## Contents

Preface iv

1 Categories defined 1
  1.1 The very idea of a category ........................................ 1
  1.2 Examples ............................................................ 2
  1.3 Diagrams ............................................................. 7
  1.4 Duality ............................................................... 8

2 Kinds of arrows 10
  2.1 Monomorphisms, epimorphisms ..................................... 10
  2.2 Sections, monics and their duals ................................... 11
  2.3 Isomorphisms .......................................................... 13

3 Functors 15
  3.1 Functors defined ..................................................... 16
  3.2 Some first examples of functors .................................... 16
  3.3 Subcategories and inclusion functors .............................. 18
  3.4 What do functors preserve? ......................................... 19
  3.5 An example from algebraic topology ............................... 19

4 More about functors and categories 22
  4.1 Functors compose ................................................... 22
  4.2 Covariant vs contravariant functors ............................... 22
  4.3 Product categories, arrow categories, and associated functors .. 23
  4.4 Slice categories and associated functors .......................... 24
  4.5 Comma categories .................................................... 26

5 Natural transformations 28
  5.1 Motivation ........................................................... 28
  5.2 Natural transformations defined .................................... 31
  5.3 Examples of natural transformations ................................ 33
  5.4 Functor categories .................................................. 34
  5.5 Horizontal composition of natural transformations ................ 36
  5.6 Natural isomorphisms (again) ...................................... 37

6 Equivalence of categories 40
  6.1 Full, faithful, and other kinds of functors ....................... 40
6.2 The categories $\text{Set}^*$ and $\text{Set}_*$ ................................. 41
6.3 Isomorphic categories .................................................. 43
6.4 Equivalent categories .................................................. 45
6.5 Skeleton categories ..................................................... 48

7 Categories of categories: issues of size ................................... 50
7.1 Categories of categories ................................................ 50
7.2 ‘Small’ and ‘locally small’ categories ................................. 51
7.3 Classes, virtual and real ................................................. 53
7.4 Categories and classes: back to basics ............................... 57
7.5 Another definition of categories ...................................... 59

8 The Yoneda embedding ..................................................... 60
8.1 Hom-sets ................................................................. 61
8.2 Hom-functors ........................................................... 62
8.3 Generating natural transformations .................................. 64
8.4 The Restricted Yoneda Lemma ....................................... 65
8.5 A new pair of fully faithful functors ................................. 66
8.6 The Yoneda embedding defined ...................................... 67

9 An aside on Cayley’s Theorem ........................................... 69
9.1 Cayley’s Theorem ....................................................... 69
9.2 Yoneda meets Cayley .................................................. 70

10 The Yoneda Lemma .......................................................... 72
10.1 Towards the full Yoneda Lemma ..................................... 72
10.2 The generalizing move ................................................ 73
10.3 Making it all natural ................................................... 73
10.4 Putting everything together ......................................... 75

11 Representables ............................................................... 77
11.1 Presheaves, diagrams, representables .............................. 77
11.2 Is the forgetful functor $\text{Mon} \to \text{Set}$ representable? .......... 78
11.3 More examples of representables ................................... 79
11.4 Representations, or universal elements ............................ 81
11.5 Categories of elements ................................................. 81

12 First examples of limits .................................................... 84
12.1 Terminal and initial objects ........................................... 84
12.2 Elements and generalized elements ................................. 86
12.3 Pairing schemes ....................................................... 87
12.4 Binary products, categorially ...................................... 90
12.5 Results about binary products .................................... 92
12.6 Two-place functions and arrows from products ............... 95
12.7 Products generalized ................................................ 97
12.8 Coproducts ............................................................. 98
12.9 Equalizers .............................................................. 101
12.10 Co-equalizers .......................................................... 104

13 Limits and colimits defined ............................................... 106
13.1 Defining limits ......................................................... 106
13.2 Limits and diagrams-as-functors ................................... 109
13.3  Another characterization of limits .............................. 110
13.4  Colimits defined ................................................. 111
13.5  More examples: pullbacks and pushouts .......................... 112

14 The existence of limits ............................................. 116
14.1  Pullbacks, products and equalizers related ...................... 116
14.2  \textbf{Set} has all finite/small limits ............................ 118
14.3  The existence of limits, more generally ........................ 119
14.4  Dualizing again .................................................... 121

15 Functors and limits .................................................. 122
15.1  Preserving limits .................................................... 122
15.2  Reflecting limits .................................................... 125
15.3  Creating limits ..................................................... 126
15.4  Hom-functors preserve limits ..................................... 126
15.5  More on hom-functors and limits .................................. 128

