Category Theory 3

Eugenia Cheng
Department of Mathematics, University of Sheffield
E-mail: e.cheng@sheffield.ac.uk

Draft: 8th November 2007

Contents

3 Functors: definition and examples

One of the basic principles of Category Theory is that we should study structures *together with their structure-preserving maps*. We should look at the "totality" of these structures and their maps, rather than just looking at individual structures in isolation.

Functors are the structure-preserving maps of categories. You may have already guessed how they should be defined; in any case this definition should remind you of the definition of group homomorphism, and we'll soon see that that isn't exactly a coincidence.

3.1 The definition

Recall our summarised definition of a category:

- Data – objects and morphisms
- Structure – composition and identities
- Properties – associativity and unit axioms

Thus a "stucture-preserving map" should preserve, that is respect, composition and identities. We now make that precise.

Definition 3.1. Let $\mathcal{C}$ and $\mathcal{D}$ be categories. A *functor* $F : \mathcal{C} \longrightarrow \mathcal{D}$ associates:

- to every object $X \in \mathcal{C}$ an object $FX \in \mathcal{D}$, and
- to every morphism $f \in \mathcal{C}(X, Y)$ a morphism $Ff \in \mathcal{D}(FX, FY)$

such that

- for all morphisms $X \xrightarrow{f} Y \xrightarrow{g} Z$, $F(g \circ f) = Fg \circ Ff$, and
- for all $X \in \mathcal{C}$ $F(1_X) = 1_{FX}$.

The conditions in this definition are generally referred to as "functoriality" conditions, and should distinctly remind you of the axioms for a group homomorphism.

3.2 Examples: functors on other mathematical structures expressed as categories

[Examples]

- If $\mathcal{C}$ and $\mathcal{D}$ are groups (expressed as one-object categories) then a functor $\mathcal{C} \longrightarrow \mathcal{D}$ is precisely a group homorphism. There is only one object in each category so the functor's action on objects gives us no information – it simply sends the single object in $\mathcal{C}$ to the single object in $\mathcal{D}$. The morphisms in $\mathcal{C}$ are the group elements, so the functor must map group elements of $\mathcal{C}$ to group elements of $\mathcal{D}$; functoriality then corresponds precisely to the axioms for a group homorphism.
- If $\mathcal{C}$ and $\mathcal{D}$ are monoids, that is, one-object categories, then a functor $\mathcal{C} \longrightarrow \mathcal{D}$ is precisely a monoid morphism.
- If $\mathcal{C}$ and $\mathcal{D}$ are posets (expressed as categories with a morphism $x \longrightarrow y$ whever $x \leq y$ then a functor $\mathcal{C} \longrightarrow \mathcal{D}$ is precisely an order-preserving map.

3.3 Examples: functors on large categories of mathematical structures

Forgetful functors

Forgetful functors forget all or part of the structure in question. That's not exactly a definition; later we'll discover that forgetful functors are *faithful*, and some people consider that all faithful functors deserve to be called "forgetful"[1].

- There is a forgetful functor

$$\mathbf{Gp} \longrightarrow \mathbf{Set}$$

  which sends a group to its underlying set; the action on morphisms is induced in the obvious way[2]. This is called a "forgetful functor" because it forgets things – it forgets the group structure on a set.

- Similarly there are forgetful functors

$$\begin{array}{rcl} \mathbf{Ring} & \longrightarrow & \mathbf{Set} \\ \mathbf{Vect} & \longrightarrow & \mathbf{Set} \\ \mathbf{Ab} & \longrightarrow & \mathbf{Set} \\ \mathbf{Top} & \longrightarrow & \mathbf{Set} \\ \mathbf{Mnd} & \longrightarrow & \mathbf{Set} \\ \mathbf{Poset} & \longrightarrow & \mathbf{Set} \\ \mathbf{Set}_* & \longrightarrow & \mathbf{Set} \end{array}$$

  and many more – this list cannot possibly be exhaustive.

- Here are some forgetful functors that only forget part of the structure:

$$\begin{array}{rcl} \mathbf{Ring} & \longrightarrow & \mathbf{Ab} \\ \mathbf{Ring} & \longrightarrow & \mathbf{Mnd} \\ \mathbf{Vect} & \longrightarrow & \mathbf{Ab} \\ \mathbf{Top}_* & \longrightarrow & \mathbf{Top} \end{array}$$

- Here are some examples of functors that just forget some *property*:

$$\begin{array}{rcl} \mathbf{Ab} & \longrightarrow & \mathbf{Gp} \\ \mathbf{Haus} & \longrightarrow & \mathbf{Top} \end{array}$$

  Forgetting a "property" is curious because you still *have* that property, whether you remember you do or not. If you forget that a group is abelian, it's still abelian! It's a bit like turning a blind eye to the fact that it is

[1] Category Theorists often try to use words from ordinary life like this to help with the intuitions behind the technical concepts. This example shows that the analogies should not be taken too far!

[2] We often sort of vaguely omit to mention what the functor does on morphisms when it's really obvious. Beware, however, that this can get us into trouble in situations where it's not obvious at all what it does on morphisms.

abelian[3]. In a minute we'll see that these examples are in fact giving *subcategories*.

Free functors

Free functors go in the opposite direction of forgetful functors: they take something with little or no structure, and create something with more structure. The new extra structure is added in "freely". Again, this isn't exactly a technical definition, but if we accept the notion of "forgetful" functor we can actually define the notion of "free" by a wonderful and intimate relationship between free and forgetful functors. This is the notion of *adjunction*, which we'll get to later. We'll start with the free functor for building monoids; the free group functor is probably more famous, but much harder to describe.

- There is a free functor

  $$\mathbf{Set} \longrightarrow \mathbf{Mnd}$$

  that takes a set X and produces the *free monoid* on it. How do we make a monoid "freely" from a set? Well, we have to ask ourselves what structure we are *lacking*, and in this case the answer is – unit and multiplication. This is what a monoid has that a set doesn't have, so we're just going to throw it in freely. The result is a set X^* which consists of "words in X", that is, finite lists of elements of X, possibly with repetitions. Order matters, since we're not doing commutative monoids. The original elements of X all appear as one-letter words. And there has to be an "empty word", which is going to be the unit. The multiplication in this monoid is then "concatenation" – we can just stick two of these list end to end and it makes a new list. Sticking the empty list onto the beginning or end of a list doesn't change anything, so the empty list is the unit for this monoid.

  It then follows that a monoid morphism $X^* \longrightarrow Y^*$ is *completely determined* by where the individual elements of X (the one-letter words) are sent, so given any function $X \longrightarrow Y$ we get a monoid morphism $X^* \longrightarrow Y^*$, and this finishes the definition of the free monoid functor. (We can easily check functoriality.)

- We can do something similar but harder to get a free functor

  $$\mathbf{Set} \longrightarrow \mathbf{Gp}.$$

  The reason it's harder is that we have to make sure everything has an *inverse* and this complicates matters rather.

- Similarly we have a free functor

  $$\mathbf{Set} \longrightarrow \mathbf{Ring}$$

[3]If you are so inclined, you can have long arguments with others about whether it is more or less forgetful to forget a property rather than a structure.

but we also have a more subtle free functor

$$\mathbf{Mnd} \longrightarrow \mathbf{Ring}.$$

3.4 Examples: functors from very small categories

Functors from "very small" categories pick out structure of that shape in the target category.

- Let $\mathbf{1}$ be a category with one object and one (necessarily identity) morphism. Then a functor $F : \mathbf{1} \longrightarrow \mathcal{C}$ just picks out an object of $\mathcal{C}$ – we just have to decide what the action of F is on objects, since on morphisms the identity will then have to go to the relevant identity. So we just have to decide where F sends the single object of $\mathbf{1}$. It's a bit like the fact that a function $\{*\} \longrightarrow X$ is just an element of the set X.
- Similarly functor from the category

  $$\cdot \longrightarrow \cdot$$

  to $\mathcal{C}$ just picks out a morphism – any morphism – in $\mathcal{C}$.
- A functor from the category

  $$\begin{array}{ccc} \cdot & \longrightarrow & \cdot \\ \downarrow & & \downarrow \\ \cdot & \longrightarrow & \cdot \end{array}$$

  to $\mathcal{C}$ just picks out a commutative square in $\mathcal{C}$

Inspired by such examples, if $\mathbb{D}$ is a small category, we often refer to functors $\mathbb{D} \longrightarrow \mathcal{C}$ as "diagrams of shape $\mathbb{D}$ in $\mathcal{C}$".

