Lecture 7: Functors, Natural Transformations, Equivalence of Categories

Lecturer: Max S. New
Scribe: Yuchen Jiang

September 17, 2025

Topics: isomorphisms, functors between categories and natural transformations.

1 Isomorphisms

Last time we talked about bijections. A function $f : X \to Y$ is a bijection if it's both injective and surjective; alternatively, it's bijective if it has a 2-sided inverse $f^{-1} : Y \to X$. We also showed that these definitions can be generalized to isomorphisms in an arbitrary category.

Definition 1. Given a morphism $f : \mathcal{C}(X, Y)$, f is an isomorphism if there exists a morphism $f^{-1} : \mathcal{C}(Y, X)$ such that $ff^{-1} = id_Y$ and $f^{-1}f = id_X$.

Remark. In this course, we write fg to denote $f \circ g$, meaning the composition of f and g. Both notations can be used interchangeably, but omitting the $\circ$ is briefer.

We can similarly generalize the notion of injectivity and surjectivity to morphisms in an arbitrary category.

1.1 Monomorphisms

The analog of injectivity in category theory is called a monomorphism.

Definition 2. $f : \mathcal{C}(X, Y)$ is a monomorphism ("is mono" or "is monic") if $\forall Z, \forall g, g' : \mathcal{C}(Z, X)$,

$$g = g' \iff fg = fg'$$

Conventionally monomorphic functions are written as $f : X \rightarrowtail Y$. We can comment on the definition that f is left cancellative. Monomorphisms enjoy the following lemma:

Lemma 1. $f : X \to Y \in \text{Set}$ *is mono* $\iff$ f *is injective*

Proof. Case split on $\iff$

Case $\implies$. Suppose f is mono. Let $x, x' \in X$ s.t. $fx = fx'$, WTS $x = x'$. We can use the same trick from last time to construct *constant functions* $Kx : 1 \to X$ and $Kx' : 1 \to X$ s.t.

$$\begin{tikzcd} 1 \arrow[r, shift left, "Kx"] \arrow[r, shift right, "Kx'"'] & X \arrow[r, "f"] & Y \end{tikzcd}$$

We can then compose both of them with f to take advantage of the fact that f is monic.

$$f \circ Kx = K(fx)$$
$$f \circ Kx' = K(fx')$$

Since RHSs are equal, we conclude that $Kx = Kx'$, and thus $x = x'$.

Case $\impliedby$. Suppose f is injective. Let $g, g' : Z \to X$ s.t. $fg = fg'$. Pick $z \in Z$. WTS $gz = g'z$. Then it suffices to show that

$$f(gz) = f(g'z)$$
$$(fg)z = (fg')z$$

, which is true since $fg = fg'$. □

1.2 Epimorphisms

There're two choices of the analog of surjectivity in category theory. The first one is called an epimorphism, which is the dual of a monomorphism.[1]

Definition 3. $f : \mathcal{C}(X, Y)$ is an epimorphism ("is epi" or "is epic") if $\forall Z. \forall g, g' : \mathcal{C}(Y, Z)$,

$$g = g' \iff gf = g'f$$

Conventionally epimorphic functions are written as $f : X \twoheadrightarrow Y$. A similar lemma to Lemma 1.1 holds for epimorphisms. The reason why it holds is that if $gf = g'f$ then $g = g'$ if applied to the image of f, which will only happen if the image of f is the entire set.

Conventionally monomorphic functions are written as $f : X \rightarrowtail Y$, and epimorphic functions are written as $f : X \twoheadrightarrow Y$.

1.3 Relations between Monomorphisms, Epimorphisms, and Isomorphisms

An interesting question rises from the definitions we've seen so far: since a function in Set is bijective if it's both injective and surjective, namely Inj + Surj $\cong$ Bijective, then

[1]What's the second one?

does this hold in general? In other words, does Mono + Ephi $\cong$ Iso? Surprisingly, the answer is NO. An counter-example could be a monoid homomorphism

$$i : \mathbb{N} \to \mathbb{Z} \in \text{Mon}$$

where i is both mono and epi, but not iso.

Remark. Being isomorphic does imply monomorphic and epimorphic. Proved by multiplying inverse f^{-1} of f on both sides for both epi and mono.

To fix this situation, we can strengthen the definition of either epimorphism and monomorphism. We can define a split epimorphism (a.k.a section).

Definition 4. $f : \mathcal{C}(X, Y)$ is a split epimorphism if it has a *section* $s : \mathcal{C}(Y, X)$ s.t.

$$fs = id_Y$$

Dually, we can define a split monomorphism (a.k.a retraction).

