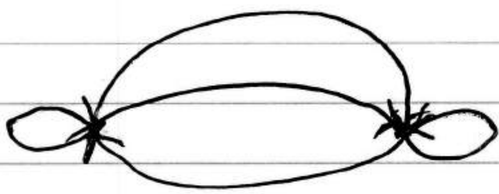
Introduction

A category is a bunch of objects. We can't call it a set, because we can't have a set of all sets. We also have arrows between objects, called morphisms.

$$a \xrightarrow{f} b$$

We are allowed zero or more arrows between objects:



These arrows are not the same.

We always have composition:

$$\begin{tikzcd} a \arrow[r, "f"'] \arrow[rr, bend left, "g \circ f"] & b \arrow[r, "g"'] & c \end{tikzcd}$$

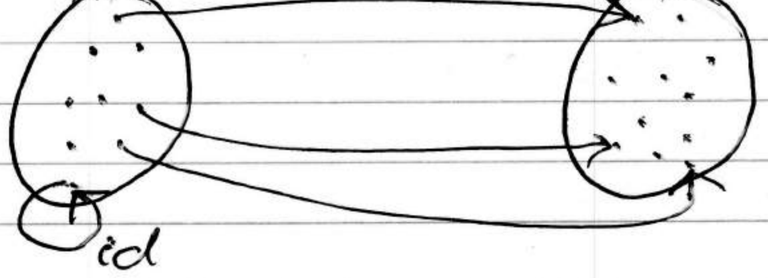
We always have identity arrows:

$$\begin{tikzcd} a \arrow[loop above, "id_a"] \arrow[r, "f"] & b \arrow[loop right, "id_b"] \end{tikzcd}$$

so $id_b \circ f = f = f \circ id_a$.

Finally, we always have associativity.

Category of Sets



the set of all sets is a bunch of objects, with functions between them & this is the category "Set".

Just because we have a function:

$$f: A \to A,$$

it does not make it an identity, but there is exactly one identity arrow for each set.

In category theory, we are not able to look inside a set, only at the arrows between them.

In programming, we have types and functions, where types are sets! By this I mean, pure and total, computable functions, otherwise we will run into some problems.

Morphisms

If we have:

$$f: A \to B \quad \text{and} \quad g: B \to A$$

such that:

$$g \circ f = id_A \quad \text{and} \quad f \circ g = id_B,$$

then we have an isomorphism:

$$\begin{tikzcd} A \arrow[loop left, "id_A"] \arrow[r, bend left, "f"] & B \arrow[l, bend left, "g"] \arrow[loop right, "id_B"] \end{tikzcd}$$

where f and g are arrows between objects.

In set theory, we have onto functions:

$$\forall y, \exists x, \ y = f(x),$$

but in Category Theory, we cannot consider elements, because we cannot "look inside objects". We have to describe this property using only arrows:

$$\forall g_1, g_2, \ g_2 \circ f = g_1 \circ f \Rightarrow g_1 = g_2$$

means that f is an _epimorphism_. Notice this means we can "cancel" f on the right!

Monomorphisms

Similarly, in set theory, we have 1-1:

$$\forall a, b,\ f(a)=f(b) \Rightarrow a=b.$$

The Category Theory variant is:

$$\forall g_1, g_2,\quad f \circ g_1 = f \circ g_2 \Rightarrow g_1 = g_2$$

means that f is a <u>monomorphism</u>, and we can cancel on the left.

Being both epic and monic does not imply an isomorphism, in general!

In the category of rings:

$$\mathbb{Z} \hookrightarrow \mathbb{Q}$$

is epic and monic, but is not an isomorphism.

(Arrows in this category are ring homomorphisms, and objects are rings).

In general, a morphism that is both epic and monic is called a <u>bimorphism</u>.

Ring Example in Detail

We claim that $\mathbb{Z} \xhookrightarrow{f} \mathbb{Q}$ is a bimorphism but not bijective.

In order to prove this, we will observe that clearly f is not onto, since $\frac{1}{2}$ is never mapped to by f, however f is both epic and monic.

Epic: Suppose we have ring homomorphisms $g_1 \neq g_2$, and $g_1 \circ f = g_2 \circ f$ for $f : a \mapsto a$.

Then: $g_1(f(a)) = g_2(f(a))$

$$\Downarrow$$

$$g_1(a) = g_2(a)$$

$$\Downarrow$$

$$g_1 = g_2 \quad \times$$

Thus, f is epic. $\square$

Monic: $g_1 \neq g_2$, $f \circ g_1 = f \circ g_2$ for $f : a \mapsto a$.

Then: $f(g_1(a)) = f(g_2(a))$

$$\Downarrow$$

$$g_1(a) = g_2(a)$$

$$\Downarrow$$

$$g_1 = g_2 \quad \times$$

Thus f is monic. $\square$

Simple types

Does the empty set correspond to a type? Well, yes: Void.

Can we have a function of type:

$$f :: \text{Void} \to \text{Int}.$$

Well, yes, since we have:

$$id_{\text{Void}} :: \text{Void} \to \text{Void}.$$

In Haskell, this function exists "in vacuum":

$$\text{absurd} :: \text{Void} \to a.$$

We also have the Unit type, which is a singleton:

$$() :: ()$$

for example:

$$\text{unit} :: a \to (),$$

or

$$\text{one} :: () \to \text{Int}.$$

We will see later, a category theoretic generalization.

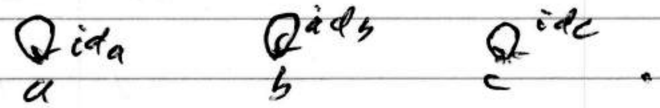
Examples

The most basic category, is the empty category, 0, with no objects.

