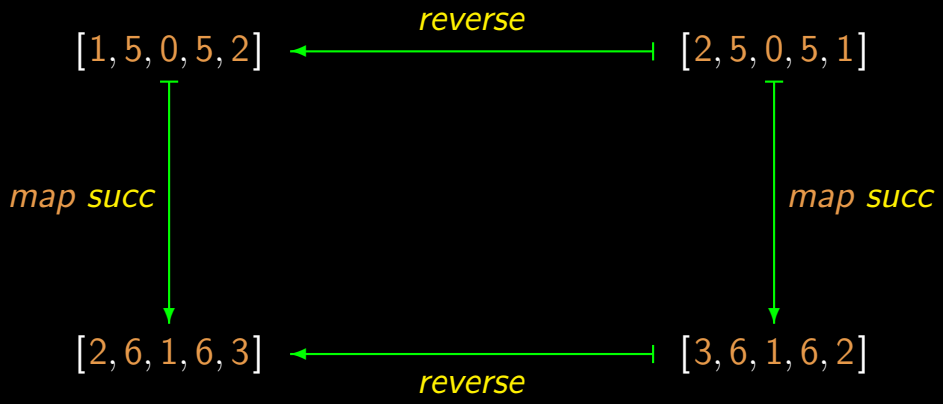
Natural properties



Natural properties

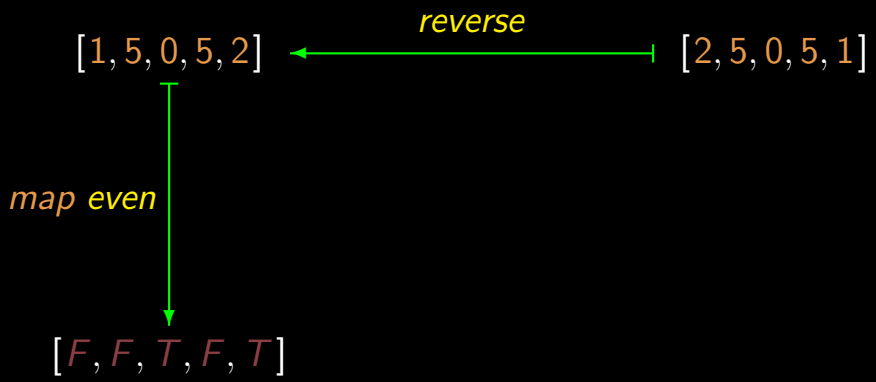


Natural properties

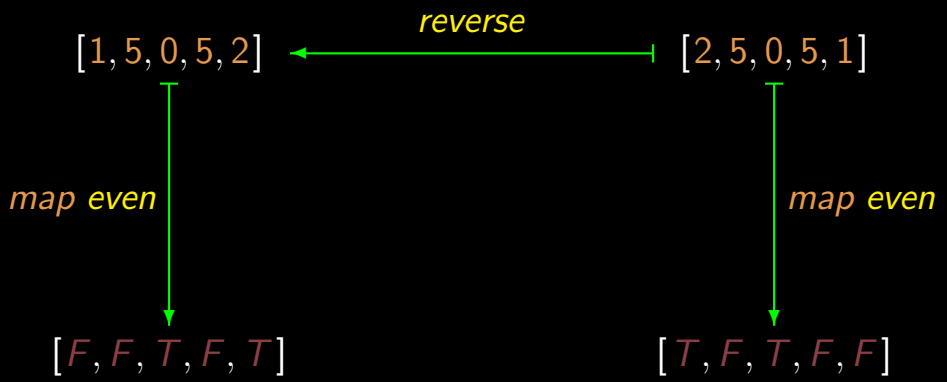


Natural properties

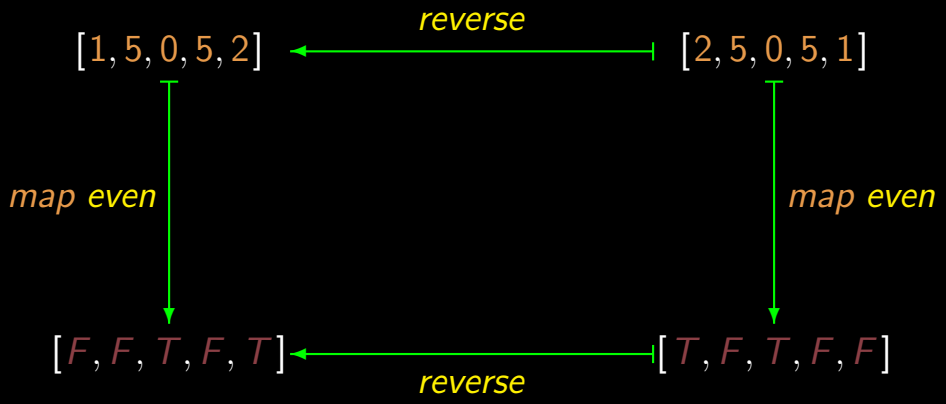
Natural properties



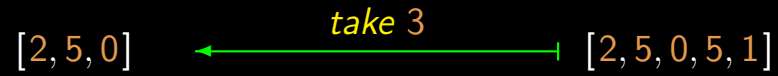
Natural properties



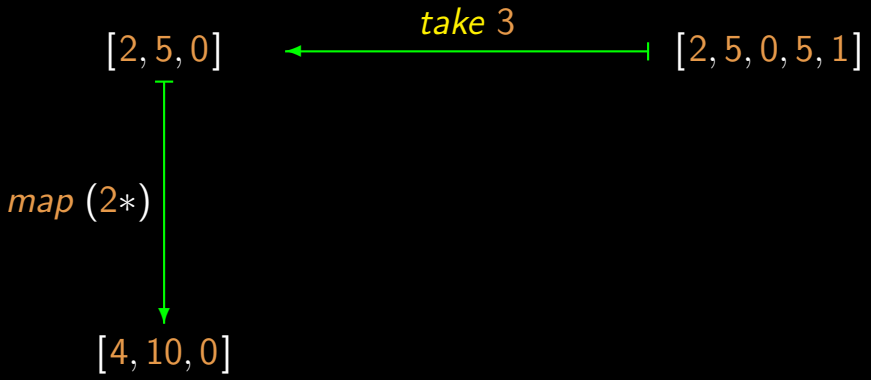
Natural properties



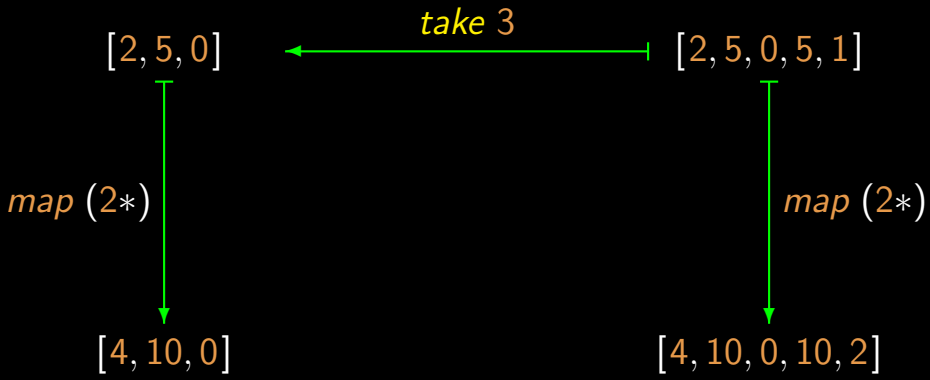
Natural properties



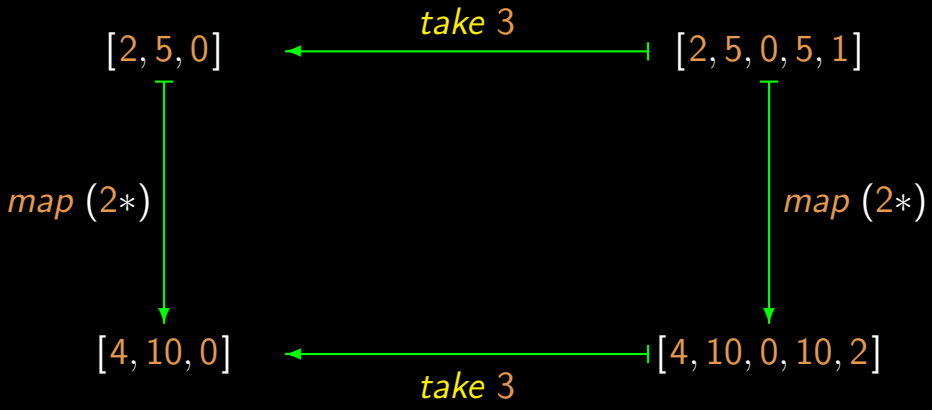
Natural properties



Natural properties

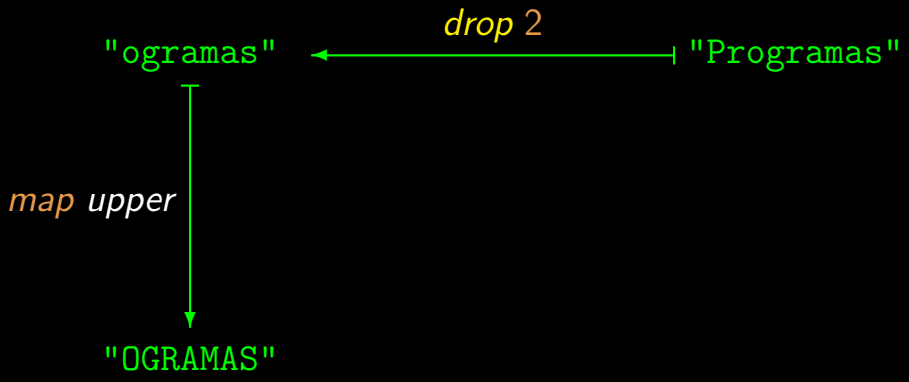


Natural properties

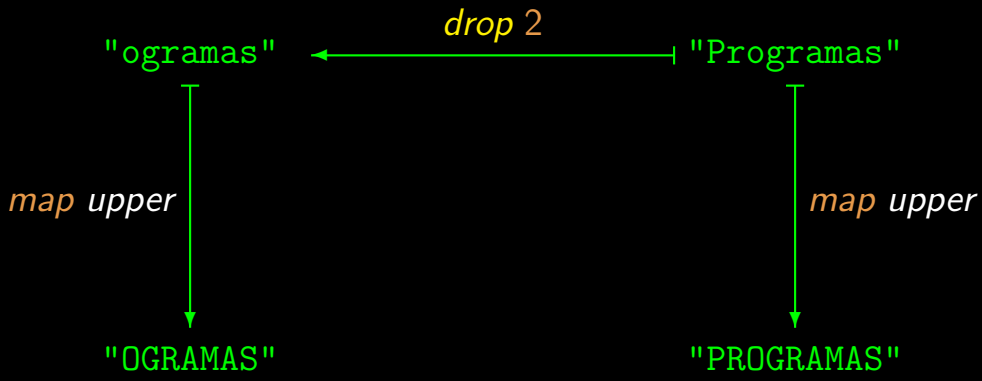


Natural properties

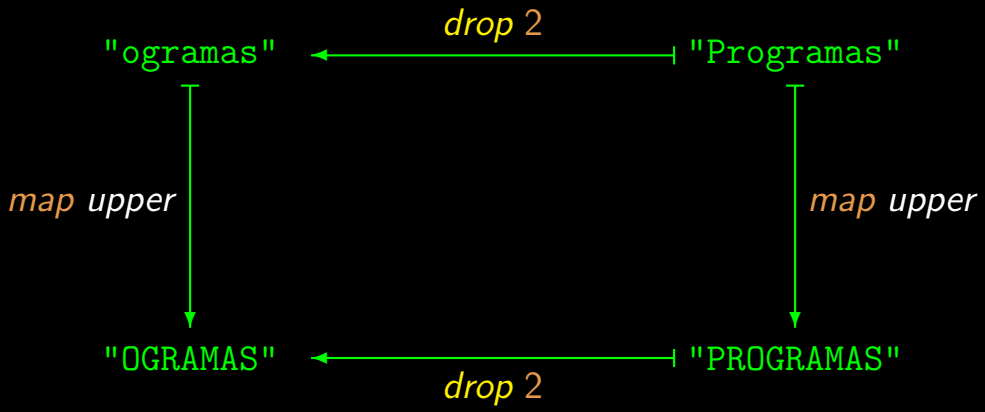
Natural properties



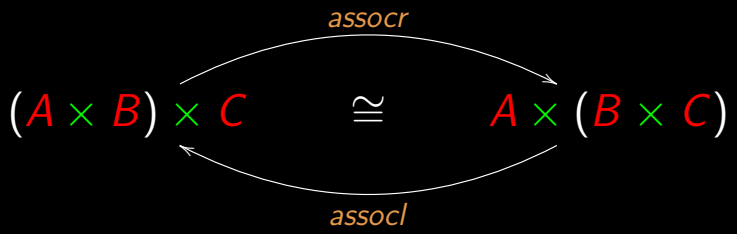
Natural properties



Natural properties



Back to...

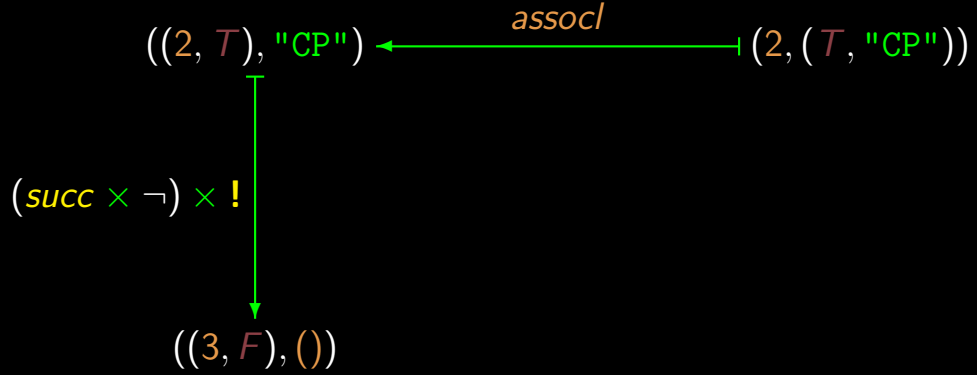


$$assocr \cdot assocl = id$$

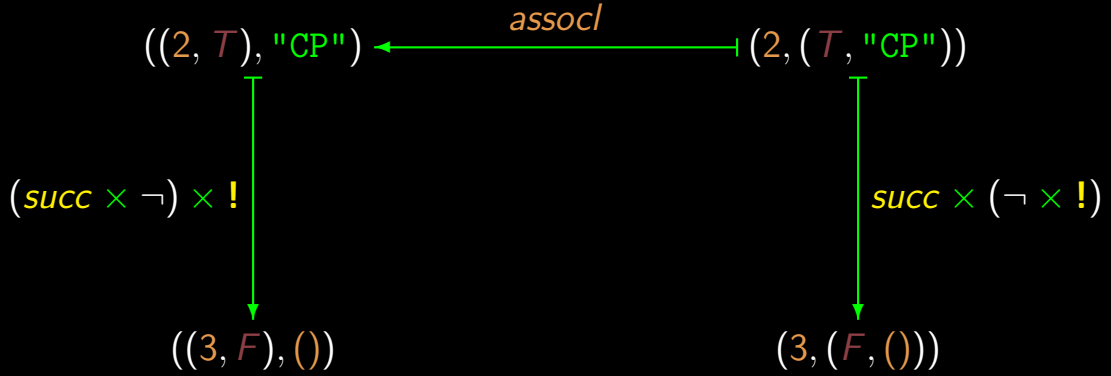
$$assocl \cdot assocr = id$$

“Natural” property

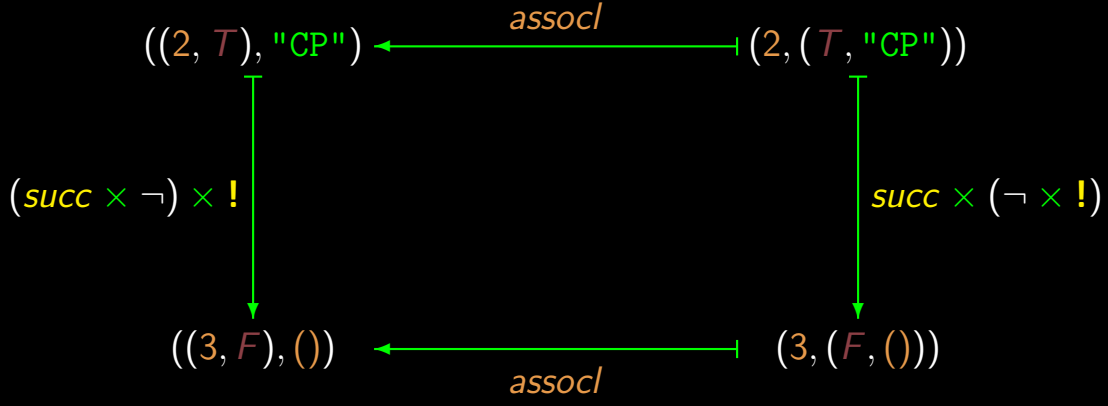
“Natural” property



“Natural” property

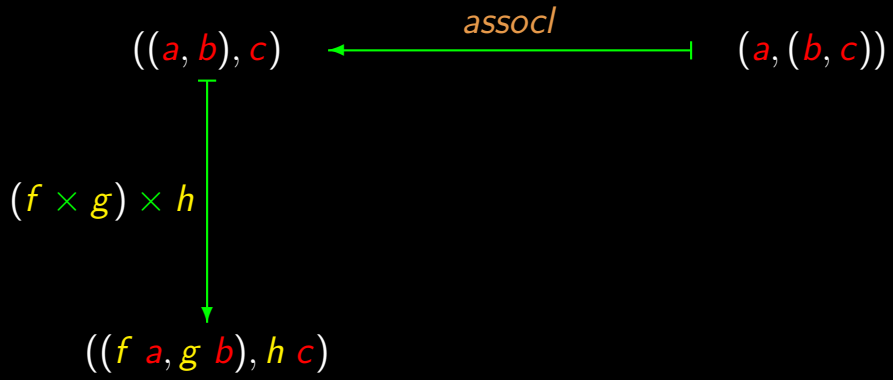


“Natural” property

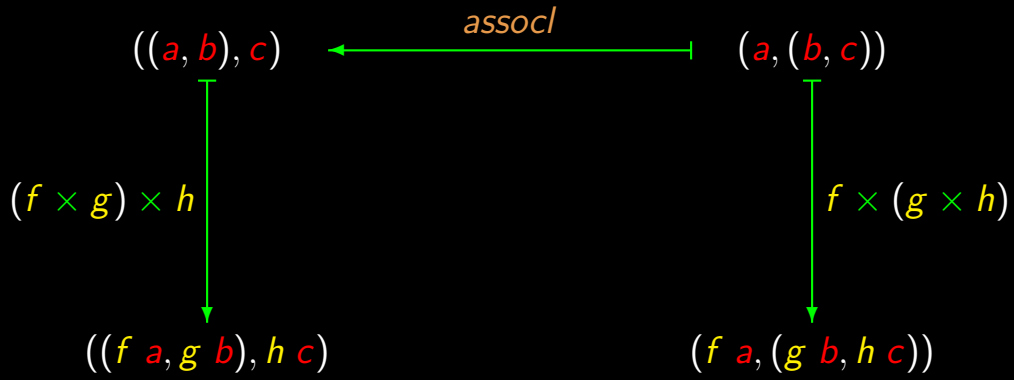


“Natural” property

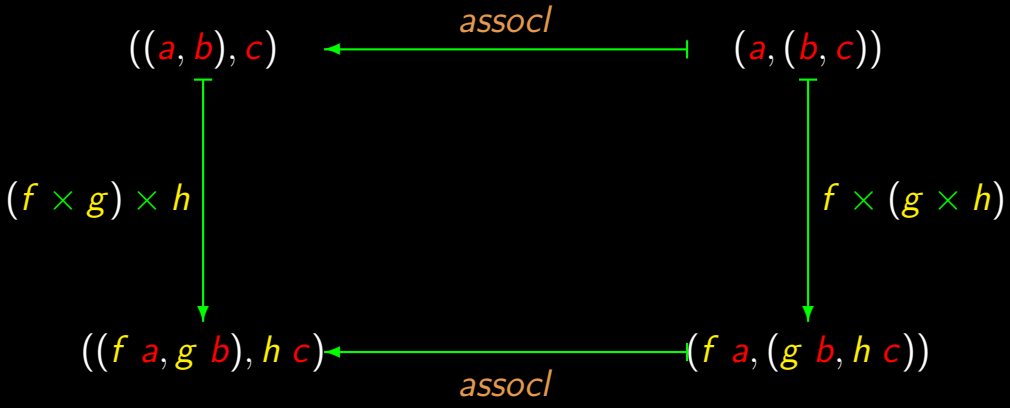
“Natural” property



“Natural” property



“Natural” property

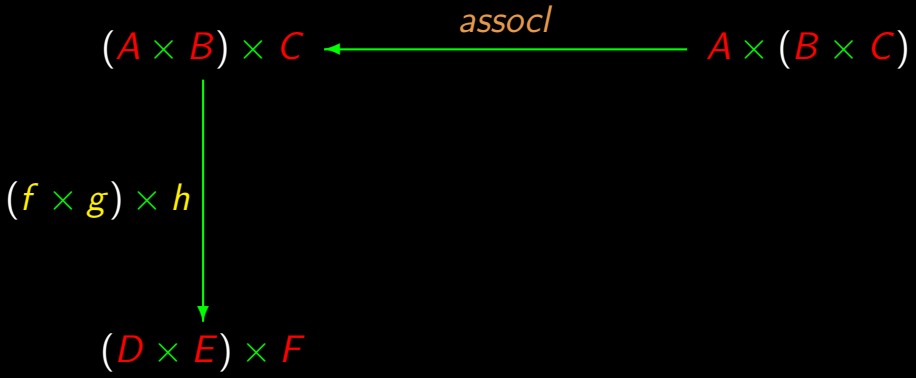


Natural properties

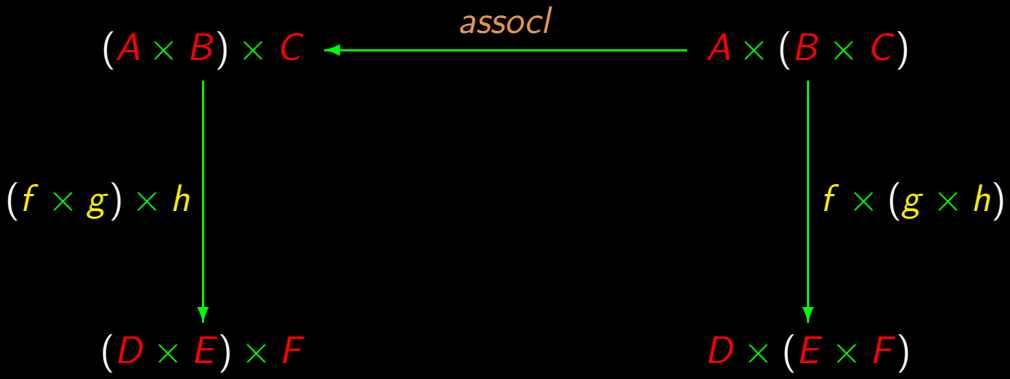
$$A \times (B \times C)$$

Natural properties

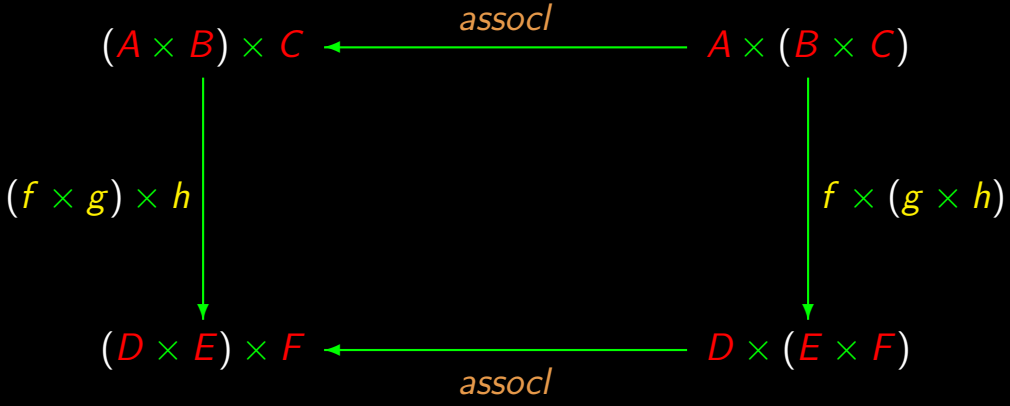
Natural properties



Natural properties

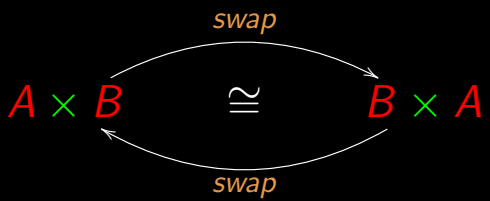


Natural properties



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

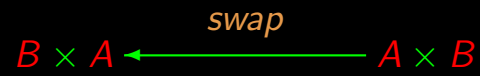
Back to...



