

Cálculo de Programas

Class T01 — 15/18-Sep

Introduction to the course.

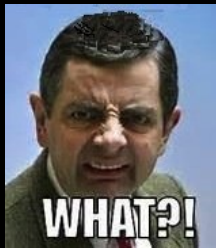
*Website: **<https://haslab.github.io/CP>***

Other administrative details.

J.N. Oliveira

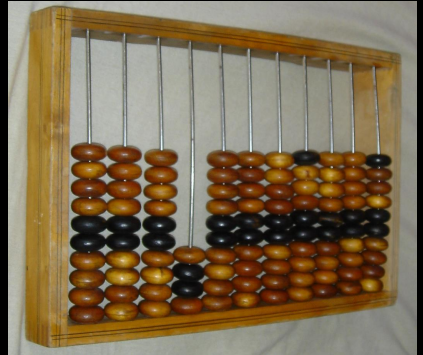


Programming by Calculation



Program versus Calculus

- ▶ **Calculus** — abacus **pebble**
- ▶ **Program** — *pro* ('before')
+ *graphein* ('write')



Programs...

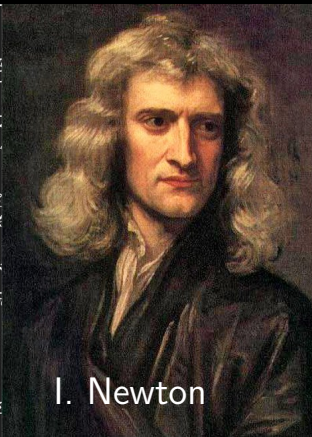
```
wc.c
1 1 #define YES 1
2 2 #define NO 0
3 3 main()
4 4 {
5 5 int c, nl, nw, nc, inword ;
6 6 inword = NO ;
7 7 nl = 0;
8 8 nw = 0;
9 9 nc = 0;
10 10 c = getchar();
11 11 while ( c != EOF ) {
12 12     nc = nc + 1;
13 13     if ( c == '\n' )
14 14         nl = nl + 1;
15 15     if ( c == ' ' || c == '\n' || c == '\t' )
16 16         inword = NO;
17 17     else if ( inword == NO ) {
18 18         inword = YES ;
19 19         nw = nw + 1;
20 20     }
21 21     c = getchar();
22 22 }
23 23 printf("%d",nl);
24 24 printf("%d",nw);
25 25 printf("%d",nc);
26 26 }
```

Calculus



G. Leibniz

$$\begin{aligned}
 & \gamma_0 = 2\pi \\
 & (\pi k; 0), k \in \mathbb{Z}, \\
 & \left(\frac{\pi}{2} + 2\pi k\right) - \max, \uparrow \left(-\frac{\pi}{2}\right) \\
 & \frac{1}{2}; \quad \frac{\sin^2 \alpha}{2} = \frac{1 - \cos 2\alpha}{2}; \quad \frac{\sin^2 \alpha}{2} \\
 & \cos^2 \alpha + \sin^2 \alpha = 1; \quad \sin(\alpha + \beta) = \sin \alpha \cos \beta + \cos \alpha \sin \beta \\
 & = \frac{\sin \alpha}{\cos \alpha}; \quad \frac{\sin \alpha}{\cos \alpha} = \tan \alpha; \quad \tan(\alpha + \beta) = \frac{\tan \alpha + \tan \beta}{1 - \tan \alpha \tan \beta} \\
 & = \frac{\cos \alpha}{\sin \alpha}; \quad \boxed{\sin x = a} \quad \boxed{x = (-1)^k \arcsin a + \pi k} \\
 & \frac{1}{2}; \quad \frac{\pi}{2} + 2\pi k; \quad \frac{\pi}{2} + 2\pi k, k \in \mathbb{Z}; \\
 & \frac{1}{2} = \frac{1 - \cos 2\alpha}{2}; \quad \cos^2 \alpha = \frac{1 + \cos 2\alpha}{2} \\
 & B) = \cos \alpha \cos \beta + \sin \alpha \sin \beta.
 \end{aligned}$$



I. Newton

```

6  #define O(b,f,u,s,c,a)b(){int o=f();switch(*p++){X u:_ o s b(
7  #define t(e,d,_,C)X e:f=fopen(B+d,_,C);C;fclose(f)
8  #define U(y,z)while(p=Q(s,y))*p++=z,*p=' '
9  #define N for(i=0;i<11*R;i++)m[i]&&
10 #define I "%d %s\n",i,m[i]
11 #define X ;break;case
12 #define _ return
13 #define R 999
14 typedef char*A;int*C,E[R],L[R],M[R],P[R],l,i,j;char B[R],F[2]
15 (),p,q,x,y,z,s,d,f,fopen();A Q(s,o)A s,o;{for(x=s;*x;x++){for
16 *z;y++)z++;if(z>o&&!*z)_ x;}_ 0;}main(){m[11*R]="E";while(p
17 )switch(*B){X'R':C=E;l=1;for(i=0;i<R;P[i++]=0);while(l){while
18 (!Q(s,"\"")){U("<>",'#');U("<=", '$');U(">=", '!');}d=B;while(*
19 ++;if(j&1||!Q("\t",F))*d++=*s;s++;}*d--=j=0;if(B[1]!='=')swi
20 X'R':B[2]!='M'&&(l=*--C)X'I':B[1]=='N'?gets(p=B),P[*d]=S():(*
21 =B+2,S())&&(p=q+4,l=S()-1)X'P':B[5]==' '?*d=0,puts(B+6):(p=B+
22 ()))X'G':p=B+4,B[2]=='S'&&(*C++=l,p++),l=S()-1 X'F':*(q=O(B,"

```

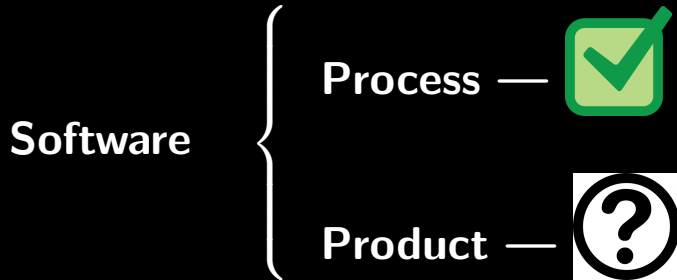
```

6  #define O(b,f,u,s,c,a)b(){int o=f();switch(*p++){X u:_ o s b(
7  #define t(e,d,_,C)X e:f=fopen(B+d,_,C);fclose(f)
8  #define U(y,z)while(p=Q(s,y))*p++=z,*
9  #define N for(i=0;i<11*R;i++)m[i]&
10 #define I "%d %s\n",i,m[i]
11 #define X ;break;case
12 #define _ return
13 #define R 999
14 typedef char*A;int*C,E[R],L[R],F[2]
15 (),p,q,x,y,z,s,d,f,fopen();A C){for
16 *z;y++)z++;if(z>0&&!*z)_ x;}while(p
17 )switch(*B){X'R':C=E;l=1;for(i=0;i<R;i++){while
18 (!Q(s,"\"")){U("<>","#");U("<=",
19 ++;if(j&1||!Q("\t",F))*d++=*s;s++;if(s[1]!='=')swi
20 X'R':B[2]!='M'&&(l=*--C)X'I':B[1]=='N'?g=p:B,P[*d]=S():(*
21 =B+2,S())&&(p=q+4,l=S()-1)X'P':B[5]==''?*d=0,puts(B+6):(p=B+
22 ())X'G':p=B+4,B[2]=='S'&&(*C++=l,p++),l=S()-1 X'F':*(q=O(B,"

```

?

Software as a problem





THE
ESSENCE
OF
SOFTWARE

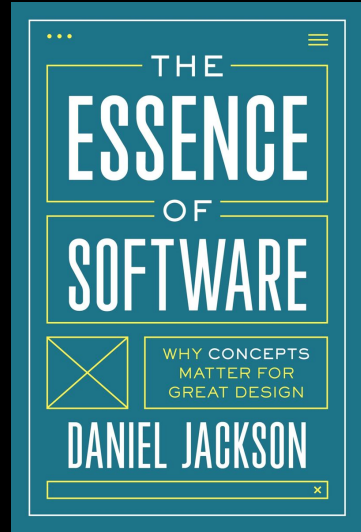


WHY CONCEPTS
MATTER FOR
GREAT DESIGN

DANIEL JACKSON



”(...) 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



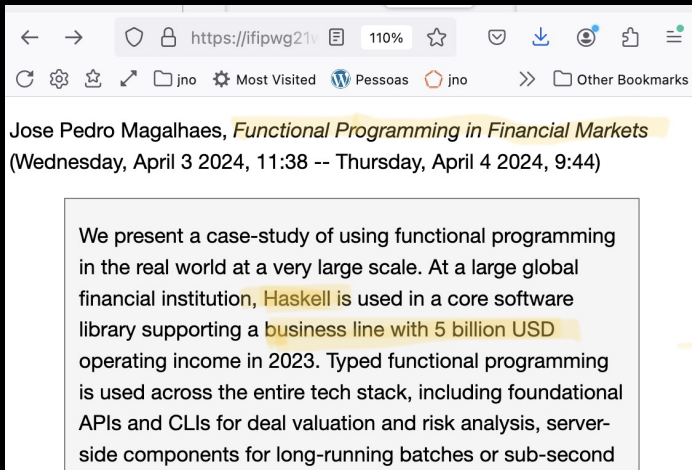
Urgently needed in software design



Functional Programming



FP = \$\$\$...



The screenshot shows a web browser window with the address bar displaying `https://ifipwg21v` at 110% zoom. The browser's bookmark bar includes folders for 'jno', 'Most Visited', 'Pessoas', and 'jno', along with a link to 'Other Bookmarks'. The main content area displays a presentation slide by Jose Pedro Magalhaes titled *Functional Programming in Financial Markets*, dated Wednesday, April 3 2024, 11:38 -- Thursday, April 4 2024, 9:44. The slide text, enclosed in a light blue box, describes a case study of using functional programming at a large global financial institution, highlighting the use of Haskell in a core software library supporting a business line with 5 billion USD operating income in 2023. It also mentions the use of typed functional programming across the entire tech stack, including foundational APIs and CLIs for deal valuation and risk analysis, and server-side components for long-running batches or sub-second operations.

← → `https://ifipwg21v` 110% ☆

🔄 ⚙️ ☆ 🔗 jno ⚙️ Most Visited 🌐 Pessoas 📁 jno >> 📁 Other Bookmarks

Jose Pedro Magalhaes, *Functional Programming in Financial Markets*
(Wednesday, April 3 2024, 11:38 -- Thursday, April 4 2024, 9:44)

We present a case-study of using functional programming in the real world at a very large scale. At a large global financial institution, Haskell is used in a core software library supporting a business line with 5 billion USD operating income in 2023. Typed functional programming is used across the entire tech stack, including foundational APIs and CLIs for deal valuation and risk analysis, server-side components for long-running batches or sub-second

FP = 'F' of "Future Programming"

Machines have evolved astronomically.

Supercomputers (incredibly **parallel** machines) are now essential to science, industry, and AI.

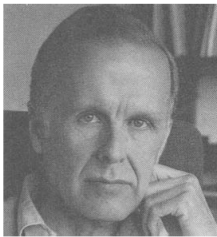
How are they programmed?



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

Computational Thinking

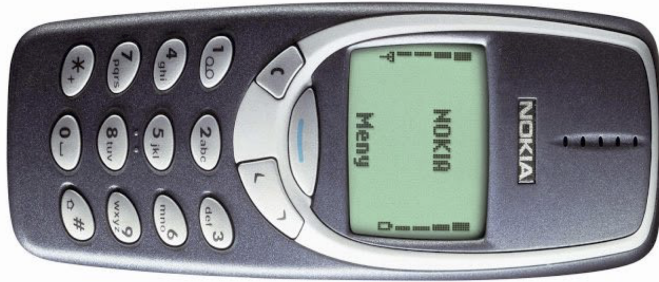
*"(...) Computational thinking is thinking **recursively**. It is **parallel processing**".*

In: Jeannette Wing,
Computational Thinking, CACM,
2006



Part I

Motivation — back to the late 1990s



From a mobile phone manufacturer

*(...) For each **list of calls** stored in the mobile phone (e.g. numbers dialled, 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.*

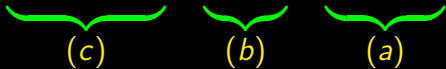
From a mobile phone manufacturer

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

From a mobile phone manufacturer

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

```
store c l = take 10 (nub (c:l))
```


(c) (b) (a)

Compare with ...

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

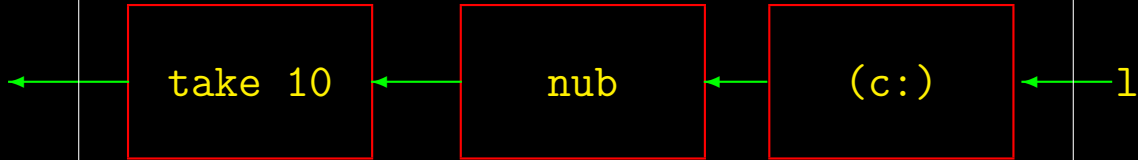
+ `filteratmost9` (next slide)

Compare with ...

