Category Theory Notes

Vatsal Limabchia

September 2019

Contents

Categories

1 What is category theory

"As a first approximation category theory is the abstract algebra of functions."
— Steve Awodey

1.1 Definition of a category

A category **C** consists of

- Objects. $A, B, C, \ldots$
- arrows $f, g, h, \ldots$ each function have a domain $dom(f)$ and codomain $cod(f)$

A category must satisfy the following laws. For each composable pair of arrows $f : A \to B$ and $g : B \to C$.

- Unit
- Associativity. The following diagram commutes.

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[rr, bend right, "f \circ g"'] & B \arrow[r, "g"] \arrow[rr, bend left, "g \circ h"] & C \arrow[r, "h"] & D
\end{tikzcd}$$

In general there can be many, even infinitely many, arrows between two objects. There can also be none at all.

1.2 Examples of categories

- **Sets**. The objects are sets and the arrows are functions.
- **1**. The category with one object and one arrow. It looks like this.

$$\begin{tikzcd}
\bullet \arrow[loop right]
\end{tikzcd}$$

- A poset $(P, \leq)$ (partially ordered set) is a set P with a binary relation $\leq$ that satisfies reflexivity, antisymmetry, and transivity. Any poset forms a category where the elements of the category are the objects of P and where for two objects a and b there is an arrow from a to b iff. $a \leq b$:

$$p \to q \iff p \leq q$$

A category based on a poset is called a *poset category.*

Note that **Poset** is a category with "few" arrows. Between two objects p and q there is at most one arrow.

- **Poset** (or **Pos**) The category where the objects are all posets and the arrows are monotone functions.
- A monoid is a triple $(S, \oplus, e)$. Any monoid forms a category. The category contains a single object. Each element in the monoid is an arrow in the category and composition of arrows is the composition from the monoid. The identity arrow is the identity element in the monoid. This construction satisfies the law of a category directly from the laws of a monoid.

  Note that a category based on a monoid is a category with "few" objects. There is only a single objects. Furthermore a category with one object is always a monoid. Thus, *a monoid is just a category with one object.*
- **Monoid** (or **Mon**) The category where the objects are all monoid and the arrows are monoid homomorphisms.
- **Rel**. The objects are sets and the arrows are relations.

2 Commutative diagrams

In category theory equations can be expressed using commutative diagrams. When we draw a diagram like the following.

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "h"'] & B \arrow[d, "g"] \\
C \arrow[r, "p"] & D
\end{tikzcd}$$

It is supposed to communicate that $f \circ g = h \circ p$. In such cases we say the "the diagram commutes" to denote the properties that the diagram indicates that the arrows have.

As another example, consider this diagram:

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[dr, "h"] & B \arrow[d, "g"] \\
 & C
\end{tikzcd}$$

We say that the triangle *commutes* if $f \circ g = h$.

In general it is the case that for a commutative diagram all paths with the same origin and end are equal.

3 Constructing categories

3.1 Opposite category

For a category C, its opposite category, denoted C^{op}, is the category obtained by formally reversing the direction of all its arrows. Each object A in $\mathbf{C}$ is an object in $\mathbf{C}^{op}$ and earch arrow $f : A \to B$ in $\mathbf{C}$ is an arrow $f : B \to A$ in $\mathbf{C}^{op}$. It is important to understand that this is a formal operation that can be applied to any category C.

One way that one should *note* think about $C^{opposite}$ is as something that reverses each arrow.

3.2 Slice category

A slice category $\mathbf{C}/X$ has as objects all arrows in $\mathbf{C}$ with X as their codomain.

Arrows in $\mathbf{C}/X$.

$$\begin{tikzcd}
A \arrow[rr, "f"] \arrow[dr, "g"] & & B \arrow[dl, "h"] \\
& X &
\end{tikzcd}$$

4 Universal Mapping Properties

A universal mapping property is used to define things in a categorical way.

Universal mapping properties determine things uniquely up to isomorphism.

Saying that there exists a *unique* something is the same as saying that there exists a *single* something which is the same as saying that there is exactly one something.

5 Functor

A functor is a mapping that is used to transfer commutative diagrams between categories.

Definition 5.1 (Functor)**.** A functor $F \to \mathbf{C} \to \mathbf{D}$ is a map between categories. It consists of two parts:

- A mapping between objects: $F_{obj} : obj(\mathbf{C}) \to (obj\mathbf{D})$.
- A mapping between arrows: $F_{arr} : arr(\mathbf{C}) \to (arr\mathbf{D})$.

However, we usually just write F and treat it as an overloaded function that can be applied to both arrows and objects. A functor must satisfy three laws:

- Preserves domain and codomain. $F(f : A \to B) = F(f) : F(A) \to F(B)$

- Preserve identities. $F(id_A) = id_{F(A)}$
- Preserve composition: $F(g \circ f) = F(g) \circ F(f)$

6 Isomorphisms

Definition 6.1 (Isomorphism)**.** An isomorphism is an arrow with an inverse.

