Where do categories come from?

- Programming languages

objects : data types ($\mathbb{N}$, $\mathbb{R}$, lists ...)

morphisms : programs $\mathbb{N} \longrightarrow \mathbb{R}$, $n \longmapsto \sqrt{n}$

composition : composition of programs.

identity maps : "output = input" program

isomorphisms : program that can "reversed"

- Type theory (typed λ-calculus)

objects : Types

morphism : terms in context

variable $x : X \vdash y(x) : Y$

= morphism $X \longrightarrow Y$

" $x \longmapsto y(x)$ "

composition : substitution

identity map : $x : X \vdash x : X$

iso : "invertible" substitutions.

- propositional calculus

ob : propositions

Mor : implication btw prop.

Compo : transitivity of implication
($A \Rightarrow B$ $B \Rightarrow C$ then $A \Rightarrow C$)

identity : reflexivity of implication
$A \Rightarrow A$

iso : logical equivalence. $\Leftrightarrow$

1st order logic

ob : formulas

Mor : functional relations

Compo : compo. of relations.

identity : equality relation

iso : not easy to say
(relations admitting an inverse)

categorical semantics for logical theories.
syntax of theory ⟶ category

examples from within mathematics

monoid = endomorphism of objects (non invertible / symmetries)

group = (invertible) symmetries of some objects

- set E $\mathrm{Aut}(E) = \mathrm{Bij}.\ E \to E$
  permutation group of E

- S^2 sphere gp of sym of S^2 is $O(3)$
  = group of rotations in space

- $\mathbb{R}^2$ plane gp translations

- - -

same for categories and groupoids
arise as symmetries (invertible or not) of a family of objects

- Set
- category of monoids
  ob: monoids
  mor: morphism of monoids
  $f: M \longrightarrow N$
  $f(a.b) = f(a).f(b)$
  can be composed
  identity: $M \xrightarrow{\text{identity fct.}} M$
  $x \longmapsto x$

- any kind of math. structure comes with a notion of morphism
  ⇝ define a category
- poset ob: poset
  mor: monotone fct
- Vect ob: vector spaces
  mor: linear maps

Categories are a mathematical structure : they have morphisms called functors

$\mathcal{C}$ cat $\quad ob(\mathcal{C})$ = set of objects

$\quad\quad\quad Arr(\mathcal{C})$ = set of morphisms.

$\mathcal{C}, \mathcal{D}$ are two cat. a functor $F : \mathcal{C} \longrightarrow \mathcal{D}$ is the data

- a fct $F_0 : ob(\mathcal{C}) \longrightarrow ob(\mathcal{D})$
- a fct $F_1 : Arr(\mathcal{C}) \longrightarrow Arr(\mathcal{D})$

satisfying : ...

- $u : x \to y$ in C

$$\left.\begin{aligned} \mathrm{dom}\left(f_1(u)\right) &= f_0\left(\mathrm{dom}(u)\right) \\ \mathrm{cod}\left(f_1(u)\right) &= f_0\left(\mathrm{cod}(u)\right) \end{aligned}\right\} \quad f_1(u) : f_0(x) \to f_0(y)$$

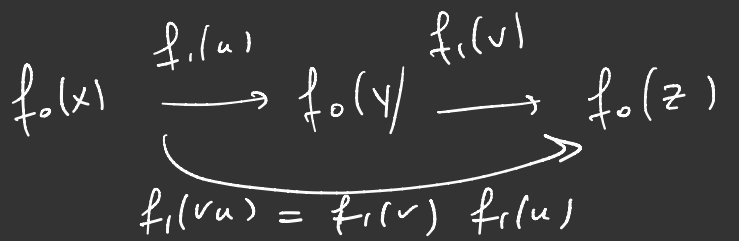
preserve domain and codomain of arrow

- $f_1(1_x) = 1_{f_0(x)}$ preserve identity arrows

- $x \xrightarrow{u} y \xrightarrow{v} z$ in C

preserve composition.

$$f_1(vu) = f_1(v)\, f_1(u)$$



monoids are particular categories (cat with single objects)

a functor $f: C \to D$ between two monoids (viewed as categories)
is the same thing as a morphism of monoids.

preorder/poset are the cat. with at most one morphism between two objects.

a functor $f: C \to D$ between two preorder (viewed as cat)
is the same thing as a monotone function.

Categories and functors do form a category !

object : categories

morph : functors

compo : exercise

identity : identity functor

$$\begin{pmatrix} f_0 = id_{ob(C)} \\ f_1 = id_{Arr(C)} \end{pmatrix}$$

} **Cat**

isomorphism of categories : $gf = id_C$, $C \xrightarrow{f} D$, $fg = id_D$, g

Examples of functors

- morph. of monoids / morph. of groups
- monotone fct between preorders
- Recall the <u>empty category</u> $\emptyset$ for any cat C
  there exists a unique functor $\emptyset \to C$

$$\left(\begin{array}{l} f_0 : \emptyset \to ob(C) \\ f_1 : \emptyset \to Arr(C) \end{array} \right)$$

- Recall the <u>punctual category</u> $\{*\}$
  for any C there exists a unique functor

$$C \to \{*\} \qquad \left(\begin{array}{l} f_0 : ob(C) \to \{*\} \\ f_1 : Arr(C) \to \{*\} \end{array} \right)$$