Definition 5. $f : \mathcal{C}(X, Y)$ is a split monomorphism if it has a *retraction* $r : \mathcal{C}(Y, X)$ s.t.

$$rf = id_X$$

Sections and retractions are defined in pairs, and we can say that section s has a retraction r and vice versa. The name "retraction" is easier to remember because it *retracts* the result of f to the identity function on X.

Now we can claim that Mono + Split Epi $\cong$ Iso.

Theorem 1. *If a morphism f is both a monomorphism and a split epimorphism, then it's an isomorphism.*

Proof. We start from f being split epi, meaning $f \circ s = id$. To prove that f is iso, we need to find its left inverse. Therefore, WTS $s \circ f = id$. We can apply f to both sides

$$\begin{aligned} f \circ (s \circ f) &= f \circ id \\ (f \circ s) \circ f &= f \circ id \\ id \circ f &= f \circ id \\ f &= f \end{aligned}$$

and we've proved that s is the left inverse of f that we're looking for. □

In fact, if we interpret Theorem 1.3 in the category Set, we can see that it's equivalent to the original formulation because of the following theorem.

Theorem 2. *In the category* Set*, all epis are split epis. (Axiom of Choice)*

But in the category Set, not all injective functions have a retract; therefore, not all monos are split monos. A counter example is $\varnothing \rightarrowtail Y$. The following theorem concludes the discussion.

Theorem 3. *Not all injective functions have a retract.*

Remark. One of the reasons why category theory is powerful is that 27:40.

We can also prove that Split Mono + Epi $\cong$ Iso. The arguments are basically the same as the proof of Theorem 1.3, except that we need to reason about retraction instead of section. This is a good example of duality, a significant concept in category theory.

We notice that the definition of mono and epi are dual to each other. Maybe we could have defined epimorphisms as monomorphisms in the *opposite* category.

2 Opposite Category

Definition 6. Let $\mathcal{C}$ be a category. The opposite category of $\mathcal{C}$, denoted by $\mathcal{C}^{op}$, is the category with the same objects as $\mathcal{C}$ and the same morphisms as $\mathcal{C}$, but with the direction of the morphisms reversed.

- $\mathcal{C}_0^{op} = \mathcal{C}_0$
- $\mathcal{C}_1^{op}(X, Y) = \mathcal{C}_1(Y, X)$ s.t.

$$\begin{aligned} id_X : \mathcal{C}_1^{op}(X, X) &= \mathcal{C}_1(X, X) \\ \text{and } \forall f : \mathcal{C}_1^{op}(X, Y), g &: \mathcal{C}_1^{op}(Y, Z), \\ g \circ^{op} f : \mathcal{C}_1^{op}(X, Z) &:= f \circ g \end{aligned}$$

where $f : \mathcal{C}_1(Y, X)$ and $g : \mathcal{C}_1(Z, Y)$, therefore $f \circ g$ is the correct direction.

Remark. We should define dual concepts in the opposite category whenever possible to not repeat ourselves and get some theorems for free. For example, epimorphism in $\mathcal{C}$ can be defined as monomorphism in $\mathcal{C}^{op}$. The benefit of defining concepts in opposite categories is that we can transform existing proofs into their duals, e.g. the proof of Theorem 1.3 can be transformed into the proof of split monomorphism + epimorphism $\cong$ isomorphism. We'll further elaborate on the definition of epimorphism after we introduce functors.

An important observation is that *op* is an involution on categories, i.e. $(\mathcal{C}^{op})^{op} = \mathcal{C}$.

3 Functors

Functors are mappings between categories.

Definition 7. A functor $F : \mathcal{C} \to \mathcal{D}$ is a function that maps objects and morphisms of $\mathcal{C}$ to objects and morphisms of $\mathcal{D}$ such that:

- $F_0 : \mathcal{C}_0 \to \mathcal{D}_0$

- $F_1 : \forall X, Y \in \mathcal{C}_0, \mathcal{C}_1(X,Y) \to \mathcal{D}_1(F_0X, F_0Y)$ s.t.

$$F_1(id_{\mathcal{C}}^X) = id_{\mathcal{D}}^{F_0(X)}$$
$$F_1(f \circ^{\mathcal{C}} g) = F_1(f) \circ^{\mathcal{D}} F_1(g)$$

These are homomorphism properties that functors must satisfy.