16 Galois connections .................................................... 129
16.1  (Probably unnecessary) reminders about posets ............... 129
16.2  Galois connections defined ....................................... 130
16.3  Galois connections re-defined .................................... 132
16.4  Some basic results about Galois connections .................... 133
16.5  Fixed points and closures ......................................... 134
16.6  One way a Galois connection can arise ........................... 136

17 An aside on syntax and semantics ................................ 137
17.1  Structures and models ............................................. 137
17.2  The Galois connection between syntax and semantics ........ 138
17.3  Drawing some consequences ....................................... 138
17.4  Is this all trivial, then? .......................................... 140
17.5  Another implication ................................................. 141
17.6  A bug? ................................................................. 141

18 Adjoints introduced .................................................. 143
18.1  Adjoint functors: a first definition ............................... 143
18.2  Examples ............................................................ 144
18.3  The uniqueness of adjoints ....................................... 146
18.4  Examples, continued ............................................... 146
18.5  Naturality .......................................................... 148
18.6  A second definition of adjoints ................................... 148

Bibliography ...................................................................... 153
These ‘Notes on Category Theory’ began life as notes written for myself to help fix ideas. They were initially prompted by following a course of 24 lectures given for Part III of the Maths Tripos by Rory Lucyshyn-Wright in the Michaelmas Term, 2014, and by some associated reading around on the material he covered. The notes rather quickly departed from the details of those lectures in both style and content, though some of the overall structure is still there, in particular in the order in which topics are currently tackled, and most of the topics in at least the first 20 lectures will be touched on. (There remain about four more chapters to come here.)

What’s distinctive, for good or ill, about these Notes – and there are a number of sets of notes on category theory available online (see here) – is that they are written by someone who frankly is still learning as he goes along rather than by a fully-formed expert. That’s why I go slowly over ideas that initially gave me pause, and have generally tried to be as clear as possible in an introductory way which I hope might help others in the same boat when starting to get to grips with categories. Hence the expansive style. Despite the length, though, the level we reach here probably falls somewhat short of what is required for Part III maths students: but my concern is to really nail down some of the basics.

Most of the proofs of theorems are easy, and often are very easy indeed. Even when a result like the Yoneda Lemma takes a bit more work, I try to break things down using intermediate lemmas so each stage is straightforward enough to prove. So you could very reasonably think of almost every statement of a theorem as implicitly presenting you (as it did me!) with an exercise which you should pause to do, and then the ensuing proof which I spell out is the answer (or at least, an answer) to the exercise.

I would have called the notes ‘Basic Category Theory’, because the level indeed remains mathematically pretty basic. But that title has been snaffled by Tom Leinster for his warmly recommended book, which covers similar ground – though he does things faster, in a different order, and goes further in places, in a somewhat more hard-core style.

Many thanks are due to RL-W for the lectures which have (re)sparked my interest in this material. The errors of one kind or another are of course all mine. Corrections, please, to ps218 at cam dot ac dot uk. And for the latest version of these notes, see the Category Theory page at the Logic Matters website.

Peter Smith
Here’s a fundamental insight: we can think of a family of mathematical structures equipped with some structure-preserving maps/functions/morphisms between them as itself forming a non-trivial mathematical structure.

We can then investigate such structures-of-structures, and go on to think about structure-preserving maps – or as they say, functors – between them. Then we can talk in turn about maps between such functors.

These layers of increasing abstraction are the topic of category theory. So if modern mathematics already abstracts (moving, for example, from concrete geometries to abstract metric and topological spaces), category theory abstracts again, and then again. What do we gain by going up another level of abstraction or two? To quote Tom Leinster,

Category theory takes a bird’s eye view of mathematics. From high in the sky, details become invisible, but we can spot patterns that were impossible to detect from ground level. (Leinster, 2014, p. 1)

So that’s one justification for ascending to the dizzying heights: we hope that we’ll find some large-scale but still illuminating patterns in mathematics which aren’t otherwise discernible. If you want a glimpse ahead, then for the very briskest of hints about what some of these patterns might turn out to be, see e.g. Adámek et al. (2009, §1).

1.1 The very idea of a category

A category \( \mathcal{C} \) consists in certain data (in the sense of ‘givens’, rather than of ‘information’!), governed by a couple of axioms.

**Definition 1.** The data for a category \( \mathcal{C} \) comes in two distinct, non-overlapping, sorts:

1. **Objects** (which we will typically notate by ‘\( A \)’, ‘\( B \)’, ‘\( C \)’, \ldots).
2. **Arrows** (which we typically notate by ‘\( f \)’, ