Recall that given categories \( C \) & \( D \), a functor \( F: C \to D \) is a map sending objects \( c \in C \) to objects \( F(c) \in D \), morphisms \( f: c \to c' \) in \( C \) to morphisms \( F(f): F(c) \to F(c') \) in \( D \), preserving composition \( F(f \circ g) = F(f) \circ F(g) \) & identities \( F(1_c) = 1_{F(c)} \).

There are many "forgetful functors" going from categories of "funny" mathematical gadgets to categories of less funny ones, forgetting some extra properties, structure, or stuff.

**Ex.** \( U: \text{Grp} \to \text{Set} \) sends any group \( G \) to its underlying set, and any homomorphism \( f: G \to H \) to its underlying function.

**Ex.** Given categories \( C \) & \( D \), there's a category \( C \times D \), whose objects are ordered pairs \( (c,d) \) with \( c \in C \), \( d \in D \), and morphisms are ordered pairs \( (f,g) \) with \( f \) a morphism in \( C \), \( g \) a morphism in \( D \): given \( f: c \to c' \) in \( C \) and \( g: d \to d' \) in \( D \) then \( (f,g): (c,d) \to (c',d') \). We define \( (f',g') \circ (f,g) = (f' \circ f, g' \circ g) \) and \( 1_{(c,d)} = (1_c,1_d) \).

In fact \( C \times D \) is the product of the objects \( C, D \in \mathbf{Cat} \), which is the category with
- (small) categories as objects
- functors as morphisms
Among other things this means we have projections
\[ C \times D \]
\[ \pi \]
\[ C \quad D \]

Set is a large category, but we can still define \( \text{Set}^2 = \text{Set} \times \text{Set} \), with pairs of sets as objects.

In the chart, let \( \pi : \text{Set}^2 \to \text{Set} \) be the projection onto the first component.
\[ (S,T) \mapsto S \]

Functions can be nice in two ways: one-to-one & onto.

Functors can be nice in three ways:

**Def** A functor \( F : C \to D \) is faithful if for any \( c, c' \in C \)
\[ F : \text{hom}(c, c') \to \text{hom}(F(c), F(c')) \]
is one-to-one.

**Def** A functor \( F : C \to D \) is full if for any \( c, c' \in C \)
\[ F : \text{hom}(c, c') \to \text{hom}(F(c), F(c')) \]
is onto.

**Def** A functor \( F : C \to D \) is essentially surjective if for any \( d \in D \), there exists \( c \in C \) such that \( F(c) \simeq d \), meaning there exists an isomorphism \( g : F(c) \to d \) in \( D \).

**Ex** Compare \( \text{FinVect}_{\mathbb{F}} \) to this category \( C \), with
- \( \mathbb{R}, \mathbb{R}^2, \mathbb{R}^3, \ldots \) as objects
- all linear maps between these as morphisms
\[ F : C \to \text{FinVect} \]
\[ \mathbb{R}^n \to \mathbb{R}^n \]
and similarly for morphisms:
\[ f : \mathbb{R} ^n \to \mathbb{R} ^n \]

This is faithful & full, not surjective on objects, but essentially surjective.

Later we'll define "equivalent" categories & see that if \( F : C \to D \) is faithful, full, & essentially surjective then \( C \) & \( D \) are equivalent.
We say:

**Def:** A functor \( U: C \to D \) forgets nothing if it is faithful, full, and essentially surjective.

- \( U \) forgets (at most) properties if \( U \)'s faithful & full.
- \( U \) forgets (at most) structure if it's faithful.
- In general, we say \( U \) forgets (at most) stuff.

\[ U: \text{Grp} \to \text{Set} \] forgets (at most) structure.

It's faithful: given \( f, f': G \to G' \) in Grp, \( U(f) \circ U(f') = f \circ f' \).

It's not full: there are usually functions \( f: U(G) \to U(G') \) that don't come from group homomorphism, e.g., \( f(gh) = f(g)f(h) \) or \( f(1) = 1 \).

\[ U_2: \text{AbGrp} \to \text{Grp} \] forgets (at most) properties: the comm. law is forgotten.