Examples.

- In a monoid category the isomorphisms are the elements that have an inverse.
- in a group every element has an inverse. Therefore, in a group every arrow is and isomorphism.

Proposition 1. If $f : f \to B$ and $g : B \to C$ are isomorphisms then their composition $f \circ g$ is also an isomorphism and the inverse is

$$(f \circ g)^{-1} = g^{-1} \circ f^{-1}$$

This means that isomorphisms compose into isomorphisms.

Proof. First we show that $g^{-1} \circ f^{-1}$ is a right inverse.

$$(f \circ g) \circ (g^{-1} \circ f^{-1}) = f \circ (g \circ g^{-1}) \circ f^{-1} = f \circ f^{-1} = id_A$$

Then we show that $g^{-1} \circ f^{-1}$ is a left inverse.

$$(g^{-1} \circ f^{-1}) \circ (f \circ g) = g^{-1} \circ (f^{-1} \circ f) \circ g = g^{-1} \circ g = id_B$$

□

7 Monomorphism and epimorphism

7.1 Monomorphism

Definition 7.1. Monomorphism In a category **C**, an arrow $f : A \to B$ is a *monomorphism* if for any $g, h : C \to A$ it is the case that $fg = fh \implies g = h$.

$$\begin{tikzcd} C \arrow[r, shift left, "g"] \arrow[r, shift right, "h"'] & A \arrow[r, "f"] & B \end{tikzcd}$$

A monomorphism is also called a *mono* and it is said to be *monic.*

Mnemonic: A monomorphism is the categorical equivalent to an *injective* function, that is a one-to-one function, and *mono* means *one.*

Proposition 2. A function $f : A \to B$ between sets is monic in **Sets** if and only if it is injective.

Proof. ($\Rightarrow$) Assume f is a monomorphism in **Sets**. We must show that f is an injective function. Let $a, a' \in A$ where $a \neq a'$. We are done if we can show that $f(a) \neq f(a')$.

($\Leftarrow$) Assume f is an injective function. We must show that f is a monomorphism. □

Definition 7.2. Epimorphism In a category **C**, an arrow $f : A \to B$ is an *epimorphism* if for any $i, j : B \to D$ it is the case that $if = jf \implies i = j$.

$$\begin{tikzcd} A \arrow[r, "f"] & B \arrow[r, shift left, "i"] \arrow[r, shift right, "j"'] & D \end{tikzcd}$$

An epimorphism is said to be *epic*.

Why does a surjective funtion satisfy $if = jf \implies i = j$. Intuitively since f reaches its entire domain then any differences in i and j whould be exposed in the composition. If there was some elements in B that where never reached by f then the functions i and j could differ for those elements such that $if = jf$ without $i = j$ being true.

Proposition 3. A function $f : A \to B$ between sets is an epimorphism in **Sets** if and only if it is surjective.

Proof. ($\Rightarrow$) Assume f is an epimorphism. We must show that f is a surjective function. Let b be an element in B. Let $d, d' \in D$ where $d \neq d'$ and define $i : B \to D$ as the function that maps every element in B to d except for b which it maps to d'.

$$i(x) = \begin{cases} d' & x = b \\ d & \text{otherwise} \end{cases}$$

Let $j : B \to D$ be a constant function that sends every element to d.

$$j(x) = d$$

Since $i \neq j$ we have $if \neq jf$. Thus there exists some $a \in A$ where $i(f(a)) \neq j(f(a))$. But i and j are equal for all arguments except b. Hence $f(a) = b$. Since b was arbitrary this shows that f is surjective.

($\Leftarrow$) Assume that f is a surjective function. We must show that f is an epimorphism. Let $i, j : B \to D$ be arbitrary functions where $i \neq j$. Then there exists some $b \in B$ where $i(b) \neq j(b)$. Since f is surjective there must exist an $a \in A$ where $b = f(a)$. Putting this together we have:

$$\begin{aligned} i(b) &\neq j(b) && \text{since } i \neq j \\ i(f(a)) &\neq j(f(a)) && \text{from how we defined } b \end{aligned}$$

Therefore $if \neq jf$. □

Proposition 4. In a poset category all arrows are both monic and epic.

Proof. Let $f : A \to B$ be an arrow in a poset category. Per the definition of a poset category there is always only a single arrow between two objects. Therefore if we have $g, h : C \to A$ such that $fg = fh$ we can conclude that $g = h$ since they are both arrows between C and A. Therefore f is monic. Similarly, if $i, j : B \to D$ such that $if = jf$ then $i = j$ and f is a epic. □

Proposition 5. In a poset category all arrows are both monic and epic.