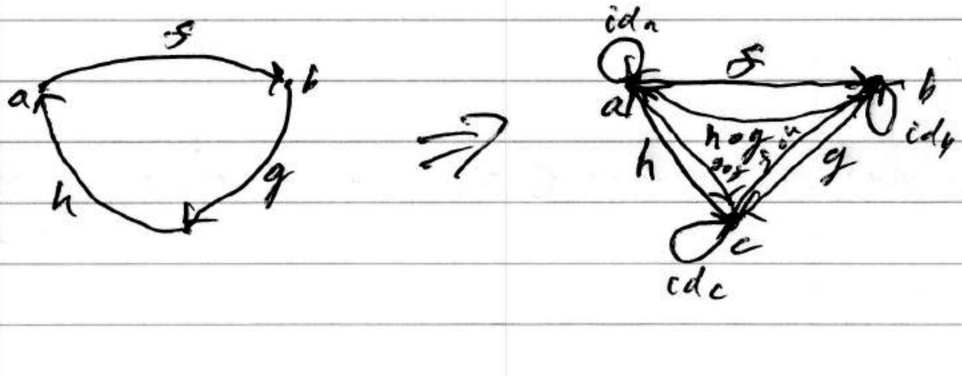
We also have a category with one object:

This category is <u>not</u> unique, unlike in group theory, where we can say <u>the</u> group of order 1.

What happens if we take 3 objects, and apply the minimum to satisfy the axioms? Well, we just set

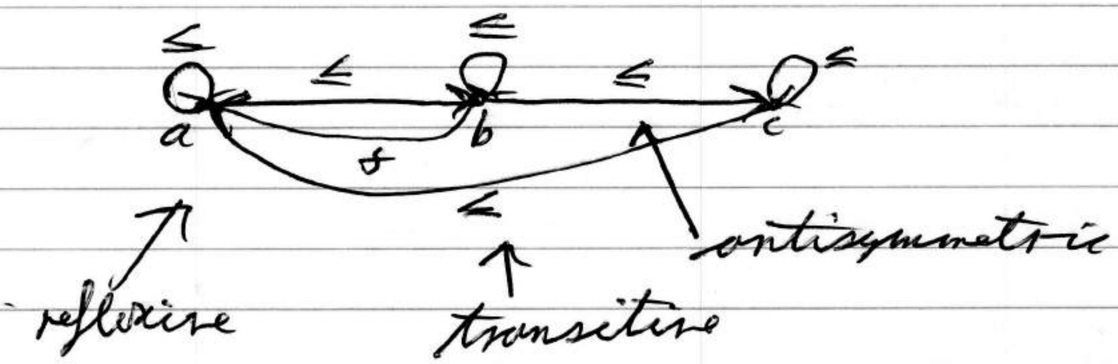


In general, we can take any digraph, and just add the minimal arrows we need. This is called a free category, and is <u>free</u> in the same way the free monoid was!



Orders

Consider the relation $\leq$, where we have an arrow when $a \leq b$.



The preorder conditions imply the arrows of a category, we call this a "thin category".

In Category Theory, a hom-set:

$$C(a, b)$$

is the set of arrows between objects a and b.

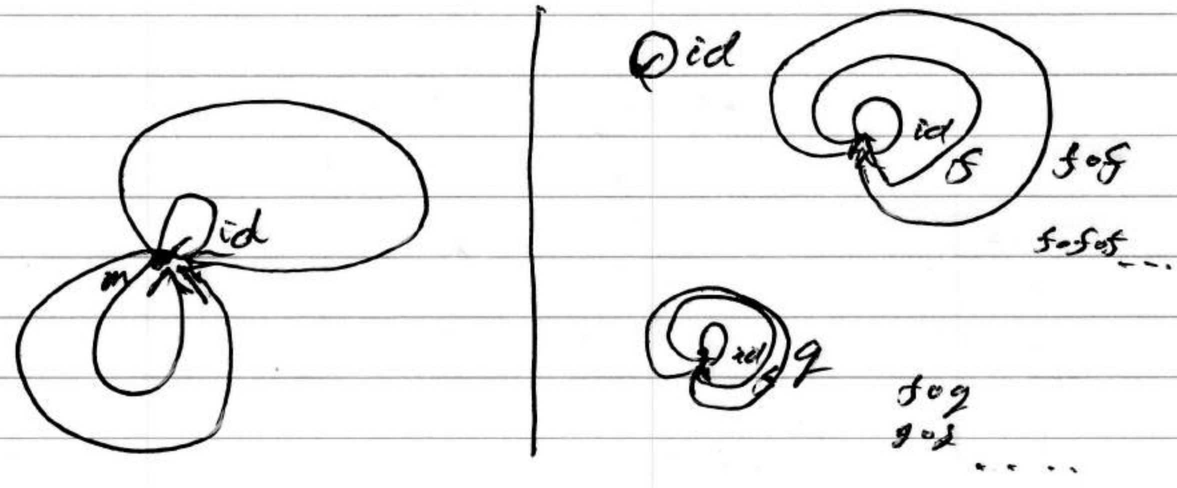
In a thin category, the hom-set must either be empty or singleton.

In a partial order, arrows like $\rightleftarrows$ are not allowed, and in a total order, then there are arrows between every object – this does not imply non-empty hom-set tho.

In a thin category, bimorphism $\not\Rightarrow$ bijection.

Monoids

We return to our one-element category, such as:



This is a <u>monoid</u> !

The elements of our monoid are the arrows, and multiplication is composition of our arrows, and unit element id.

If we are adding arrows without constraint, well then, we have the <u>free</u> monoid.

We have the hom-set:

$$M(m, m).$$

This is the definition of monoid M.

N.B. Every functional is composable. We have weak typing !