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

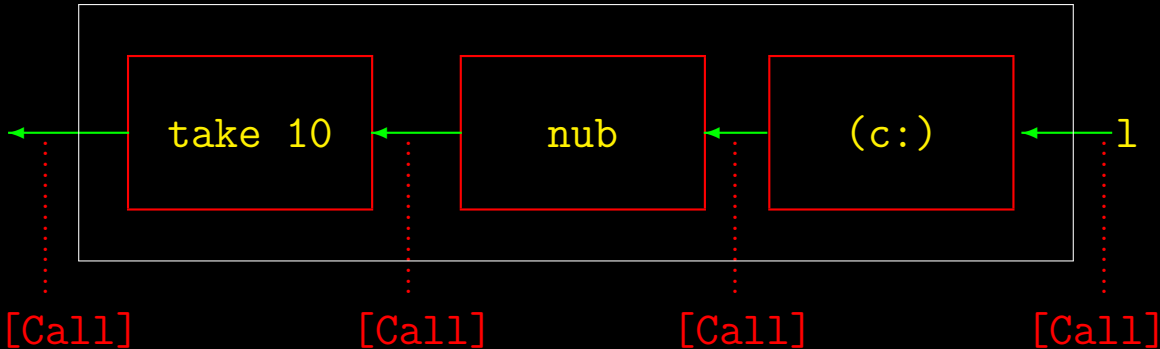
From a mobile phone manufacturer

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

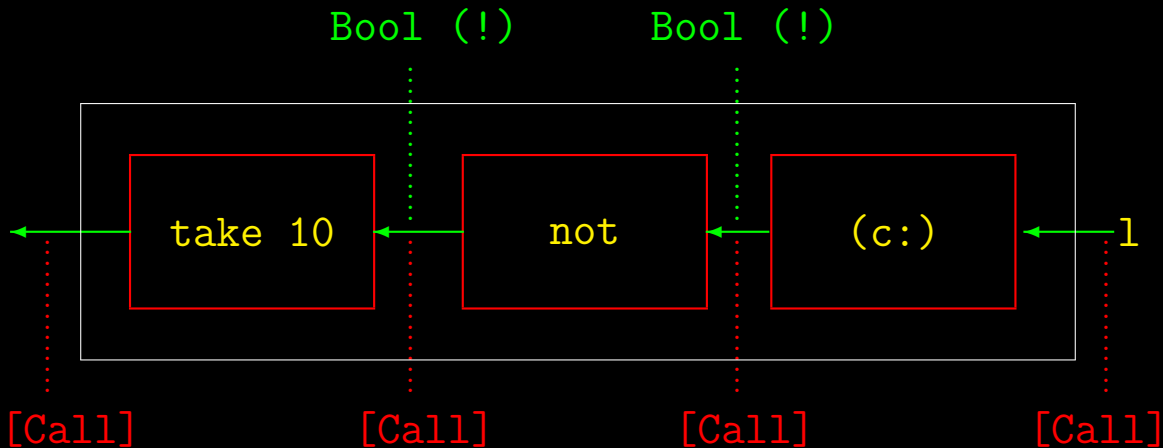


From a mobile phone manufacturer

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

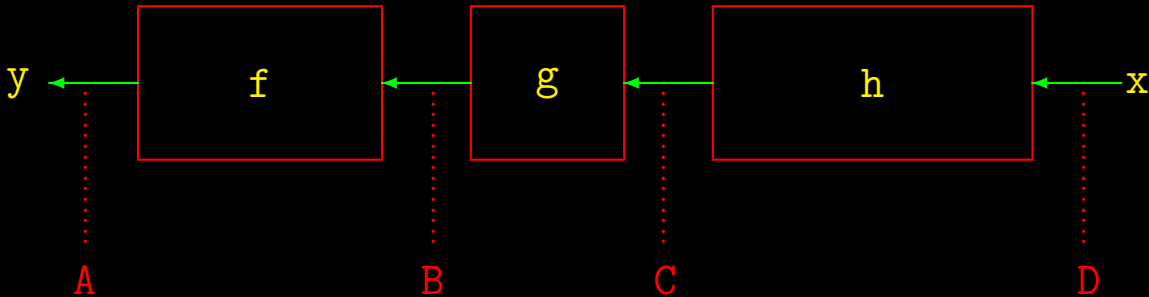


Uups!



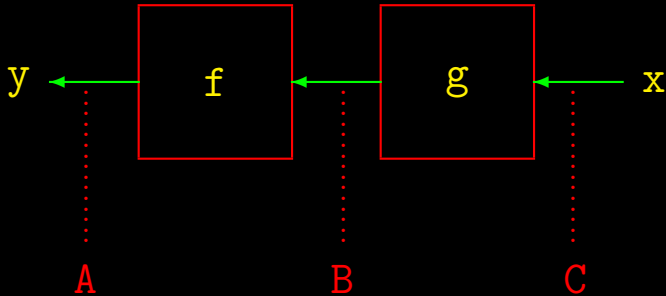
In general

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



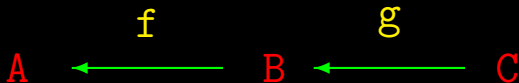
In general

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

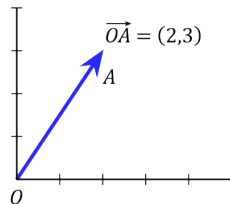
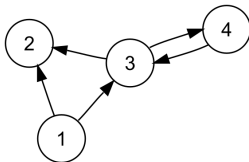
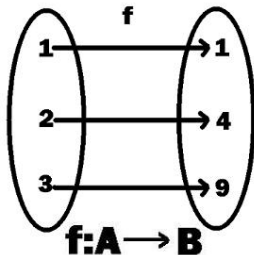
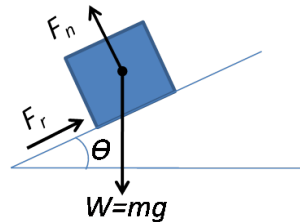
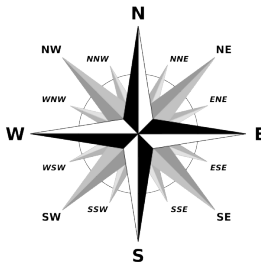


Simplification

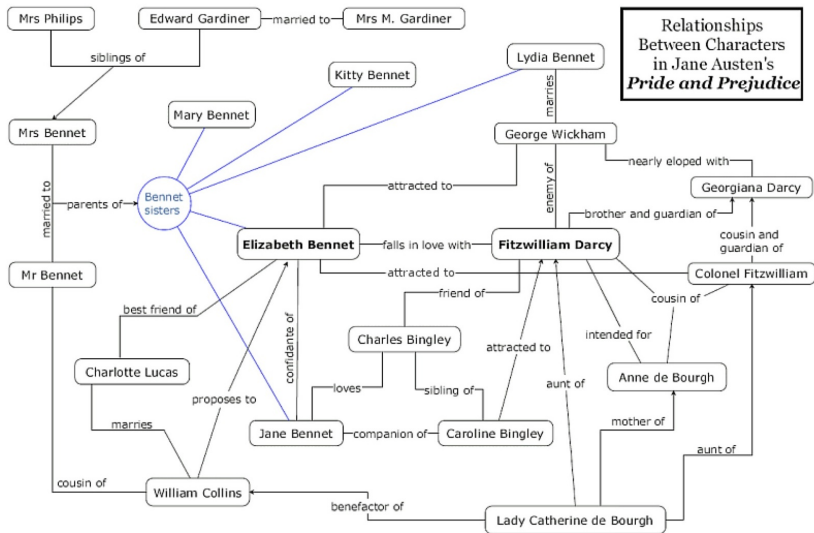
$$y = f(g \ x)$$



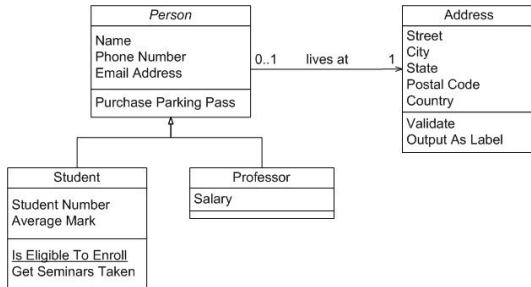
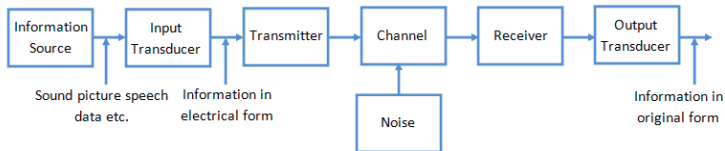
Check the pictures...



Arrows

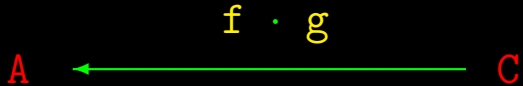
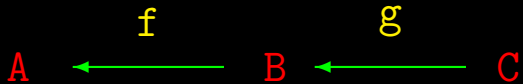


More arrows...

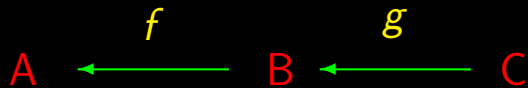


Composition

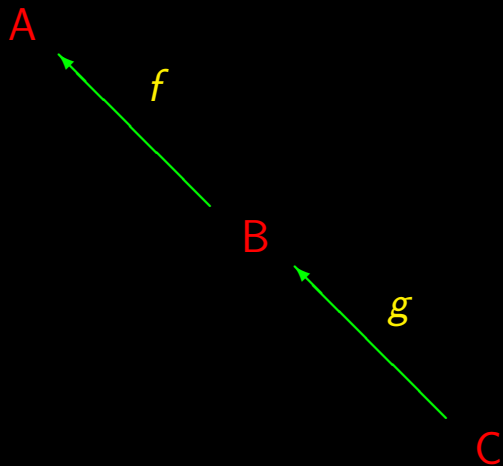
$$y = f(g \ x)$$



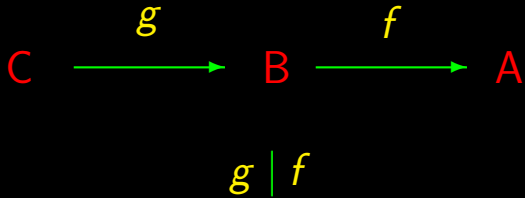
$$y = (f \cdot g) \ x$$



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



Cf. Unix/Linux pipes



Composition

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

Composition

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

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

Composition

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

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

$$f \cdot g \cdot h$$

$$a + b + c$$

Composition

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

Composition

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

i.e.

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

Composition

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

i.e.

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

the same as

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

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

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

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

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

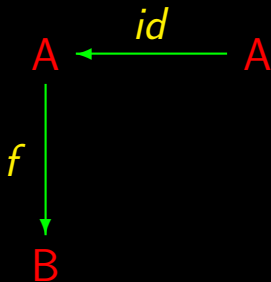
$$f \cdot ? = ? \cdot f = f$$

$$A \xleftarrow{id} A$$

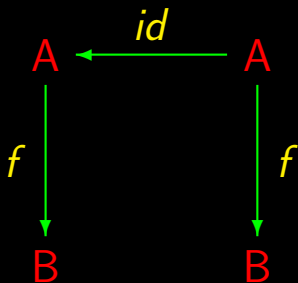
$$A \xleftarrow{id} A$$

$$id\ a = a$$

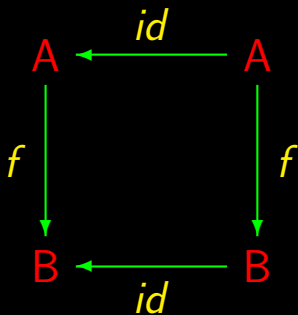
Identity



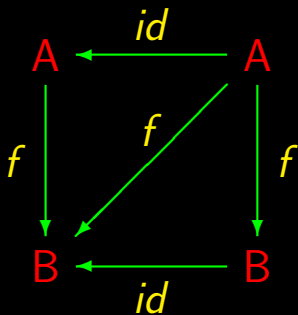
Identity



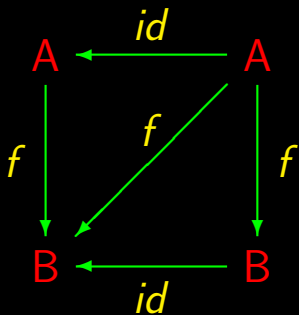
Identity



Identity



Identity



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

Composition and identity

Associativity:

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

Composition and identity

Associativity:

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

“Natural-*id*”:

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

$$f \cdot g$$

$$f \cdot g$$

$$f \times g ?$$

$$f \cdot g$$

$$f \times g ?$$

$$f + g ?$$

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

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

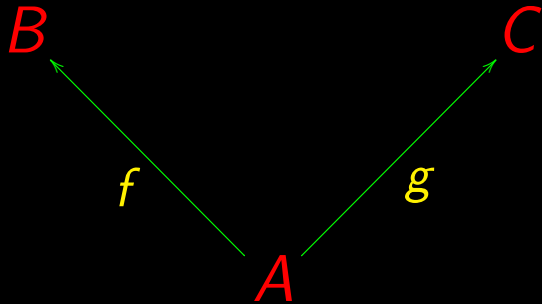
$$\text{Composition: } A \xrightarrow{f \cdot g} B$$

What about

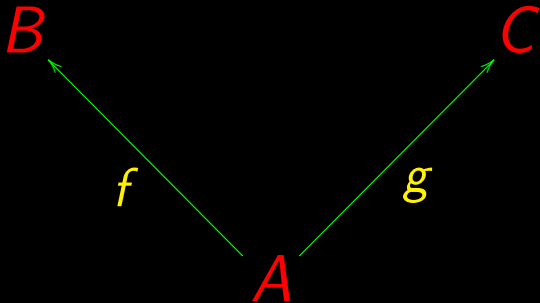
$$\begin{array}{ccc} D & \xrightarrow{f} & B \\ A & \xrightarrow{g} & C \end{array}$$

?

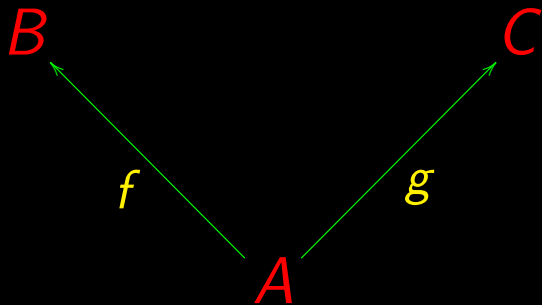
Try $D = A \dots$



Try $D = A \dots$



$f \ a \dots \ g \ a$



$(f\ a, g\ a)$

Cartesian product

$$B \times C = \{(b, c) \mid b \in B \wedge c \in C\}$$

Cartesian product

$$B \times C = \{(b, c) \mid b \in B \wedge c \in C\}$$

$$f \ a \in B$$

Cartesian product

$$B \times C = \{(b, c) \mid b \in B \wedge c \in C\}$$

$$f \ a \in B$$

$$g \ a \in C$$

Cartesian product

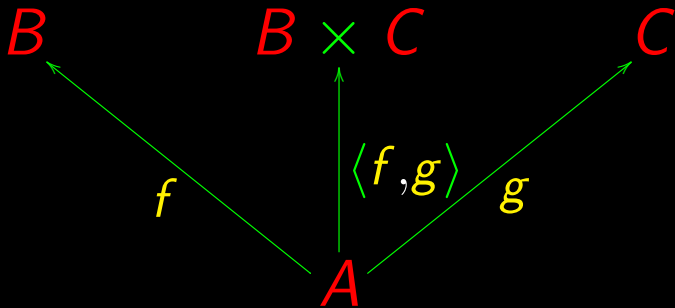
$$B \times C = \{(b, c) \mid b \in B \wedge c \in C\}$$

$$f \ a \in B$$

$$g \ a \in C$$

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

Parallel (“split”)



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

Product

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

Product

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

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

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

Product

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

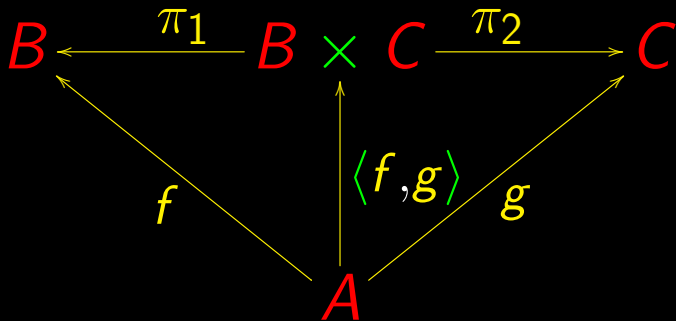
$$\pi_1 : A \times B \rightarrow A$$

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

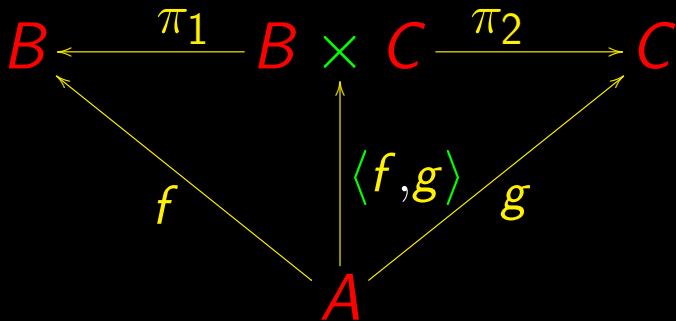
$$\pi_2 : A \times B \rightarrow B$$

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

Product

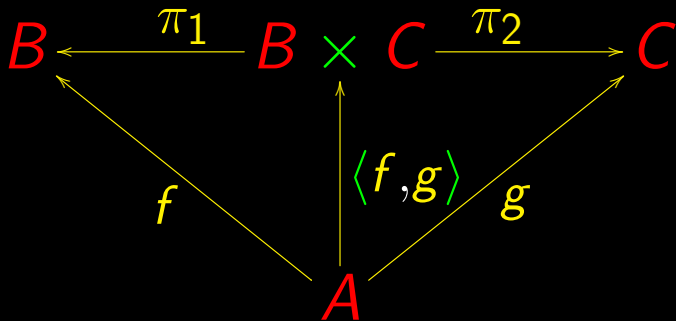


Product



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

Product



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

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

Parallel combinator

$$\langle f, g \rangle$$

Parallel combinator

$$\langle f, g \rangle$$

f and g in parallel

f “split” g

Parallel combinator

$$\langle f, g \rangle$$

f and g in parallel

f “split” g

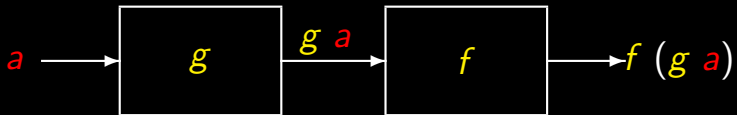
$$\langle f, g \rangle a = (f a, g a)$$

Cálculo de Programas

Class T02 — 22/25-Sep

Recall...