Proof. Let $f : A \to B$ be an arrow in a poset category. Per the definition of a poset category there is always only a single arrow between two objects. Therefore if we have $g, h : C \to A$ such that $fg = fh$ we can conclude that $g = h$ since they are both arrows between C and A. Therefore f is monic. Similarly, if $i, j : B \to D$ such that $if = jf$ then $i = j$ and f is a epic. □

Proposition 6. Every isomorphism is also a monomorphism and epimorphism.

Proof. (Monomorphism) Let m be an isomorphism with inverse m^{-1}, then $mx = my$ implies that $x = m^{-1}mx = m^{-1}my = y$ giving us that m is a monomorphism.

(Epimorphism) Let m be an isomorphism with inverse m^{-1}, then $xm = ym$ implies that $x = xm^{-1}m = ym^{-1}m = y$ giving us that m is an epimorphism. □

Consider a situration where we have arrows $s : A \to B$ and $r : B \to A$ such that the following diagram commutes.

$$\begin{tikzcd}
A \arrow[r, "s"] \arrow[dr, "id_A"'] & B \arrow[d, "r"] \\
 & A
\end{tikzcd}$$

We have $rs = id_A$ (but not necessarily $sr = id_B$). We call r both a *left-inverse* for s and a *retraction of* s. A morphism that is a retraction of some morphism is called a *retraction.* Symmetrically s is called both a *right-inverse* of r and a *section* of r. A morphism that is a section of some morphism is called a *section.*

Proposition 7. Every *retraction* is epic and every *section* is monic.

Proof. Assume r is a retraction. Then r has a left inverse s. Then

$$fr = gr \implies frs = grs \implies f = g$$

Thus r is epic.

The argument for sections is similar. □

Definition 7.3 (Split mono and split epic)**.** An arrow $f : A \to B$ is a *split mono* if it has a *left* inverse. That is, an arrow $g : C \to A$ such that

$$gf = id_A$$

An arrow $f : A \to B$ is a *split epi* if it has a *right* inverse. That is, an arrow $g : B \to C$ such that

$$fg = id_B$$

When a mono is a split mono we say that the *mono splits*.

7.2 A summary of monos, epis, and split monos and epis

- A mono cancels on the *left*: $\forall g, h : fg = fh \implies g = h$.
- A split mono is an arrow with a *left* inverse: $\exists g : gf = id_A$.
- An epi cancels on the *right* $\forall i, j : if = jf \implies i = j$.
- A split epi is an arrow with a *right* inverse: $\exists g : fg = id_B$.

Proposition 8 (Inverses are unique)**.** If an arrow $f : A \to B$ has two inverses $g, g' : B \to A$ then $g = g'$.

Proof. Consider the commutative diagram below.

$$\begin{tikzcd}
B \arrow[r, "id_B"] \arrow[d, "g"'] & B \arrow[d, "g'"] \\
A \arrow[r, "id_A"'] \arrow[ur, "f"] & A
\end{tikzcd}$$

From the diagram we can derive the following equality.

$$g' = g' id_B = g'(fg) = (g'f)g = id_A g = g$$

□

8 Initial and terminal objects

Definition 8.1 (Initial)**.** An object 0 is *initial* if for any object C there is a unique morphism $F : 0 \to C$.

$$\begin{tikzcd}
0 \arrow[r, dashed, "f"] & C
\end{tikzcd}$$

In other words an initial object is an object that has a single arrow to all other objects. Visually it looks like this.

$$\begin{tikzcd}
 & 0 \arrow[dl] \arrow[d] \arrow[dr] & \\
A & B & C
\end{tikzcd}$$

Note that this is a universal mapping property. $\forall C \exists! f$.

Definition 8.2 (Terminal)**.** An object 1 is *terminal* if for any object C there is a unique morphism $F : C \to 1$.

$$\begin{tikzcd} C \arrow[r, dashed, "f"] & 1 \end{tikzcd}$$

In other words a terminal object is an object that every other object has a single arrow towards. Visually it looks like this.

$$\begin{tikzcd} A \arrow[dr] & B \arrow[d] & C \arrow[dl] \\ & 1 & \end{tikzcd}$$

Note that this is an universal mapping property. $\forall C \exists! f$.

Terminal objects are also sometimes called *final*.

8.1 Examples of initial and terminal object

In *Sets* is the empty set inital and the all singleton sets are terminal. Thus in **Sets** there are only one initial but infinitely many terminal objects.

8.2 Poset

In a poset an initial object is one that is smaller than or equal to any other object. A terminal object is one that is greater than or equal to any other object.

