Algebra, categories and more

Aranya Kumar Bal

Abstract

In this report, I shall exposit upon how some coincidences pave the way for a natural definition of a category and related notions. Then I will show some applications of category theory to pure math, computer science and maybe some more areas.

1 Introduction

The birth of category theory is in modern algebraic topology, and the theory was fully extended for the needs of algebraic geometry. To even appreciate why the theory needed development takes a lot of time, so in this report, I will try to give a flavour of category theory from a different perspective : *generalization*. At one point, I will remark how this generalization actually captures almost all of mathematics in a sense.

The question of *why category theory* is still prevalent because it's so general that not everyone can see the appeal of it. But hopefully, I will be able to instill some amount of curiosity in you to get you interested in the subject.

2 Some remarkable coincidences... or is it?

In this section, I shall provide clues as to what the general theory might look like; the clues come from what we already know.

2.1 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 $f : A \to B$.

2.2 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 $f : G_1 \to G_2$.

2.3 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 $f : R_1 \to R_2$.

2.4 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 $f : R_1 \to R_2$.

The above examples clearly shows that there is some common structure among algebraic structures. How about other structures that are not algebraic?

2.5 Topological spaces

Definition 2.1 (Topological space)**.** 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 $\bigcup_{i \in \mathcal{I}} A_i \in \tau$

Example 2.1. (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 2.2 (Continuous function)**.** 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 2.2. 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.
- 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 to show another such structure. This time, it's in logic.

2.6 Propositions

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

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

- 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! These coincidences are too good to be true... So why not take the properties we have been looking at as axioms for some *generalized structure*?

3 Categories

Definition 3.1 (Category)**.** A category $\mathcal{C}$ consists of the following data

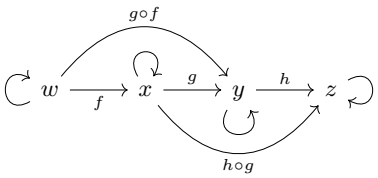
- A collection of objects denoted by $\mathrm{Ob}(\mathcal{C})$
- A collection of *arrows*, called morphisms between pairs of objects; $x \xrightarrow{f} y$ means f is a morphism from x to y. The collection of all morphisms from x to y is called the homset of x to y, denoted as $\hom_{\mathcal{C}}(x, y)$, and the collection of all the morhisms is denoted $\mathrm{Hom}_{\mathcal{C}}$.
- 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 $x \xrightarrow{f} y$.
- 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 3.1. We will look at several examples. First a very simple one. Consider a category with only one object and only one morphism. Obviously the morphism has to go from the objct to itself, and the morphism can only be the identity morphism (by definition). So we can represent it as

A slightly more non-trivial category is



In fact, the above examples generalise a little with an easy proof.

Theorem 3.1. Given a directed graph G, the reflexive and transitive closure of G is a category with the vertices as objects and edges as morphisms.

The proof is just a routine check of the conditions for being a category.

Also, from the discussion in the previous section, we have that $\mathbf{Set}, \mathbf{Grp}, \mathbf{Ring}, \mathbf{Vect}_k, \mathbf{Top}, \mathbf{Prop}$ 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. So one should always be on the lookout!

3.1 Functors

The actual strength of category theory comes from the following:

Definition 3.2 (Covariant functor)**.** 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

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

Example 3.2. 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 3.3 (Contravariant functor)**.** 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

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

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

Example 3.3. 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.

The above example of dual vector space is actually a special case of a more general functor. Given a (small) category $\mathcal{C}$, every object defines a function $\hom_{\mathcal{C}}(-, X) : \mathcal{C} \to \mathbf{Set}$ that takes an object Y and returns the set $\hom(Y, X)$, and takes a morphism $f : Y \to Z$ and returns the function

$$f^* : \hom(Z, X) \to \hom(Y, X)$$
$$g \mapsto g \circ f$$

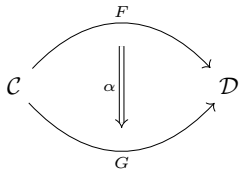
This is called the Hom-functor and it is a contravariant functor.

Functors allow 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 by means of moving through functors. In fact, the observant reader would have noticed that functors also look like morphisms of some sort, which prompts the following definition.

Definition 3.4 (Category of categories)**.** Consider the following structure where objects are small categories[1] and morphisms are functors between them. This forms a category called **Cat**.

There's another fun category, but before that we need a definition.