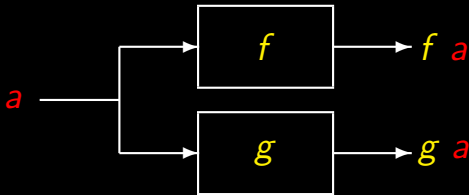
$$(f \cdot g) a = f (g a) \quad (2.6)$$



Sequential composition

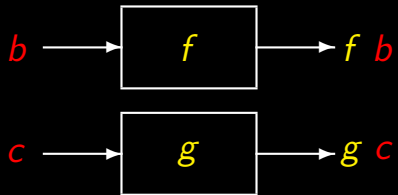
Recall...

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

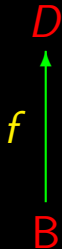


Parallel composition

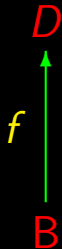
What about...



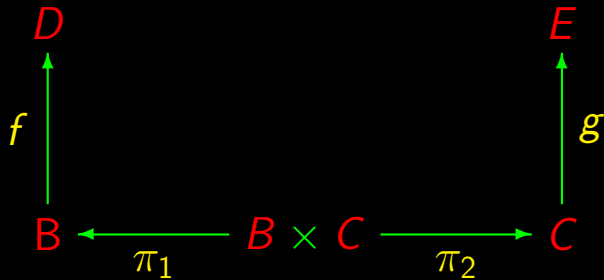
Let us draw...



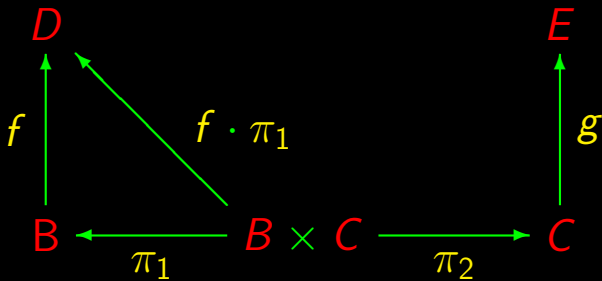
Let us draw...



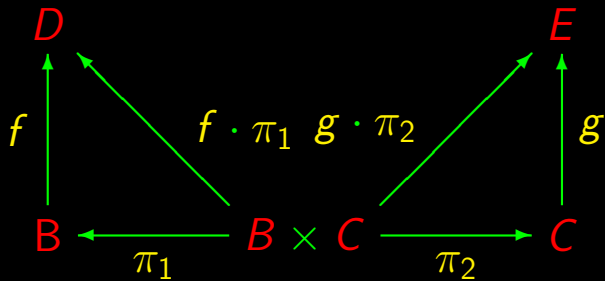
Let us draw...



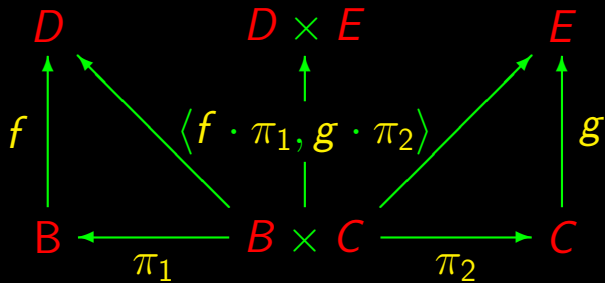
Let us draw...



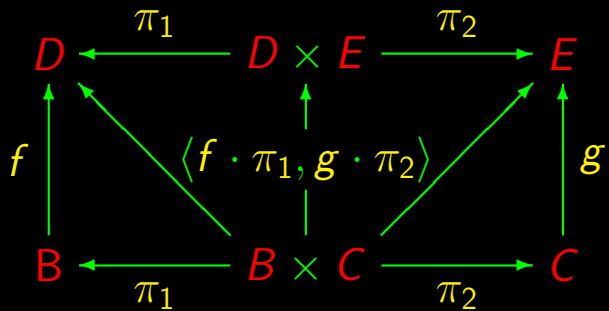
Let us draw...



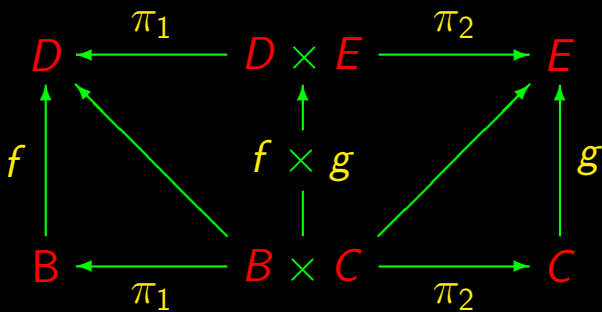
Let us draw...



Let us draw...



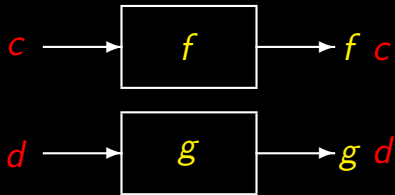
Product



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

Product

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



Asynchronous **parallel** composition ("product")

Summing up

$$f \cdot g$$

Sequential composition

Summing up

$f \cdot g$

$\langle f, g \rangle$

Sequential composition

Parallel composition

Summing up

$f \cdot g$

$\langle f, g \rangle$

Sequential composition

Parallel composition (**synchronous**)

Summing up

$f \cdot g$

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

Problem

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

$name : Iden \rightarrow Name$

$Iden = CC \cup NIF$

Problem!

$$CC = N$$

$$NIF = N$$

Problem!

$$CC = \mathbb{N}$$

$$NIF = \mathbb{N}$$

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

Problem!

$$CC = \mathbb{N}$$

$$NIF = \mathbb{N}$$

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

$$name : \mathbb{N} \rightarrow Name (!)$$

In general

We need to fix:

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

In general

We need to fix:

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

Let us start from

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

Disjoint union

Needing something like...

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

Disjoint union

Needing something like...

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

... we define:

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

Disjoint union

Clearly,

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

once we define:

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

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

Disjoint union

Types:

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

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

Disjoint union

Types:

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

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

Now think of:

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

Disjoint union

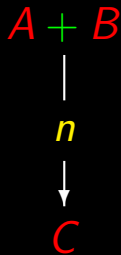
Types:

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

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

Now think of:

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



Disjoint union

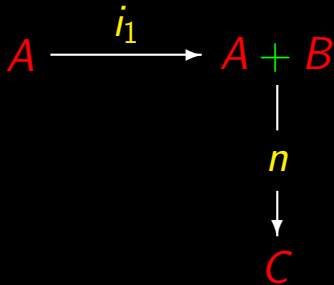
Types:

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

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

Now think of:

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



Disjoint union

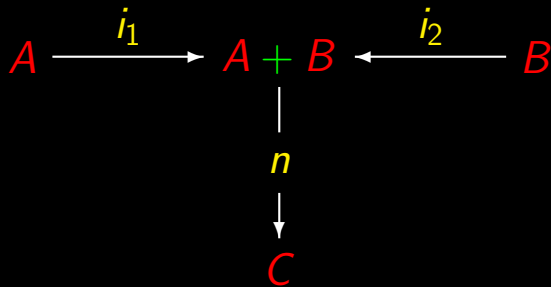
Types:

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

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

Now think of:

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



Disjoint union

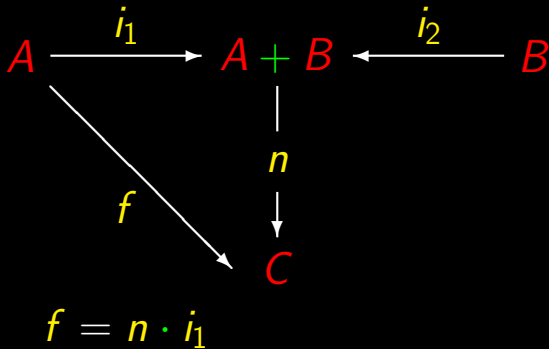
Types:

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

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

Now think of:

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



Disjoint union

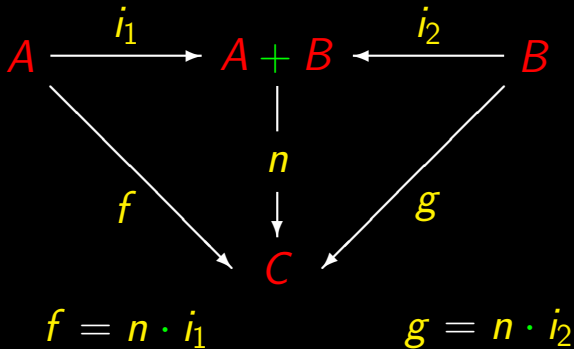
Types:

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

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

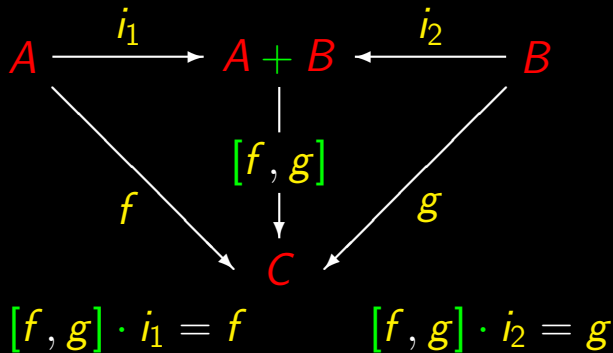
Now think of:

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



Alternative composition $[f, g]$

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



Alternative composition $[f, g]$

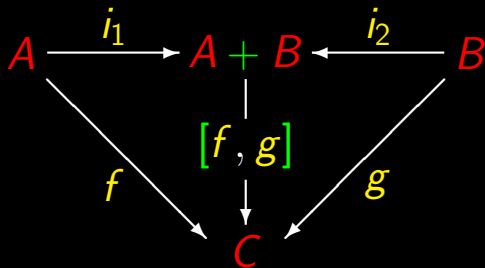
$$\begin{cases} [f, g] \cdot i_1 = f \\ [f, g] \cdot i_2 = g \end{cases}$$

Pointwise:

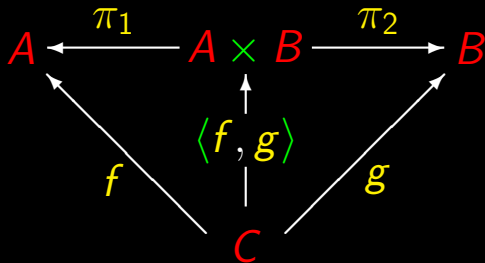
$$[f, g] : A + B \rightarrow C$$

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

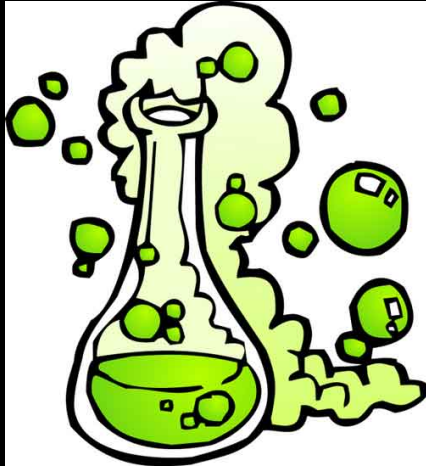
Compare...



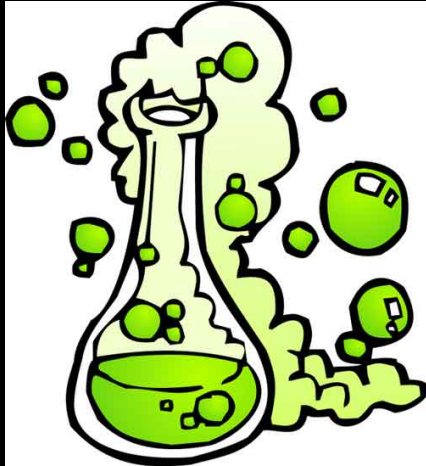
Compare...

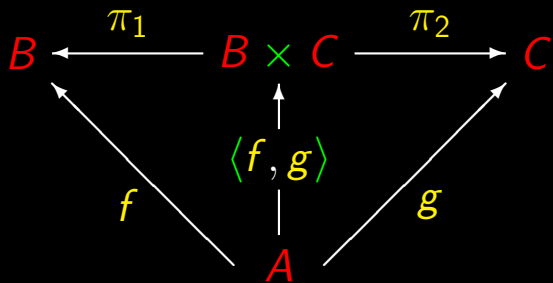


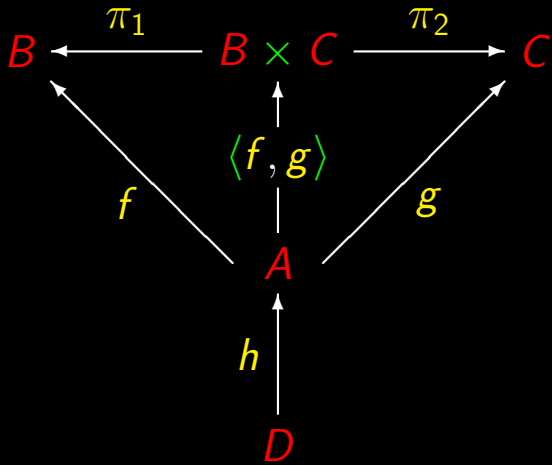
What about... Calculus?