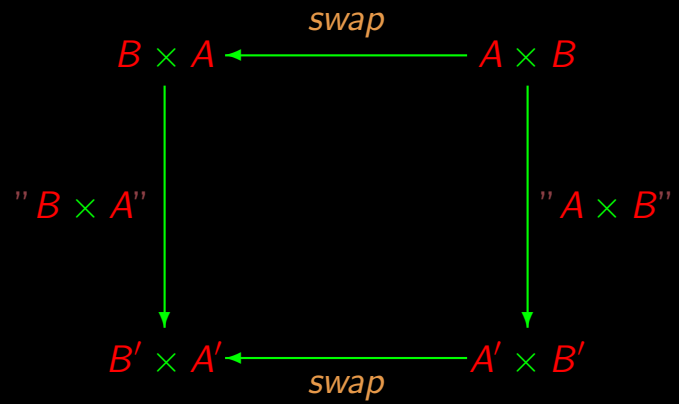
$$swap \cdot swap = id$$

Practical rule

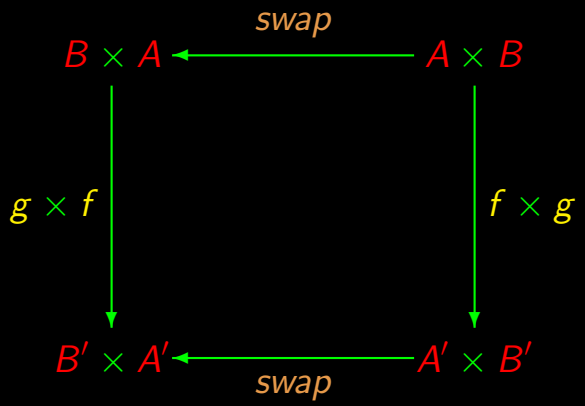
Practical rule



Practical rule



Practical rule



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

Practical rule

In the vertical arrows,

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

Practical rule

In the vertical arrows,

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

Practical rule

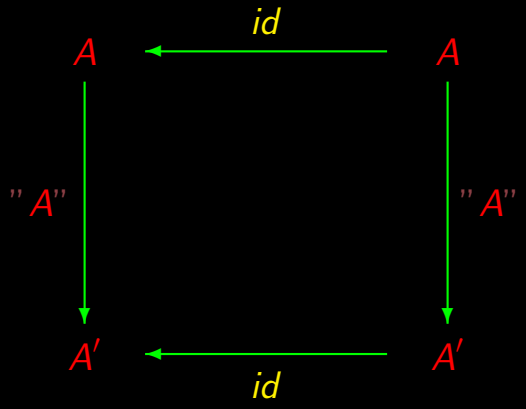
In the vertical arrows,

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

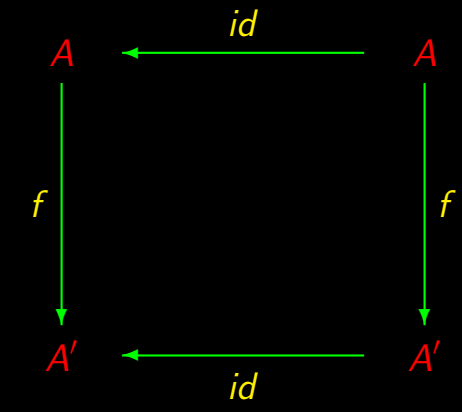
Simplest case...

Simplest case...

Simplest case...

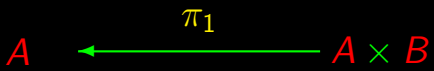


Natural-*id*

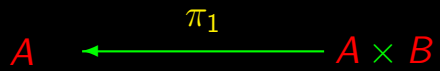


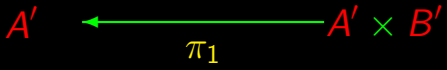
$$f \cdot id = id \cdot f$$

Natural-π_1

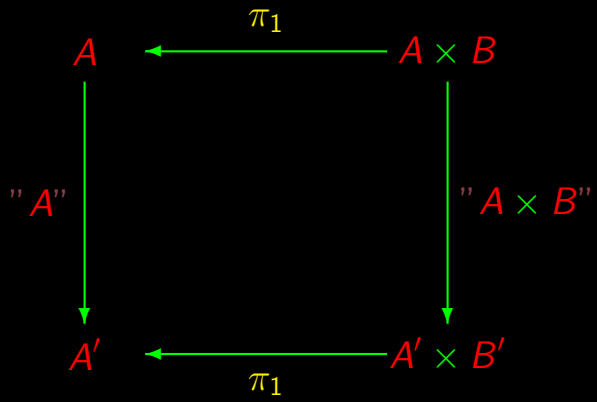


Natural-π_1

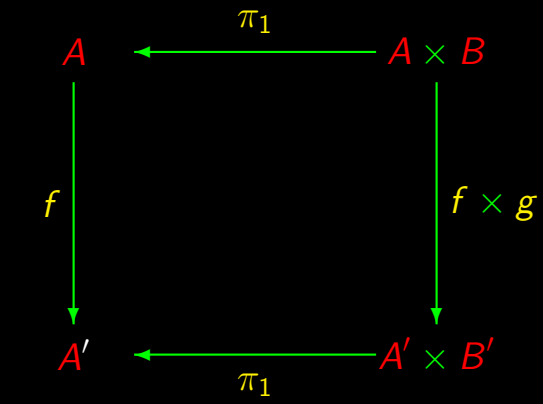




Natural-π_1

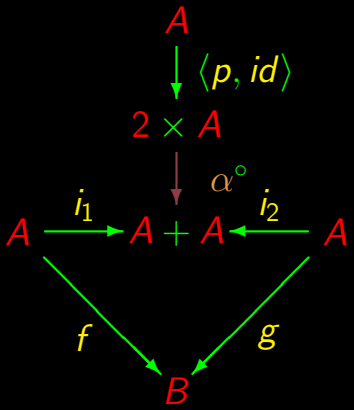


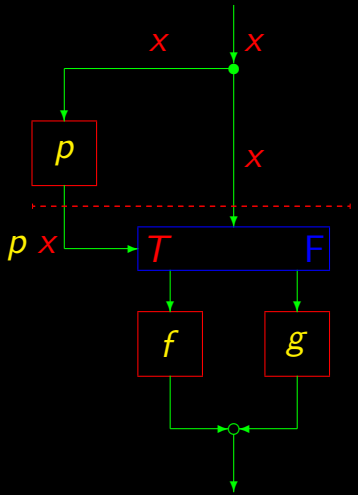
Natural-π_1



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

Back to "if-then-else"





2ª fusion law of conditionals

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

2ª fusion law of conditionals

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

follows from

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

2^a fusion law of conditionals

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

follows from

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

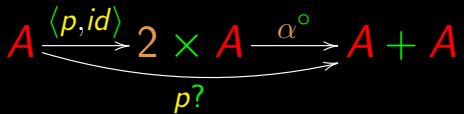
So we are left with the **proof** of this thene...

Let us do it now

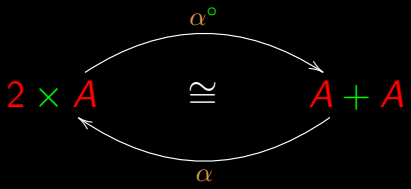
Prove

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

knowing:



Recall



$$\alpha = [\langle \underline{T}, id \rangle, \langle \underline{F}, id \rangle]$$
$$\alpha^\circ = \cdots ?$$

Let us try it

$$p? \cdot f$$

Let us try it

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

Let us try it

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

Let us try it

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

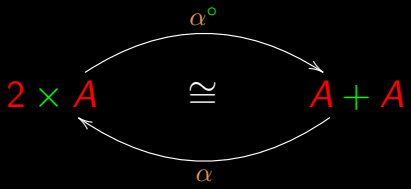
Let us try it

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

Let us try it

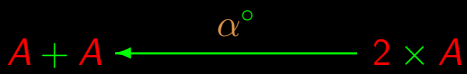
$$
\begin{aligned}
& p? \cdot f \\
= & \quad \{ \text{ substitution } p? = \alpha^\circ \cdot \langle p, id \rangle \} \\
& \alpha^\circ \cdot \langle p, id \rangle \cdot f \\
= & \quad \{ \times\text{-fusion} \} \\
& \alpha^\circ \cdot \langle p \cdot f, id \cdot f \rangle \\
= & \quad \{ \text{ natural-}id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

Recall

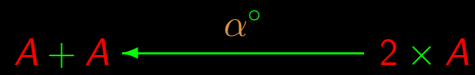


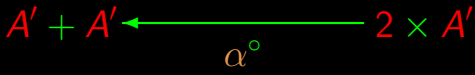
$$\alpha = [\langle \underline{T}, id \rangle, \langle \underline{F}, id \rangle]$$
$$\alpha^\circ = \cdots ?$$

Natural-α°

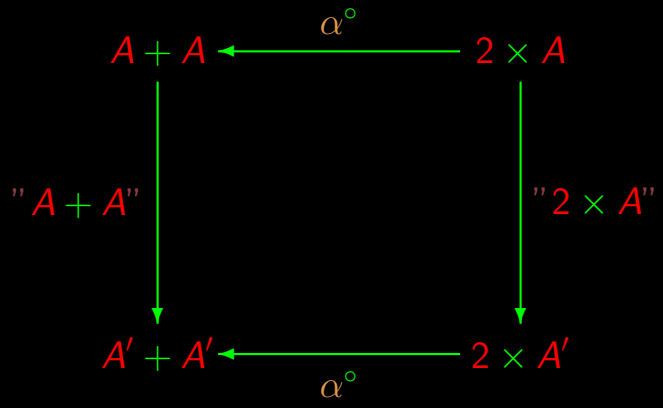


Natural-α°

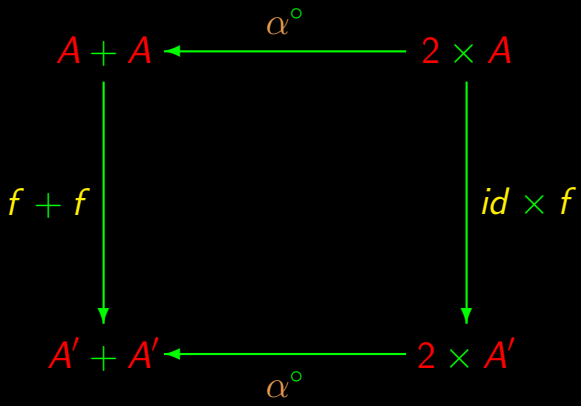




Natural-α°



Natural-α°



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

... and this finishes the proof!

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

... and this finishes the proof!

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

... and this finishes the proof!

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

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
$2\ 3\ +$	**posfix** notation	operator after

Notation variants

$2 + 3$	**infix** notation	operator between
$+\ 2\ 3$	**prefix** notation	operator before
$2\ 3\ +$	**posfix** 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

$$\pi_1 : A \times B \to A$$

Example

$$\pi_1 : A \times B \to A$$
$$\pi_1\ (a, b) = a$$

Example

$$\pi_1 : A \times B \to A$$
$$\pi_1\ (a, b) = a$$

(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

$$\pi_1\prime = \texttt{curry}\ \pi_1$$

In Haskell

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

In Haskell

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

Important:

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

In Haskell

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

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

$$\texttt{curry} :: ((a, b) \to c) \to (a \to b \to c)$$

$$\texttt{uncurry} :: (a \to b \to c) \to (a, b) \to c$$

Curry | uncurry

$$\texttt{curry} :: ((a, b) \to c) \to (a \to b \to c)$$

$$\texttt{uncurry} :: (a \to b \to c) \to (a, b) \to c$$

Curry | uncurry

$$\texttt{curry} :: ((a, b) \to c) \to (a \to b \to c)$$

$$\texttt{uncurry} :: (a \to b \to c) \to (a, b) \to c$$

Curry | uncurry

$$\texttt{curry} :: ((a, b) \to c) \to (a \to b \to c)$$

$$\texttt{uncurry} :: (a \to b \to c) \to (a, b) \to c$$

each other inverses?

Curry | uncurry

$$\texttt{curry} :: ((a, b) \to c) \to a \to b \to c$$

Curry | uncurry

$$\texttt{curry} :: ((a, b) \to c) \to a \to b \to c$$

$$\texttt{curry}$$

Curry | uncurry

$$\texttt{curry} :: ((a, b) \to c) \to a \to b \to c$$

$$\texttt{curry}\ f$$

Curry | uncurry

$$\texttt{curry} :: ((a, b) \to c) \to a \to b \to c$$

$$\texttt{curry}\ f\ a$$

Curry | uncurry

$$\texttt{curry} :: ((a, b) \to c) \to a \to b \to c$$

$$\texttt{curry}\ f\ a\ b$$

Curry | uncurry

$$\texttt{curry} :: ((a, b) \to c) \to a \to b \to c$$
$$\texttt{curry}\ f\ a\ b = $$

Curry | uncurry

$$\texttt{curry} :: ((a, b) \to c) \to a \to b \to c$$
$$\texttt{curry}\ f\ a\ b = f\ (a, b)$$

Curry | uncurry

$$\texttt{curry} :: ((a, b) \to c) \to a \to b \to c$$
$$\texttt{curry}\ f\ a\ b = f\ (a, b)$$

$$\texttt{uncurry} :: (a \to b \to c) \to (a, b) \to c$$

Curry | uncurry

$$\texttt{curry} :: ((a, b) \to c) \to a \to b \to c$$
$$\texttt{curry}\ f\ a\ b = f\ (a, b)$$

$$\texttt{uncurry} :: (a \to b \to c) \to (a, b) \to c$$
$$\texttt{uncurry}$$

Curry | uncurry

$$\texttt{curry} :: ((a, b) \to c) \to a \to b \to c$$
$$\texttt{curry}\ f\ a\ b = f\ (a, b)$$

$$\texttt{uncurry} :: (a \to b \to c) \to (a, b) \to c$$
$$\texttt{uncurry}\ g$$

Curry | uncurry

$$\texttt{curry} :: ((a, b) \to c) \to a \to b \to c$$
$$\texttt{curry}\ f\ a\ b = f\ (a, b)$$

$$\texttt{uncurry} :: (a \to b \to c) \to (a, b) \to c$$
$$\texttt{uncurry}\ g\ (a, b)$$

Curry | uncurry

$$\texttt{curry} :: ((a, b) \to c) \to a \to b \to c$$
$$\texttt{curry}\ f\ a\ b = f\ (a, b)$$

$$\texttt{uncurry} :: (a \to b \to c) \to (a, b) \to c$$
$$\texttt{uncurry}\ g\ (a, b) =$$

Curry | uncurry

$$\texttt{curry} :: ((a, b) \to c) \to a \to b \to c$$
$$\texttt{curry}\ f\ a\ b = f\ (a, b)$$

$$\texttt{uncurry} :: (a \to b \to c) \to (a, b) \to c$$
$$\texttt{uncurry}\ g\ (a, b) = g\ a\ b$$

Curry | uncurry

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

Curry | uncurry

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

Curry | uncurry

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

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

Curry | uncurry

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

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

Curry | uncurry

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

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

Curry | uncurry

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

$$id\ x = x$$

Curry | uncurry

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

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

Curry | uncurry

$$\texttt{uncurry} \cdot \texttt{curry} = \mathit{id}$$

□

Curry | uncurry

$$\texttt{curry}\ (\texttt{uncurry}\ f) = f\ ?$$

Curry | uncurry

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

Curry | uncurry

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

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

Curry | uncurry

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

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

Curry | uncurry

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

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

Curry | uncurry

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

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

Curry | uncurry

$$(\texttt{curry} \cdot \texttt{uncurry})\, g = g$$

$$id\ x = x$$

Curry | uncurry

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

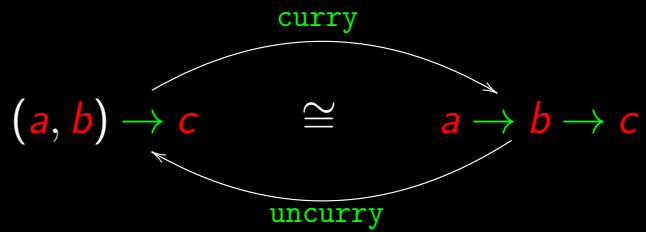
$$f\ x = g\ x \Leftrightarrow f = g$$

Curry | uncurry

$$\texttt{curry} \cdot \texttt{uncurry} = \mathit{id}$$

□

Isomorphism



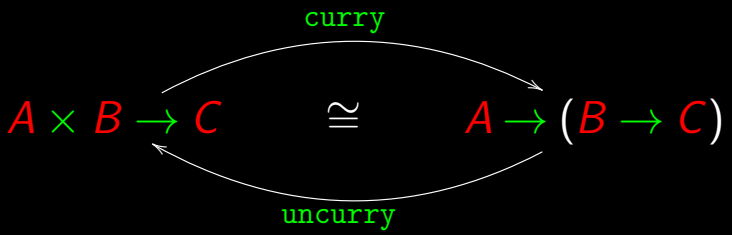
The type B^A

The type B^A

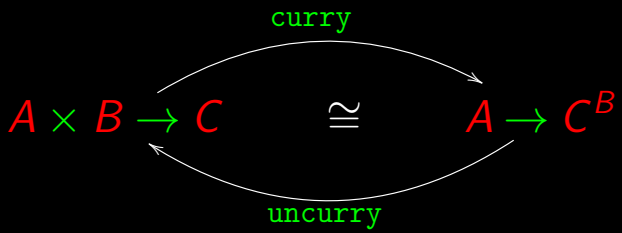
$$B^A = \{f \mid f : A \to B\}$$

NB: functions that map A to B

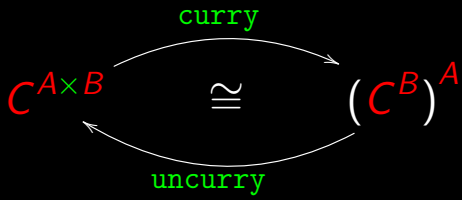
Isomorphism



Isomorphism



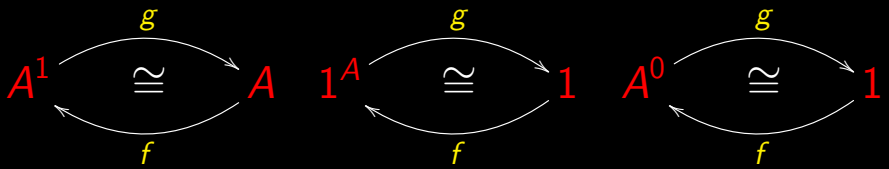
Isomorphism



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

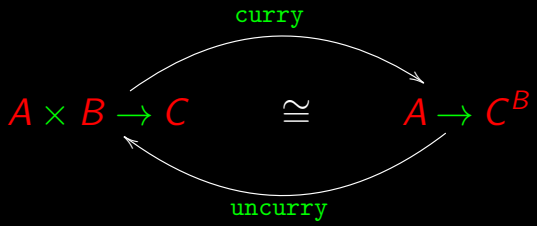
More isomorphisms

$a^1 = a$ $\qquad$ $1^a = 1$ $\qquad$ $a^0 = 1$

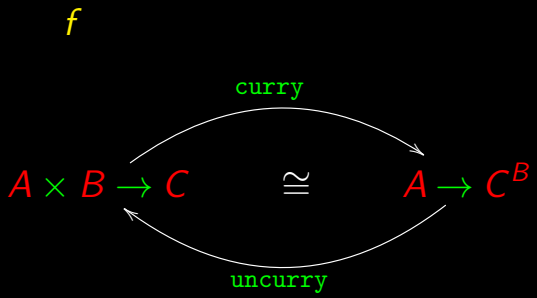


$$\begin{cases} f\ a = \underline{a} \\ g\ \underline{a} = a \end{cases} \qquad \begin{cases} f\ () = \mathbf{!} \\ g = \mathbf{!} \end{cases} \qquad \begin{cases} f\ () = \bot \\ g = \mathbf{!} \end{cases}$$

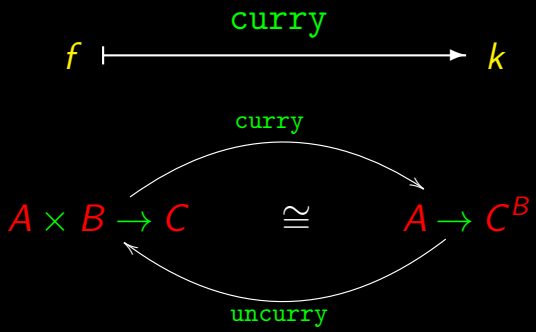
Curry | uncurry



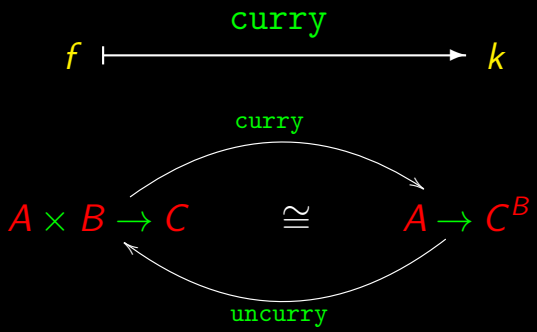
Curry | uncurry



Curry | uncurry

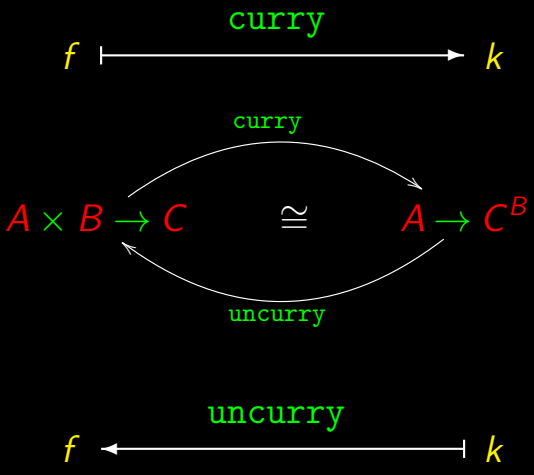


Curry | uncurry



k

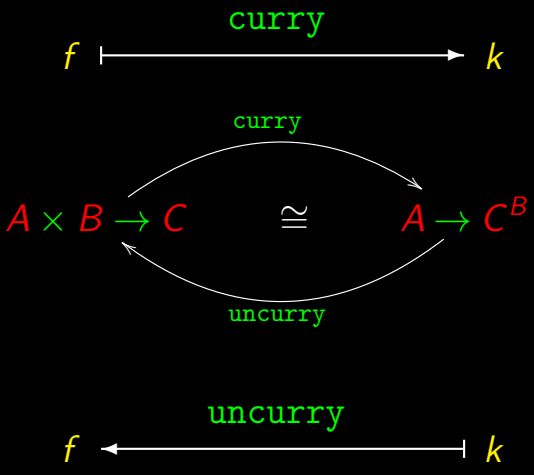
Curry | uncurry



Cálculo de Programas

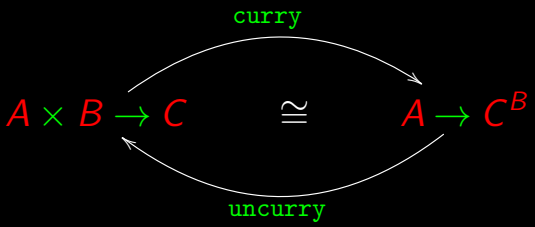
Class T06

Curry | uncurry



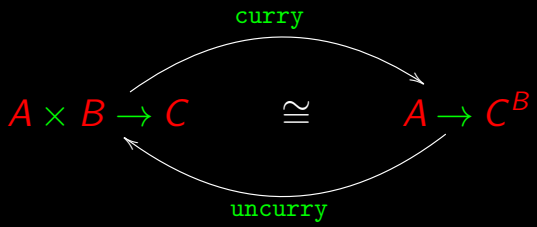
Curry | uncurry

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



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

Curry | uncurry



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

Curry | uncurry

$$
\begin{aligned}
& k = \texttt{curry}\ f \\
\Leftrightarrow \quad & \{\ \text{isomorphism (last slide)}\ \} \\
& \texttt{uncurry}\ k = f
\end{aligned}
$$

Curry | uncurry

$$
\begin{aligned}
& k = \texttt{curry}\ f \\
\Leftrightarrow \quad & \{\ \text{isomorphism (last slide)}\ \} \\
& \texttt{uncurry}\ k = f \\
\Leftrightarrow \quad & \{\ \text{add variables}\ \} \\
& \texttt{uncurry}\ k\ (a, b) = f\ (a, b)
\end{aligned}
$$

Curry | uncurry

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

$$
\begin{aligned}
& k = \texttt{curry}\ f \\
\Leftrightarrow \quad & \{\ \text{isomorphism (last slide)}\ \} \\
& \texttt{uncurry}\ k = f \\
\Leftrightarrow \quad & \{\ \text{add variables}\ \} \\
& \texttt{uncurry}\ k\ (a, b) = f\ (a, b) \\
\Leftrightarrow \quad & \{\ \text{definition of } \texttt{uncurry}\ k\ \} \\
& k\ a\ b = f\ (a, b)
\end{aligned}
$$

Curry | uncurry

$$
\begin{aligned}
& k = \texttt{curry}\ f \\
\Leftrightarrow\ & \quad \{\ \text{isomorphism (last slide)}\ \} \\
& \texttt{uncurry}\ k = f \\
\Leftrightarrow\ & \quad \{\ \text{add variables}\ \} \\
& \texttt{uncurry}\ k\ (a, b) = f\ (a, b) \\
\Leftrightarrow\ & \quad \{\ \text{definition of } \texttt{uncurry}\ k\ \} \\
& k\ a\ b = f\ (a, b)
\end{aligned}
$$

$$
\begin{aligned}
& (k\ a)\ b = f\ (a, b) \\
\Leftrightarrow\ & \quad \{\ \text{introduce } ap\ (f, x) = f\ x\ \} \\
& ap\ (k\ a, b) = f\ (a, b)
\end{aligned}
$$

Curry | uncurry

$$
\begin{aligned}
& k = \texttt{curry}\ f \\
\Leftrightarrow\ & \quad \{\ \text{isomorphism (last slide)}\ \} \\
& \texttt{uncurry}\ k = f \\
\Leftrightarrow\ & \quad \{\ \text{add variables}\ \} \\
& \texttt{uncurry}\ k\ (a, b) = f\ (a, b) \\
\Leftrightarrow\ & \quad \{\ \text{definition of } \texttt{uncurry}\ k\ \} \\
& k\ a\ b = f\ (a, b)
\end{aligned}
$$

$$
\begin{aligned}
& (k\ a)\ b = f\ (a, b) \\
\Leftrightarrow\ & \quad \{\ \text{introduce } ap\ (f, x) = f\ x\ \} \\
& ap\ (k\ a, b) = f\ (a, b) \\
\Leftrightarrow\ & \quad \{\ \text{add } id\ \} \\
& ap\ (k\ a, id\ b) = f\ (a, b)
\end{aligned}
$$

Curry | uncurry

$$
\begin{aligned}
& k = \texttt{curry}\ f \\
\Leftrightarrow\ & \quad \{\ \text{isomorphism (last slide)}\ \} \\
& \texttt{uncurry}\ k = f \\
\Leftrightarrow\ & \quad \{\ \text{add variables}\ \} \\
& \texttt{uncurry}\ k\ (a, b) = f\ (a, b) \\
\Leftrightarrow\ & \quad \{\ \text{definition of } \texttt{uncurry}\ k\ \} \\
& k\ a\ b = f\ (a, b)
\end{aligned}
$$

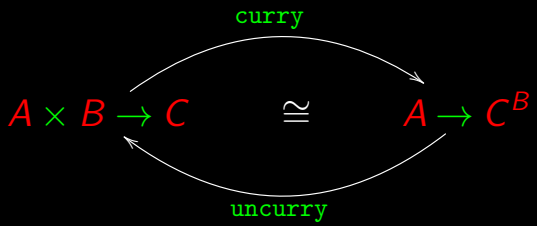
$$
\begin{aligned}
& (k\ a)\ b = f\ (a, b) \\
\Leftrightarrow\ & \quad \{\ \text{introduce } ap\ (f, x) = f\ x\ \} \\
& ap\ (k\ a, b) = f\ (a, b) \\
\Leftrightarrow\ & \quad \{\ \text{add } id\ \} \\
& ap\ (k\ a, id\ b) = f\ (a, b) \\
\Leftrightarrow\ & \quad \{\ \text{composition and } \times\ \} \\
& (ap \cdot (k \times id))\ (a, b) = f\ (a, b)
\end{aligned}
$$

Curry | uncurry

$$
\begin{aligned}
& k = \texttt{curry}\ f \\
\Leftrightarrow & \quad \{\ \text{isomorphism (last slide)}\ \} \\
& \texttt{uncurry}\ k = f \\
\Leftrightarrow & \quad \{\ \text{add variables}\ \} \\
& \texttt{uncurry}\ k\ (a, b) = f\ (a, b) \\
\Leftrightarrow & \quad \{\ \text{definition of } \texttt{uncurry}\ k\ \} \\
& k\ a\ b = f\ (a, b)
\end{aligned}
$$

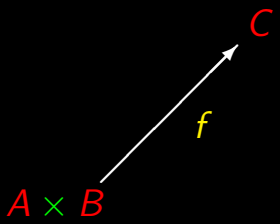
$$
\begin{aligned}
& (k\ a)\ b = f\ (a, b) \\
\Leftrightarrow & \quad \{\ \text{introduce } ap\ (f, x) = f\ x\ \} \\
& ap\ (k\ a, b) = f\ (a, b) \\
\Leftrightarrow & \quad \{\ \text{add } id\ \} \\
& ap\ (k\ a, id\ b) = f\ (a, b) \\
\Leftrightarrow & \quad \{\ \text{composition and } \times\ \} \\
& (ap \cdot (k \times id))\ (a, b) = f\ (a, b) \\
\Leftrightarrow & \quad \{\ \text{drop variables}\ \} \\
& ap \cdot (k \times id) = f
\end{aligned}
$$

Universal property

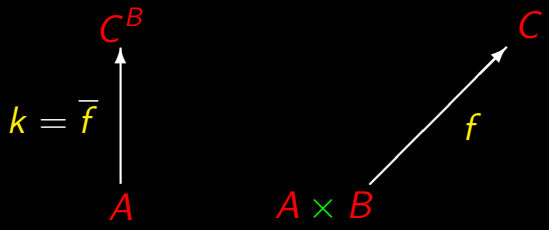


$$k = \texttt{curry}\ f \ \Leftrightarrow\ ap \cdot (k \times id)$$

Diagrams

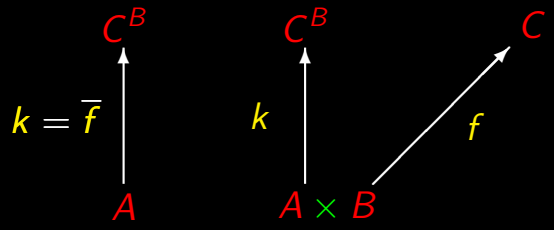


Diagrams



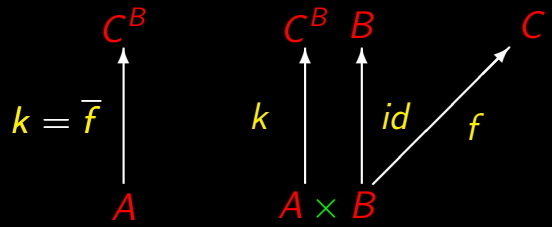
$\overline{f}$ abbreviates `curry` f

Diagrams



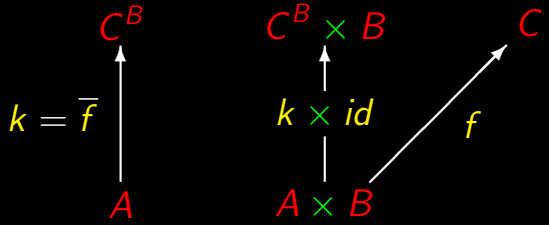
$\overline{f}$ abbreviates `curry` f

Diagrams

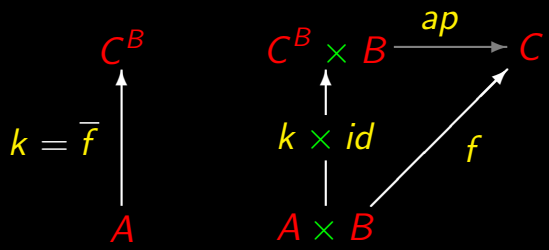


$\overline{f}$ abbreviates `curry` f

Universal property

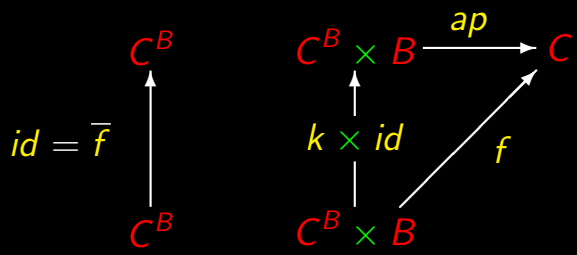


Universal property



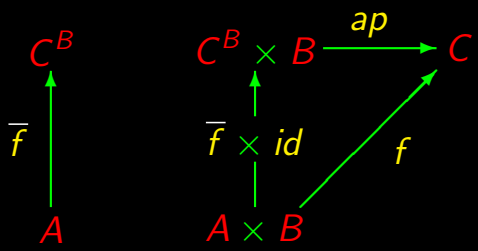
$$k = \overline{f} \;\Leftrightarrow\; ap \cdot (k \times id)\; k = f$$

Reflexion ($k := id$)

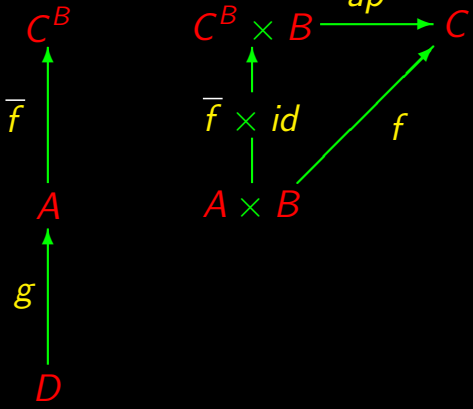


$$id = \overline{f} \Leftrightarrow ap = f$$

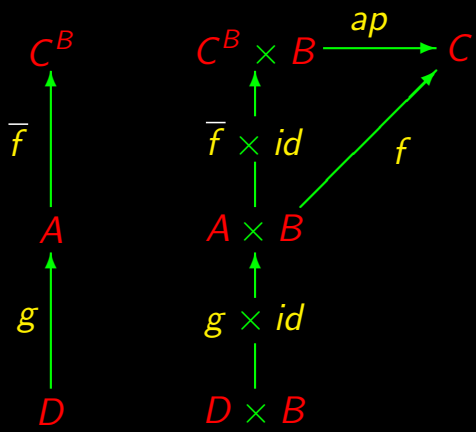
Fusion



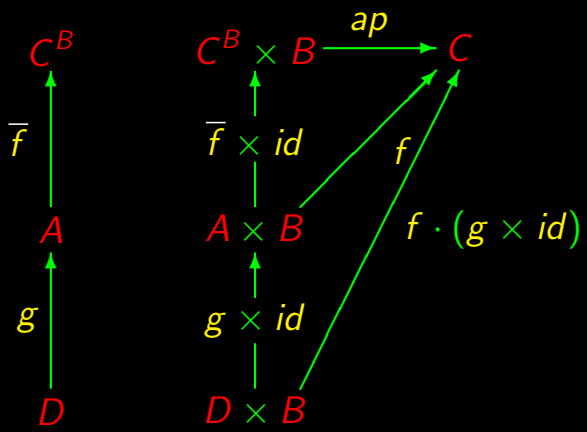
Fusion



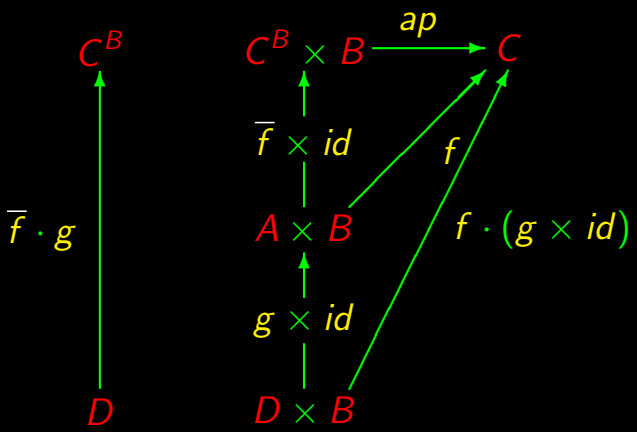
Fusion


C^B
$\overline{f}$
A
g
D
$C^B \times B$
ap
C
$\overline{f} \times id$
f
$A \times B$
$g \times id$
$D \times B$

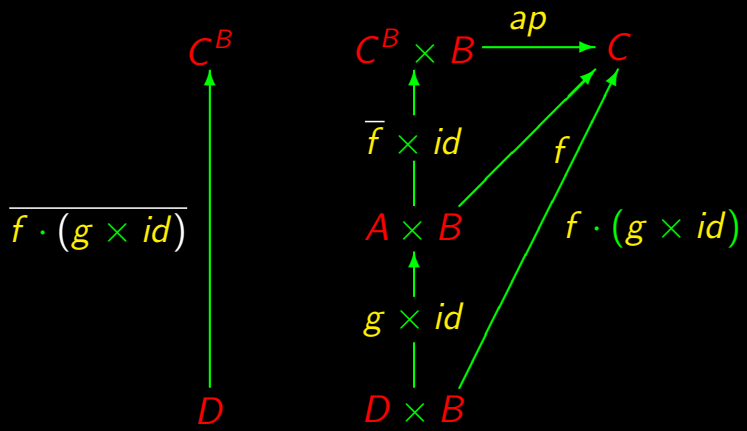
Fusion


C^B
$\overline{f}$
A
g
D
$C^B \times B$
ap
C
$\overline{f} \times id$
f
$A \times B$
$f \cdot (g \times id)$
$g \times id$
$D \times B$

Fusion

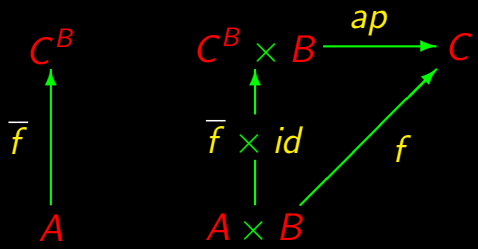

C^B
$\overline{f} \cdot g$
D
$C^B \times B$
ap
C
$\overline{f} \times id$
f
$A \times B$
$f \cdot (g \times id)$
$g \times id$
$D \times B$

Fusion

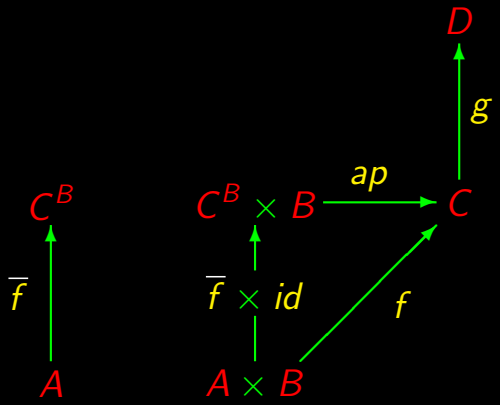


$$\overline{f} \cdot g = \overline{f \cdot (g \times id)}$$

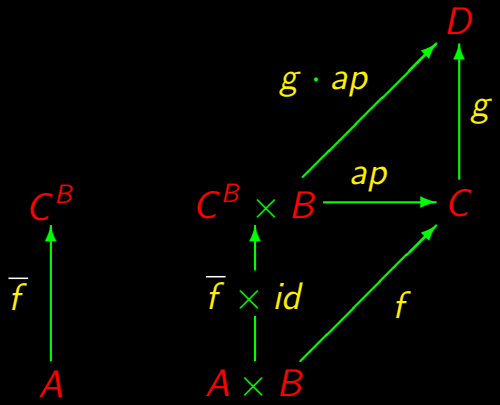
Absorption (exponentials)



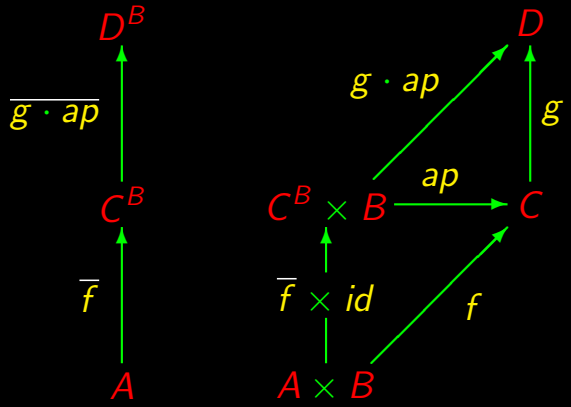
Absorption (exponentials)



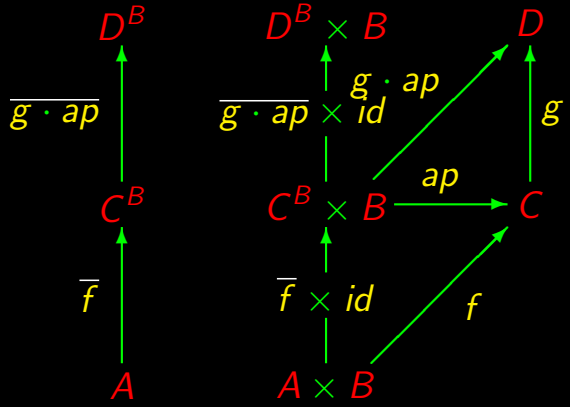
Absorption (exponentials)



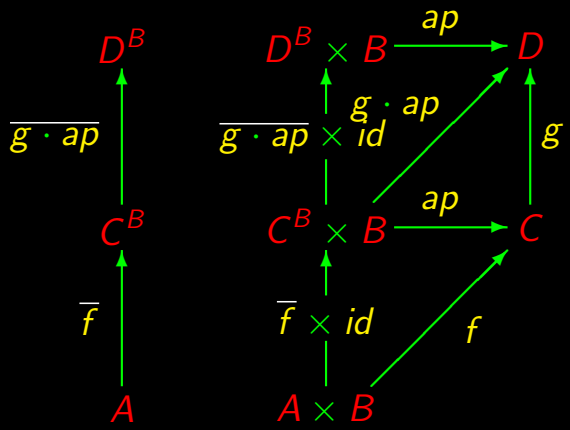
Absorption (exponentials)



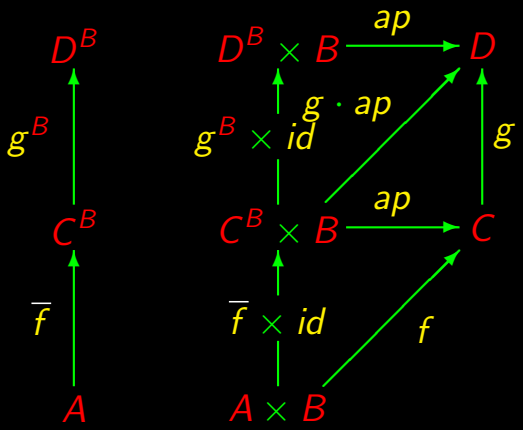
Absorption (exponentials)



Absorption (exponentials)

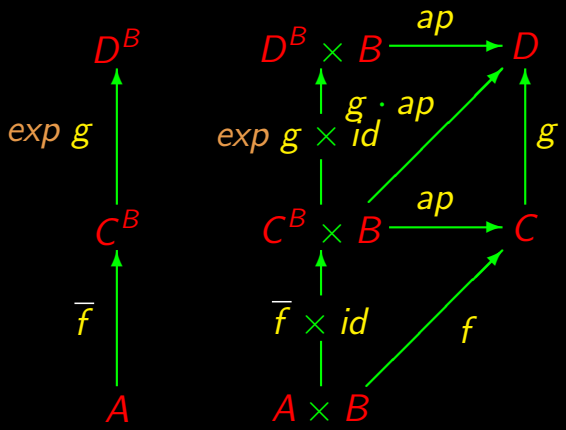


Absorption



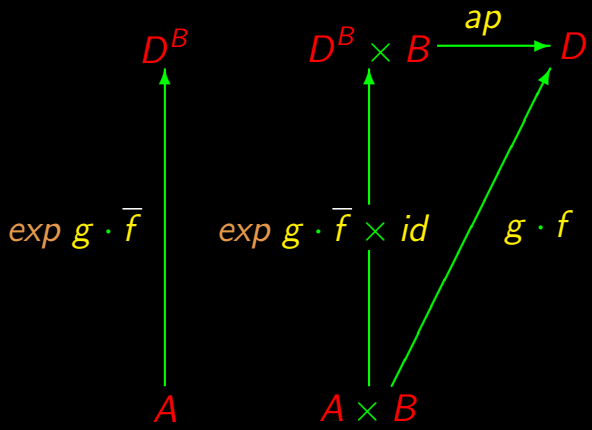
$$g^B = \overline{g \cdot ap}$$

Absorption

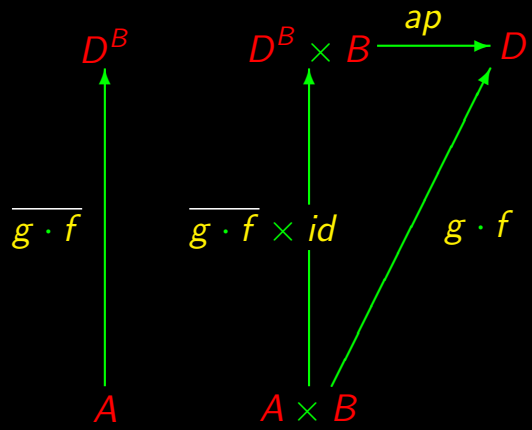


$$exp\ g = \overline{g \cdot ap}$$

Absorption

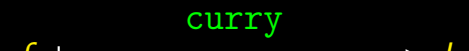

D^B
$D^B \times B$
ap
D
$exp\ g \cdot \overline{f}$
$exp\ g \cdot \overline{f} \times id$
$g \cdot f$
A
$A \times B$

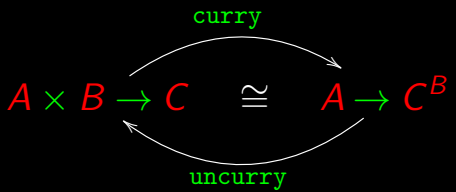
Absorption

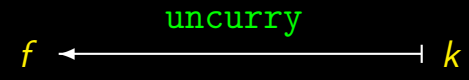


$$\overline{g \cdot f} = exp\ g \cdot \overline{f}$$

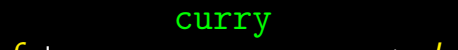
Curry | uncurry

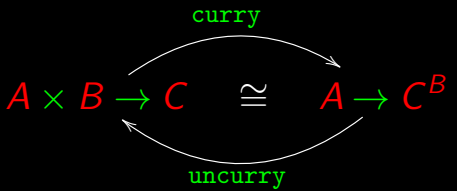


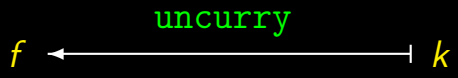




Curry | uncurry



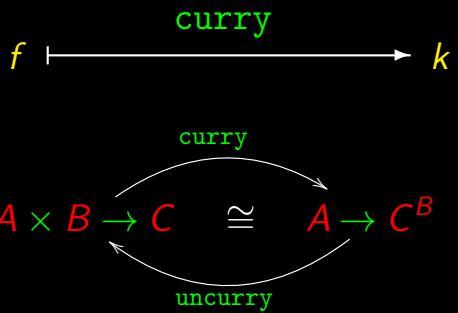




Abbreviations:

$$\texttt{curry}\ f = \overline{f}$$

Curry | uncurry

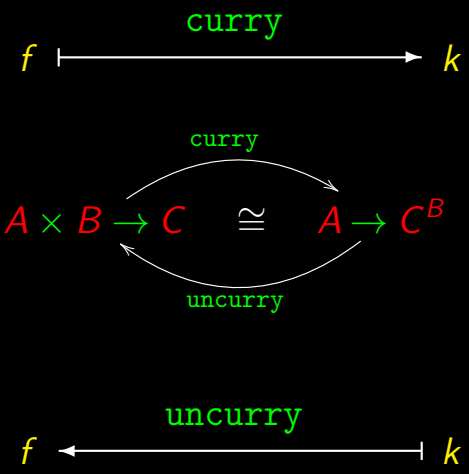


Abbreviations:

$$\texttt{curry}\ f = \overline{f}$$

$$\texttt{uncurry}\ f = \widehat{f}$$

Curry | uncurry



Abbreviations:

$$\texttt{curry}\ f = \overline{f}$$
$$\texttt{uncurry}\ f = \widehat{f}$$

Isomorphisms:

$$f = g \Leftrightarrow \overline{f} = \overline{g}$$
$$k = h \Leftrightarrow \widehat{k} = \widehat{h}$$

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
$[f, g]$, $f + g$, $p \to f\ , g$	alternation

Summary

$f \cdot g$	sequential
$\langle f, g \rangle$, $f \times g$	parallel
$[f, g]$, $f + g$, $p \to f\ , g$	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
$\overline{f}$, $exp\ f$	higher order

Summary

$f \cdot g$	sequential
$\langle f, g \rangle$, $f \times g$	parallel
$[f, g]$, $f + g$, $p \rightarrow f\ , g$	alternation
$\overline{f}$, $exp\ f$	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?

$$\begin{cases} a \times 0 = 0 \\ a \times 1 = a \\ a \times (b + c) = a \times b + a \times c \end{cases}$$

Where do programs come from?

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

Where do programs come from?

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

Where do programs come from?

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

Where do programs come from?

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

Where do programs come from?

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

Where do programs come from?

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

Where do programs come from?

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

Where do programs come from?

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

Where do programs come from?

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

What is a program?

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

What is a program?

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

What is a program?

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

What is a program?

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

What is a program?

$$\begin{cases} (a\times)\ (\underline{0}\ x) = \underline{0}\ x \\ \\ (a\times)\ (b+1) = (a+)\ ((a\times)\ b) \end{cases}$$

What is a program?

$$\begin{cases} (a\times)\ (\underline{0}\ x) = \underline{0}\ x \\ \\ (a\times)\ (succ\ b) = (a+)\ ((a\times)\ b) \end{cases}$$

What is a program?

$$\begin{cases} ((a\times)\cdot \underline{0})\ x = \underline{0}\ x \\ \\ (a\times)\ (succ\ b) = (a+)\ ((a\times)\ b) \end{cases}$$

What is a program?

$$\begin{cases} ((a\times)\cdot\underline{0})\ x = \underline{0}\ x \\ \\ ((a\times)\cdot \mathit{succ})\ b = (a+)\ ((a\times)\ b) \end{cases}$$

What is a program?

$$\begin{cases} ((a\times)\cdot\underline{0})\ x = \underline{0}\ x \\ \\ ((a\times)\cdot succ)\ b = ((a+)\cdot(a\times))\ b \end{cases}$$

What is a program?

$$\begin{cases} (a\times)\cdot\underline{0} = \underline{0} \\ \\ ((a\times)\cdot succ)\ b = ((a+)\cdot(a\times))\ b \end{cases}$$

What is a program?

$$\begin{cases} (a\times)\cdot \underline{0} = \underline{0} \\ \\ (a\times)\cdot succ = (a+)\cdot(a\times) \end{cases}$$

What is a program?

$$\begin{cases} (a\times)\cdot \underline{0} = \underline{0} \\ \\ (a\times)\cdot succ = (a+)\cdot(a\times) \end{cases}$$

What is a program?

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

What is a program?

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

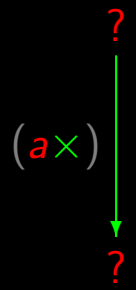
What is a program?

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

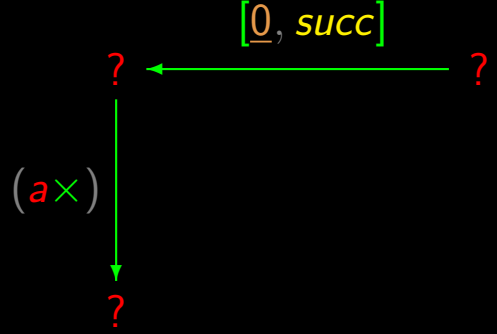
What is a program?

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

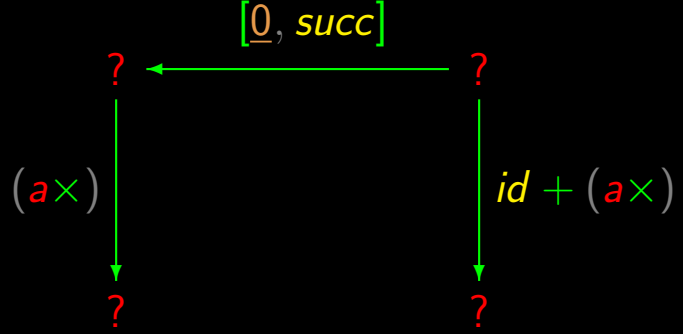
$$(a\times)\cdot[\underline{0}, succ] = [\underline{0}, (a+)]\cdot(id + (a\times))$$



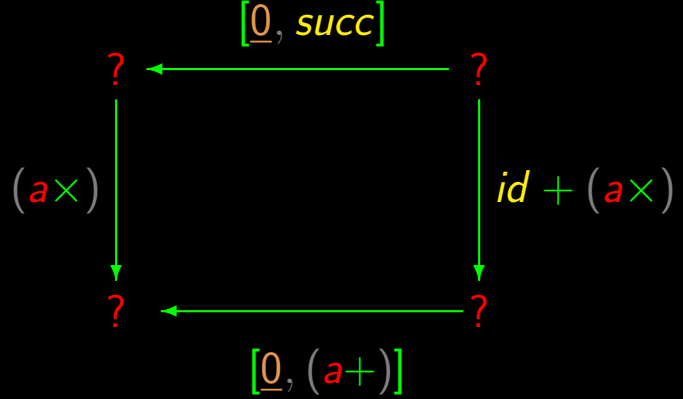
$$(a\times)\cdot[\underline{0}, succ] = [\underline{0}, (a+)]\cdot(id + (a\times))$$



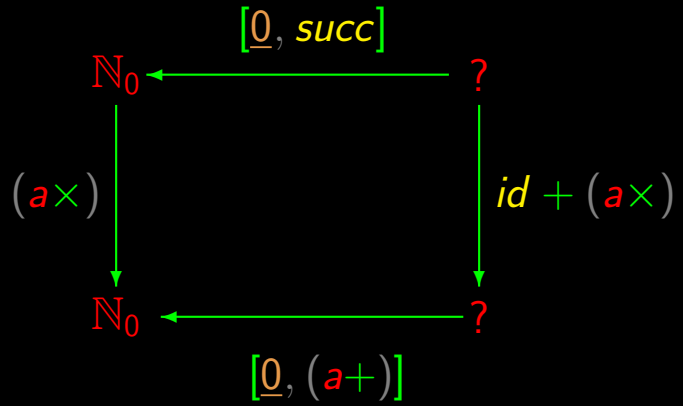
$$(a\times)\cdot[\underline{0}, succ] = [\underline{0}, (a+)]\cdot(id + (a\times))$$



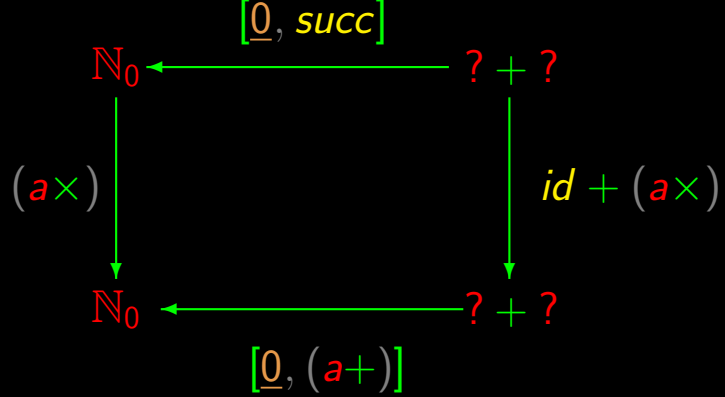
$$(a\times)\cdot[\underline{0}, succ] = [\underline{0}, (a+)]\cdot(id + (a\times))$$



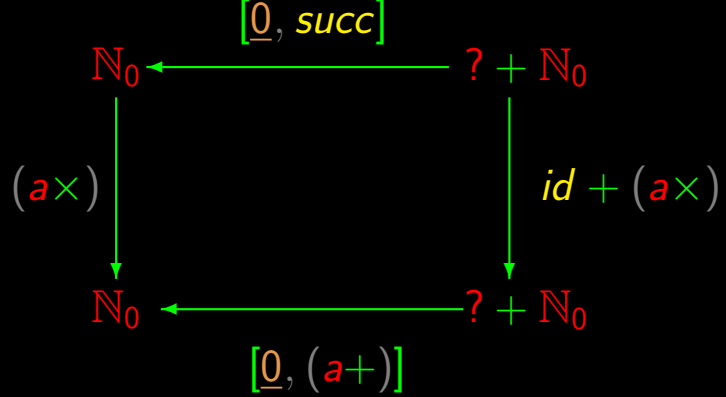
A program is a “rectangle”



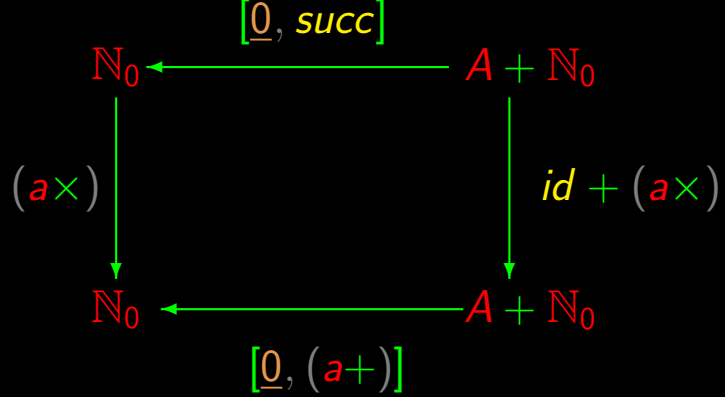
A program is a “rectangle”



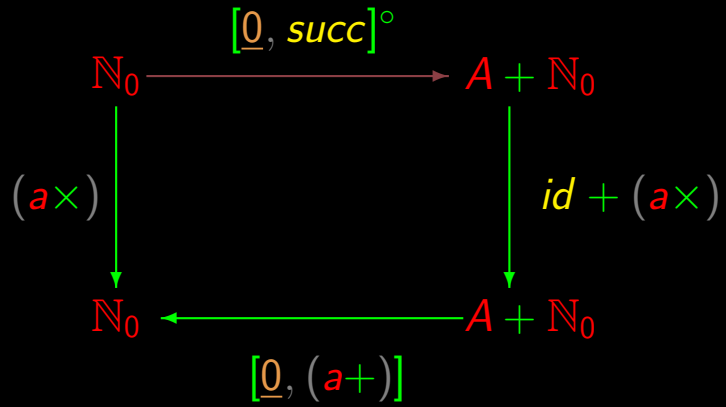
A program is a “rectangle”



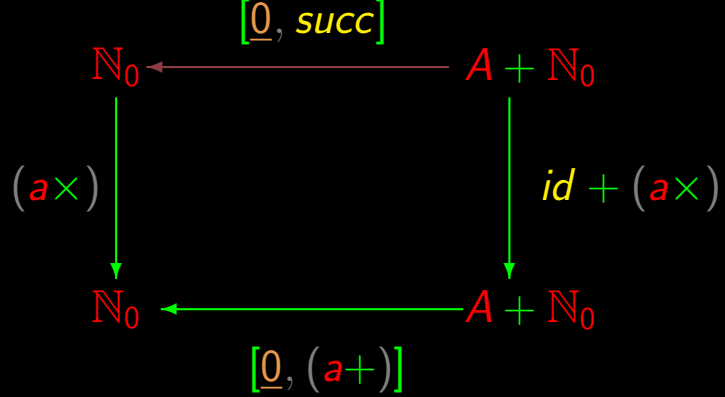
A program is a “rectangle”



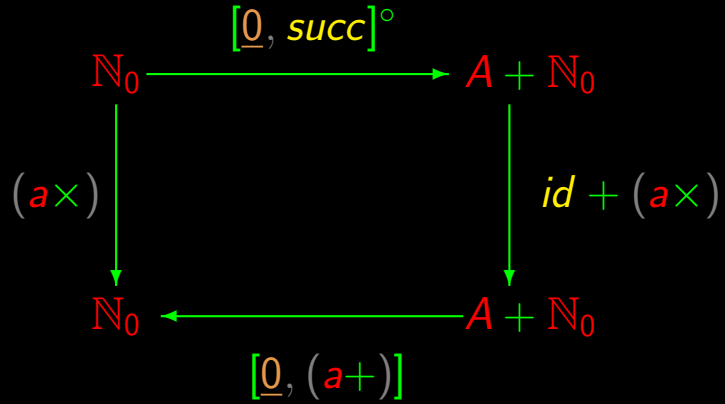
Does $[\underline{0}, succ]^\circ$ exist?



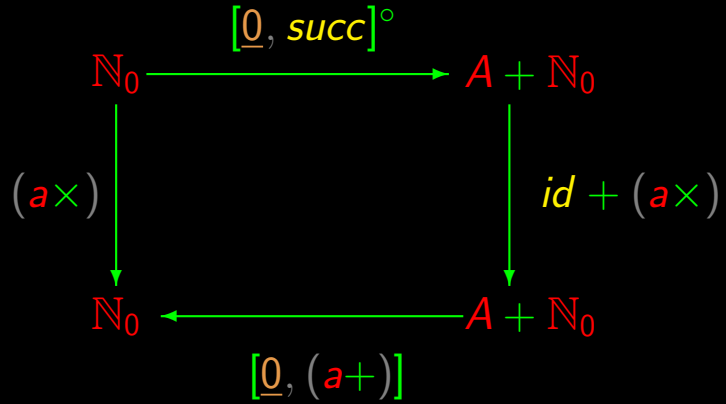
Does $[\underline{0}, succ]^\circ$ exist?



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



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



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

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

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

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

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

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

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

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

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

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

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

$$
\begin{array}{rl}
 & \mathbf{out} \cdot [\underline{0}, succ] = id \\
\Leftrightarrow & \quad \{ \ +\text{-fusion} \ \} \\
 & [\mathbf{out} \cdot \underline{0}, \mathbf{out} \cdot succ] = id \\
\Leftrightarrow & \quad \{ \ \text{universal-}+ \ \} \\
 & \left\{ \begin{array}{l} \mathbf{out} \cdot \underline{0} = i_1 \\ \mathbf{out} \cdot succ = i_2 \end{array} \right.
\end{array}
$$

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

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

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

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

$\Leftrightarrow$ { introduce variables }

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

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

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

$\Leftrightarrow$ { introduce variables }

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

$\Leftrightarrow$ { simplificar }

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

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

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

$\Leftrightarrow$ { introduce variables }

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

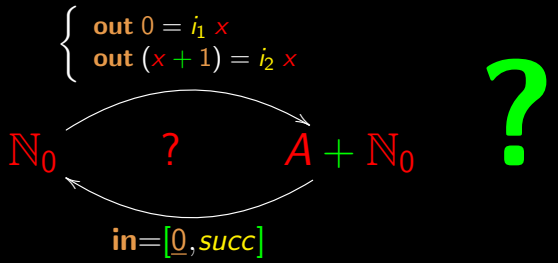
$\Leftrightarrow$ { simplificar }

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

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

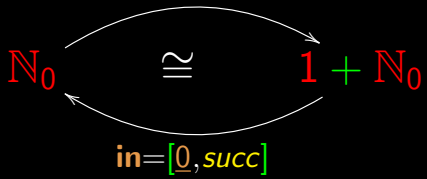
?

Problem



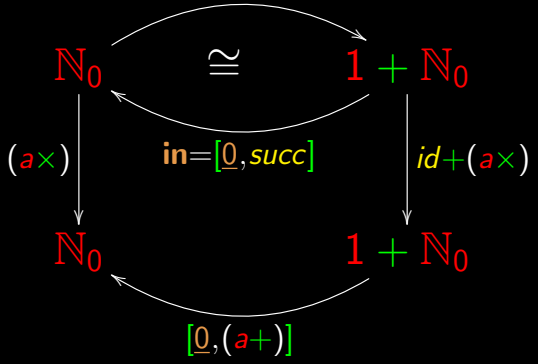
Solution

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

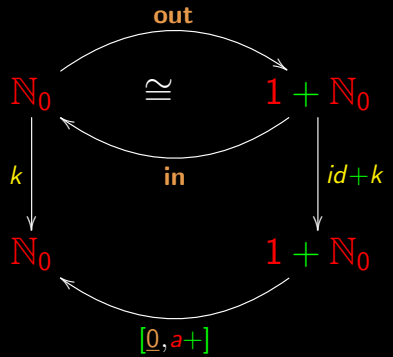


Solution

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

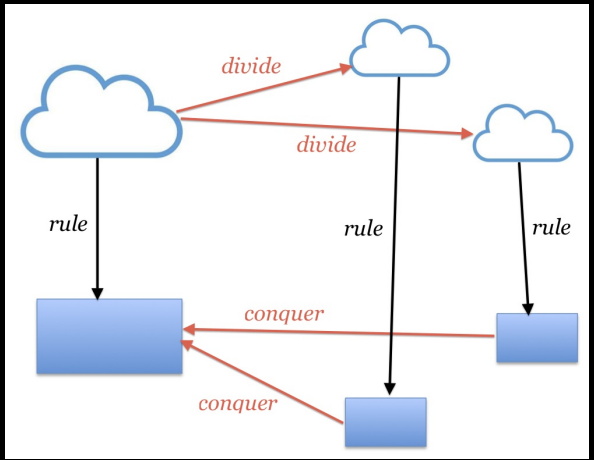


Diagrama

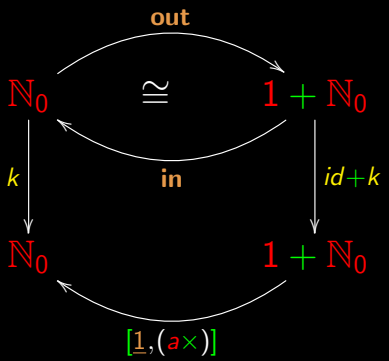


$$(a\times) = k \textbf{ where } k = [\underline{0}, (a+)] \cdot (id + k) \cdot \textbf{out}$$

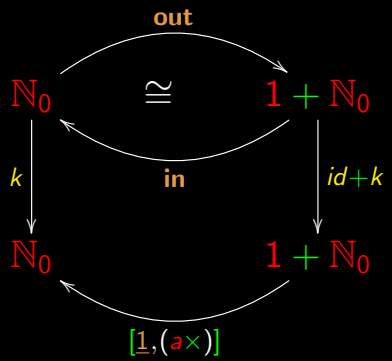
Anticipation of what is to come...



Variations in diagram



Variations in diagram

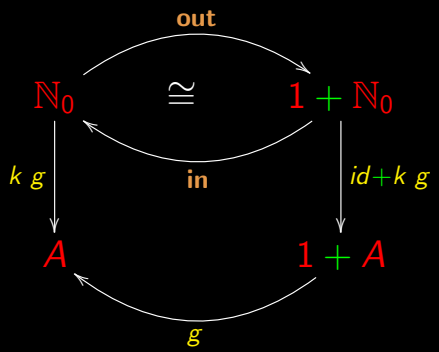


$$a^b = k \text{ where } k = [\underline{1}, (a\times)] \cdot (id + k) \cdot \text{out}$$

Generalizing the diagram

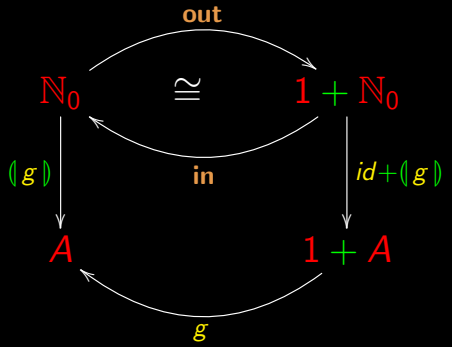
$$\begin{cases} a^0 = 1 \\ a^{b+1} = a \times a^b \end{cases}$$

Generalizing the diagram



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

Generalizing the diagram



$$(\!|g|\!) = g \cdot (id + (\!|g|\!)) \cdot \mathbf{out}$$

Catamorphism

$$(\!|\, g \,|\!)$$

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

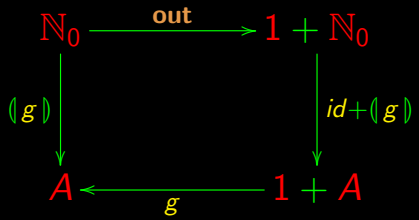
Catamorphism

$(\!|\, g \,|\!)$

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

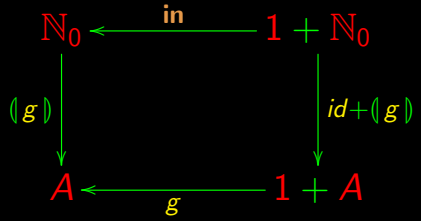
"Downwards transformation"

Definition



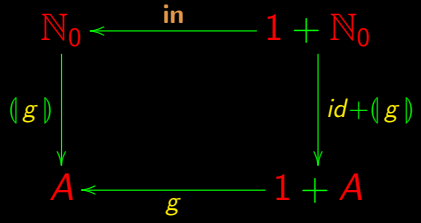
$$(\!|g|\!) = g \cdot (id + (\!|g|\!)) \cdot \mathbf{out}$$

Property



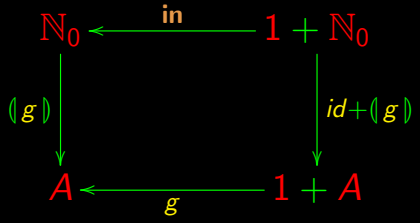
$$(\!|\, g \,|\!) \cdot \mathsf{in} = g \cdot (id + (\!|\, g \,|\!))$$

Property



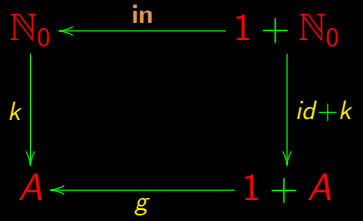
$$k = (\!|g|\!) \Rightarrow k \cdot \mathsf{in} = g \cdot (id + k)$$

Uniqueness



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

Universal property



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

Universal-$\times$:

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

Universal-$\times$:

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

Universal-$+$:

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

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:

$$k = \overline{f} \;\Leftrightarrow\; ap \cdot (k \times id) = f$$

Universal-$\times$:

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

Universal-$+$:

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

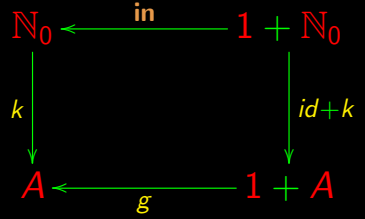
Universal-expo:

$$k = \overline{f} \;\Leftrightarrow\; ap \cdot (k \times id) = f$$

Cata-Universal:

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

Cata-universal



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

Cata-cancellation

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

Cata-cancellation

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

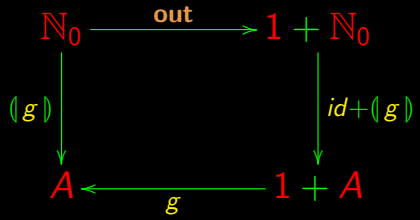
Cata-cancellation

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

Cata-definition

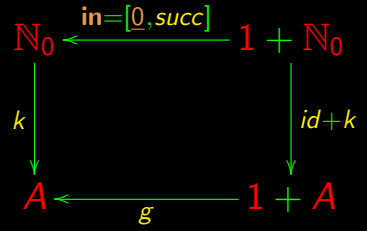
$$(\!|\, g \,|\!) = g \cdot (id + (\!|\, g \,|\!)) \cdot \mathbf{out}$$

Cata-definition



$$(\!|g|\!) = g \cdot (id + (\!|g|\!)) \cdot \mathbf{out}$$

Cata-universal



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

Cata-reflexion

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

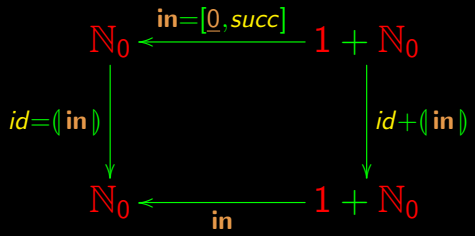
Cata-reflexion

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

Cata-reflexion

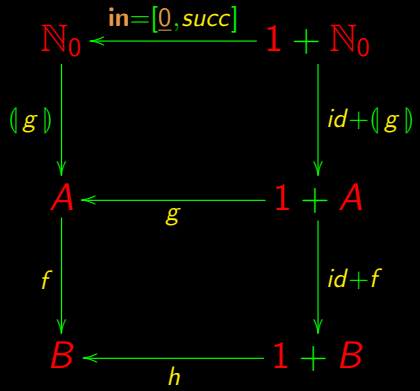
$$id = (\!|\, g \,|\!) \Leftrightarrow \mathbf{in} = g$$

Cata-reflexion

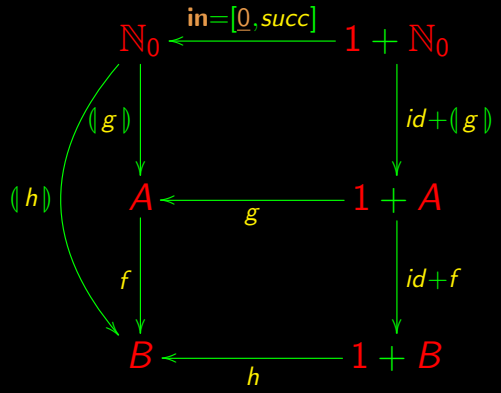


$$(|\,\mathsf{in}\,|) = id$$

Cata-fusion



Cata-fusion



Cata-fusion

$$f \cdot (\!|\, g \,|\!) = (\!|\, h \,|\!)$$

Cata-fusion

$$
\begin{aligned}
& f \cdot (\!|\, g \,|\!) = (\!|\, h \,|\!) \\
\Leftrightarrow \quad & \{ \text{ Cata-universal for } k = f \cdot (\!|\, g \,|\!) \} \\
& f \cdot (\!|\, g \,|\!) \cdot \mathbf{in} = h \cdot (id + f \cdot (\!|\, g \,|\!))
\end{aligned}
$$

Cata-fusion

$$
\begin{aligned}
& f \cdot (\!|\, g \,|\!) = (\!|\, h \,|\!) \\
\Leftrightarrow \quad & \{\ \text{Cata-universal for } k = f \cdot (\!|\, g \,|\!)\ \} \\
& f \cdot (\!|\, g \,|\!) \cdot \mathbf{in} = h \cdot (id + f \cdot (\!|\, g \,|\!)) \\
\Leftrightarrow \quad & \{\ \text{Cata-cancellation}\ \} \\
& f \cdot g \cdot (id + (\!|\, g \,|\!)) = h \cdot (id + f \cdot (\!|\, g \,|\!))
\end{aligned}
$$

Cata-fusion

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

Cata-fusion

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

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

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

Cata-fusion

$$
\begin{aligned}
& f \cdot g \cdot (id + (\!| g |\!)) = h \cdot (id \cdot id + f \cdot (\!| g |\!)) \\
\Leftrightarrow & \quad \{ \text{ functor-+ } \} \\
& f \cdot g \cdot (id + (\!| g |\!)) = h \cdot (id + f) \cdot (id + (\!| g |\!)) \\
\Leftarrow & \quad \{ \text{ Leibniz } \} \\
& f \cdot g = h \cdot (id + f)
\end{aligned}
$$

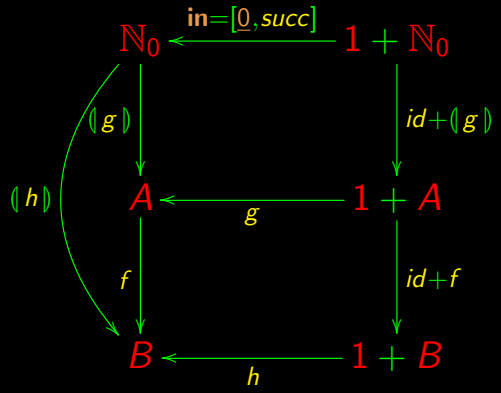
Cata-fusion

$$
\begin{aligned}
& f \cdot g \cdot (id + (\!| g |\!)) = h \cdot (id \cdot id + f \cdot (\!| g |\!)) \\
\Leftrightarrow & \quad \{ \text{ functor-+ } \} \\
& f \cdot g \cdot (id + (\!| g |\!)) = h \cdot (id + f) \cdot (id + (\!| g |\!)) \\
\Leftarrow & \quad \{ \text{ Leibniz } \} \\
& f \cdot g = h \cdot (id + f)
\end{aligned}
$$

Summing up:

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

Cata-fusion



What is a program?

What is a program?

Programs are “rectangles” 

What is a program?

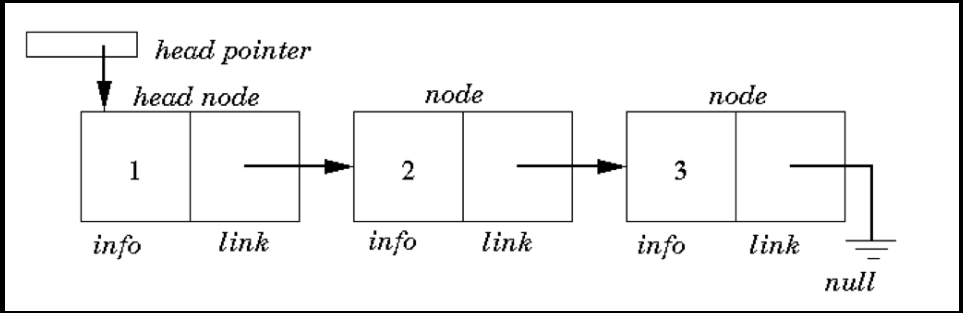
Programs are “rectangles” 

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

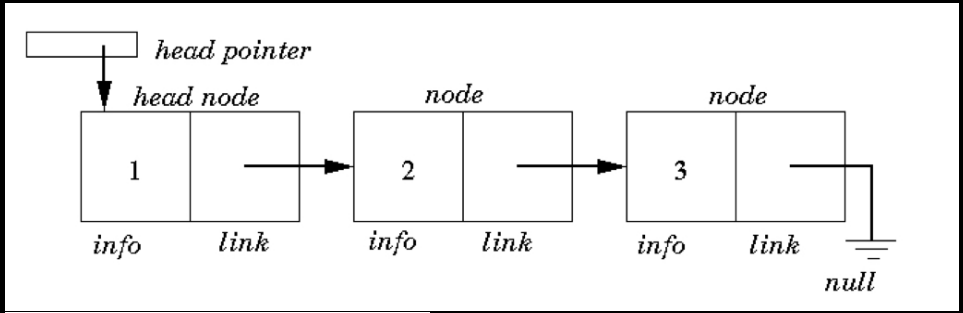
Cálculo de Programas

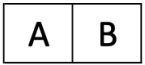
Class T07

Linked list


head pointer
head node
node
node
1
2
3
info
link
info
link
info
link
null

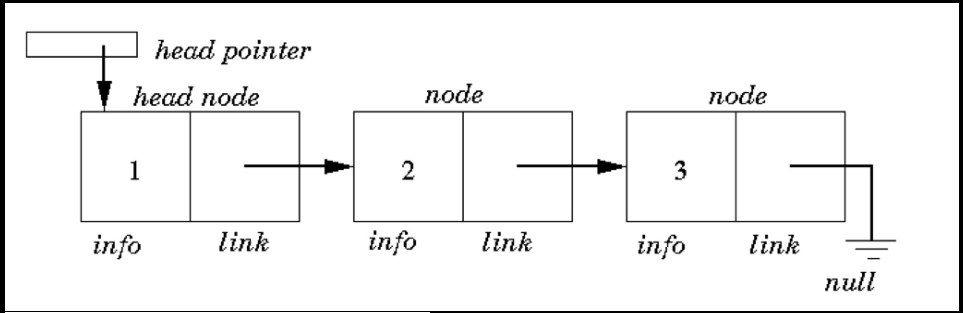
Linked list

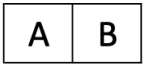




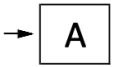
$A \times B$

Linked list



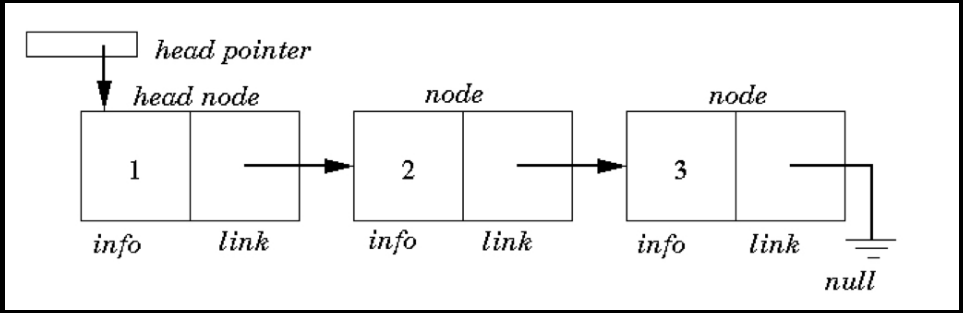


$A \times B$



$1 + A$

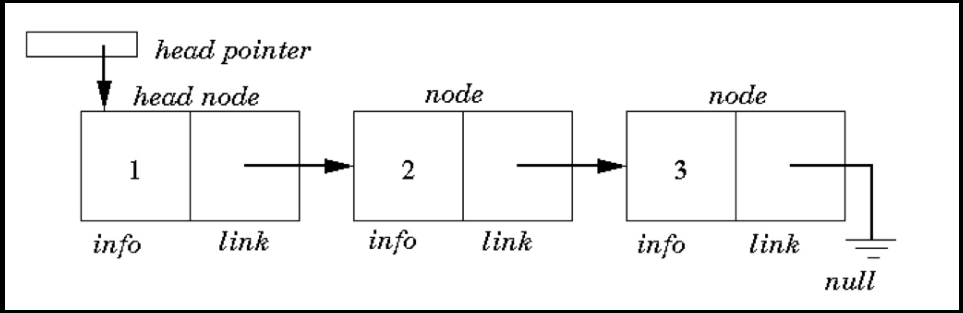
Linked list



“pointer”

$L = 1 + N$

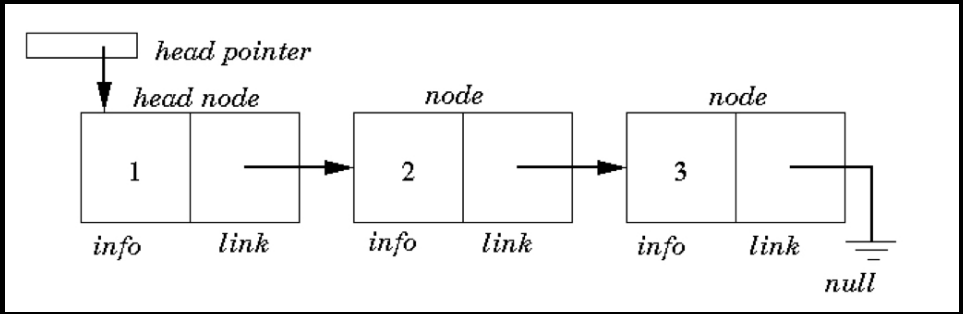
Linked list



“pointer” $L = 1 + N$

“node” $N = \mathbb{N}_0 \times L$

Linked list

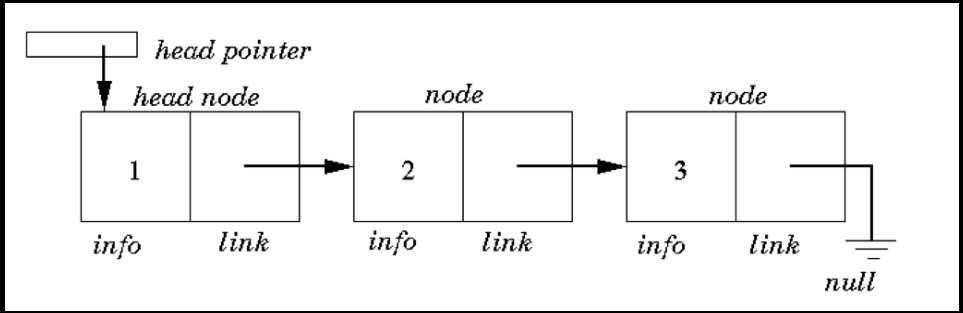


Substitution

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

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

Linked list



In fact

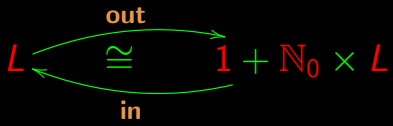
$$L \cong 1 + \mathbb{N}_0 \times L$$

Linked list

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

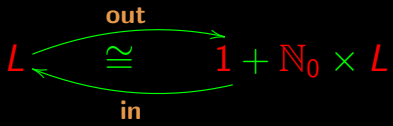
Linked list

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



Linked list

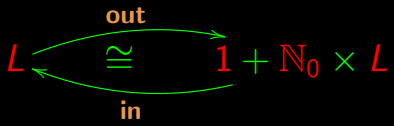
$$L \cong 1 + \mathbb{N}_0 \times L$$



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

Linked list

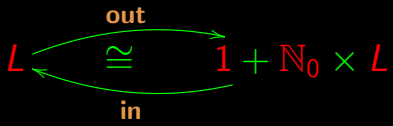
$$L \cong 1 + \mathbb{N}_0 \times L$$



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

Linked list

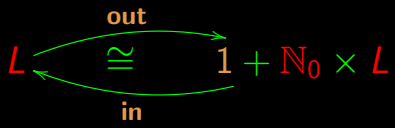
$$L \cong 1 + \mathbb{N}_0 \times L$$



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

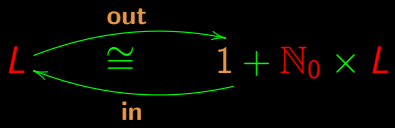
$$\mathbf{out} \cdot \mathbf{in} = id$$

Linked list



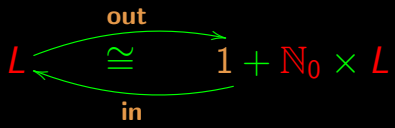
$$\begin{cases} \mathbf{out} \cdot \mathbf{in} = id \\ \mathbf{in} = [nil, cons] \end{cases}$$

Linked list



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

Linked list



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

Length of a list

Recall list **concatenation** $x +\!\!+ y$

Length of a list

Recall list **concatenation** $x +\!\!+ y$ such that

$$[\,] +\!\!+ x = x$$

Length of a list

Recall list **concatenation** $x +\!\!+ y$ such that

$$[\,] +\!\!+ x = x$$
$$[a] +\!\!+ x = a : x$$

Length of a list

Recall list **concatenation** $x +\!\!+ y$ such that

$$[\,] +\!\!+ x = x$$
$$[a] +\!\!+ x = a : x$$

Properties:

Length of a list

Recall list **concatenation** $x +\!\!+ y$ such that

$$[\,] +\!\!+ x = x$$
$$[a] +\!\!+ x = a : x$$

Properties:

$$\mathit{length}\ [\,] = 0$$

Length of a list

Recall list **concatenation** $x +\!\!+ y$ such that

$$[\,] +\!\!+ x = x$$
$$[a] +\!\!+ x = a : x$$

Properties:

$$\mathit{length}\ [\,] = 0$$
$$\mathit{length}\ [a] = 1$$

Length of a list

Recall list **concatenation** $x +\!\!+ y$ such that

$$[\,] +\!\!+ x = x$$
$$[a] +\!\!+ x = a : x$$

Properties:

$$\mathit{length}\ [\,] = 0$$
$$\mathit{length}\ [a] = 1$$
$$\mathit{length}\ (x +\!\!+ y) = \mathit{length}\ x + \mathit{length}\ y$$

Length of a list

$$
\begin{aligned}
&\mathit{length}\ [\,] = 0 \\
&\mathit{length}\ [a] = 1 \\
&\mathit{length}\ ([a] +\!\!+\ y) = \mathit{length}\ [a] + \mathit{length}\ y
\end{aligned}
$$

Length of a list

$$length\ [\,] = 0$$
$$length\ [a] = 1$$
$$length\ ([a] +\!\!+\ y) = length\ [a] + length\ y$$

$$length\ (a : y) = 1 + length\ y$$

Length of a list

$$length\ [\,] = 0$$
$$length\ (a : y) = 1 + length\ y$$

Length of a list

$$length\ [\,] = 0$$
$$length\ (a : y) = 1 + length\ y$$

$$(length \cdot nil)\ \_ = \underline{0}\ \_$$
$$(length \cdot cons)\ (a, y) = 1 + length\ (\pi_2\ (a, y))$$

Length of a list

$$\mathit{length}\ [\,] = 0$$
$$\mathit{length}\ (a : y) = 1 + \mathit{length}\ y$$

$$(\mathit{length} \cdot \mathit{nil})\ \_ = \underline{0}\ \_$$
$$(\mathit{length} \cdot \mathit{cons})\ (a, y) = 1 + \mathit{length}\ (\pi_2\ (a, y))$$

$$\mathit{length} \cdot \mathit{nil} = \underline{0}$$
$$\mathit{length} \cdot \mathit{cons} = \mathit{succ} \cdot \mathit{length} \cdot \pi_2$$

Length of a list

$$\mathit{length} \cdot \mathit{nil} = \underline{0}$$
$$\mathit{length} \cdot \mathit{cons} = \mathit{succ} \cdot \mathit{length} \cdot \pi_2$$

Length of a list

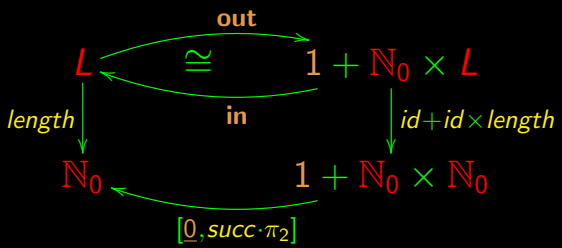
$$length \cdot nil = \underline{0}$$
$$length \cdot cons = succ \cdot length \cdot \pi_2$$

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

Length of a list

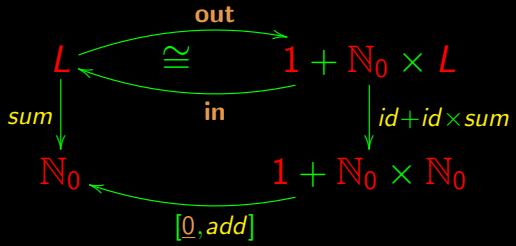
$$length \cdot nil = \underline{0}$$
$$length \cdot cons = succ \cdot length \cdot \pi_2$$

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



$$\begin{cases} \mathbf{in} = [nil, cons] \\ \mathbf{out} = \mathbf{in}^{\circ} \end{cases}$$

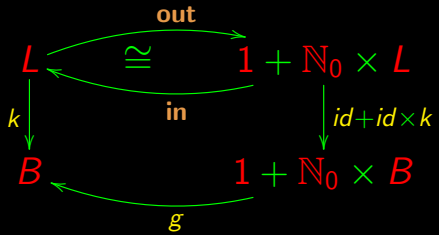
List sum



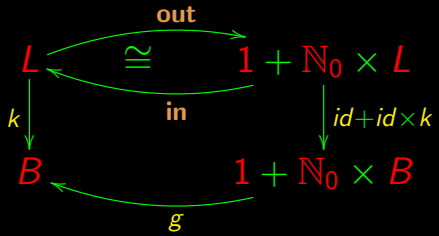
$$\begin{cases} \mathbf{in} = [nil, cons] \\ \mathbf{out} = \mathbf{in}^{\circ} \end{cases}$$

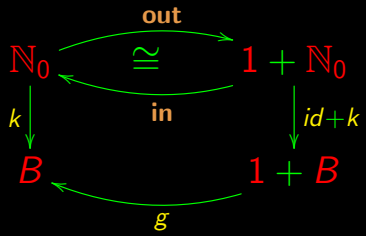
$$sum \cdot \mathbf{in} = [\underline{0}, add] \cdot (id + id \times sum)$$

Generalizing

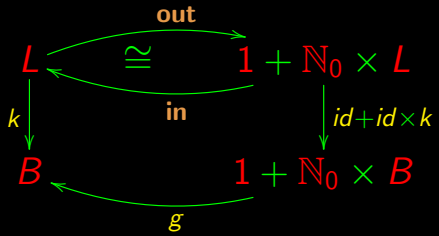


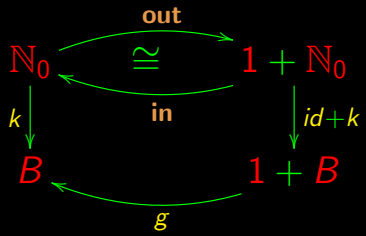
Generalizing





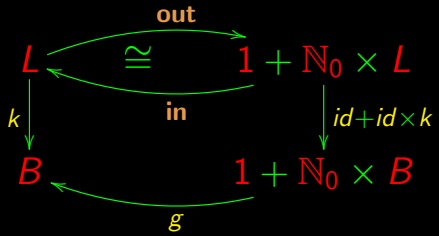
Generalizing

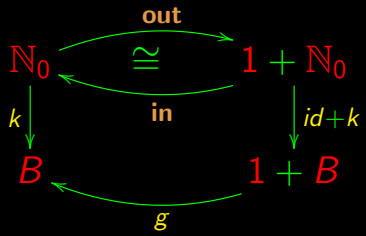




$$k \cdot \mathbf{in} = g \cdot \underbrace{id + id \times k}_{\mathbf{F}\ k}$$

Generalizing

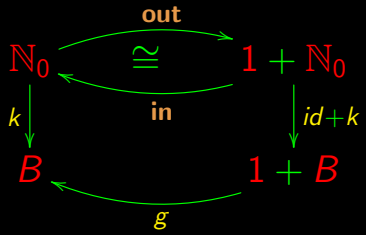




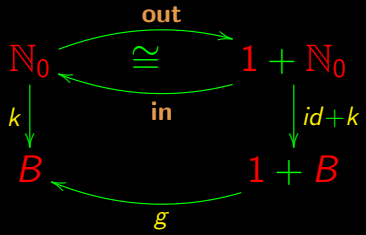
$$k \cdot \mathbf{in} = g \cdot \underbrace{id + id \times k}_{\mathbf{F}\ k}$$

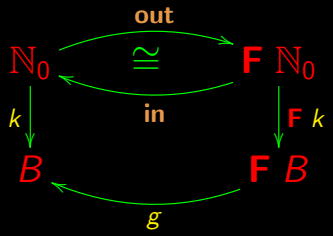
$$k \cdot \mathbf{in} = g \cdot \underbrace{id + k}_{\mathbf{F}\ k}$$

Abstraction ($\mathbb{N}_0$)