A functor F is a mapping that preserves the identity and composition of morphisms. For example, we can diagrammatically show that F preserves the composition of morphisms:

$$F\left(\begin{tikzcd} & Y \arrow[dd, "g"] \\ X \arrow[ur, "f"] \arrow[dr, "h"'] & \\ & Z \end{tikzcd}\right) = \begin{tikzcd} & FY \arrow[dd, "Fg"] \\ FX \arrow[ur, "Ff"] \arrow[dr, "Fh"'] & \\ & FZ \end{tikzcd}$$

If $h = g \circ f$, then $Fh = Fg \circ Ff$ in the diagram above.

3.1 Examples

Preorders

Given P, Q preorders, $f : P \to Q$ monotone function, then

$$F : \text{PreCat}(P) \to \text{PreCat}(Q)$$

is a functor equivalent to f where PreCat turns preorders into categories. The order relation is preserved during the upgrade from f to F, and all theorems in PreCat(P) can be lifted to PreCat(Q) by F.

Monoids

Given M, N monoids, $f : M \to N$ monoid homomorphism, then

$$F : \text{MonCat}(M) \to \text{MonCat}(N)$$

is a functor where MonCat upgrades monoids into one-object categories where elements of M are the morphisms, and the monoid operation is the composition of morphisms.

Forgetful Functor

Given a category of preorders Preorder, we can take its underlying set with U : Preorder $\to$ Set where

- $U_0P = |P|$

- $U_1 f = f$ but we forget the fact that f is monotone

U is called a forgetful functor, where U stands for "underlying". Similarly, we can write other forgetful functors that takes monoids to sets Mon $\to$ Set, finite set to set FinSet $\to$ Set, set of injective functions to set Inj $\to$ Set, graphs to set Graph $\to$ Set, etc.

Compiling STLC to Set

A functor from syntactic category of STLC to semantic category Set can be defined as STLC $\xrightarrow{[\![\cdot]\!]}$ Set.

To recap, the objects are types A and the morphisms are terms of one variable $x : A \to M : B$. To define the functor, we need to preserve identity and composition. Therefore, we define the functor $[\![\cdot]\!]$ as

$$\begin{aligned} \text{identity: } & [\![x : A \vdash x : A]\!](\tilde{x}) := \tilde{x} \\ \text{composition: } & [\![M[N/x]]\!] = [\![M]\!] \circ [\![N]\!] \end{aligned}$$

There is a different approach to define the category of STLC, which we discussed in the homework PS1, that is to define the notion of substitution $\gamma : \Delta \to \Gamma$ as a function that maps variables in Γ to terms in Δ. The functor $[\![\cdot]\!]$ is then defined as

$$\begin{aligned} \text{identity: } & [\![id_\Gamma]\!] := id \\ \text{composition: } & [\![\gamma \circ \delta]\!] = [\![\gamma]\!] \circ [\![\delta]\!] \end{aligned}$$

We'll be discussing many different functors from STLC into other semantic categories when we touch on the topic of initiality for STLC in the future.

List

An example from functional programming. Functor List : Set $\to$ Set is defined as

$$\begin{aligned} \text{List}(X) :=&\ \text{finite sets of } X \text{ elements} \\ \text{List}(f)([x_1, \dots]) =&\ [f(x_1), \dots] \end{aligned}$$

s.t. the following properties hold:

- $\text{List}(id) = id$
- $\text{List}(f \circ g) = \text{List}(f) \circ \text{List}(g)$

Powerset and Contravariant Functor

The powerset operation on set $\mathscr{P}$: Set $\to$ Set is a functor that takes a set to its powerset.

$$\begin{aligned} & \mathscr{P}_0(X) = \text{the powerset of } X \\ & \mathscr{P}_1(f : X \to Y) : \mathscr{P}(X) \to \mathscr{P}(Y) \text{ s.t.} \\ & \mathscr{P}_1(f)(S) = \{y \in Y \mid \exists x \in S, f(x) = y\} \end{aligned}$$

Changing exists into forall in the last definition also gives us a functor.

An interesting fact is that the same operation on objects can be made into a contravariant functor. $\mathscr{P} : \text{Set}^{op} \to \text{Set}$ is a contravariant functor defined as

$$\begin{aligned}
&\mathscr{P}_0(X) = \mathscr{P}(X) \\
&\mathscr{P}_1(f : Y \to X) : \mathscr{P}(X) \to \mathscr{P}(Y) \text{ s.t.} \\
&\mathscr{P}_1(f)(S \subseteq X) = \{y \mid f(y) \in S\} \text{ (preimage)}
\end{aligned}$$

A contravariant functor is a functor that reverses the direction of the morphisms. The powerset operation defined above is a contravariant functor exactly because it's taking things out of an opposite category, which flips the arrows. To be clear, as long as the direction of morphisms is flipped, the functor is considered a contravariant functor; it doesn't matter whether it's taking things out of an opposite category or into one.

