Cálculo de Programas

Class T01

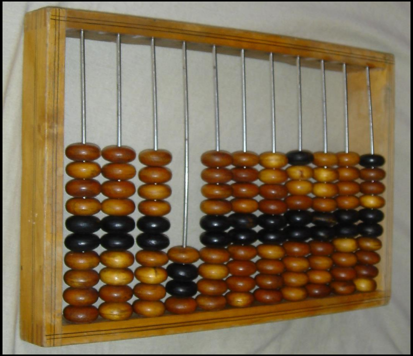
Introduction to the course. **Website** and other administrative details.

J.N. Oliveira

Programming
by
Calculation

Program versus Calculus

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

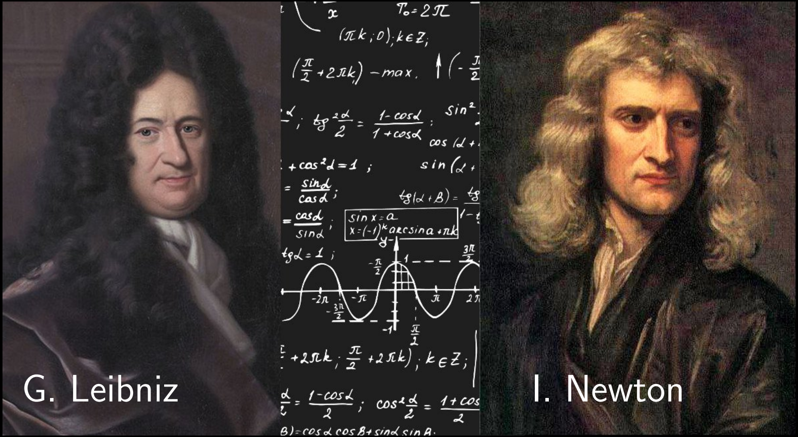


Programs...

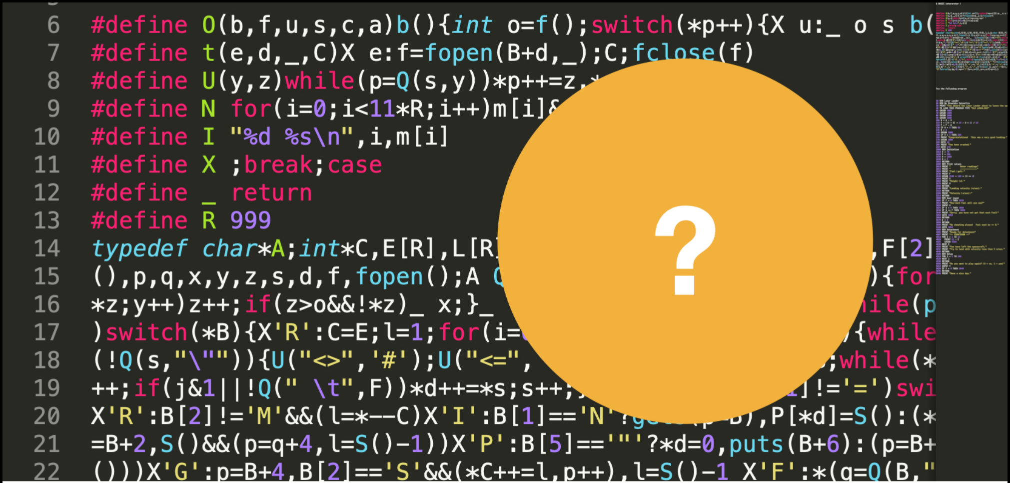
wc.c

```c
#define YES 1
#define NO 0
main()
{
int c, nl, nw, nc, inword ;
inword = NO ;
nl = 0;
nw = 0;
nc = 0;
c = getchar();
while ( c != EOF ) {
     nc = nc + 1;
     if ( c == '\n')
     nl = nl + 1;
     if ( c == ' ' || c == '\n' || c == '\t')
           inword = NO;
     else if (inword == NO) {
           inword = YES ;
           nw = nw + 1;
     }
     c = getchar();
}
printf("%d",nl);
printf("%d",nw);
printf("%d",nc);
}
```

Calculus



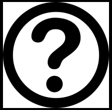
```
#define O(b,f,u,s,c,a)b(){int o=f();switch(*p++){X u:_ o s b(
#define t(e,d,_,C)X e:f=fopen(B+d,_);C;fclose(f)
#define U(y,z)while(p=Q(s,y))*p++=z,*p=' '
#define N for(i=0;i<11*R;i++)m[i]&&
#define I "%d %s\n",i,m[i]
#define X ;break;case
#define _ return
#define R 999
typedef char*A;int*C,E[R],L[R],M[R],P[R],l,i,j;char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


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

Software as a problem

Software

Process — 

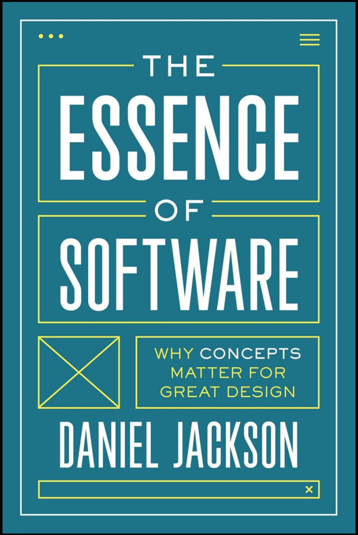
Product — 

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

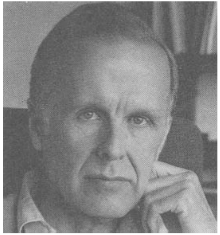
... small number
... well defined
... easy to understand

Urgently needed in software design

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

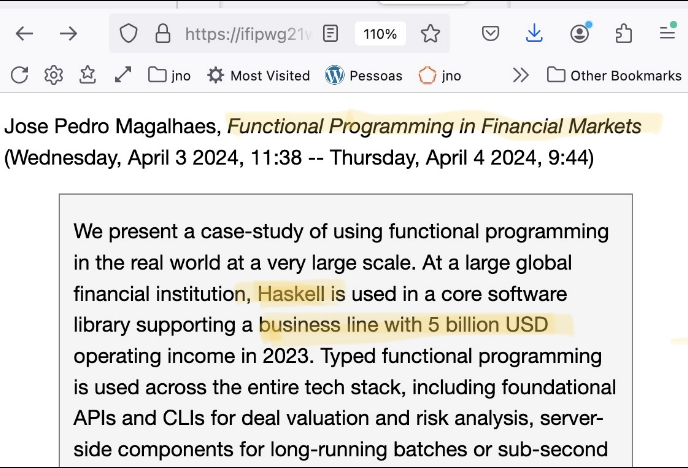


Conventional programming languages are growing ever more enormous, but not stronger. Inherent defects at the most basic level cause them to be both fat and weak: their primitive word-at-a-time style of programming inherited from their common ancestor—the von Neumann computer, their close coupling of 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

Functional Programming

FP = $$$...



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

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]
```

$$\texttt{store c l = }\underbrace{\texttt{take 10}}_{(c)}\ \texttt{(}\underbrace{\texttt{nub}}_{(b)}\ \underbrace{\texttt{(c:l)}}_{(a)}\texttt{)}$$

Compare with ...

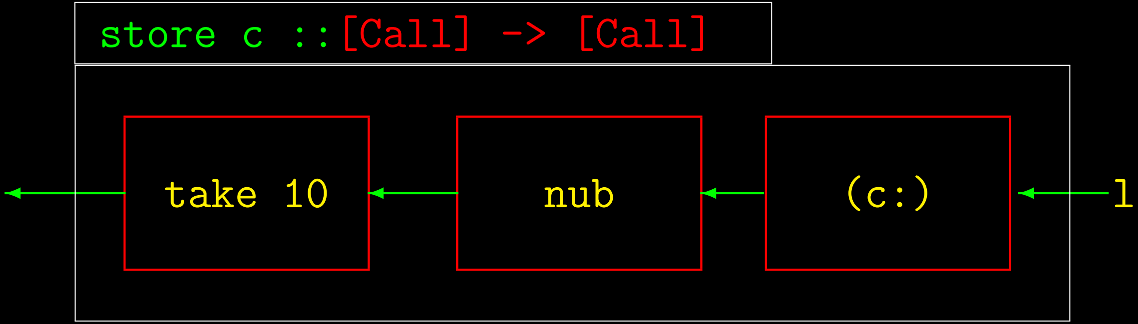
```
public void store10(string phoneNumber)
{
  System.Collections.ArrayList auxList =
      new System.Collections.ArrayList();
  auxList.Add(phoneNumber);
  auxList.AddRange(
      this.filteratmost9(phoneNumber) );
  this.callList = auxList;
}
```

+ `filteratmost9` (next slide)

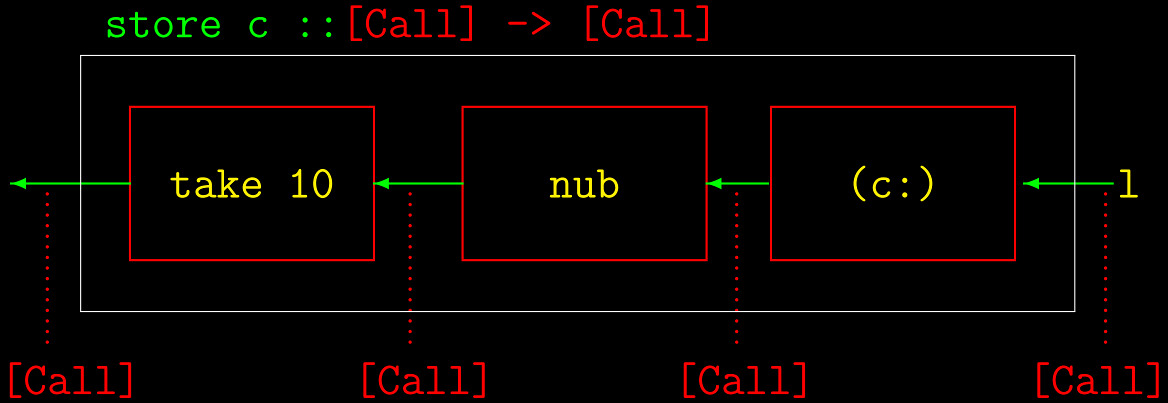
Compare with ...

```
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;
}
```

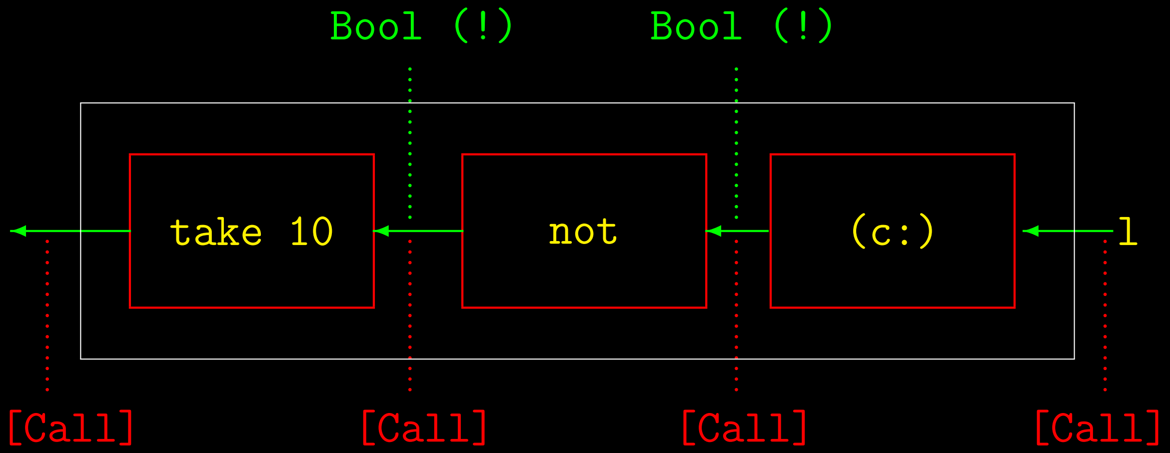
From a mobile phone manufacturer



From a mobile phone manufacturer

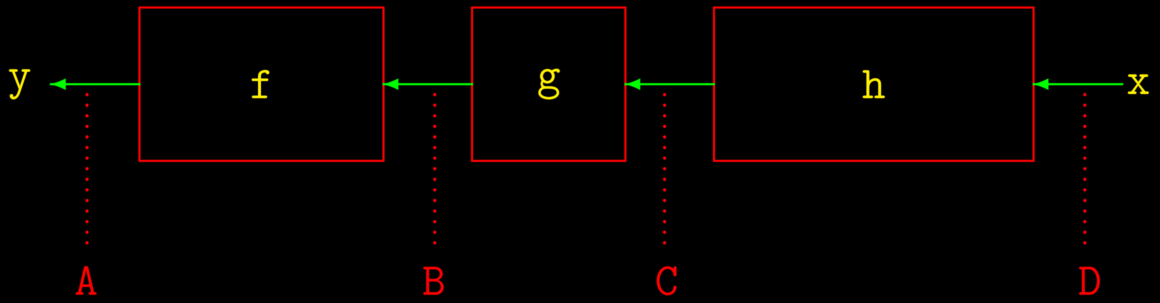


Uups!



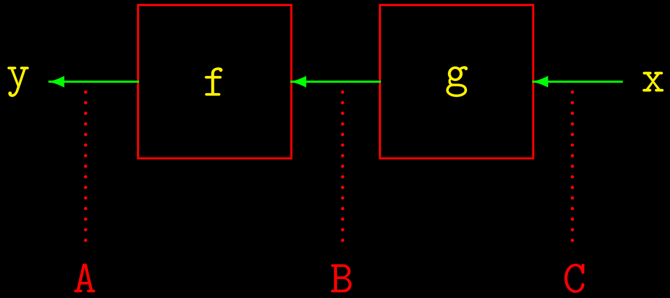
In general

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



In general

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



Simplification

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

$$\begin{tikzcd} A & B \arrow[l, "f"'] & C \arrow[l, "g"'] \end{tikzcd}$$

Composition

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

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

$$A \xleftarrow{\qquad f \cdot g \qquad} C$$

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

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

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

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

Cf. Unix/Linux pipes

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

$$g \mid f$$

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

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

Composition

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

i.e.

$$take\ 10 \cdot (nub \cdot (c:))$$

Composition

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

i.e.

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

the same as

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

Composition

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

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

Composition

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

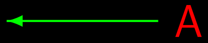
$$a + 0 = 0 + a = a$$

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


A

$$A \longleftarrow A$$

$$A \xleftarrow{id} A$$

$$A \xleftarrow{id} A$$

$$id\ a = a$$

Identity

$$\begin{tikzcd} A & A \arrow[l, "id"'] \\ B \arrow[from=1-1, "f"'] & \end{tikzcd}$$

Identity

$$
\begin{tikzcd}
A \arrow[d, "f"'] & A \arrow[l, "id"'] \arrow[d, "f"] \\
B & B
\end{tikzcd}
$$

Identity

$$
\begin{tikzcd}
A \arrow[d, "f"'] & A \arrow[l, "id"'] \arrow[d, "f"] \\
B & B \arrow[l, "id"]
\end{tikzcd}
$$

Identity

$$
\begin{tikzcd}
A \arrow[d, "f"'] & A \arrow[l, "id"'] \arrow[d, "f"] \arrow[dl, "f"'] \\
B & B \arrow[l, "id"]
\end{tikzcd}
$$

Identity

$$
\begin{tikzcd}
A \arrow[d, "f"'] & A \arrow[l, "id"'] \arrow[d, "f"] \arrow[dl, "f"'] \\
B & B \arrow[l, "id"]
\end{tikzcd}
$$

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

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*" :

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

$$f \cdot g$$

$$f \cdot g$$

$$f \times g\ ?$$

$$f \cdot g$$

$$f \times g \;?$$

$$f + g \;?$$

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

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

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

What about

$$\begin{matrix} D \xrightarrow{f} B \\ A \xrightarrow{g} C \end{matrix}$$

?

Try $D = A$ ...

$$
\begin{tikzcd}
B & & C \\
& A \arrow[ul, "f"] \arrow[ur, "g"'] &
\end{tikzcd}
$$

Try $D = A$ ...

$$\begin{tikzcd} B & & C \\ & A \arrow[ul, "f"] \arrow[ur, "g"'] & \end{tikzcd}$$

$f\ a$ ... $g\ a$

$$
\begin{tikzcd}
B & & C \\
& A \arrow[ul, "f"] \arrow[ur, "g"'] &
\end{tikzcd}
$$

$$(f\ a, g\ a)$$

Cartesian product

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

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

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

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

“Split”

$$\begin{tikzcd}
B & B \times C & C \\
& A \arrow[ul, "f"] \arrow[u, "{\langle f, g \rangle}"'] \arrow[ur, "g"'] &
\end{tikzcd}$$

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

Product

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

Product

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

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

Product

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

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

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

Product

$$\begin{tikzcd} B & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \\ & A \arrow[ul, "f"] \arrow[u, "\langle f, g \rangle"'] \arrow[ur, "g"'] & \end{tikzcd}$$

Product

$$
\begin{tikzcd}
B & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \\
 & A \arrow[ul, "f"] \arrow[u, "\langle f, g \rangle"'] \arrow[ur, "g"'] &
\end{tikzcd}
$$

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

Product

$$
\begin{tikzcd}
B & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \\
& A \arrow[ul, "f"] \arrow[u, "\langle f, g \rangle"'] \arrow[ur, "g"'] &
\end{tikzcd}
$$

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

Product

$$\langle f, g \rangle$$

Product

$$\langle f, g \rangle$$

f and g in **parallel**

f “split” g

Product

$$\langle f, g \rangle$$

f and g in **parallel**

f “split” g

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

Cálculo de Programas

Class T02

Product

$$\begin{tikzcd} D \\ B \arrow[u, "f"] \end{tikzcd}$$

Product

$$\begin{tikzcd} D & & & E \\ B \arrow[u, "f"] & & & C \arrow[u, "g"'] \end{tikzcd}$$

Product

$$
\begin{tikzcd}
D & & E \\
B \arrow[u, "f"] & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \arrow[u, "g"']
\end{tikzcd}
$$

Product

$$\begin{tikzcd}
D & & E \\
B \arrow[u, "f"] & B \times C \arrow[l, "\pi_1"'] \arrow[ul, "f \cdot \pi_1"'] \arrow[r, "\pi_2"'] & C \arrow[u, "g"']
\end{tikzcd}$$

Product

$$
\begin{tikzcd}
D & & E \\
B \arrow[u, "f"] & B \times C \arrow[l, "\pi_1"'] \arrow[ul, "f \cdot \pi_1"'] \arrow[r, "\pi_2"] \arrow[ur, "g \cdot \pi_2"] & C \arrow[u, "g"']
\end{tikzcd}
$$

Product

$$
\begin{tikzcd}
D & D \times E & E \\
B \arrow[u, "f"] & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[ul] \arrow[ur] \arrow[u, "\langle f \cdot \pi_1, g \cdot \pi_2 \rangle" description] & C \arrow[u, "g"']
\end{tikzcd}
$$

Product

$$
\begin{tikzcd}
D & D \times E \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & E \\
B \arrow[u, "f"] & B \times C \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] \arrow[u, "\langle f \cdot \pi_1, g \cdot \pi_2 \rangle" description] \arrow[ul] \arrow[ur] & C \arrow[u, "g"']
\end{tikzcd}
$$

Product

$$
\begin{tikzcd}
D & D \times E \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & E \\
B \arrow[u, "f"] & B \times C \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] \arrow[u, "f \times g" description] \arrow[ul] \arrow[ur] & C \arrow[u, "g"']
\end{tikzcd}
$$

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

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)

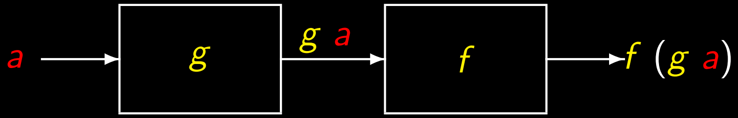
Summing up

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

Compositional programming

In pictures

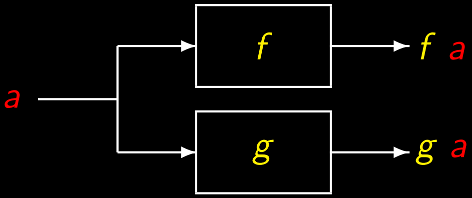
$$(f \cdot g)\ a = f\ (g\ a) \tag{2.6}$$



Function **composition**

In pictures

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



Functional "splits"

In pictures

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

$$\begin{tikzcd} c \arrow[r] & \boxed{f} \arrow[r] & f\ c \\ d \arrow[r] & \boxed{g} \arrow[r] & g\ d \end{tikzcd}$$

Functional **products**

$f \cdot g$ Sequential composition

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

$f \times g$ Parallel composition (asynchronous)

Compositional programming

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).

$$address : Iden \to Address$$

$$Iden = CC \cup NIF$$

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!

$$CC = \mathbb{N}$$

$$NIF = \mathbb{N}$$

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

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

In general

We need to fix:

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

In general

We need to fix:

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

Let us start from

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

Disjoint union

In need of something like...

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

Disjoint union

In need of something like...

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

... we define:

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

Disjoint union

Clearly,

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

once we define:

$$i_1\ a = (1, a)$$
$$i_2\ b = (2, b)$$

Disjoint union

Types:

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

Disjoint union

Types:

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

Now think of:

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

Disjoint union

Types:

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

Now think of:

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

$$\begin{tikzcd} A + B \arrow[d, "m"] \\ C \end{tikzcd}$$

Disjoint union

Types:

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

Now think of:

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

$$\begin{tikzcd} A \arrow[r, "i_1"] & A + B \arrow[d, "m"] \\ & C \end{tikzcd}$$

Disjoint union

Types:

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

Now think of:

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

$$\begin{tikzcd} A \arrow[r, "i_1"] & A + B \arrow[d, "m"] & B \arrow[l, "i_2"'] \\ & C & \end{tikzcd}$$

Disjoint union

Types:

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

Now think of:

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

$$\begin{tikzcd} A \arrow[r, "i_1"] \arrow[dr, "f"'] & A + B \arrow[d, "m"] & B \arrow[l, "i_2"'] \\ & C & \end{tikzcd}$$

$$f = m \cdot i_1$$

Disjoint union

Types:

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

Now think of:

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

$$\begin{tikzcd}
A \arrow[r, "i_1"] \arrow[dr, "f"'] & A + B \arrow[d, "m"] & B \arrow[l, "i_2"'] \arrow[dl, "g"] \\
& C &
\end{tikzcd}$$

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

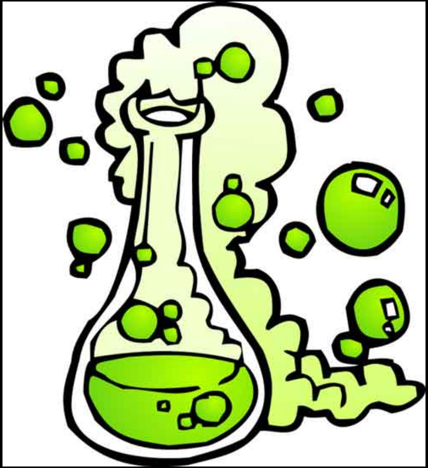
Compare...

$$\begin{tikzcd} A \arrow[r, "i_1"] \arrow[dr, "f"'] & A + B \arrow[d, "{[f,g]}"] & B \arrow[l, "i_2"'] \arrow[dl, "g"] \\ & C & \end{tikzcd}$$

Compare...

$$\begin{tikzcd}
A & A \times B \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & B \\
& C \arrow[ul, "f"] \arrow[u, "\langle f, g \rangle"] \arrow[ur, "g"'] &
\end{tikzcd}$$

Calculus?



$$
\begin{tikzcd}
B & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \\
& A \arrow[ul, "f"] \arrow[u, "\langle f , g \rangle"] \arrow[ur, "g"'] &
\end{tikzcd}
$$

$$
\begin{tikzcd}
B & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \\
& A \arrow[ul, "f"] \arrow[u, "\langle f, g \rangle"] \arrow[ur, "g"'] & \\
& D \arrow[u, "h"] &
\end{tikzcd}
$$

$$
\begin{tikzcd}
B & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \\
& D \arrow[u, "\langle f, g \rangle \cdot h" description] &
\end{tikzcd}
$$

$$
\begin{tikzcd}
B & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \\
& A \arrow[ul, "f"] \arrow[u, "\langle f , g \rangle"] \arrow[ur, "g"'] & \\
& D \arrow[u, "h"] &
\end{tikzcd}
$$

$$
\begin{tikzcd}
B & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \\
& A \arrow[ul, "f"] \arrow[u, "{\langle f, g \rangle}"] \arrow[ur, "g"'] & \\
& D \arrow[u, "h"] \arrow[uul, "f \cdot h"] &
\end{tikzcd}
$$

$$
\begin{tikzcd}
B & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \\
 & A \arrow[ul, "f"] \arrow[u, "\langle f , g \rangle"] \arrow[ur, "g"'] & \\
 & D \arrow[u, "h"] \arrow[uul, "f \cdot h"] \arrow[uur, "g \cdot h"'] &
\end{tikzcd}
$$

$$
\begin{tikzcd}
B & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \\
 & D \arrow[ul, "f \cdot h"] \arrow[u, "\langle f \cdot h, g \cdot h \rangle"] \arrow[ur, "g \cdot h"'] &
\end{tikzcd}
$$

$$\begin{tikzcd}
B & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \\
& D \arrow[u, "\langle f, g \rangle \cdot h = \langle f \cdot h, g \cdot h \rangle" description] &
\end{tikzcd}$$

$\times$-Fusion

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

$$\begin{tikzcd}
B & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \\
 & A \arrow[ul, "f"] \arrow[u, "\langle f, g \rangle"] \arrow[ur, "g"'] & \\
 & D \arrow[u, "h"] \arrow[uul, "f \cdot h"] \arrow[uur, "g \cdot h"'] &
\end{tikzcd}$$

$$
\begin{tikzcd}
B & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \\
& A \arrow[ul, "f"] \arrow[u, "\langle f, g \rangle"] \arrow[ur, "g"'] &
\end{tikzcd}
$$

$$
\begin{tikzcd}
D & & \\
B \arrow[u, "h"] & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \\
& A \arrow[ul, "f"] \arrow[u, "{\langle f, g \rangle}"] \arrow[ur, "g"'] &
\end{tikzcd}
$$

$$
\begin{tikzcd}
D & & E \\
B \arrow[u, "h"] & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \arrow[u, "k"'] \\
& A \arrow[ul, "f"] \arrow[u, "\langle f, g \rangle"] \arrow[ur, "g"'] &
\end{tikzcd}
$$

$$
\begin{tikzcd}
D & D \times E & E \\
B \arrow[u, "h"] & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[u, "h \times k"] & C \arrow[u, "k"'] \\
& A \arrow[ul, "f"] \arrow[u, "\langle f, g \rangle"] \arrow[ur, "g"'] &
\end{tikzcd}
$$

$$
\begin{tikzcd}
D & D \times E \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & E \\
B \arrow[u, "h"] & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[u, "h \times k" description] & C \arrow[u, "k"'] \\
& A \arrow[ul, "f"] \arrow[u, "\langle f , g \rangle" description] \arrow[ur, "g"'] &
\end{tikzcd}
$$

$$
\begin{tikzcd}
D & D \times E \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & E \\
B \arrow[u, "h"] & & C \arrow[u, "k"'] \\
& A \arrow[ul, "f"] \arrow[ur, "g"'] \arrow[uu, "(h \times k) \cdot \langle f, g \rangle" description] &
\end{tikzcd}
$$

$$
\begin{tikzcd}
D & & D \times E \arrow[ll, "\pi_1"'] \arrow[rr, "\pi_2"] & & E \\
B \arrow[u, "h"] & & & & C \arrow[u, "k"'] \\
& & A \arrow[ull, "f"] \arrow[urr, "g"'] \arrow[uull, "h \cdot f"'] & &
\end{tikzcd}
$$

$$
\begin{tikzcd}
D & & D \times E \arrow[ll, "\pi_1"'] \arrow[rr, "\pi_2"] & & E \\
B \arrow[u, "h"] & & & & C \arrow[u, "k"'] \\
& & A \arrow[llu, "f"] \arrow[rru, "g"'] \arrow[lluu, "h \cdot f"'] \arrow[rruu, "k \cdot g"] & &
\end{tikzcd}
$$

$$
\begin{tikzcd}
D & D \times E \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & E \\
B \arrow[u, "h"] & & C \arrow[u, "k"'] \\
& A \arrow[ul, "f"] \arrow[ur, "g"'] \arrow[uu, "\langle h \cdot f, k \cdot g \rangle" description] &
\end{tikzcd}
$$

$$
\begin{tikzcd}
D & D \times E \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & E \\
B \arrow[u, "h"] & & C \arrow[u, "k"'] \\
& A \arrow[ul, "f"] \arrow[ur, "g"'] \arrow[uu, "(h \times k) \cdot \langle f, g \rangle" description] &
\end{tikzcd}
$$

$\times$-Absorption

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

$$\begin{tikzcd}
A & A \times B \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & B \\
D \arrow[u, "h"] & D \times E \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[u, "h \times k"'] & E \arrow[u, "k"'] \\
& C \arrow[ul, "f"] \arrow[u, "\langle f, g \rangle"] \arrow[ur, "g"'] &
\end{tikzcd}$$

$$
\begin{tikzcd}
D & D \times E \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & E \\
B \arrow[u, "h"] & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[u, "h \times k"] & C \arrow[u, "k"'] \\
& A \arrow[ul, "f"] \arrow[u, "\langle f, g \rangle"] \arrow[ur, "g"'] &
\end{tikzcd}
$$

$$
\begin{tikzcd}
D & D \times E \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & E \\
B \arrow[u, "h"] & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[u, "h \times k"] & C \arrow[u, "k"']
\end{tikzcd}
$$

$$
\begin{tikzcd}
D & D \times E \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & E \\
B \arrow[u, "h"] & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[u, "h \times k"] & C \arrow[u, "k"']
\end{tikzcd}
$$

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

$$
\begin{tikzcd}
D & D \times E \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & E \\
B \arrow[u, "h"] & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[u, "h \times k"] & C \arrow[u, "k"']
\end{tikzcd}
$$

$$\pi_1 \cdot (h \times k) = h \cdot \pi_1 \qquad \pi_2 \cdot (h \times k) = k \cdot \pi_2$$

Natural-π_1, natural-π_2

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

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

$$\begin{tikzcd}
D & D \times E \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & E \\
B \arrow[u, "h"] & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[u, "h \times k"'] & C \arrow[u, "k"']
\end{tikzcd}$$

So far...

$f \cdot g$ Sequential composition

$\langle f, g \rangle$ Parallel composition

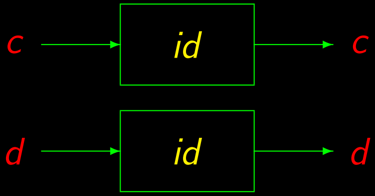
$f \times g$ Product composition

$[f, g]$ Alternative composition

Compositional programming

Functor-id-$\times$

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



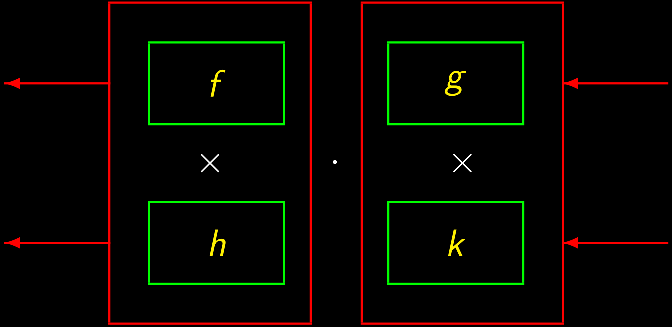
Product of two identities is an identity.

$\times$-Functor

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

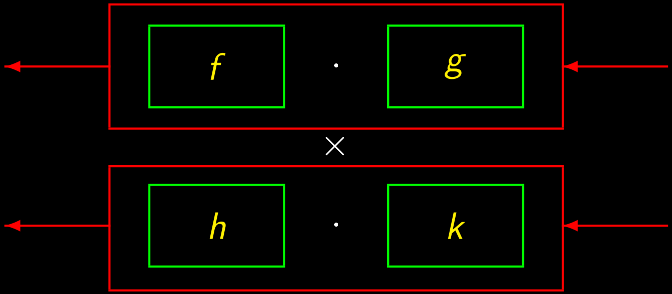
Composition of products is a **product** of compositions.

$\times$-Functor



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

$\times$-Functor



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

Two basic laws still missing

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

Two basic laws still missing

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

$\times$-Eq $$\langle i, j \rangle = \langle f, g \rangle \;\Leftrightarrow\; \left\{ \begin{array}{l} i = f \\ j = g \end{array} \right. \tag{2.64}$$

Before them, the most important...

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

$$
\begin{tikzcd}
B & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \\
& A \arrow[ul, "f"] \arrow[u, "\langle f, g\rangle"'] \arrow[ur, "g"'] &
\end{tikzcd}
$$

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

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

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

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

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

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

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

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

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

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

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

Equations!

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

Equations!

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

Substituting:

$$\mathit{id} = \langle \pi_1, \pi_2 \rangle$$

Equations!

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

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

$$
\begin{aligned}
& \langle h, k \rangle = \langle f, g \rangle \\
\Leftrightarrow \quad & \{ \ \times\text{-universal} \ \} \\
& \begin{cases} \pi_1 \cdot \langle h, k \rangle = f \\ \pi_2 \cdot \langle h, k \rangle = g \end{cases} \\
\Leftrightarrow \quad & \{ \ \times\text{-cancellation} \ \} \\
& \begin{cases} h = f \\ k = g \end{cases}
\end{aligned}
$$

Calculation

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

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

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

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

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

$\times$-Eq !

$$\langle \pi_1, \pi_2 \rangle = \mathit{id}$$

$\langle \pi_1, \pi_2 \rangle = \mathit{id}$

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

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

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

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

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

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

$$\begin{tikzcd}
B & \\
& A \times B \arrow[ul, "\pi_2"] \arrow[u, "\langle \pi_2, \pi_1 \rangle"']
\end{tikzcd}$$

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

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

$$
\begin{tikzcd}
B & & A \\
& A \times B \arrow[ul, "\pi_2"] \arrow[u, "\langle \pi_2, \pi_1 \rangle"] \arrow[ur, "\pi_1"'] &
\end{tikzcd}
$$

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

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

$$\begin{tikzcd}
B & B \times A & A \\
& A \times B \arrow[ul, "\pi_2"] \arrow[u, "\langle \pi_2, \pi_1 \rangle"] \arrow[ur, "\pi_1"'] &
\end{tikzcd}$$

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

$$\begin{tikzcd}
B & B \times A \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & A \\
& A \times B \arrow[ul, "\pi_2"] \arrow[u, "\langle \pi_2, \pi_1 \rangle"] \arrow[ur, "\pi_1"'] &
\end{tikzcd}$$

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

$$\begin{tikzcd} A \times B \arrow[rr, bend left, "\langle \pi_2, \pi_1 \rangle"] & & B \times A \arrow[ll, bend left, "k"] \end{tikzcd}$$

Calculation

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

Calculation

$$
\begin{array}{ll}
 & \langle \pi_2, \pi_1 \rangle \cdot k = \mathit{id} \\
\Leftrightarrow & \qquad \{ \ \times\text{-fusion} \ \} \\
 & \langle \pi_2 \cdot k, \pi_1 \cdot k \rangle = \mathit{id}
\end{array}
$$

Calculation

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

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

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

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

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

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

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

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

Swap

$$
\begin{tikzcd}
A \times B \arrow[r, bend left, "\langle \pi_2, \pi_1 \rangle"] & B \times A \arrow[l, bend left, "\langle \pi_2, \pi_1 \rangle"]
\end{tikzcd}
$$

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

Swap

$$\begin{tikzcd} A \times B \arrow[rr, bend left, "\langle \pi_2, \pi_1 \rangle"] & & B \times A \arrow[ll, bend left, "\langle \pi_2, \pi_1 \rangle"] \end{tikzcd}$$

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

$$swap \cdot swap = id$$

So far

$f \cdot g$ Sequential composition

So far

$f \cdot g$ Sequential composition

$\langle f, g \rangle$ 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...

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

$$\begin{tikzcd}
A & \\
& D \arrow[ul, "f"']
\end{tikzcd}$$

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

$$
\begin{tikzcd}
A & & B \\
& D \arrow[ul, "f"] \arrow[ur, "g"'] &
\end{tikzcd}
$$

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

$$\begin{tikzcd}
A & A \times B \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & B \\
& D \arrow[ul, "f"] \arrow[u, "\langle f, g\rangle" description] \arrow[ur, "g"'] &
\end{tikzcd}$$

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

$$\begin{tikzcd} A \times B & & C \\ & D \arrow[ul, "\langle f, g \rangle"'] \arrow[ur, "h"] & \end{tikzcd}$$

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

$$
\begin{tikzcd}
A \times B & {} & C \\
& D \arrow[ul, "{\langle f, g\rangle}"'] \arrow[u, "{\langle\langle f, g\rangle, h\rangle}" description] \arrow[ur, "h"'] &
\end{tikzcd}
$$

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

$$\begin{tikzcd}
A \times B & (A \times B) \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \\
& D \arrow[ul, "\langle f, g\rangle"] \arrow[u, "\langle\langle f, g\rangle, h\rangle" description] \arrow[ur, "h"'] &
\end{tikzcd}$$

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

$$\begin{tikzcd} B & & C \\ & D \arrow[ul, "g"] \arrow[ur, "h"'] & \end{tikzcd}$$

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

$$
\begin{tikzcd}
B & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \\
& D \arrow[ul, "g"] \arrow[u, "{\langle g, h \rangle}"] \arrow[ur, "h"'] &
\end{tikzcd}
$$

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

$$\begin{tikzcd} A & & B \times C \\ & D \arrow[ul, "f"] \arrow[ur, "\langle g, h \rangle"'] & \end{tikzcd}$$

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

$$\begin{tikzcd} A & & B \times C \\ & D \arrow[ul, "f"] \arrow[ur, "\langle g, h\rangle"'] \arrow[u, "{\langle f, \langle g, h \rangle\rangle}"] & \end{tikzcd}$$

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

$$\begin{tikzcd}
A & A \times (B \times C) \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & B \times C \\
& D \arrow[ul, "f"] \arrow[u, "{\langle f, \langle g, h \rangle \rangle}"] \arrow[ur, "{\langle g, h \rangle}"'] &
\end{tikzcd}$$

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

$$
\begin{tikzcd}
A \times B & (A \times B) \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \\
& D \arrow[ul, "\langle f, g\rangle"] \arrow[u, "\langle\langle f, g\rangle, h\rangle" description] \arrow[ur, "h"'] &
\end{tikzcd}
$$

$$
\begin{tikzcd}
(A \times B) \times C \arrow[rr, bend left, "k"] & & A \times (B \times C) \\
& D \arrow[ul, "\langle \langle f, g \rangle, h \rangle"] \arrow[ur, "\langle f, \langle g, h \rangle \rangle"'] &
\end{tikzcd}
$$

$$
\begin{tikzcd}
(A \times B) \times C \arrow[rr, bend left, "k"] & & A \times (B \times C) \\
& D \arrow[ul, "\langle\langle f, g\rangle, h\rangle"] \arrow[ur, "\langle f, \langle g, h\rangle\rangle"'] &
\end{tikzcd}
$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Slight improvement...

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

Slight improvement...

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

$$\Leftrightarrow \qquad \{ \; \pi_2 = id \cdot \pi_2 \; \}$$

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

Slight improvement...

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

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 $k = y + 2$.

$$
\begin{tikzcd}
(A \times B) \times C \arrow[rr, bend left, "\langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle"] & & A \times (B \times C) \\
& D \arrow[ul, "\langle \langle f, g \rangle, h \rangle"] \arrow[ur, "\langle f, \langle g, h \rangle \rangle"'] &
\end{tikzcd}
$$

$$\begin{tikzcd}
(A \times B) \times C \arrow[rr, bend left, "\langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle"] & & A \times (B \times C) \arrow[ll, bend left, "j"] \\
& D \arrow[ul, "\langle \langle f, g \rangle, h \rangle"] \arrow[ur, "\langle f, \langle g, h \rangle \rangle"'] &
\end{tikzcd}$$

$$
\begin{tikzcd}
(A \times B) \times C \arrow[rr, bend left, "\langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle"] & & A \times (B \times C) \arrow[ll, bend left, "\langle id \times \pi_1, \pi_2 \cdot \pi_2 \rangle"]
\end{tikzcd}
$$

Cálculo de Programas

Class T03

$$\begin{tikzcd}
(A \times B) \times C \arrow[r, bend left, "assocr"] & A \times (B \times C) \arrow[l, bend left, "assocl"]
\end{tikzcd}$$

$$\begin{aligned}
assocr &= \langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle \\
assocl &= \langle id \times \pi_1, \pi_2 \cdot \pi_2 \rangle
\end{aligned}$$

$$
\begin{tikzcd}
(A \times B) \times C \arrow[r, bend left, "assocr"] & A \times (B \times C) \arrow[l, bend left, "assocl"]
\end{tikzcd}
$$

$$assocr \cdot assocl = id$$

$$
\begin{tikzcd}
(A \times B) \times C \arrow[rr, bend left, "assocr"] & & A \times (B \times C) \arrow[ll, bend left, "assocl"]
\end{tikzcd}
$$

$$assocr \cdot assocl = id$$

$$assocl \cdot assocr = id$$

Isomorphism

$$\begin{tikzcd}(A \times B) \times C \arrow[rr, bend left, "assocr"] \arrow[rr, phantom, "\cong"] & & A \times (B \times C) \arrow[ll, bend left, "assocl"]\end{tikzcd}$$

$$assocr \cdot assocl = id$$

$$assocl \cdot assocr = id$$

Isomorphism

$$\begin{tikzcd} A \times B \arrow[rr, bend left, "swap"] \arrow[rr, phantom, "\cong"] & & B \times A \arrow[ll, bend left, "swap"] \end{tikzcd}$$

$$swap \cdot swap = id$$

Isomorphism

$$\begin{tikzcd} A \arrow[r, bend left, "f"] \arrow[r, phantom, "\cong"] & B \arrow[l, bend left, "g"] \end{tikzcd}$$

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

Isomorphism

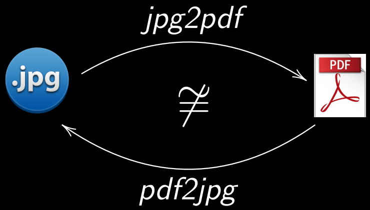
$$\underbrace{iso}_{the\ \ same}\ (\iota\sigma o)$$

Isomorphism

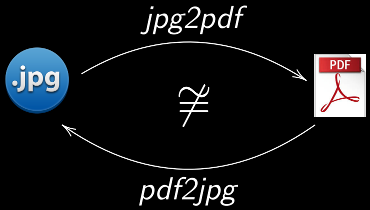
$$\underbrace{\textit{iso}}_{\textit{the same}}\ (\iota\sigma o) + \underbrace{\textit{morphism}}_{\textit{shape}}\ \ (\mu o\rho\phi\iota\sigma\mu o\zeta)$$

"Similar shape"

Practical problem!



Practical problem!



$$\mathit{jpg2pdf} \cdot \mathit{pdf2jpg} \neq \mathit{id}$$

$$\mathit{pdf2jpg} \cdot \mathit{jpg2pdf} \neq \mathit{id}$$

Format conversion

Need

$$\begin{tikzcd} A \arrow[d, "f"'] \\ B \end{tikzcd}$$

Format conversion

Need	Reusable
$A \xrightarrow{f} B$	$C \xrightarrow{r} D$

Format conversion

$$\begin{tikzcd} \text{Need} & \text{Reusable} \\ A \arrow[r, "export"] \arrow[d, "f"'] & C \arrow[d, "r"] \\ B & D \end{tikzcd}$$

Format conversion

Need | Reusable

$$
\begin{tikzcd}
A \arrow[r, "export"] \arrow[d, "f"'] & C \arrow[d, "r"] \\
B & D \arrow[l, "import"]
\end{tikzcd}
$$

Format conversion

Need Reusable

$$\begin{tikzcd} A \arrow[r, "export"] \arrow[d, "f"'] & C \arrow[d, "r"] \\ B & D \arrow[l, "import"] \end{tikzcd}$$

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

Format conversion

Need   Reusable

$$\begin{tikzcd}
A \arrow[rr, "\mathit{export}"] \arrow[d, "f"'] & & C \arrow[d, "r"] \\
B \arrow[rru, phantom, "(\cong)"] & & D \arrow[ll, "\mathit{import}"]
\end{tikzcd}$$

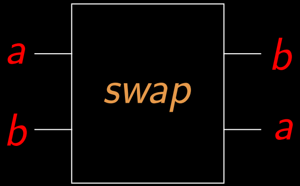
$$f = \mathit{import} \cdot r \cdot \mathit{export}$$

By the way — *swap*

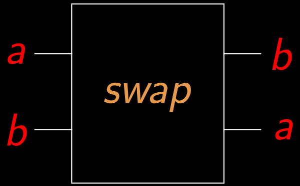
$$
\begin{tikzcd}
A \times B \arrow[rr, bend left, "\mathit{swap}"] \arrow[rr, phantom, "\cong"] & & B \times A \arrow[ll, bend left, "\mathit{swap}"]
\end{tikzcd}
$$

Isomorphisms are **reversible** computations

By the way — *swap*

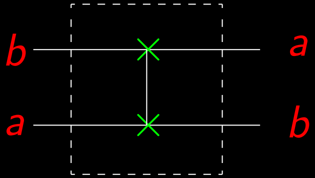


By the way — *swap*



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

By the way — *swap*



What about alternation $[f, g]$?...

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

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

Recall...

$$
\begin{tikzcd}
A \arrow[r, "i_1"] \arrow[dr, "f"'] & A + B \arrow[d, "{[f, g]}"] & B \arrow[l, "i_2"'] \arrow[dl, "g"] \\
& C &
\end{tikzcd}
$$

Compare...

$$\begin{tikzcd}
A & A \times B \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & B \\
 & C \arrow[ul, "f"] \arrow[u, "\langle f, g \rangle"] \arrow[ur, "g"'] &
\end{tikzcd}$$

$+$-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 $f + g$

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

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

$f + g$?

Sum of two functions

$$\begin{tikzcd} D \arrow[d, "f"'] \\ B \end{tikzcd}$$

Sum of two functions

$$\begin{tikzcd} D \arrow[d, "f"'] & & E \arrow[d, "g"] \\ B & & C \end{tikzcd}$$

Sum of two functions

$$\begin{tikzcd}
D \arrow[d, "f"'] & & E \arrow[d, "g"] \\
B \arrow[r, "i_1"'] & B + C & C \arrow[l, "i_2"]
\end{tikzcd}$$

Sum of two functions

$$
\begin{tikzcd}
D \arrow[d, "f"'] \arrow[dr] & & E \arrow[d, "g"] \\
B \arrow[r, "i_1"'] & B + C & C \arrow[l, "i_2"]
\end{tikzcd}
$$

Sum of two functions

$$
\begin{tikzcd}
D \arrow[d, "f"'] \arrow[dr] & & E \arrow[dl] \arrow[d, "g"] \\
B \arrow[r, "i_1"'] & B + C & C \arrow[l, "i_2"]
\end{tikzcd}
$$

Sum of two functions

$$\begin{tikzcd}
D \arrow[r, "i_1"] \arrow[d, "f"'] \arrow[dr] & D + E \arrow[d, "f + g"] & E \arrow[l, "i_2"'] \arrow[d, "g"] \arrow[dl] \\
B \arrow[r, "i_1"'] & B + C & C \arrow[l, "i_2"]
\end{tikzcd}$$

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

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

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

Coproduct laws

"Just reverse the arrows", cf.

+-Absorption	$[h, k] \cdot (f + g) = [h \cdot f, k \cdot g]$	(2.43)
+-Fusion	$f \cdot [h, k] = [f \cdot h, f \cdot k]$	(2.42)
+-Reflexion	$[i_1, i_2] = id$	(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
$$[h, k] = [f, g] \Leftrightarrow \left\{ \begin{array}{l} h = f \\ k = g \end{array} \right. \tag{2.66}$$

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

+-Functor-*id*
$$id + id = id \tag{2.45}$$

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
$f + g$	Structural alternation (coproduct)

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

What about...

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

What about...

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

...?

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

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

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

Recall

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

$$\begin{tikzcd} A \arrow[rr, bend left, "id"] \arrow[rr, phantom, "\cong"] & & A \arrow[ll, bend left, "id"] \end{tikzcd}$$

Recall

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

$$\begin{tikzcd} A \arrow[r, bend left, "id"] \arrow[r, phantom, "\cong"] & A \arrow[l, bend left, "id"] \end{tikzcd}$$

$$\begin{tikzcd} A \arrow[r, bend left, "f"] \arrow[r, phantom, "\cong"] & B \arrow[l, bend left, "g"] \end{tikzcd}$$

$$f \cdot g = id$$

$$g \cdot f = id$$

Recall

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

Recall

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

$$\begin{tikzcd} A \times B \arrow[rr, bend left, "swap"] \arrow[rr, phantom, "\cong"] & & B \times A \arrow[ll, bend left, "swap"] \end{tikzcd}$$

Recall

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

$$\begin{tikzcd} A \times B \arrow[rr, bend left, "swap"] \arrow[rr, phantom, "\cong"] & & B \times A \arrow[ll, bend left, "swap"] \end{tikzcd}$$

$$\begin{tikzcd} A \arrow[rr, bend left, "?"] \arrow[rr, phantom, "?"] & & A \times B \arrow[ll, bend left, "\pi_1"] \end{tikzcd}$$

Recall

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

$$\begin{tikzcd} A \times B \arrow[rr, bend left, "swap"] \arrow[rr, phantom, "\cong"] & & B \times A \arrow[ll, bend left, "swap"] \end{tikzcd}$$

$$\begin{tikzcd} A \arrow[rr, bend left, "?"] \arrow[rr, phantom, "?"] & & A \times B \arrow[ll, bend left, "\pi_1"] \end{tikzcd}$$

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

Recall

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

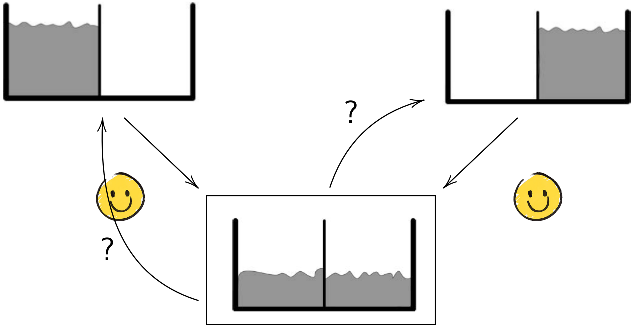
$$\begin{tikzcd} A \times B \arrow[rr, bend left, "swap"] \arrow[rr, phantom, "\cong"] & & B \times A \arrow[ll, bend left, "swap"] \end{tikzcd}$$

$$\begin{tikzcd} A \arrow[rr, bend left, "?"] \arrow[rr, phantom, "?"] & & A \times B \arrow[ll, bend left, "\pi_1"] \end{tikzcd}$$

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

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

Irreversibility...



Rolf Landauer (1927–99)

Even worse than 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...

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

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

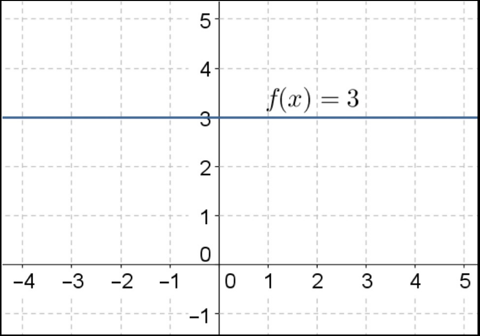
$$\mathit{one}\ \texttt{"string"} = 1$$

$$\mathit{zero}\ [50\,..\,1000] = 0$$

Even worse than π_1...

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

$$f\ x = 3$$



$\mathit{one}\ 10 = 1$

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

$\mathit{zero}\ [50\,..\,1000] = 0$

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

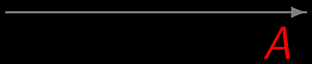
$3 \in \mathbb{N}_0$

$f\ x = 3$

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

$b_0 \in B$

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



Constant functions

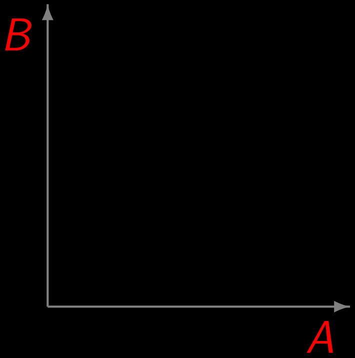
$3 \in \mathbb{N}_0$

$f\ x = 3$

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

$b_0 \in B$

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



Constant functions

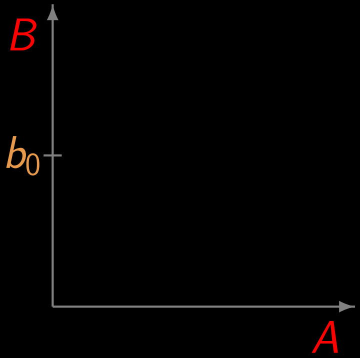
$$3 \in \mathbb{N}_0$$

$$b_0 \in B$$

$$f\ x = 3$$

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

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



Constant functions

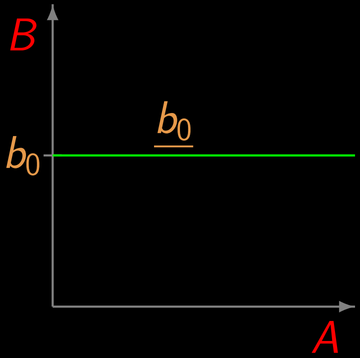
$3 \in \mathbb{N}_0$

$f\ x = 3$

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

$b_0 \in B$

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



Constant functions

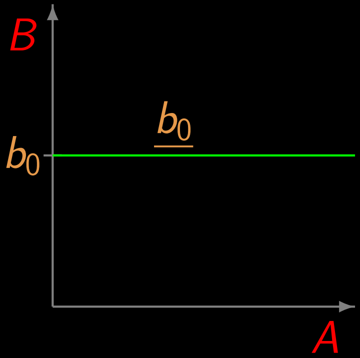
$3 \in \mathbb{N}_0$

$f\ x = 3$

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

$b_0 \in B$

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



(**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

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

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

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

Properties

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

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

$$\begin{tikzcd}
 & C \arrow[d, "f"] \\
B & A \arrow[l, "\underline{b}"']
\end{tikzcd}$$

Properties

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

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

$$\begin{tikzcd}
 & C \arrow[dl, "\underline{b}"'] \arrow[d, "f"] \\
B & A \arrow[l, "\underline{b}"]
\end{tikzcd}$$

Properties

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

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

$$\begin{tikzcd}
 & C \arrow[dl, "\underline{b}"] \arrow[d, "f"] \\
B \arrow[d, "g"'] & A \arrow[l, "\underline{b}"'] \\
D &
\end{tikzcd}$$

Properties

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

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

$$\begin{tikzcd}
 & C \arrow[dl, "\underline{b}"'] \arrow[d, "f"] \\
B \arrow[d, "g"'] & A \arrow[l, "\underline{b}"'] \arrow[dl, "\underline{g\ b}"] \\
D &
\end{tikzcd}$$

No loss of information?

Isomorphisms shining...

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

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

Isomorphisms

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

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

Isomorphisms

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

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

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

Isomorphisms

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

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

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

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

Isomorphisms

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

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

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

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

$$A + 0 \cong A \qquad ?$$

Isomorphisms

$$A + A \;\cong\; 2 \times A \qquad ?$$

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

$$A \times 1 \;\cong\; A \qquad ?$$

$$A \times 0 \;\cong\; 0 \qquad ?$$

$$A + 0 \;\cong\; A \qquad ?$$

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

Distributivity

$$
\begin{tikzcd}
A \times (B + C) \arrow[rr, bend left, "distr"] \arrow[r, phantom, "\cong"] & & A \times B + A \times C \arrow[ll, bend left, "undistr"]
\end{tikzcd}
$$

Distributivity

$$\begin{tikzcd} A \times (B + C) \arrow[rr, bend left, "distr"] \arrow[rr, phantom, "\cong"] & & A \times B + A \times C \arrow[ll, bend left, "undistr"] \end{tikzcd}$$

$$\begin{tikzcd} (B + C) \times A \arrow[rr, bend left, "distl"] \arrow[rr, phantom, "\cong"] & & B \times A + C \times A \arrow[ll, bend left, "undistl"] \end{tikzcd}$$

Distributivity

$$
\begin{tikzcd}
A \times (B + C) \arrow[rr, bend left, "distr"] & \cong & A \times B + A \times C \arrow[ll, bend left, "undistr"]
\end{tikzcd}
$$

Distributivity

$$\begin{tikzcd} A \times (B + C) \arrow[rr, bend left, "distr"] \arrow[rr, phantom, "\cong"] & & A \times B + A \times C \arrow[ll, bend left, "undistr"] \end{tikzcd}$$

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

Distributivity

$$\begin{tikzcd} A \times (B + C) \arrow[rr, bend left, "distr"] \arrow[r, phantom, "\cong"] & & A \times B + A \times C \arrow[ll, bend left, "undistr"] \end{tikzcd}$$

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

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

Distributivity

$$\begin{tikzcd} A \times (B + C) \arrow[rr, bend left, "distr"] & \cong & A \times B + A \times C \arrow[ll, bend left, "undistr"] \end{tikzcd}$$

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

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

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

Distributivity

$$
\begin{tikzcd}
A \times (B + C) \arrow[rr, bend left, "distr"] & \cong & A \times B + A \times C \arrow[ll, bend left, "undistr"]
\end{tikzcd}
$$

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

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

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

Distributivity

$$
\begin{tikzcd}
A \times (B + C) \arrow[rr, bend left, "distr"] \arrow[r, phantom, "\cong"] & {} & A \times B + A \times C \arrow[ll, bend left, "undistr"]
\end{tikzcd}
$$

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

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

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

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

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

$$\begin{tikzcd}
A \times (B + C) \arrow[rr, bend left, "\mathit{distr}"] \arrow[rr, phantom, "\cong"] & & A \times B + A \times C \arrow[ll, bend left, "\mathit{undistr}"]
\end{tikzcd}$$

Distributivity

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

$$\begin{tikzcd} A \times (B + C) \arrow[rr, bend left, "distr"] \arrow[r, phantom, "\cong"] & {} & A \times B + A \times C \arrow[ll, bend left, "undistr"] \end{tikzcd}$$

By definition of $f \times g$

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

Cálculo de Programas

Class T04

The exchange law

$$\big\downarrow \alpha$$

The exchange law

$$\begin{tikzcd} A + B \arrow[d, "\alpha"] \\ {} \end{tikzcd}$$

The exchange law

$$\begin{tikzcd} A + B \arrow[d, "\alpha"] \\ C \times D \end{tikzcd}$$

Exchange law

$$\begin{tikzcd} A \arrow[r, "i_1"] \arrow[dr, "i"'] & A + B \arrow[d, "{[i, j]}"] & B \arrow[l, "i_2"'] \arrow[dl, "j"] \\ & C \times D & \end{tikzcd}$$

Exchange law

$$
\begin{tikzcd}
 & A + B \arrow[dl, "m"'] \arrow[d, "\langle m, n \rangle"] \arrow[dr, "n"] & \\
C & C \times D \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] & D
\end{tikzcd}
$$

Exchange law

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

One equation, four unknowns

Exchange law

$$
\begin{tikzcd}
 & A + B \arrow[dl, "m"'] \arrow[d, "\langle m, n \rangle"] \arrow[dr, "n"] & \\
C & C \times D \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] & D
\end{tikzcd}
$$

Exchange law

$$
\begin{tikzcd}
A \arrow[r, "i_1"] & A + B \arrow[dl, "m"'] \arrow[d, "\langle m, n \rangle"] \arrow[dr, "n"] & B \arrow[l, "i_2"'] \\
C & C \times D \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] & D
\end{tikzcd}
$$

Exchange law

$$
\begin{tikzcd}
A \arrow[r, "i_1"] \arrow[d, "f"] & A + B & B \arrow[l, "i_2"'] \arrow[dll, "g"] \\
C & C \times D \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] & D
\end{tikzcd}
$$

Exchange law

$$\begin{tikzcd}
A \arrow[r, "i_1"] \arrow[drr, "h"'] & A + B & B \arrow[l, "i_2"'] \arrow[d, "k"] \\
C & C \times D \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] & D
\end{tikzcd}$$

Exchange law

$$
\begin{tikzcd}[column sep=huge, row sep=huge]
A \arrow[r, "i_1"] & A + B \arrow[dl, "{m = [f, g]}"'] \arrow[d, "{\langle m, n \rangle}"] \arrow[dr, "{n = [h, k]}"] & B \arrow[l, "i_2"'] \\
C & C \times D \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] & D
\end{tikzcd}
$$

Exchange law

$$
\begin{tikzcd}
A \arrow[r, "i_1"] \arrow[dr, "i = \langle f, h \rangle"'] & A + B \arrow[d, "{[i, j]}" description] & B \arrow[l, "i_2"'] \arrow[dl, "j = \langle g, k \rangle"] \\
C & C \times D \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] & D
\end{tikzcd}
$$

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

Exchange law — Example

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

$$\begin{tikzcd}
A \times (B + C) \arrow[rr, bend left, "distr"] \arrow[r, phantom, "\cong"] & & A \times B + A \times C \arrow[ll, bend left, "undistr"]
\end{tikzcd}$$

Exchange law — Example

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

$$\begin{tikzcd} A \times (B + C) \arrow[rr, bend left, "distr"] \arrow[r, phantom, "\cong"] & {} & A \times B + A \times C \arrow[ll, bend left, "undistr"] \end{tikzcd}$$

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

The types 0, 1 and 2...

The type 1

In Haskell:

$$() \; :: \; ()$$

Then

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

Isomorphism

$$\begin{tikzcd} A \times 1 \arrow[rr, bend left, "\pi_1"] \arrow[rr, phantom, "\cong"] & & A \arrow[ll, bend left, "\langle id, \mathbf{!} \rangle"] \end{tikzcd}$$

The type $\mathbf{1}$

In Haskell:

$$() :: ()$$

Then

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

Isomorphism

$$\begin{tikzcd} A \times \mathbf{1} \arrow[rr, bend left, "\pi_1"] \arrow[rr, phantom, "\cong"] & & A \arrow[ll, bend left, "\langle id, \mathbf{!} \rangle"] \end{tikzcd}$$

... "$\mathbf{!}$" ?

The type $\mathbb{1}$

Fact functions involving type $\mathbb{1}$
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

$$\begin{tikzcd}
A \times 0 \arrow[rr, bend left, "id"] \arrow[r, phantom, ""] & = & 0 \arrow[ll, bend left, "id"]
\end{tikzcd}$$

The type 0

$$\begin{tikzcd} A \times 0 \arrow[rr, bend left, "id"] & = & 0 \arrow[ll, bend left, "id"] \end{tikzcd}$$

In fact

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

The type 0

$$\begin{tikzcd} A \times 0 \arrow[rr, bend left, "id"] \arrow[rr, phantom, "="] & & 0 \arrow[ll, bend left, "id"] \end{tikzcd}$$

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

$$\begin{tikzcd} A \times 0 \arrow[rr, bend left, "id"] \arrow[rr, phantom, "="] & & 0 \arrow[ll, bend left, "id"] \end{tikzcd}$$

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 $1 + A$

$$
\begin{tikzcd}
1 \arrow[r, "i_1"] & 1 + A & A \arrow[l, "i_2"'] \arrow[dl, "f"] \\
 & B &
\end{tikzcd}
$$

The type $1 + A$

$$\begin{tikzcd}
1 \arrow[r, "i_1"] \arrow[dr, "\underline{b_0}"'] & 1 + A & A \arrow[l, "i_2"'] \arrow[dl, "f"] \\
& B &
\end{tikzcd}$$

The type $1 + A$

$$\begin{tikzcd}
1 \arrow[r, "i_1"] \arrow[dr, "\underline{b_0}"'] & 1 + A \arrow[d, "{[\underline{b_0}, f]}"] & A \arrow[l, "i_2"'] \arrow[dl, "f"] \\
& B &
\end{tikzcd}$$

The type $1 + A$

$$\begin{tikzcd}
1 \arrow[r, "i_1"] \arrow[dr, "\underline{b_0}"'] & 1 + A \arrow[d, "{[\underline{b_0}, f]}"] & A \arrow[l, "i_2"'] \arrow[dl, "f"] \\
& B &
\end{tikzcd}$$

$1 + A$ "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!

$$
\begin{tikzcd}
\mathit{Int} \arrow[r, "i_1"] \arrow[dr, "\texttt{CC}"'] & \mathit{Int} + \mathit{Int} \arrow[d, "{[\texttt{CC}, \texttt{NIF}]}"] & \mathit{Int} \arrow[l, "i_2"'] \arrow[dl, "\texttt{NIF}"] \\
& \texttt{Iden} &
\end{tikzcd}
$$

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

More isomorphisms!

$$
\begin{tikzcd}
\texttt{Iden} \arrow[rr, bend left, "\mathbf{out}=\mathbf{in}^{\circ}"] \arrow[r, phantom, "\cong"] & \, & Int + Int \arrow[ll, bend left, "\mathbf{in}=[\texttt{CC},\texttt{NIF}]"]
\end{tikzcd}
$$

More isomorphisms!

$$\begin{tikzcd}[column sep=huge] \mathtt{Iden} \arrow[r, bend left, "\mathbf{out}=\mathbf{in}^{\circ}"] \arrow[r, phantom, "\cong"] & \mathit{Int} + \mathit{Int} \arrow[l, bend left, "\mathbf{in}={[\mathtt{CC},\mathtt{NIF}]}"] \end{tikzcd}$$

Concrete syntax

More isomorphisms!

$$\begin{tikzcd}[column sep=huge]
\mathtt{Iden} \arrow[rr, bend left, "\mathsf{out} = \mathsf{in}^{\circ}"] \arrow[rr, phantom, "\cong"] & & Int + Int \arrow[ll, bend left, "\mathsf{in} = {[\mathtt{CC}, \mathtt{NIF}]}"]
\end{tikzcd}$$

Concrete syntax

Abstract syntax

More isomorphisms!

$$\begin{tikzcd}
\mathtt{Iden} \arrow[rr, bend left, "\mathbf{out}=\mathbf{in}^{\circ}"] \arrow[rr, phantom, "\cong"] & & \mathit{Int} + \mathit{Int} \arrow[ll, bend left, "\mathbf{in}=[\mathtt{CC},\mathtt{NIF}]"]
\end{tikzcd}$$

Concrete syntax

Abstract syntax

in ("get in")

More isomorphisms!

$$\begin{tikzcd} \mathtt{Iden} \arrow[r, bend left, "\mathbf{out}=\mathbf{in}^{\circ}"] \arrow[r, phantom, "\cong"] & Int + Int \arrow[l, bend left, "\mathbf{in}={[}\mathtt{CC},\mathtt{NIF}{]}"] \end{tikzcd}$$

Concrete syntax

Abstract syntax

in (“get in”)

out (“get out”)

Maybe

$$\begin{tikzcd}
1 \arrow[r, "i_1"] \arrow[dr, "\underline{\texttt{Nothing}}"'] & 1 + A \arrow[d, "{[\underline{\texttt{Nothing}}, \texttt{Just}]}" description] & A \arrow[l, "i_2"'] \arrow[dl, "\texttt{Just}"] \\
& \texttt{Maybe}\ A &
\end{tikzcd}$$

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

Maybe

$$\begin{tikzcd}
\mathtt{Maybe}\ A \arrow[rr, bend left, "\mathbf{out} = \mathbf{in}^{\circ}"] \arrow[r, phantom, "\cong"] & {} & 1 + A \arrow[ll, bend left, "\mathbf{in} \ = \ {[\underline{\mathtt{Nothing}}, \mathtt{Just}]}"]
\end{tikzcd}$$

Maybe

$$\begin{tikzcd} \mathtt{Maybe}\ A \arrow[rr, bend left, "\mathsf{out}=\mathsf{in}^{\circ}"] \arrow[r, phantom, "\cong"] & & 1 + A \arrow[ll, bend left, "\mathsf{in}\ =\ {[\underline{\mathtt{Nothing}}, \mathtt{Just}]}"] \end{tikzcd}$$

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

Maybe — calculate **out**

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

Maybe — calculate out

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

Maybe — calculate out

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

Maybe — calculate out

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

$$\Leftrightarrow \quad \{\ \mathsf{in} = [\underline{\mathtt{Nothing}}, \mathtt{Just}]\ \}$$

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

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

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

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

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

Maybe

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

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

Maybe

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

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

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

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

Maybe

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

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

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

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

$$\begin{tikzcd}
\mathtt{Maybe}\ A \arrow[rr, bend left, "\mathbf{out}"] \arrow[rr, phantom, "\cong"] & & 1 + A \arrow[ll, bend left, "\mathbf{in}\ =\ {[\underline{\mathtt{Nothing}}, \mathtt{Just}]}"]
\end{tikzcd}$$

Either

$$\begin{tikzcd} \texttt{Either}\ A\ B \arrow[rr, bend left, "\textsf{out}"] \arrow[rr, phantom, "\cong"] & & A + B \arrow[ll, bend left, "\textsf{in} = [\texttt{Left},\texttt{Right}]"] \end{tikzcd}$$

Either

$$\begin{tikzcd} \texttt{Either}\ A\ B \arrow[rr, bend left, "\mathsf{out}"] \arrow[rr, phantom, "\cong"] & & A + B \arrow[ll, bend left, "{\mathsf{in} = [\texttt{Left},\texttt{Right}]}"] \end{tikzcd}$$

$$\begin{tikzcd} A \arrow[r, "i_1"] \arrow[dr, "\texttt{Left}"'] & A + B \arrow[d, "{[\texttt{Left},\texttt{Right}]}"] & B \arrow[l, "i_2"'] \arrow[dl, "\texttt{Right}"] \\ & \texttt{Either}\ A\ B & \end{tikzcd}$$

$[f, g]$

$\texttt{either}\ f\ g$

Polymorphism
(and why it matters!)

Polymorphism

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

Polymorphism

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

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

Polymorphism

$$reverse :: [a] \rightarrow [a]$$

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

Polymorphism

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

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

Polymorphism

$$\mathit{reverse} :: [a] \to [a]$$

$$\mathit{reverse} :: [\mathit{Int}] \to [\mathit{Int}]$$
$$\mathit{reverse} :: \mathit{String} \to \mathit{String}$$
$$\ldots$$

"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

$$[1, 5, 0, 5, 2] \xleftarrow{\textit{reverse}} [2, 5, 0, 5, 1]$$

Natural properties

$$
\begin{tikzcd}
{[1, 5, 0, 5, 2]} \arrow[d, mapsto, "\mathit{succs}"'] & {[2, 5, 0, 5, 1]} \arrow[l, mapsto, "\mathit{reverse}"'] \\
{[2, 6, 1, 6, 3]} &
\end{tikzcd}
$$

Natural properties

$$
\begin{tikzcd}
{[1,5,0,5,2]} \arrow[d, mapsto, "succs"'] & {[2,5,0,5,1]} \arrow[l, mapsto, "reverse"'] \arrow[d, mapsto, "succs"] \\
{[2,6,1,6,3]} & {[3,6,1,6,2]}
\end{tikzcd}
$$

Natural properties

$$
\begin{tikzcd}
{[1, 5, 0, 5, 2]} \arrow[d, mapsto, "\mathit{succs}"'] & {[2, 5, 0, 5, 1]} \arrow[l, mapsto, "\mathit{reverse}"'] \arrow[d, mapsto, "\mathit{succs}"] \\
{[2, 6, 1, 6, 3]} & {[3, 6, 1, 6, 2]} \arrow[l, mapsto, "\mathit{reverse}"]
\end{tikzcd}
$$

Natural properties

$$\begin{tikzcd}
{[1, 5, 0, 5, 2]} \arrow[d, mapsto, "\mathit{map}\ \mathit{succ}"'] & {[2, 5, 0, 5, 1]} \arrow[l, mapsto, "\mathit{reverse}"'] \arrow[d, mapsto, "\mathit{map}\ \mathit{succ}"] \\
{[2, 6, 1, 6, 3]} & {[3, 6, 1, 6, 2]} \arrow[l, mapsto, "\mathit{reverse}"]
\end{tikzcd}$$

Natural properties

$$[1, 5, 0, 5, 2] \xleftarrow{\textit{reverse}} [2, 5, 0, 5, 1]$$

Natural properties

$$\begin{tikzcd}
{[1,5,0,5,2]} \arrow[d, mapsto, "map\ even"'] & & {[2,5,0,5,1]} \arrow[ll, mapsto, "reverse"'] \\
{[F,F,T,F,T]} & &
\end{tikzcd}$$

Natural properties

$$\begin{tikzcd}
{[1,5,0,5,2]} \arrow[d, mapsto, "map\ even"'] & & {[2,5,0,5,1]} \arrow[ll, mapsto, "reverse"'] \arrow[d, mapsto, "map\ even"] \\
{[F,F,T,F,T]} & & {[T,F,T,F,F]}
\end{tikzcd}$$

Natural properties

$$
\begin{tikzcd}
{[1, 5, 0, 5, 2]} \arrow[d, mapsto, "\mathit{map}\ \mathit{even}"'] & & {[2, 5, 0, 5, 1]} \arrow[ll, mapsto, "\mathit{reverse}"'] \arrow[d, mapsto, "\mathit{map}\ \mathit{even}"] \\
{[F, F, T, F, T]} & & {[T, F, T, F, F]} \arrow[ll, mapsto, "\mathit{reverse}"]
\end{tikzcd}
$$

Natural properties

$$[2, 5, 0] \xleftarrow{\mathit{take}\ 3} [2, 5, 0, 5, 1]$$

Natural properties

$$\begin{tikzcd}[2, 5, 0] \arrow[d, mapsto, "map\ (2*)"'] & & [2, 5, 0, 5, 1] \arrow[ll, mapsto, "take\ 3"'] \\ [4, 10, 0] & & \end{tikzcd}$$

Natural properties

$$
\begin{tikzcd}
{[2,5,0]} \arrow[d, mapsto, "map\ (2*)"'] & & {[2,5,0,5,1]} \arrow[ll, mapsto, "take\ 3"'] \arrow[d, mapsto, "map\ (2*)"] \\
{[4,10,0]} & & {[4,10,0,10,2]}
\end{tikzcd}
$$

Natural properties

$$
\begin{tikzcd}
{[2,5,0]} \arrow[d, mapsto, "map\ (2*)"'] & {[2,5,0,5,1]} \arrow[l, mapsto, "take\ 3"'] \arrow[d, mapsto, "map\ (2*)"] \\
{[4,10,0]} & {[4,10,0,10,2]} \arrow[l, mapsto, "take\ 3"]
\end{tikzcd}
$$

Natural properties

$$\texttt{"ogramas"} \xleftarrow{\;drop\ 2\;} \texttt{"Programas"}$$

Natural properties

$$\begin{tikzcd}
\texttt{"ogramas"} \arrow[d, mapsto, "\mathit{map}\ \mathit{upper}"'] & & \texttt{"Programas"} \arrow[ll, mapsto, "\mathit{drop}\ 2"'] \\
\texttt{"OGRAMAS"} & &
\end{tikzcd}$$

Natural properties

$$
\begin{tikzcd}
\texttt{"ogramas"} \arrow[d, mapsto, "\mathit{map}\ \mathit{upper}"'] & \texttt{"Programas"} \arrow[l, mapsto, "\mathit{drop}\ 2"'] \arrow[d, mapsto, "\mathit{map}\ \mathit{upper}"] \\
\texttt{"OGRAMAS"} & \texttt{"PROGRAMAS"}
\end{tikzcd}
$$

Natural properties

$$
\begin{tikzcd}
\texttt{"ogramas"} \arrow[d, mapsto, "\mathit{map}\ \mathit{upper}"'] & \texttt{"Programas"} \arrow[l, mapsto, "\mathit{drop}\ 2"'] \arrow[d, mapsto, "\mathit{map}\ \mathit{upper}"] \\
\texttt{"OGRAMAS"} & \texttt{"PROGRAMAS"} \arrow[l, mapsto, "\mathit{drop}\ 2"]
\end{tikzcd}
$$

Back to...

$$\begin{tikzcd}(A \times B) \times C \arrow[rr, bend left, "assocr"] \arrow[rr, phantom, "\cong"] & & A \times (B \times C) \arrow[ll, bend left, "assocl"]\end{tikzcd}$$

$$assocr \cdot assocl = id$$

$$assocl \cdot assocr = id$$

"Natural" property

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

“Natural” property

$$\begin{tikzcd}
((2, T), \texttt{"CP"}) \arrow[d, mapsto, "(succ \times \neg) \times !"'] & (2, (T, \texttt{"CP"})) \arrow[l, mapsto, "assocl"'] \\
((3, F), ())
\end{tikzcd}$$

"Natural" property

$$
\begin{tikzcd}
((2, T), \texttt{"CP"}) \arrow[d, mapsto, "(succ \times \neg) \times\, !"'] & & (2, (T, \texttt{"CP"})) \arrow[ll, mapsto, "assocl"'] \arrow[d, mapsto, "succ \times (\neg \times\, !)"] \\
((3, F), ()) & & (3, (F, ()))
\end{tikzcd}
$$

“Natural” property

$$
\begin{tikzcd}
((2, T), \texttt{"CP"}) \arrow[d, mapsto, "(\mathit{succ} \times \neg) \times \mathbf{!}"'] & (2, (T, \texttt{"CP"})) \arrow[l, mapsto, "\mathit{assocl}"'] \arrow[d, mapsto, "\mathit{succ} \times (\neg \times \mathbf{!})"] \\
((3, F), ()) & (3, (F, ())) \arrow[l, mapsto, "\mathit{assocl}"]
\end{tikzcd}
$$

“Natural” property

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

“Natural” property

$$\begin{tikzcd}
((a, b), c) \arrow[d, mapsto, "(f \times g) \times h"'] & & (a, (b, c)) \arrow[ll, mapsto, "assocl"'] \\
((f\ a, g\ b), h\ c) & &
\end{tikzcd}$$

“Natural” property

$$
\begin{tikzcd}
((a, b), c) \arrow[d, mapsto, "(f \times g) \times h"'] & & (a, (b, c)) \arrow[ll, mapsto, "assocl"'] \arrow[d, mapsto, "f \times (g \times h)"] \\
((f\ a, g\ b), h\ c) & & (f\ a, (g\ b, h\ c))
\end{tikzcd}
$$

“Natural” property

$$\begin{tikzcd}
((a, b), c) \arrow[d, mapsto, "(f \times g) \times h"'] & & (a, (b, c)) \arrow[ll, mapsto, "assocl"'] \arrow[d, mapsto, "f \times (g \times h)"] \\
((f\ a, g\ b), h\ c) & & (f\ a, (g\ b, h\ c)) \arrow[ll, mapsto, "assocl"]
\end{tikzcd}$$

Natural properties

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

Natural properties

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

Natural properties

$$
\begin{tikzcd}
(A \times B) \times C \arrow[d, "(f \times g) \times h"'] & & A \times (B \times C) \arrow[ll, "assocl"'] \\
(D \times E) \times F & &
\end{tikzcd}
$$

Natural properties

$$
\begin{tikzcd}
(A \times B) \times C \arrow[d, "(f \times g) \times h"'] & & A \times (B \times C) \arrow[ll, "assocl"'] \arrow[d, "f \times (g \times h)"] \\
(D \times E) \times F & & D \times (E \times F)
\end{tikzcd}
$$

Natural properties

$$\begin{tikzcd}
(A \times B) \times C \arrow[d, "(f \times g) \times h"'] & & A \times (B \times C) \arrow[ll, "assocl"'] \arrow[d, "f \times (g \times h)"] \\
(D \times E) \times F & & D \times (E \times F) \arrow[ll, "assocl"]
\end{tikzcd}$$

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

Practical rule

Back to...

$$
\begin{tikzcd}
A \times B \arrow[rr, bend left, "swap"] \arrow[rr, phantom, "\cong"] & & B \times A \arrow[ll, bend left, "swap"]
\end{tikzcd}
$$

$$swap \cdot swap = id$$

Practical rule

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

Practical rule

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

$$B' \times A' \xleftarrow[swap]{} A' \times B'$$

Practical rule

$$\begin{tikzcd}
B \times A \arrow[d, "\text{"}B \times A\text{"}"'] & A \times B \arrow[l, "swap"'] \arrow[d, "\text{"}A \times B\text{"}"] \\
B' \times A' & A' \times B' \arrow[l, "swap"]
\end{tikzcd}$$

Practical rule

$$\begin{tikzcd}
B \times A \arrow[d, "g \times f"'] & A \times B \arrow[l, "swap"'] \arrow[d, "f \times g"] \\
B' \times A' & A' \times B' \arrow[l, "swap"]
\end{tikzcd}$$

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

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...

$$A \xleftarrow{id} A$$

Simplest case...

$$\begin{tikzcd} A & A \arrow[l, "id"'] \\ A' & A' \arrow[l, "id"] \end{tikzcd}$$

Simplest case...

$$\begin{tikzcd}
A \arrow[d, "{``A''}"'] & A \arrow[l, "id"'] \arrow[d, "{``A''}"] \\
A' & A' \arrow[l, "id"]
\end{tikzcd}$$

Natural-*id* 

$$\begin{tikzcd} A \arrow[d, "f"'] & A \arrow[l, "id"'] \arrow[d, "f"] \\ A' & A' \arrow[l, "id"] \end{tikzcd}$$

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

Natural-π_1

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

Natural-π_1

$$\begin{tikzcd}
A & A \times B \arrow[l, "\pi_1"'] \\
A' & A' \times B' \arrow[l, "\pi_1"]
\end{tikzcd}$$

Natural-π_1

$$\begin{tikzcd}
A \arrow[d, "\text{``}A\text{''}"'] & A \times B \arrow[l, "\pi_1"'] \arrow[d, "\text{``}A \times B\text{''}"] \\
A' & A' \times B' \arrow[l, "\pi_1"]
\end{tikzcd}$$

Natural-π_1

$$\begin{tikzcd}
A \arrow[d, "f"'] & A \times B \arrow[l, "\pi_1"'] \arrow[d, "f \times g"] \\
A' & A' \times B' \arrow[l, "\pi_1"]
\end{tikzcd}$$

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

If ... then ... else...?

if-then-else ?

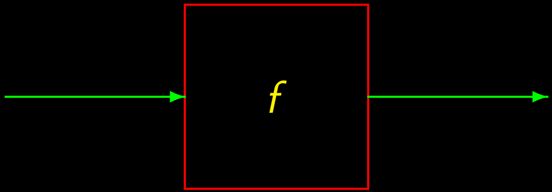
$$y = \textbf{if } p\ x \textbf{ then } f\ x \textbf{ else } g\ x$$

if-then-else ?

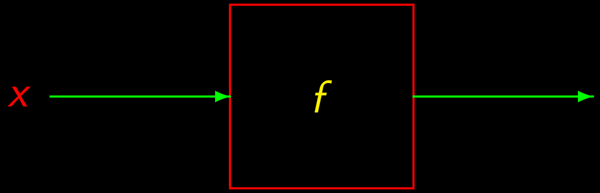
$$y = \textbf{if } p\ x \textbf{ then } f\ x \textbf{ else } g\ x$$

x should **flow** to f **or** g depending on p...

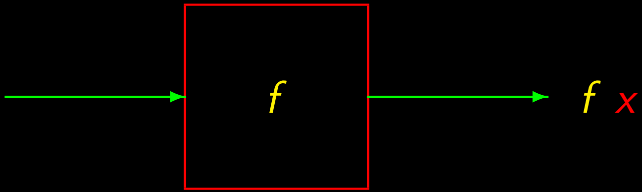
Data flow



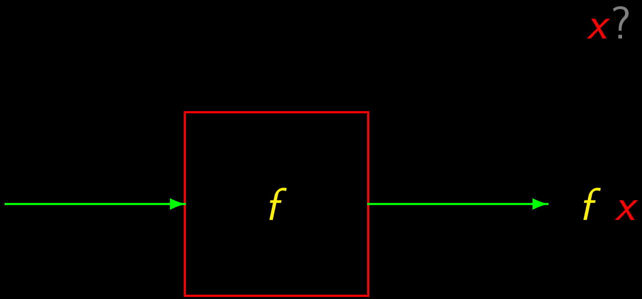
Data flow



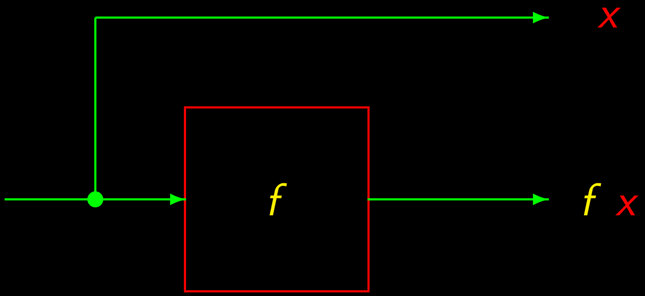
Data flow



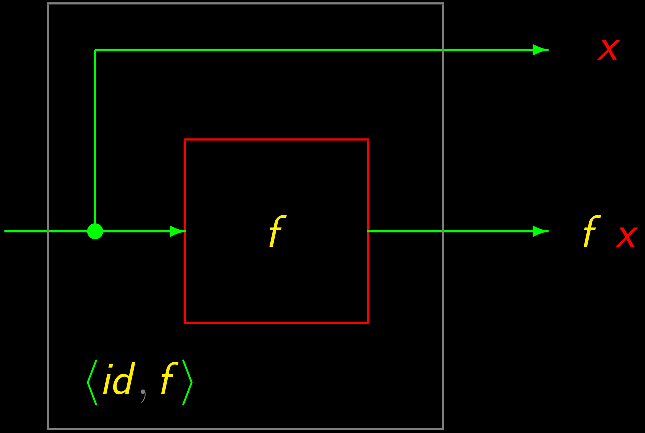
Data flow



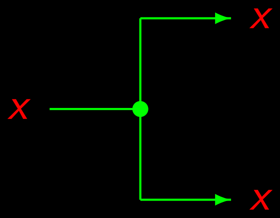
Data flow



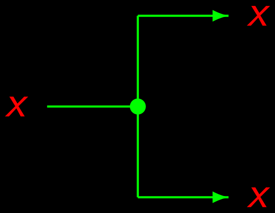
Data flow



Data duplication

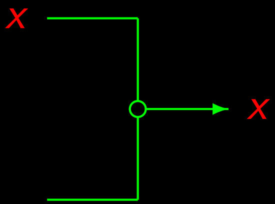


Data duplication



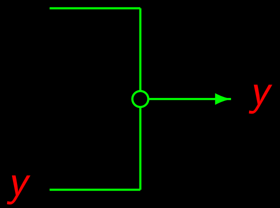
$$dup = \langle id, id \rangle$$

Data joins



$$join = [id, id]$$

Data joins

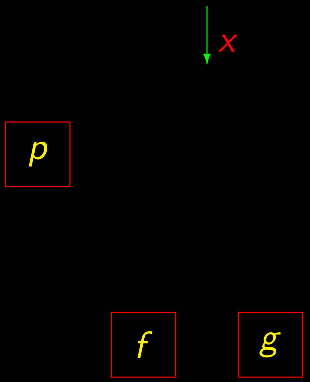


$$join = [id, id]$$

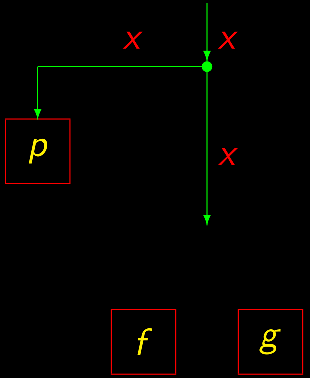
if-then-else

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

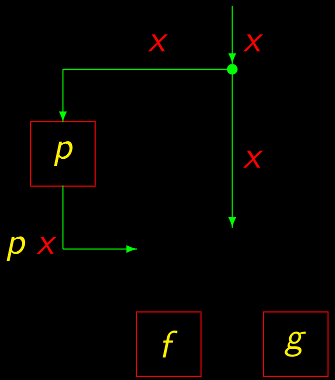
if-then-else



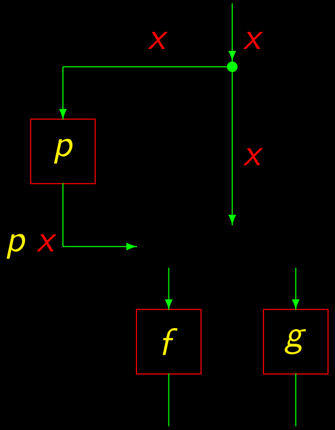
if-then-else



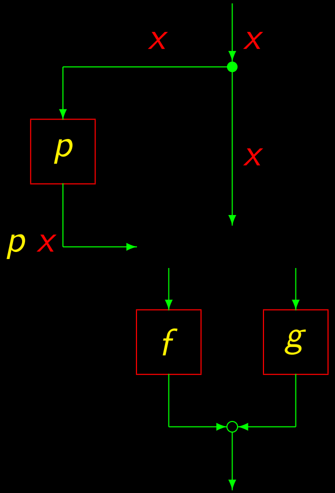
if-then-else



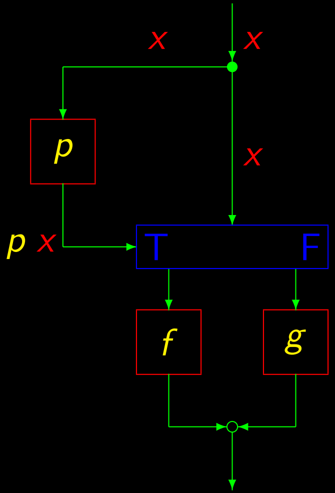
if-then-else



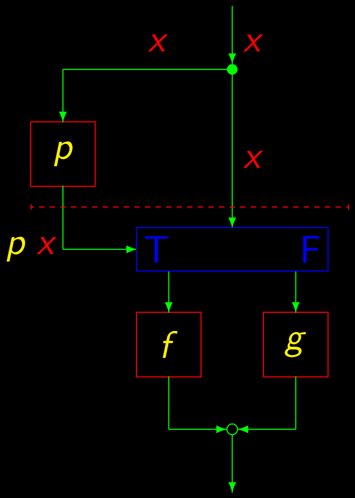
if-then-else



if-then-else



if-then-else



if-then-else (“point-free”)

$x \in A$

if-then-else ("point-free")

$$x \in A$$

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

if-then-else (“point-free”)

$$x \in A$$

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

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

if-then-else ("point-free")

$x \in A$

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

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

$\mathbb{B}$ in Haskell:

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

if-then-else ("point-free")

$x \in A$

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

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

$\mathbb{B}$ in Haskell:

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

T represented by `True`

F represented by `False`

The type 2

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

The type 2

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

$$\begin{tikzcd}
\mathbb{B} \arrow[rr, bend left, "g"] \arrow[rr, phantom, "\cong"] & & \{0, 1\} \arrow[ll, bend left, "f"]
\end{tikzcd}$$

The type 2

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

$$\begin{tikzcd} \mathbb{B} \arrow[rr, bend left, "g"] \arrow[rr, phantom, "\cong"] & & \{0,1\} \arrow[ll, bend left, "f"] \end{tikzcd}$$

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

The type 2

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

$$\begin{tikzcd}
\mathbb{B} \arrow[rr, bend left, "g"] \arrow[rr, phantom, "\cong"] & & \{0,1\} \arrow[ll, bend left, "f"]
\end{tikzcd}$$

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

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

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

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

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

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

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

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

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

Is it the case that...?

$$\begin{tikzcd} 2 \times A \arrow[rr, bend left, "g"] \arrow[rr, phantom, "\cong"] & & A + A \arrow[ll, bend left, "f"] \end{tikzcd}$$

$$\begin{tikzcd} 2 \times A \arrow[rr, bend left, "g"] \arrow[rr, phantom, "\cong"] & & A + A \arrow[ll, bend left, "f"] \end{tikzcd}$$

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

$$g = \cdots (?)$$

Notation

$$\begin{tikzcd} 2 \times A \arrow[rr, bend left, "\alpha^\circ"] \arrow[rr, phantom, "\cong"] & & A + A \arrow[ll, bend left, "\alpha"] \end{tikzcd}$$

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

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

Notation

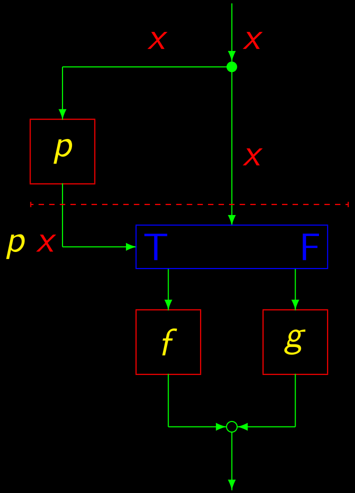
$$\begin{tikzcd} 2 \times A \arrow[rr, bend left, "\alpha^{\circ}"] \arrow[rr, phantom, "\cong"] & & A + A \arrow[ll, bend left, "\alpha"] \end{tikzcd}$$

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

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

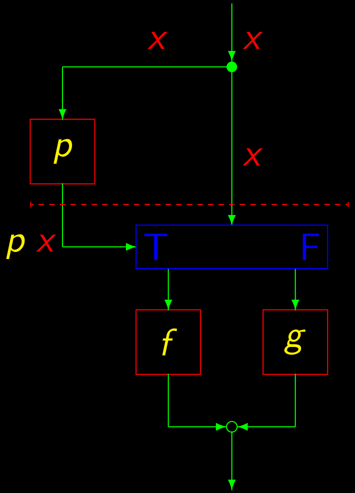
α determines α°

if-then-else



if-then-else

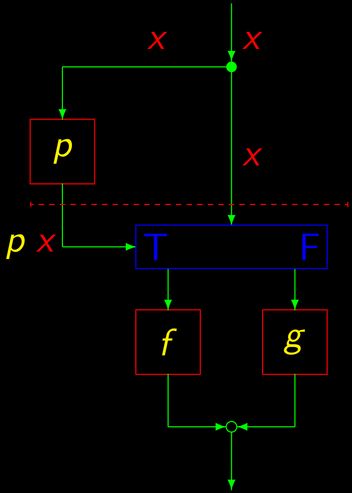
$$A \xrightarrow{\langle p,\, id \rangle} 2 \times A$$



if-then-else

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

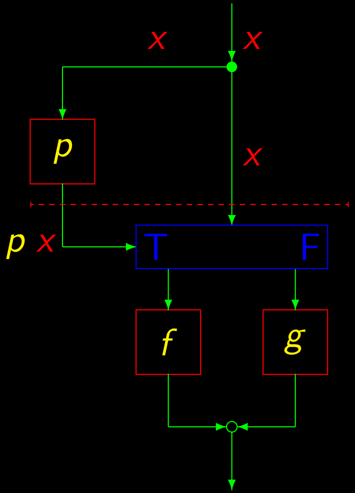
$$A \xrightarrow{f} B \qquad A \xrightarrow{g} B$$



if-then-else

$$
\begin{tikzcd}
A \arrow[d, "\langle p, id \rangle"] \\
2 \times A
\end{tikzcd}
$$

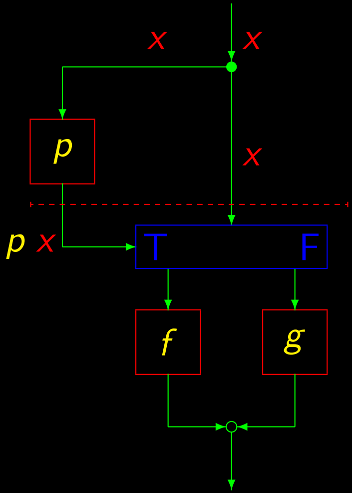
$$
\begin{tikzcd}
A \arrow[r, "i_1"] \arrow[d, "f"'] & A + A \arrow[d, dashed, "f + g"] & A \arrow[l, "i_2"'] \arrow[d, "g"] \\
B \arrow[r, "i_1"'] & B + B & B \arrow[l, "i_2"]
\end{tikzcd}
$$



if-then-else

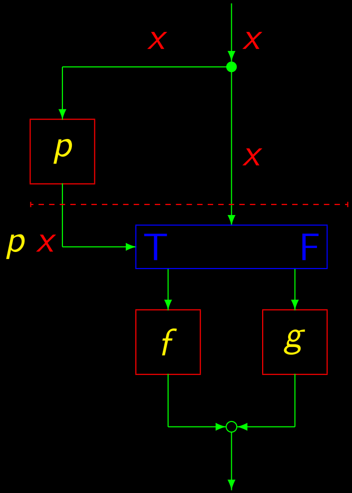
$$
\begin{tikzcd}
A \arrow[d, "\langle p,\, id \rangle"] \\
2 \times A
\end{tikzcd}
$$

$$
\begin{tikzcd}
A \arrow[r, "i_1"] \arrow[d, "f"'] & A + A \arrow[d, "f + g"] & A \arrow[l, "i_2"'] \arrow[d, "g"] \\
B \arrow[r, "i_1"'] & B + B \arrow[d, "join"] & B \arrow[l, "i_2"] \\
& B &
\end{tikzcd}
$$



if-then-else

$$
\begin{tikzcd}
 & A \arrow[d, "\langle p, id\rangle"] & \\
 & 2\times A \arrow[d, "\alpha^\circ"] & \\
A \arrow[r, "i_1"] \arrow[d, "f"'] & A+A \arrow[d, "f+g"] & A \arrow[l, "i_2"'] \arrow[d, "g"] \\
B \arrow[r, "i_1"'] & B+B \arrow[d, "join"] & B \arrow[l, "i_2"] \\
 & B &
\end{tikzcd}
$$



McCarthy Conditional $p \to f\ ,\ g$

From the diagram:

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

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

From the diagram:

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

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

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

From the diagram:

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

Since

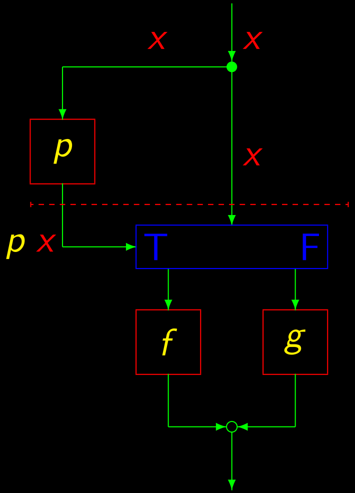
$$join \cdot (f + g) = [id, id] \cdot (f + g) = [f, g]$$

we have:

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

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

$$\begin{tikzcd}
 & A \arrow[d, "p?"] & \\
A \arrow[r, "i_1"] \arrow[dr, "f"'] & A + A \arrow[d, "{[f\,,\,g]}"] & A \arrow[l, "i_2"'] \arrow[dl, "g"] \\
 & B &
\end{tikzcd}$$



McCarthy Conditional $p \to f\ ,\ g$

p-guard:

$$\begin{tikzcd} A \arrow[r, "\langle p, id \rangle"] \arrow[rr, bend right, "p?"'] & 2 \times A \arrow[r, "\alpha^\circ"] & A + A \end{tikzcd}$$

Conditional:

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

John McCarthy (1927-2011)

Turing Award

Founding father of **Artificial Intelligence**

PhD in mathematics (Princeton, 1951)

Inventor of **LISP**

Cálculo de Programas

Class T05

Conditionals pointwise

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

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

Conditionals 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

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

Conditionals 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?

Conditionals 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"?

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

Between conditional and composition

Any "chemistry"?

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

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

“Calculemus”

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

“Calculemus”

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

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

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

"Calculemus"

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

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

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

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

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

“Calculemus”

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

Conditional — 1st fusion law

Altogether:

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

Conditional — 1st fusion law

Altogether:

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

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

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

Conditional — 2nd fusion law

Intuitively, one has:

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

Conditional — 2nd fusion law

Intuitively, one has:

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

Intuitively?

Conditional — 2nd fusion law

Intuitively, one has:

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

Intuitively?

Proof ?

2nd fusion law of conditionals

It can be shown that

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

follows from

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

2nd fusion law of conditionals

It can be shown that

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

follows from

$$\boxed{p? \cdot f \;=\; (f + f) \cdot (p \cdot f)?}$$

So we are left with the **proof** of the equality above...

Let us do it now

Prove

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

knowing:

$$\begin{tikzcd} A \arrow[r, "\langle p, id \rangle"] \arrow[rr, bend right, "p?"'] & 2 \times A \arrow[r, "\alpha^\circ"] & A + A \end{tikzcd}$$

Recall

$$\begin{tikzcd} 2 \times A \arrow[rr, bend left, "\alpha^{\circ}"] \arrow[rr, phantom, "\cong"] & & A + A \arrow[ll, bend left, "\alpha"] \end{tikzcd}$$

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

$$\alpha^{\circ} = \cdots ?$$

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

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

Let us try it

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

Let us try it

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

Let us try it

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

Recall

$$\begin{tikzcd} 2 \times A \arrow[rr, bend left, "\alpha^{\circ}"] \arrow[rr, phantom, "\cong"] & & A + A \arrow[ll, bend left, "\alpha"] \end{tikzcd}$$

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

$$\alpha^{\circ} = \cdots ?$$

Natural-α°

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

Natural-α°

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

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

Natural-α°

$$\begin{tikzcd}
A + A \arrow[d, "\text{''}A + A\text{''}"'] & 2 \times A \arrow[l, "\alpha^\circ"'] \arrow[d, "\text{''}2 \times A\text{''}"] \\
A' + A' & 2 \times A' \arrow[l, "\alpha^\circ"]
\end{tikzcd}$$

Natural-α°

$$
\begin{tikzcd}
A + A \arrow[d, "f + f"'] & & 2 \times A \arrow[ll, "\alpha^\circ"'] \arrow[d, "id \times f"] \\
A' + A' & & 2 \times A' \arrow[ll, "\alpha^\circ"]
\end{tikzcd}
$$

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

... 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!

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

Higher-order functions
(and why they matter!)

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

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

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

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

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

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

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

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

Curry | uncurry

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

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

Curry | uncurry

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

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

Each other inverses?

Curry | uncurry

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

Curry | uncurry

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

`curry`

Curry | uncurry

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

$$\texttt{curry}\ f$$

Curry | uncurry

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

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

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

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

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

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

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

$\square$

Curry | uncurry

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

Curry | uncurry

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

Curry | uncurry

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

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

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

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

$$\mathit{id}\ x = x$$

Curry | uncurry

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

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

Curry | uncurry

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

$\square$

Isomorphism

$$\begin{tikzcd}
(a, b) \to c \arrow[rr, bend left, "\texttt{curry}"] \arrow[r, phantom, "\cong"] & {} & a \to b \to c \arrow[ll, bend left, "\texttt{uncurry}"]
\end{tikzcd}$$

The type B^A

The type B^A

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

NB: functions that map A to B

Isomorphism

$$\begin{tikzcd} A \times B \to C \arrow[rr, bend left, "\texttt{curry}"] \arrow[rr, phantom, "\cong"] & & A \to (B \to C) \arrow[ll, bend left, "\texttt{uncurry}"] \end{tikzcd}$$

Isomorphism

$$
\begin{tikzcd}
A \times B \to C \arrow[rr, bend left, "\texttt{curry}"] & \cong & A \to C^B \arrow[ll, bend left, "\texttt{uncurry}"]
\end{tikzcd}
$$

Isomorphism

$$\begin{tikzcd} C^{A\times B} \arrow[rr, bend left, "\texttt{curry}"] \arrow[rr, phantom, "\cong"] & & \left(C^{B}\right)^{A} \arrow[ll, bend left, "\texttt{uncurry}"] \end{tikzcd}$$

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

Curry | uncurry

$$\begin{tikzcd}
A \times B \to C \arrow[rr, bend left, "\texttt{curry}"] \arrow[rr, phantom, "\cong"] & & A \to C^B \arrow[ll, bend left, "\texttt{uncurry}"]
\end{tikzcd}$$

Curry | uncurry

f

$$
\begin{tikzcd}
A \times B \to C \arrow[rr, bend left, "\texttt{curry}"] \arrow[r, phantom, "\cong"] & & A \to C^B \arrow[ll, bend left, "\texttt{uncurry}"]
\end{tikzcd}
$$

Curry | uncurry

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

$$\begin{tikzcd} A \times B \to C \arrow[rr, bend left, "\texttt{curry}"] \arrow[r, phantom, "\cong"] & & A \to C^B \arrow[ll, bend left, "\texttt{uncurry}"] \end{tikzcd}$$

Curry | uncurry

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

$$\begin{tikzcd} A \times B \to C \arrow[rr, bend left, "\texttt{curry}"] \arrow[rr, phantom, "\cong"] & & A \to C^B \arrow[ll, bend left, "\texttt{uncurry}"] \end{tikzcd}$$

k

Curry | uncurry

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

$$\begin{tikzcd} A \times B \to C \arrow[rr, bend left, "\texttt{curry}"] \arrow[rr, phantom, "\cong"] & & A \to C^B \arrow[ll, bend left, "\texttt{uncurry}"] \end{tikzcd}$$

$$f \xleftarrow{\texttt{uncurry}} \!\!| \; k$$

Curry | uncurry

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

$$\begin{tikzcd}
A \times B \to C \arrow[rr, bend left, "\texttt{curry}"] \arrow[rr, phantom, "\cong"] & & A \to C^B \arrow[ll, bend left, "\texttt{uncurry}"]
\end{tikzcd}$$

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

Curry | uncurry

$$\begin{tikzcd} A \times B \to C \arrow[rr, bend left, "\texttt{curry}"] & \cong & A \to C^B \arrow[ll, bend left, "\texttt{uncurry}"] \end{tikzcd}$$

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

Curry | uncurry

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

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

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

Curry | uncurry

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

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

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

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

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

Curry | uncurry

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

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

Curry | uncurry

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

$\Leftrightarrow$ { isomorphism (last slide) }

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

$\Leftrightarrow$ { add variables }

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

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

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

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

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

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

Curry | uncurry

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

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

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

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

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

Universal property

$$\begin{tikzcd} A \times B \to C \arrow[rr, bend left, "\texttt{curry}"] & \cong & A \to C^B \arrow[ll, bend left, "\texttt{uncurry}"] \end{tikzcd}$$

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

Universal property

$$\begin{tikzcd}
A \times B \to C \arrow[rr, bend left, "\overline{(_)}"] \arrow[rr, phantom, "\cong"] & & A \to C^B \arrow[ll, bend left, "\widehat{(_)}"]
\end{tikzcd}$$

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

Diagrams

$$\begin{tikzcd} & C \\ A \times B \arrow[ur, "f"'] & \end{tikzcd}$$

Diagrams

$$
\begin{tikzcd}
C^B & & C \\
A \arrow[u, "k = \overline{f}"] & A \times B \arrow[ur, "f"'] &
\end{tikzcd}
$$

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

Diagrams

$$
\begin{tikzcd}
C^B \\
A \arrow[u, "k = \overline{f}"]
\end{tikzcd}
\qquad
\begin{tikzcd}
C^B & & C \\
A \times B \arrow[u, "k"] \arrow[urr, "f"'] & &
\end{tikzcd}
$$

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

Diagrams

$$\begin{tikzcd} C^B \\ A \arrow[u, "k = \overline{f}"] \end{tikzcd} \qquad \begin{tikzcd} C^B & B & & C \\ A \arrow[u, "k"] & \times & B \arrow[ul, "id"'] \arrow[ur, "f"'] \end{tikzcd}$$

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

Universal property

$$
\begin{tikzcd}
C^B & C^B \times B \arrow[r, phantom] & C \\
A \arrow[u, "k = \overline{f}"] & A \times B \arrow[u, "k \times id"] \arrow[ur, "f"'] &
\end{tikzcd}
$$

Universal property

$$\begin{tikzcd} C^B \\ A \arrow[u, "k = \overline{f}"] \end{tikzcd} \qquad \begin{tikzcd} C^B \times B \arrow[r, "ap"] & C \\ A \times B \arrow[u, "k \times id"] \arrow[ur, "f"'] & \end{tikzcd}$$

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

Reflexion $(k := id)$

$$
\begin{tikzcd}
C^B \\
C^B \arrow[u, "id = \overline{f}"]
\end{tikzcd}
\qquad
\begin{tikzcd}
C^B \times B \arrow[r, "ap"] & C \\
C^B \times B \arrow[u, "id \times id"] \arrow[ur, "f"'] &
\end{tikzcd}
$$

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

Fusion

$$\begin{tikzcd} C^B \\ A \arrow[u, "\overline{f}"] \end{tikzcd} \qquad \begin{tikzcd} C^B \times B \arrow[r, "ap"] & C \\ A \times B \arrow[u, "\overline{f} \times id"] \arrow[ur, "f"'] & \end{tikzcd}$$

Fusion

$$
\begin{tikzcd}
C^B \\
A \arrow[u, "\overline{f}"] \\
D \arrow[u, "g"]
\end{tikzcd}
\qquad
\begin{tikzcd}
C^B \times B \arrow[r, "ap"] & C \\
A \times B \arrow[u, "\overline{f} \times id"] \arrow[ur, "f"'] &
\end{tikzcd}
$$

Fusion

$$\begin{tikzcd} C^B \\ A \arrow[u, "\overline{f}"] \\ D \arrow[u, "g"] \end{tikzcd} \qquad \begin{tikzcd} C^B \times B \arrow[r, "ap"] & C \\ A \times B \arrow[u, "\overline{f} \times id"] \arrow[ur, "f"'] \\ D \times B \arrow[u, "g \times id"] \end{tikzcd}$$

Fusion

$$
\begin{tikzcd}
C^B & & C^B \times B \arrow[r, "ap"] & C \\
A \arrow[u, "\overline{f}"] & & A \times B \arrow[u, "\overline{f} \times id"] \arrow[ur, "f"] & \\
D \arrow[u, "g"] & & D \times B \arrow[u, "g \times id"] \arrow[uur, "f \cdot (g \times id)"'] &
\end{tikzcd}
$$

Fusion

$$
\begin{tikzcd}
C^B \\
\\
D \arrow[uu, "\overline{f}\cdot g"]
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
C^B \times B \arrow[r, "ap"] & C \\
A \times B \arrow[u, "\overline{f}\times id"] \arrow[ur, "f"'] & \\
D \times B \arrow[u, "g\times id"] \arrow[uur, "f\cdot(g\times id)"'] &
\end{tikzcd}
$$

Fusion

$$
\begin{tikzcd}
C^B & C^B \times B \arrow[r, "ap"] & C \\
 & A \times B \arrow[u, "\overline{f} \times id"] \arrow[ur, "f"] & \\
D \arrow[uu, "\overline{f \cdot (g \times id)}"] & D \times B \arrow[u, "g \times id"] \arrow[uur, "f \cdot (g \times id)"'] &
\end{tikzcd}
$$

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

Finally

$$\begin{tikzcd} C^B \\ A \arrow[u, "\overline{f}"] \end{tikzcd} \qquad \begin{tikzcd} C^B \times B \arrow[r, "ap"] & C \\ A \times B \arrow[u, "\overline{f} \times id"] \arrow[ur, "f"'] & \end{tikzcd}$$

Finally

$$
\begin{tikzcd}
 & & & D \\
C^B & & C^B \times B \arrow[r, "ap"] \arrow[ur, "g \cdot ap"] & C \arrow[u, "g"'] \\
A \arrow[u, "\overline{f}"] & & A \times B \arrow[u, "\overline{f} \times id"] \arrow[ur, "f"'] &
\end{tikzcd}
$$

Finally

$$
\begin{tikzcd}
D^B \\
C^B \arrow[u, "g^B"] \\
A \arrow[u, "\overline{f}"]
\end{tikzcd}
\qquad
\begin{tikzcd}
D^B \times B & D \\
C^B \times B \arrow[u, "g^B \times id"] \arrow[r, "ap"] \arrow[ur, "g \cdot ap"] & C \arrow[u, "g"'] \\
A \times B \arrow[u, "\overline{f} \times id"] \arrow[ur, "f"'] &
\end{tikzcd}
$$

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

Finally

$$
\begin{tikzcd}
D^B \\
C^B \arrow[u, "g^B"] \\
A \arrow[u, "\overline{f}"]
\end{tikzcd}
\qquad
\begin{tikzcd}
D^B \times B \arrow[r, "ap"] & D \\
C^B \times B \arrow[u, "g^B \times id"] \arrow[r, "ap"] \arrow[ur, "g \cdot ap"] & C \arrow[u, "g"'] \\
A \times B \arrow[u, "\overline{f} \times id"] \arrow[ur, "f"'] &
\end{tikzcd}
$$

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

Finally

$$
\begin{tikzcd}
D^B & & D^B \times B \arrow[r, "ap"] & D \\
C^B \arrow[u, "g^B"] & & C^B \times B \arrow[u, "g^B \times id"] \arrow[r, "ap"] \arrow[ur, "g \cdot ap"] & C \arrow[u, "g"'] \\
A \arrow[u, "\overline{f}"] & & A \times B \arrow[u, "\overline{f} \times id"] \arrow[ur, "f"'] &
\end{tikzcd}
$$

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

$$g^B\ f = g \cdot f$$

Summary

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

$$\begin{tikzcd} A \times B \to C \arrow[r, bend left, "\texttt{curry}"] & A \to C^B \arrow[l, bend left, "\texttt{uncurry}"] \end{tikzcd}$$

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

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

Summary

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

$$\begin{tikzcd} A \times B \to C \arrow[r, bend left, "\texttt{curry}"] \arrow[r, phantom, "\cong"] & A \to C^B \arrow[l, bend left, "\texttt{uncurry}"] \end{tikzcd}$$

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

Abbreviations:

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

Summary

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

$$\begin{tikzcd} A \times B \to C \arrow[r, bend left, "\texttt{curry}"] \arrow[r, phantom, "\cong"] & A \to C^B \arrow[l, bend left, "\texttt{uncurry}"] \end{tikzcd}$$

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

Abbreviations:

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

Summary

$$\begin{tikzcd} f \arrow[rr, mapsto, "\texttt{curry}"] & & k \end{tikzcd}$$

$$\begin{tikzcd} A \times B \to C \arrow[rr, bend left, "\texttt{curry}"] \arrow[r, phantom, "\cong"] & & A \to C^B \arrow[ll, bend left, "\texttt{uncurry}"] \end{tikzcd}$$

$$\begin{tikzcd} f & & k \arrow[ll, mapsto, "\texttt{uncurry}"'] \end{tikzcd}$$

Abbreviations:

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

Isomorphisms:

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

More isomorphisms

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

$$\begin{tikzcd} A^1 \arrow[r, bend left, "g"] \arrow[r, phantom, "\cong"] & A \arrow[l, bend left, "f"] \end{tikzcd} \qquad \begin{tikzcd} 1^A \arrow[r, bend left, "g"] \arrow[r, phantom, "\cong"] & 1 \arrow[l, bend left, "f"] \end{tikzcd} \qquad \begin{tikzcd} A^0 \arrow[r, bend left, "g"] \arrow[r, phantom, "\cong"] & 1 \arrow[l, bend left, "f"] \end{tikzcd}$$

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

(Even more) isomorphisms

$$\begin{tikzcd} A \times A \arrow[rr, bend left, "g"] \arrow[rr, phantom, "\cong"] & & A^2 \arrow[ll, bend left, "f"] \end{tikzcd}$$

$$f\ k = (k\ F, k\ T)$$

(Even more) isomorphisms

$$\begin{tikzcd} A \times A \arrow[r, bend left, "g"] \arrow[r, phantom, "\cong"] & A^2 \arrow[l, bend left, "f"] \end{tikzcd}$$

$$f\ k = (k\ F, k\ T)$$

$$\begin{cases} g\ (a_1, a_2)\ F = a_1 \\ g\ (a_1, a_2)\ T = a_2 \end{cases}$$

cf. $a \times a = a^2$

(Even more) isomorphisms

(Even more) isomorphisms

$$c^{a+b} = c^a \times c^b$$

(Even more) isomorphisms

$$c^{a+b} = c^a \times c^b \qquad (a \times b)^c = a^c \times b^c$$

(Even more) isomorphisms

$$c^{a+b} = c^a \times c^b \qquad (a \times b)^c = a^c \times b^c$$

$$\begin{tikzcd} C^{A+B} \arrow[r, bend left, "g"] \arrow[r, phantom, "\cong"] & C^A \times C^B \arrow[l, bend left, "f"] \end{tikzcd}$$

(Even more) isomorphisms

$$c^{a+b} = c^a \times c^b \qquad (a \times b)^c = a^c \times b^c$$

$$\begin{tikzcd} C^{A+B} \arrow[r, bend left, "g"] \arrow[r, phantom, "\cong"] & C^A \times C^B \arrow[l, bend left, "f"] \end{tikzcd} \qquad \begin{tikzcd} (A \times B)^C \arrow[r, bend left, "g"] \arrow[r, phantom, "\cong"] & A^C \times B^C \arrow[l, bend left, "f"] \end{tikzcd}$$

(Even more) isomorphisms

$$c^{a+b} = c^a \times c^b \qquad (a \times b)^c = a^c \times b^c$$

$$\begin{tikzcd} C^{A+B} \arrow[r, bend left, "g"] \arrow[r, phantom, "\cong"] & C^A \times C^B \arrow[l, bend left, "f"] \end{tikzcd} \qquad \begin{tikzcd} (A \times B)^C \arrow[r, bend left, "g"] \arrow[r, phantom, "\cong"] & A^C \times B^C \arrow[l, bend left, "f"] \end{tikzcd}$$

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

(Even more) isomorphisms

$$c^{a+b} = c^a \times c^b \qquad (a \times b)^c = a^c \times b^c$$

$$\begin{tikzcd} C^{A+B} \arrow[r, bend left, "g"] \arrow[r, phantom, "\cong"] & C^A \times C^B \arrow[l, bend left, "f"] \end{tikzcd} \qquad \begin{tikzcd} (A \times B)^C \arrow[r, bend left, "g"] \arrow[r, phantom, "\cong"] & A^C \times B^C \arrow[l, bend left, "f"] \end{tikzcd}$$

$$\begin{cases} f\ (h, k) = [h, k] \\ g\ k = (k \cdot i_1, k \cdot i_2) \end{cases} \qquad \begin{cases} f\ (h, k) = \langle h, k \rangle \\ g\ k = (\pi_1 \cdot k, \pi_2 \cdot k) \end{cases}$$

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

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

$\overline{f}$ higher order

Summary

$f \cdot g$	sequential
$\langle f, g \rangle$, $f \times g$	parallel
$[f, g]$, $f + g$, $p \rightarrow f\ ,\ g$	alternation
$\overline{f}$	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 (coming up soon)

Precedence rules

In order to save parentheses in pointfree expressions:

- **unary** operators take precedence over binary ones ;
- **composition** binds tighter than any other binary operator;
- **product** $(f \times g)$ binds tighter than **coproduct** $(f + g)$.

Analogy...

Our small set of **combinators** resembles

RISC — *Reduced instruction-set computer*

an idea championed by Prof. David Patterson (UCB, Google)

Analogy...

[... and since you are here, don't miss reading Prof. Patterson's "16 life lessons" (+ "9 magic words"), a recent post in CACM, 10-Oct.]

Cálculo de Programas

Class T06

Part II

Where do programs come from?

What is a program?

Where do programs come from?

What is an **algorithm**?

Where do programs come from?

What **is** an **algorithm**?

> *"Detailed* **instructions** *defining a* **computational process** *(which is then said to be* **algorithmic***), which begins with an arbitrary* **input** *(...), and with instructions aimed at obtaining a result (or* **output***) which is fully determined by the input."*

(`https://encyclopediaofmath.org/index.php?title=Algorithm`)

Where do programs come from?

The algorithms we know were born from what?

Where do programs come from?

The algorithms we know were born from what?

For instance, **quicksort**?

(See video 05b, t=3.25)

Where do programs come from?

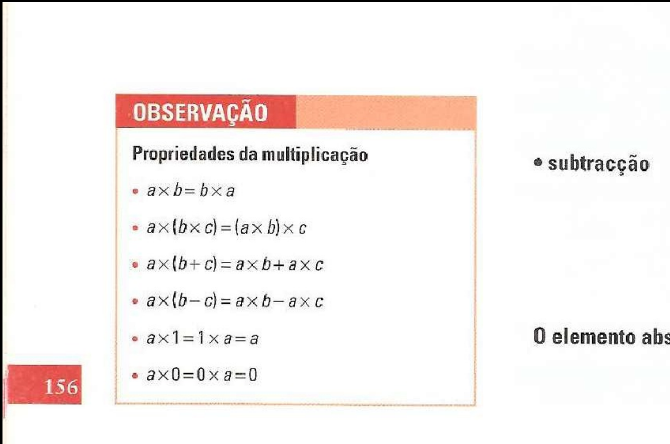
What is an **algorithm**?

> *"(...) For instance, the rules taught in elementary schools for column-wise* **addition**, **subtraction**, **multiplication** *and* **division** *are algorithms; (...)*

(`https://encyclopediaofmath.org/index.php?title=Algorithm`)

Where do programs come from?

From a Y7 maths textbook:



OBSERVAÇÃO

Propriedades da multiplicação

- $a \times b = b \times a$
- $a \times (b \times c) = (a \times b) \times c$
- $a \times (b + c) = a \times b + a \times c$
- $a \times (b - c) = a \times b - a \times c$
- $a \times 1 = 1 \times a = a$
- $a \times 0 = 0 \times a = 0$

• subtracção

O elemento abs

156

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?

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

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?

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

What **is** a program?

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

What **is** a program?

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

What **is** a program?

$$\begin{cases} (a\times)\cdot \underline{0} = \underline{0} \\[2em] (a\times)\cdot \mathit{succ} = (a+)\cdot(a\times) \end{cases}$$

What is a program?

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

What is a program?

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

What **is** a program?

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

What **is** a program?

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

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

$$\begin{tikzcd} ? \arrow[d, "(a\times)"'] \\ ? \end{tikzcd}$$

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

$$\begin{tikzcd} ? \arrow[d, "(a\times)"'] & ? \arrow[l, "{[\underline{0}, succ]}"'] \\ ? & \end{tikzcd}$$

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

$$\begin{tikzcd}
? \arrow[d, "(a\times)"'] & ? \arrow[l, "{[\underline{0}, succ]}"'] \\
? & ? \arrow[l, "{[\underline{0}, (a+)]}"]
\end{tikzcd}$$

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

$$\begin{tikzcd}
? \arrow[d, "(a\times)"'] & ? \arrow[l, "{[\underline{0}, succ]}"'] \arrow[d, "id + (a\times)"] \\
? & ? \arrow[l, "{[\underline{0}, (a+)]}"]
\end{tikzcd}$$

A program is a "rectangle" 

$$
\begin{tikzcd}
\mathbb{N}_0 \arrow[d, "(a\times)"'] & ? \arrow[l, "{[\underline{0}, succ]}"'] \arrow[d, "id + (a\times)"] \\
\mathbb{N}_0 & ? \arrow[l, "{[\underline{0}, (a+)]}"]
\end{tikzcd}
$$

A program is a "rectangle"

$$
\begin{tikzcd}
\mathbb{N}_0 \arrow[d, "(a\times)"'] & ? + ? \arrow[l, "{[\underline{0}, succ]}"'] \arrow[d, "id + (a\times)"] \\
\mathbb{N}_0 & ? + ? \arrow[l, "{[\underline{0}, (a+)]}"]
\end{tikzcd}
$$

A program is a "rectangle"

$$
\begin{tikzcd}
\mathbb{N}_0 \arrow[d, "(a\times)"'] & ? + \mathbb{N}_0 \arrow[l, "{[\underline{0},\, succ]}"'] \arrow[d, "id + (a\times)"] \\
\mathbb{N}_0 & ? + \mathbb{N}_0 \arrow[l, "{[\underline{0},\, (a+)]}"]
\end{tikzcd}
$$

A program is a "rectangle"

$$
\begin{tikzcd}
\mathbb{N}_0 \arrow[d, "(a\times)"'] & & A + \mathbb{N}_0 \arrow[ll, "{[\underline{0}, succ]}"'] \arrow[d, "id + (a\times)"] \\
\mathbb{N}_0 & & A + \mathbb{N}_0 \arrow[ll, "{[\underline{0}, (a+)]}"]
\end{tikzcd}
$$

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

$$
\begin{tikzcd}
\mathbb{N}_0 \arrow[r, "{[\underline{0}, succ]^{\circ}}"] \arrow[d, "(a\times)"'] & A + \mathbb{N}_0 \arrow[d, "id + (a\times)"] \\
\mathbb{N}_0 & A + \mathbb{N}_0 \arrow[l, "{[\underline{0}, (a+)]}"]
\end{tikzcd}
$$

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

$$
\begin{tikzcd}
\mathbb{N}_0 \arrow[d, "(a\times)"'] & A + \mathbb{N}_0 \arrow[l, "{[\underline{0}, succ]}"'] \arrow[d, "id + (a\times)"] \\
\mathbb{N}_0 & A + \mathbb{N}_0 \arrow[l, "{[\underline{0}, (a+)]}"]
\end{tikzcd}
$$

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

$$
\begin{tikzcd}
\mathbb{N}_0 \arrow[r, "{[\underline{0}, succ]^{\circ}}"] \arrow[d, "(a\times)"'] & A + \mathbb{N}_0 \arrow[d, "id + (a\times)"] \\
\mathbb{N}_0 & A + \mathbb{N}_0 \arrow[l, "{[\underline{0}, (a+)]}"]
\end{tikzcd}
$$

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

$$\begin{tikzcd}
\mathbb{N}_0 \arrow[rr, "{[\underline{0}, succ]^\circ}"] \arrow[d, "(a\times)"'] & & A + \mathbb{N}_0 \arrow[d, "id + (a\times)"] \\
\mathbb{N}_0 & & A + \mathbb{N}_0 \arrow[ll, "{[\underline{0}, (a+)]}"]
\end{tikzcd}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\Leftrightarrow \qquad \{ \text{ introduce variables } \}$$

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

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?

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

$$\Leftrightarrow \quad \{ \text{ introduce variables } \}$$

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

$$\Leftrightarrow \quad \{ \text{ simplificar } \}$$

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

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

?

Problem

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

$$\begin{tikzcd} \mathbb{N}_0 \arrow[rr, bend left] \arrow[rr, phantom, "?"] & & A + \mathbb{N}_0 \arrow[ll, bend left, "\mathbf{in} = [\underline{0}, succ]"] \end{tikzcd}$$

?

Solution

$$\left\{\begin{array}{l} \mathbf{out}\ 0 = i_1\ () \\ \mathbf{out}\ (x+1) = i_2\ x \end{array}\right.$$

$$\begin{tikzcd}[column sep=huge] \mathbb{N}_0 \arrow[r, bend left] \arrow[r, phantom, "\cong"] & 1 + \mathbb{N}_0 \arrow[l, bend left, "\mathbf{in}=[\underline{0}, succ]"] \end{tikzcd}$$

Solution

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

$$\begin{tikzcd}[column sep=huge, row sep=huge]
\mathbb{N}_0 \arrow[r, bend left] \arrow[d, "(a\times)"'] \arrow[r, phantom, "\cong"] & 1 + \mathbb{N}_0 \arrow[l, bend left, "\textbf{in}=[\underline{0}, succ]"] \arrow[d, "id+(a\times)"] \\
\mathbb{N}_0 & 1 + \mathbb{N}_0 \arrow[l, bend left, "{[\underline{0}, (a+)]}"]
\end{tikzcd}$$

Diagram

$$\begin{tikzcd}
\mathbb{N}_0 \arrow[rr, bend left, "\mathsf{out}"] \arrow[rr, phantom, "\cong"] \arrow[d, "(a\times)"'] & & 1 + \mathbb{N}_0 \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "id+(a\times)"] \\
\mathbb{N}_0 & & 1 + \mathbb{N}_0 \arrow[ll, bend left, "{[\underline{0}, a+]}"]
\end{tikzcd}$$

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

Renaming

$$
\begin{tikzcd}
\mathbb{N}_0 \arrow[r, bend left, "\mathsf{out}"] \arrow[r, phantom, "\cong"] \arrow[d, "k"'] & 1 + \mathbb{N}_0 \arrow[l, bend left, "\mathsf{in}"] \arrow[d, "id + k"] \\
\mathbb{N}_0 & 1 + \mathbb{N}_0 \arrow[l, bend left, "{[\underline{0}, a+]}"]
\end{tikzcd}
$$

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

Variations in diagram

$$
\begin{tikzcd}
\mathbb{N}_0 \arrow[rr, bend left, "\mathsf{out}"] \arrow[d, "k"'] \arrow[rr, phantom, "\cong"] & & 1 + \mathbb{N}_0 \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "id+k"] \\
\mathbb{N}_0 & & 1 + \mathbb{N}_0 \arrow[ll, bend left, "{[\underline{1},(a\times)]}"]
\end{tikzcd}
$$

Variations in diagram

$$
\begin{tikzcd}
\mathbb{N}_0 \arrow[rr, bend left, "\mathsf{out}"] \arrow[d, "k"'] \arrow[rr, phantom, "\cong"] & & 1 + \mathbb{N}_0 \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "id+k"] \\
\mathbb{N}_0 & & 1 + \mathbb{N}_0 \arrow[ll, bend left, "{[\underline{1}, (a\times)]}"]
\end{tikzcd}
$$

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

Variations in diagram

$$\begin{tikzcd}[column sep=huge, row sep=huge]
\mathbb{N}_0 \arrow[r, bend left, "\mathsf{out}"] \arrow[r, phantom, "\cong"] \arrow[d, "k"'] & 1 + \mathbb{N}_0 \arrow[l, bend left, "\mathsf{in}"] \arrow[d, "id + k"] \\
\mathbb{N}_0 & 1 + \mathbb{N}_0 \arrow[l, bend left, "{[\underline{1}, (a\times)]}"]
\end{tikzcd}$$

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

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

Generalizing the diagram

$$
\begin{tikzcd}
\mathbb{N}_0 \arrow[rr, bend left, "\mathsf{out}"] \arrow[rr, phantom, "\cong"] \arrow[d, "k\ g"'] & & 1 + \mathbb{N}_0 \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "id + k\ g"] \\
A & & 1 + A \arrow[ll, bend left, "g"]
\end{tikzcd}
$$

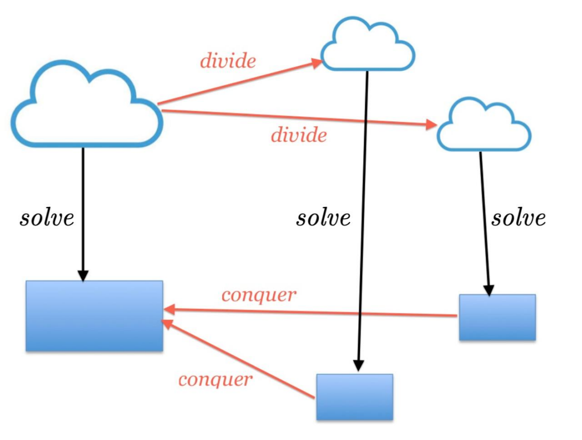
$$k\ g = g \cdot (id + k\ g) \cdot \mathsf{out}$$

Generalizing the diagram

$$
\begin{tikzcd}
\mathbb{N}_0 \arrow[rr, bend left, "\mathsf{out}"] \arrow[d, "(\!|\, g \,|\!)"'] \arrow[rr, phantom, "\cong"] & & 1 + \mathbb{N}_0 \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "id + (\!|\, g \,|\!)"] \\
A & & 1 + A \arrow[ll, bend left, "g"]
\end{tikzcd}
$$

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

Anticipation of what is to come...



Catamorphism

$(\!| g |\!)$

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

Catamorphism

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

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

"Downwards transformation"

Definition

$$\begin{tikzcd}
\mathbb{N}_0 \arrow[r, "\mathbf{out}"] \arrow[d, "(\!| g |\!)"'] & 1 + \mathbb{N}_0 \arrow[d, "id + (\!| g |\!)"] \\
A & 1 + A \arrow[l, "g"]
\end{tikzcd}$$

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

Property

$$\begin{tikzcd}
\mathbb{N}_0 \arrow[d, "(\!| g |\!)"'] & 1 + \mathbb{N}_0 \arrow[l, "\mathsf{in}"'] \arrow[d, "id + (\!| g |\!)"] \\
A & 1 + A \arrow[l, "g"]
\end{tikzcd}$$

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

Property

$$\begin{tikzcd}
\mathbb{N}_0 \arrow[d, "(\!| g |\!)"'] & 1 + \mathbb{N}_0 \arrow[l, "\mathbf{in}"'] \arrow[d, "id + (\!| g |\!)"] \\
A & 1 + A \arrow[l, "g"]
\end{tikzcd}$$

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

Uniqueness

$$\begin{tikzcd} \mathbb{N}_0 \arrow[d, "(\!| g |\!)"'] & 1 + \mathbb{N}_0 \arrow[l, "\mathbf{in}"'] \arrow[d, "id + (\!| g |\!)"] \\ A & 1 + A \arrow[l, "g"] \end{tikzcd}$$

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

Universal property

$$\begin{tikzcd} \mathbb{N}_0 \arrow[d, "k"'] & 1 + \mathbb{N}_0 \arrow[l, "\mathbf{in}"'] \arrow[d, "id+k"] \\ A & 1 + A \arrow[l, "g"] \end{tikzcd}$$

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

Universal-$\times$:

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

Universal-$\times$:

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

Universal-$+$:

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

Universal-$\times$:

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

Universal-$+$:

$$k = [f, g] \;\Leftrightarrow\; \left\{ \begin{array}{l} k \cdot i_1 = f \\ k \cdot i_2 = g \end{array} \right.$$

Universal-expo:

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

Universal-$\times$:

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

Universal-$+$:

$$k = [f, g] \;\Leftrightarrow\; \left\{ \begin{array}{l} k \cdot i_1 = f \\ k \cdot i_2 = g \end{array} \right.$$

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 ($\mathbb{N}_0$)

$$\begin{tikzcd} \mathbb{N}_0 \arrow[d, "k"'] & & 1 + \mathbb{N}_0 \arrow[ll, "\mathbf{in}"'] \arrow[d, "id+k"] \\ A & & 1 + A \arrow[ll, "g"] \end{tikzcd}$$

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

Cata-reflexion

To be derived from

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

Cata-reflexion $(k := id)$

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

Cata-reflexion $(k := id)$

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

Cata-reflexion

$$\begin{tikzcd}
\mathbb{N}_0 \arrow[d, "id = (\!|\,\mathbf{in}\,|\!)"'] & 1 + \mathbb{N}_0 \arrow[l, "\mathbf{in} = [\underline{0}, succ]"'] \arrow[d, "id + (\!|\,\mathbf{in}\,|\!)"] \\
\mathbb{N}_0 & 1 + \mathbb{N}_0 \arrow[l, "\mathbf{in} = [\underline{0}, succ]"]
\end{tikzcd}$$

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

Cata-cancellation

Another corollary of

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

for $k := (\!| g |\!)$.

Cata-cancellation $(k := (\!| g |\!))$

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

Cata-cancellation

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

Cata-definition

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

Cata-definition

$$
\begin{tikzcd}
\mathbb{N}_0 \arrow[r, "\mathsf{out}"] \arrow[d, "(\!|\, g \,|\!)"'] & 1 + \mathbb{N}_0 \arrow[d, "id + (\!|\, g \,|\!)"] \\
A & 1 + A \arrow[l, "g"]
\end{tikzcd}
$$

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

Cata-fusion

$$
\begin{tikzcd}
\mathbb{N}_0 \arrow[d, "(\!| g |\!)"'] & 1 + \mathbb{N}_0 \arrow[l, "\mathbf{in} = [\underline{0}, succ]"'] \arrow[d, "id + (\!| g |\!)"] \\
A \arrow[d, "f"'] & 1 + A \arrow[l, "g"] \arrow[d, "id + f"] \\
B & 1 + B \arrow[l, "h"]
\end{tikzcd}
$$

Cata-fusion

$$
\begin{tikzcd}
\mathbb{N}_0 \arrow[d, "(\!| g |\!)"'] \arrow[dd, bend right, "(\!| h |\!)"'] & & 1 + \mathbb{N}_0 \arrow[ll, "\mathbf{in}=[\underline{0}, succ]"'] \arrow[d, "id+(\!| g |\!)"] \\
A \arrow[d, "f"'] & & 1 + A \arrow[ll, "g"] \arrow[d, "id+f"] \\
B & & 1 + B \arrow[ll, "h"]
\end{tikzcd}
$$

Cata-fusion

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

Cata-fusion

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

Cata-fusion

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

Cata-fusion

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

Cata-fusion

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

Cata-fusion

$$
\begin{array}{ll}
& f \cdot g \cdot (id + (\!|\, g \,|\!)) = h \cdot (id \cdot id + f \cdot (\!|\, g \,|\!)) \\
\Leftrightarrow & \quad \{ \text{ functor-}+ \text{ } \} \\
& f \cdot g \cdot (id + (\!|\, g \,|\!)) = h \cdot (id + f) \cdot (id + (\!|\, g \,|\!))
\end{array}
$$

Cata-fusion

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

Cata-fusion

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

Summing up:

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

Cata-fusion

$$
\begin{tikzcd}
\mathbb{N}_0 \arrow[d, "\llparenthesis\, g \,\rrparenthesis"'] \arrow[dd, bend right=60, "\llparenthesis\, h \,\rrparenthesis"'] & 1 + \mathbb{N}_0 \arrow[l, "\mathbf{in}=[\underline{0}, succ]"'] \arrow[d, "id+\llparenthesis\, g \,\rrparenthesis"] \\
A \arrow[d, "f"'] & 1 + A \arrow[l, "g"] \arrow[d, "id+f"] \\
B & 1 + B \arrow[l, "h"]
\end{tikzcd}
$$

What is a program?

What **is** a program?

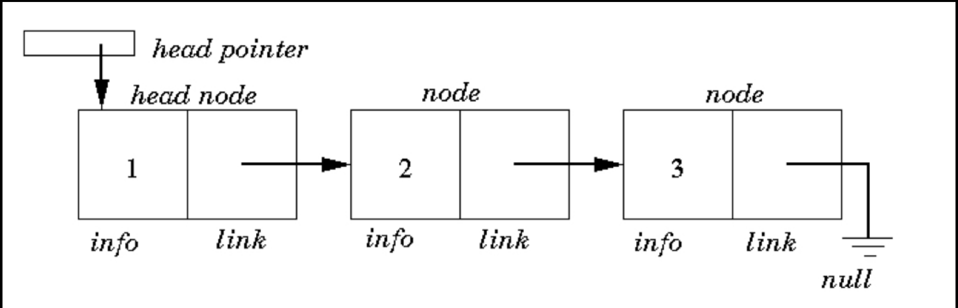
Programs are “rectangles” 

What **is** a program?

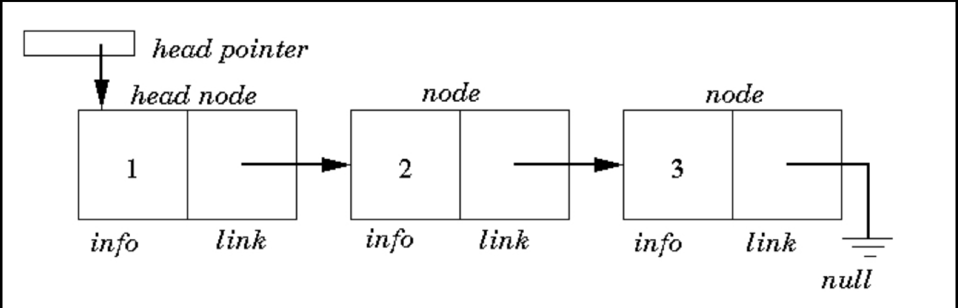
Programs are "rectangles"

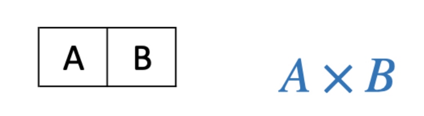
But there is a long way to the general case...

Linked list

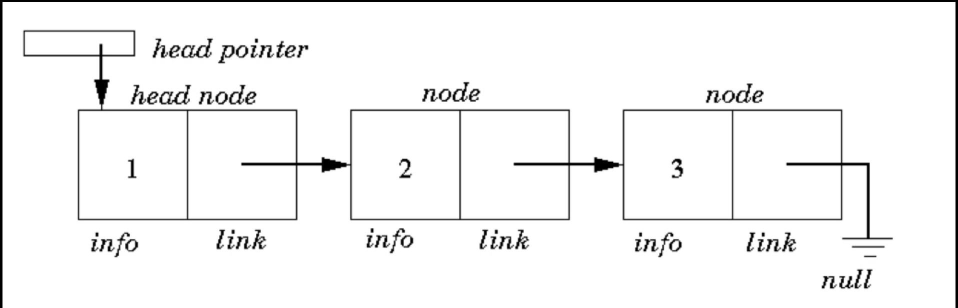

head pointer
head node
node
node
1
2
3
info
link
info
link
info
link
null

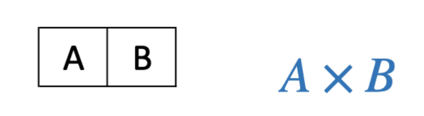
Linked list


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


A
B
$A \times B$

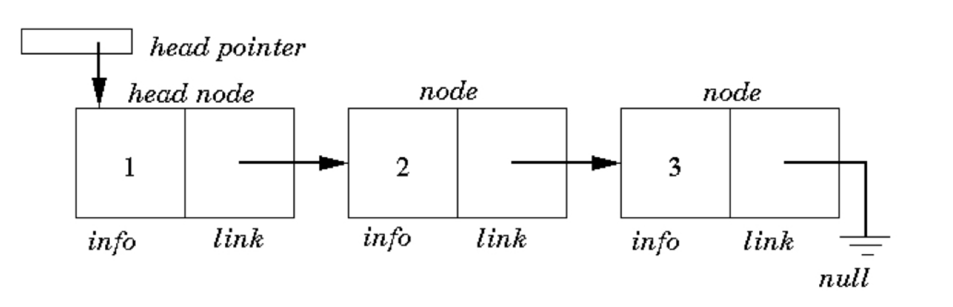
Linked list


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


A
B
$A \times B$


A
$1 + A$

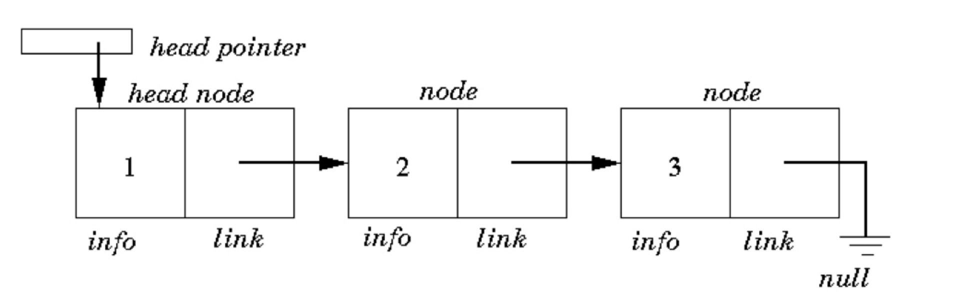
Linked list



“pointer”

$L = 1 + N$

Linked list



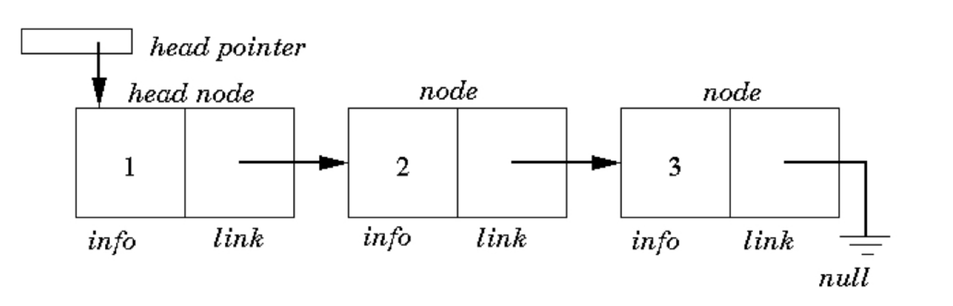
“pointer”

$$L = 1 + N$$

“node”

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

Linked list

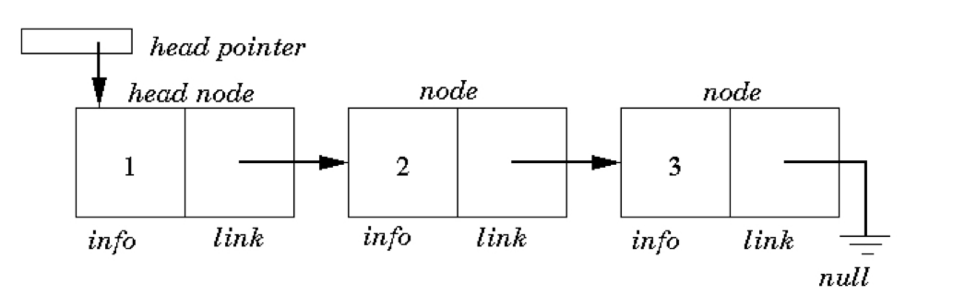


Substitution

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

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

Linked list



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

Linked list

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

Linked list

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

$$\begin{tikzcd} L \arrow[r, bend left, "?"] \arrow[r, phantom, "\cong"] & 1 + \mathbb{N}_0 \times L \arrow[l, bend left, "?"] \end{tikzcd}$$

Cálculo de Programas

Class T07

Length of a list

Recall list **concatenation** $x \mathbin{+\!\!+} y$

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 ('specification'):

Length of a list

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

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

Properties ('specification'):

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

Length of a list

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

$$[\,] \mathbin{+\!\!+} x = x$$
$$[a] \mathbin{+\!\!+} x = a : x$$

Properties ('specification'):

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

Length of a list

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

$$[\,] \mathbin{+\!\!+} x = x$$
$$[a] \mathbin{+\!\!+} x = a : x$$

Properties ('specification'):

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

Length of a list

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

Length of a list

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

$$\mathit{length}\ (a : y) = 1 + \mathit{length}\ y$$

Length of a list

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

Length of a list

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

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

Length of a list

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

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

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

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

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

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

$$\begin{tikzcd} L & \cong & 1 + \mathbb{N}_0 \times L \arrow[ll, bend left, "\mathbf{in} = [\mathit{nil}, \mathit{cons}]"] \end{tikzcd}$$

Length of a list

$$
\begin{aligned}
length \cdot nil &= \underline{0} \\
length \cdot cons &= succ \cdot length \cdot \pi_2
\end{aligned}
$$

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

$$
\begin{tikzcd}
L \arrow[rr, bend left, "\mathbf{out}"] \arrow[rr, phantom, "\cong"] \arrow[d, "length"'] & & 1 + \mathbb{N}_0 \times L \arrow[ll, bend left, "\mathbf{in}"] \arrow[d, "id+id\times length"] \\
\mathbb{N}_0 & & 1 + \mathbb{N}_0 \times \mathbb{N}_0 \arrow[ll, bend left, "{[\underline{0}, succ \cdot \pi_2]}"]
\end{tikzcd}
$$

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

List sum

$$\begin{tikzcd}
L \arrow[d, "sum"'] \arrow[rr, bend left, "\mathbf{out}"] \arrow[rr, phantom, "\cong"] & & 1 + \mathbb{N}_0 \times L \arrow[ll, bend left, "\mathbf{in}"] \arrow[d, "id + id \times sum"] \\
\mathbb{N}_0 & & 1 + \mathbb{N}_0 \times \mathbb{N}_0 \arrow[ll, bend left, "{[\underline{0}, add]}"]
\end{tikzcd}$$

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

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

Linked list

As earlier on:

$$\begin{tikzcd} L \arrow[rr, bend left, "\mathsf{out}"] \arrow[rr, phantom, "\cong"] & & 1 + \mathbb{N}_0 \times L \arrow[ll, bend left, "\mathsf{in}"] \end{tikzcd}$$

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

Linked list

As earlier on:

$$
\begin{tikzcd}
L \arrow[rr, bend left, "\mathbf{out}"] \arrow[rr, phantom, "\cong"] & & 1 + \mathbb{N}_0 \times L \arrow[ll, bend left, "\mathbf{in}"]
\end{tikzcd}
$$

$$\left\{ \begin{array}{l} \mathbf{out} \cdot \mathbf{in} = id \\ \mathbf{in} = [nil, cons] \end{array} \right. \quad \text{leads to} \quad \left\{ \begin{array}{l} \mathbf{out}\ [\,] = i_1\ () \\ \mathbf{out}\ (a : x) = i_2\ (a, x) \end{array} \right.$$

Generalizing

$$\begin{tikzcd}
L \arrow[rr, bend left, "\mathsf{out}"] \arrow[rr, phantom, "\cong"] \arrow[d, "k"'] & & 1 + \mathbb{N}_0 \times L \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "id + id \times k"] \\
B & & 1 + \mathbb{N}_0 \times B \arrow[ll, bend left, "g"]
\end{tikzcd}$$

Generalizing

$$
\begin{tikzcd}[column sep=huge]
L \arrow[r, bend left, "\mathsf{out}"] \arrow[r, phantom, "\cong"] \arrow[d, "k"'] & 1 + \mathbb{N}_0 \times L \arrow[l, bend left, "\mathsf{in}"] \arrow[d, "id + id \times k"] \\
B & 1 + \mathbb{N}_0 \times B \arrow[l, bend left, "g"]
\end{tikzcd}
\qquad
\begin{tikzcd}[column sep=huge]
\mathbb{N}_0 \arrow[r, bend left, "\mathsf{out}"] \arrow[r, phantom, "\cong"] \arrow[d, "k"'] & 1 + \mathbb{N}_0 \arrow[l, bend left, "\mathsf{in}"] \arrow[d, "id + k"] \\
B & 1 + B \arrow[l, bend left, "g"]
\end{tikzcd}
$$

Generalizing

$$\begin{tikzcd}[column sep=huge, row sep=large]
L \arrow[r, bend left, "\mathbf{out}"] \arrow[d, "k"'] \arrow[r, phantom, "\cong"] & 1 + \mathbb{N}_0 \times L \arrow[l, bend left, "\mathbf{in}"] \arrow[d, "id + id \times k"] \\
B & 1 + \mathbb{N}_0 \times B \arrow[l, bend left, "g"]
\end{tikzcd}
\qquad
\begin{tikzcd}[column sep=huge, row sep=large]
\mathbb{N}_0 \arrow[r, bend left, "\mathbf{out}"] \arrow[d, "k"'] \arrow[r, phantom, "\cong"] & 1 + \mathbb{N}_0 \arrow[l, bend left, "\mathbf{in}"] \arrow[d, "id + k"] \\
B & 1 + B \arrow[l, bend left, "g"]
\end{tikzcd}$$

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

Generalizing

$$\begin{tikzcd}
L \arrow[rr, bend left, "\mathbf{out}"] \arrow[rr, phantom, "\cong"] \arrow[d, "k"'] & & 1 + \mathbb{N}_0 \times L \arrow[ll, bend left, "\mathbf{in}"] \arrow[d, "id + id \times k"] \\
B & & 1 + \mathbb{N}_0 \times B \arrow[ll, bend left, "g"]
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathbb{N}_0 \arrow[rr, bend left, "\mathbf{out}"] \arrow[rr, phantom, "\cong"] \arrow[d, "k"'] & & 1 + \mathbb{N}_0 \arrow[ll, bend left, "\mathbf{in}"] \arrow[d, "id + k"] \\
B & & 1 + B \arrow[ll, bend left, "g"]
\end{tikzcd}$$

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

Abstraction ($\mathbb{N}_0$)

$$\begin{tikzcd}
\mathbb{N}_0 \arrow[r, bend left, "\mathsf{out}"] \arrow[r, phantom, "\cong"] \arrow[d, "k"'] & 1 + \mathbb{N}_0 \arrow[l, bend left, "\mathsf{in}"] \arrow[d, "id+k"] \\
B & 1 + B \arrow[l, bend left, "g"]
\end{tikzcd}$$

Abstraction ($\mathbb{N}_0$)

$$
\begin{tikzcd}
\mathbb{N}_0 \arrow[rr, bend left, "\mathsf{out}"] \arrow[rr, phantom, "\cong"] \arrow[d, "k"'] & & 1 + \mathbb{N}_0 \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "id+k"] \\
B & & 1 + B \arrow[ll, bend left, "g"]
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathbb{N}_0 \arrow[rr, bend left, "\mathsf{out}"] \arrow[rr, phantom, "\cong"] \arrow[d, "k"'] & & \mathbf{F}\,\mathbb{N}_0 \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "\mathbf{F}\,k"] \\
B & & \mathbf{F}\,B \arrow[ll, bend left, "g"]
\end{tikzcd}
$$

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

Abstraction (lists)

$$\begin{tikzcd}
L \arrow[rr, bend left, "\mathbf{out}"] \arrow[rr, phantom, "\cong"] \arrow[d, "k"'] & & 1 + \mathbb{N}_0 \times L \arrow[ll, bend left, "\mathbf{in}"] \arrow[d, "id+id\times k"] \\
B & & 1 + \mathbb{N}_0 \times B \arrow[ll, bend left, "g"]
\end{tikzcd}$$

Abstraction (lists)

$$\begin{tikzcd}
L \arrow[rr, bend left, "\mathsf{out}"] \arrow[rr, phantom, "\cong"] \arrow[d, "k"'] & & 1 + \mathbb{N}_0 \times L \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "id + id \times k"] \\
B & & 1 + \mathbb{N}_0 \times B \arrow[ll, bend left, "g"]
\end{tikzcd}$$

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

Abstraction (lists)

$$\begin{tikzcd}
L \arrow[rr, bend left, "\mathsf{out}"] \arrow[rr, phantom, "\cong"] \arrow[d, "k"'] & & 1 + \mathbb{N}_0 \times L \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "id + id \times k"] \\
B & & 1 + \mathbb{N}_0 \times B \arrow[ll, bend left, "g"]
\end{tikzcd}$$

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

$$\begin{tikzcd}
L \arrow[rr, bend left, "\mathsf{out}"] \arrow[rr, phantom, "\cong"] \arrow[d, "k"'] & & \mathsf{F}\ L \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "\mathsf{F}\ k"] \\
B & & \mathsf{F}\ B \arrow[ll, bend left, "g"]
\end{tikzcd}$$

Abstraction

$$
\begin{tikzcd}
T \arrow[r, bend left, "\mathsf{out}"] \arrow[r, phantom, "\cong"] \arrow[d, "k"'] & \mathbf{F}\ T \arrow[l, bend left, "\mathsf{in}"] \arrow[d, "\mathbf{F}\ k"] \\
B & \mathbf{F}\ B \arrow[l, bend left, "g"]
\end{tikzcd}
$$

Abstraction

$$\begin{tikzcd} T \arrow[r, bend left, "\mathbf{out}"] \arrow[r, phantom, "\cong"] \arrow[d, "k"'] & \mathbf{F}\ T \arrow[l, bend left, "\mathbf{in}"] \arrow[d, "\mathbf{F}\ k"] \\ B & \mathbf{F}\ B \arrow[l, bend left, "g"] \end{tikzcd}$$

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

Abstraction

$$
\begin{tikzcd}[column sep=huge, row sep=large]
T \arrow[r, bend left, "\mathbf{out}"] \arrow[r, phantom, "\cong"] \arrow[d, "k"'] & \mathbf{F}\, T \arrow[l, bend left, "\mathbf{in}"] \arrow[d, "\mathbf{F}\, k"] \\
B & \mathbf{F}\, B \arrow[l, bend left, "g"]
\end{tikzcd}
$$

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

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

Abstraction

$$\begin{tikzcd} T \arrow[rr, bend left, "\mathsf{out}"] \arrow[rr, phantom, "\cong"] \arrow[d, "k"'] & & \mathsf{F}\, T \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "\mathsf{F}\, k"] \\ B & & \mathsf{F}\, B \arrow[ll, bend left, "g"] \end{tikzcd}$$

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

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

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

Catamorphisms

Catamorphism

$$\begin{tikzcd}
T \arrow[r, bend left, "\mathsf{out}"] \arrow[d, "(\!| g |\!)"'] \arrow[r, phantom, "\cong"] & \mathbf{F}\ T \arrow[l, bend left, "\mathsf{in}"] \arrow[d, "\mathbf{F}\ (\!| g |\!)"] \\
B & \mathbf{F}\ B \arrow[l, bend left, "g"]
\end{tikzcd}$$

Catamorphism

$$
\begin{tikzcd}
T \arrow[r, bend left, "\mathbf{out}"] \arrow[r, phantom, "\cong"] \arrow[d, "(\!|\, g \,|\!)"'] & \mathbf{F}\ T \arrow[l, bend left, "\mathbf{in}"] \arrow[d, "\mathbf{F}\ (\!|\, g \,|\!)"] \\
B & \mathbf{F}\ B \arrow[l, bend left, "g"]
\end{tikzcd}
$$

Universal property:

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

Summary

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

Summary

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

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

Summary

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

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

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

Summary

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

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

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

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

Examples already seen

Natural numbers:

$$(a\times) = (\!|\, [\underline{0}, (a+)] \,|\!)$$
$$a^b = (\!|\, [\underline{1}, (a\times)] \,|\!)\ b$$

Lists:

$$sum = (\!|\, [\underline{0}, add] \,|\!)$$
$$length = (\!|\, [\underline{0}, succ \cdot \pi_2] \,|\!)$$

Examples already seen

Natural numbers:

$$(a\times) = (\!|\, [\underline{0}, (a+)] \,|\!) = \mathbf{for}\ (a+)\ 0$$
$$a^b = (\!|\, [\underline{1}, (a\times)] \,|\!)\ b = \mathbf{for}\ (a\times)\ 1\ b$$

Lists:

$$sum = (\!|\, [\underline{0}, add] \,|\!) = \mathbf{foldr}\ 0\ (+)$$
$$length = (\!|\, [\underline{0}, succ \cdot \pi_2] \,|\!) = \mathbf{foldr}\ 0\ \overline{succ \cdot \pi_2}$$

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

$$
\begin{array}{ll}
& (\!| g |\!) = (\!| g |\!) \Leftrightarrow (\!| g |\!) \cdot \mathbf{in} = g \cdot \mathbf{F}\ (\!| g |\!) \\
\Leftrightarrow & \qquad \{\ x = x \Leftrightarrow \mathit{true}\ \} \\
& \mathit{true} \Leftrightarrow (\!| g |\!) \cdot \mathbf{in} = g \cdot \mathbf{F}\ (\!| g |\!)
\end{array}
$$

Cata-cancellation

In

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

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

$$
\begin{array}{cl}
 & (\!| g |\!) = (\!| g |\!) \;\Leftrightarrow\; (\!| g |\!) \cdot \mathbf{in} = g \cdot \mathbf{F}\, (\!| g |\!) \\
\Leftrightarrow & \quad \{\; x = x \Leftrightarrow \mathit{true} \;\} \\
 & \mathit{true} \;\Leftrightarrow\; (\!| g |\!) \cdot \mathbf{in} = g \cdot \mathbf{F}\, (\!| g |\!) \\
\Leftrightarrow & \quad \{\; \text{trivial} \;\} \\
 & (\!| g |\!) \cdot \mathbf{in} = g \cdot \mathbf{F}\, (\!| g |\!)
\end{array}
$$

Cata-cancellation

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

$\Leftrightarrow$ $\{$ isomorphism **in** / **out** $\}$

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

Cata-cancellation

$$
\begin{array}{ll}
& (\!| g |\!) \cdot \mathsf{in} = g \cdot \mathsf{F}\,(\!| g |\!) \\
\Leftrightarrow & \qquad \{ \text{ isomorphism } \mathsf{in} \,/\, \mathsf{out} \ \} \\
& (\!| g |\!) = g \cdot \mathsf{F}\,(\!| g |\!) \cdot \mathsf{out}
\end{array}
$$

$$
\begin{tikzcd}
T \arrow[r, bend left, "\mathsf{out}"] \arrow[r, phantom, "\cong"] \arrow[d, "(\!| g |\!)"'] & \mathsf{F}\,T \arrow[l, bend left, "\mathsf{in}"] \arrow[d, "\mathsf{F}\,(\!| g |\!)"] \\
B & \mathsf{F}\,B \arrow[l, bend left, "g"]
\end{tikzcd}
$$

Cata-cancellation

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

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

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

$$\begin{tikzcd}
T \arrow[rr, bend left, "\mathsf{out}"] \arrow[rr, phantom, "\cong"] \arrow[d, "(\!|\, g \,|\!)"'] & & \mathsf{F}\ T \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "\mathsf{F}\,(\!|\, g \,|\!)"] \\
B & & \mathsf{F}\ B \arrow[ll, bend left, "g"]
\end{tikzcd}$$

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

executable definition

Functors

What does $\mathbf{F}$ — in $k \cdot \mathbf{in} = g \cdot \mathbf{F}\ k$ — mean, exactly?

$$\begin{tikzcd}
A \arrow[r, dotted, dash] \arrow[d, "f"'] & \mathbf{F}\ A \arrow[d, "\mathbf{F}\ f"] \\
B \arrow[r, dotted, dash] & \mathbf{F}\ B
\end{tikzcd}$$

Functors

What does **F** — in $k \cdot \mathbf{in} = g \cdot \mathbf{F}\ k$ — mean, exactly?

$$\begin{tikzcd}
A \arrow[r, dotted, no head] \arrow[d, "f"'] & \mathbf{F}\ A \arrow[d, "\mathbf{F}\ f"] \\
B \arrow[r, dotted, no head] & \mathbf{F}\ B
\end{tikzcd}$$

Recall **natural** (free) **properties**

Functors

What does $\mathbf{F}$ — $\mathsf{in}\ k \cdot \mathbf{in} = g \cdot \mathbf{F}\ k$ — mean, exactly?

$$
\begin{tikzcd}
A \arrow[r, dotted, no head] \arrow[d, "f"'] & \mathbf{F}\ A \arrow[d, "\mathbf{F}\ f"] \\
B \arrow[r, dotted, no head] & \mathbf{F}\ B
\end{tikzcd}
$$

For instance:

$$\mathbf{F}\ X = 1 + \mathbb{N}_0 \times X$$

Recall **natural** (free) **properties**

Functors

What does $\mathbf{F}$ — $\mathsf{in}\ k \cdot \mathbf{in} = g \cdot \mathbf{F}\ k$ — mean, exactly?

$$
\begin{tikzcd}
A \arrow[r, dotted, no head] \arrow[d, "f"'] & \mathbf{F}\ A \arrow[d, "\mathbf{F}\ f"] \\
B \arrow[r, dotted, no head] & \mathbf{F}\ B
\end{tikzcd}
$$

Recall **natural** (free) **properties**

For instance:

$$\mathbf{F}\ X = 1 + \mathbb{N}_0 \times X$$

Substitute uppercase (X) by lowercase f ($X := f$):

Functors

What does F — in $k \cdot \mathbf{in} = g \cdot \mathsf{F}\ k$ — mean, exactly?

$$\begin{tikzcd} A \arrow[d, "f"'] \arrow[r, dotted, no head] & \mathsf{F}\ A \arrow[d, "\mathsf{F}\ f"] \\ B \arrow[r, dotted, no head] & \mathsf{F}\ B \end{tikzcd}$$

Recall **natural** (free) **properties**

For instance:

$$\mathsf{F}\ X = 1 + \mathbb{N}_0 \times X$$

Substitute uppercase (X) by lowercase f ($X := f$):

$$\mathsf{F}\ f = id + id \times f$$

Properties

$$\begin{aligned} \mathbf{F}\ id &= id \\ \mathbf{F}\ (g \cdot f) &= (\mathbf{F}\ g) \cdot (\mathbf{F}\ f) \end{aligned}$$

$$\mathbf{F}\ id = id$$

$$\begin{tikzcd}
A \arrow[r, dotted, no head] \arrow[d, "f"'] & \mathbf{F}\ A \arrow[d, "\mathbf{F}\ f"] \\
B \arrow[r, dotted, no head] & \mathbf{F}\ B
\end{tikzcd}$$

$$\mathbf{F}\ id = id$$

$$\begin{tikzcd}
A \arrow[r, dotted, no head] \arrow[d, "id"'] & \mathbf{F}\ A \arrow[d, "\mathbf{F}\ id"] \\
A \arrow[r, dotted, no head] & \mathbf{F}\ A
\end{tikzcd}$$

$$\mathbf{F}\ id = id$$

$$\begin{tikzcd}
A \arrow[r, dotted, no head] \arrow[d, "id"'] & \mathbf{F}\ A \arrow[d, "\mathbf{F}\ id"] \arrow[d, bend left=60, "id"] \\
A \arrow[r, dotted, no head] & \mathbf{F}\ A
\end{tikzcd}$$

$$\mathbf{F}\,(g \cdot f) = (\mathbf{F}\,g) \cdot (\mathbf{F}\,f)$$

$$\begin{tikzcd}
A \arrow[r, dotted, no head] \arrow[d, "f"'] & \mathbf{F}\,A \arrow[d, "\mathbf{F}\,f"] \\
B \arrow[r, dotted, no head] \arrow[d, "g"'] & \mathbf{F}\,B \arrow[d, "\mathbf{F}\,g"] \\
C \arrow[r, dotted, no head] & \mathbf{F}\,C
\end{tikzcd}$$

$$\mathbf{F}\ (g \cdot f) = (\mathbf{F}\ g) \cdot (\mathbf{F}\ f)$$

$$\begin{tikzcd}
A \arrow[r, dotted, no head] \arrow[d, "f"'] \arrow[dd, bend right=60, "g \cdot f"'] & \mathbf{F}\ A \arrow[d, "\mathbf{F}\ f"] \\
B \arrow[r, dotted, no head] \arrow[d, "g"'] & \mathbf{F}\ B \arrow[d, "\mathbf{F}\ g"] \\
C \arrow[r, dotted, no head] & \mathbf{F}\ C
\end{tikzcd}$$

$$\mathbf{F}\,(g \cdot f) = (\mathbf{F}\,g) \cdot (\mathbf{F}\,f)$$

$$\begin{tikzcd}
A \arrow[d, "f"'] \arrow[dd, bend right=60, "g\cdot f"'] \arrow[r, dotted, no head] & \mathbf{F}\,A \arrow[d, "\mathbf{F}\,f"] \arrow[dd, bend left=60, "\mathbf{F}\,g\cdot\mathbf{F}\,f"] \arrow[dd, bend left=90, "\mathbf{F}\,(g\cdot f)"] \\
B \arrow[d, "g"'] \arrow[r, dotted, no head] & \mathbf{F}\,B \arrow[d, "\mathbf{F}\,g"] \\
C \arrow[r, dotted, no head] & \mathbf{F}\,C
\end{tikzcd}$$

Well known example

List functor:

$$\mathbf{F}\ f = f^{\star}$$

Well known example

List functor:

$$\mathbf{F}\ f = f^\star \quad \textbf{where} \quad f^\star\ x = [f\ a \mid a \leftarrow x] = \mathsf{map}\ f\ x$$

$$\begin{tikzcd} A \arrow[r, dotted, no head] \arrow[d, "f"'] & A^\star \arrow[d, "f^\star"] \\ B \arrow[r, dotted, no head] & B^\star \end{tikzcd}$$

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

$$
\begin{aligned}
\mathsf{F}\ id &= id \\
\mathsf{F}\ (g \cdot f) &= (\mathsf{F}\ g) \cdot (\mathsf{F}\ f)
\end{aligned}
$$

These properties are essential for reasoning about **catamorphisms**.

See examples next.

Cata-reflexion $(k := id)$

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

Cata-reflexion $(k := id)$

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

Thanks to $\mathbf{F}\ id = id$, one immediately gets

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

that is:

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

Cata-fusion

$$
\begin{tikzcd}
T \arrow[d, "(\!| g |\!)"'] \arrow[dd, bend right=60, "(\!| h |\!)"'] & \mathbf{F}\ T \arrow[l, "\mathsf{in}"'] \arrow[d, "\mathbf{F}\ (\!| g |\!)"] \\
A \arrow[d, "f"'] & \mathbf{F}\ A \arrow[l, "g"] \arrow[d, "\mathbf{F}\ f"] \\
B & \mathbf{F}\ B \arrow[l, "h"]
\end{tikzcd}
$$

Cata-fusion

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

Cata-fusion

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

Cata-fusion

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

Cata-fusion

$$f \cdot g \cdot \mathsf{F}\, (\!|\, g \,|\!) = h \cdot \mathsf{F}\, (f \cdot (\!|\, g \,|\!))$$

Cata-fusion

$$
\begin{array}{ll}
& f \cdot g \cdot \mathbf{F}\,(\!|\, g \,|\!) = h \cdot \mathbf{F}\,(f \cdot (\!|\, g \,|\!)) \\
\Leftrightarrow & \quad \{ \ \text{functor-}\mathbf{F}\ (1) \ \} \\
& f \cdot g \cdot \mathbf{F}\,(\!|\, g \,|\!) = h \cdot \mathbf{F}\, f \cdot \mathbf{F}\,(\!|\, g \,|\!)
\end{array}
$$

Cata-fusion

$$
\begin{array}{ll}
 & f \cdot g \cdot \mathbf{F}\,(\!|\,g\,|\!) = h \cdot \mathbf{F}\,(f \cdot (\!|\,g\,|\!)) \\
\Leftrightarrow & \qquad \{\ \text{functor-}\mathbf{F}\ (1)\ \} \\
 & f \cdot g \cdot \mathbf{F}\,(\!|\,g\,|\!) = h \cdot \mathbf{F}\,f \cdot \mathbf{F}\,(\!|\,g\,|\!) \\
\Leftarrow & \qquad \{\ \text{Leibniz}\ \} \\
 & f \cdot g = h \cdot (\mathbf{F}\,f)
\end{array}
$$

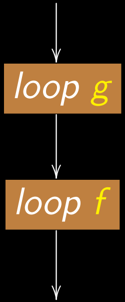
Cata-fusion

$$
\begin{array}{ll}
 & f \cdot g \cdot \mathbf{F}\,(\!|\, g \,|\!) = h \cdot \mathbf{F}\,(f \cdot (\!|\, g \,|\!)) \\
\Leftrightarrow & \quad \{ \text{ functor-}\mathbf{F}\text{ (1) } \} \\
 & f \cdot g \cdot \mathbf{F}\,(\!|\, g \,|\!) = h \cdot \mathbf{F}\, f \cdot \mathbf{F}\,(\!|\, g \,|\!) \\
\Leftarrow & \quad \{ \text{ Leibniz } \} \\
 & f \cdot g = h \cdot (\mathbf{F}\, f)
\end{array}
$$

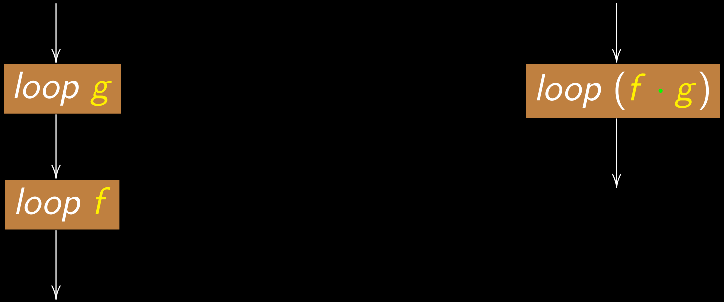
Summing up:

$$
f \cdot (\!|\, g \,|\!) = (\!|\, h \,|\!) \quad \Leftarrow \quad f \cdot g = h \cdot (\mathbf{F}\, f)
$$

Fusion in programming — "sequential"



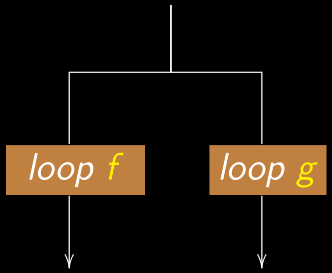
Fusion in programming — "sequential"



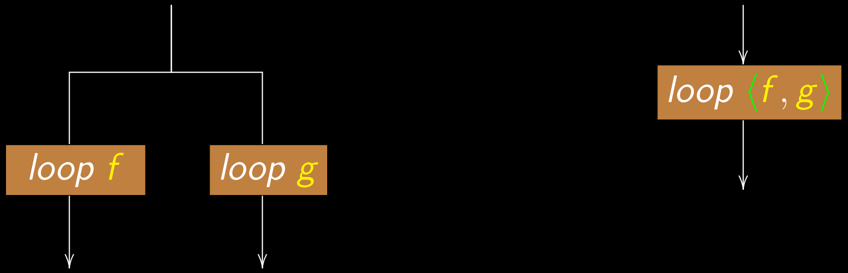
Cálculo de Programas

Class T08

Fusion in programming — "parallel"



Fusion in programming — "parallel"



The pattern $\langle f, h\rangle = (\!| g |\!)$

"Catas" that yield pairs:

$$\begin{tikzcd}
T \arrow[rr, bend left, "\mathbf{out}"] \arrow[rr, phantom, "\cong"] \arrow[d, "(\!| g |\!)"'] & & \mathbf{F}\ T \arrow[ll, bend left, "\mathbf{in}"] \arrow[d, "\mathbf{F}\ (\!| g |\!)"] \\
A \times B & & \mathbf{F}\ (A \times B) \arrow[ll, bend left, "g"]
\end{tikzcd}$$

Catas that yield pairs

$$
\begin{tikzcd}
T \arrow[r, bend left, "\mathsf{out}"] \arrow[r, phantom, "\cong"] \arrow[d, "{\langle f,h\rangle = (\!|\, g \,|\!)}"'] & \mathbf{F}\,T \arrow[l, bend left, "\mathsf{in}"] \arrow[d, "{\mathbf{F}\,\langle f,h\rangle}"] \\
A \times B & \mathbf{F}\,(A \times B) \arrow[l, bend left, "g"]
\end{tikzcd}
$$

Catas that yield pairs

$$
\begin{tikzcd}
T \arrow[r, bend left, "\mathsf{out}"] \arrow[r, phantom, "\cong"] \arrow[d, swap, "\langle f, h\rangle = (\!|\, \langle k, m\rangle \,|\!)"] & \mathbf{F}\ T \arrow[l, bend left, "\mathsf{in}"] \arrow[d, "\mathbf{F}\ \langle f, h\rangle"] \\
A \times B & \mathbf{F}\ (A \times B) \arrow[l, bend left, "g = \langle k, m\rangle"]
\end{tikzcd}
$$

Catas that yield pairs

$$\langle f, h \rangle = (\!| \langle k, m \rangle |\!)$$

$$\Leftrightarrow \qquad \{ \text{ Cata-universal } \}$$

Catas that yield pairs

$$\langle f, h \rangle = (\!| \langle k, m \rangle |\!)$$

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

$$\langle f, h \rangle \cdot \mathbf{in} = \langle k, m \rangle \cdot \mathsf{F}\, \langle f, h \rangle$$

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

Catas that yield pairs

$$
\begin{array}{cl}
 & \langle f, h \rangle = (\!|\, \langle k, m \rangle \,|\!) \\
\Leftrightarrow & \quad \{ \text{ Cata-universal } \} \\
 & \langle f, h \rangle \cdot \mathsf{in} = \langle k, m \rangle \cdot \mathsf{F}\ \langle f, h \rangle \\
\Leftrightarrow & \quad \{ \times\text{-fusion } \} \\
 & \langle f \cdot \mathsf{in}, h \cdot \mathsf{in} \rangle = \langle k \cdot \mathsf{F}\ \langle f, h \rangle, m \cdot \mathsf{F}\ \langle f, h \rangle \rangle \\
\Leftrightarrow & \quad \{ \text{ Eq-}\times \text{ } \}
\end{array}
$$

Catas that yield pairs

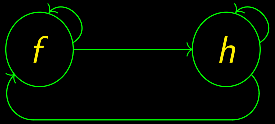
$$
\begin{array}{ll}
& \langle f, h \rangle = (\!| \langle k, m \rangle |\!) \\
\Leftrightarrow & \quad \{ \text{ Cata-universal } \} \\
& \langle f, h \rangle \cdot \mathsf{in} = \langle k, m \rangle \cdot \mathsf{F}\ \langle f, h \rangle \\
\Leftrightarrow & \quad \{ \ \times\text{-fusion } \} \\
& \langle f \cdot \mathsf{in}, h \cdot \mathsf{in} \rangle = \langle k \cdot \mathsf{F}\ \langle f, h \rangle, m \cdot \mathsf{F}\ \langle f, h \rangle \rangle \\
\Leftrightarrow & \quad \{ \text{ Eq-}\times \ \} \\
& \left\{ \begin{array}{l} f \cdot \mathsf{in} = k \cdot \mathsf{F}\ \langle f, h \rangle \\ h \cdot \mathsf{in} = m \cdot \mathsf{F}\ \langle f, h \rangle \end{array} \right.
\end{array}
$$

Mutual recursion

$$\begin{tikzcd}
T \arrow[rr, bend left, "\mathsf{out}"] \arrow[rr, phantom, "\cong"] \arrow[d, "\langle f,h\rangle = (\!|\, \langle k,m\rangle \,|\!)"'] & & \mathbf{F}\ T \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "\mathbf{F}\ \langle f,h\rangle"] \\
A \times B & & \mathbf{F}\ (A \times B) \arrow[ll, bend left, "g = \langle k,m\rangle"]
\end{tikzcd}$$

$$\langle f, h\rangle = (\!|\, \langle k, m\rangle \,|\!) \Leftrightarrow \left\{ \begin{array}{l} f \cdot \mathsf{in} = k \cdot \mathbf{F}\ \langle f, h\rangle \\ h \cdot \mathsf{in} = m \cdot \mathbf{F}\ \langle f, h\rangle \end{array} \right.$$

Mutual recursion



$$\left\{ \begin{array}{l} f \cdot \mathbf{in} = k \cdot \mathbf{F}\ \langle f, h \rangle \\ h \cdot \mathbf{in} = m \cdot \mathbf{F}\ \langle f, h \rangle \end{array} \right. \Leftrightarrow \langle f, h \rangle = (\!|\ \langle k, m \rangle\ |\!)$$

Mutual recursion — case $T = \mathbb{N}_0$

$$\langle f, g \rangle = (\!|\, \langle h, k \rangle \,|\!)$$

$$\Leftrightarrow \quad \{ \text{ mutual recursion with } \mathbf{in} = [\underline{0}, succ], \ \mathbf{F}\ f = id + f \text{ etc } \}$$

$$\begin{cases} f \cdot [\underline{0}, succ] = h \cdot (id + \langle f, g \rangle) \\ g \cdot [\underline{0}, succ] = k \cdot (id + \langle f, g \rangle) \end{cases}$$

$$\Leftrightarrow \quad \{ \text{ let } h = [h_1, h_2] \text{ and } k = [k_1, k_2] \ \}$$

$$\begin{cases} f \cdot [\underline{0}, succ] = [h_1, h_2] \cdot (id + \langle f, g \rangle) \\ g \cdot [\underline{0}, succ] = [k_1, k_2] \cdot (id + \langle f, g \rangle) \end{cases}$$

Mutual recursion — case $T = \mathbb{N}_0$

Moving on:

$$
\begin{array}{ll}
& \langle f , g \rangle = (\!|\, \langle [h_1 , h_2], [k_1 , k_2] \rangle \,|\!) \\
\Leftrightarrow & \qquad \{ \text{ coproduct laws } \} \\
& \left\{ \begin{array}{l} \left\{ \begin{array}{l} f \cdot \underline{0} = h_1 \\ f \cdot succ = h_2 \cdot \langle f , g \rangle \end{array} \right. \\ \left\{ \begin{array}{l} g \cdot \underline{0} = k_1 \\ g \cdot succ = k_2 \cdot \langle f , g \rangle \end{array} \right. \end{array} \right.
\end{array}
$$

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:

$$f\ n = \mathit{fib}\ (n + 1)$$

Then:

$$\begin{aligned} &f\ 0 = \mathit{fib}\ 1 = 1 \\ &f\ (n+1) = \mathit{fib}\ (n+2) = f\ n + \mathit{fib}\ n \end{aligned}$$

Example — Fibonacci

Concerning *fib*:

$$\begin{aligned} fib\ 0 &= 1 \\ fib\ (n+1) &= f\ n \end{aligned}$$

Summing up, we have the following mutual recursion:

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} f\ 0 = 1 \\ f\ (n+1) = f\ n + fib\ n \end{array} \right. \\ \left\{ \begin{array}{l} fib\ 0 = 1 \\ fib\ (n+1) = f\ n \end{array} \right. \end{array} \right.$$

Example — Fibonacci

By applying the mutual recursion law for $T = \mathbb{N}_0$:

$$\left\{\begin{array}{l}\left\{\begin{array}{l} f\ 0 = a \\ f\ (n+1) = h_2\ (f\ n, \mathit{fib}\ n) \end{array}\right. \\ \left\{\begin{array}{l} \mathit{fib}\ 0 = b \\ \mathit{fib}\ (n+1) = k_2\ (f\ n, \mathit{fib}\ n) \end{array}\right.\end{array}\right. \quad \Leftrightarrow \quad \langle f, \mathit{fib} \rangle = (\!|\ \langle [\underline{a}, h_2], [\underline{b}, k_2] \rangle\ |\!)$$

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:

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

$$\begin{aligned}
& \langle f, fib \rangle = (\!|\, \langle [\underline{1}, add], [\underline{1}, \pi_1] \rangle \,|\!) \\
\Leftrightarrow \quad & \{ \text{ exchange law } \} \\
& \langle f, fib \rangle = (\!|\, [\langle \underline{1}, \underline{1} \rangle, \langle add, \pi_1 \rangle] \,|\!) \\
\Leftrightarrow \quad & \{ \text{ recall exercise sheet 3 } \} \\
& \langle f, fib \rangle = (\!|\, [\underline{(1, 1)}, \langle add, \pi_1 \rangle] \,|\!) \\
\Leftrightarrow \quad & \{ \text{ definition of } \textbf{for}\ b\ i \ \} \\
& \langle f, fib \rangle = \textbf{for}\ \langle add, \pi_1 \rangle\ (1, 1)
\end{aligned}$$

Example — Fibonacci

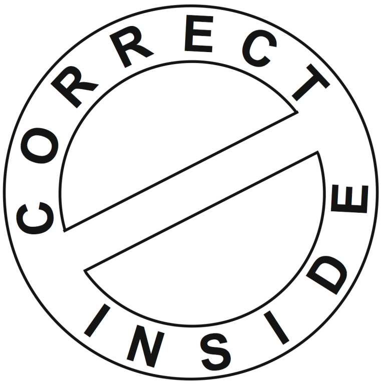
In summary:

$$fib = \pi_2 \cdot (\mathbf{for}\ \langle add, \pi_1 \rangle\ (1, 1))$$

Cf. the same in the syntax of **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;
};
```

Program Calculation



Efficiency without sacrificing **correction**.

(Who would care about an efficient *fib* giving possibly wrong outputs?)

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

$$\langle f, fac \rangle = (\!|\, \langle [\underline{a}, h_2], [\underline{b}, k_2] \rangle \,|\!) \quad \Leftrightarrow \quad \left\{ \begin{array}{l} \left\{ \begin{array}{l} f\ 0 = a \\ f\ (n+1) = h_2\ (f\ n, fac\ n) \end{array} \right. \\ \left\{ \begin{array}{l} fac\ 0 = b \\ fac\ (n+1) = k_2\ (f\ n, fac\ n) \end{array} \right. \end{array} \right.$$

$$k_2 = mul$$
$$b = 1$$
$$f\ n = n + 1$$
$$a = 1$$

(Still need h_2...)

Example — factorial

$$\begin{aligned}
& f\ (n+1) \\
= \ & (n+1)+1 \\
= \ & 1+(n+1) \\
= \ & 1+f\ n \\
= \ & 1+\pi_1\ (f\ n, fac\ n) \\
= \ & \underbrace{succ \cdot \pi_1}_{h_2}\ (f\ n, fac\ n)
\end{aligned}$$

Then:

$$\begin{aligned}
k_2 &= mul \\
b &= 1 \\
h_2 &= succ \cdot \pi_1 \\
a &= 1
\end{aligned}$$

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

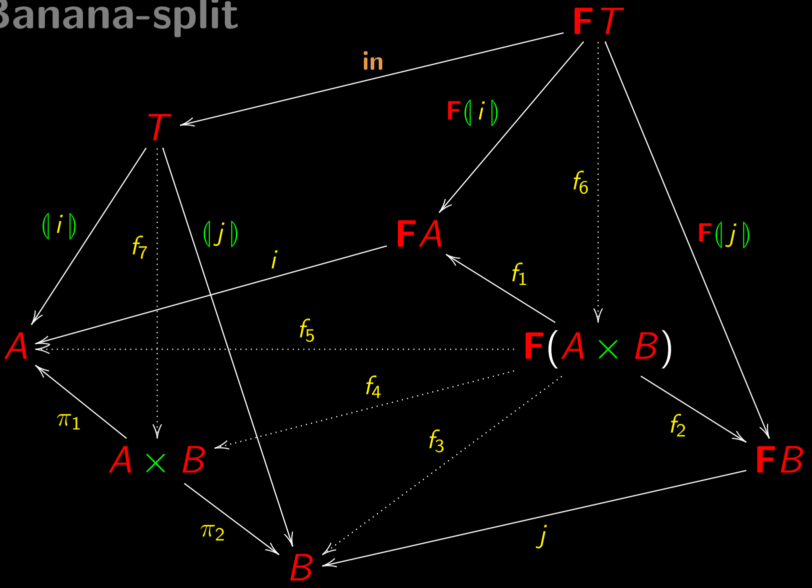
$$\langle \llparenthesis \_ \rrparenthesis, \llparenthesis \_ \rrparenthesis \rangle$$

$$\llparenthesis \langle \_, \_ \rangle \rrparenthesis$$

$$\langle \llparenthesis i \rrparenthesis, \llparenthesis j \rrparenthesis \rangle$$

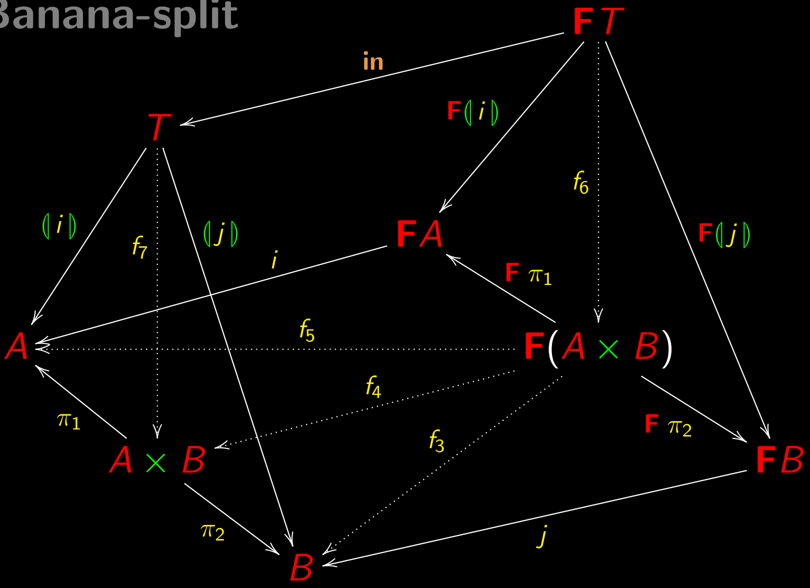
Banana-split

$\langle (\!| i |\!), (\!| j |\!) \rangle$



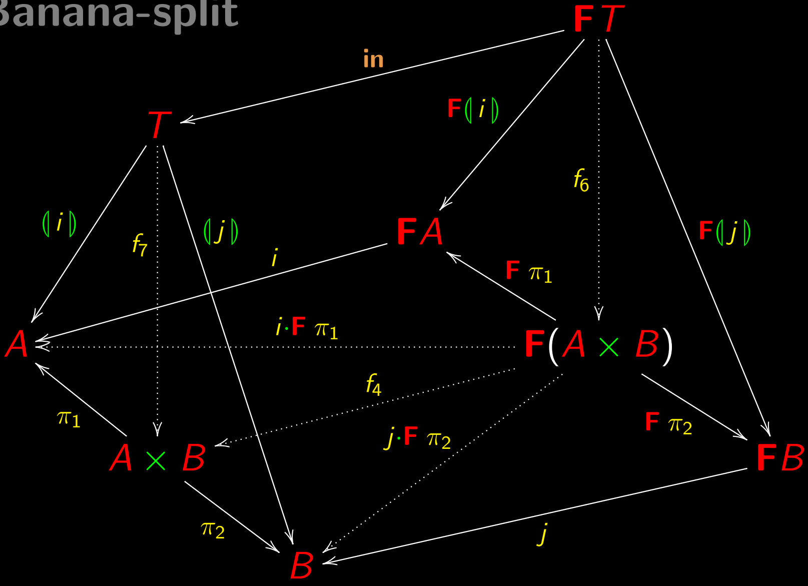
Banana-split

$\langle (\!|\, i \,|\!), (\!|\, j \,|\!) \rangle$



Banana-split

$\langle (\!| i |\!), (\!| j |\!) \rangle$



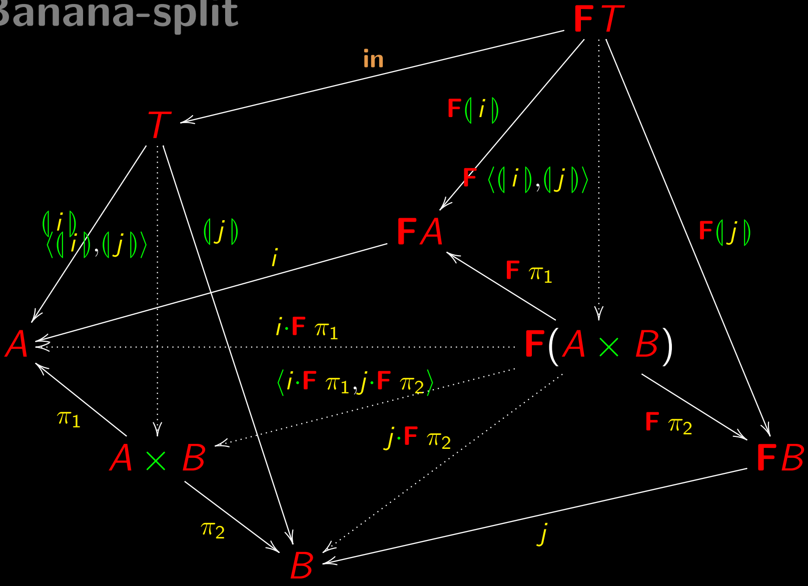
Banana-split

$$\langle \llparenthesis\, i \,\rrparenthesis, \llparenthesis\, j \,\rrparenthesis \rangle$$

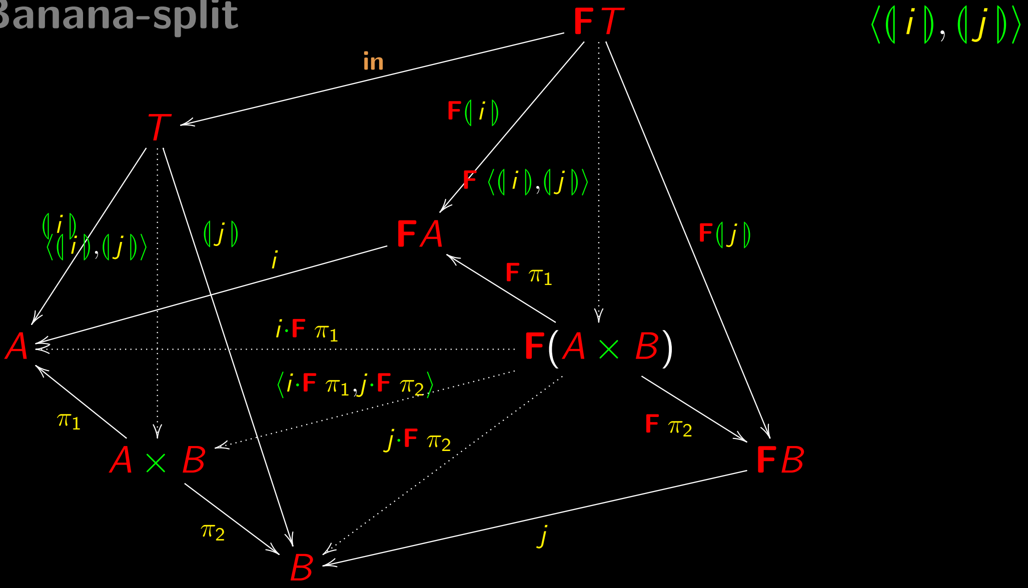
$$\begin{tikzcd}
 & & & & \mathbf{F}\,T \arrow[dlll, "\mathsf{in}"'] \arrow[dl, "\mathbf{F}\llparenthesis\, i \,\rrparenthesis"'] \arrow[ddd, dotted, "f_6"] \arrow[ddddr, "\mathbf{F}\llparenthesis\, j \,\rrparenthesis"] & \\
 & T \arrow[ddl, "\llparenthesis\, i \,\rrparenthesis"'] \arrow[ddd, dotted, "f_7"'] \arrow[ddddr, "\llparenthesis\, j \,\rrparenthesis"] & & & & \\
 & & & \mathbf{F}\,A \arrow[dlll, "i"'] & & \\
A & & & & \mathbf{F}(A \times B) \arrow[ul, "\mathbf{F}\ \pi_1"'] \arrow[llll, dotted, "i \cdot \mathbf{F}\ \pi_1"'] \arrow[dlll, dotted, "\langle i \cdot \mathbf{F}\ \pi_1, j \cdot \mathbf{F}\ \pi_2 \rangle"'] \arrow[ddll, dotted, "j \cdot \mathbf{F}\ \pi_2"] \arrow[dr, "\mathbf{F}\ \pi_2"'] & \\
 & A \times B \arrow[ul, "\pi_1"] \arrow[dr, "\pi_2"'] & & & & \mathbf{F}\,B \arrow[dlll, "j"] \\
 & & B & & &
\end{tikzcd}$$

Banana-split

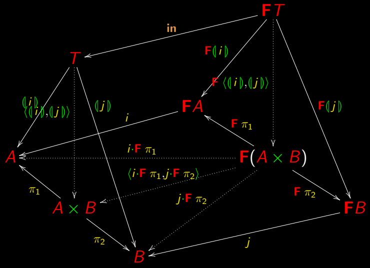
$\langle (\!| i |\!), (\!| j |\!) \rangle$



Banana-split



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:

$$\mathit{average} = (/) \cdot \langle \mathit{sum}, \mathit{length} \rangle$$

where

$$\mathit{sum} = (\!|\, [0, \mathit{add}] \,|\!)$$

$$\mathit{length} = (\!|\, [0, \mathit{succ} \cdot \pi_2] \,|\!)$$

Example

$$average = (/) \cdot \underbrace{\langle sum, length \rangle}_{aux}$$

By banana-split:

$$\begin{aligned} aux &= (\!|\, \langle f_5, f_3 \rangle \,|\!) \textbf{ where} \\ f_5 &= [0, add] \cdot (id + id \times \pi_1) \\ f_3 &= [0, succ \cdot \pi_2] \cdot (id + id \times \pi_2) \end{aligned}$$

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,

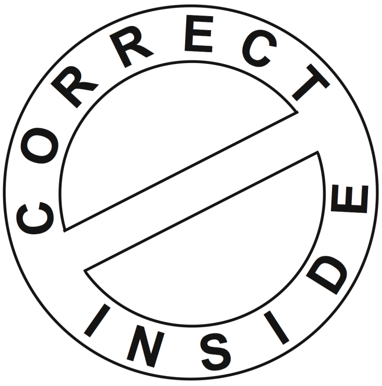
$$
\begin{aligned}
& \mathit{average}\ l = x \mathbin{/} y\ \textbf{where} \\
& \quad (x, y) = \mathit{aux}\ l \\
& \quad \mathit{aux}\ [\,] = (0, 0) \\
& \quad \mathit{aux}\ (a : l) = (a + x, y + 1)\ \textbf{where}\ (x, y) = \mathit{aux}\ l
\end{aligned}
$$

twice as fast as

$$
\begin{aligned}
& \mathit{average}\ l = (\mathit{sum}\ l) \mathbin{/} (\mathit{length}\ l)\ \textbf{where} \\
& \quad \mathit{sum}\ [\,] = 0 \\
& \quad \mathit{sum}\ (h : t) = h + \mathit{sum}\ t \\
& \quad \mathit{length}\ [\,] = 0 \\
& \quad \mathit{length}\ (h : t) = 1 + \mathit{length}\ t
\end{aligned}
$$

and **guaranteed** to be correct.

Program Calculation



Efficiency without sacrificing **correction**.

Program Calculation

Efficiency should never sacrifice **correction**!

Recap

Inductive type T and the functor $\mathbf{F}$ that determines its pattern of recursion:

$$\begin{tikzcd} T \arrow[r, bend left, "\mathsf{out}"] \arrow[r, phantom, "\cong"] & \mathbf{F}\ T \arrow[l, bend left, "\mathsf{in}"] \end{tikzcd}$$

Recap

Inductive type T and the functor $\mathbf{F}$ that determines its pattern of recursion:

$$\begin{tikzcd} T \arrow[r, bend left, "\mathsf{out}"] \arrow[r, phantom, "\cong"] & \mathbf{F}\ T \arrow[l, bend left, "\mathsf{in}"] \end{tikzcd}$$

Two examples so far: $T = \mathbb{N}_0$ and $T = A^\star$

Recap

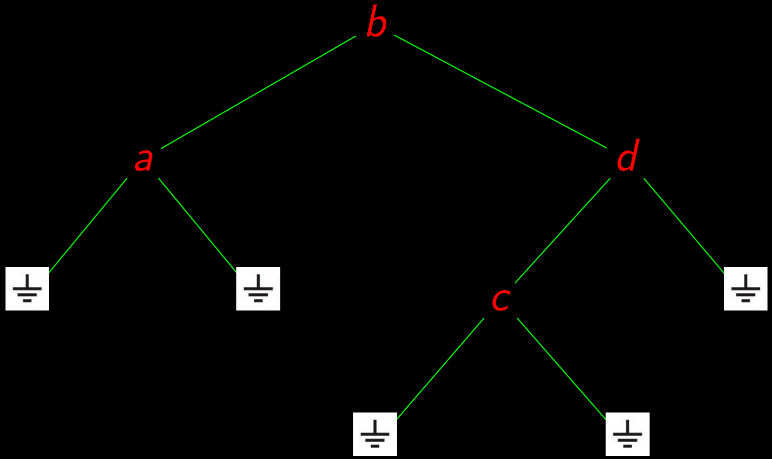
Inductive type T and the functor $\mathbf{F}$ that determines its pattern of recursion:

$$\begin{tikzcd} T \arrow[rr, bend left, "\mathsf{out}"] \arrow[rr, phantom, "\cong"] & & \mathbf{F}\ T \arrow[ll, bend left, "\mathsf{in}"] \end{tikzcd}$$

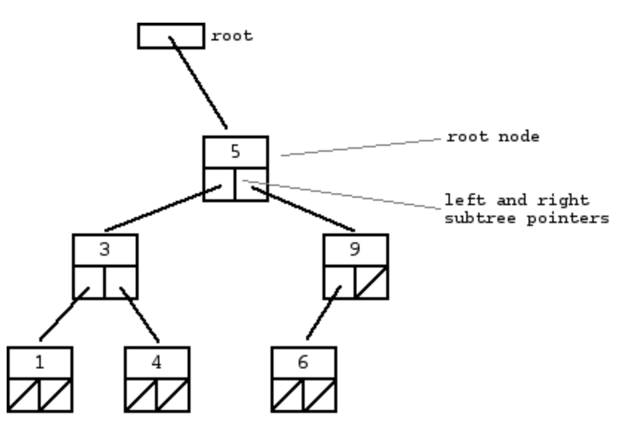
Two examples so far: $T = \mathbb{N}_0$ and $T = A^{\star}$

Let us see some more.

Trees



Trees



Binary trees

Trees with data of type A in their nodes, for example

$$\texttt{data}\ \mathbf{BTree} = \mathbf{Empty} \mid \mathbf{Node}\ (A, (\mathbf{BTree}, \mathbf{BTree}))$$

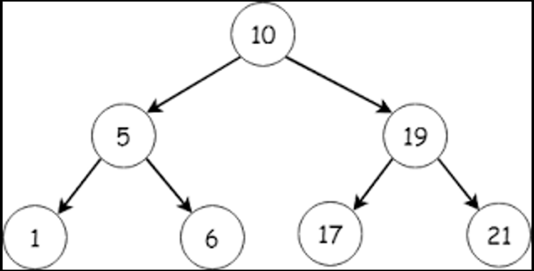
(A assumed pre-defined.)

Binary trees

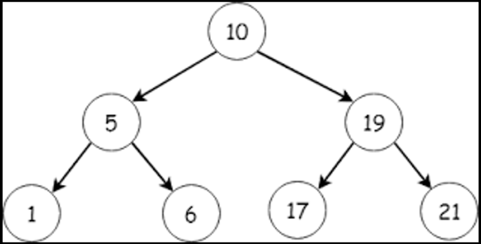
Trees with data of type A in their nodes, for example

$$\texttt{data}\ \mathbf{BTree} = \mathbf{Empty} \mid \mathbf{Node}\ (A, (\mathbf{BTree}, \mathbf{BTree}))$$

(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
```

