Algebra, categories and more

Aranya Kumar Bal, JRF-CS

Indian Statistical Institute, Kolkata

April 21, 2024

Outline

Some remarkable coincidences... or is it?

Enter categories

Some applications of category theory

Table of Contents

Some remarkable coincidences... or is it?

Throughout the history of mathematics, people have defined structures on sets in various different ways. In recent times it was realised that there's something very fundamentally similar to almost all constructions. Let's look at some of them.

Sets

Consider **Set** to be the collection of all sets (whether this is a set itself is irrelevant here) and functions between them. Then the following are true about them.

- Composition of functions is again a function.

Sets

Consider **Set** to be the collection of all sets (whether this is a set itself is irrelevant here) and functions between them. Then the following are true about them.

- Composition of functions is again a function.
- Composition of functions is associative.

Sets

Consider **Set** to be the collection of all sets (whether this is a set itself is irrelevant here) and functions between them. Then the following are true about them.

- Composition of functions is again a function.
- Composition of functions is associative.
- For any functions $f_1 : A \to B$ and $f_2 : B \to C$, there is a function $g : A \to C$, namely $g = f_2 \circ f_1$.

Sets

Consider **Set** to be the collection of all sets (whether this is a set itself is irrelevant here) and functions between them. Then the following are true about them.

- Composition of functions is again a function.
- Composition of functions is associative.
- For any functions $f_1 : A \to B$ and $f_2 : B \to C$, there is a function $g : A \to C$, namely $g = f_2 \circ f_1$.
- For any set S, there is an identity function $\mathrm{id}_S : S \to S$, that is, $\mathrm{id}_B \circ f = f = f \circ \mathrm{id}_A$ for any function $f : A \to B$.

Groups

Consider **Grp** to be the collection of all groups and group homomorphisms between them. Then the following are true about them.

- Composition of group homomorphisms is again a group homomorphism.

Groups

Consider **Grp** to be the collection of all groups and group homomorphisms between them. Then the following are true about them.

- Composition of group homomorphisms is again a group homomorphism.
- Composition of group homomorphisms is associative.

Groups

Consider **Grp** to be the collection of all groups and group homomorphisms between them. Then the following are true about them.

- Composition of group homomorphisms is again a group homomorphism.
- Composition of group homomorphisms is associative.
- For any group homomoprphisms $f_1 : G_1 \to G_2$ and $f_2 : G_2 \to G_3$, there is a group homomorphism $g : G_1 \to G_3$, namely $g = f_2 \circ f_1$.

Groups

Consider **Grp** to be the collection of all groups and group homomorphisms between them. Then the following are true about them.

- Composition of group homomorphisms is again a group homomorphism.
- Composition of group homomorphisms is associative.
- For any group homomoprphisms $f_1 : G_1 \to G_2$ and $f_2 : G_2 \to G_3$, there is a group homomorphism $g : G_1 \to G_3$, namely $g = f_2 \circ f_1$.
- For any group G, there is an identity group homomorphism $\mathrm{id}_G : G \to G$, that is, $\mathrm{id}_{G_2} \circ f = f = f \circ \mathrm{id}_{G_1}$ for any group homomorphism $f : G_1 \to G_2$.

Rings

Consider **Ring** to be the collection of all rings and ring homomorphisms between them. Then the following are true about them.

- Composition of ring homomorphisms is again a ring homomorphism.

Rings

Consider **Ring** to be the collection of all rings and ring homomorphisms between them. Then the following are true about them.

- Composition of ring homomorphisms is again a ring homomorphism.
- Composition of ring homomorphisms is associative.

Rings

Consider **Ring** to be the collection of all rings and ring homomorphisms between them. Then the following are true about them.

- Composition of ring homomorphisms is again a ring homomorphism.
- Composition of ring homomorphisms is associative.
- For any ring homomoprphisms $f_1 : R_1 \to R_2$ and $f_2 : R_2 \to R_3$, there is a ring homomorphism $g : R_1 \to R_3$, namely $g = f_2 \circ f_1$.

Rings

Consider **Ring** to be the collection of all rings and ring homomorphisms between them. Then the following are true about them.

- Composition of ring homomorphisms is again a ring homomorphism.
- Composition of ring homomorphisms is associative.
- For any ring homomoprphisms $f_1 : R_1 \to R_2$ and $f_2 : R_2 \to R_3$, there is a ring homomorphism $g : R_1 \to R_3$, namely $g = f_2 \circ f_1$.
- For any ring R, there is an identity ring homomorphism $\mathrm{id}_R : R \to R$, that is, $\mathrm{id}_{R_2} \circ f = f = f \circ \mathrm{id}_{R_1}$ for any ring homomorphism $f : R_1 \to R_2$.