Monomorphisms and Epimorphisms

Let's revisit monomorphisms and epimorphisms. Given a function $f : X \to Y$, if f is mono, then we can write $f : X \rightarrowtail Y$; if f is epi, then we can write $f : X \twoheadrightarrow Y$. Then what if we compose morphisms that have these properties?

- $id : X \rightarrowtail\!\!\!\!\twoheadrightarrow Y$ because id is mono and epi.
- Given $f : X \rightarrowtail Y$ and $g : Y \rightarrowtail Z$, then $g \circ f : X \rightarrowtail Z$. We conclude that monos are closed under composition. Similarly, given $f : X \twoheadrightarrow Y$ and $g : Y \twoheadrightarrow Z$, then $g \circ f : X \twoheadrightarrow Z$.
- Given $g : Y \rightarrowtail Z$ and $g \circ f$ is mono, then f is mono, as shown in the last homework.

Monomorphisms and epimorphisms can be defined as functors as well.

$$\begin{aligned}
\text{Mono}(\mathcal{C})_0 &:= \mathcal{C}_0 \\
\text{Mono}(\mathcal{C})_1(X, Y) &:= \{f \in \mathcal{C}_1(X, Y) \mid f \text{ mono}\} \\
\text{Epi}(\mathcal{C}) &:= \text{Mono}(\mathcal{C}^{op})^{op}
\end{aligned}$$

Scriber remark: take a minute to justify the need of two opposite categories in the definition of Epi($\mathcal{C}$) and understand more about duality. The following diagrams might help.

$$\begin{tikzcd}
\bullet \arrow[r, shift left, "g"] \arrow[r, shift right, "g'"'] & \bullet \arrow[r, tail, "f"] & \bullet
\end{tikzcd}
\qquad
\begin{tikzcd}
\bullet & \bullet \arrow[l, shift right, "g"'] \arrow[l, shift left, "g'"] & \bullet \arrow[l, two heads, "f"']
\end{tikzcd}$$

3.2 Category of Categories

The category of categories CAT is a category where the objects are categories and the morphisms are functors.

$$\begin{aligned}
\mathrm{CAT}_0 &:= \text{categories} \\
\mathrm{CAT}_1(\mathcal{C}, \mathcal{D}) &:= \text{functors } \mathcal{C} \to \mathcal{D} \\
id &: \mathcal{C} \to \mathcal{C} \\
G \circ F &: \mathcal{C} \to \mathcal{E} \text{ given } F : \mathcal{C} \to \mathcal{D} \text{ and } G : \mathcal{D} \to \mathcal{E} \\
(G \circ F)_0(X) &= G_0(F_0(X)) \\
(G \circ F)_1(f) &= G_1(F_1(f))
\end{aligned}$$

To show that CAT is a category, we need to show that it satisfies the properties of identity and composition, which is labor-intensive. We'll show some strategies to deal with this later this semester.

Special Categories

We can define structures that look like simply-typed lambda calculus.

We start with the *terminal category* **1**.

$$\begin{aligned}
\mathbf{1}_0 &:= \{*\} \\
\mathbf{1}_1(*, *) &:= \{id\}
\end{aligned}$$

There's a nice property that the terminal category has:

$$\mathbf{1} \to \mathcal{C} \cong \mathcal{C}_0$$

The next is the empty category, which is also called the *initial category* **0**.

$$\mathbf{0}_0 := \varnothing$$

Similarly, the empty category satisfies:

$$\mathbf{0} \to \mathcal{C} \cong \mathbf{1}$$

Next is the *product category.* Given $\mathcal{C}, \mathcal{D}$, the product category $\mathcal{C} \times \mathcal{D}$ is a category where the objects are pairs of objects from $\mathcal{C}$ and $\mathcal{D}$ and the morphisms are pairs of morphisms from $\mathcal{C}$ and $\mathcal{D}$.

$$\begin{aligned}
(\mathcal{C} \times \mathcal{D})_0 &= \mathcal{C}_0 \times \mathcal{D}_0 \\
(\mathcal{C} \times \mathcal{D})_1((X, Y), (A, B)) &= \mathcal{C}_1(X, Y) \times \mathcal{D}_1(A, B)
\end{aligned}$$

The product category works in a way that $\mathcal{C}$ and $\mathcal{D}$ are "independent" of each other, and they don't interleave with each other. Written as a property:

$$\mathcal{E} \to \mathcal{C} \times \mathcal{D} \cong (\mathcal{E} \to \mathcal{C}) \text{ and } (\mathcal{E} \to \mathcal{D})$$