Definition 3.5 (Natural transformation)**.** Given two categories $\mathcal{C}$ and $\mathcal{D}$ and functors F, G from $\mathcal{C}$ to D, a natural transformation $\alpha : F \to G$ (denoted by the following diagram sometimes)



is an assignment to every object X of $\mathcal{C}$ of a morphism $\alpha_X : F(X) \to G(X)$ in $\mathcal{D}$ such that for any morphism $f : X \to Y$ in $\mathcal{C}$, the following diagram commutes in $\mathcal{D}$

$$\begin{array}{ccc} F(X) & \xrightarrow{F(f)} & F(Y) \\ \downarrow{\scriptstyle \alpha_X} & & \downarrow{\scriptstyle \alpha_Y} \\ G(X) & \xrightarrow[G(f)]{} & G(Y) \end{array}$$

Here's the category now.

Definition 3.6 (Functor category)**.** Given categories $\mathcal{C}$ and $\mathcal{D}$, the functor category is the category whose collection of objects is the collection of all functors $F : \mathcal{C} \to \mathcal{D}$ and the morphisms are natural transofrmations between these functors. This category is often written as $\mathcal{D}^{\mathcal{C}}$.

Let me end this section with an open problem with extremely huge ramifications (pun intended) in number theory, algebraic geometry and several other fields (including physics)

Conjecture 1 (categorical unramified geometric Langlands conjecture)*.* For G a reductive group and for Σ an algebraic curve, there are functors (two funtors in opposite directions, such that they are *inverses* in a sense) between stable $(1, \infty)$-categories of, on the one hand, D-modules on the moduli stack of G-principal bundles on Σ, and, on the other hand, nilpotent ind-objects of quasi-coherent sheaves on the ${}^L G$-moduli stack of local systems on Σ

$$(\mathrm{Ind}(\mathcal{O}\,\mathrm{Mod}(\mathrm{Loc}_{L_G}(\Sigma))))_{\mathrm{Nilp}_{\mathrm{Glob}}} \to \mathcal{D}\,\mathrm{Mod}(\mathrm{Bun}_G(\Sigma))$$

for ${}^L G$ the Langlands dual group.

In 2024, the group of D. Arinkin, D. Beraldo, L. Chen, D. Gaitsgory, J. Faergeman, K. Lin, S. Raskin and N. Rozenblyum has claimed a proof of the conjecture.

[1] Okay, so the word small category has come up twice now, so it deserves an explanation. A category is called small if it's homset is a set. Plain and simple.

4 Some applications of categories

In this section, we wills see how category theory allows us to prove things generally, construct objects very naturally etc.

4.1 Diagonalization proofs

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

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

then Y has the fixed point property.

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

Theorem 4.2. Let T, 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 $f : T \times T \to Y$ such that for all $g : T \to Y$ there exists $x : 1 \to T$ such that for all $a : 1 \to T$
$$\langle t, x\rangle f = t.g$$
2. there exists $\alpha : Y \to Y$ such that for all $y : 1 \to Y$ such that $y.\alpha \neq y$.

We are technically not equipped to solve this since this does rquire some category theoretical stuff that we haven't encountered. But, there's a more down-to-earth version of this that is easier to understand and would also serve our purpose. The theorem is due to Yanofsky [2].

Theorem 4.3. If Y is a set and there exists a function $\alpha : Y \to Y$ without a fixed point (for all $y \in Y, \alpha(y) \neq y$) then for all sets T and for all functions $f : T \times T \to Y$, there exists a function $g : T \to Y$ such that for all $t \in T$, $g(-) \neq f(-, t)$.

Proof. Suppose the conditions of the statement hold. Now, there is a function $\Delta : T \to T \times T$ sending t to (t, t). Then let $g : T \to Y$ be defined as follows

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

In words, $g(t) = \alpha(f(t, t))$. If $g(-) = f(-, t)$ for some $t = t_0$, then at t_0 we have

$$f(t_0, t_0) = g(t_0) = \alpha(f(t_0, t_0))$$

contradicting that α does not have a fixed point. □

How does one even use this?

4.1.1 Cantor's theorem

As a start, let's show that there is no bijection from $\mathbb{N}$ to $\mathcal{P}(\mathbb{N})$ (This is cantor's theorem.) If there was, then there would an enumeration of the elements of $\mathcal{P}(\mathbb{N})$, say $S_1, S_2, \ldots$. Let $\mathbf{2}$ be the set $\{0, 1\}$ and consider $\alpha : \mathbf{2} \to \mathbf{2}$ such that $\alpha(0) = 1$ and $\alpha(1) = 0$. Let $f : \mathbb{N} \times \mathbb{N} \to \mathbf{2}$ be such that

$$f(n, m) = \begin{cases} 1 \text{ if } n \in S_m \\ 0 \text{ if } n \notin S_m \end{cases}$$

that is, $f = \chi_{S_m}$ Let g be defined as

$$\begin{array}{ccc} \mathbb{N} \times \mathbb{N} & \xrightarrow{f} & \mathbf{2} \\ \Delta \uparrow & & \downarrow \alpha \\ \mathbb{N} & \xrightarrow[g]{} & \mathbf{2} \end{array}$$

If we let $G = \{n \in \mathbb{N} \mid n \notin S_n\} \subseteq \mathbb{N}$, then $g = \chi_G$.