Vector spaces over a field k

Consider $\mathbf{Vect}_k$ to be the collection of all vector spaces over the field k and linear maps between them. Then the following are true about them.

- Composition of linear maps is again a linear map.

Vector spaces over a field k

Consider $\mathbf{Vect}_k$ to be the collection of all vector spaces over the field k and linear maps between them. Then the following are true about them.

- Composition of linear maps is again a linear map.
- Composition of linear maps is associative.

Vector spaces over a field k

Consider $\mathbf{Vect}_k$ to be the collection of all vector spaces over the field k and linear maps between them. Then the following are true about them.

- Composition of linear maps is again a linear map.
- Composition of linear maps is associative.
- For any linear maps $f_1 : V_1 \to V_2$ and $f_2 : V_2 \to V_3$, there is a linear map $g : V_1 \to V_3$, namely $g = f_2 \circ f_1$.

Vector spaces over a field k

Consider $\mathbf{Vect}_k$ to be the collection of all vector spaces over the field k and linear maps between them. Then the following are true about them.

- Composition of linear maps is again a linear map.
- Composition of linear maps is associative.
- For any linear maps $f_1 : V_1 \to V_2$ and $f_2 : V_2 \to V_3$, there is a linear map $g : V_1 \to V_3$, namely $g = f_2 \circ f_1$.
- For any vector space V, there is an identity linear map $\mathrm{id}_R : R \to R$, that is, $\mathrm{id}_{R_2} \circ f = f = f \circ \mathrm{id}_{R_1}$ for any linear map $f : R_1 \to R_2$.

Well... seems like algebra has a very common structure throughout. What about things outside algebra?

Topological spaces

Definition

A topological space is a tuple (X, τ) where X is a set and τ is a collection of subsets of X such that the following holds:

1. $\varnothing$ and X are in τ.
2. If $A, B \in \tau$, then $A \cap B \in \tau$.
3. For any indexing set $\mathcal{I}$, if $A_i \in \tau$ for all $i \in \mathcal{I}$, then $\cup_{i \in \mathcal{I}} A_i \in \tau$

Topological spaces

Definition

A topological space is a tuple (X, τ) where X is a set and τ is a collection of subsets of X such that the following holds:

1. $\varnothing$ and X are in τ.
2. If $A, B \in \tau$, then $A \cap B \in \tau$.
3. For any indexing set $\mathcal{I}$, if $A_i \in \tau$ for all $i \in \mathcal{I}$, then $\cup_{i \in \mathcal{I}} A_i \in \tau$

Example

(X, τ) with $X = \{0, 1\}$ and $\tau = \{\varphi, \{0\}, \{0, 1\}\}$
$(\mathbb{R}, \tau)$ where τ contains the union of open intervals in $\mathbb{R}$.

Topological spaces

Definition

A topological space is a tuple (X, τ) where X is a set and τ is a collection of subsets of X such that the following holds:

1. $\varnothing$ and X are in τ.
2. If $A, B \in \tau$, then $A \cap B \in \tau$.
3. For any indexing set $\mathcal{I}$, if $A_i \in \tau$ for all $i \in \mathcal{I}$, then $\cup_{i \in \mathcal{I}} A_i \in \tau$

Example

(X, τ) with $X = \{0, 1\}$ and $\tau = \{\varphi, \{0\}, \{0, 1\}\}$
$(\mathbb{R}, \tau)$ where τ contains the union of open intervals in $\mathbb{R}$.

τ is called a topology on X, the sets in τ are called the open sets of X, and we just write X if the topology understood.

Definition

A function $f : (X, \tau_1) \to (Y, \tau_2)$ is said to be *continuous* if $U \in \tau_2 \Rightarrow f^{-1}(U) \in \tau_1$ where f^{-1} means the preimage.

In other words, a function is continuous if preimage of open sets is open.

Definition

A function $f : (X, \tau_1) \to (Y, \tau_2)$ is said to be *continuous* if $U \in \tau_2 \Rightarrow f^{-1}(U) \in \tau_1$ where f^{-1} means the preimage.
In other words, a function is continuous if preimage of open sets is open.

Example

Consider the topological space $(X, \mathcal{P}(X))$ and consider any other topological space (Y, τ). Then any function $f : (X, \mathcal{P}(X)) \to (Y, \tau)$ is continuous.

The collection **Top** of topological space and continuous functions satisfies the following properties (as you might have guessed by now):

- Composition of continuous maps is again a continuous map.

The collection **Top** of topological space and continuous functions satisfies the following properties (as you might have guessed by now):

- Composition of continuous maps is again a continuous map.
- Composition of continuous maps is associative.