$$\mathbf{Node} : A \times (\mathbf{BTree} \times \mathbf{BTree}) \to \mathbf{BTree}$$

$$\underline{\mathbf{Empty}} : 1 \to \mathbf{BTree}$$

Binary trees

$$\mathsf{in} = [\underline{\mathsf{Empty}}, \mathsf{Node}]$$

Binary trees

$$\mathsf{in} = [\underline{\mathsf{Empty}}, \mathsf{Node}]$$

$$\begin{tikzcd} \mathsf{BTree} \arrow[rr, bend left, "\mathsf{out}"] \arrow[rr, phantom, "\cong"] & & \underbrace{1 + A \times (\mathsf{BTree} \times \mathsf{BTree})}_{\mathsf{F\ BTree}} \arrow[ll, bend left, "\mathsf{in}"] \end{tikzcd}$$

$$\mathtt{data}\ \mathsf{BTree} = \mathsf{Empty} \mid \mathsf{Node}\ (A, (\mathsf{BTree}, \mathsf{BTree}))$$

Binary trees

$$\begin{tikzcd} \mathbf{BTree} \arrow[r, bend left, "\mathsf{out}"] \arrow[r, phantom, "\cong"] & \underbrace{1 + A \times (\mathbf{BTree} \times \mathbf{BTree})}_{\mathsf{F}\ \mathbf{BTree}} \arrow[l, bend left, "\mathsf{in}"] \end{tikzcd}$$

$$T = \mathbf{BTree}$$

$$\begin{cases} \mathsf{F}\ X = 1 + A \times (X \times X) \\ \mathsf{F}\ f = id + id \times (f \times f) \end{cases}$$

$$\mathsf{in} = [\underline{\mathsf{Empty}}, \mathsf{Node}]$$

Catamorphisms of binary trees

$$
\begin{tikzcd}[column sep=huge, row sep=huge]
\mathbf{BTree} \arrow[r, bend left=15, "\mathsf{out}"] \arrow[r, phantom, "\cong"] \arrow[d, "(\!| g |\!)"'] & \underbrace{1 + A \times (\mathbf{BTree} \times \mathbf{BTree})}_{\mathbf{F}\ \mathbf{BTree}} \arrow[l, bend left=15, "\mathsf{in}"] \arrow[d, "\underbrace{id + id \times (\!| g |\!) \times (\!| g |\!)}_{\mathbf{F}\ (\!| g |\!)}"] \\
B & \underbrace{1 + A \times (B \times B)}_{\mathbf{F}\ B} \arrow[l, bend left=10, "g"]
\end{tikzcd}
$$

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

$$f : \mathbf{BTree} \to \mathbb{N}_0$$

that computes the **size** of the tree (total number of nodes).

Catamorphisms of binary trees

Example: function

$$f : \mathbf{BTree} \to \mathbb{N}_0$$

that computes the **size** of the tree (total number of nodes).

It suffices to focus on "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

$$\mathbb{N}_0 \xleftarrow{\quad g \quad} 1 + A \times (\mathbb{N}_0 \times \mathbb{N}_0)$$

Catamorphisms of binary trees

$$\mathbb{N}_0 \xleftarrow{g = [g_1\,,\,g_2]} 1 + A \times (\mathbb{N}_0 \times \mathbb{N}_0)$$

Catamorphisms of binary trees

$$\mathbb{N}_0 \xleftarrow{g \,=\, [g_1\,,\, g_2]} \overbrace{1}^{\text{empty tree}} + A \times (\mathbb{N}_0 \times \mathbb{N}_0)$$

$$g_1 \; \_ = 0$$

Catamorphisms of binary trees

empty tree

$$\mathbb{N}_0 \xleftarrow{\;g \,=\, [g_1\,,\, g_2]\;} 1 + A \times (\mathbb{N}_0 \times \mathbb{N}_0)$$

$$g_1\ \_ = 0$$

total size

size of left sub-tree

size of right sub-tree

Catamorphisms of binary trees

empty tree

$$\mathbb{N}_0 \xleftarrow{\;g = [g_1, g_2]\;} 1 + A \times (\mathbb{N}_0 \times \mathbb{N}_0)$$

$$g_1\ \_ = 0$$

$$g_2\ (a, (l, r)) = 1 + l + r$$

total size

size of left sub-tree

size of right sub-tree

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
```

Catamorphisms of binary trees

Haskell (**pointfree**):

```
size = cataBTree(either g1 g2) where
   g1 _ = 0
   g2(a,(l,r)) = 1 + l + r
```

Catamorphisms of binary trees

Haskell (**pointfree**):

```
size = cataBTree(either g1 g2) where
    g1 _ = 0
    g2(a,(l,r)) = 1 + l + r
```

Haskell (**pointwise**):

```
size Empty = 0
size (Node(a,(t1,t2))) = 1 + size t1 + size t2
```

Inductive type repertoire (BTree)

(Assuming A predefined, for the moment).

Trees whose data of type A are stored in their **nodes**:

$$\mathbf{T} = \mathbf{BTree} \quad \text{where} \quad \left\{ \begin{array}{l} \mathsf{F}\ X = 1 + A \times X^2 \\ \mathsf{F}\ f = id + id \times f^2 \\ \mathbf{in} = [\underline{\mathbf{Empty}}, \mathbf{Node}] \end{array} \right.$$

Haskell:

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

Inductive type repertoire (LTree)

Trees with data in their **leaves**:

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

Haskell:

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

Inductive type repertoire (FTree)

(Assuming A and B predefined, for the moment).

Trees bearing data in **both** leaves and nodes:

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

Haskell:

```
data FTree b a = Unit b | Comp (a,(FTree b a, FTree b
a))
```

Inductive type repertoire (**Expr**)

(Assume V and O predefined, for the moment).

Expression trees:

$$\mathbf{T} = \mathbf{Expr} \quad \text{where} \quad \left\{ \begin{array}{l} \mathbf{F}\ X = V + O \times X^{\star} \\ \mathbf{F}\ f = id + id \times map\ f \\ \mathbf{in} = [\mathbf{Var}, \mathbf{Term}] \end{array} \right.$$

Haskell:

```
data Expr v o = Var v | Term (o,[Expr v o])
```

Inductive type repertoire (Expr)

```
<html>
  <head>
    <title>Title of the document</title>
  </head>
  <body>

    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>

  </body>
</html>
```

Inductive type repertoire (Expr)