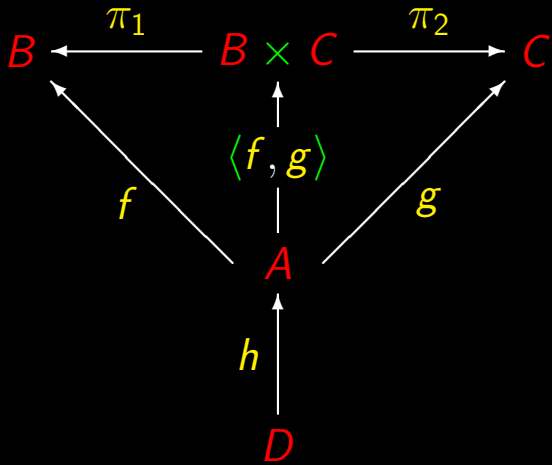
Calculus of parallel composition $\langle f, g \rangle$?

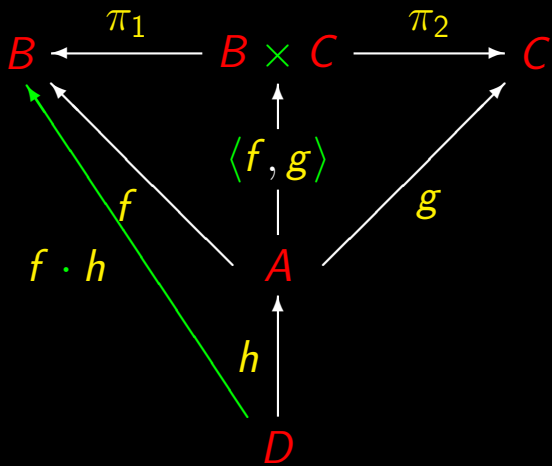


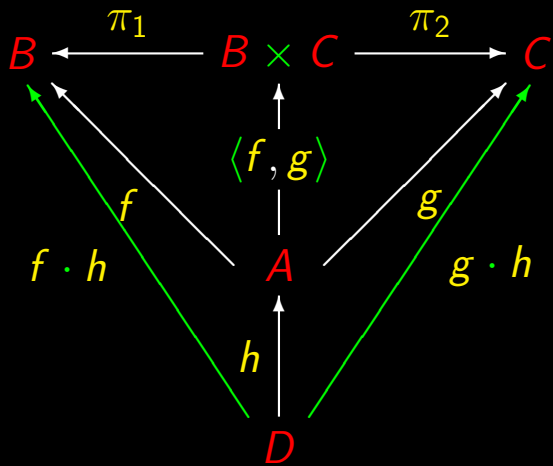


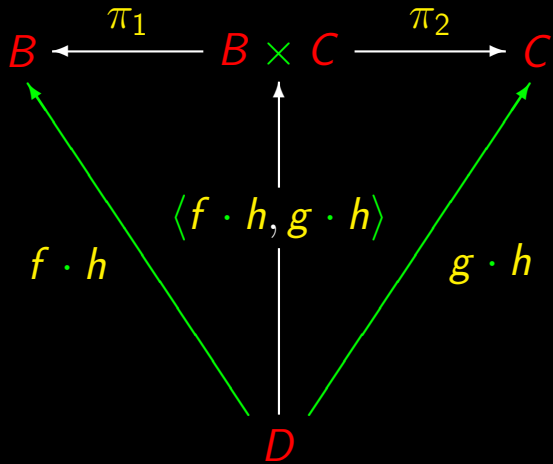


$$\begin{array}{ccccc} & \xleftarrow{\pi_1} & B \times C & \xrightarrow{\pi_2} & \\ & & \uparrow & & \\ & & \langle f, g \rangle \cdot h & & \\ & & \downarrow & & \\ & & D & & \end{array}$$





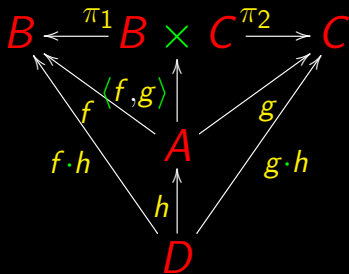


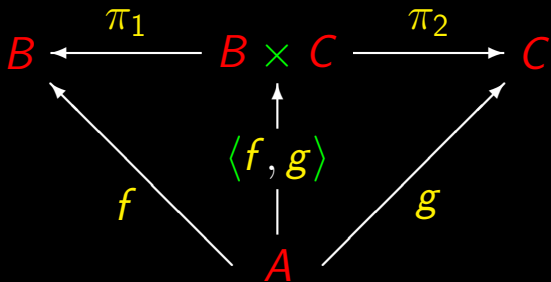


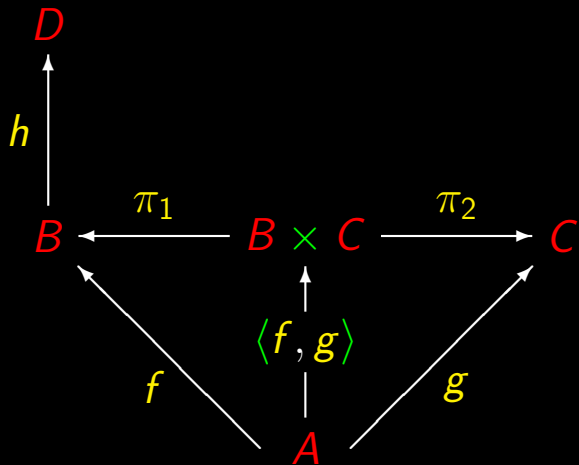
$$\begin{array}{ccccc}
 & \xleftarrow{\pi_1} & B \times C & \xrightarrow{\pi_2} & \\
 & & \uparrow & & \\
 \langle f, g \rangle \cdot h = \langle f \cdot h, g \cdot h \rangle & & & & \\
 & & \downarrow & & \\
 & & D & &
 \end{array}$$

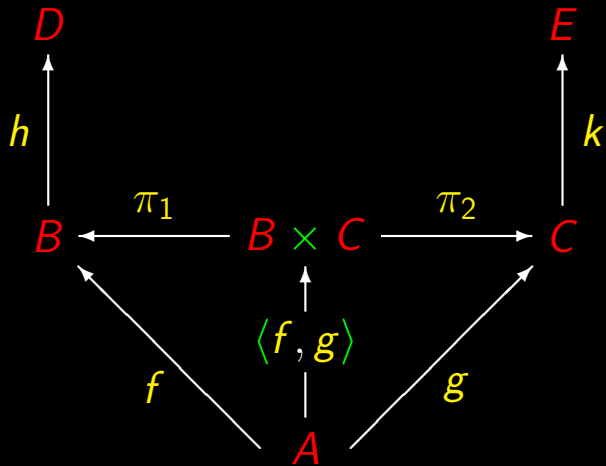
$\times$ -Fusion

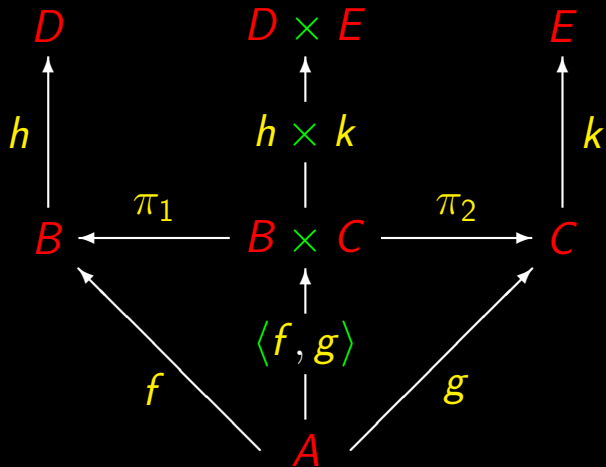
$$\langle f, g \rangle \cdot h = \langle f \cdot h, g \cdot h \rangle \quad (2.26)$$