The setup is ready. Now note that if for some $m_0 \in \mathbb{N}$ we have $g(-) = f(-, m_0)$, then $f(m_0, m_0) = g(m_0) = \alpha(f(m_0, m_0))$ implying α has a fixed point, which we clearly see is false. Thus for all m, $g(-) = f(-, m_0)$ and thus $\chi_G \neq \chi_{S_m}$ for all m and hence G is not in the enumeration, contradicting that $\mathcal{P}(\mathbb{N})$ is enumerable. This finishes the proof.

4.1.2 Russel's paradox

Let's look at Russel's paradox, which states that the collection of all sets that are not members of themselves is both a member of itself and not a member of itself.

Let $Sets$ be some "universe" of sets. Again let α be as defined above. Let $f : Sets \times Sets \to \mathbf{2}$ be defined as

$$f(s,t) = \begin{cases} 1 \text{ if } s \in t \\ 0 \text{ if } s \notin t \end{cases}$$

We construct g as

$$\begin{array}{ccc} Sets \times Sets & \xrightarrow{f} & \mathbf{2} \\ \Delta \uparrow & & \downarrow \alpha \\ Sets & \xrightarrow[g]{} & \mathbf{2} \end{array}$$

Again if for some t_0, $g(-) = f(-, t_0)$, then

$$f(t_0, t_0) = g(t_0) = \alpha(f(t_0, t_0))$$

contradicting that α has no fixed point. This for all sets t, $g(-) \neq f(-, t)$. Since g is basically the characteristic function of those sets who do not contain themselves, the only way to avoid the paradox is to declare that this collection in $Sets$ does not form a set itself.

4.1.3 The strong liar paradox

It's a fun exercise to solve this extension of the liar paradox. Here we will be dealing with a ternary logic system, consisting or true, false and meaningless. Let the set of these three truth values be $\mathbf{3} = \{T, F, M\}$. Consider the sentence

'yields falsehood or meaninglessness
when appended to its own quotation'
yields falsehood or meaninglessness
when appended to its own quotation

This is an issue because if this is true, then it is false or meaningless. If it is false, then it is true and hence not meaningless. If it is meaningless, then it is true and not meaningless.

This can be encoded in terms of the theorem we proved,

Consider the set of english sentences $Sent$. Let $f : Sent \times Sent \to \mathbf{3}$ such that

$$f(s_1, s_2) = \begin{cases} T \text{ if } a_2 \text{ describes } a_1 \\ M \text{ if it is meaningless for } a_2 \text{ to describe } a_1 \\ F \text{ if } a_2 \text{ does not describe } a_1 \end{cases}$$

Now let $\alpha : \mathbf{3} \to \mathbf{3}$ such that $\alpha(T) = F$ and $\alpha(M) = T = \alpha(F)$. Let g be defined as

$$\begin{array}{ccc} Sent \times Sent & \xrightarrow{f} & \mathbf{3} \\ \Delta \uparrow & & \downarrow \alpha \\ Sent & \xrightarrow[g]{} & \mathbf{3} \end{array}$$

Then g is the characteristic function of sentences (that is, maps to T) that are neither false nor meaningless when describing themselves. This is again similar to Russel's paradox.

4.1.4 Halting problem

This case is interesting because just dealing with sets will not be enough. We will add more structure to the sets by attaching a notion of computable structures and recursively enumerable sets (this technically uses the nmore general version that Lawvere proved, but the essence of the easier theorem we proved is still there).

Let us define what we mean by a computable universe $\mathbf{U}$. It is a category with the following properties

1. $\mathbb{N}$ and $\mathbf{2}$ are objects in $\mathbf{U}$.
2. For every object C in $\mathrm{Ob}(U)$, there is some enumeration of the elements of C. An enumeration is a total isomorphism $e_C : \mathbb{N} \to C$ (these objects are essentially the computable structures like trees, graphs, strings etc).
3. For every partial function $f : C \to C'$ (these are the morphisms) there is a corresponding number $\langle f \rangle \in \mathbb{N}$. This can be thought of as the Gödel numbering of the program that computes f (or encoding in terms of turing machines if one likes).
4. For every partial function $f : C \to C'$, there is a corresponding recursively enumerable set $W_f \subseteq \mathbb{N}$. For every $c \in C$ (elements can be talked of in an arbitrary category, look at the section on types. But here, all objects are sets so this makes sense naturally), f has a value at c if and only if $e_C^{-1}(c) \in W_f$.

In such a computable universe, $Halt$ is a total function $Halt : \mathbb{N} \times \mathbb{N} \to \mathbf{2}$ in $\mathbf{U}$ such that for all $f : C \to C'$ we have

$$Halt(n, m) = \begin{cases} 1 \text{ if } n \in W_m \\ 0 \text{ if } n \notin W_m \end{cases}$$

that is

$$Halt(-, \langle f \rangle) = \chi_{W_f}$$

Consider $\alpha : \mathbf{2} \to \mathbf{2}$ to be defined as $\alpha(0) = 1$ and $\alpha(1) = \uparrow$ (α is a partial function). We construct g as

$$\begin{array}{ccc} \mathbb{N} \times \mathbb{N} & \xrightarrow{Halt} & \mathbf{2} \\ \Delta \uparrow & & \downarrow \alpha \\ \mathbb{N} & \xrightarrow[g]{} & \mathbf{2} \end{array}$$

We can now show that $Halt$ is not total. Suppose $Halt$ is defined at $\langle g \rangle$. Then

$$Halt(\langle g \rangle, \langle g \rangle) = 1 \iff \langle g \rangle \in W_g \iff g(\langle g \rangle) = 1 \iff \alpha(Halt(\langle g \rangle, \langle g \rangle)) = 1 = Halt(\langle g \rangle, \langle g \rangle)$$

which is not true since α has no fixed point. Thus we cannot tell if all computable functions halt or not.

4.2 Types, data types and structures

In this section, we shall see some concrete applications of category theory to computer science stuff, more specifically, data types and structures.

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.[2]

4.2.1 Types

We must first talk about elements of a object in any category. We know that objects might not have any *elements* inside them, because, well objects can be almost anything and may not have to even look like sets. But in sets, we can talk about elements of a set A by looking at all possible functions from a set with exactly one element to A, $\{\cdot\} \to A$.

Note that $\{\cdot\}$ is special. Every set has exactly one function that maps to this set (namely, the function that sends everything in that set to the point), so this is called a *terminal* object. Borrowing this setup in any arbitrary category with a terminal object T, we define the morphism $T \xrightarrow{f} X$ is a (generalised) element of X.

[2]By the way, this is not some ideal curiosity, the programming language of Haskell is essentially built upon a categorical framework. In fact, Haskell is almost a category itself; there's a small technical issue which can be mended in the language itself.

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.

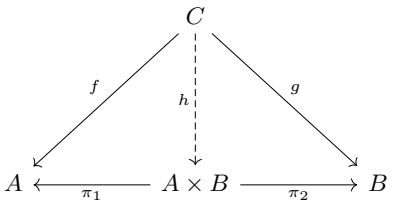
Composition of morphisms is what is known in type theory as substitution. So if we have $A \xrightarrow{f} \xrightarrow{g} C = A \xrightarrow{g \circ f} C$, this corresponds to

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

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. The examples here are mostly from Tatsuya hagino's beautiful thesis [3].

4.2.2 Data types and Data structures

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 of objects in any category 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:

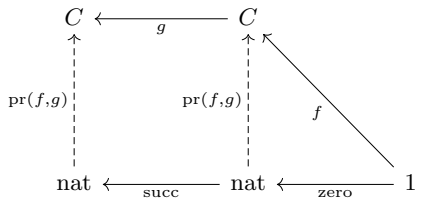
right object $A \times B$ with $\langle\ ,\ \rangle$ is
$\pi_1 : A \times B \to A$
$\pi_2 : A \times B \to B$
end object

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.

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

left object nat with pr (,) is
zero : $1 \to$ nat
succ : nat $\to$ nat
end object

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. The commutative diagram for this structure is



The type of pr(,) is

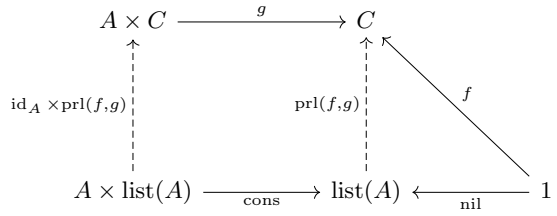
$$\frac{f : 1 \to C \qquad g : C \to C}{\mathrm{pr}(f, g) : \mathrm{nat} \to C}$$

Here are several data structures defined similarly.