3.5 Contravariant functors

Definition 3.2. A *contravariant* functor F is a functor

$$F : \mathcal{C}^{\mathrm{op}} \longrightarrow \mathcal{D}.$$

What does this mean in practice? We usually secretly just think of this as a functor from $\mathcal{C}$ to $\mathcal{D}$ that turns morphisms round instead of keeping them facing the same way (which is what the "contravariant" is supposed to mean). The point is:

$$f : X \longrightarrow Y \in \mathcal{C}$$

is to be thought of as

$$f : Y \longrightarrow X \in \mathcal{C}^{\mathrm{op}}$$

so applying the functor F gives

$$Ff : FY \longrightarrow FX \in \mathcal{D}$$

so effectively we have

$$X \xrightarrow{f} Y \quad \mapsto \quad FY \xrightarrow{Ff} FX.$$

We also have to be a bit careful about functoriality, but here's a hint – if you draw the morphisms out as arrows with marked source and target, you can't go wrong. The point now is:

a composable pair in $\mathcal{C}$

$$X \xrightarrow{f} Y \xrightarrow{g} Z \in \mathcal{C}$$

is to be secretly thought of as

$$Z \xrightarrow{g} Y \xrightarrow{f} X \in \mathcal{C}^{\mathrm{op}}$$

so applying the functor gives

$$FZ \xrightarrow{Fg} FY \xrightarrow{Ff} FX \in \mathcal{D}$$

so functoriality must say

$$\forall\ X \xrightarrow{f} Y \xrightarrow{g} Z \in \mathcal{C},\ F(g \circ f) = Ff \circ Fg.$$

A non-contravariant functor is sometimes referred to as *covariant* for emphasis[4].

3.6 Example: vector spaces

The category $\mathbf{Vect}_k$ of vector spaces over a fixed field k of scalars is a lovely category with all sorts of interesting structure. One such piece of structure is that every vector space V has a *dual* V^*. The dual space of V is defined to be the space of all linear maps $V \longrightarrow k$. In fact this notion of "dual" can be formalised and we can look for analogous things in other categories[5].

There are also duals for linear maps – given a linear map

$$f : V \longrightarrow W$$

we get a dual map

$$f^* : W^* \longrightarrow V^*$$

[4]This "co" prefix is not to be confused with "co" prefixes which mean we've taken the dual of the usual notion. It's true that covariant functors are dual to contravariant ones, but in this case it's really the covariant ones that are the starting point.

[5]This is not, unfortunately, to be confused with the dual of a *category*.

which simply sends the map

$$W \xrightarrow{h} k$$

to the composite map

$$V \xrightarrow{f} W \xrightarrow{h} k.$$

The fact that the dual map goes "backwards" is a source of great confusion to many a student in linear maths classes; however if you write out the maps as arrows with their sources and targets labelled, it's hard to go wrong.

Anyway, this backwards business is *really* saying that there's a contravariant functor $\mathbf{Vect}_k^{\mathrm{op}} \longrightarrow \mathbf{Vect}_k$ which assigns:

- to each vector space V its dual space V^*
- to each linear map $f : V \longrightarrow W$ its dual $f^* : W^* \longrightarrow V^*$.

Functoriality is then the fact that dual maps compose, although you have to be a bit careful with the directions:

$$(g \circ f)^* = f^* \circ g^*.$$

In fact we can somewhat copy this example for categories other than **Vect** and get the general notion of *representable functors*[6].

3.7 Example: presheaves

Definition 3.3. A functor $\mathcal{C}^{\mathrm{op}} \longrightarrow \mathbf{Set}$ is called a *presheaf* on $\mathcal{C}$.