This is faithful and also full: if you have any group homomorphism \( f: U_2(A) \to U_2(A') \), then \( U(f) \) for some homomorphism of abelian groups \( f: A \to A' \).

But it's not ess. surjective: if \( G \) is nonabelian, \( G \cong U_2(A) \) for some \( A \in \text{AbGrp} \).

\[ U_3: \text{Set}^2 \to \text{Set} \]

forgets stuff: \( U_3(S, S') = S \) if it forgets the second set in the pair.

Technically, it's not faithful:

we can have 2 different morphisms \( (f, g), (f', g'): (S, S') \to (T, T') \)

with \( U_3(f, g) = f = U_3(f', g') \).

In our chat, every forgetful functor \( U: C \to D \) has a "left adjoint" \( F: D \to C \) which freely creates the stuff, structure or properties that \( U \) forgets.

\[ F: \text{Set} \to \text{Grp} \] takes a set \( S \) and forms the free group on \( S \), \( F(S) \).

\[ F_2: \text{Grp} \to \text{AbGrp} \] abelianizes any group \( G \), forming

\[ F_2(G) = \frac{G}{\langle xyx^{-1}y^{-1} \rangle} \]

normal subgroup gen. by these elements

\[ L_0: \text{Set} \to \text{Set}^2 \]

\[ S \mapsto (S, \emptyset) \]
To define adjoint functors (and many other things) we need...

Natural Transformations

Given 2 functors \( F, G : C \to D \) we can define a natural transformation \( \alpha : F \Rightarrow G \)

\[
\begin{array}{ccc}
F(x) & \xrightarrow{F(f)} & F(y) \\
\downarrow{\alpha_x} & & \downarrow{\alpha_y} \\
G(x) & \xrightarrow{G(f)} & G(y)
\end{array}
\]

This square commutes:

\( \alpha \) is a natural transformation.

Definition: Given functors \( F, G : C \to D \) a transformation \( \alpha : F \Rightarrow G \) is a function sending each object \( x \in C \) to a morphism \( \alpha_x : F(x) \to G(x) \). We say \( \alpha \) is a natural transformation if for each morphism \( f : x \to y \) in

\[
\begin{array}{ccc}
F(x) & \xrightarrow{F(f)} & F(y) \\
\downarrow{\alpha_x} & & \downarrow{\alpha_y} \\
G(x) & \xrightarrow{G(f)} & G(y)
\end{array}
\]

Proposition: Given categories \( C \) & \( D \) there's a category, the functor category \( D^C \), with:

- objects being functors \( F : C \to D \)
- morphisms being natural transformations \( \alpha : F \Rightarrow G \)

In \( D^C \) we compose \( \alpha : F \Rightarrow G \), \( \beta : G \Rightarrow H \) to get \( \beta \circ \alpha \) as follows:

\( \beta \circ \alpha : F \Rightarrow H \) given by \( (\beta \circ \alpha)_x = \beta_{G(x)} \circ \alpha_x \).

In \( D^C \) the identity \( 1_F : F \Rightarrow F \):

\( 1_F : F(x) \to F(x) \) \( x \in C \) is

given by \( 1_{F(x)} \).

Proof:

Check that the composite \( \beta \circ \alpha \) is natural:

given \( f : x \to y \) in \( C \), want this to commute:

\[
\begin{array}{ccc}
F(x) & \xrightarrow{F(f)} & F(y) \\
\downarrow{(\beta \circ \alpha)_x} & & \downarrow{(\beta \circ \alpha)_y} \\
H(x) & \xrightarrow{H(f)} & H(y)
\end{array}
\]

(continued)
Pl: (continued)

\[ \begin{array}{ccc}
F(x) & \xrightarrow{F(f)} & F(y) \\
\downarrow \alpha & \Downarrow \beta & \downarrow \gamma \\
\phi(x) & \xrightarrow{\phi(f)} & \phi(y) \\
\downarrow \beta & \Downarrow \gamma & \downarrow \alpha \\
H(x) & \xrightarrow{H(f)} & H(y)
\end{array} \]

\[ \text{\textcircled{2 \text{natural \rightarrow \text{this commute}}} \text{\textcircled{2 \text{natural \rightarrow \text{this commute}}}} \]