- **Lists**. The constructor is

$$\begin{aligned}&\text{left object list}(A)\text{ with prl (,) is}\\&\quad\text{nil }:1\to\text{ list}(A)\\&\quad\text{cons }:A\times\text{list}(A)\ \to\ \text{list}(A)\\&\text{end object}\end{aligned}$$

The nil operation defines an empty list, whereas the cons operation appends an element to the list (lists are recursively constructed as tuples; in fact prl stands for primitive recursion on lists). The commutative diagram for this is



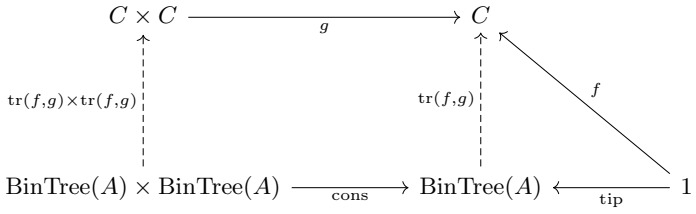
The type of prl(,) is

$$\frac{f:1\to C\qquad g:A\times C\to C}{\mathrm{prl}(f,g):\mathrm{list}(A)\to C}$$

- **Binary trees** : The constructor is

$$\begin{aligned}&\text{left object BinTree}(T)\text{ with prbt (,) is}\\&\quad\text{tip }:A\to\text{ BinTree}(A)\\&\quad\text{join }:\text{BinTree}(A)\ \times\text{BinTree}(A)\ \to\ \text{BinTree}(A)\\&\text{end object}\end{aligned}$$

The tip operation defines a root, and the join operators attaches two trees to a root as left and right subtrees (and prbt stands for primitive recursion on binary trees). The commutative diagram is



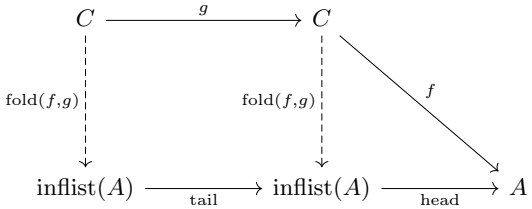
The type of prbt(,) is

$$\frac{f:1\to C\qquad g:C\times C\to C}{\mathrm{prbt}(f,g):\mathrm{BinTree}(A)\to C}$$

- **Infinite Lists**. This is a fun data structure. Normally programming languages doesn't allow infinite lists; but categorically it is possible to define them (and in fact there are use cases of this, and Haskell does use them). The constructor is

$$\begin{aligned}&\text{right object inflist}(A)\text{ with fold (,) is}\\&\quad\text{head }:\text{ inflist}(A)\to A\\&\quad\text{tail }:\text{ inflist}(A)\to\text{ inflist}(A)\\&\text{end object}\end{aligned}$$

The commutative diagram is as follows



And the type of fold(,) is

$$\frac{f : C \to A \qquad g : C \times C}{\text{fold}(f, g) : C \to \text{inflist}(A)}$$

In fact, with a little more work one can define a constructor for Moore automa, for mealy automata etc. We leave them for now since they require slightly more categorical nuance to define.

5 Conclusion

How deep does the water run? How far can we stretch the abstractness of category theory to get better understanding of different fields? Turns out that the answer is *very deep*. John Baez [4] provided a table that shows how category theory essentially says that extremely different looking things in completely different fields are actually the manifestation of the same thing; he called it the Rosetta Stone (for categories). The table in all its glory is as follows:

Category Theory	*Physics*	*Topology*	*Logic*	*Computation*
Object X	Hilbert Space X	Manifold X	Proposition X	Data type X
Morphism $f : X \to Y$	Operator $f : X \to Y$	Cobordism $f : X \to Y$	Proof $f : X \to Y$	Program $f : X \to Y$
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 1: The Rosetta Stone

Maybe I will exposit on this table in a separate report someday. Someday... For now, I end with 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?

6 Acknowledgement

This report shall never have been possible without my algebra class asking for a presentation, and for that I am indebted to professor Sujata Ghosh. The report was also vastly imporoved due to a help of a couple of friends who suggested modifications and proof-read the material.

7 Bibliography

References

[1] 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.

[2] Noson S. Yanofsky. A Universal Approach to Self-Referential Paradoxes, Incompleteness and Fixed Points. *Bulletin of Symbolic Logic*, 9(3):362–386, September 2003.

[3] Tatsuya Hagino. A Categorical Programming Language, 2020.

[4] J. Baez and M. Stay. *Physics, Topology, Logic and Computation: A Rosetta Stone*, page 95–172. Springer Berlin Heidelberg, 2010.