The collection **Top** of topological space and continuous functions satisfies the following properties (as you might have guessed by now):

- Composition of continuous maps is again a continuous map.
- Composition of continuous maps is associative.
- For any continuous maps $f_1 : X_1 \to X_2$ and $f_2 : X_2 \to X_3$, there is a linear map $g : X_1 \to X_3$, namely $g = f_2 \circ f_1$.

The collection **Top** of topological space and continuous functions satisfies the following properties (as you might have guessed by now):

- Composition of continuous maps is again a continuous map.
- Composition of continuous maps is associative.
- For any continuous maps $f_1 : X_1 \to X_2$ and $f_2 : X_2 \to X_3$, there is a linear map $g : X_1 \to X_3$, namely $g = f_2 \circ f_1$.
- For any topological space X, there is an identity continuous map $\mathrm{id}_X : X \to X$, that is, $\mathrm{id}_{X_2} \circ f = f = f \circ \mathrm{id}_{X_1}$ for any continuous map $f : X_1 \to X_2$.

One more example before we close this section.

Consider **Prop** to be the collection of propositions (in some system) and (equivalence classes of) proofs between them. Then the following are true about them.

- Composition of proofs is again a proof.

One more example before we close this section.

Consider **Prop** to be the collection of propositions (in some system) and (equivalence classes of) proofs between them. Then the following are true about them.

- Composition of proofs is again a proof.
- Composition of proofs is associative.

One more example before we close this section.

Consider **Prop** to be the collection of propositions (in some system) and (equivalence classes of) proofs between them. Then the following are true about them.

- Composition of proofs is again a proof.
- Composition of proofs is associative.
- For any proofs $X_1 \vdash X_2$ and $X_2 \vdash X_3$, there is a proof $X_1 \vdash X_3$, simply by composing the two proofs (almost all logic system have this inference rule). Logicians write this as

$$\frac{X \vdash Y \quad Y \vdash Z}{X \vdash Z}$$

One more example before we close this section.

Consider **Prop** to be the collection of propositions (in some system) and (equivalence classes of) proofs between them. Then the following are true about them.

- Composition of proofs is again a proof.
- Composition of proofs is associative.
- For any proofs $X_1 \vdash X_2$ and $X_2 \vdash X_3$, there is a proof $X_1 \vdash X_3$, simply by composing the two proofs (almost all logic system have this inference rule). Logicians write this as

$$\frac{X \vdash Y \quad Y \vdash Z}{X \vdash Z}$$

- For any proposition X, there is a proof that proves X from X itself, namely $X \vdash X$

So in fact topological spaces and logic (to some extent) also follow a similar structure! This coincidences are too good to be true... So why not take the properties we have been looking at as axioms for some *generalized structure*?

Table of Contents

Definition

A category $\mathcal{C}$ consists of the following data.

- A collection of objects denoted by $\mathrm{Ob}(\mathcal{C})$

Definition

A category $\mathcal{C}$ consists of the following data.

- A collection of objects denoted by $\mathrm{Ob}(\mathcal{C})$
- A collection of *arrows*, called morphisms between pairs of objects; $x \xrightarrow{f} y$ means f is a morphism from x to y. The collection of all morphisms from x to y is called the homset of x to y, denoted as $\hom_{\mathcal{C}}(x, y)$

Definition

A category $\mathcal{C}$ consists of the following data.

- A collection of objects denoted by $\mathrm{Ob}(\mathcal{C})$
- A collection of *arrows*, called morphisms between pairs of objects; $x \xrightarrow{f} y$ means f is a morphism from x to y. The collection of all morphisms from x to y is called the homset of x to y, denoted as $\hom_{\mathcal{C}}(x, y)$
- A composition rule : given $x \xrightarrow{f} y$ and $y \xrightarrow{g} z$, there is a morphism $x \xrightarrow{g \circ f} z$.

And we want these to satisfy some properties.

- Each object x has an identity morphism $x \xrightarrow{\mathrm{id}_x} x$ which satisfies $\mathrm{id}_y \circ f = f = f \circ \mathrm{id}_x$ for any $x \xrightarrow{f} y$.

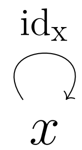
And we want these to satisfy some properties.

- Each object x has an identity morphism $x \xrightarrow{\mathrm{id}_x} x$ which satisfies $\mathrm{id}_y \circ f = f = f \circ \mathrm{id}_x$ for any $x \xrightarrow{f} y$.

- The composition is associative, that is, $f \circ (g \circ h) = (f \circ g) \circ h$ whenever $w \xrightarrow{h} x \xrightarrow{g} y \xrightarrow{f} z$.

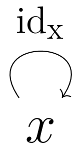
Example

As an example of a category, consider the category with just one object x and only one morphism, which has to be $x \xrightarrow{\mathrm{id}_x} x$

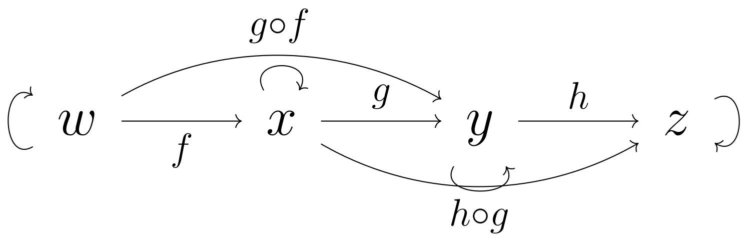


Example

As an example of a category, consider the category with just one object x and only one morphism, which has to be $x \xrightarrow{\mathrm{id}_x} x$



Another example is the following slightly non trivial one (the unmarked self loops are the identity morphisms)



From the discussion in the first section, **Set**, **Grp**, **Ring**, **Vect**, **Top**, **Prop** are all categories.

From the discussion in the first section, **Set**, **Grp**, **Ring**, **Vect**, **Top**, **Prop** are all categories.

Some more categories include that of abelian groups and group homomorphisms between them (called **AbGrp**), partially ordered sets and monotone functions (called **Pos**) etc.

From the discussion in the first section, **Set**, **Grp**, **Ring**, **Vect**, **Top**, **Prop** are all categories.

Some more categories include that of abelian groups and group homomorphisms between them (called **AbGrp**), partially ordered sets and monotone functions (called **Pos**) etc.

Almost any structure you see in math, along with their respective structure-preserving maps *probably* forms a category.

From the discussion in the first section, **Set**, **Grp**, **Ring**, **Vect**, **Top**, **Prop** are all categories.

Some more categories include that of abelian groups and group homomorphisms between them (called **AbGrp**), partially ordered sets and monotone functions (called **Pos**) etc.

Almost any structure you see in math, along with their respective structure-preserving maps *probably* forms a category.

But the actual strength of category theory comes from the following.

Definition

Let $\mathcal{C}$ and $\mathcal{D}$ be two categories. Then F is called a *covariant functor* between these two categories if it does the following:

- For any object A in $\mathrm{Ob}(\mathcal{C})$, it gives an object $F(A)$ in $\mathrm{Ob}(\mathcal{D})$.

Definition

Let $\mathcal{C}$ and $\mathcal{D}$ be two categories. Then F is called a *covariant functor* between these two categories if it does the following:

- For any object A in $\mathrm{Ob}(\mathcal{C})$, it gives an object $F(A)$ in $\mathrm{Ob}(\mathcal{D})$.

- For any morphism $A \xrightarrow{f} B$ in $\mathcal{C}$, it gives a morphism $F(A) \xrightarrow{F(f)} F(B)$ in $\mathcal{D}$ such that following diagram commutes

$$\begin{array}{ccc} A & \xrightarrow{\quad f \quad} & B \\ {\scriptstyle F}\big\downarrow & & \big\downarrow{\scriptstyle F} \\ F(A) & \xrightarrow[\quad F(f) \quad]{} & F(B) \end{array}$$

This allows us to relate different theories of math together. What can be done in one theory can be borrowed or looked at differently in other theories.

Example

Consider the functor F that maps **Grp** to **Set**. It takes in a group G and gives $F(G)$, the underlying set of the group. As for morphisms, F takes a group homomorphism and gives the underlying function (that is, this functor essentially *forgets* the group structure.) One can verify that this is indeed a functor.

This allows us to relate different theories of math together. What can be done in one theory can be borrowed or looked at differently in other theories.

Example

Consider the functor F that maps **Grp** to **Set**. It takes in a group G and gives $F(G)$, the underlying set of the group. As for morphisms, F takes a group homomorphism and gives the underlying function (that is, this functor essentially *forgets* the group structure.) One can verify that this is indeed a functor.

A more interesting example is the homotopy and homology functors. They allow us to study topological spaces by actually looking at groups.

Definition

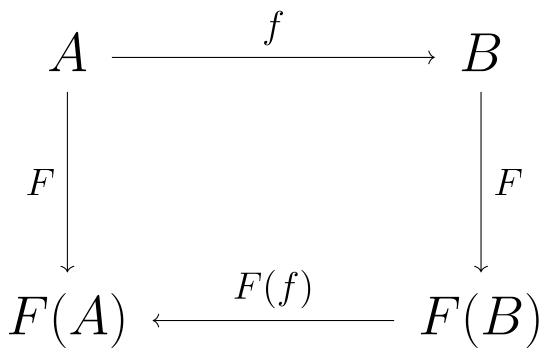
Let $\mathcal{C}$ and $\mathcal{D}$ be two categories. Then F is called a *contravariant functor* between these two categories if it does the following:

- For any object A in $\mathrm{Ob}(\mathcal{C})$, it gives an object $F(A)$ in $\mathrm{Ob}(\mathcal{D})$.

Definition

Let $\mathcal{C}$ and $\mathcal{D}$ be two categories. Then F is called a *contravariant functor* between these two categories if it does the following:

- For any object A in $\mathrm{Ob}(\mathcal{C})$, it gives an object $F(A)$ in $\mathrm{Ob}(\mathcal{D})$.
- For any morphism $A \xrightarrow{f} B$ in $\mathcal{C}$, it gives a morphism $F(B) \xrightarrow{F(f)} F(A)$ in $\mathcal{D}$ such that following diagram commutes



Note that the bottom arrow has now flipped. That's the main difference between covariant and contravariant functors.

Example

Consider the functor F that maps $\mathbf{Vect}_k$ to $\mathbf{Vect}_k$. It takes in a vector space V and outputs the dual vector space $V^* = F(V)$. As for morphisms, it takes in a linear map $f : V \to W$ and outputs the dual mapo $F(f) := f^* : W^* \to V^*$ which acts as $f^*(\varphi) = \varphi \circ f$.

One quickly verifies that this is a contravariant functor.

Note that the bottom arrow has now flipped. That's the main difference between covariant and contravariant functors.

Example

Consider the functor F that maps $\mathbf{Vect}_k$ to $\mathbf{Vect}_k$. It takes in a vector space V and outputs the dual vector space $V^* = F(V)$. As for morphisms, it takes in a linear map $f : V \to W$ and outputs the dual mapo $F(f) := f^* : W^* \to V^*$ which acts as $f^*(\varphi) = \varphi \circ f$.

One quickly verifies that this is a contravariant functor.

A more interesting example is that of cohomology functors. They again allow us to study topological spaces using rings this time.

Table of Contents

In this section, I will tell some stories without being extremely pedantic. I will not define everything but treat things as fairy tales.

Let's start by looking at a very known kind of proof : diagonalization. We know that this proof idea is used to show that there is no surjection from $\mathbb{N}$ to $\mathcal{P}(\mathbb{N})$, that there are more real numbers than natural numbers, that the halting problem is undecidable and many more. Lawvere (1969) [1] generalised this massively to just one theorem.

Theorem

In any cartesian closed category, if there exists an object A and a weakly point-surjective morphism

$$A \xrightarrow{g} Y^A$$

then Y has the fixed point property.

Theorem

In any cartesian closed category, if there exists an object A and a weakly point-surjective morphism

$$A \xrightarrow{g} Y^A$$

then Y has the fixed point property.

Lawvere in fact generalised this even further, which is much easier to understand.

Theorem

In any cartesian closed category, if there exists an object A and a weakly point-surjective morphism

$$A \xrightarrow{g} Y^A$$

then Y has the fixed point property.

Lawvere in fact generalised this even further, which is much easier to understand.

Theorem

Let A, Y be any objects in any category with finite products (including the empty product 1). Then the following two statements cannot both be true:

1. *there exists $f : A \times A \to Y$ such that for all $g : A \to Y$ there exists $x : 1 \to A$ such that for all $a : 1 \to A$*

$$\langle a, x \rangle f = a.g$$

2. *there exists $\alpha : Y \to Y$ such that for all $y : 1 \to Y$ such that $y.\alpha \neq y$*

The proof is very small, but there's many things we need to understand first so I will skip that. I will only show that there is just one main diagram that is used in the proof (the construction of g)

The proof is very small, but there's many things we need to understand first so I will skip that. I will only show that there is just one main diagram that is used in the proof (the construction of g)

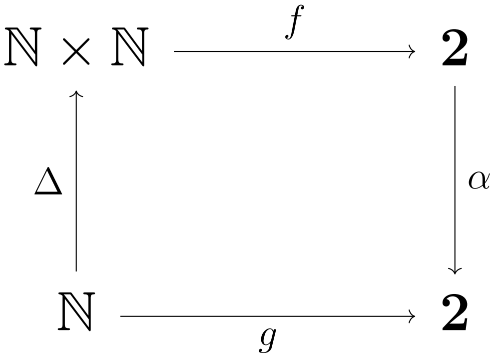
$$\begin{array}{ccc} A \times A & \xrightarrow{\quad f \quad} & Y \\ {\scriptstyle \Delta}\big\uparrow & & \big\downarrow{\scriptstyle \alpha} \\ A & \xrightarrow[\quad g \quad]{} & Y \end{array}$$

The proof is very small, but there's many things we need to understand first so I will skip that. I will only show that there is just one main diagram that is used in the proof (the construction of g)

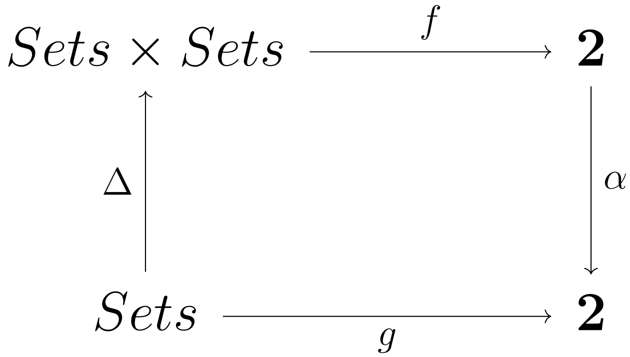
$$
\begin{array}{ccc}
A \times A & \xrightarrow{\quad f \quad} & Y \\
\Delta \Big\uparrow & & \Big\downarrow \alpha \\
A & \xrightarrow[\quad g \quad]{} & Y
\end{array}
$$

I will now show how different problems are just different instances of this same diagram; it might look like I am just writing the same thing, but that's the fun of category theory! (This is meant to invoke curiosity, the beautiful paper by Yanofsky [2] from where I read all this will be in the bibliography)

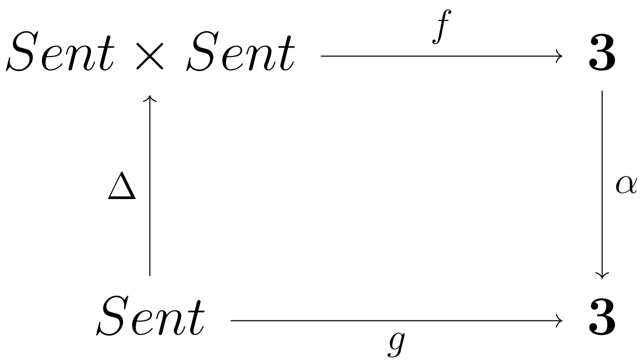
This is one of the more beautiful results out of elementary category theory that I have seen!



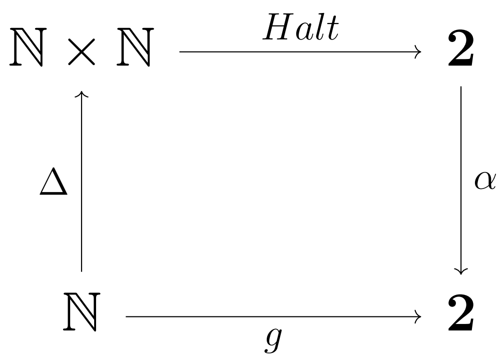
(a) Cantor's theorem



(b) Russel's paradox



(a) Strong liar paradox



(b) Halting problem

(If I ever reach here in my slides while presenting, high five to myself)

(If I ever reach here in my slides while presenting, high five to myself)

Let's talk about concrete computer science stuff now. We know how data types are the basics of any (typed) programming languages (it's fun to think of what an untyped language might actually look like : For more info, look up the **Bash** language).

(If I ever reach here in my slides while presenting, high five to myself)

Let's talk about concrete computer science stuff now. We know how data types are the basics of any (typed) programming languages (it's fun to think of what an untyped language might actually look like : For more info, look up the **Bash** language).

Category theory gives us a nifty little way to generalise types, and actually make more sense out of type theory. I will give a very brief look into that world now

(By the way, this is not some ideal curiosity, the programming language of Haskell is essentially built upon a categorical framework).

For brevity, just note that there is a way to talk of elements of objects (even if they are not sets or anything) in category theory, details will be in the report. We move forward.

For brevity, just note that there is a way to talk of elements of objects (even if they are not sets or anything) in category theory, details will be in the report. We move forward.

Suppose $A \xrightarrow{f} B$ is a morphism in some category $\mathcal{C}$. We can then write this as $x : A \vdash f(x) : B$, that is, the element (free variable) x is of type A, and f sends it to an element $f(x)$ of type B.

For brevity, just note that there is a way to talk of elements of objects (even if they are not sets or anything) in category theory, details will be in the report. We move forward.

Suppose $A \xrightarrow{f} B$ is a morphism in some category $\mathcal{C}$. We can then write this as $x : A \vdash f(x) : B$, that is, the element (free variable) x is of type A, and f sends it to an element $f(x)$ of type B.