This word is borrowed from topology because this is a generalisation of the original notion of "presheaf", the idea being that's it's the basic data behind a "sheaf" – the sheaves are those presheaves that satisfy some more conditions. We'll go into this a bit more later[7].

Presheaves turn out to be very important for categorical reasons. It may seem a bit odd at this point to be so interested in the contravariant functors into set rather than the covariant ones, but we'll later see that taking the presheaves on $\mathcal{C}$ gives us a very special relationship with $\mathcal{C}$ – it gives us the "colimit completion". This is a bit like the closure of an open set, and we even get a sense in which $\mathcal{C}$ is "dense" in its closure, just like the rationals are dense in the reals.

[6]The case of **Vect** is a bit more special because the set of linear maps between two spaces actually itself forms a *vector space*, and not just a set. This is an example of enrichment/closedness.

[7]If you know some topology you may think that "presheaf" usually means something that looks completely different. Well it's actually the same if you look at it the right way! Given a topological space X we can form a category from its open sets – it's the poset of open sets, ordered by inclusion. Then a presheaf (in the topological sense) is exactly a presheaf on this category in the above sense.

3.8 Examples: representable functors

Here is a very useful class of functor, which also gives an example of contravariant functors. Let $\mathcal{C}$ be a locally small category[8] and fix U an object in $\mathcal{C}$. Then there are two "representable functors" we can make, one which is covariant and one which is contravariant. We'll do the covariant one first so we can get the hang of things.

Covariant representable

Given $U \in \mathcal{C}$ we have a functor $H^U : \mathcal{C} \longrightarrow \mathbf{Set}$ defined as follows:

- on objects $X \mapsto \mathcal{C}(U, X)$;
- on morphisms $f \mapsto f \circ \_$.

We'd better immediately explain what on earth this notation means. Suppose we have a morphism $f : X \longrightarrow Y \in \mathcal{C}$. Then when we apply our functor H^U we are looking for a morphism

$$H^U(X) \longrightarrow H^U(Y)$$

that is a function

$$\mathcal{C}(U, X) \longrightarrow \mathcal{C}(U, Y)$$

that is, a function that sends morphisms $U \longrightarrow X$ to morphisms $U \longrightarrow Y$. But look: we started off with a morphism $X \xrightarrow{f} Y$ so we can just do

$$U \longrightarrow X \quad \mapsto \quad U \longrightarrow X \xrightarrow{f} Y$$

which on a particular element $s \in \mathcal{C}(U, X)$ gives

$$s \mapsto f \circ s$$

and that's why we call this function $f \circ \_$ – the "blank" sign says that's where we're supposed to feed our variable in (in this example s). This function is sometimes also written $\mathcal{C}(1, f)$.

Note that we had to insist $\mathcal{C}$ was locally small because we wanted each $\mathcal{C}(X, Y)$ to be a *set*, so that this really is a functor from $\mathcal{C}$ to **Set**.

Contravariant representable

Now that we've warmed up, let's do the contravariant one. Given $U \in \mathcal{C}$ we have a functor $H_U : \mathcal{C}^{\mathrm{op}} \longrightarrow \mathbf{Set}$ defined as follows:

- on objects $X \mapsto \mathcal{C}(X, U)$;

[8] Remember, locally small means that all the $\mathcal{C}(X, Y)$ are actually sets.

- on morphisms

$$\begin{array}{ccc} X \xrightarrow{f} Y & \mapsto & \mathcal{C}(Y,U) \xrightarrow{\_ \circ f} \mathcal{C}(X,U) \\ & & s \quad \mapsto \quad s \circ f \end{array}$$

Note that if we write out sources and targets, the action of the function $\_ \circ f$ is

$$Y \xrightarrow{s} U \quad \mapsto \quad X \xrightarrow{f} Y \xrightarrow{s} U$$

so if you're ever wondering which side your composition is supposed to be on, you can just write these arrows out and you can't go wrong.

