Bartosz Milewski

⇒ free construction → adding constraints, just to satisfy axioms and not any other additional structure to form a category

or elements

Orders → Categories in which arrows (morphisms) doesn't represent functions but rather relations

Eg: $a \leq b$
↳ relation

Types of order : * preorder * partial order * total order

There is either 0 or 1 morphism from one obj. to another
↘ This kind of category is called 'THIN CATEGORY'
⇓
Mathematically,

$C(a, b) = 0$ or 1

where C is a HomSet (Set of arrows from a to b in a cat.)

It follows identity & composition AUTOMATICALLY (pre-)

'Every' objects have an order in a category

PARTIAL ORDER

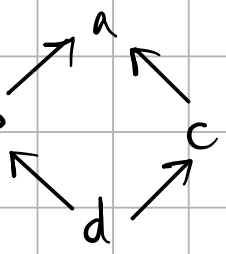
There should be only one arrow 'b/w' any two objects.

$a \longrightarrow b \longrightarrow c$

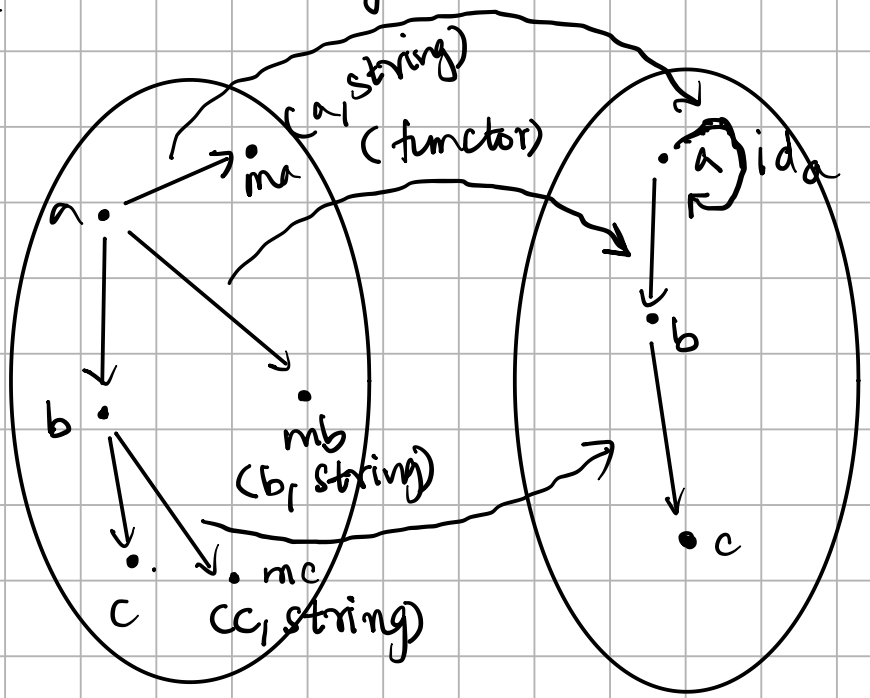
(No $b \rightarrow a$)

In pre-order $a \rightleftarrows b$ can exist as it obeys the rule

* $\leq$ category is a preorder. It is also a partial order as if $a \leq b$ then $b \leq a$ 'can' be true for a condition only ($a = b$). It is not a total order because:



$\mathcal{C}$: $\Rightarrow$ Relation b/w b & c is not needed or given but it is still a category.



$\mathcal{C}$

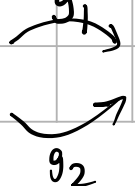
KLEISLI Category.