Composition of morphisms is what is known in type theory as substitution. So if we have $A \xrightarrow{f}\xrightarrow{g} C = A \xrightarrow{g \circ f} C$, this corresponds to

$$\frac{x : A \vdash f(x) : B \quad y : B \vdash g(y) : C}{x : A \vdash g(f(x)) : C}$$

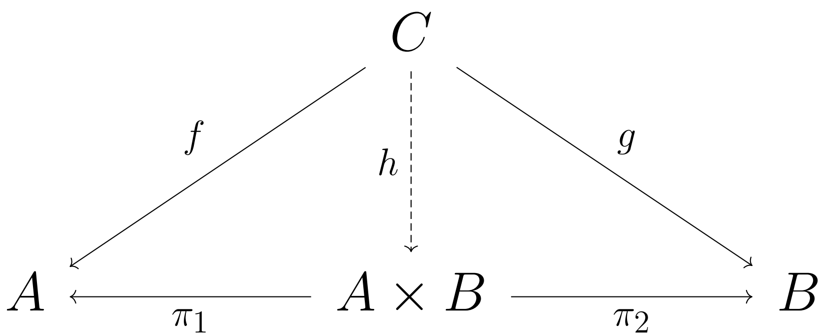
This way of categorical thinking actually allows us to make different data types out of categories. In fact, the power of this approach allows us to define data structures as well.

This way of categorical thinking actually allows us to make different data types out of categories. In fact, the power of this approach allows us to define data structures as well.

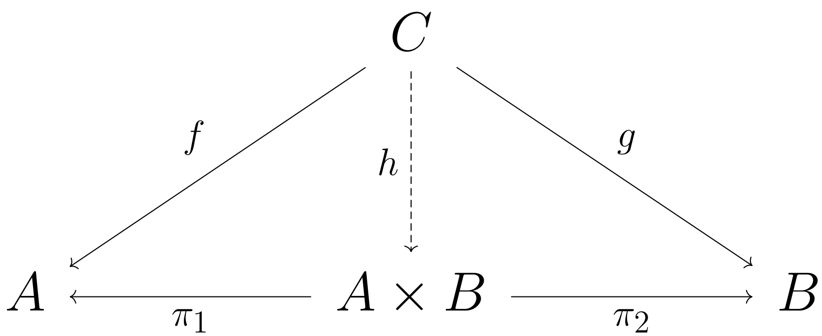
All the examples next are taken from Tatsuya Hagino's beautiful 2020 thesis [3] (the paper will be in the bibliography).

To get some context, let's look at **Set** again. The cartesian product satisfies this commutative diagram (where π_1 and π_2 are projections on the first and second coordinates respectively)

To get some context, let's look at **Set** again. The cartesian product satisfies this commutative diagram (where π_1 and π_2 are projections on the first and second coordinates respectively)

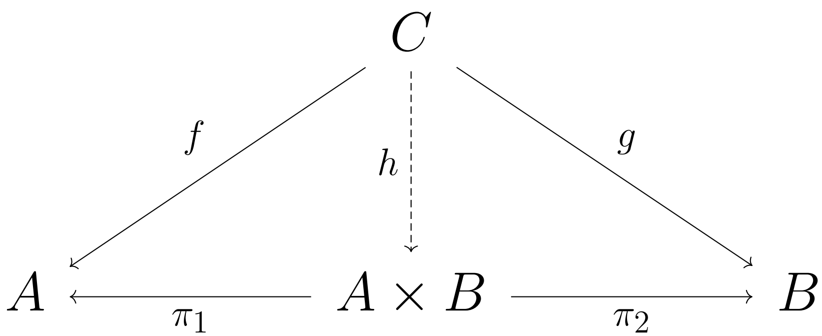


To get some context, let's look at **Set** again. The cartesian product satisfies this commutative diagram (where π_1 and π_2 are projections on the first and second coordinates respectively)



This just says that $A \times B$ is the unique (upto isomorphism, here bijection) such object that looks like a product. We generalise this to any product by just saying this diagram defines what a product of two objects is.

To get some context, let's look at **Set** again. The cartesian product satisfies this commutative diagram (where π_1 and π_2 are projections on the first and second coordinates respectively)



This just says that $A \times B$ is the unique (upto isomorphism, here bijection) such object that looks like a product. We generalise this to any product by just saying this diagram defines what a product of two objects is.

Writing h as $\langle f, g \rangle$, we can create a *constructor* for a product data type:

right object $A \times B$ with $\langle\ ,\ \rangle$ is

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

$$\pi_2 : A \times B \to B$$

end object

The word *right* is a technical categorical detail (called an adjunction) that allows us to hide a lot of details like the diagram etc. And the $\langle\ ,\ \rangle$ is another technical detail required for the construction of every data type.

$$
\begin{aligned}
&\text{right object } A \times B \text{ with } \langle\ ,\ \rangle \text{ is} \\
&\qquad \pi_1 : A \times B \to A \\
&\qquad \pi_2 : A \times B \to B \\
&\text{end object}
\end{aligned}
$$

The word *right* is a technical categorical detail (called an adjunction) that allows us to hide a lot of details like the diagram etc. And the $\langle\ ,\ \rangle$ is another technical detail required for the construction of every data type.

Here's another data type, commonly known as the natural numbers:

$$
\begin{aligned}
&\text{left object nat with pr } (\ ,\ ) \text{ is} \\
&\qquad \text{zero} : 1 \to \text{ nat} \\
&\qquad \text{succ} : \text{ nat} \to \text{ nat} \\
&\text{end object}
\end{aligned}
$$

One can clearly see what this means. The natural numbers are initially defined by 0, and then you can get any natural number by always continuing to add 1.

I will present some more data structures now (which can also be treated as data types in programming); the left is a list, the right is a binary tree, and the last one is an infinite list.

I will present some more data structures now (which can also be treated as data types in programming); the left is a list, the right is a binary tree, and the last one is an infinite list.

left object list(A) with prl (,) is

  nil : $1 \to$ list(A)

  cons : $A \times$ list(A) $\to$ list(A)

end object

left object BinTree(T) with prlTree (,) is

  tip : $A \to$ BinTree(A)

  join : BinTree(A) $\times$ BinTree(A) $\to$ BinTree(A)

end object

right object inflist(A) with fold (,) is

  hd : inflist(A) $\to A$

  cons : inflist(A) $\to$ inflist(A)

end object

I would like to end this with one view that stuck with me from a long time ago; how category theory is really a bird's eye view of all of mathematics, and also of processes in general. I will let John Baez [4] show the correspondences.

I would like to end this with one view that stuck with me from a long time ago; how category theory is really a bird's eye view of all of mathematics, and also of processes in general. I will let John Baez [4] show the correspondences.

Category Theory	Physics	Topology	Logic	Computation
object X	Hilbert space X	manifold X	proposition X	data type X
morphism $f\colon X \to Y$	operator $f\colon X \to Y$	cobordism $f\colon X \to Y$	proof $f\colon X \to Y$	program $f\colon X \to Y$
tensor product of objects: $X \otimes Y$	Hilbert space of joint system: $X \otimes Y$	disjoint union of manifolds: $X \otimes Y$	conjunction of propositions: $X \otimes Y$	product of data types: $X \otimes Y$
tensor product of morphisms: $f \otimes g$	parallel processes: $f \otimes g$	disjoint union of cobordisms: $f \otimes g$	proofs carried out in parallel: $f \otimes g$	programs executing in parallel: $f \otimes g$
internal hom: $X \multimap Y$	Hilbert space of 'anti-X and Y': $X^* \otimes Y$	disjoint union of orientation-reversed X and Y: $X^* \otimes Y$	conditional proposition: $X \multimap Y$	function type: $X \multimap Y$

Table 4: The Rosetta Stone (larger version)

Figure 3: The correspondences

And a quote from Tom Leinster

Category theory takes a bird's eye view of mathematics. From high in the sky, details become invisible, but we can spot patterns that were impossible to detect from ground level. How is the lowest common multiple of two numbers like the direct sum of two vector spaces? What do discrete topological spaces, free groups, and fields of fractions have in common?

And that's it! Details for everything will be in the report, and if you have any questions you can find me on Discord at *nekomatism* or *Nekoma#2259*. Obviously mail is always open at hayatea90@gmail.com

Bibliography

Barry Mitchell, Jan-Erik Roos, Friedrich Ulmer, Hans-Berndt Brinkmann, Stephen U Chase, Paul Dedecker, RR Douglas, PJ Hilton, F Sigrist, Charles Ehresmann, et al.

Diagonal arguments and cartesian closed categories.

In *Category Theory, Homology Theory and their Applications II: Proceedings of the Conference held at the Seattle Research Center of the Battelle Memorial Institute, June 24–July 19, 1968 Volume Two*, pages 134–145. Springer, 1969.

Noson S. Yanofsky.

A Universal Approach to Self-Referential Paradoxes, Incompleteness and Fixed Points.

Bulletin of Symbolic Logic, 9(3):362–386, September 2003.

Tatsuya Hagino.

A Categorical Programming Language, 2020.

J. Baez and M. Stay.

Physics, Topology, Logic and Computation: A Rosetta Stone, page 95–172.

Springer Berlin Heidelberg, 2010.