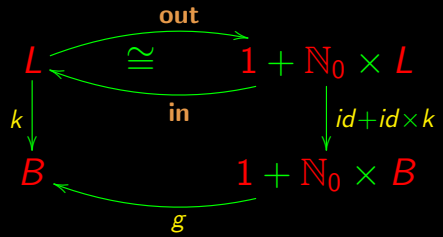
Abstraction ($\mathbb{N}_0$)



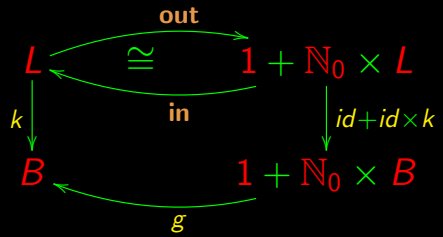


$$\begin{cases} \mathbf{F}\ k = id + k \\ \mathbf{F}\ X = 1 + X \end{cases}$$

Abstraction (lists)

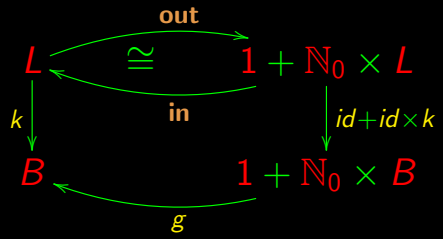


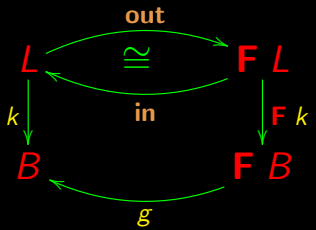
Abstraction (lists)



$$\begin{cases} \mathbf{F}\ k = id + id \times k \\ \mathbf{F}\ X = 1 + \mathbb{N}_0 \times X \end{cases}$$

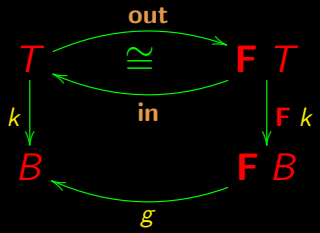
Abstraction (lists)



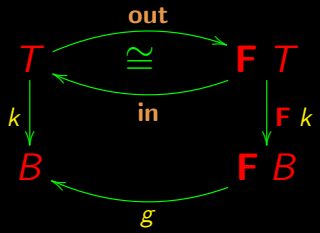


$$\begin{cases} \mathbf{F}\ k = id + id \times k \\ \mathbf{F}\ X = 1 + \mathbb{N}_0 \times X \end{cases}$$

Abstraction

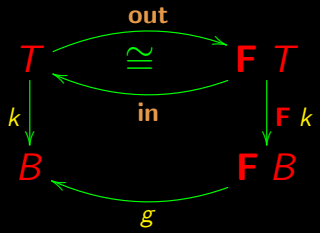


Abstraction



$$k \cdot \mathsf{in} = g \cdot \mathsf{F}\ k$$

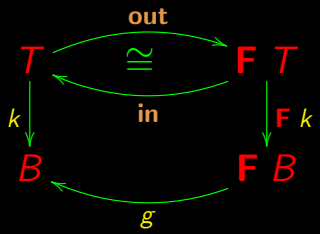
Abstraction



$$k = (\!|\, g \,|\!)$$

$$k \cdot \mathsf{in} = g \cdot \mathsf{F}\ k$$

Abstraction

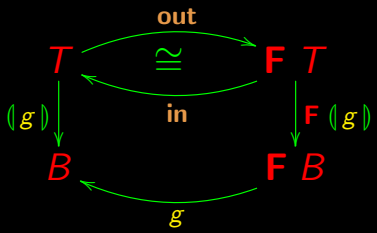


$$k = (\!|\, g \,|\!)$$

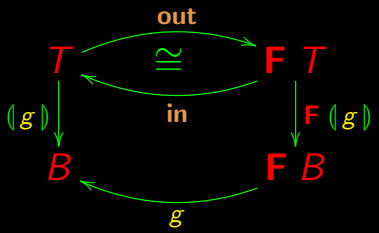
$$k \cdot \mathsf{in} = g \cdot \mathsf{F}\ k$$

Read $k = (\!|\, g \,|\!)$ as before: k is the *catamorphism* of g.

Catamorphism



Catamorphism



Universal property:

$$k = (\!|\, g \,|\!) \quad \Leftrightarrow \quad k \cdot \mathbf{in} = g \cdot \mathbf{F}\ k$$

Summary

Natural numbers: $\begin{cases} T = \mathbb{N}_0 \\ \mathbf{in} = [\underline{0}, succ] \\ \begin{cases} \mathbf{F}\ X = 1 + X \\ \mathbf{F}\ f = id + f \end{cases} \end{cases}$

Summary

Natural numbers: $\left\{ \begin{array}{l} T = \mathbb{N}_0 \\ \mathbf{in} = [\underline{0}, \mathit{succ}] \\ \left\{ \begin{array}{l} \mathbf{F}\ X = 1 + X \\ \mathbf{F}\ f = \mathit{id} + f \end{array} \right. \end{array} \right.$

$\mathbf{for}\ b\ i = (\!|\, [\underline{i}, b] \,|\!)$

Summary

Natural numbers: $\left\{ \begin{array}{l} T = \mathbb{N}_0 \\ \mathbf{in} = [\underline{0}, succ] \\ \left\{ \begin{array}{l} \mathbf{F}\ X = 1 + X \\ \mathbf{F}\ f = id + f \end{array} \right. \end{array} \right.$

for $b\ i = (\!|\, [\underline{i}, b] \,|\!)$

Lists: $\left\{ \begin{array}{l} T = L \\ \mathbf{in} = [nil, cons] \\ \left\{ \begin{array}{l} \mathbf{F}\ X = 1 + \mathbb{N}_0 \times X \\ \mathbf{F}\ f = id + id \times f \end{array} \right. \end{array} \right.$

Summary

Natural numbers:
$$\begin{cases} T = \mathbb{N}_0 \\ \mathbf{in} = [\underline{0}, succ] \\ \begin{cases} \mathbf{F}\ X = 1 + X \\ \mathbf{F}\ f = id + f \end{cases} \end{cases}$$

$$\mathbf{for}\ b\ i = (\!|\, [\underline{i}, b] \,|\!)$$

Lists:
$$\begin{cases} T = L \\ \mathbf{in} = [nil, cons] \\ \begin{cases} \mathbf{F}\ X = 1 + \mathbb{N}_0 \times X \\ \mathbf{F}\ f = id + id \times f \end{cases} \end{cases}$$

$$\mathbf{foldr}\ f\ i = (\!|\, [\underline{i}, \widehat{f}] \,|\!)$$

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

$$k = (\!|\, g \,|\!) \;\Leftrightarrow\; k \cdot \mathbf{in} = g \cdot \mathbf{F}\ k$$

make $k := (\!|\, g \,|\!)$:

Cata-cancellation

In

$$k = (\!|g|\!) \;\Leftrightarrow\; k \cdot \mathbf{in} = g \cdot \mathbf{F}\ k$$

make $k := (\!|g|\!)$:

$$(\!|g|\!) = (\!|g|\!) \;\Leftrightarrow\; (\!|g|\!) \cdot \mathbf{in} = g \cdot \mathbf{F}\ (\!|g|\!)$$

Cata-cancellation

In

$$k = (\!|g|\!) \;\Leftrightarrow\; k \cdot \mathbf{in} = g \cdot \mathbf{F}\ k$$

make $k := (\!|g|\!)$:

$$(\!|g|\!) = (\!|g|\!) \;\Leftrightarrow\; (\!|g|\!) \cdot \mathbf{in} = g \cdot \mathbf{F}\ (\!|g|\!)$$

$$\Leftrightarrow \qquad \{\ x = x \Leftrightarrow \mathit{true}\ \}$$

$$\mathit{true} \;\Leftrightarrow\; (\!|g|\!) \cdot \mathbf{in} = g \cdot \mathbf{F}\ (\!|g|\!)$$

Cata-cancellation

In

$$k = (\!|\, g \,|\!) \;\Leftrightarrow\; k \cdot \mathbf{in} = g \cdot \mathbf{F}\ k$$

make $k := (\!|\, g \,|\!)$:

$$(\!|\, g \,|\!) = (\!|\, g \,|\!) \;\Leftrightarrow\; (\!|\, g \,|\!) \cdot \mathbf{in} = g \cdot \mathbf{F}\ (\!|\, g \,|\!)$$

$$\Leftrightarrow \qquad \{\ x = x \Leftrightarrow \mathit{true}\ \}$$

$$\mathit{true} \;\Leftrightarrow\; (\!|\, g \,|\!) \cdot \mathbf{in} = g \cdot \mathbf{F}\ (\!|\, g \,|\!)$$

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

$$(\!|\, g \,|\!) \cdot \mathbf{in} = g \cdot \mathbf{F}\ (\!|\, g \,|\!)$$

Cata-cancellation

$$(\!|\, g \,|\!) \cdot \mathbf{in} = g \cdot \mathbf{F}\ (\!|\, g \,|\!)$$

$$\Leftrightarrow \qquad \{\ \text{isomorphism } \mathbf{in} \,/\, \mathbf{out}\ \}$$

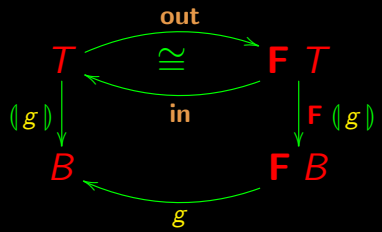
$$(\!|\, g \,|\!) = g \cdot \mathbf{F}\ (\!|\, g \,|\!) \cdot \mathbf{out}$$

Cata-cancellation

$$(\!|\, g \,|\!) \cdot \mathbf{in} = g \cdot \mathbf{F}\ (\!|\, g \,|\!)$$

$$\Leftrightarrow \quad \{\ \text{isomorphism } \mathbf{in} \,/\, \mathbf{out}\ \}$$

$$(\!|\, g \,|\!) = g \cdot \mathbf{F}\ (\!|\, g \,|\!) \cdot \mathbf{out}$$

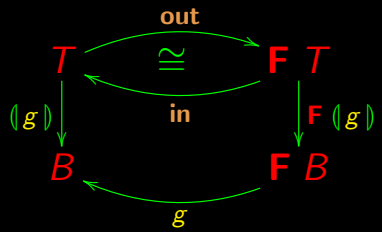


Cata-cancellation

$$(\!|\, g \,|\!) \cdot \mathbf{in} = g \cdot \mathbf{F}\ (\!|\, g \,|\!)$$

$$\Leftrightarrow \quad \{\ \text{isomorphism } \mathbf{in} \,/\, \mathbf{out}\ \}$$

$$(\!|\, g \,|\!) = g \cdot \mathbf{F}\ (\!|\, g \,|\!) \cdot \mathbf{out}$$

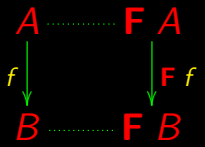


$(\!|\, g \,|\!) = g \cdot \mathbf{F}\ (\!|\, g \,|\!) \cdot \mathbf{out}$

executable definition

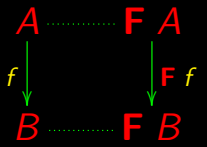
Functors

What does **F** mean, exactly?



Functors

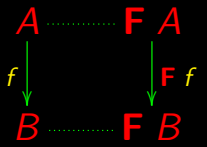
What does **F** mean, exactly?



Recall **natural** (free)
properties

Functors

What does **F** mean, exactly?



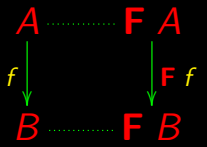
For instance:

$$\mathbf{F}\, X = 1 + \mathbb{N}_0 \times X$$

Recall **natural** (free) **properties**

Functors

What does **F** mean, exactly?



Recall **natural** (free) **properties**

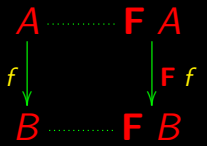
For instance:

$$\mathbf{F}\, X = 1 + \mathbb{N}_0 \times X$$

Substitute uppercase (X) by lowercase f ($X \to f$):

Functors

What does **F** mean, exactly?



Recall **natural** (free) **properties**

For instance:

$$\mathbf{F}\ X = 1 + \mathbb{N}_0 \times X$$

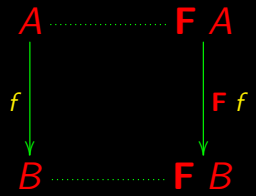
Substitute uppercase (X) by lowercase f ($X \to f$):

$$\mathbf{F}\ f = id + id \times f$$

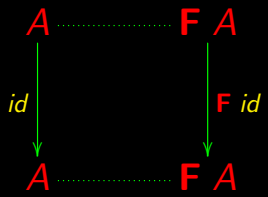
Properties

$$\begin{aligned} \mathbf{F}\ id &= id \\ \mathbf{F}\ (g \cdot f) &= (\mathbf{F}\ g) \cdot (\mathbf{F}\ f) \end{aligned}$$

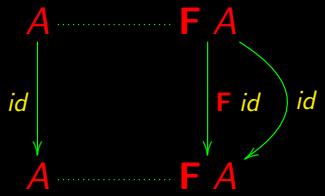
$$\mathbf{F}\ id = id$$



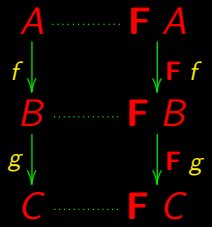
$$\mathbf{F}\ id = id$$



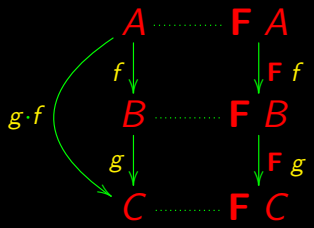
$$\mathbf{F}\ id = id$$



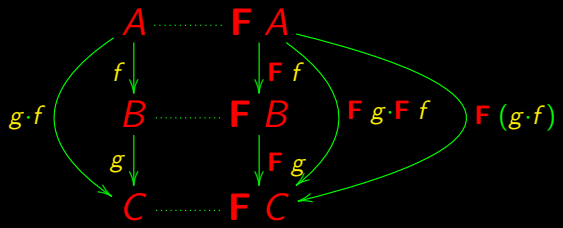
$$\mathbf{F}\ (g \cdot f) = (\mathbf{F}\ g) \cdot (\mathbf{F}\ f)$$



$$\mathbf{F}\ (g \cdot f) = (\mathbf{F}\ g) \cdot (\mathbf{F}\ f)$$



$$\mathbf{F}\ (g \cdot f) = (\mathbf{F}\ g) \cdot (\mathbf{F}\ f)$$



Well known example

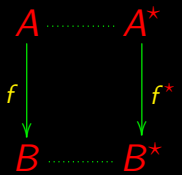
List functor:

$$\mathbf{F}\ f = f^{\star}$$

Well known example

List functor:

$$\mathbf{F}\ f = f^\star \quad \textbf{where} \quad f^\star\ x = [f\ a \mid a \leftarrow x] = \mathsf{map}\ f\ x$$



In fact

$$id^{\star}\ x = [a \mid a \leftarrow x] = x$$

In fact

$$id^\star\ x = [a \mid a \leftarrow x] = x$$

$$(g^\star \cdot f^\star)\ x$$

In fact

$$id^\star\ x = [a \mid a \leftarrow x] = x$$

$$(g^\star \cdot f^\star)\ x = [g\ b \mid b \leftarrow [f\ a \mid a \leftarrow x]]$$

In fact

$$id^\star\ x = [a \mid a \leftarrow x] = x$$

$$(g^\star \cdot f^\star)\ x = [g\ b \mid b \leftarrow [f\ a \mid a \leftarrow x]] = (g \cdot f)^\star\ x$$

Properties

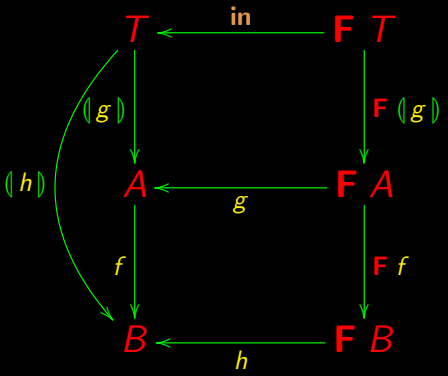
Recall

$$\begin{aligned} \mathbf{F}\ id &= id \\ \mathbf{F}\ (g \cdot f) &= (\mathbf{F}\ g) \cdot (\mathbf{F}\ f) \end{aligned}$$

These properties are essential for reasoning about **catamorphisms**.

See example next.

Cata-fusion



Cata-fusion

$$f \cdot (\!|\, g \,|\!) = (\!|\, h \,|\!)$$

Cata-fusion

$$
\begin{aligned}
& f \cdot (\!|\, g \,|\!) = (\!|\, h \,|\!) \\
\Leftrightarrow \quad & \{ \text{ Cata-universal for } k = f \cdot (\!|\, g \,|\!) \} \\
& f \cdot (\!|\, g \,|\!) \cdot \mathsf{in} = h \cdot \mathsf{F}\ (f \cdot (\!|\, g \,|\!))
\end{aligned}
$$

Cata-fusion

$$
\begin{aligned}
& f \cdot (\!|\, g \,|\!) = (\!|\, h \,|\!) \\
\Leftrightarrow \quad & \{ \text{ Cata-universal for } k = f \cdot (\!|\, g \,|\!) \} \\
& f \cdot (\!|\, g \,|\!) \cdot \mathsf{in} = h \cdot \mathsf{F}\ (f \cdot (\!|\, g \,|\!)) \\
\Leftrightarrow \quad & \{ \text{ Cata-cancellation } \} \\
& f \cdot g \cdot \mathsf{F}\ (\!|\, g \,|\!) = h \cdot \mathsf{F}\ (f \cdot (\!|\, g \,|\!))
\end{aligned}
$$

Cata-fusion

$$f \cdot g \cdot \mathbf{F}\ (\!|\, g \,|\!) = h \cdot \mathbf{F}\ (f \cdot (\!|\, g \,|\!))$$

Cata-fusion

$$
\begin{aligned}
& f \cdot g \cdot \mathsf{F}\ (\!|\, g \,|\!) = h \cdot \mathsf{F}\ (f \cdot (\!|\, g \,|\!)) \\
\Leftrightarrow \quad & \{\ \text{functor-}\mathsf{F}\ (1)\ \} \\
& f \cdot g \cdot \mathsf{F}\ (\!|\, g \,|\!) = h \cdot \mathsf{F}\ f \cdot \mathsf{F}\ (\!|\, g \,|\!)
\end{aligned}
$$

Cata-fusion

$$
\begin{aligned}
& f \cdot g \cdot \mathsf{F}\ (\!|\, g \,|\!) = h \cdot \mathsf{F}\ (f \cdot (\!|\, g \,|\!)) \\
\Leftrightarrow \quad & \{\ \text{functor-}\mathsf{F}\ (1)\ \} \\
& f \cdot g \cdot \mathsf{F}\ (\!|\, g \,|\!) = h \cdot \mathsf{F}\ f \cdot \mathsf{F}\ (\!|\, g \,|\!) \\
\Leftarrow \quad & \{\ \text{Leibniz}\ \} \\
& f \cdot g = h \cdot (\mathsf{F}\ f)
\end{aligned}
$$

Cata-fusion

$$
\begin{aligned}
& f \cdot g \cdot \mathsf{F}\ (\!|\, g \,|\!) = h \cdot \mathsf{F}\ (f \cdot (\!|\, g \,|\!)) \\
\Leftrightarrow \quad & \{\ \text{functor-}\mathsf{F}\ (1)\ \} \\
& f \cdot g \cdot \mathsf{F}\ (\!|\, g \,|\!) = h \cdot \mathsf{F}\ f \cdot \mathsf{F}\ (\!|\, g \,|\!) \\
\Leftarrow \quad & \{\ \text{Leibniz}\ \} \\
& f \cdot g = h \cdot (\mathsf{F}\ f)
\end{aligned}
$$

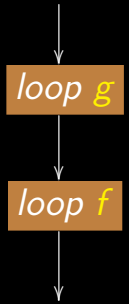
Summing up:

$$f \cdot (\!|\, g \,|\!) = (\!|\, h \,|\!) \ \Leftarrow \ f \cdot g = h \cdot (\mathsf{F}\ f)$$

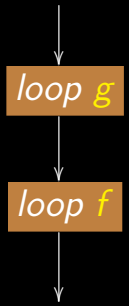
Cálculo de Programas

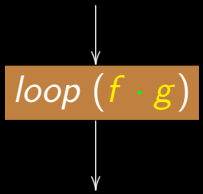
Class T08

Fusion in programming — "sequential"

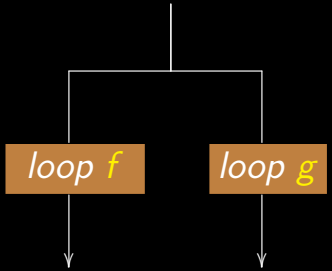


Fusion in programming — "sequential"

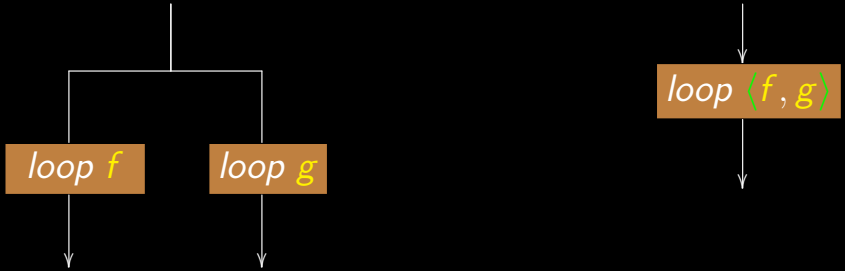




Fusion in programming — "parallel"

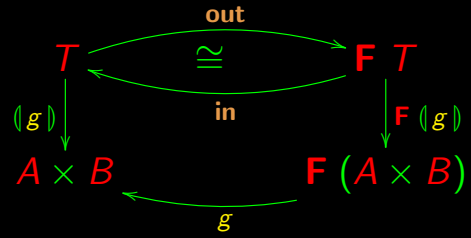


Fusion in programming — "parallel"

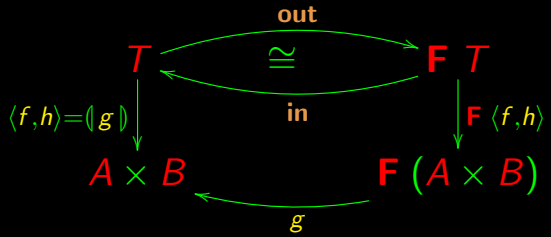


The pattern $\langle f, g \rangle = (\!| h |\!)$

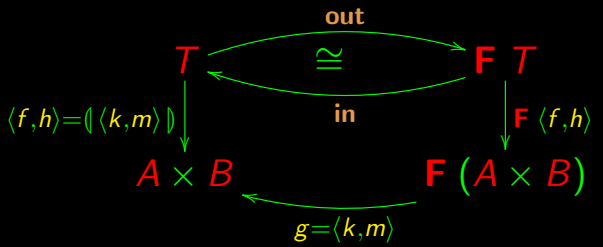
Catas that yield pairs:



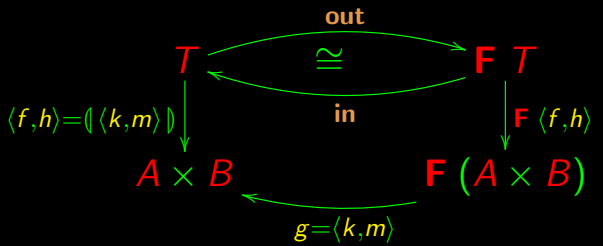
Catas that yield pairs



Mutual recursion



Mutual recursion



$$\langle f, h\rangle = (\!|\, \langle k, m\rangle \,|\!) \Leftrightarrow \begin{cases} f \cdot \mathbf{in} = k \cdot \mathbf{F}\ \langle f, h\rangle \\ h \cdot \mathbf{in} = m \cdot \mathbf{F}\ \langle f, h\rangle \end{cases}$$

Mutual recursion

$$
\begin{aligned}
& \langle f, h \rangle = (\!| \langle k, m \rangle |\!) \\
\Leftrightarrow & \quad \{ \text{ Cata-universal } \} \\
& \langle f, h \rangle \cdot \mathbf{in} = \langle k, m \rangle \cdot \mathbf{F}\ \langle f, h \rangle \\
\Leftrightarrow & \quad \{ \text{ } \times\text{-fusion } \} \\
& \langle f \cdot \mathbf{in}, h \cdot \mathbf{in} \rangle = \langle k \cdot \mathbf{F}\ \langle f, h \rangle, m \cdot \mathbf{F}\ \langle f, h \rangle \rangle \\
\Leftrightarrow & \quad \{ \text{ Eq-}\times \text{ } \} \\
& \begin{cases} f \cdot \mathbf{in} = k \cdot \mathbf{F}\ \langle f, h \rangle \\ h \cdot \mathbf{in} = m \cdot \mathbf{F}\ \langle f, h \rangle \end{cases}
\end{aligned}
$$

Mutual recursion — case $T = \mathbb{N}_0$

$$\langle f, g \rangle = (\!| \langle h, k \rangle |\!)$$

$$\Leftrightarrow \quad \{ \text{ mutual recursion with } \mathbf{in} = [\underline{0}, succ] \text{ etc } \}$$

$$\begin{cases} f \cdot [\underline{0}, succ] = h \cdot (id + \langle f, g \rangle) \\ g \cdot [\underline{0}, succ] = k \cdot (id + \langle f, g \rangle) \end{cases}$$

$$\Leftrightarrow \quad \{ \text{ let } h = [h_1, h_2] \text{ and } k = [k_1, k_2] \}$$

$$\begin{cases} f \cdot [\underline{0}, succ] = [h_1, h_2] \cdot (id + \langle f, g \rangle) \\ g \cdot [\underline{0}, succ] = [k_1, k_2] \cdot (id + \langle f, g \rangle) \end{cases}$$

Mutual recursion — case $T = \mathbb{N}_0$

Moving on:

$$\langle f, g \rangle = (\!|\, \langle [h_1, h_2], [k_1, k_2] \rangle \,|\!)$$

$$\Leftrightarrow \quad \{ \text{ coproduct laws } \}$$

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} f \cdot \underline{0} = h_1 \\ f \cdot succ = h_2 \cdot \langle f, g \rangle \end{array} \right. \\ \left\{ \begin{array}{l} g \cdot \underline{0} = k_1 \\ g \cdot succ = k_2 \cdot \langle f, g \rangle \end{array} \right. \end{array} \right.$$

Mutual recursion — case $T = \mathbb{N}_0$

Finally, add variables to righthand side, with $h_1 = \underline{a}$ and $k_1 = \underline{b}$:

$$\langle f, g \rangle = (\!| \langle [\underline{a}, h_2], [\underline{b}, k_2] \rangle |\!) \quad \Leftrightarrow \quad \left\{ \begin{array}{l} \left\{ \begin{array}{l} f\ 0 = a \\ f\ (n+1) = h_2\ (f\ n, g\ n) \end{array} \right. \\ \left\{ \begin{array}{l} g\ 0 = b \\ g\ (n+1) = k_2\ (f\ n, g\ n) \end{array} \right. \end{array} \right.$$

Example — Fibonacci

Clearly, the following function is not a catamorphism of $\mathbb{N}_0$:

$$\begin{aligned} &\mathit{fib}\ 0 = 1 \\ &\mathit{fib}\ 1 = 1 \\ &\mathit{fib}\ (n+2) = \mathit{fib}\ (n+1) + \mathit{fib}\ n \end{aligned}$$

But it can be transformed into one using mutual recursion.