Representable functors later

We will later define representable functors in general to be functors which are *naturally isomorphic* to the representables we've just defined.

3.9 Categories of categories

Categories and functors form a category. Or rather, small categories and functors form a large category. Large categories and functors form a super-large category. And so on, thus we avoid Russell's paradox.

However, we haven't yet defined identities and composition of functors.

- Given a category $\mathcal{C}$ there is an identity functor

$$\begin{array}{rcl} 1_{\mathcal{C}} : \mathcal{C} & \longrightarrow & \mathcal{C} \\ X & \mapsto & X \\ f & \mapsto & f \end{array}$$

- Given functors $\mathcal{C} \xrightarrow{F} \mathcal{D} \xrightarrow{G} \mathcal{E}$ we have a composite functor $\mathcal{C} \xrightarrow{GF} \mathcal{E}$ defined in the obvious way by composing the underlying functions.

Definition 3.4.
We write **Cat** for the (large) category of small categories and functors.
We write **CAT** for the (super-large) category of large categories and functors.

The category **Cat** is a rather nice category with a lot of interesting and useful structure. For example:

- **Cat** has terminal objects – any category with only one object and only one (identity) morphism. So often write this category as **1**.
- **Cat** has an initial object – the empty category.

- **Cat** has products: given categories $\mathcal{C}$ and $\mathcal{D}$, we can form the cartesian product $\mathcal{C} \times \mathcal{D}$, whose objects are pairs (c, d), and a morphism

$$(c, d) \longrightarrow (c', d')$$

is just a pair

$$\begin{array}{c} c \xrightarrow{f} c' \\ d \xrightarrow{g} d' \end{array}$$

Formally we have

  - $\mathrm{ob}(\mathcal{C} \times \mathcal{D}) = \mathrm{ob}\mathcal{C} \times \mathrm{ob}\mathcal{D}$
  - $\mathcal{C} \times \mathcal{D}\big((c, d), (c'd')\big) = \mathcal{C}(c, c') \times \mathcal{D}(d, d')$

3.10 Functors in topology

Category Theory largely grew from considering a formal approach to algebraic topology, and indeed modern algebraic topology is swarming with categories, functors, and many other categorical notions. Here are a few examples for those who know some topology.

- There is a functor $\Pi_1 : \mathbf{Top}_* \longrightarrow \mathbf{Gp}$ that sends a based space to its fundamental group. This functor has excellent properties, and this fact is a way of explaining why the study of spaces via their fundamental groups is so fruitful.

- There is likewise a functor $\Pi_n : \mathbf{Top}_* \longrightarrow \mathbf{Gp}$ for each $n \geq 1$ that sends a based space to its nth homotopy group. Of course, for $n \geq 2$ this functor actually lands in **Ab**.

- There is for each n a homology functor $H_n : \mathbf{Top} \longrightarrow \mathbf{Gp}$ which sends each space to its nth homology group. These can be fruitfully compiled into a single functor $H_\bullet$ from **Top** to the category of "graded groups"; a graded group is basically a sequence of groups $G_n, n \in \mathbb{N}$.

- For cohomology we get contravariant functors $H^n : \mathbf{Top}^{\mathrm{op}} \longrightarrow \mathbf{Gp}$ which can also be compiled into one functor $H^\bullet$.

- Taking the *singular complex* of a space gives us a functor from **Top** to **sSet**, the category of simplicial sets. Conversely, taking the *geometric realisation* of a space gives us a functor from **sSet** to **Top**. These two functors are intimately related, as we'll see later ***ref.

3.11 Properties of functors

Functors are particularly useful if they are "well-behaved". Intuitively this usually has something to do with how accurately they set up a correspondence between the structure in the source category with the structure in the target category. Of course, functors that are *isomorphisms* set up a very precise correspondence, this is much too unforgiving for most uses – we're interested in categories that correspond to each other in much less rigid ways. We can consider pairs of functors

$$\mathcal{C} \rightleftarrows \mathcal{D}$$

that aren't actually an isomorphism, but something a bit more subtle – we'll later define *equivalences* and *adjunctions*. These give us ways of studying structure in $\mathcal{C}$ via structure in $\mathcal{D}$, and vice versa. For now, we'll just look at some good properties that functors might have that get us going.