In a bounded lattice the bottom element $\perp$ is the only initial object and the top element $\perp$ is the only terminal object. As such we may write

$$0 = \perp, \quad 1 = \top$$

This example provides a rationale and some intuition for the names 0 and 1.

Proposition 9. Initial/terminal objects are unique up to a unique isomorphism. That is,

- If 0 and $0'$ are both initial objects in a category $\mathbf{C}$ then $0 \cong 0'$ and the isomorphism is unique.
- If 1 and $1'$ are both terminal objects in a category $\mathbf{C}$ then $1 \cong 1'$ and the isomorphism is unique.

Proof. Let 0 and $0'$ be two initial objects. Since 0 is an initial object there exists a unique arrow $f : 0 \to 0'$. Since any object has the identity arrow, f must be the identity arrow $f = id_0$. As both 0 and $0'$ are initial objects there exist a unique arrow $f : 0 \to 0'$ and a unique arrow $g : 0' \to 0$. The composition $f \circ g$ is an arrow from $0'$ to 0. But we know that the only arrow from $0'$ to 0 is

$id_{0'}$. Similarly, $g \circ f$ is an arrow from 0 to $0'$ and it must be equal to id_0. Put together this means that g is an inverse to f. Hence f is an isomorphism.

$$\begin{tikzcd}
0 \arrow[r, "f"] \arrow[dr, "id_0"'] & 0' \arrow[d, "g"] \arrow[dr, "id_{0'}"] & \\
 & 0 \arrow[r, "f"] & 0'
\end{tikzcd}$$

$$\begin{tikzcd}
0 \arrow[loop left, "id_0"] \arrow[r, bend left, "f"] & 0' \arrow[l, bend left, "g"] \arrow[loop right, "id_{0'}"]
\end{tikzcd}$$

□

9 Products

9.1 Binary product

Since a category is a thing that can basically only do composition one might think that there is not much one can define based on this. However, we will see that it is actually possible to create many definitions based only on this. Now we will see how to define a notion of a *product* or a pair. Intuitively a pair is some data-structure that can hold two elements. But that is not very precise. It turns out that by using category theory we can make this much more precise.

Definition 9.1 (Product)**.** A product of two objects A and B consists of an object P and two projection arrows $p_1 : P \to A$ and $p_2 : P \to B$.

$$\begin{tikzcd}
A & P \arrow[l, "p_1"'] \arrow[r, "p_2"] & B
\end{tikzcd}$$

The above must satisfy the universal mapping property that $\forall x, f_1, f_2 \exists! u$ such that the following diagram commutes.

$$\begin{tikzcd}
 & X \arrow[dl, "x_1"'] \arrow[d, dashed, "u"] \arrow[dr, "x_1"] & \\
A & P \arrow[l, "p_1"] \arrow[r, "p_2"'] & B
\end{tikzcd}$$

Let us try an unpack that definition. Note that the definition doesn't say anything about what a product *is*. It only describes a property that a product should have in order for it to be a product. But, why does that definition make sense?

Consider a programming library that implements a product or pair data-structure in a programming language. It would have to expose a constructor for pairs `makePair :  A -> B -> Pair A B` and two functions for extracting values again `fst : Pair A B -> A` and `snd : Pair A B -> B`. As a user of such a library we would expect that creating a pair of two elements and extracting them again gives us the exact same elements that we put in. In other words we expect that for any a it is the case that `fst (makePair a b) == a`. This is exactly what the existence part of the definition says. If P is a product and we have an object X from which we can extract two values x_1 and x_2 (i.e. X contains x_1 and x_2) then there must be a way to construct a product based on X such that when we later extract the two values with p_1 and p_2 we get back the exact same value that we had in X. The UMP ensures that the pair data-structure doesn't "mess" with the elements we insert into it. The uniqueness part of the definition requires that there is only one possible way for P to store the two values that we have in X. This means that it must not be possible for P to support two different `makePair` functions.

In the general case it is not possible to turn two objects into a pair. That is, if A and B is two objects $A \times B$ may not exist in the category. If a category has a product for every pair of objects then the category is said to *have binary products* and is said to be *a category with binary products.*

9.1.1 Notation

If two objects A and B have a product, we write

$$A \xleftarrow{p_1} A \times B \xrightarrow{p_2} B$$

Now for any X, x_1, x_2 the notation $\langle x_1, x_2 \rangle$ denotes the unique arrow $u : X \to A \times B$ from the UMP. Beware! One might think that $\langle x, y \rangle$ would mean an actual pair of elements similarly to how (x, y) denotes a pair. This is not the case. It is an arrow as shown in the diagram below.

$$\begin{tikzcd}
& X \arrow[dl, "x_1"'] \arrow[d, dashed, "\langle x_1{,}x_2 \rangle"] \arrow[dr, "x_1"] & \\
A & A \times B \arrow[l, "p_1"'] \arrow[r, "p_2"'] & B
\end{tikzcd}$$

9.2 Non-examples of products

- Define a product of sets as follows:

  $$A \times B = \{(n, a, b) \mid n \in \{0, 1\}, a \in A, b \in B\}$$

  With the two projection functions.

  $$\pi_1(n, a, b) = \begin{cases} a & n = 0 \\ b & n = 1 \end{cases} \qquad \pi_2(n, a, b) = \begin{cases} b & n = 0 \\ a & n = 1 \end{cases}$$

  This is not a product since it does not satisfy the uniqueness constraint in the UMP that a product must satisfy.