So just as given 2 sets \( X, Y \), there's a set \( Y^X \) of all \( f \)s \( f : X \to Y \)

2 categories \( X, Y \), there's a category \( Y^X \) of all functors \( F : X \to Y \)

Given 2 sets \( X \& Y \), they have a product:
\[ X \times Y = \{ (x, y) : x \in X, y \in Y \} \]

Notice \( X \times Y \neq Y \times X \)
but \( X \times Y \cong Y \times X \)

And there's a specific "good" isomorphism
\[ \alpha : X \times Y \xrightarrow{\cong} Y \times X \]
\[ (x, y) \mapsto (y, x) \]

It's "good" because it's natural, in the sense we just defined.

There are 2 functors from \( \text{Set}^2 \) to \( \text{Set} \),
\[ F : (X, Y) \mapsto X \times Y \]
\[ G : (X, Y) \mapsto Y \times X \]

and \( \alpha \) is a natural transformation from \( F \) to \( G \).

In fact it's a "natural isomorphism":

**Def.** If \( F, G : C \to D \) are functors and \( \alpha : F \Rightarrow G \) is a nat. tran., we say \( \alpha \) is a natural isomorphism if \( \alpha_x : F(x) \to G(x) \) is an isomorphism \( Y \times x \in C \).

**Prop.** \( \alpha : F \Rightarrow G \) is a natural isomorphism iff it has an inverse \( \alpha^{-1} : G \Rightarrow F \) in \( D^C \).

**Pf.**

Key idea: \( (\alpha^{-1})_x = (\alpha_x)^{-1} \)
Prop Suppose \( C \) is a category with binary products: any pair of objects \( x, y \in C \) has a product. Then we can choose, for any pair \( x, y \in C \), a specific product:

\[
\begin{align*}
x \times y & \rightarrow x \\
y & \rightarrow x \\
\end{align*}
\]

and then there is a functor \( x : C^2 \rightarrow C \)

\[
(x, y) \mapsto x \times y
\]

In fact there are 2 functors:

\[
\begin{align*}
F : C^2 & \rightarrow C \\
(x, y) & \mapsto x \times y
\end{align*}
\]

\[
\begin{align*}
G : C^2 & \rightarrow C \\
(x, y) & \mapsto y \times x
\end{align*}
\]

and these are naturally isomorphic.

We say "products are commutative up to natural isomorphism."

Also, products are associative up to natural isomorphism:

\[
\alpha_{x,y,z} : (x \times y) \times z \rightarrow x \times (y \times z)
\]

(Just keep using universal property of product.)

Def A cartesian category is a category with binary products and a terminal object. (I.e. it's a category where any finite set of objects has a product — a finite products category)

One can show that in a cartesian category, we have natural isomorphisms:

\[
\begin{align*}
l_x & : 1 \times x \rightarrow x \\
r_x & : x \times 1 \rightarrow x
\end{align*}
\]

All this works similarly in a cat \( W \) with finite coproducts:

\[
\begin{align*}
B_{x,y} & : x + y \rightarrow y + x \\
\alpha_{x,y,z} & : (x + y) + z \rightarrow x + (y + z) \\
l_x & : 0 + x \rightarrow x \\
r_x & : x + 0 \rightarrow x
\end{align*}
\]
In the case $\mathcal{C} = \text{FinSet}$ (finite sets & functions) these give familiar land of arithmetic: $\mathbb{N}$ is the set of isomorphism classes of objects in FinSet.

Another example: A group is a category $\mathcal{G}$ with one object and with all morphisms invertible:

What's a functor $F: \mathcal{G} \to \text{Set}$?

$F$ picks out a set $X = F(\mathbb{1})$ and for each group element $f$ it picks out a function $F(f): X \to X$ s.t. $F(ff') = F(f)F(f')$ & $F(\mathbb{1}) = 1_X$.

So: $X$ is a set acted by the group $\mathcal{G}$, or a $\mathcal{G}$-set.

So: a functor $F: \mathcal{G} \to \text{Set}$ is a $\mathcal{G}$-Set. What's a natural transformation between 2 such functors?