Properties that all functors have

Functors, by definition, preserve any property that is expressible as a commutative diagram; this is what functoriality is all about. For example, if f is an isomorphism then Ff is also an isomorphism. However, the converse is not true: it is possible for Ff to be an isomorphism even if f was not an isomorphism. However it is quite desirable for a functor to send *only* isomorphisms to isomorphisms; the next definition characterises some functors that certainly do this.

Full and faithful functors

Definition 3.5. A functor $F : \mathcal{C} \longrightarrow \mathcal{D}$ is called:

- *faithful* if the map $\mathcal{C}(X, Y) \longrightarrow \mathcal{D}(FX, FY)$ is injective for all X, Y,
- *full* if the map $\mathcal{C}(X, Y) \longrightarrow \mathcal{D}(FX, FY)$ is surjective for all X, Y,
- *full and faithful*[9] if the map $\mathcal{C}(X, Y) \longrightarrow \mathcal{D}(FX, FY)$ is bijective for all X, Y.

Note that these conditions are different from just saying that the functor is injective/surjective/bijective on morphisms. A functor could be non-injective on morphisms overall but still faithful; a functor could be surjective on morphisms but still *not* full.

Proposition 3.6. *Let $F : \mathcal{C} \longrightarrow \mathcal{D}$ be full and faithful. Then*

$$Ff \text{ is an isomorphism} \iff f \text{ is an isomorphism.}$$

[9] Some people call this "fully faithful" although the full and the faithful are really separate things here; John Baez has remarked that if one has an affair, it may or may not have anything to do with having had a large dinner.

Proof. Exercise. □

Note that we are not so interested in injectivity and surjectivity on objects. Philosophically, this would involve writing definitions that include equalities of objects, and one of the principles of Category Theory is that we should really be replacing those with isomorphisms. When we study equivalences of categories we'll see that we're much more interested in a notion of "surjectivity up to isomorphism"; interestingly "injectivity up to isomorphism" follows from fullness (see Exercises).

3.12 Subcategories

A subcategory is (hopefully) exactly what you think it should be:

Definition 3.7. A *subcategory* $\mathcal{D}$ of a category $\mathcal{C}$ consists of subcollections

- $\mathrm{ob}\mathcal{D} \subseteq \mathrm{ob}\mathcal{C}$, and
- $\mathrm{arr}\mathcal{D} \subseteq \mathrm{arr}\mathcal{C}$

with composition and identities inherited from $\mathcal{C}$.

Note that this ensures that the inclusion

$$\mathcal{D} \hookrightarrow \mathcal{C}$$

is in fact a functor.

Definition 3.8. We say that $\mathcal{D}$ is a *full* subcategory of $\mathcal{C}$ if for all objects $X, Y \in \mathcal{D}$

$$\mathcal{D}(X, Y) = \mathcal{C}(X, Y).$$

Intuitively this means we have "not necessarily all the objects, but definitely all the morphisms between them". This is equivalent to saying that the inclusion functor is full. Note that the inclusion functor is automatically faithful.

The reverse situation is to have "all the objects, not necessarily all the morphisms".

Definition 3.9. We say that $\mathcal{D}$ is a *lluf* subcategory of $\mathcal{C}$ if

$$\mathrm{ob}\mathcal{D} = \mathrm{ob}\mathcal{C}.$$

.

This is a joke. Or is it? Those who do use this term tend to apply a deadly serious straight face and the Welsh pronunciation of "ll".

Examples of subgroups

- **Ab** is a full subcategory of **Gp** – a homomorphism of abelian groups is just a homomorphism of them as groups.
- Similarly **Haus** is a full subcategory of **Top**.
- If G is a subgroup of H then G is a subcategory of H when we express them as one-object categories. G is only a full subcategory if G is actually the whole of H.