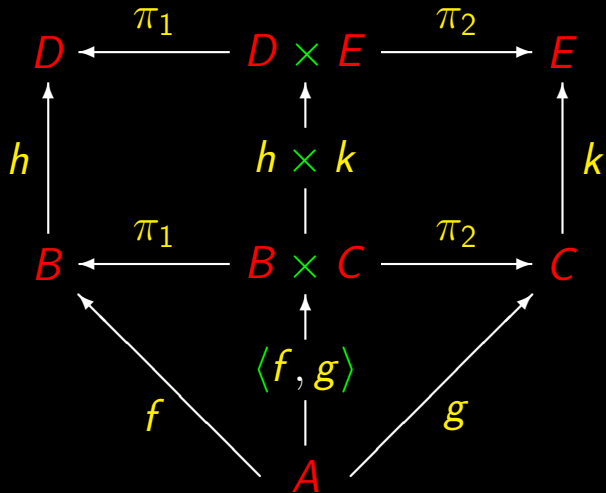


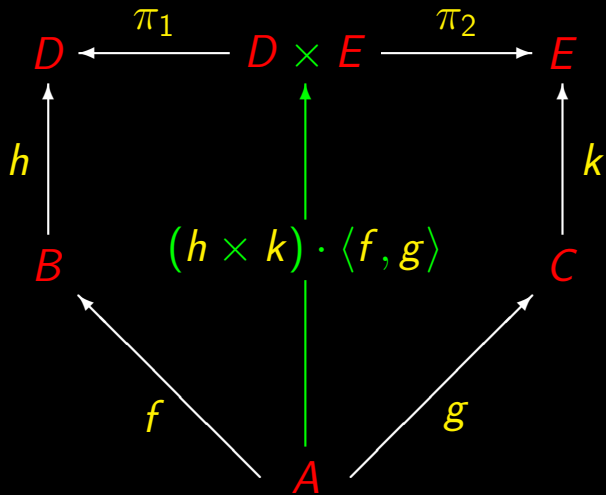


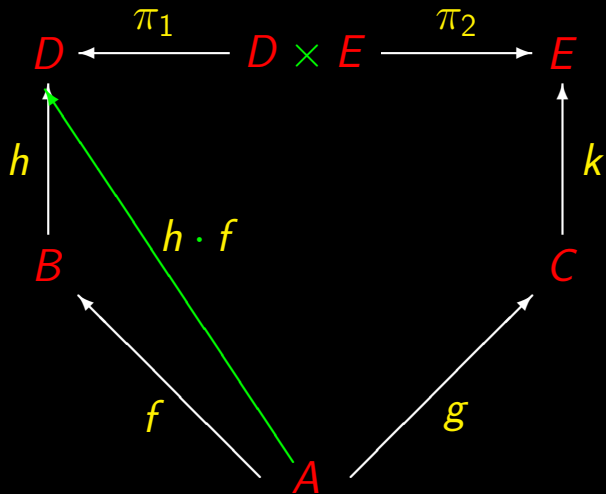


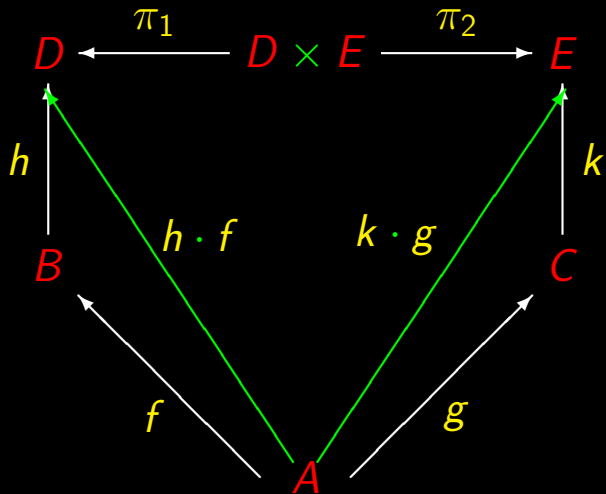


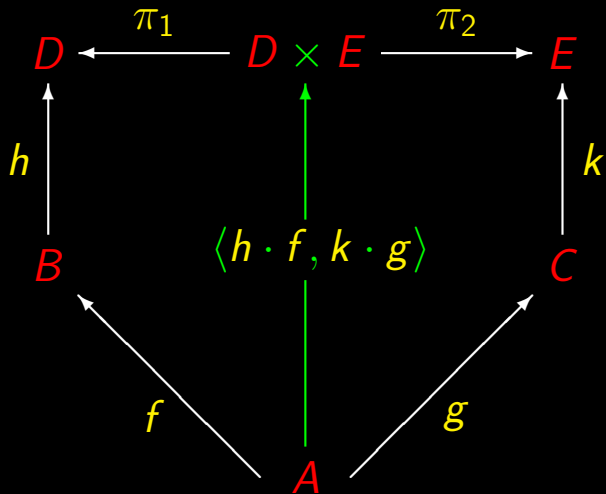


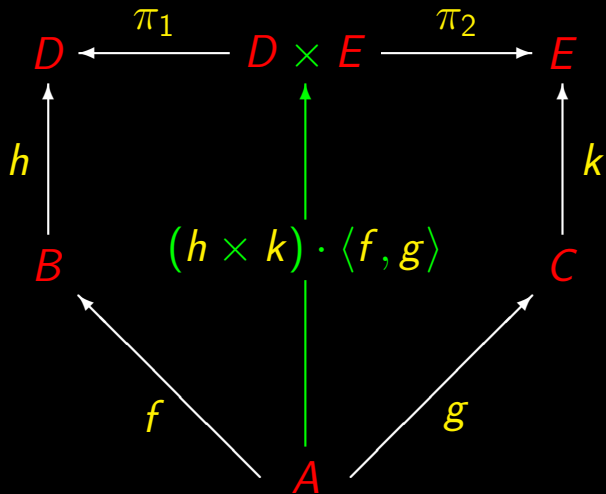






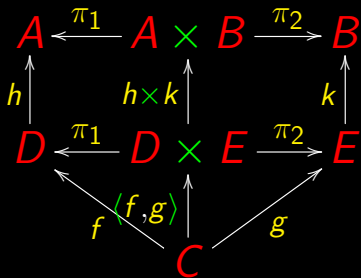


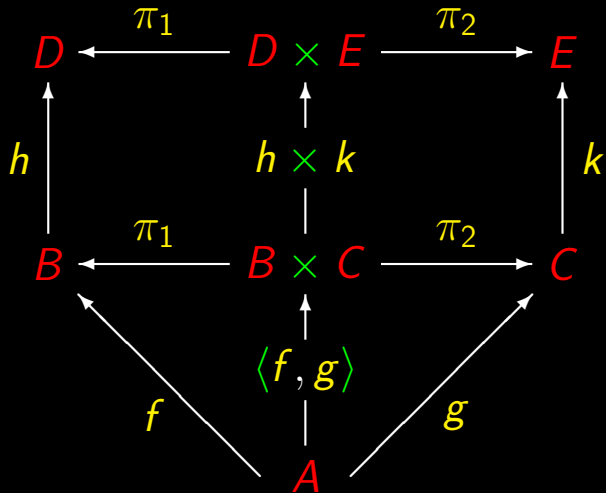


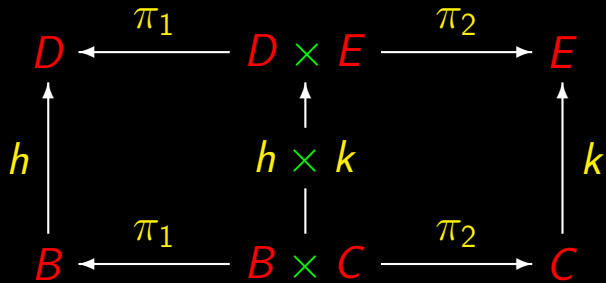


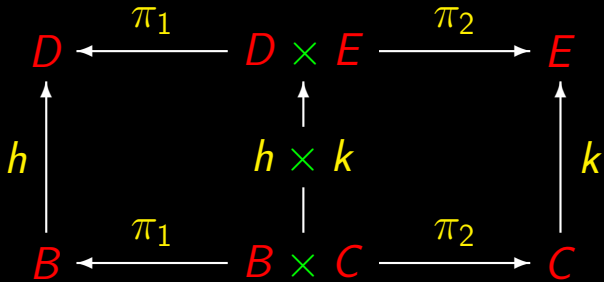
$\times$ -Absorption

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

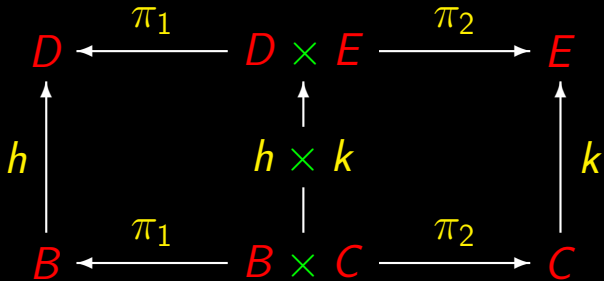








$$\pi_1 \cdot (h \times k) = h \cdot \pi_1$$



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

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

Natural- π_1 , natural- π_2

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

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

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

So far...

$f \cdot g$

$\langle f, g \rangle$

$f \times g$

$[f, g]$

Sequential composition

Parallel composition

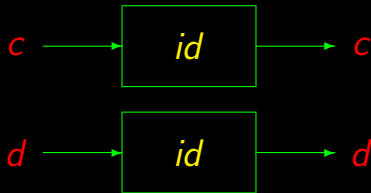
Product composition

Alternative composition

Compositional programming 😊

Functor-*id*-×

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



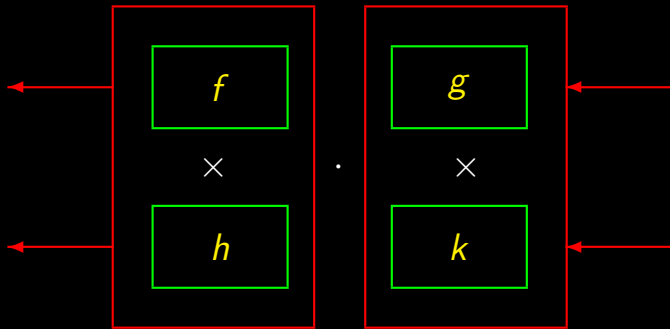
Product of two identities is an identity.

$\times$ -Functor

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

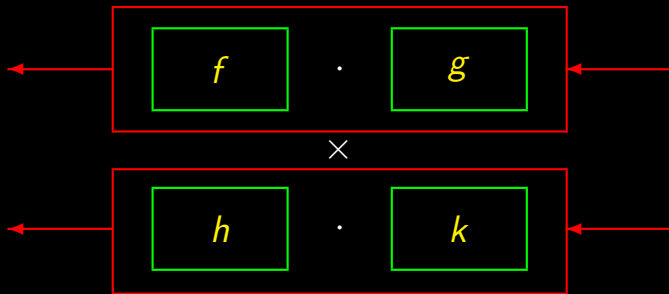
Composition of **products** is a **product** of compositions.

$\times$ -Functor



$$(f \times h) \cdot (g \times k)$$

$\times$ -Functor



$$(f \cdot g) \times (h \cdot k)$$

Two basic laws still missing

×-Reflexion

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

Two basic laws still missing

×-Reflexion

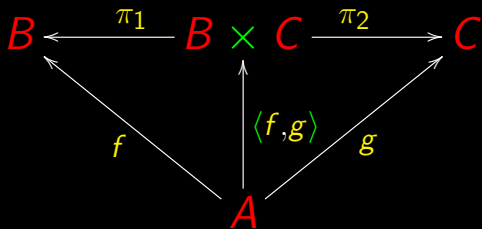
$$\langle \pi_1, \pi_2 \rangle = id \quad (2.32)$$

×-Eq

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

Before them, the most important...

Recall $\times$ -cancellation:



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

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

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

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

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

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

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

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

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

×-Universal

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

×-Universal

Existence

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

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

×-Universal

Unicity

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

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

Equations!

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

Equations!

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

Equations!

Problem

Solve the equation

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

for f and g .

Equations!

Calculation

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

Equations!

Calculation

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

Equations!

Calculation

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

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

Equations!

Calculation

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

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

Equations!

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

Equations!

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

Substituting:

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

Equations!

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

Substituting:

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

×-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

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

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

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

Calculation

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

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

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

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

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

Calculation

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

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

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

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

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

x-Eq !



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

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

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

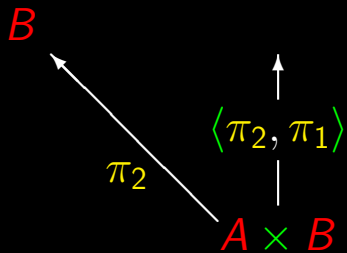
$$\langle \pi_1, \pi_2 \rangle = id$$

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

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

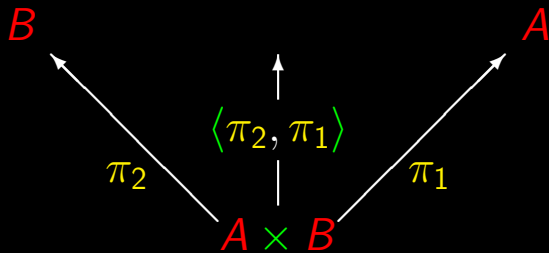
$$\langle \pi_1, \pi_2 \rangle = id$$

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



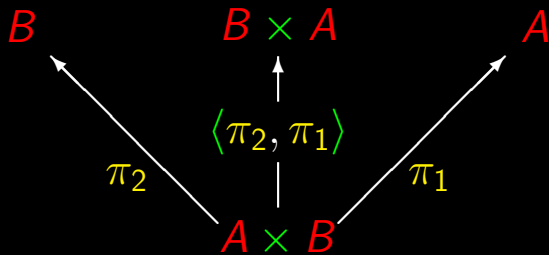
$$\langle \pi_1, \pi_2 \rangle = id$$

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



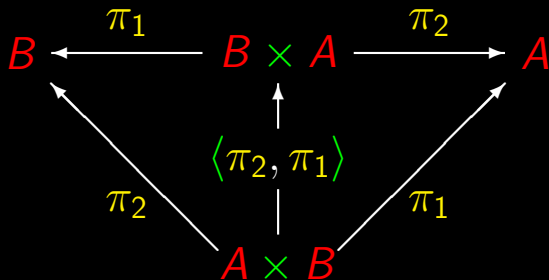
$$\langle \pi_1, \pi_2 \rangle = id$$

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



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

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

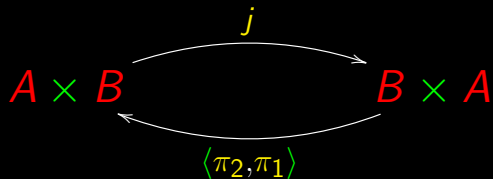


Problem A

Solve

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

for j in:

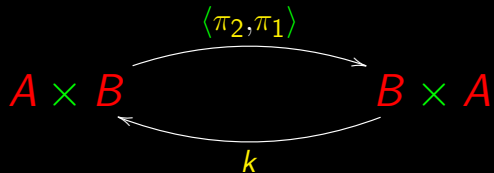


Problem B

Solve

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

for k in:



Calculation of problem B

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

Calculation of problem B

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

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

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

Calculation of problem B

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

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

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

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

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

Calculation of problem B

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

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

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

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

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

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

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

Calculation of problem B

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

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

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

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

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

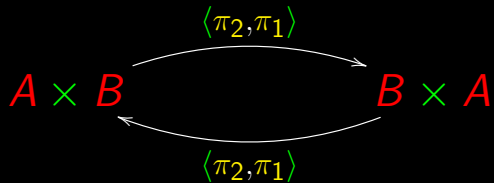
$$\Leftrightarrow \{ \text{trivial} \}$$

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

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

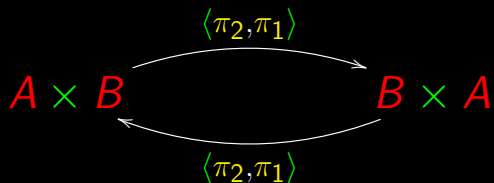
$$k = \langle \pi_2, \pi_1 \rangle$$

Swap



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

Swap



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

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

swap is its own inverse.

So far

$f \cdot g$

Sequential composition

So far

$f \cdot g$

$\langle f, g \rangle$

Sequential composition

Parallel composition

So far

$f \cdot g$

Sequential composition

$\langle f, g \rangle$

Parallel composition

Associativity

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

So far

$f \cdot g$

Sequential composition

$\langle f, g \rangle$

Parallel composition

Associativity

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

Associativity?

$$\langle \langle f, g \rangle, h \rangle \stackrel{?}{=} \langle f, \langle g, h \rangle \rangle$$

So far

$f \cdot g$

Sequential composition

$\langle f, g \rangle$

Parallel composition

Associativity

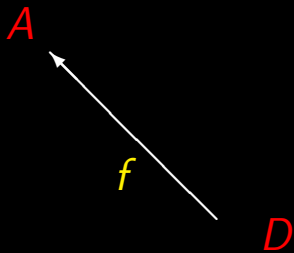
$$(f \cdot g) \cdot h = f \cdot (g \cdot h)$$

Associativity?

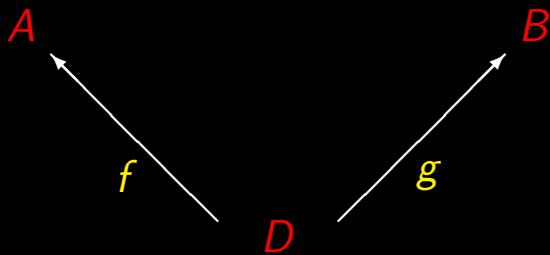
$$\langle \langle f, g \rangle, h \rangle \stackrel{?}{=} \langle f, \langle g, h \rangle \rangle$$

No! but...

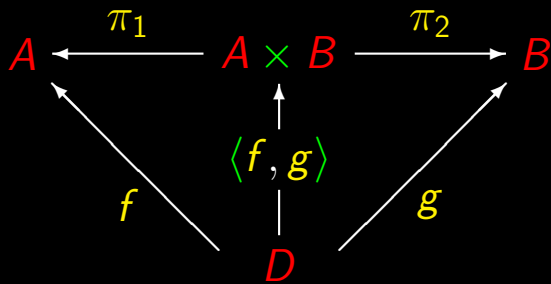
$$\langle \langle f, g \rangle, h \rangle$$



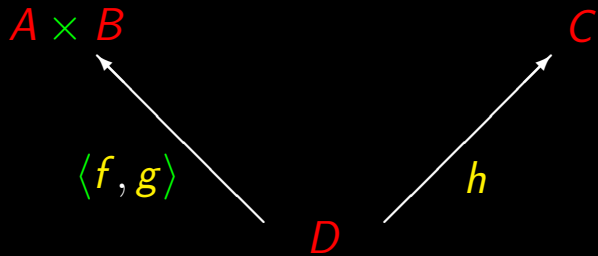
$$\langle \langle f, g \rangle, h \rangle$$



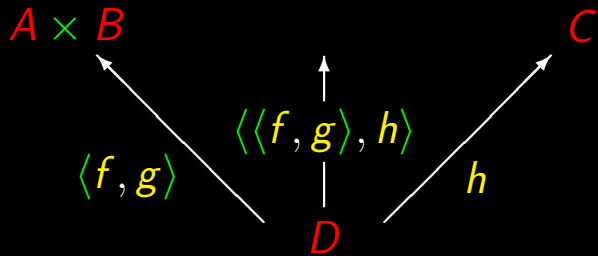
$$\langle \langle f, g \rangle, h \rangle$$



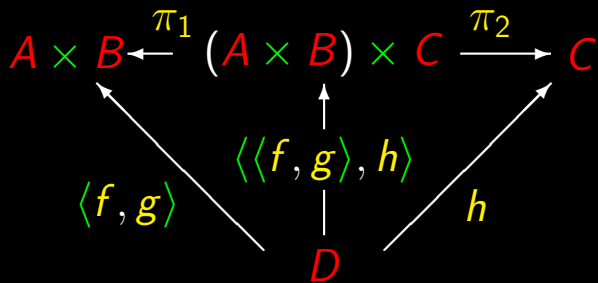
$$\langle \langle f, g \rangle, h \rangle$$



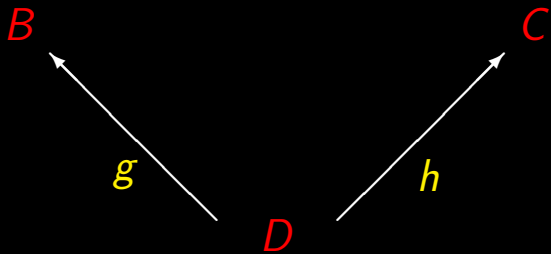
$$\langle \langle f, g \rangle, h \rangle$$



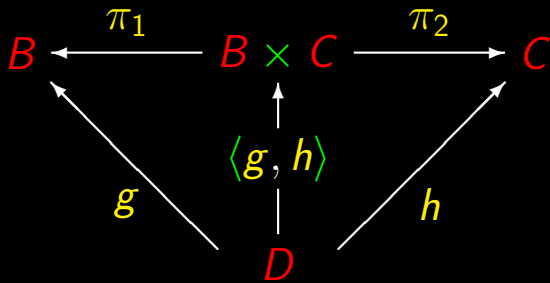
$$\langle \langle f, g \rangle, h \rangle$$



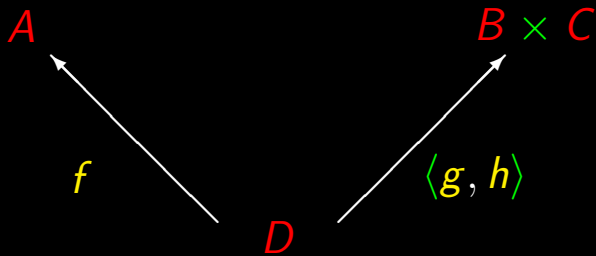
$$\langle f, \langle g, h \rangle \rangle$$