Lemma 1. Products are unique up to isomorphism. That is, if $A \times B$ and $A' \times B'$ are products in a category **C** then.

$$A \times B \cong A' \times B'$$

Proof. TODO □

The above theorem tells us that all products are the same except for unimportant details.

Lemma 2. In a category with binary products the binary product operation $A \times B$ is associative up to isomorphism.

$$(A \times B) \times C \cong A \times (B \times C)$$

Proof. □

10 Hom-sets

Definition 10.1. Given a locally small category **C** and objects A and B the set $\mathrm{Hom}(A, B)$ is all the arrows from A to B.

$$\begin{aligned}\mathrm{Hom}_{\mathbf{C}}(A, B) &= \text{ all arrows from } A \text{ to } B \\ &= \{\, f \in \mathbf{C} \mid f : A \to B \,\}\end{aligned}$$

Such a set of arrows is called a *Hom-set*. Since $\mathrm{Hom}(A, B)$ is an element in **Sets** this function gives us a way to turn any pair of objects in any (locally small) category into an *object in* **Sets**.

Additionally, this we can define a function.

$$\begin{aligned}\mathrm{Hom}_{\mathbf{C}}(A, g) : \mathrm{Hom}(A, B) &\to \mathrm{Hom}(A, B') \\ f &\mapsto g \circ f\end{aligned}$$

Note that the notation is overloaded! $\mathrm{Hom}(A, B)$ is the set of all arrows from A to B. But $\mathrm{Hom}(A, g)$ is a function from arrows from A to B ($\mathrm{Hom}(A, B)$) to arrows from A to B' ($\mathrm{Hom}(A, B')$).

- For an *object* B $\mathrm{Hom}(A, B)$ is an *object* in **Sets**.
- For an *arrow* g $\mathrm{Hom}(A, g)$ is an *arrow* in **Sets**.

Note that the overloading above is the exact same overloading that we use with functors. In other words we have exactly the ingredients for a functor.

11 Contravariant functor

A functor F sends an arrow $f : A \to B$ to an arrow $F(f) : F(A) \to F(B)$. However, in some cases a functor like contstruction arises that is a functor except that it it “flips” arrows and composition around. It reverses the domain and codomain of arrows such that f is mapped to $F(f) : F(A) \to F(B)$.

Such a contravariant functor from C to D is just a normal functor from C^{op} to D.

12 Duality

12.1 Isomorphism

Recal that an isomorphism is an arrow $f : A \to B$ with an inverse $f^{-1} : B \to A$. Pictorially it looks like this

$$\begin{tikzcd} A \arrow[r, bend left, "f"] & B \arrow[l, bend left, "f^{-1}"] \end{tikzcd}$$

If we take the dual of the above by reversing both arrows we end up with

$$\begin{tikzcd} A \arrow[r, bend right, "f^{-1}"'] & B \arrow[l, bend right, "f"'] \end{tikzcd}$$

Which is still two inverse arrows going back and forth between A and B. This leads to the slogan: "The dual of isomorphism is isomorphism".

13 Equalizers

Definition 13.1. Given two arrows $f, g : A \to B$. For an element E an arrow $e : E \to A$ is an equalizer if

$$f \circ e = g \circ e$$

and if for any $z : Z \to A$ there exists a unique u such that $e \circ u = z$.

$$\begin{tikzcd} E \arrow[r, "e"] & A \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & B \\ Z \arrow[u, dashed, "u"] \arrow[ur, "z"] & & \end{tikzcd}$$

And equalizer e is like a proccess that we can apply to things before we put them into f and g. If we apply this process before f and g these two processes do the exact same thing. So e somehow removes all the input for which f and g produces different results. As such it makes sense that e is called an equalizer since it equalizes f and g.

The universal mapping property ensures that e does *as little as possible* to make f and g equal.

Proposition 10. Equalizers are always monomorphisms.

Proof. Let e be an equalizer. □

Remark. A category **C** *has equalizers* if for any two arrows $f, g : A \to B$ there exists an equalizer for the arrows.

14 Coequalizers

$$\begin{tikzcd}
A \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & B \arrow[r, "q"] \arrow[dr, "z"] & Q \arrow[d, dashed, "u"] \\
 & & Z
\end{tikzcd}$$

Note that the diagram above is equivalent to the dual diagram for equalizers except it has also been flipped.

15 Subobjects

Definition 15.1 (Subobject)**.** If X is an object in a category $\mathbf{C}$ then a *subobject* of X is a monomorphism $m : M \to X$.

There is not anything new in this definition. A subobject and a monomorphism is exactly the same thing. It is nothing but a different perspective, a new way of looking at, the definition of a monomorphism.

Note that, perhaps confusingly, a subobject of X is *not* an object in the same category as X.

$\mathrm{Sub}_{\mathbf{C}}(X)$ denotes the subobjects of X in $\mathbf{C}$.

16 Pullbacks

Pullbacks is a generalization of both intersection and inverse image.

A pullback $A \times_C B$ can be though of as a pair of elements (a, b) and f and g can be though of as functions where $f(a) = g(b)$.

$$\begin{tikzcd}
Z \arrow[dr, dashed, "u"] \arrow[drr, "z_2"] \arrow[ddr, "z_1"'] & & \\
 & P \arrow[r, "p_2"] \arrow[d, "p_1"] & B \arrow[d, "g"] \\
 & A \arrow[r, "f"] & C
\end{tikzcd}$$

17 Coproduct

17.1 Coproduct in Cat

If $\mathbf{C}$ and $\mathbf{D}$ are a categories then $\mathbf{C} + \mathbf{D}$ forms a category as follows:

- The object in $\mathbf{C} + \mathbf{D}$ is the disjoint union of the objects in $\mathbf{C}$ with the objects in $\mathbf{D}$. That is

$$Obj(\mathbf{C} + \mathbf{D}) = Obj(\mathbf{C}) + Obj(\mathbf{D})$$

- An arrow in $\mathbf{C} + \mathbf{D}$ is

Limits and colimits

18 Diagram

Consider a functor $D : \mathbf{J} \to \mathbf{C}$. The functor maps object and arrows in $\mathbf{D}$ to objects and arrows in $\mathbf{C}$. In other words, for every object j in $\mathbf{J}$ the functor D maps j to an object in $\mathbf{C}$. Instead of thinking of this as a map we can also think of the object in $\mathbf{J}$ as indexing the object in $\mathbf{C}$. With this point of view we may write D_j to denote $D(j)$. Instead of thinking of D as a functor we can think of it as being objects and arrows in $\mathbf{C}$ indexed by objects and arrows in $\mathbf{J}$. When we adopt this point of view we call D a *diagram* of type $\mathbf{J}$ in $\mathbf{C}$. Formally a diagram is the same thing as a functor. But the change in name reflects the change of perspective.

19 Cone

A *cone* to a diagram D is an object C and a family of arrows in $\mathbf{C}$ where the arrows c_i, c_j $\ldots$, c_k are from C to each object in the diagram D. Making a cone shape with the object C as the top where everything commutes. An example could be

$$\begin{tikzcd}
 & C \arrow[dl, "c_i"'] \arrow[d, "c_j"] \arrow[dr, "c_k"] & \\
D_i \arrow[r, "D_\alpha"] & D_j \arrow[r, "D_\beta"] & D_k
\end{tikzcd}$$

Here $\alpha : i \to j$ and $\beta : j \to k$ are arrows in $\mathbf{J}$. A cone is sometimes written as a pair (C, c_j) where C is the "top" and c_j is the family of morphisms from C to the objects in diagram D.

20 Morphism of Cones

A morphism $\vartheta : (C, c_j) \to (C', c'_j)$ between cones

$$\begin{tikzcd}
 & C \arrow[dl, "c_i"'] \arrow[d, "c_j"] \arrow[dr, "c_k"] \arrow[dd, bend right, "\vartheta"] & \\
D_i \arrow[r, "D_\alpha"] & D_j \arrow[r, "D_\beta"] & D_k \\
 & C' \arrow[ul, "c'_i"] \arrow[u, "c'_j"'] \arrow[ur, "c'_k"'] &
\end{tikzcd}$$

where $c_i = c'_i \circ \vartheta$ for all $i \in \mathbf{J}$.

Now we have the category of cones **Cones**.

20.1 Limits

Definition 20.1 (Limit)**.**

Exponentials

Recall that in set theory a *function space* is a set of function between two sets. A function space consisting of functions from the domain A to the codomain B may be denoted as B^A. In category theory *exponentials* generalize this notion of function spaces. We want to write B^A where A and B are two objects in a category and we want for the meaning to capture the properties that a function space has in set theory.

$$\tilde{f} : A \to C^B$$

The arrow $\tilde{f}$ is like a curried version of f. It is called the exponential transpose of f.