3.13 A non-example of a functor

There is no functor $Z : \mathbf{Grp} \longrightarrow \mathbf{Grp}$ taking a group to its centre. Recall that the centre of a group is the part that "commutes with everything":

$$Z(G) = \{g \in G \mid \forall\ h \in G\ gh = hg\}.$$

This is a subgroup of G because if g and g' both satisfy this defining property, then given any $h \in G$ we also have:

$$gg'h = ghg' = hgg'.$$

The fact that Z can't be expressed as a functor demonstrates something it is important to remember about Category Theory:

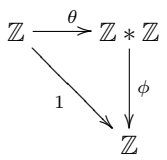
> **Not everything can be expressed in terms of category theory.**

Category Theory can't be expected to do everything, but it is interesting to come to an understanding of the things that are and aren't "categorical" concepts.

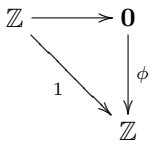
Explanation of the non-example

For those who are curious and known enough about groups, we'll now explain this non-example. To see that Z can't be expressed as a functor, we need to have a good example of a very non-abelian group – that is, a group with *no* elements that commute with everything else in the group, so the centre is the trivial group $\mathbf{0}$. One such example is $\mathbb{Z} * \mathbb{Z}$, the "free product" of $\mathbb{Z}$ with itself, otherwise known as the *free non-abelian group on two generators*. We can think of its elements as being generated by a and b, say, and so the elements are finite strings of a's, b's, and their inverses.

Now, there is a commutative diagram in groups:



where θ is defined[10] by sending 1 to a, and ϕ is defined by sending both a and b to 1. Now, if there were a functor sending every group to its centre, we could apply it to this commutative diagram. The centre of $\mathbb{Z}$ is $\mathbb{Z}$, since it is already abelian, so we'd get a diagram:



But this can't possibly be a commutative diagram in groups – $\mathbf{0}$ is both initial and terminal, so we know exactly what the horizontal and vertical homomorphisms have to be. The horizontal one sends everything to 0, and the vertical one has to preserve the identity, so it just sends 0 to 0. So the composite would send the whole of $\mathbb{Z}$ to 0, and the homomorphism that does that is far from being the identity homomorphism. This means that there cannot be a functor sending every group to its centre.

3.14 Exercises

1. Let $F : \mathcal{C} \longrightarrow \mathcal{D}$ be a functor. Show that if f is an ismorphism in $\mathcal{C}$ then Ff is an isomorphism in $\mathcal{D}$. Show that the converse is not true.

2. Let $F : \mathcal{C} \longrightarrow \mathcal{D}$ be a functor. Show that F preserves commutative diagrams, first making clear exactly what this means.

3. Let $F : \mathcal{C} \longrightarrow \mathcal{D}$ be full and faithful. Show that

$$Ff \text{ is an isomorphism} \iff f \text{ is an isomorphism.}$$

4. Let $F : \mathcal{C} \longrightarrow \mathcal{D}$ be a functor. Try to guess the definition of "injective on objects up to isomorphism" and "surjective on objects up to isomorphism" by replacing equalities by isomorphisms in the usual definitions of injective and surjective. Show that if F is full and faithful then F is "injective on objects up to isomorphism", but not necessarily "surjective on objects up to isomorphism".

5. What are the subcategories of a poset (regarded as a category)?

6. i) Show that there is a functor $O : \mathbf{Cat} \longrightarrow \mathbf{Set}$ taking a category to its set of objects. Is this functor faithful? Is it full?

   ii) Show that there is a functor $A : \mathbf{Cat} \longrightarrow \mathbf{Set}$ taking a category to its set of morphisms. Is this functor faithful? Is it full?

7. Let G be a group, regarded as a category with one object.

[10] Recall that to define a group homomorphism from $\mathbb{Z}$ to any other group, we only have to say what it does to 1 because everything else will then happen automatically.

i) What is a functor $G \longrightarrow \mathbf{Set}$?

ii) What is a functor $G \longrightarrow \mathbf{Vect}$?

8. Show that there is a functor $\mathbf{Cat} \longrightarrow \mathbf{Cat}$ sending every category to its opposite. *(This defines the functor on objects; you will need to define it on morphisms.)*