$a \longrightarrow ma$

This relation is one of the def. of a 'MONAD'

* UNIVERSAL CONSTRUCTION: Determining properties of obj. based only on the relationships of this obj. with others (only arrows)

like we defined monomorphism,



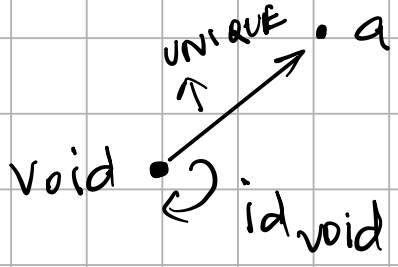
$a \xrightarrow{f} b \Rightarrow g_1 = g_2$ if f is monic

By, this constr. we have certain obj. called terminal & initial obj.

* Terminal OBJ: An object that has a 'unique' arrow from 'any' other object.

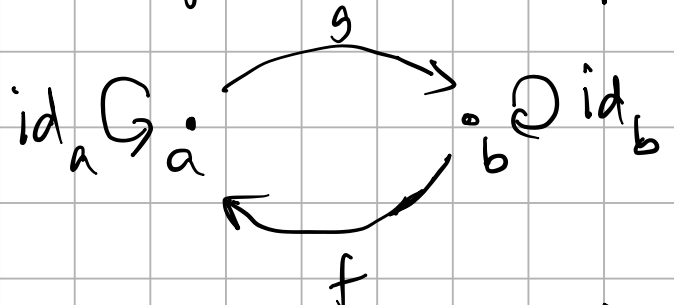
In $\leq$ category, Terminal object is the smallest no. in the category.

* Intial OBJ: In Sets, Take empty set(void)



In a category, initial obj. is an obj. which has a unique relation to all other objects (OPP. TO A TERMINAL OBJ.)

*2 Terminal obj.s are uniquely isomorphic

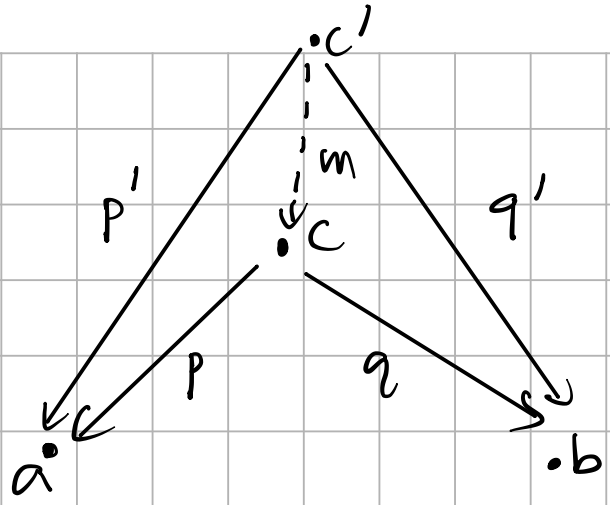


g, f exist as per def. of T.O.

id_a must exist, but g is already existing & a can't have any other morph. So, $fog = id_a$ is implicit.

* T.O. is used as a way to pick elements

Products :



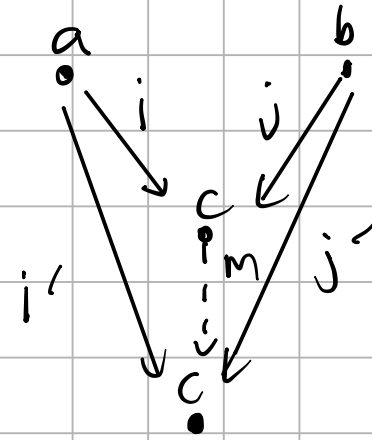
⇒ Any obj. that has 2 projection maps are not products. An actual product is say c if has projections to a & b & for any other c' which also has same proj. if there is a unique map from c' to c then c is the product.

The map $c' \xrightarrow{m} c$ must also follow $p \circ m = q \circ m$

⇒ 'm' is the extra unnecessary projection removal function so, that c' becomes c. c is a precise & complete proj.

coproducts :

i, j → embeds a & b in c like a UNION



i, j, i', j' are injections

$i' = m \circ i$

$j' = m \circ j$

c is a co-product