Naturality

The difference between faithfulnes and injective on arrows is that in the former case we fix two arrows A and B and only in this case consider if the functor is surjective on arrows.

20.2 Codiagonal functor

The codiagonal functor $\nabla : \mathbf{C} + \mathbf{C} \to \mathbf{C}$ is the unique functor that satisfies the diagram

$$\begin{tikzcd}
C \arrow[r, "i_1"] \arrow[dr, "id_C"'] & C + C \arrow[d, "\nabla"] & C \arrow[l, "i_2"'] \arrow[dl, "id_C"] \\
 & C &
\end{tikzcd}$$

In other words the ∇ functor simply extracts the value from the sum and throws away the tag.

The codiagonal functor is faithful but not injective on arrows.

It being faithful means that for all $A, B \in \mathbf{C} + \mathbf{C}$ the function

$$\nabla_{A,B} : \mathrm{Hom}_{\mathbf{C}+\mathbf{C}}(A, B) \to \mathrm{Hom}_{\mathbf{C}}(\nabla A, \nabla B)$$

$f \mapsto \Delta(f)$ is injective. This is the case. Given two arrows with the same domain and codomain ∇ maps them to different arrows if they are different.

It not being injective on arrows means that there exists arrows f and g where $f \neq g$ but where $\nabla(f) = \nabla(g)$. Indeed, if k is an arrow in $\mathbf{C}$ then $i_1(k)$ and $i_2(k)$ are two different arrows in $\mathbf{C} \times \mathbf{C}$ but $\nabla(i_1(k)) = \nabla(i_2(k))$. But, note that f and g have different domains a codomains, this is why this does not affect the faithfulness of the functor.

20.3 Natural transformation

Definition 20.2 (Natural transformation)**.** Given two categories $\mathbf{C}$ and $\mathbf{D}$ and two functors between them $F, G : \mathbf{C} \to \mathbf{D}$ a **natural transformation** from F to G, denoted $\vartheta : \mathbf{C} \to \mathbf{D}$, is a collection of a arrows in $\mathbf{D}$.

For each object $C \in \mathbf{C}$ there is an arrow $\vartheta_C : F(C) \to G(C)$. Each such arrow must, for any $f : C \to C'$, satisfy the following

$$\begin{tikzcd}
F(C) \arrow[r, "\vartheta_C"] \arrow[d, "F(f)"] & G(C) \arrow[d, "G(f)"] \\
F(C') \arrow[r, "\vartheta_{C'}"] & G(C')
\end{tikzcd}$$

That is $\vartheta_{C'} \circ F(f) = G(f) \circ \vartheta_C$.

In words, we have two objects in $\mathbf{C}$: C and C', and a way to go from the first to the later: $f : C \to C'$. If we apply the functor F to f, followg that arrow to $F(C')$ and then follow the arrow $\vartheta_{C'}$ from the natural transformation then that is the same as first following the arrow ϑ_C from the natural transformation and then following the arrow $G(f)$.

In other words, the two functors F and G gives us two ways to move from the category $\mathbf{C}$ to the category $\mathbf{D}$. A natural transformation is a form of relationship between these two different translations. This relationship consists of arrows between the object in $\mathbf{D}$. This makes sense because for each object C in $\mathbf{C}$ both $F(C)$ and $G(C)$ offers a translation of this object to an object in $\mathbf{D}$ offers a translation of this object to an object in $\mathbf{D}$. Relating the functors, or translations, then consists of a way of relating each "translation result" with an arrow. The diagram above is then a way of saying that these relationships should "still agree" after following an arrow in $\mathbf{C}$ from C to some other object C' in $\mathbf{C}$.

20.4 The functor category

Natural transformations gives us arrows between functors. This indicates that we can create a category of functors.

Definition 20.3 (Natural isomorphism)**.** A natural transformation $\vartheta : F \to G$ is a *natural isomorphism* if it is an isomorphism in the corresponding functor category, namely $\mathrm{Fun}(\mathbf{C}, \mathbf{D})$.

What does it take for a natural transformation $\vartheta : F \to G$ to be an isomorphism?

Categories of Diagrams

Adjoints

Hyperdoctrines

Exercises

21 1

21.1 11

22 2

22.1 2.4

Consider the commutative triangle,

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[dr, "h"] & B \arrow[d, "g"] \\
 & B
\end{tikzcd}$$

In other words the situration where $h = gf$.

- if f and g are isos then so is h. This is true because isomorphisms compose into isomorphisms.
- if f and g are monos then so is h.

$$\begin{aligned}
ha &= ha' \\
gfa &= gfa' \\
fa &= fa' && \text{because } g \text{ is a mono} \\
a &= a' && \text{because } f \text{ is a mono}
\end{aligned}$$

Therefore h is a mono.