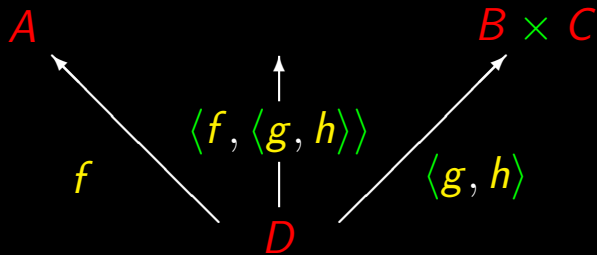
$$\langle f, \langle g, h \rangle \rangle$$



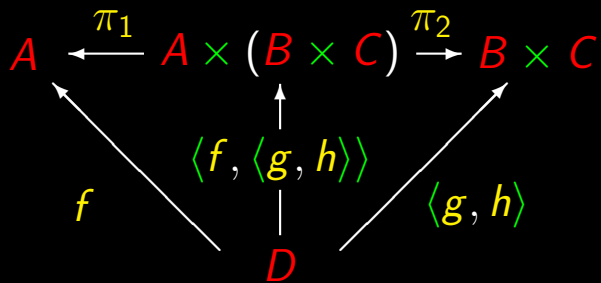
$$\langle f, \langle g, h \rangle \rangle$$



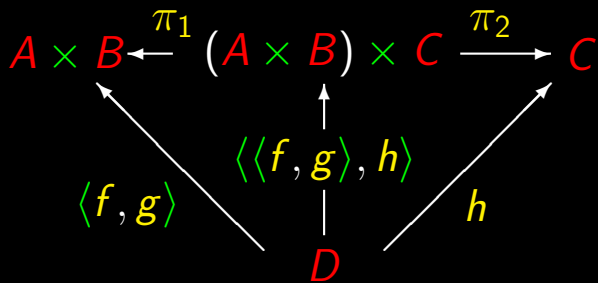
$$\langle f, \langle g, h \rangle \rangle$$



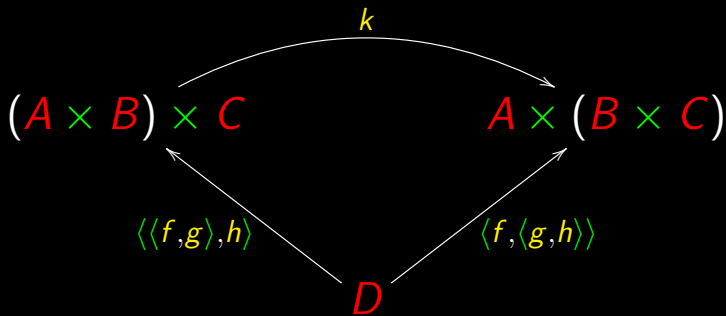
$$\langle f, \langle g, h \rangle \rangle$$



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



Find k



such that

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

Solving the equation for k

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

Solving the equation for k

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

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

Solving the equation for k

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

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



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

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

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

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

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

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

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

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

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

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

Substitute solutions

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

in

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

Substitute solutions

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

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

Slight improvement...

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

Slight improvement...

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

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

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

Slight improvement...

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

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

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

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

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

Analogy

In an **arithmetic** context, find k such that

$$k + (x - y) = x + 2$$

holds for any x and y .

Further assume that you **don't know** property:

$$a + b = c \Leftrightarrow a = c - b.$$

How can you find k ?

Analogy

Try to cancel $x - y$, ie. solve $x - y = 0$ for x .

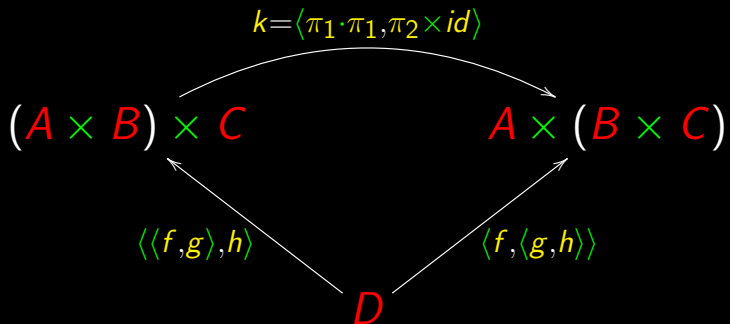
You get $x = y$.

Substitute x :

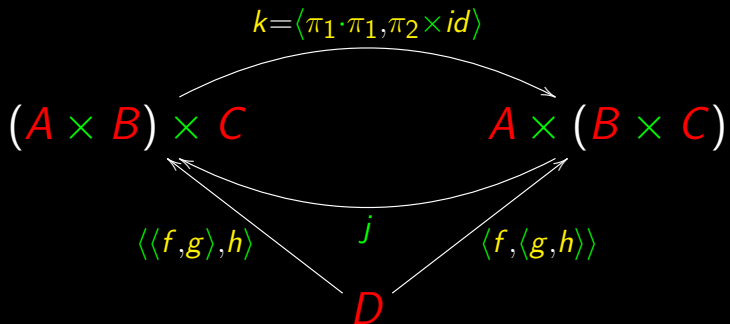
$$k + (y - y) = y + 2$$

You get $k = y + 2$.

Back to



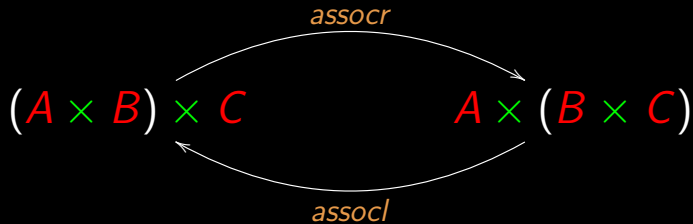
Opposite way



By a similar calculation

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

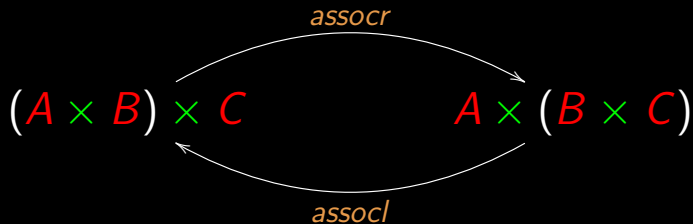
The two "associators"



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

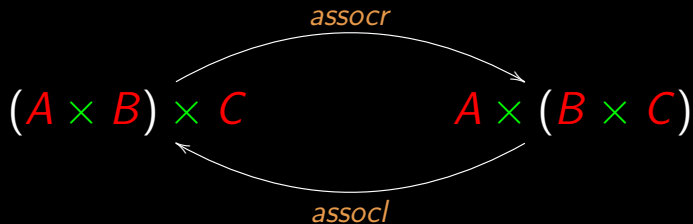
$$\text{assocl} = \langle id \times \pi_1, \pi_2 \cdot \pi_2 \rangle$$

The two "associators"



$$assocr \cdot assocl = id$$

The two "associators"



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

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

Isomorphism

A commutative diagram illustrating the isomorphism between two expressions of the Cartesian product of three sets. The left expression is $(A \times B) \times C$ and the right expression is $A \times (B \times C)$. A curved arrow labeled $assocr$ points from the left expression to the right expression. A curved arrow labeled $assocl$ points from the right expression back to the left expression. In the center, between the two expressions, is the symbol $\cong$ (isomorphism).

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

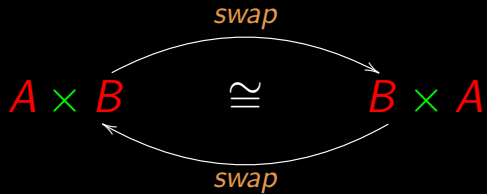
assocr

assocl

$$assocr \cdot assocl = id$$

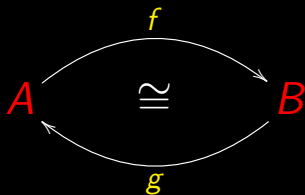
$$assocl \cdot assocr = id$$

Isomorphism



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

Isomorphism



$$f \cdot g = id$$

$$g \cdot f = id$$

Isomorphism

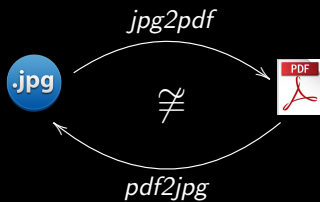
iso (*lso*)
the same

Isomorphism

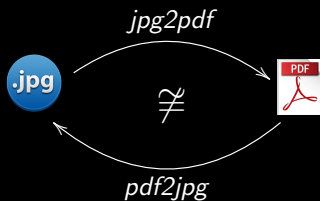
$\underbrace{\text{iso}}_{\text{the same}} (\iota\sigma\omicron) + \underbrace{\text{morphism}}_{\text{shape}} (\mu\omicron\rho\phi\iota\sigma\mu\omicron\zeta)$

“Similar shape”

Practical problem!



Practical problem!



$$jpg2pdf \cdot pdf2jpg \neq id$$

$$pdf2jpg \cdot jpg2pdf \neq id$$

(But our eyes don't spot the difference, in general)

Cálculo de Programas

Class T03 — 29-Sep/2-Oct

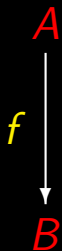
Format conversion

Need



Format conversion

Need



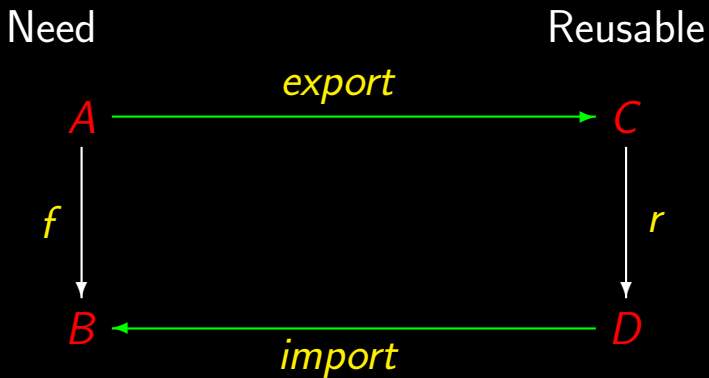
Reusable



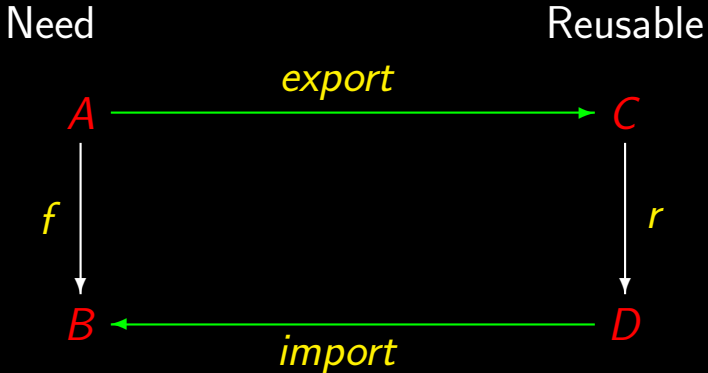
Format conversion



Format conversion

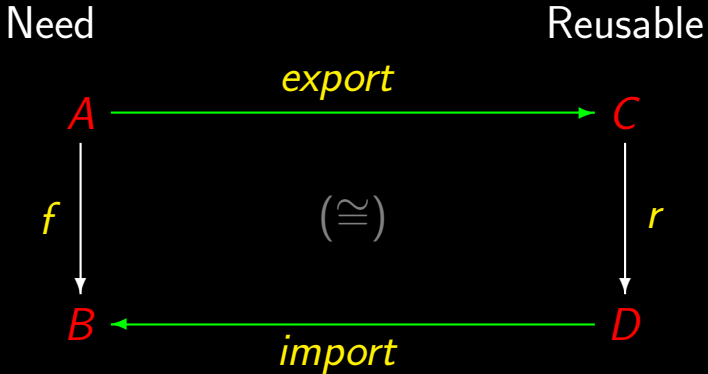


Format conversion



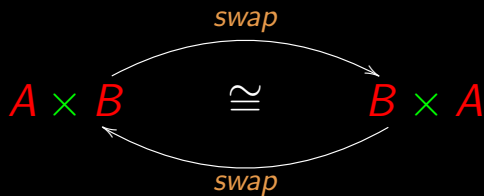
$$f = import \cdot r \cdot export$$

Format conversion



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

By the way — *swap*

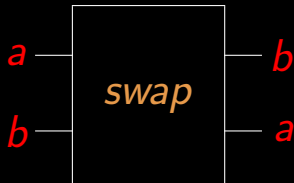


Isomorphisms are **reversible** computations

By the way — *swap*

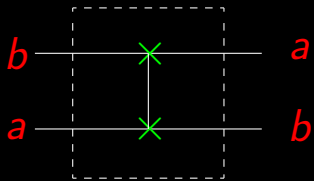


By the way — *swap*

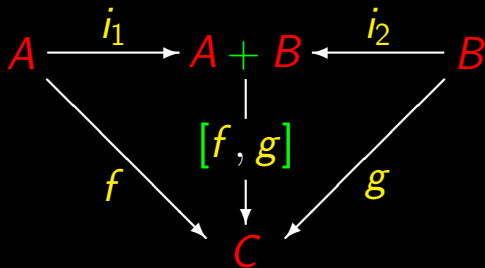


swap is a basic gate in **quantum programming**.

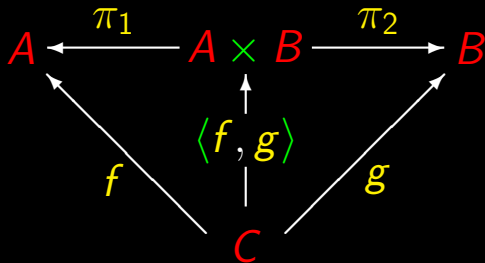
By the way — *swap*



Calculus of alternative composition?



Compare with...