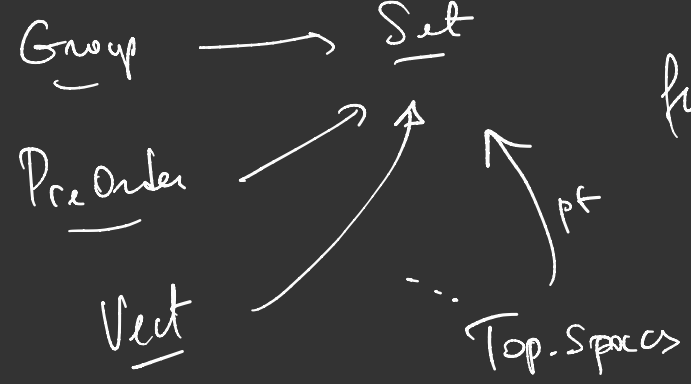
- **Mon** cat of monoids and morph of monoids

any monoid has an underlying set

$$\mathbf{Mon} \longrightarrow \mathbf{Set}$$

functor forgetting the monoid structure and extracting the underlying set.

other forgetful functors



functors extracting the underlying sets

- Free monoid. E set $\longmapsto$ $M(E)$ monoid.
  list(E)

  is a functor

M : Set $\longrightarrow$ Mon

$E \xrightarrow{f} F$

fct

$M(E) \dashrightarrow M(F)$

$x \longmapsto f(x)$

$x_1 x_2 \dots x_n \longmapsto f(x_1) f(x_2) \dots f(x_n)$

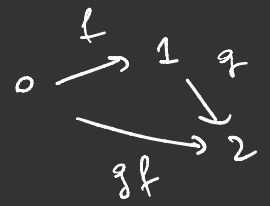
$1 \longmapsto 1$

is a morphism of monoid

Remark for later

the two functors

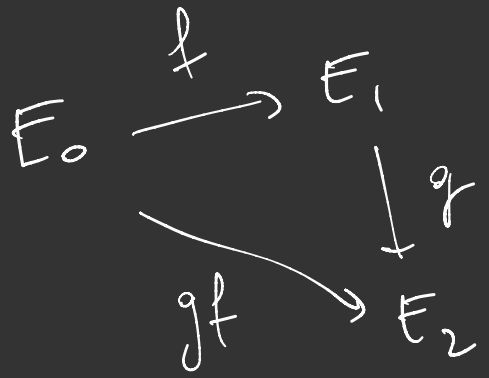
Set $\underset{\text{forget}}{\overset{\text{Free}}{\rightleftarrows}}$ Mon are an example of an adjunction.

- Recall 

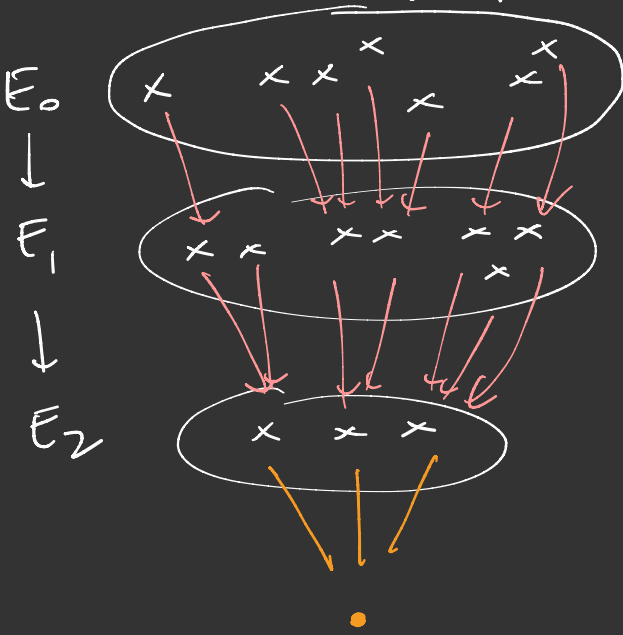
commutative triangle

denoted $[2]$

functor $[2] \longrightarrow$ Set is called a commutative triangle of sets



one way to encode trees of height 3



- $n \in \mathbb{N}$ $[n]$ cat

$Ob = \{0, 1, \dots n\}$

Mor = order relation

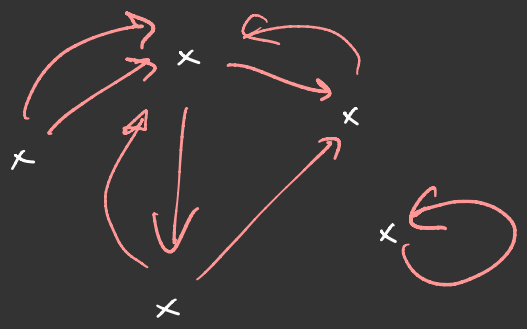
ex of a cat associated to a poset

functors $[n] \longrightarrow Set$

E_0
$\downarrow$
E_1
$\downarrow$
E_2
$\vdots$
$\downarrow$
E_n

"trees of height $n+1$"

– oriented graphs



Set of Vertices

dom ↑ ↑ cod.

Set of Edges

new category "the parallel pair"

$\underline{\mathbb{P}}$: $0 \overset{u}{\underset{v}{\rightrightarrows}} 1$

$ob = \{0, 1\}$

$Mor = \{1_0, 1_1, u, v\}$

an oriented graph is a functors

$\mathbb{P} \longrightarrow \underline{Set}$

$0 \longmapsto$ Set of edges

$1 \longmapsto$ Set of vertices.