- if f and g are epis then so is h.

$$\begin{aligned}
ah &= a'h \\
agf &= a'gf \\
ag &= a'g && \text{because } f \text{ is an epi} \\
a &= a' && \text{because } g \text{ is an epi}
\end{aligned}$$

  Therefore g is an epi. b

- if h is monic then so is f. Assume h is monic.

$$fx = fx' \implies gfx = gfx' \implies hx = hx' \implies x = x'$$

  So f is monic.

- if h is monic then so is g. Assume g is monic.

$$xg = x'g \implies xgf = x'gf \implies xh = x'h \implies x = x'$$

- h can be monic without g being monic. Consider the category **Sets**. Let $h : \{1\} \to \{3\}$. Define $f : \{1\} \to \{1, 2\}$ as the function that maps 1 to 2. Definet $g : \{1, 2\} \to \{3\}$ as the function that maps both 1 and 2 to 3. Clearly g is not surjective and thus not monic. But $h = g \circ f$ is surjective and thus monic. Is an injective function.

An arrow is not nececarilly an isomorphism even if it is monic and epic.

22.2 2.5

(a) $\implies$ (b): Assume f is an isomorphism. We know that any isomorphism is a mono. $ff^{-1} = id_B$ so f has a left inverse and is a split epi.

(b) -¿ (c):

$$fid = f = (ff^{-1})f = f(f^{-1}f)$$

Derfor per mono er

$$f^{-1}f = id$$

(b) $\implies$ (c): Assume f is a mono and a split epi. There exists an e such that $fe = id$. To show that f is an epi let $i, j : B \to D$ and consider

$$if = jf \implies ife = jfe \implies i = j$$

So f is an epi. To show that f is a split mono we must find a g such that $gf = id_A$.

(c) $\implies$ (d):

(d) $\implies$ (a): Assume f is both a split mono and a split epi. We know that there exists a $g : B \to A$ such that $gf = id_A$ and a $g' : B \to A$ such that $fg' = id_B$. Putting this together we have.

$$g = gid_B = g(fg') = (gf)g' = id_A = g'$$

Thus $gf = id_A$ and $fg = id_B$. This makes g an inverse for f which makes f and isomorphism.

22.3 2.15

Assume A and B have a product in C: $A \times B = (P, p_1, p_2)$. Clearly this triple is an object in $\mathbf{C}$. Consider now any object $Q = (X, x_1, x_2)$ in $\mathbf{C}$. Since $A \times B$ is a product we have, from the UMP for products, that there is a unique morphism from Q to $A \times B$.

Assume that $\mathbf{C}_{A,B}$ has a terminal object (X, x_1, x_2). Observe that an arrow $f : Y, y_1, y_2 \to X, x_1, x_2$ is one that makes the following diagram commute.

$$\begin{tikzcd}
 & Y \arrow[dl, "y_1"'] \arrow[d, dashed, "f"] \arrow[dr, "y_1"] & \\
A & X \arrow[l, "x_1"] \arrow[r, "x_2"'] & B
\end{tikzcd}$$

The fact that (X, x_1, x_2) is a terminal objects mean that there exists a unique f that makes the diagram commute. But this is exactly the UMP for products and implies that (X, x_1, x_2) is a product for A and B in $\mathbf{C}$.

22.4 2.17

23 3

23.1 1

In any category $\mathbf{C}$ the below

$$\begin{tikzcd}
A \arrow[r, "c_1"] & C & B \arrow[l, "c_2"]
\end{tikzcd}$$

is a coproduct in $\mathbf{C}$ if and only

$$\begin{tikzcd}
A & C \arrow[l, "c_1"] \arrow[r, "c_2"] & B
\end{tikzcd}$$

is a product in $\mathbf{C}^{op}$. We know that this is a product if and only if the function

$$\begin{gathered}
\mathrm{Hom}(Z, C) \to \mathrm{Hom}(Z, A) \times \mathrm{Hom}(Z, B) \\
f \mapsto (c_1 \circ f, c_2 \circ f)
\end{gathered}$$

is an isomorphism. The dual of this function is

$$\begin{aligned}\mathrm{Hom}(C, Z) \to \mathrm{Hom}(A, Z) \times \mathrm{Hom}(B, Z) \\ f \mapsto (f \circ c_1, f \circ c_2\rangle\end{aligned}$$

And since duality preserves isomorphisms we have a coproduct if an only if the map is an isomorphism.

23.2 2

$$\begin{tikzcd}
''|M(A)|'' \arrow[r, "|f|"] & ''|N|'' \\
A \curlywedge &
\end{tikzcd}$$

23.3 3

23.4 4

23.5 6

23.6 10

23.7 11

23.8 12

23.9 13

23.10 14

24 5

25 1

26 2

27 3

28 4

29 6