A less commonly used category is the *disjoint union* of categories. Given $\mathcal{C}, \mathcal{D}$, the disjoint union category $\mathcal{C} \uplus \mathcal{D}$ is defined as

$$\begin{aligned}
(\mathcal{C} \uplus \mathcal{D})_0 &:= \mathcal{C}_0 \uplus \mathcal{D}_0 \\
(\mathcal{C} \uplus \mathcal{D})_1(X, Y) &:= \mathcal{C}_1(X, Y) \\
(\mathcal{C} \uplus \mathcal{D})_1(A, B) &:= \mathcal{D}_1(A, B) \\
(\mathcal{C} \uplus \mathcal{D})_1(X, B) &:= \varnothing \\
(\mathcal{C} \uplus \mathcal{D})_1(A, Y) &:= \varnothing
\end{aligned}$$

And a similar property holds:

$$\mathcal{C} \uplus \mathcal{D} \to \mathcal{E} \cong (\mathcal{C} \to \mathcal{E}) \text{ and } (\mathcal{D} \to \mathcal{E})$$

Lastly, the most interesting case, the category of the function type $\mathcal{C} \Rightarrow \mathcal{D}$, also denoted as $\mathcal{D}^{\mathcal{C}}$.

$$\begin{aligned}
\mathcal{D}^{\mathcal{C}}_0 &:= \text{Functors from } \mathcal{C} \text{ to } \mathcal{D} \\
\mathcal{D}^{\mathcal{C}}_1(F, G) &:= \text{Natural transformations from } F \text{ to } G
\end{aligned}$$

What is a *natural transformation*? Let's start by looking at the properties that we want the morphisms of $\mathcal{D}^{\mathcal{C}}_1(F, G)$ to satisfy. In STLC, we want the functions to satisfy currying:

$$\mathcal{E} \to \mathcal{D}^{\mathcal{C}} \cong \mathcal{E} \times \mathcal{C} \to \mathcal{D}$$

in which $\mathcal{E} \times \mathcal{C} \to \mathcal{D}$ is called a *bifunctor*. Now with this property in mind, we'll define what's a natural transformation.

4 Natural Transformations

The morphisms of $\mathcal{D}^{\mathcal{C}}_1(F, G)$ are the natural transformations α between functors F and G.

- $\forall X \in \mathcal{C}_0, \alpha_X \in \mathcal{D}(FX, GX)$
- Natuality: $\forall f : \mathcal{C}_1(X, Y)$, the diagram (called the *naturality square*)

$$\begin{tikzcd}
FX \arrow[r, "\alpha_X"] \arrow[d, "Ff"'] & GX \arrow[d, "Gf"] \\
FY \arrow[r, "\alpha_Y"'] & GY
\end{tikzcd}$$

commutes, regardless of the choice of X and f.

There's a lot of different ways to think about the property of natural transformations (which is also known as the *uniformity condition*). People with a programming language background might think of natural transformations as closely related to parametricity: type constructors have to behave uniformly across all instantiations, or rather, generic in the $\mathcal{C}_0$ component in that the structure doesn't depend on f in any way. The other way to motivate it is that natural transformations are the morphisms in the category of functors, like what we did in the previous section.

Historically, the category theory was first invented by Eilenberg and MacLane for the sole purpose of generalizing and formalizing the notion of natural transformations. The focus back then was instead natural isomorphisms.

Finally, we'll define identity and composition of natural transformations. The identity natural transformation is defined as

$$id : F \Rightarrow F$$
$$id_X : FX \to FX$$

s.t. the following diagram commutes

$$\begin{tikzcd}
FX \arrow[r, "id_X"] \arrow[d, "Ff"'] & FX \arrow[d, "Ff"] \\
FY \arrow[r, "id_Y"'] & FY
\end{tikzcd}$$

And the composition of natural transformations is defined as given $\alpha : F \Rightarrow G$ and $\beta : G \Rightarrow H$,

$$\beta \circ \alpha : F \Rightarrow H$$
$$(\beta \circ \alpha)_X : FX \to HX$$

s.t. the following diagram commutes

$$\begin{tikzcd}
FX \arrow[r, "\alpha_X"] \arrow[d, "Ff"'] & GX \arrow[r, "\beta_X"] \arrow[d, "Gf"] & HX \arrow[d, "Hf"] \\
FY \arrow[r, "\alpha_Y"'] & GY \arrow[r, "\beta_Y"'] & HY
\end{tikzcd}$$

It's beneficial to unpack the underlying proofs that the diagram shows.