+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

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

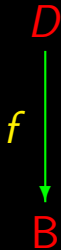
From $[f, g]$ to $f + g$

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

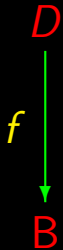
$$[f, g] x = \begin{cases} x = i_1 a \Rightarrow f a \\ x = i_2 b \Rightarrow g b \end{cases}$$

$$f + g \quad ?$$

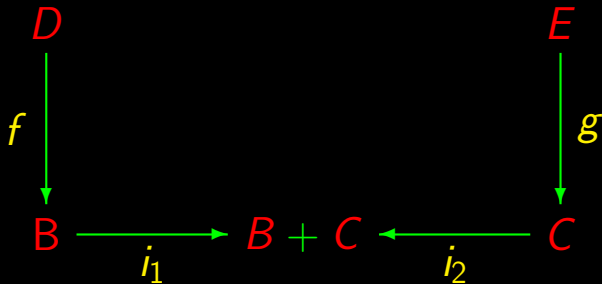
Sum of two functions



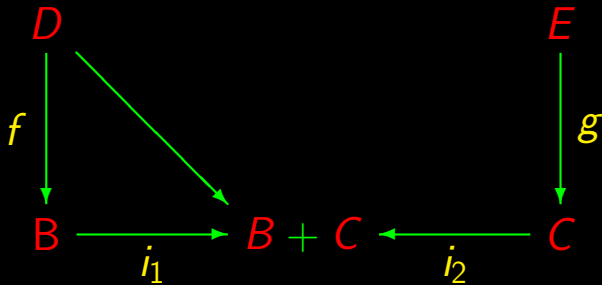
Sum of two functions



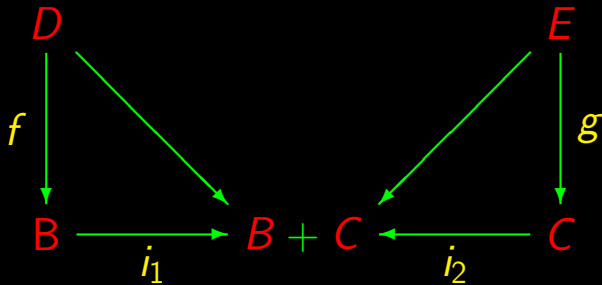
Sum of two functions



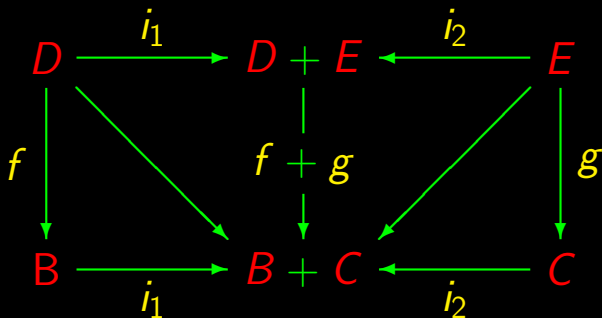
Sum of two functions



Sum of two functions



Sum of two functions



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

Coproduct laws

“Just reverse the arrows”, cf.

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

Coproduct laws

“Just reverse the arrows”, cf.

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

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

Coproduct laws

“Just reverse the arrows”, cf.

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

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

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

Coproduct laws

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

Coproduct laws

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

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

Coproduct laws

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

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

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

and so on

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

(WWW → **Material**)

Summary

$$f \cdot g$$

$$\langle f, g \rangle$$

$$f \times g$$

Sequential composition

Parallel composition

Product composition

Summary

$f \cdot g$

Sequential composition

$\langle f, g \rangle$

Parallel composition

$f \times g$

Product composition

$[f, g]$

Alternative composition

$f + g$

Structural alternation (coproduct)

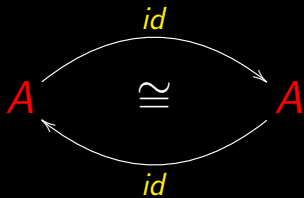
Compositional programming



On information loss

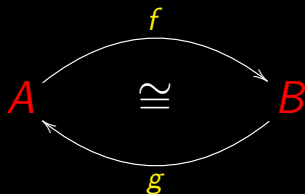
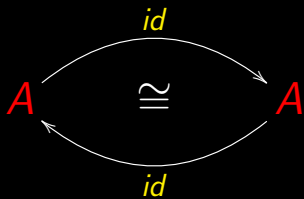
Recall

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



Recall

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



$$f \cdot g = id$$

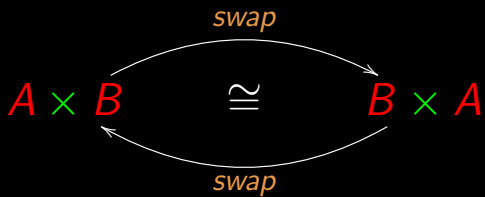
$$g \cdot f = id$$

Recall

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

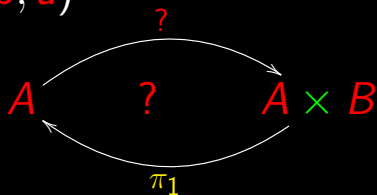
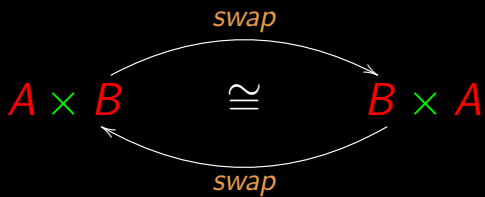
Recall

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



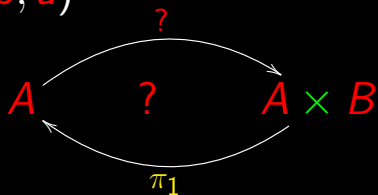
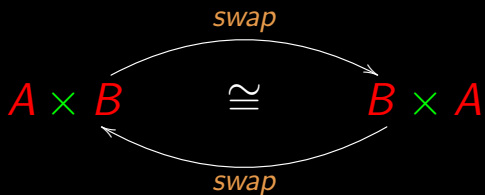
Recall

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



Recall

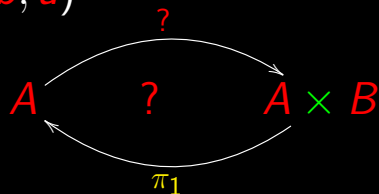
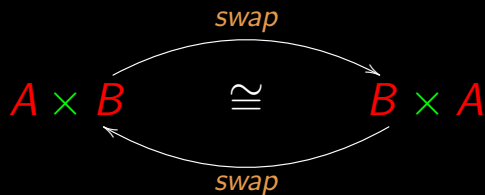
$$\begin{cases} \text{swap} : A \times B \rightarrow B \times A \\ \text{swap}(a, b) = (b, a) \end{cases}$$



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

Recall

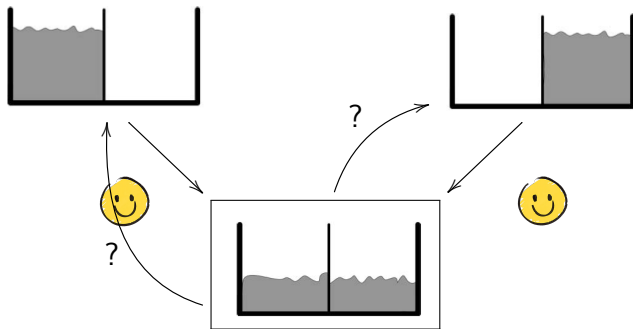
$$\begin{cases} \text{swap} : A \times B \rightarrow B \times A \\ \text{swap}(a, b) = (b, a) \end{cases}$$



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

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

Irreversibility...



Rolf Landauer (1927–99)



Even worse than $\pi_1 \dots$

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

Even worse than $\pi_1 \dots$

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

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

Even worse than $\pi_1 \dots$

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

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

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

Even worse than $\pi_1 \dots$

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

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

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

$$\text{zero } [50 \dots 1000] = 0$$

Even worse than $\pi_1 \dots$

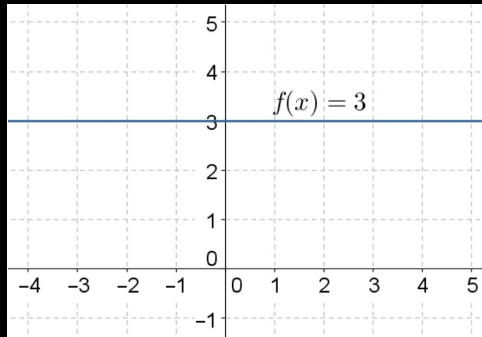
$$\begin{cases} \text{zero } x = 0 \\ \text{one } x = 1 \end{cases}$$

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

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

$$\text{zero } [50 \dots 1000] = 0$$

$$f \ x = 3$$



Constant functions

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

$$f\ x = 3$$

Constant functions

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

$$f\ x = 3$$

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

Constant functions

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

$$b_0 \in B$$

$$f\ x = 3$$

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

Constant functions

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

$$b_0 \in B$$

$$f \ x = 3$$

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

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

Constant functions

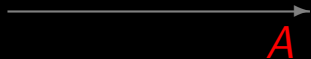
$$3 \in \mathbb{N}_0$$

$$b_0 \in B$$

$$f \ x = 3$$

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

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



Constant functions

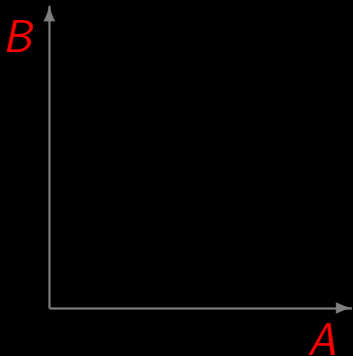
$$3 \in \mathbb{N}_0$$

$$f \ x = 3$$

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

$$b_0 \in B$$

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



Constant functions

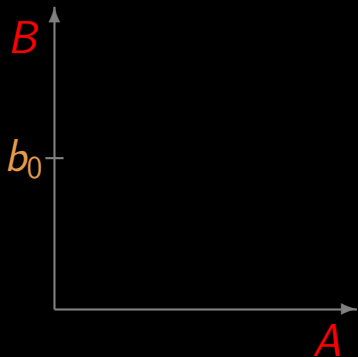
$$3 \in \mathbb{N}_0$$

$$f \ x = 3$$

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

$$b_0 \in B$$

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



Constant functions

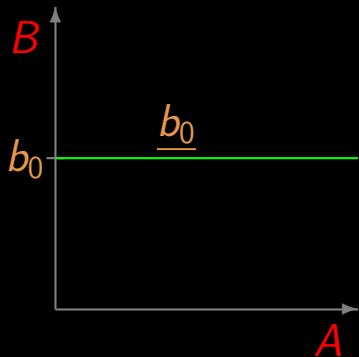
$$3 \in \mathbb{N}_0$$

$$f \ x = 3$$

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

$$b_0 \in B$$

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



Constant functions

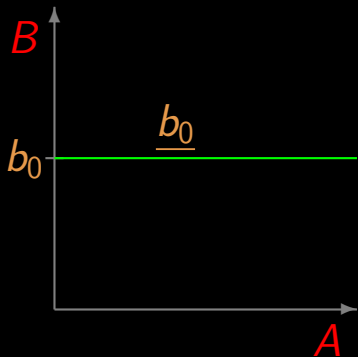
$$3 \in \mathbb{N}_0$$

$$f \ x = 3$$

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

$$b_0 \in B$$

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



(Haskell: `const b0`)

Properties

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

Properties

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

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

Properties

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

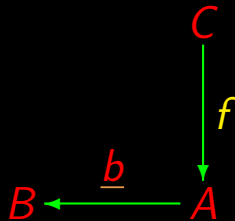
$$g \cdot \underline{b} = \underline{g \ b}$$

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

Properties

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

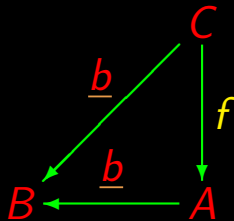
$$g \cdot \underline{b} = \underline{g \ b}$$



Properties

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

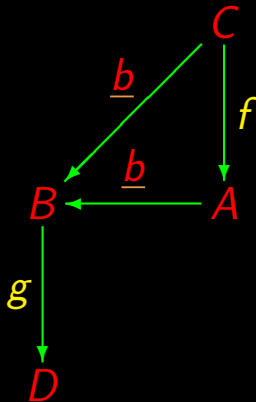
$$g \cdot \underline{b} = \underline{g \ b}$$



Properties

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

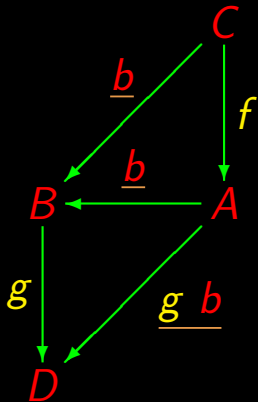
$$g \cdot \underline{b} = \underline{g \ b}$$



Properties

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

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



Back to **no loss** of information

Recall...

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

What about...

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

Recall...

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

What about...

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

or ...

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

Recall...

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

What about...

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

or ...

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

Moreover, any “0” such that

Recall...

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

What about...

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

or ...

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

Moreover, any “0” such that

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

Isomorphisms shine

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



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



Isomorphisms shine

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



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



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

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

Isomorphisms

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

Isomorphisms

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

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

Isomorphisms

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

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

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

Isomorphisms

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

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

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

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

Isomorphisms

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

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

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

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

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

Isomorphisms

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

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

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

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

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

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

Distributivity

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

Distributivity

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

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

Distributivity

A diagram illustrating the distributivity property. It features two mathematical expressions: $A \times (B + C)$ on the left and $A \times B + A \times C$ on the right. The variables A , B , and C are color-coded: A is red, B is green, and C is red. Between the two expressions is a double-lined equivalence symbol $\cong$. Two curved arrows connect the expressions: a top arrow pointing from left to right labeled *distr*, and a bottom arrow pointing from right to left labeled *undistr*.