```
Term "html" [
   Term "head" [
     Term "title"
        [Var "Title of the document" ]
   ],
     Term "body" [
        Term "h1" [
           Var "This is a heading" ],
        Term "p" [
           Var "This is a paragraph." ]
     ]
]
```

Inductive types

(a) Trees with data in their leaves :

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

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

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

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

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

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

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

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

(d) Expression trees:

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

Haskell: $\textbf{data}\ \mathsf{Expr}\ v\ \ o = \mathit{Var}\ \ v \mid \mathit{Term}\ (o, [\mathsf{Expr}\ v\ \ o])$

Cálculo de Programas

Class T09

Recall

$$
\begin{tikzcd}
B \arrow[r, "i_1"] \arrow[dr, "f"'] & B + C \arrow[d, "{[f,g]}"] & C \arrow[l, "i_2"'] \arrow[dl, "g"] \\
& A &
\end{tikzcd}
$$

Recall

$$\begin{tikzcd} B \arrow[r, "i_1"] \arrow[dr, "f"'] & B + C \arrow[d, "{[f,g]}"] & C \arrow[l, "i_2"'] \arrow[dl, "g"] \\ & A & \end{tikzcd} \qquad \begin{tikzcd} B & B \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \\ & A \arrow[ul, "f"] \arrow[u, "{\langle f,g \rangle}"'] \arrow[ur, "g"'] & \end{tikzcd}$$

Compare

$$
\begin{tikzcd}[column sep=huge, row sep=large]
T \arrow[d, "k"'] \arrow[r, bend right, "\mathbf{out}"'] \arrow[r, phantom, "\cong"] & \mathbf{F}\ T \arrow[l, bend right, "\mathbf{in}"'] \arrow[d, "\mathbf{F}\ k"] \\
C & \mathbf{F}\ C \arrow[l, bend left, "g"]
\end{tikzcd}
$$

Compare

$$
\begin{tikzcd}
T \arrow[r, bend right, "\mathbf{out}"'] \arrow[r, phantom, "\cong"] \arrow[d, "k"'] & \mathbf{F}\,T \arrow[l, bend right, "\mathbf{in}"'] \arrow[d, "\mathbf{F}\,k"] \\
C & \mathbf{F}\,C \arrow[l, bend left, "g"]
\end{tikzcd}
$$

$\kappa\alpha\tau\alpha$ (“cata”)

Compare

$$\begin{tikzcd}[column sep=huge, row sep=large]
T \arrow[r, bend right, "\mathsf{out}"'] \arrow[d, "k"'] \arrow[r, phantom, "\cong"] & \mathbf{F}\ T \arrow[l, bend right, "\mathsf{in}"'] \arrow[d, "\mathbf{F}\ k"] \\
C & \mathbf{F}\ C \arrow[l, bend left, "g"]
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}[column sep=huge, row sep=large]
T \arrow[r, bend right, "\mathsf{out}"'] \arrow[r, phantom, "\cong"] & \mathbf{F}\ T \arrow[l, bend right, "\mathsf{in}"'] \\
C \arrow[u, "k"] \arrow[r, bend right, "g"'] & \mathbf{F}\ C \arrow[u, "\mathbf{F}\ k"']
\end{tikzcd}$$

$\kappa\alpha\tau\alpha$ (“cata”)

Compare

$$
\begin{tikzcd}[column sep=huge, row sep=large]
T \arrow[d, "k"'] \arrow[r, bend right, "\mathsf{out}"'] \arrow[r, phantom, "\cong"] & \mathsf{F}\ T \arrow[l, bend right, "\mathsf{in}"'] \arrow[d, "\mathsf{F}\ k"] \\
C & \mathsf{F}\ C \arrow[l, bend left, "g"]
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}[column sep=huge, row sep=large]
T \arrow[r, bend right, "\mathsf{out}"'] \arrow[r, phantom, "\cong"] & \mathsf{F}\ T \arrow[l, bend right, "\mathsf{in}"'] \\
C \arrow[u, "k"] \arrow[r, bend right, "g"'] & \mathsf{F}\ C \arrow[u, "\mathsf{F}\ k"']
\end{tikzcd}
$$

$\kappa\alpha\tau\alpha$ (“cata”) $\alpha\nu\alpha$ (“ana”),

Anamorphisms

Anamorphisms

$$\begin{tikzcd} T \arrow[rr, bend right, "\mathsf{out}"'] \arrow[rr, phantom, "\cong"] & & \mathbf{F}\ T \arrow[ll, bend right, "\mathsf{in}"'] \\ C \arrow[u, "k"] \arrow[rr, bend right, "g"'] & & \mathbf{F}\ C \arrow[u, "\mathbf{F}\ k"'] \end{tikzcd}$$

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

Anamorphisms

Ana-universal	$k = [\!(g)\!] \;\Leftrightarrow\; \mathbf{out} \cdot k = \mathsf{F}\, k \cdot g$
Ana-cancellation	$\mathbf{out} \cdot [\!(g)\!] = \mathsf{F}\, [\!(g)\!] \cdot g$
Ana-reflexion	$[\!(\mathbf{out})\!] = id$
Ana-fusion	$[\!(g)\!] \cdot f = [\!(h)\!] \;\Leftarrow\; g \cdot f = (\mathsf{F}\, f) \cdot h$

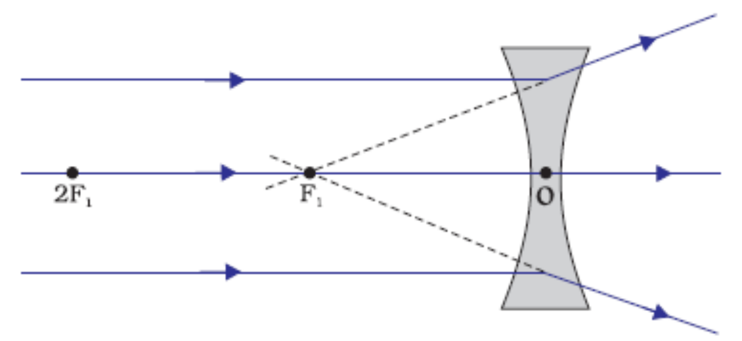
Anamorphisms

Ana-universal	$k = [\!(g)\!] \;\Leftrightarrow\; \mathbf{out} \cdot k = \mathsf{F}\, k \cdot g$
Ana-cancellation	$\mathbf{out} \cdot [\!(g)\!] = \mathsf{F}\, [\!(g)\!] \cdot g$
Ana-reflexion	$[\!(\mathbf{out})\!] = \mathit{id}$
Ana-fusion	$[\!(g)\!] \cdot f = [\!(h)\!] \;\Leftarrow\; g \cdot f = (\mathsf{F}\, f) \cdot h$

Execution (cf. cancellation):

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

Anamorphisms



$[\!(\,\_\,\_\,)\!]$

Example

$$\begin{tikzcd}
\mathbb{N}_0{}^{\star} & & 1 + \mathbb{N}_0 \times \mathbb{N}_0{}^{\star} \arrow[ll, "\mathbf{in}"'] \\
\mathbb{N}_0 \arrow[u, "k"] \arrow[r, "\mathbf{out}_{\mathbb{N}0}"'] & 1 + \mathbb{N}_0 \arrow[r, "id + \langle succ, id \rangle"'] & 1 + \mathbb{N}_0 \times \mathbb{N}_0 \arrow[u, "id + id \times k"']
\end{tikzcd}$$

$$k = [\![(id + \langle succ, id \rangle) \cdot \mathbf{out}_{\mathbb{N}0}]\!]$$

Example

$$k = [\![(id + \langle succ, id \rangle) \cdot \mathbf{out}_{\mathbb{N}0}]\!]$$

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

$$k = \mathbf{in} \cdot (id + id \times k) \cdot (id + \langle succ, id \rangle) \cdot \mathbf{out}_{\mathbb{N}0}$$

$$\Leftrightarrow \quad \{ \text{ isomorphism } \mathbf{in}_{\mathbb{N}0} \,/\, \mathbf{out}_{\mathbb{N}0} \}$$

$$k \cdot \mathbf{in}_{\mathbb{N}0} = \mathbf{in} \cdot (id + id \times k) \cdot (id + \langle succ, id \rangle)$$

$$\Leftrightarrow \quad \{ \text{ functor-}+\text{; absorption-}\times \}$$

$$k \cdot \mathbf{in}_{\mathbb{N}0} = \mathbf{in} \cdot (id + \langle succ, k \rangle)$$

Example

$$\Leftrightarrow \quad \{ \text{ definitions of } \mathbf{in} \text{ and } \mathbf{in}_{\mathbb{N}0}\text{; fusion and absorption-}+ \}$$

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

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

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

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

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

Example

$$
\begin{tikzcd}
\mathbb{N}_0{}^{\star} & & & 1 + \mathbb{N}_0 \times \mathbb{N}_0{}^{\star} \arrow[lll, "\mathbf{in}"'] \\
\mathbb{N}_0 \arrow[u, "k"] \arrow[r, "\mathbf{out}_{\mathbb{N}0}"'] & 1 + \mathbb{N}_0 \arrow[rr, "id + \langle succ, id \rangle"'] & & 1 + \mathbb{N}_0 \times \mathbb{N}_0 \arrow[u, "id + id \times k"']
\end{tikzcd}
$$

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

Compare...

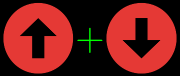
$$\begin{tikzcd}
\mathbb{N}_0{}^{\star} & & 1 + \mathbb{N}_0 \times \mathbb{N}_0{}^{\star} \arrow[ll, "\mathsf{in}"'] \\
\mathbb{N}_0 \arrow[u, "k"] \arrow[r, "\mathsf{out}_{\mathbb{N}0}"'] & 1 + \mathbb{N}_0 \arrow[r, "id + \langle succ, id \rangle"'] & 1 + \mathbb{N}_0 \times \mathbb{N}_0 \arrow[u, "id + id \times k"']
\end{tikzcd}$$

Compare...

$$\begin{tikzcd}
\mathbb{N}_0^\star & & & 1 + \mathbb{N}_0 \times \mathbb{N}_0^\star \arrow[lll, "\mathsf{in}"'] \\
\mathbb{N}_0 \arrow[u, "k"] \arrow[r, "\mathsf{out}_{\mathbb{N}0}"'] & 1 + \mathbb{N}_0 \arrow[rr, "id + \langle succ, id \rangle"'] & & 1 + \mathbb{N}_0 \times \mathbb{N}_0 \arrow[u, "id + id \times k"']
\end{tikzcd}$$

$$\begin{tikzcd}
\mathbb{N}_0^\star \arrow[d, "m"'] & & 1 + \mathbb{N}_0 \times \mathbb{N}_0^\star \arrow[ll, "\mathsf{in}"'] \arrow[d, "id + id \times m"] \\
\mathbb{N}_0 & & 1 + \mathbb{N}_0 \times \mathbb{N}_0 \arrow[ll, "{[\underline{1}, mul]}"']
\end{tikzcd}$$

… combine …



$$\begin{tikzcd}
\mathbb{N}_0 & & 1 + \mathbb{N}_0 \times \mathbb{N}_0 \arrow[ll, "{[\underline{1}, mul]}"'] \\
\mathbb{N}_0^{\star} \arrow[u, "m"] & & 1 + \mathbb{N}_0 \times \mathbb{N}_0^{\star} \arrow[ll, "\mathsf{in}"'] \arrow[u, "id + id \times m"'] \\
\mathbb{N}_0 \arrow[u, "k"] \arrow[r, "\mathsf{out}_{\mathbb{N}0}"'] & 1 + \mathbb{N}_0 \arrow[r, "id + \langle succ, id \rangle"'] & 1 + \mathbb{N}_0 \times \mathbb{N}_0 \arrow[u, "id + id \times k"']
\end{tikzcd}$$

… and calculate

$$f = m \cdot k$$

$$\Leftrightarrow \quad \{ \; m = (\![[\underline{1}, mul]]\!) \text{ and } k = [\!((id + \langle succ, id \rangle) \cdot \mathbf{out}_{\mathbb{N}0})\!] \; \}$$

$$f = (\![[\underline{1}, mul]]\!) \cdot [\!((id + \langle succ, id \rangle) \cdot \mathbf{out}_{\mathbb{N}0})\!]$$

$$\Leftrightarrow \quad \{ \; \text{cancellations (ana and cata)} \; \}$$

$$f = [\underline{1}, mul] \cdot \mathbf{F}\ m \cdot \mathbf{out} \cdot \mathbf{in} \cdot \mathbf{F}\ k \cdot (id + \langle succ, id \rangle) \cdot \mathbf{out}_{\mathbb{N}0}$$

$$\Leftrightarrow \quad \{ \; \mathbf{in} \cdot out = id \;;\; \text{functor } \mathbf{F}\text{: } (\mathbf{F}\ m) \cdot (\mathbf{F}\ k) = \mathbf{F}\ (m \cdot k) \; \}$$

$$f = [\underline{1}, mul] \cdot \mathbf{F}\ (m \cdot k) \cdot (id + \langle succ, id \rangle) \cdot \mathbf{out}_{\mathbb{N}0}$$

$$\Leftrightarrow \quad \{ \; \text{isomorphism } \mathbf{in}_{\mathbb{N}0} \,/\, \mathbf{out}_{\mathbb{N}0};\ m \cdot k = f \;;\; \mathbf{F}\ f = id + id \times f \; \}$$

… and calculate

$$f \cdot \mathbf{in}_{\mathbb{N}0} = [\underline{1}, mul] \cdot (id + id \times f) \cdot (id + \langle succ, id \rangle)$$

$$\Leftrightarrow \quad \{ \text{ +-absorption ; ×-absorption ; etc } \}$$

$$f \cdot \mathbf{in}_{\mathbb{N}0} = [\underline{1}, mul \cdot \langle succ, f \rangle]$$

$$\Leftrightarrow \quad \{ \text{ Eq-+ ; } \mathbf{in}_{\mathbb{N}0} = [\underline{0}, succ] \}$$

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

$$\Leftrightarrow \quad \{ \text{ go pointwise } \}$$

… and calculate

$$\Leftrightarrow \qquad \{ \text{ go pointwise } \}$$

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

... and calculate

$$\Leftrightarrow \qquad \{ \text{ go pointwise } \}$$

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

Function $f = m \cdot k$ is the **factorial** function:

$$fac = (\![\underline{1}, mul]\!) \cdot [\![(id + \langle succ, id \rangle) \cdot \mathbf{out}_{\mathbb{N}0}]\!]$$

... and calculate

$$\Leftrightarrow \qquad \{ \text{ go pointwise } \}$$

$$\left\{ \begin{array}{l} f\ 0 = 1 \\ f\ (n+1) = (n+1) \times f\ n \end{array} \right.$$

Function $f = m \cdot k$ is the **factorial** function:

$$fac = (\!| [\underline{1}, mul] |\!) \cdot [\!((id + \langle succ, id \rangle) \cdot \mathbf{out}_{\mathbb{N}0})\!]$$

— an **anamorphism** composed with a **catamorphism**.

Hylomorphisms

Hylomorphism

The composition of an **anamorphism** with a **catamorphism** is called a **hylomorphism**.

The following notation is normally used:

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

Hylomorphism

The composition of an **anamorphism** with a **catamorphism** is called a **hylomorphism**.

The following notation is normally used:

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

"**Hylo** + morphism" — terminology derived from the Greek $\upsilon\lambda o\sigma$ (hylos) = "matter".

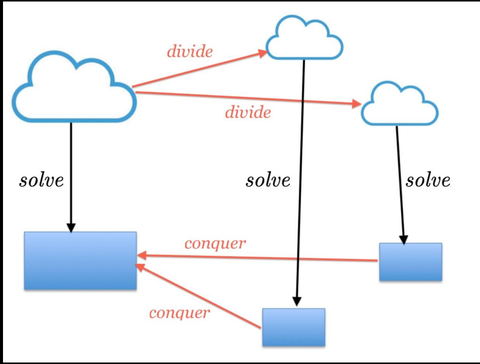
Hylomorphism

$$
\begin{tikzcd}
A \arrow[r, bend left, "g"] \arrow[d, "{[\!(g)\!]}"'] & \mathbf{F}\,A \arrow[d, "\mathbf{F}\,{[\!(g)\!]}"] \\
T \arrow[r, bend left, "\mathsf{out}"] \arrow[r, phantom, "\cong"] \arrow[d, "(\!|f|\!)"'] & \mathbf{F}\,T \arrow[l, bend left, "\mathsf{in}"] \arrow[d, "\mathbf{F}\,(\!|f|\!)"] \\
B & \mathbf{F}\,B \arrow[l, bend left, "f"]
\end{tikzcd}
$$

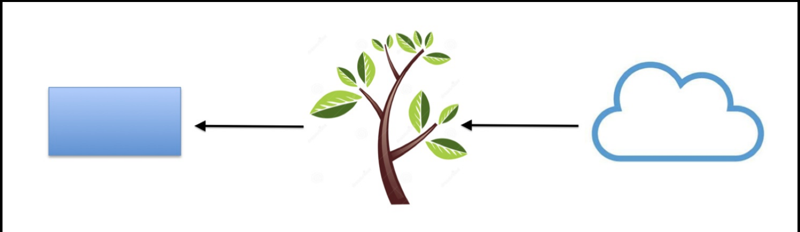
$$[\![f, g]\!] = (\!|f|\!) \cdot [\!(g)\!]$$

Hylomorphism = ‘divide & conquer’

$$\begin{tikzcd}
\mathbf{A} \arrow[r, bend left, "\mathit{divide}"] \arrow[d, "{[\!(\mathit{divide})\!]}"'] & \mathsf{F}\,A \arrow[d, "{\mathsf{F}\,[\!(\mathit{divide})\!]}"] \\
T \arrow[r, bend left, "\mathsf{out}"] \arrow[r, phantom, "\cong"] \arrow[d, "{(\!|\,\mathit{conquer}\,|\!)}"'] & \mathsf{F}\,T \arrow[l, bend left, "\mathsf{in}"] \arrow[d, "{\mathsf{F}\,(\!|\,\mathit{conquer}\,|\!)}"] \\
B & \mathsf{F}\,B \arrow[l, bend left, "\mathit{conquer}"]
\end{tikzcd}$$



Hylomorphism = ‘divide & conquer’



$$C \xleftarrow{(\!|\, \mathit{conquer} \,|\!)} \mathsf{T} \xleftarrow{[\!(\, \mathit{divide} \,)\!]} A$$

Ana, cata & hylo trilogy

$$C \xleftarrow{(\!| f |\!)} \mathbf{T} \xleftarrow{[\!(dg)\!]} A$$

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

Ana, cata & hylo trilogy

$$C \xleftarrow{(\!| f |\!)} \mathbf{T} \xleftarrow{[\!(dg)\!]} A$$

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

By the reflexion laws,

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

Ana, cata & hylo trilogy

$$C \xleftarrow{(\!| f |\!)} \mathbf{T} \xleftarrow{[\!(dg)\!]} A$$

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

By the reflexion laws,

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

we have:

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

The hylomorphism equation

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

$$
\begin{tikzcd}
A \arrow[rr, bend left, "g"] \arrow[d, "{[\!(g)\!]}"'] & & \mathbf{F}\,A \arrow[d, "\mathbf{F}\,{[\!(g)\!]}"] \\
T \arrow[rr, bend left, "\mathsf{out}"] \arrow[d, "{(\!| f |\!)}"'] & \cong & \mathbf{F}\,T \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "\mathbf{F}\,{(\!| f |\!)}"] \\
B & & \mathbf{F}\,B \arrow[ll, bend left, "f"]
\end{tikzcd}
$$

Then is the following equation holds:

$$k = f \cdot \mathbf{F}\,k \cdot g$$

Every algorithm is a hylomorphism

Example already seen:

$$\mathbb{N}_0 \xleftarrow{(\!|\, conquer \,|\!)} \mathbb{N}_0^{\star} \xleftarrow{[\!(\, divide \,)\!]} \mathbb{N}_0$$

$$fac = [\![conquer, divide]\!] = (\!|\, conquer \,|\!) \cdot [\!(\, divide \,)\!]$$

for

$$divide = (id + \langle succ, id \rangle) \cdot \mathbf{out}_{\mathbb{N}0}$$
$$conquer = [\underline{1}, mul]$$

Every algorithm is a hylomorphism

We shall see several more examples of **algorithms** expressed as **hylomorphisms**.

Every algorithm is a hylomorphism

We shall see several more examples of **algorithms** expressed as **hylomorphisms**.

We will also see how to decompose any **algorithm** in its **ana** + **cata** part.

Every algorithm is a hylomorphism

We shall see several more examples of **algorithms** expressed as **hylomorphisms**.

We will also see how to decompose any **algorithm** in its **ana** + **cata** part.

In order to do that we need to study a very important topic:

Parametric inductive types

Parametric Inductive Types

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, how they become **functors**.

Inductive parametric types

Let us now show how recursive types become **parametric**.

That is, how they become **functors**.

It all amounts to **generalize** *map* (lists) to other recursive types.

Recall...

Functor $\mathbf{F}$:

$$\begin{tikzcd}
A \arrow[r, dotted, no head] \arrow[d, "f"'] & \mathbf{F}\,A \arrow[d, "\mathbf{F}\,f"] \\
B \arrow[r, dotted, no head] & \mathbf{F}\,B
\end{tikzcd}$$

Recall...

Functor $\mathbf{F}$:

$$\begin{tikzcd} A \arrow[r, dotted, no head] \arrow[d, "f"'] & \mathbf{F}\,A \arrow[d, "\mathbf{F}\,f"] \\ B \arrow[r, dotted, no head] & \mathbf{F}\,B \end{tikzcd}$$

$$\begin{cases} \mathbf{F}\ id = id \\ \mathbf{F}\,(g \cdot f) = (\mathbf{F}\ g) \cdot (\mathbf{F}\ f) \end{cases}$$

map f

Functor $\mathbf{F}\ A = A^\star$:

$$\begin{tikzcd}
A \arrow[r, dotted, no head] \arrow[d, "f"'] & A^\star \arrow[d, "f^\star = map\ f"] \\
B \arrow[r, dotted, no head] & B^\star
\end{tikzcd}$$

$map\ f$

Functor $\mathbf{F}\ A = A^\star$:

$$
\begin{tikzcd}
A \arrow[r, dotted, dash] \arrow[d, "f"'] & A^\star \arrow[d, "f^\star = map\ f"] \\
B \arrow[r, dotted, dash] & B^\star
\end{tikzcd}
$$

$$
\begin{cases}
map\ id = id \\
map\ (g \cdot f) = (map\ g) \cdot (map\ f)
\end{cases}
$$

How do we **prove** this? How do we **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):

$$map\ f\ [\,] = [\,]$$
$$map\ f\ (h : t) = f\ h : map\ f\ t$$

$map\ f$

The *implementation*

$$map\ f\ [\,] = [\,]$$
$$map\ f\ (h : t) = f\ h : map\ f\ t$$

$map\ f$

The *implementation*

$$map\ f\ [\,] = [\,]$$
$$map\ f\ (h : t) = f\ h : map\ f\ t$$

is the catamorphism:

$$map\ f = (\!|\,[nil, g_2]\,|\!)\ \textbf{where}$$
$$g_2\ (h, x) = f\ h : x$$

map f

The *implementation*

$$map\ f\ [\,] = [\,]$$
$$map\ f\ (h : t) = f\ h : map\ f\ t$$

is the catamorphism:

$$map\ f = (\![[nil, g_2]]\!) \ \textbf{where}$$
$$g_2\ (h, x) = f\ h : x$$

Pointfree equivalent: $map\ f = (\![[nil, cons] \cdot (id + f \times id)]\!)$

$map\ f$

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 $sq\ x = x^2$

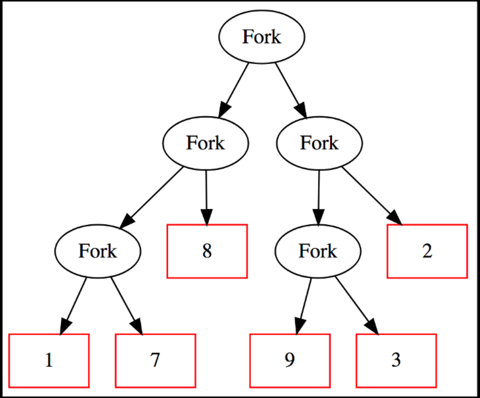
$$map\ sq\ [1, 7, 8, 9, 3, 2] = [1, 49, 64, 81, 9, 4]$$

But still we haven't proved

$$map\ id = id$$
$$map\ (f \cdot g) = (map\ f) \cdot (map\ g)$$

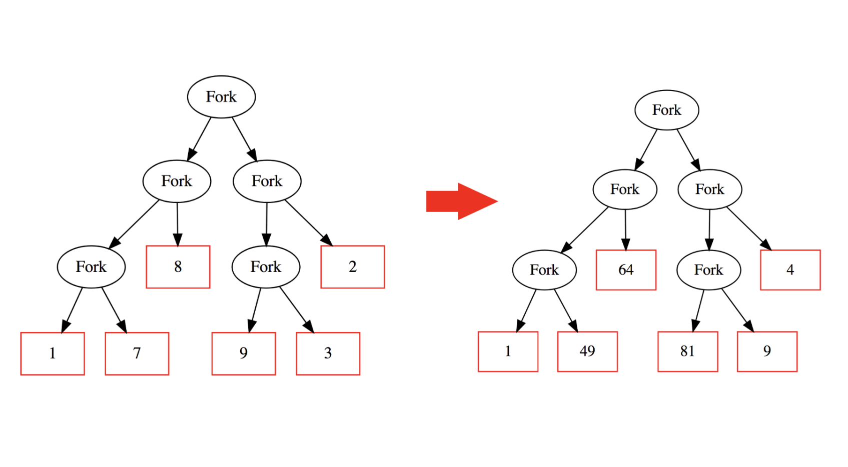
(Coming up soon.)

Inductive parametric types



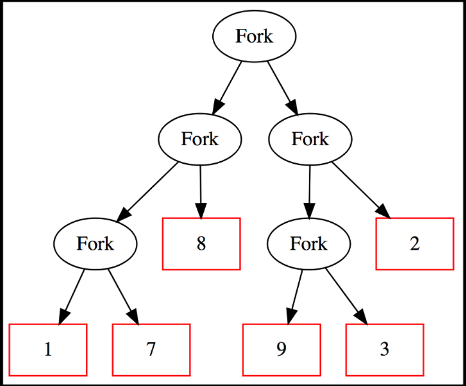
$\texttt{data}$ **LTree** $=$ **Leaf** *Int* $|$ **Fork** (**LTree**, **LTree**)

Inductive parametric types



Inductive parametric types

$$
\begin{aligned}
t = {} & \mathsf{Fork} \\
& \quad (\mathsf{Fork} \\
& \qquad (\mathsf{Fork}\ (\mathsf{Leaf}\ 1, \mathsf{Leaf}\ 7), \\
& \qquad\quad \mathsf{Leaf}\ 8), \\
& \quad \mathsf{Fork} \\
& \qquad\quad (\mathsf{Fork}\ (\mathsf{Leaf}\ 9, \mathsf{Leaf}\ 3), \\
& \qquad\qquad \mathsf{Leaf}\ 2))
\end{aligned}
$$



Inductive parametric types

$$k\ (\mathbf{Leaf}\ a) = \mathbf{Leaf}\ a^2$$
$$k\ (\mathbf{Fork}\ (l, r)) = \mathbf{Fork}\ (k\ l, k\ r)$$

$$\begin{aligned} k\ t = {} & \mathbf{Fork} \\ & \quad (\mathbf{Fork} \\ & \quad\quad (\mathbf{Fork}\ (\mathbf{Leaf}\ 1, \mathbf{Leaf}\ 49), \\ & \quad\quad \mathbf{Leaf}\ 64), \\ & \quad \mathbf{Fork} \\ & \quad\quad (\mathbf{Fork}\ (\mathbf{Leaf}\ 81, \mathbf{Leaf}\ 9), \\ & \quad\quad \mathbf{Leaf}\ 4)) \end{aligned}$$

Inductive parametric types

$$k\ (\textbf{Leaf}\ a) = \textbf{Leaf}\ a^2$$
$$k\ (\textbf{Fork}\ (l, r)) = \textbf{Fork}\ (k\ l, k\ r)$$

Inductive parametric types

$$k\ (\mathbf{Leaf}\ a) = \mathbf{Leaf}\ a^2$$
$$k\ (\mathbf{Fork}\ (l, r)) = \mathbf{Fork}\ (k\ l, k\ r)$$

Generalize a^2 to $f\ a$, by analogy with $map\ f$:

$$k\ f\ (\mathbf{Leaf}\ a) = \mathbf{Leaf}\ (f\ a)$$
$$k\ f\ (\mathbf{Fork}\ (l, r)) = \mathbf{Fork}\ (k\ f\ l, k\ f\ r)$$

Inductive parametric types

$$map\ id = id$$
$$map\ (f \cdot g) = (map\ f) \cdot (map\ g)$$

Inductive parametric types

$$map\ id = id$$
$$map\ (f \cdot g) = (map\ f) \cdot (map\ g)$$

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

?

Cálculo de Programas

Class T10

Inductive parametric types

$$\begin{cases} k\ f\ (\mathbf{Leaf}\ a) = \mathbf{Leaf}\ (f\ \ a) \\ k\ f\ (\mathbf{Fork}\ (l, r)) = \mathbf{Fork}\ (k\ f\ l, k\ f\ r) \end{cases}$$

Inductive parametric types

$$\begin{cases} k\ f\ (\mathsf{Leaf}\ a) = \mathsf{Leaf}\ (f\ \ a) \\ k\ f\ (\mathsf{Fork}\ (l, r)) = \mathsf{Fork}\ (k\ f\ l, k\ f\ r) \end{cases}$$

$$\Leftrightarrow \qquad \{ \text{ go pointfree } \}$$

$$\begin{cases} k\ f \cdot \mathsf{Leaf} = \mathsf{Leaf} \cdot f \\ k\ f \cdot \mathsf{Fork} = \mathsf{Fork} \cdot (k\ f \times k\ f) \end{cases}$$

Inductive parametric types

$$\left\{\begin{array}{l} k\ f\ (\mathbf{Leaf}\ a) = \mathbf{Leaf}\ (f\ \ a) \\ k\ f\ (\mathbf{Fork}\ (l, r)) = \mathbf{Fork}\ (k\ f\ l, k\ f\ r) \end{array}\right.$$

$$\Leftrightarrow \qquad \{ \text{ go pointfree } \}$$

$$\left\{\begin{array}{l} k\ f \cdot \mathbf{Leaf} = \mathbf{Leaf} \cdot f \\ k\ f \cdot \mathbf{Fork} = \mathbf{Fork} \cdot (k\ f \times k\ f) \end{array}\right.$$

$$\Leftrightarrow \qquad \{ +\text{-Eq} \ ; \ +\text{-fusion} \ ; \ +\text{-absorption} \}$$

Inductive parametric types

$$k\ f \cdot [\mathsf{Leaf}, \mathsf{Fork}] = [\mathsf{Leaf}, \mathsf{Fork}] \cdot (f + k\ f \times k\ f)$$

Inductive parametric types

$$k\ f \cdot [\mathsf{Leaf}, \mathsf{Fork}] = [\mathsf{Leaf}, \mathsf{Fork}] \cdot (f + k\ f \times k\ f)$$

$$\Leftrightarrow \qquad \{\ [\mathsf{Leaf}, \mathsf{Fork}] = \mathsf{in}\ ;\ \text{functor-}+\ \}$$

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

Inductive parametric types

$$\begin{tikzcd}
Int \arrow[d, "f"'] & \mathbf{LTree} \arrow[d, "k\ f"'] & & Int + \mathbf{LTree}^2 \arrow[ll, "\mathbf{in}"'] \arrow[d, "id+(k\ f)^2"] \\
Int & \mathbf{LTree} & & Int + \mathbf{LTree}^2 \arrow[ll, "g"] \arrow[dl, "f+id"] \\
& & Int + \mathbf{LTree}^2 \arrow[ul, "\mathbf{in}"] &
\end{tikzcd}$$

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

NB: $\mathbf{LTree}^2$ abbreviates $\mathbf{LTree} \times \mathbf{LTree}$

Inductive parametric types

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

equivalent to

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

by the cata-universal property.

Generalizing *Int*...

$$
\begin{tikzcd}
A \arrow[d, "f"'] & \mathbf{LTree} \arrow[d, "k\ f"'] & & A + \mathbf{LTree}^2 \arrow[ll, "\mathbf{in}"'] \arrow[d, "id + (k\ f)^2"] \\
A & \mathbf{LTree} & & A + \mathbf{LTree}^2 \arrow[ll, "g"] \arrow[dl, "f + id"] \\
& & A + \mathbf{LTree}^2 \arrow[ul, "\mathbf{in}"] &
\end{tikzcd}
$$

Generalizing A...

$$
\begin{tikzcd}
A \arrow[d, "f"'] & \mathbf{LTree} \arrow[d, "k\ f"'] & & A + \mathbf{LTree}^2 \arrow[ll, "\mathbf{in}"'] \arrow[d, "id + (k\ f)^2"] \\
B & \mathbf{LTree} & & A + \mathbf{LTree}^2 \arrow[ll, "g"] \arrow[dl, "f + id"] \\
 & & B + \mathbf{LTree}^2 \arrow[ul, "\mathbf{in}"] &
\end{tikzcd}
$$

Generalizing A...

$$\begin{tikzcd}
A \arrow[d, "f"'] & \mathsf{LTree} \arrow[d, "k\ f"'] & & A + \mathsf{LTree}^2 \arrow[ll, "\mathsf{in}"'] \arrow[d, "id + (k\ f)^2"] \\
B & \mathsf{LTree} & & A + \mathsf{LTree}^2 \arrow[ll, "g"] \arrow[dl, "f + id"] \\
& & B + \mathsf{LTree}^2 \arrow[ul, "\mathsf{in}"] &
\end{tikzcd}$$

... does not type **!!**

Inductive parametric types

$$\begin{tikzcd}
A \arrow[d, "f"'] & \mathbf{LTree}\ A \arrow[d, "k\ f = (\!| g |\!)"'] & & A + (\mathbf{LTree}\ A)^2 \arrow[ll, "\mathbf{in}_A"'] \arrow[d, "id + (k\ f)^2"] \\
B & \mathbf{LTree}\ B & & A + (\mathbf{LTree}\ B)^2 \arrow[ll, "g"] \arrow[dl, "f + id"] \\
& & B + (\mathbf{LTree}\ B)^2 \arrow[ul, "\mathbf{in}_B"] &
\end{tikzcd}$$

Inductive parametric types

$$
\begin{tikzcd}
A \arrow[d, "f"'] & \mathsf{LTree}\ A \arrow[d, "k\ f = (\!|\, g \,|\!)"'] & & A + (\mathsf{LTree}\ A)^2 \arrow[ll, "\mathbf{in}_A"'] \arrow[d, "id + (k\ f)^2"] \\
B & \mathsf{LTree}\ B & & A + (\mathsf{LTree}\ B)^2 \arrow[ll, "g"] \arrow[dl, "f + id"] \\
 & & B + (\mathsf{LTree}\ B)^2 \arrow[ul, "\mathbf{in}_B"] &
\end{tikzcd}
$$

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

LTree type functor

$$\begin{tikzcd} A \arrow[r, dotted, no head] \arrow[d, "f"'] & \mathsf{LTree}\ A \arrow[d, "\mathsf{LTree}\ f \ = \ (\!|\, \mathsf{in} \cdot (f + id) \,|\!)"] \\ B \arrow[r, dotted, no head] & \mathsf{LTree}\ B \end{tikzcd}$$

LTree type functor

$$\begin{tikzcd} \mathbf{LTree}\ A \arrow[rr, bend left, "\mathsf{out}"] \arrow[rr, phantom, "\cong"] & & \underbrace{A + (\mathbf{LTree}\ A)^2}_{\mathbf{F}\ (\mathbf{LTree}\ A)} \arrow[ll, bend left, "\mathsf{in}"] \end{tikzcd}$$

$$\begin{tikzcd} \mathbf{LTree}\ B \arrow[rr, bend left, "\mathsf{out}"] \arrow[rr, phantom, "\cong"] & & \underbrace{B + (\mathbf{LTree}\ B)^2}_{\mathbf{F}\ (\mathbf{LTree}\ B)} \arrow[ll, bend left, "\mathsf{in}"] \end{tikzcd}$$

LTree type functor

$$\mathbf{LTree}\ A \underset{\mathsf{in}}{\overset{\mathsf{out}}{\cong}} \underbrace{A + (\mathbf{LTree}\ A)^2}_{\mathbf{F}\ (\mathbf{LTree}\ A)}$$

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

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

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

LTree base bifunctor

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

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

Bifunctors

$$
\begin{tikzcd}
A \arrow[r, dash, dotted] \arrow[d, "f"'] & C \arrow[r, dash, dotted] \arrow[d, "g"'] & \mathbf{B}(A, C) \arrow[d, "\mathbf{B}(f, g)"] \\
D \arrow[r, dash, dotted] & E \arrow[r, dash, dotted] & \mathbf{B}(D, E)
\end{tikzcd}
$$

Bifunctors

$$\begin{tikzcd}
A \arrow[r, dotted, no head] \arrow[d, "f"'] & C \arrow[r, dotted, no head] \arrow[d, "g"'] & \mathbf{B}\,(A, C) \arrow[d, "\mathbf{B}\,(f, g)"] \\
D \arrow[r, dotted, no head] & E \arrow[r, dotted, no head] & \mathbf{B}\,(D, E)
\end{tikzcd}$$

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

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

Bifunctors (example)

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

Bifunctors (example)

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

$$\begin{tikzcd}
A \arrow[r, dotted, no head] \arrow[d, "f"'] & C \arrow[r, dotted, no head] \arrow[d, "g"'] & A \times C \arrow[d, "f \times g"] \\
D \arrow[r, dotted, no head] & E \arrow[r, dotted, no head] & D \times E
\end{tikzcd}$$

Bifunctors (example)

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

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

$$
\begin{tikzcd}
A \arrow[r, dotted, no head] \arrow[d, "f"'] & C \arrow[r, dotted, no head] \arrow[d, "g"'] & A \times C \arrow[d, "f \times g"] \\
D \arrow[r, dotted, no head] & E \arrow[r, dotted, no head] & D \times E
\end{tikzcd}
$$

$$id \times id = id$$

$$h \cdot f \times k \cdot g = h \times k \cdot f \times g$$

Catamorphisms go parametric

$$
\begin{tikzcd}
T \arrow[rr, bend left, "\mathbf{out}"] \arrow[rr, phantom, "\cong"] \arrow[d, "k"'] & & \mathbf{F}\ T \arrow[ll, bend left, "\mathbf{in}"] \arrow[d, "\mathbf{F}\ k"] \\
C & & \mathbf{F}\ C \arrow[ll, bend left, "g"]
\end{tikzcd}
$$

Catamorphisms go parametric

$$
\begin{tikzcd}
T \arrow[rr, bend left, "\mathbf{out}"] \arrow[d, "k"'] \arrow[rr, phantom, "\cong"] & & \mathbf{F}\ T \arrow[ll, bend left, "\mathbf{in}"] \arrow[d, "\mathbf{F}\ k"] \\
C & & \mathbf{F}\ C \arrow[ll, bend left, "g"]
\end{tikzcd}
\quad \mapsto \quad
\begin{tikzcd}
\mathbf{T}\ A \arrow[rr, bend left, "\mathbf{out}"] \arrow[d, "k = (\!|\, g \,|\!)"'] \arrow[rr, phantom, "\cong"] & & \mathbf{B}\ (A, \mathbf{T}\ A) \arrow[ll, bend left, "\mathbf{in}"] \arrow[d, "\mathbf{B}\ (id, k)"] \\
C & & \mathbf{B}\ (A, C) \arrow[ll, bend left, "g"]
\end{tikzcd}
$$

Catamorphisms go parametric

$$
\begin{tikzcd}[column sep=huge, row sep=large]
T \arrow[r, bend left, "\mathbf{out}"] \arrow[d, "k"'] \arrow[r, phantom, "\cong"] & \mathbf{F}\ T \arrow[l, bend left, "\mathbf{in}"] \arrow[d, "\mathbf{F}\ k"] \\
C & \mathbf{F}\ C \arrow[l, bend left, "g"]
\end{tikzcd}
\quad \longmapsto \quad
\begin{tikzcd}[column sep=huge, row sep=large]
\mathbf{T}\ A \arrow[r, bend left, "\mathbf{out}"] \arrow[d, "k = (\!| g |\!)"'] \arrow[r, phantom, "\cong"] & \mathbf{B}\ (A, \mathbf{T}\ A) \arrow[l, bend left, "\mathbf{in}"] \arrow[d, "\mathbf{B}\ (id, k)"] \\
C & \mathbf{B}\ (A, C) \arrow[l, bend left, "g"]
\end{tikzcd}
$$

Universal property

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

Type functor

$$\begin{tikzcd}
A \arrow[d, "f"'] & \mathbf{T}\ A \arrow[d, "\mathbf{T}\ f"'] \arrow[rr, bend left, "\mathsf{out}"] \arrow[rr, phantom, "\cong"] & & \mathbf{B}\ (A, \mathbf{T}\ A) \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "\mathbf{B}\ (id, \mathbf{T}\ f)"] \\
C & \mathbf{T}\ C & & \mathbf{B}\ (A, \mathbf{T}\ C) \arrow[ll, bend left, "\mathsf{in} \cdot \mathbf{B}\ (f, id)"]
\end{tikzcd}$$

Type functor

$$
\begin{tikzcd}
A \arrow[d, "f"'] & \mathbf{T}\ A \arrow[d, "\mathbf{T}\ f"'] \arrow[rr, bend left, "\mathbf{out}"] \arrow[rr, phantom, "\cong"] & & \mathbf{B}\ (A, \mathbf{T}\ A) \arrow[ll, bend left, "\mathbf{in}"] \arrow[d, "\mathbf{B}\ (id, \mathbf{T}\ f)"] \\
C & \mathbf{T}\ C & & \mathbf{B}\ (A, \mathbf{T}\ C) \arrow[ll, bend left, "\mathbf{in} \cdot \mathbf{B}\ (f, id)"]
\end{tikzcd}
$$

That is:

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

Catamorphism laws

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

Anamorphisms go parametric

$$
\begin{tikzcd}
T \arrow[r, bend right, "\mathbf{out}"'] \arrow[r, phantom, "\cong"] & \mathbf{F}\ T \arrow[l, bend right, "\mathbf{in}"'] \\
C \arrow[u, "k = [\!(g)\!]"] \arrow[r, bend right, "g"'] & \mathbf{F}\ C \arrow[u, "\mathbf{F}\ k"']
\end{tikzcd}
$$

Anamorphisms go parametric

$$
\begin{tikzcd}
T \arrow[rr, bend left=20, "\mathbf{out}"'] \arrow[rr, phantom, "\cong"] & & \mathbf{F}\ T \arrow[ll, bend right=20, "\mathbf{in}"'] \\
C \arrow[u, "k = [\![(g)]\!]"] \arrow[rr, bend right=20, "g"'] & & \mathbf{F}\ C \arrow[u, "\mathbf{F}\ k"']
\end{tikzcd}
\quad \longmapsto \quad
\begin{tikzcd}
\mathbf{T}\ A \arrow[rr, bend left=10, "\mathbf{out}"'] \arrow[rr, phantom, "\cong"] & & \mathbf{B}\ (A, \mathbf{T}\ A) \arrow[ll, bend right=10, "\mathbf{in}"'] \\
C \arrow[u, "k = [\![(g)]\!]"] \arrow[rr, bend right=10, "g"'] & & \mathbf{B}\ (A, C) \arrow[u, "\mathbf{B}\ (id, k) = \mathbf{F}\ k"']
\end{tikzcd}
$$

Anamorphisms go parametric

$$
\begin{tikzcd}
T \arrow[rr, bend right, "\mathbf{out}"'] \arrow[rr, phantom, "\cong"] & & \mathbf{F}\ T \arrow[ll, bend right, "\mathbf{in}"'] \\
C \arrow[u, "k = [\!(g)\!]"] \arrow[rr, bend right, "g"'] & & \mathbf{F}\ C \arrow[u, "\mathbf{F}\ k"']
\end{tikzcd}
\quad \longmapsto \quad
\begin{tikzcd}
\mathbf{T}\ A \arrow[rr, bend right, "\mathbf{out}"'] \arrow[rr, phantom, "\cong"] & & \mathbf{B}\ (A, \mathbf{T}\ A) \arrow[ll, bend right, "\mathbf{in}"'] \\
C \arrow[u, "k = [\!(g)\!]"] \arrow[rr, bend right, "g"'] & & \mathbf{B}\ (A, C) \arrow[u, "\mathbf{B}\ (id, k) = \mathbf{F}\ k"']
\end{tikzcd}
$$

Universal property

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

Anamorphism laws

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

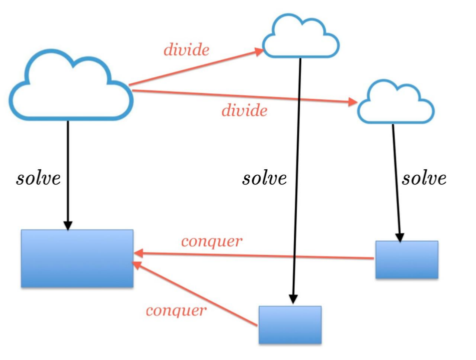
Hylomorphisms go parametric

$$
\begin{tikzcd}[row sep=large, column sep=huge]
A \arrow[r, bend left, "g"] \arrow[d, "{[\!(g)\!]}"'] & \mathbf{F}\,A \arrow[d, "{\mathbf{F}\,[\!(g)\!]}"] \\
T \arrow[r, bend left, "\mathsf{out}"] \arrow[r, phantom, "\cong"] \arrow[d, "{(\!|\, f \,|\!)}"'] & \mathbf{F}\,T \arrow[l, bend left, "\mathsf{in}"] \arrow[d, "{\mathbf{F}\,(\!|\, f \,|\!)}"] \\
C & \mathbf{F}\,C \arrow[l, bend left, "f"]
\end{tikzcd}
$$

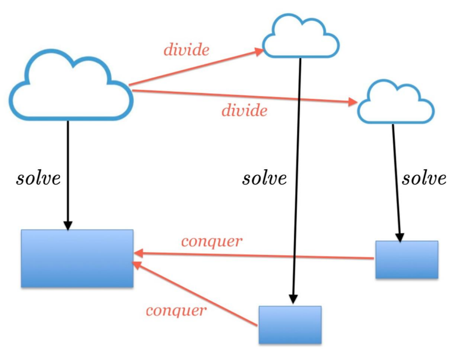
Hylomorphisms go parametric

$$
\begin{tikzcd}
A \arrow[rr, bend left, "g"] \arrow[d, "\llbracket g \rrbracket"'] & & \mathbf{F}\ A \arrow[d, "\mathbf{F}\ \llbracket g \rrbracket"] \\
T \arrow[rr, bend left, "\mathbf{out}"] \arrow[rr, phantom, "\cong"] \arrow[d, "\llparenthesis f \rrparenthesis"'] & & \mathbf{F}\ T \arrow[ll, bend left, "\mathbf{in}"] \arrow[d, "\mathbf{F}\ \llparenthesis f \rrparenthesis"] \\
C & & \mathbf{F}\ C \arrow[ll, bend left, "f"]
\end{tikzcd}
\quad \mapsto \quad
\begin{tikzcd}
A \arrow[rr, bend left, "g"] \arrow[d, "\llbracket g \rrbracket"'] & & \mathbf{B}\ (X, A) \arrow[d, "\mathbf{B}\ (id, \llbracket g \rrbracket)"] \\
T\ X \arrow[rr, bend left, "\mathbf{out}"] \arrow[rr, phantom, "\cong"] \arrow[d, "\llparenthesis f \rrparenthesis"'] & & \mathbf{B}\ (X, T\ X) \arrow[ll, bend left, "\mathbf{in}"] \arrow[d, "\mathbf{B}\ (id, \llparenthesis f \rrparenthesis)"] \\
C & & \mathbf{B}\ (X, C) \arrow[ll, bend left, "f"]
\end{tikzcd}
$$

'Divide & conquer' = Hylomorphism



'Divide & conquer' = Hylomorphism



$$
\begin{tikzcd}
A \arrow[rr, bend left, "divide"] \arrow[d, "[\![divide]\!]"'] & & \mathbf{B}\,(X, A) \arrow[d, "\mathbf{B}\,(id, [\![di"] \\
T\ X \arrow[rr, bend left, "\mathbf{out}"] \arrow[d, "(\!| conquer |\!)"'] \arrow[rr, phantom, "\cong"] & & \mathbf{B}\,(X, T\ X) \arrow[ll, bend left, "\mathbf{in}"] \arrow[d, "\mathbf{B}\,(id, (\!| co"] \\
C & & \mathbf{B}\,(X, C) \arrow[ll, bend left, "conquer"]
\end{tikzcd}
$$

Follow-up: `cp2425s12.pdf`