Example — Fibonacci

Let us define the auxiliary function:

$$f\ n = fib\ (n + 1)$$

Then:

$$f\ 0 = fib\ 1 = 1$$
$$f\ (n + 1) = fib\ (n + 2) = f\ n + fib\ n$$

Example — Fibonacci

Concerning *fib*:

$$\begin{aligned} &\mathit{fib}\ 0 = 1 \\ &\mathit{fib}\ (n+1) = f\ n \end{aligned}$$

Summing up, we have the following mutual recursion:

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} f\ 0 = 1 \\ f\ (n+1) = f\ n + \mathit{fib}\ n \end{array} \right. \\ \left\{ \begin{array}{l} \mathit{fib}\ 0 = 1 \\ \mathit{fib}\ (n+1) = f\ n \end{array} \right. \end{array} \right.$$

Example — Fibonacci

By applying the mutual recursion law for $T = \mathbb{N}_0$:

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} f\ 0 = a \\ f\ (n+1) = h_2\ (f\ n, fib\ n) \end{array} \right. \\ \left\{ \begin{array}{l} fib\ 0 = b \\ fib\ (n+1) = k_2\ (f\ n, fib\ n) \end{array} \right. \end{array} \right. \quad \Leftrightarrow \quad \langle f, fib \rangle = (\!| \langle [\underline{a}, h_2], [\underline{b}, k_2] \rangle |\!)$$

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:

$$\textbf{for}\ b\ i = (\!|\ [\underline{i}, b]\ |\!)$$

Fibonacci becomes a for-loop

Given $\langle f, fib \rangle$ (a $\mathbb{N}_0$ **catamorphism**), we can convert it into a **for-loop** according to the definition:

$$\textbf{for}\ b\ i = (\!|\, [\underline{i}, b] \,|\!)$$

$$\begin{array}{lll}
 & \langle f, fib \rangle & = (\!|\, \langle [\underline{1}, add], [\underline{1}, \pi_1] \rangle \,|\!) \\
\Leftrightarrow & \quad \{ \text{ exchange law } \} & \\
 & \langle f, fib \rangle & = (\!|\, [\langle \underline{1}, \underline{1} \rangle, \langle add, \pi_1 \rangle] \,|\!) \\
\Leftrightarrow & \quad \{ \text{ recall exercise sheet 3 } \} & \\
 & \langle f, fib \rangle & = (\!|\, [\underline{(1,1)}, \langle add, \pi_1 \rangle] \,|\!) \\
\Leftrightarrow & \quad \{ \text{ definition of } \textbf{for}\ b\ i \ \} & \\
 & \langle f, fib \rangle & = \textbf{for}\ \langle add, \pi_1 \rangle\ (1,1)
\end{array}$$

Example — Fibonacci

In summary:

$$fib = \pi_2 \cdot (\textbf{for } \langle add, \pi_1 \rangle \ (1,1))$$

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

$$
\begin{aligned}
&fac\ 0 = 1 \\
&fac\ (n+1) = \underbrace{n+1}_{f\ n} \times fac\ n
\end{aligned}
$$

Then:

$$
\langle f, fac \rangle = (\!| \langle [\underline{a}, h_2], [\underline{b}, k_2] \rangle |\!) \quad \Leftrightarrow \quad \left\{ \begin{array}{l} \left\{ \begin{array}{l} f\ 0 = a \\ f\ (n+1) = h_2\ (f\ n, fac\ n) \end{array} \right. \\ \left\{ \begin{array}{l} fac\ 0 = b \\ fac\ (n+1) = k_2\ (f\ n, fac\ n) \end{array} \right. \end{array} \right.
$$

Example — factorial

$$\langle f, fac \rangle = (\!| \langle [\underline{a}, h_2], [\underline{b}, k_2] \rangle |\!) \quad \Leftrightarrow \quad \begin{cases} \begin{cases} f\ 0 = a \\ f\ (n+1) = h_2\ (f\ n, fac\ n) \end{cases} \\ \begin{cases} fac\ 0 = b \\ fac\ (n+1) = k_2\ (f\ n, fac\ n) \end{cases} \end{cases}$$

$$k_2 = mul$$
$$b = 1$$
$$f\ n = n + 1$$
$$a = 1$$

(Still need h_2...)

Example — factorial

$$
\begin{aligned}
& f\ (n+1) \\
= \ & (n+1)+1 \\
= \ & 1+(n+1) \\
= \ & 1+f\ n \\
= \ & 1+\pi_1\ (f\ n, fac\ n) \\
= \ & \underbrace{succ \cdot \pi_1}_{h_2}\ (f\ n, fac\ n)
\end{aligned}
$$

Then:

$$
\begin{aligned}
k_2 &= mul \\
b &= 1 \\
h_2 &= succ \cdot \pi_1 \\
a &= 1
\end{aligned}
$$

Example — factorial becomes a for-loop

Finally:

$$
\begin{aligned}
& \langle f, fac \rangle \\
= \; & (\!| \langle [\underline{1}, succ \cdot \pi_1], [\underline{1}, mul] \rangle |\!) \\
= \; & (\!| [\underline{(1,1)}, \langle succ \cdot \pi_1, mul \rangle] |\!) \\
= \; & \textbf{for } \langle succ \cdot \pi_1, mul \rangle \; ((1,1)) \\
= \; & \textbf{for } g \; (1,1) \textbf{ where } g \; (x,y) = (x+1, x \times y)
\end{aligned}
$$

For-loops (in C)

In general, $k = \textbf{for}\ f\ i$ 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

$$fac = \pi_2\ (\textbf{for}\ g\ (1,1)\ \textbf{where}\ g\ (x,y) = (x+1, x \times y))$$

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

$$(\!|\langle \_, \_ \rangle|\!)$$

Banana-split

$$(\!|\langle \_, \_ \rangle|\!)$$

$$\langle (\!|\_|\!), (\!|\_|\!) \rangle$$

Banana-split

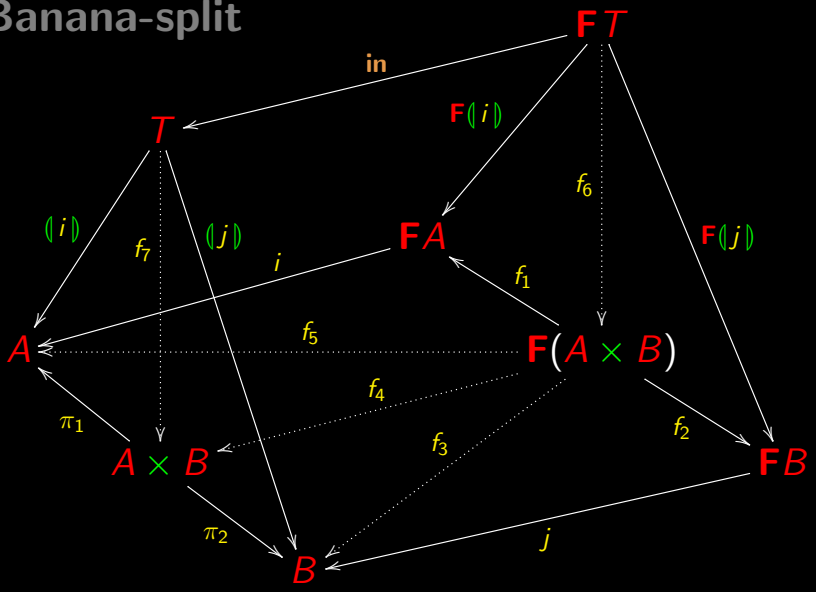
$$(\!|\langle \_, \_ \rangle|\!)$$

$$\langle (\!|\_|\!), (\!|\_|\!) \rangle$$

$$\langle (\!|i|\!), (\!|j|\!) \rangle$$

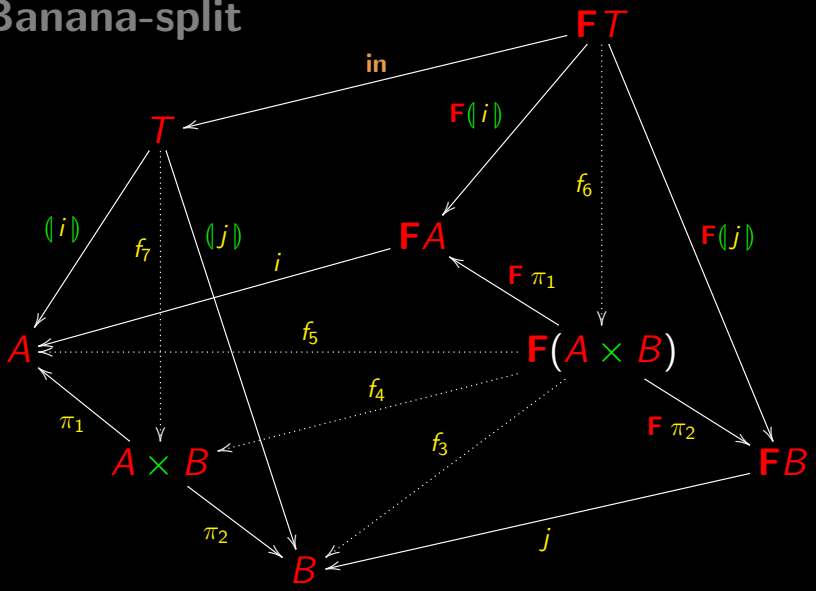
Banana-split

$\langle (\!| i |\!), (\!| j |\!) \rangle$



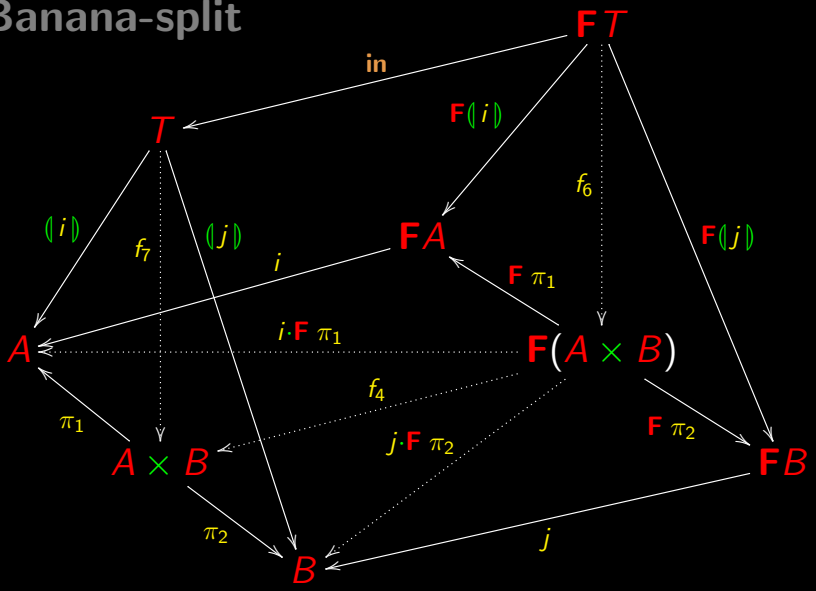
Banana-split

$\langle (\!| i |\!), (\!| j |\!) \rangle$



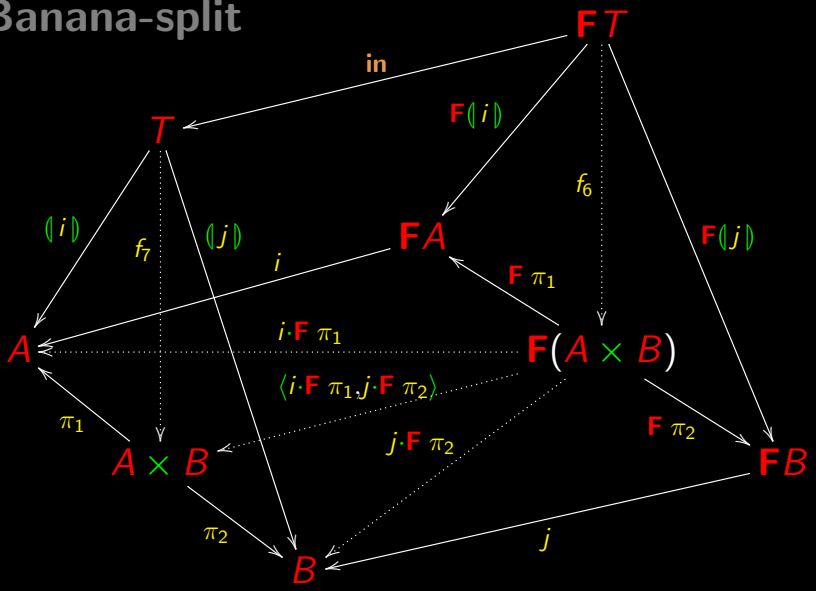
Banana-split

$\langle (\!| i |\!), (\!| j |\!) \rangle$



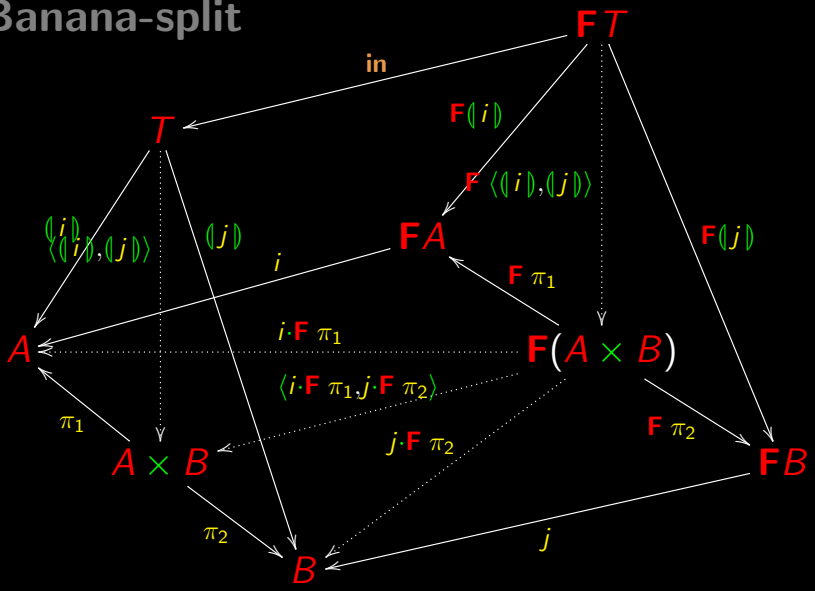
Banana-split

$\langle (\!| i |\!), (\!| j |\!) \rangle$



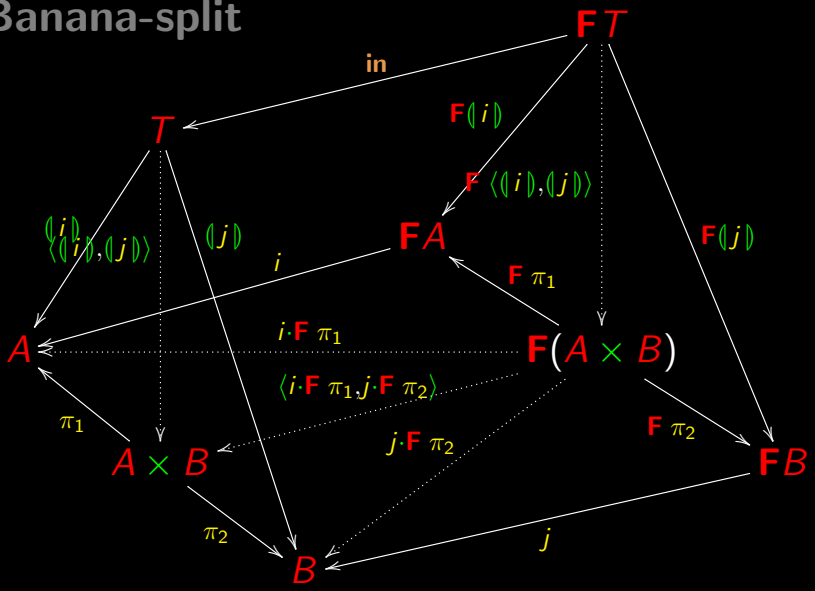
Banana-split

$\langle (\!| i |\!), (\!| j |\!) \rangle$

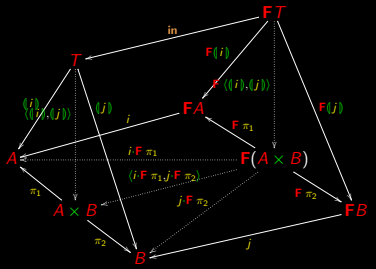


Banana-split

$\langle (\!| i |\!), (\!| j |\!) \rangle$



Banana-split



$$\langle (\!| i |\!), (\!| j |\!) \rangle = (\!| \langle i \cdot \mathbf{F}\ \pi_1, j \cdot \mathbf{F}\ \pi_2 \rangle |\!)$$

Example

Average of a (non empty) list:

$$average = (/) \cdot \langle sum, length \rangle$$

where

$$sum = (\!|\, [0, add] \,|\!)$$
$$length = (\!|\, [0, succ \cdot \pi_2] \,|\!)$$

Example

$$average = (/) \cdot \underbrace{\langle sum, length \rangle}_{aux}$$

By banana-split:

$$aux = (\!| \langle f_5, f_3 \rangle |\!) \textbf{ where}$$
$$f_5 = [0, add] \cdot (id + id \times \pi_1)$$
$$f_3 = [0, succ \cdot \pi_2] \cdot (id + id \times \pi_2)$$

Example

Finally, solve $aux = (\!|\langle f_5, f_3 \rangle|\!)$ and convert everything to pointwise notation:

$$
\begin{aligned}
&average\ l = x \,/\, y \ \textbf{where} \\
&\quad (x, y) = aux\ l \\
&\quad aux\ [\,] = (0, 0) \\
&\quad aux\ (a : l) = (a + x, y + 1) \ \textbf{where}\ (x, y) = aux\ l
\end{aligned}
$$

Example

Summing up,

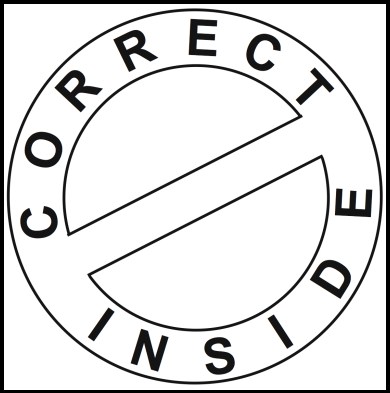
$$
\begin{aligned}
&\mathit{average}\ l = x \,/\, y \ \textbf{where} \\
&\quad (x, y) = \mathit{aux}\ l \\
&\quad \mathit{aux}\ [\,] = (0, 0) \\
&\quad \mathit{aux}\ (a : l) = (a + x, y + 1) \ \textbf{where}\ (x, y) = \mathit{aux}\ l
\end{aligned}
$$

twice as fast as

$$
\begin{aligned}
&\mathit{average}\ l = (\mathit{sum}\ l) \,/\, (\mathit{length}\ l) \ \textbf{where} \\
&\quad \mathit{sum}\ [\,] = 0 \\
&\quad \mathit{sum}\ (h : t) = h + \mathit{sum}\ t \\
&\quad \mathit{length}\ [\,] = 0 \\
&\quad \mathit{length}\ (h : t) = 1 + \mathit{length}\ t
\end{aligned}
$$

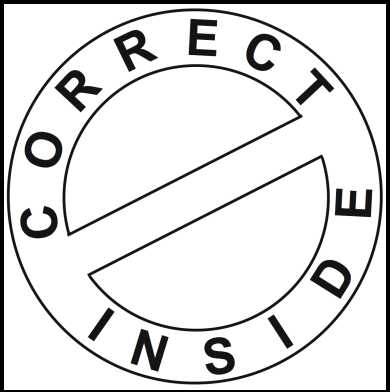
and **guaranteed** to be correct.

Program Calculation



Efficiency without sacrificing **correction**.

Program Calculation



Efficiency without sacrificing **correction**.

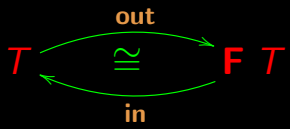
Who cares about an efficient program that gives wrong outputs?

Program Calculation

Efficiency should never sacrifice **correction**!

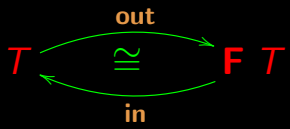
Recap

Inductive type T and the functor $\mathbf{F}$ that determines its pattern of recursion:



Recap

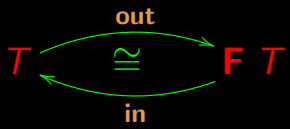
Inductive type T and the functor $\mathbf{F}$ that determines its pattern of recursion:



We have seen examples: $T = \mathbb{N}_0$, $T = A^\star$ etc.

Recap

Inductive type T and the functor $\mathbf{F}$ that determines its pattern of recursion:



We have seen examples: $T = \mathbb{N}_0$, $T = A^\star$ etc.

Let us see some more.