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

distr

undistr

Distributivity

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

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

Distributivity

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

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

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

Distributivity

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

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

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

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

Distributivity

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

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

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

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

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

Distributivity

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

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

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

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

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

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

Distributivity

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

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

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

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

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

Distributivity

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

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

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

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

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

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

Distributivity

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

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

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

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

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

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

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

Distributivity

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

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

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

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

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

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

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

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

Distributivity

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

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

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

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

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

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

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

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

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

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

Distributivity

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

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

Distributivity

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

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

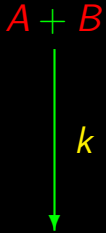
By definition of $f \times g$

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

The exchange law

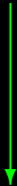


The exchange law



The exchange law

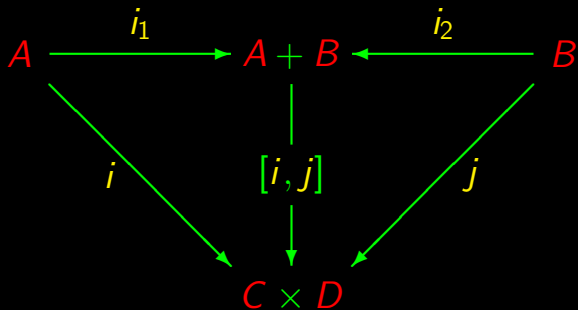
$A + B$



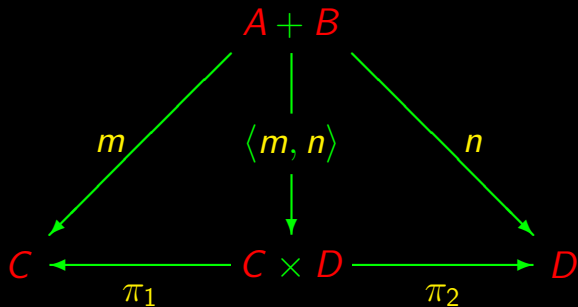
k

$C \times D$

Exchange law



Exchange law

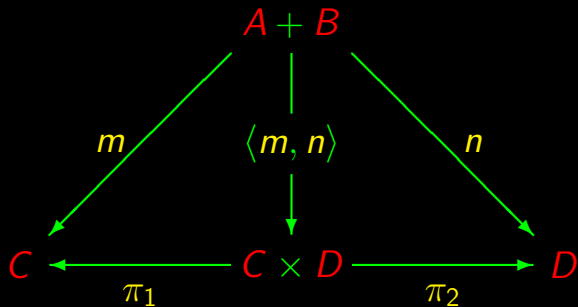


Exchange law

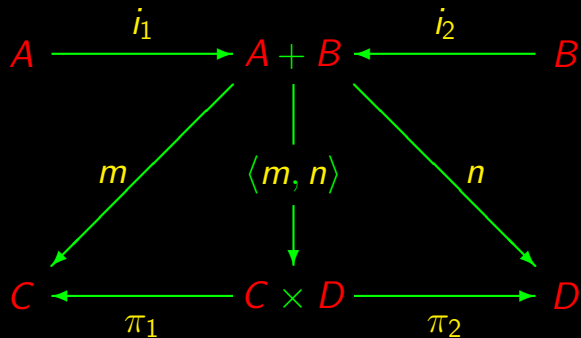
$$\langle m, n \rangle = [i, j]$$

One equation, four unknowns

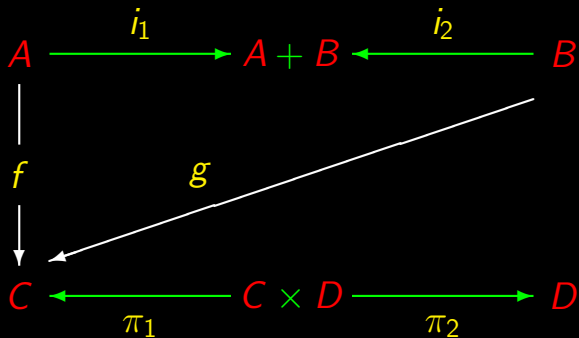
Exchange law



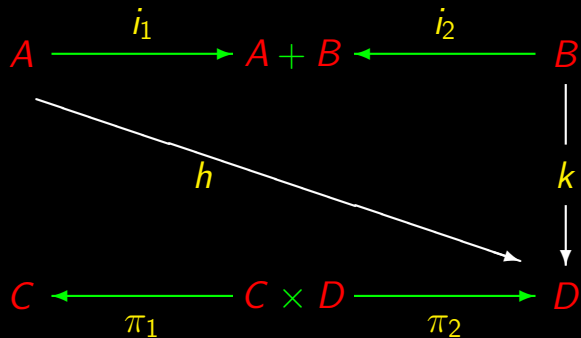
Exchange law



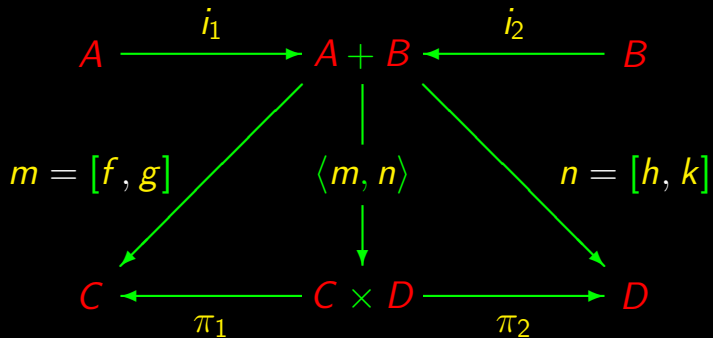
Exchange law



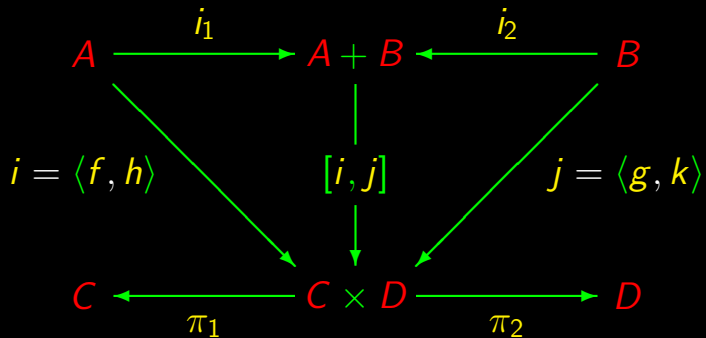
Exchange law



Exchange law



Exchange law



Summing up

Exchange law:

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

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

... and vice-versa

Exchange law — Example

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

A commutative diagram illustrating the distributive law. It consists of two expressions connected by two curved arrows forming a loop. The left expression is $A \times (B + C)$ and the right expression is $A \times B + A \times C$. Between them is a vertical symbol $\cong$. The top arrow points from left to right and is labeled distr . The bottom arrow points from right to left and is labeled undistr .

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

distr

undistr

Exchange law — Example

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

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

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

The types 0, 1 and 2...

The type 1

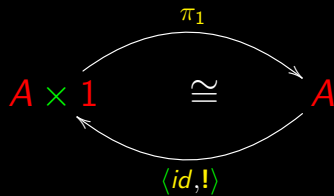
In Haskell:

$() :: ()$

Then

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

Isomorphism



The type 1

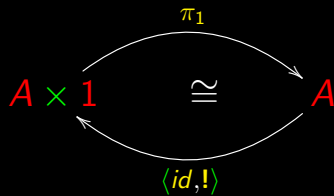
In Haskell:

`() :: ()`

Then

`1 = { () }`

Isomorphism



... "!" ?

The type 1

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

The type 1

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

"Bang":

$$A \xrightarrow{!} 1$$

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

The type 1

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

"Bang":

$$A \xrightarrow{!} 1$$

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

"Points":

$$1 \xrightarrow{a} A$$

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

(provided $a \in A$)

The type 0

$$0 = \{\}$$

The type 0

$$0 = \{\}$$

Immediate in e.g. Haskell:

```
data Zero
```

The type 0

$$0 = \{\}$$

Immediate in e.g. Haskell:

```
data Zero
```

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

The type 0

$$0 = \{\}$$

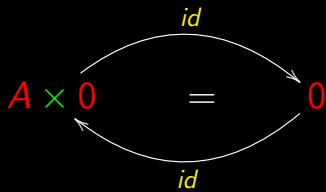
Immediate in e.g. Haskell:

```
data Zero
```

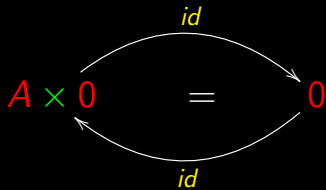
0 is not inhabited: $\neg (a \in 0)$
for all a

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

The type 0



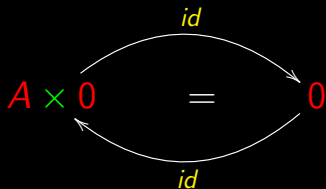
The type 0



In fact

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

The type 0

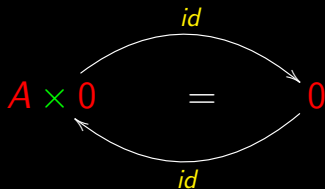


In fact

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

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

The type 0



In fact

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

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

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

The type $1 + A$

$$1 + A$$

The type $1 + A$

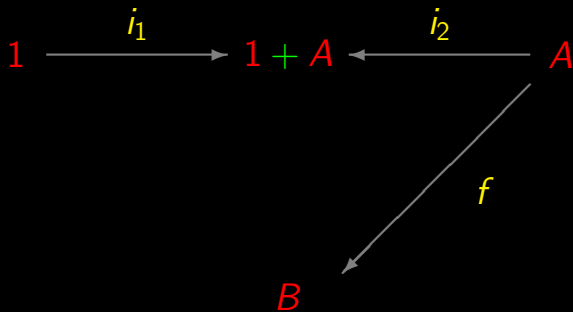
$$1 \xrightarrow{i_1} 1 + A \xleftarrow{i_2} A$$

The type $1 + A$

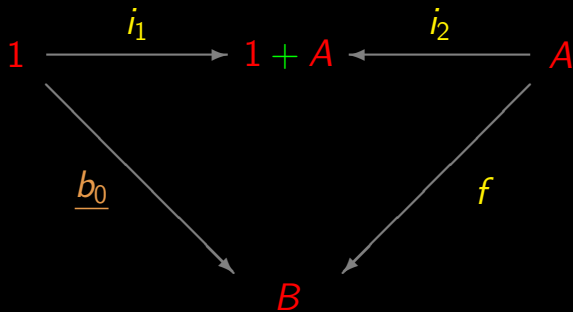
$$1 \xrightarrow{i_1} 1 + A \xleftarrow{i_2} A$$

B

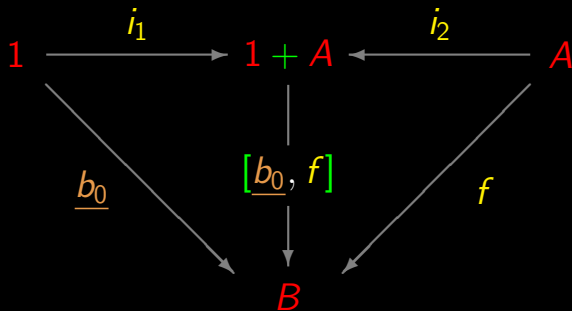
The type $1 + A$



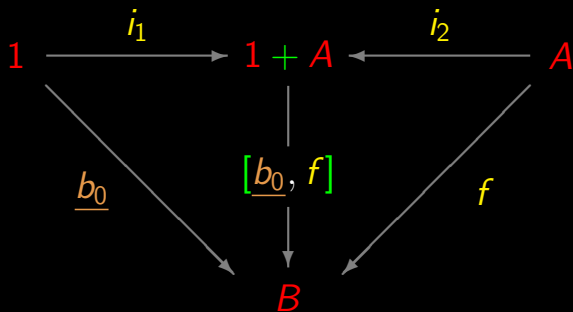
The type $1 + A$



The type $1 + A$



The type $1 + A$



$1 + A$

"pointer to A "

More isomorphisms!

More isomorphisms!

Back to

*Retrieve the name 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).

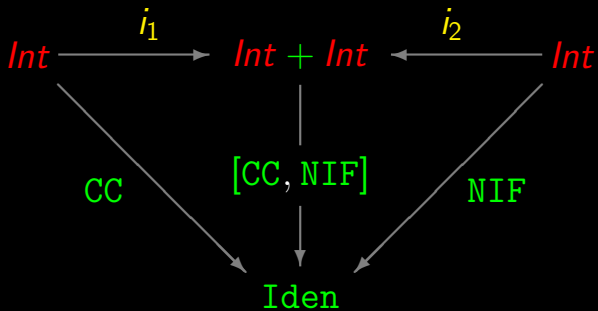
In Haskell (Portugal):

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

In Haskell (Germany):

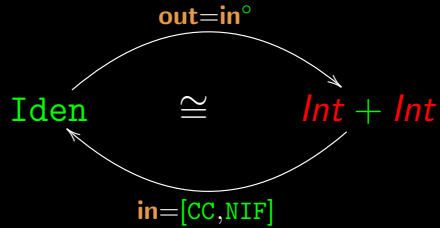
```
data Iden = IDK Int | SZN Int
```

More isomorphisms!

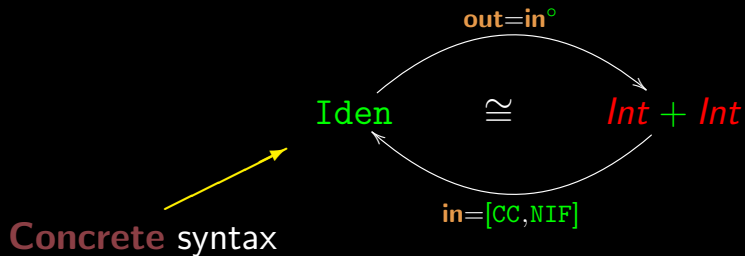


```
data Iden = CC Int | NIF Int
```

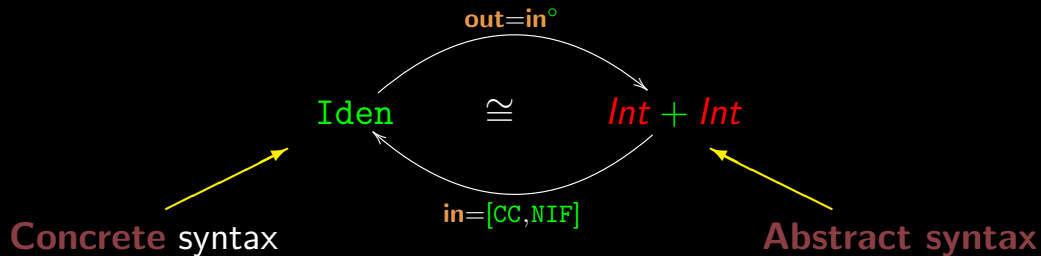
More isomorphisms!



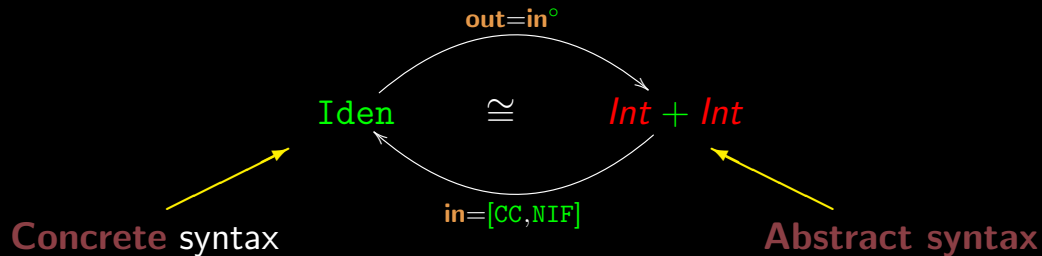
More isomorphisms!



More isomorphisms!

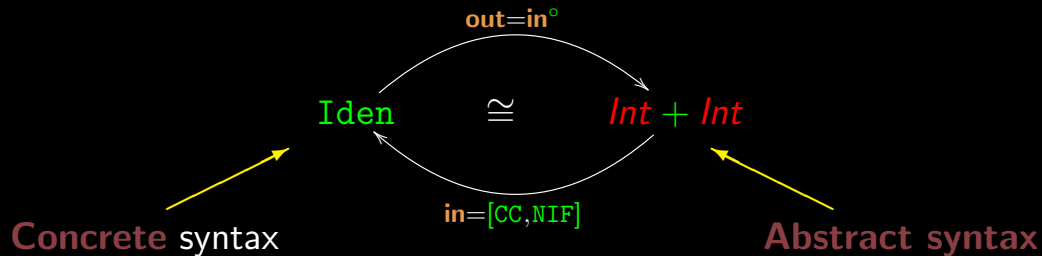


More isomorphisms!



in ("get in")

More isomorphisms!



in ("get in")

out ("get out")

Cálculo de Programas

Class T04 — 6/9-Oct