Binary trees

Trees with data of type A in their nodes, for example

$$\texttt{data}\ BTree = Empty \mid Node\ (A, (BTree, BTree))$$

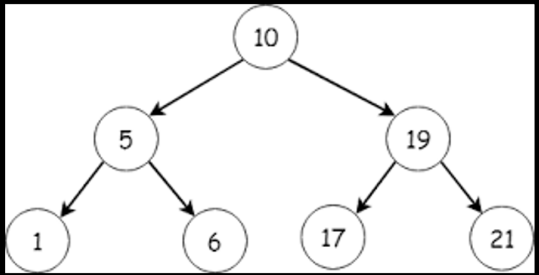
(A assumed pre-defined.)

Binary trees

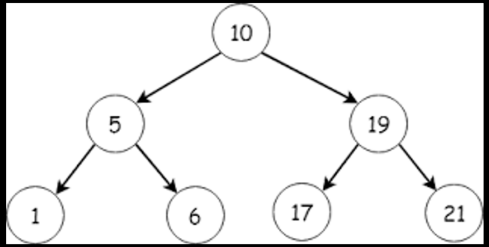
Trees with data of type A in their nodes, for example

$$\texttt{data}\ BTree = Empty \mid Node\ (A, (BTree, BTree))$$

(A assumed pre-defined.) For instance



Binary trees




```
Node (10,
   (Node (5,
      (Node (1,
         (Empty,Empty)),
       Node (6,
         (Empty,Empty)))),
    Node (19,
       (Node (17,
         (Empty,Empty)),
        Node (21,
         (Empty,Empty))))))
```

Binary trees

```
: data BTree = Empty | Node(A, (BTree, BTree))
```

```
: :t Node
```

```
Node :: (A, (BTree, BTree)) -> BTree
```

```
: :t Empty
```

```
Empty :: BTree
```

Binary trees

```
: data BTree = Empty | Node(A, (BTree, BTree))
```

```
: :t Node
```

```
Node :: (A, (BTree, BTree)) -> BTree
```

```
: :t Empty
```

```
Empty :: BTree
```

$$Node : A \times (BTree \times BTree) \to BTree$$

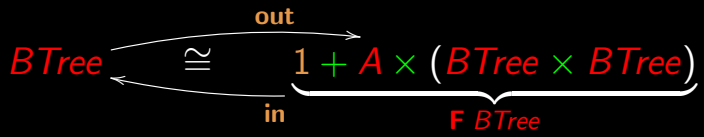
$$\underline{Empty} : 1 \to BTree$$

Binary trees

$$\mathbf{in} = [\underline{\mathit{Empty}}, \mathit{Node}]$$

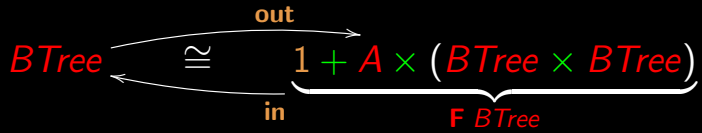
Binary trees

$$\mathbf{in} = [\underline{Empty}, Node]$$



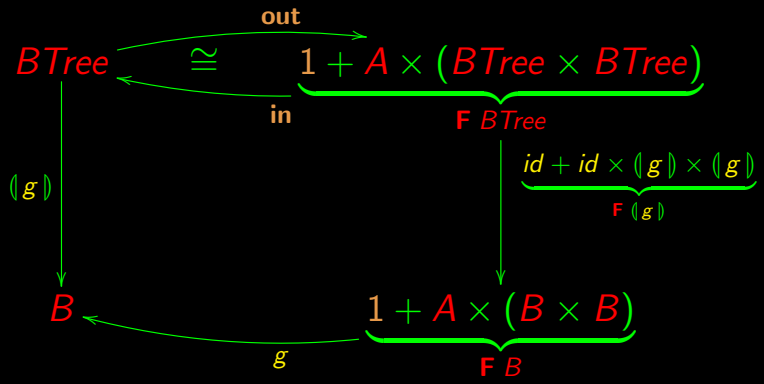
$$\texttt{data}\ BTree = Empty \mid Node\ (A, (BTree, BTree))$$

Binary trees



$$
\begin{aligned}
T &= BTree \\
&\begin{cases} \mathbf{F}\ X = 1 + A \times (X \times X) \\ \mathbf{F}\ f = id + id \times (f \times f) \end{cases} \\
\mathbf{in} &= [\underline{Empty}, Node]
\end{aligned}
$$

Catamorphisms of binary trees



Catamorphisms of binary trees

In Haskell:

```
cataBTree g = g . (recBTree (cataBTree g)) . outBTree

outBTree Empty               = Left ()
outBTree (Node (a,(t1,t2))) = Right(a,(t1,t2))

recBTree f = id -|- id >< (f >< f)

inBTree = either (const Empty) Node
```

Catamorphisms of binary trees

Example: function

$$f : BTree \to \mathbb{N}_0$$

that computes the size of the tree (total number of nodes).

Catamorphisms of binary trees

Example: function

$$f : BTree \to \mathbb{N}_0$$

that computes the size of the tree (total number of nodes).

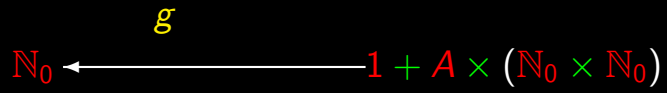
Enough caring about the "gene" g in

$$f = (\!|\, g \,|\!)$$

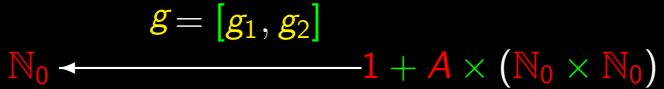
where

$$\mathbb{N}_0 \xleftarrow{g} 1 + A \times (\mathbb{N}_0 \times \mathbb{N}_0)$$

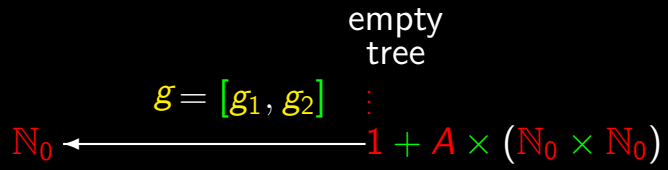
Catamorphisms of binary trees



Catamorphisms of binary trees

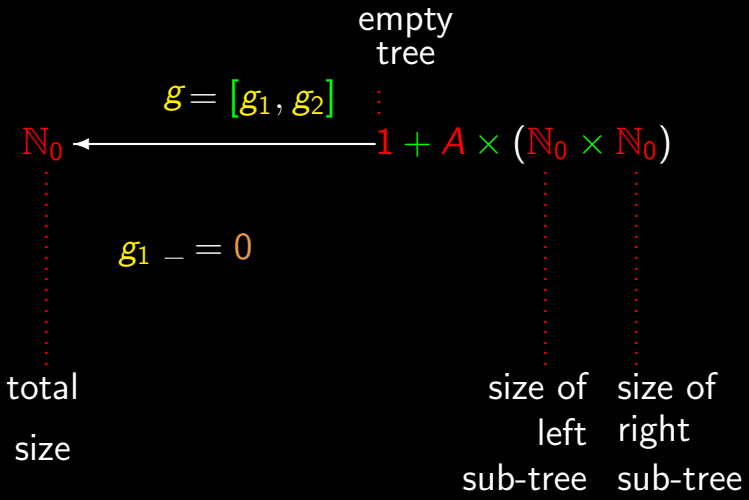


Catamorphisms of binary trees

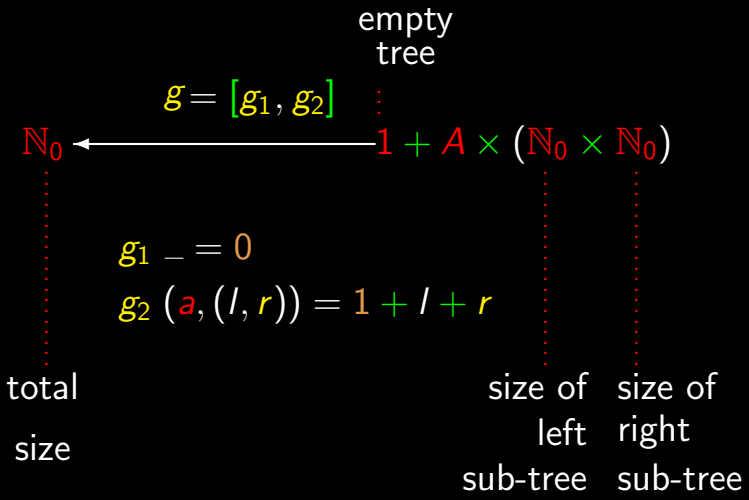


$g_1\ \_ = 0$

Catamorphisms of binary trees



Catamorphisms of binary trees



Catamorphisms of binary trees

Summing up,

$$f = (\!|\, [g_1, g_2] \,|\!) \textbf{ where}$$
$$g_1\ \_ = 0$$
$$g_2\ (a, (l, r)) = 1 + l + r$$

Catamorphisms of binary trees

Summing up,

$$f = (\!|\, [g_1, g_2] \,|\!) \textbf{ where}$$
$$g_1\ \_ = 0$$
$$g_2\ (a, (l, r)) = 1 + l + r$$

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

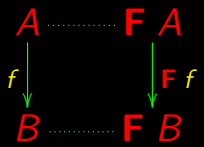
Let us now show how recursive types become **parametric**.

That is, they become **functors**.

It all amounts to generalize *map* to other recursive types.

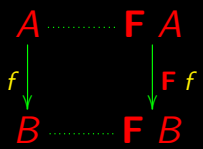
Recall...

Functor **F**:



Recall...

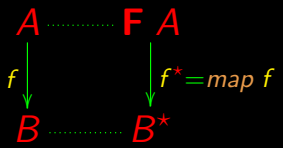
Functor $\mathbf{F}$:



$$\begin{cases} \mathbf{F}\ id = id \\ \mathbf{F}\ (g \cdot f) = (\mathbf{F}\ g) \cdot (\mathbf{F}\ f) \end{cases}$$

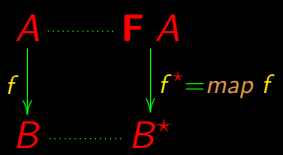
map f

Functor $\mathbf{F}\ A = A^\star$:



map f

Functor $\mathbf{F}\ A = A^\star$:



$$\begin{cases} map\ id = id \\ map\ (g \cdot f) = (map\ g) \cdot (map\ f) \end{cases}$$

How do we **prove** this? **Generalize** this?

$map\ f$

From *specification*

$$map\ f\ [\,] = [\,]$$
$$map\ f\ [a] = [f\ a]$$
$$map\ f\ (x +\!\!+ y) = map\ f\ x +\!\!+ map\ f\ y$$

easy to obtain (as before):

$$map\ f\ [\,] = [\,]$$
$$map\ f\ (h : t) = f\ h : map\ f\ t$$

map f

Then *implementation*

$$map\ f\ [\,] = [\,]$$
$$map\ f\ (h : t) = f\ h : map\ f\ t$$

map f

Then *implementation*

$$map\ f\ [\,] = [\,]$$
$$map\ f\ (h : t) = f\ h : map\ f\ t$$

is

$$map\ f = (\!|\, [nil, g_2] \,|\!) \textbf{ where}$$
$$g_2\ (h, x) = f\ h : x$$

$map\ f$

Then *implementation*

$$map\ f\ [\,] = [\,]$$
$$map\ f\ (h : t) = f\ h : map\ f\ t$$

is

$$map\ f = (\!|\, [nil, g_2] \,|\!) \textbf{ where}$$
$$g_2\ (h, x) = f\ h : x$$

the same thing as $map\ f = (\!|\, [nil, cons] \cdot (id + f \times id) \,|\!)$

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 $sq\ x = x^2$

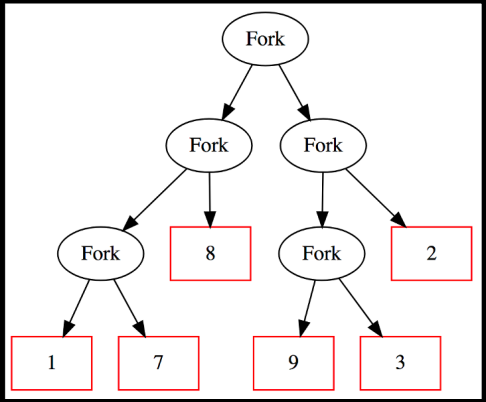
$$map\ sq\ [1, 7, 8, 9, 3, 2] = [1, 49, 64, 81, 9, 4]$$

But still we haven't proved

$$map\ id = id$$
$$map\ (f \cdot g) = (map\ f) \cdot (map\ g)$$

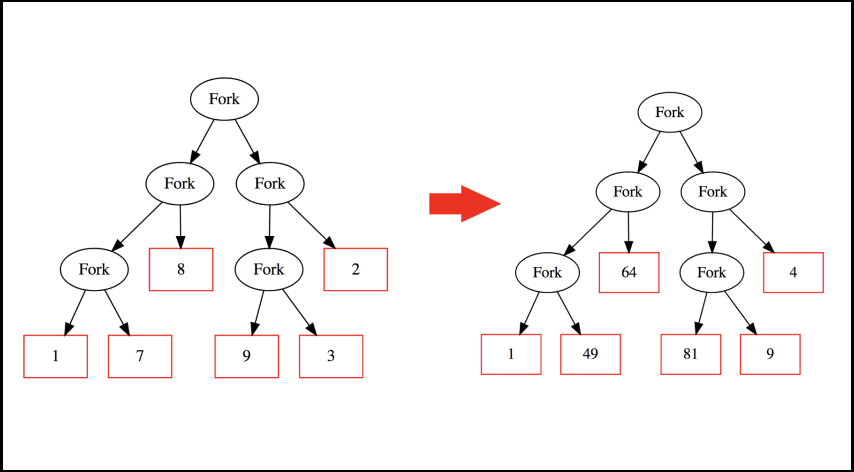
(Coming up soon.)

Inductive parametric types



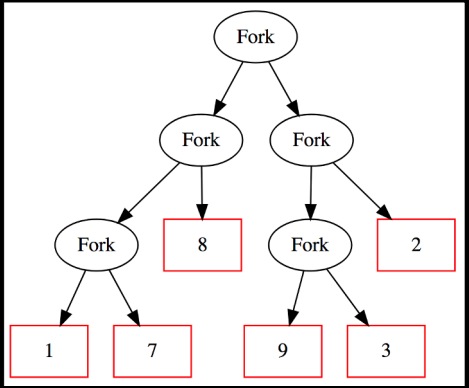
$$\texttt{data}\ \mathit{LTree} = \mathit{Leaf}\ \mathit{Int} \mid \mathit{Fork}\ (\mathit{LTree}, \mathit{LTree})$$

Inductive parametric types



Inductive parametric types

$$
\begin{aligned}
t = \; & \mathit{Fork} \\
& (\mathit{Fork} \\
& \quad (\mathit{Fork}\ (\mathit{Leaf}\ 1, \mathit{Leaf}\ 7), \\
& \qquad \mathit{Leaf}\ 8), \\
& \quad \mathit{Fork} \\
& \qquad (\mathit{Fork}\ (\mathit{Leaf}\ 9, \mathit{Leaf}\ 3), \\
& \qquad\quad \mathit{Leaf}\ 2))
\end{aligned}
$$



Inductive parametric types

$$k\ (Leaf\ a) = Leaf\ (a * a)$$
$$k\ (Fork\ (l, r)) = Fork\ (k\ l, k\ r)$$

$$\begin{aligned}
&k\ t = Fork \\
&\quad (Fork \\
&\qquad (Fork\ (Leaf\ 1, Leaf\ 49), \\
&\qquad Leaf\ 64), \\
&\quad Fork \\
&\qquad (Fork\ (Leaf\ 81, Leaf\ 9), \\
&\qquad Leaf\ 4))
\end{aligned}$$

Inductive parametric types

$$k\ (Leaf\ a) = Leaf\ (a * a)$$
$$k\ (Fork\ (l, r)) = Fork\ (k\ l, k\ r)$$

Inductive parametric types

$$k\ (Leaf\ a) = Leaf\ (a * a)$$
$$k\ (Fork\ (l, r)) = Fork\ (k\ l, k\ r)$$

By analogy with $map\ f$:

$$k\ f\ (Leaf\ a) = Leaf\ (f\ a)$$
$$k\ f\ (Fork\ (l, r)) = Fork\ (k\ f\ l, k\ f\ r)$$

Inductive parametric types

$$map\ id = id$$
$$map\ (f \cdot g) = (map\ f) \cdot (map\ g)$$

Inductive parametric types

$$map\ id = id$$
$$map\ (f \cdot g) = (map\ f) \cdot (map\ g)$$

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

?

Inductive parametric types

$$\begin{cases} k\ f\ (Leaf\ a) = Leaf\ (f\ a) \\ k\ f\ (Fork\ (l, r)) = Fork\ (k\ f\ l, k\ f\ r) \end{cases}$$

Inductive parametric types

$$\begin{cases} k\ f\ (Leaf\ a) = Leaf\ (f\ a) \\ k\ f\ (Fork\ (l, r)) = Fork\ (k\ f\ l, k\ f\ r) \end{cases}$$

$$\Leftrightarrow \qquad \{ \text{ go pointfree } \}$$

$$\begin{cases} k\ f \cdot Leaf = Leaf \cdot f \\ k\ f \cdot Fork = Fork \cdot (k\ f \times k\ f) \end{cases}$$

Inductive parametric types

$$\begin{cases} k\ f\ (Leaf\ a) = Leaf\ (f\ a) \\ k\ f\ (Fork\ (l, r)) = Fork\ (k\ f\ l, k\ f\ r) \end{cases}$$

$\Leftrightarrow$ { go pointfree }

$$\begin{cases} k\ f \cdot Leaf = Leaf \cdot f \\ k\ f \cdot Fork = Fork \cdot (k\ f \times k\ f) \end{cases}$$

$\Leftrightarrow$ { $+$-Eq ; $+$-fusion ; $+$-absorption }

Inductive parametric types

$$k\ f \cdot [Leaf, Fork] = [Leaf, Fork] \cdot (f + k\ f \times k\ f)$$

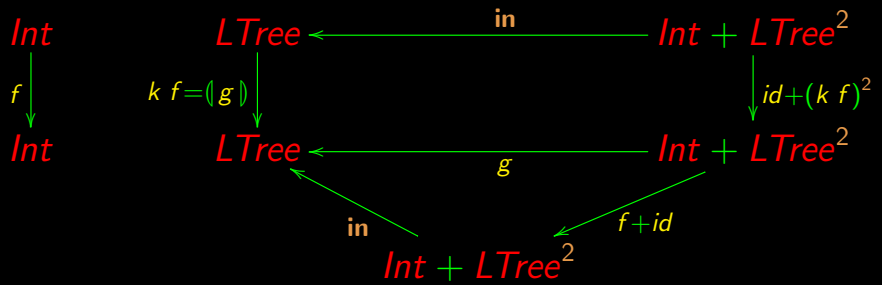
Inductive parametric types

$$k\ f \cdot [Leaf\,, Fork] = [Leaf\,, Fork] \cdot (f + k\ f \times k\ f)$$

$$\Leftrightarrow \quad \{\ [Leaf\,, Fork] = \mathbf{in}\ ;\ \text{functor-}+\ \}$$

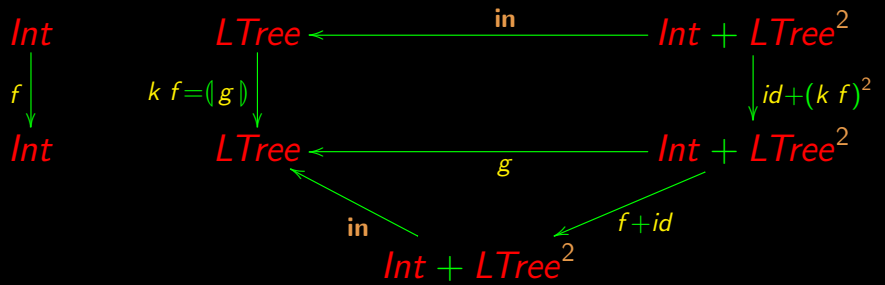
$$k\ f \cdot \mathbf{in} = \mathbf{in} \cdot (f + id) \cdot (id + k\ f \times k\ f)$$

Inductive parametric types



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

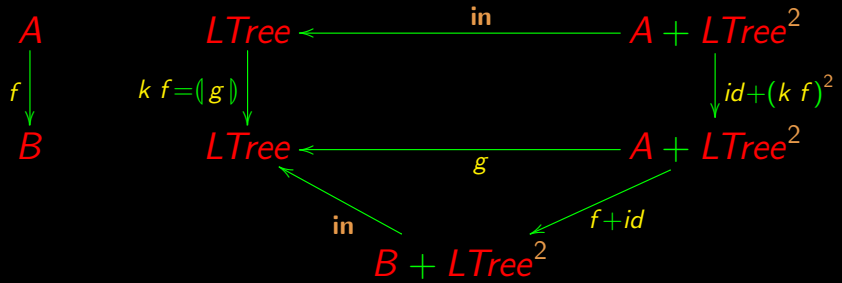
Inductive parametric types



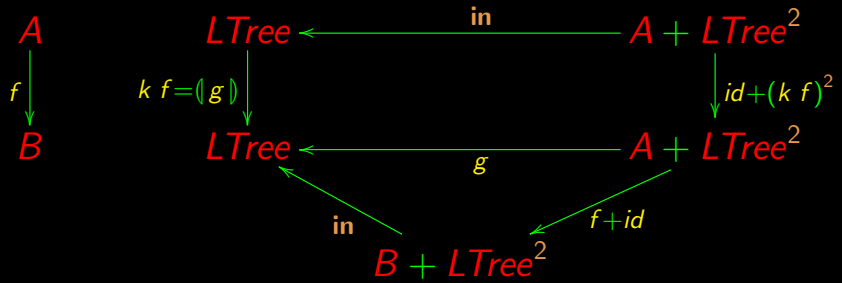
$$k\ f \cdot \mathbf{in} = \mathbf{in} \cdot (f + id) \cdot (id + k\ f \times k\ f)$$

NB: $LTree^2$ abbreviates $LTree \times LTree$

Inductive parametric types

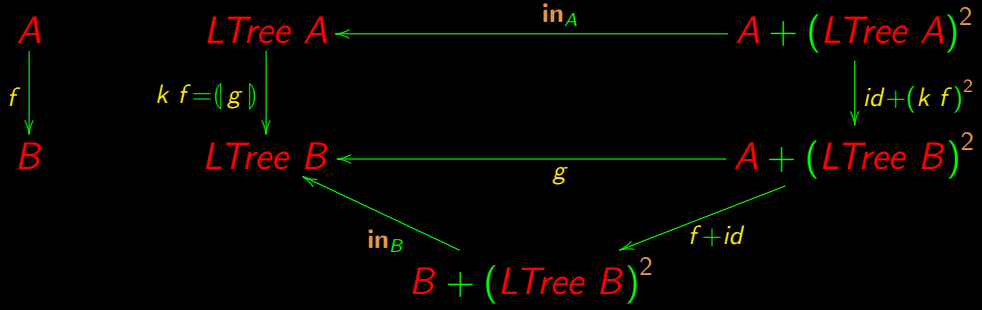


Inductive parametric types

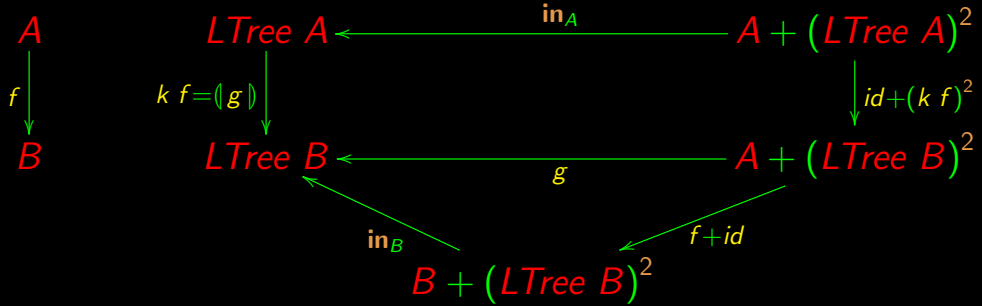


!!

Inductive parametric types



Inductive parametric types



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

Follow-up: `cp2223s12.pdf`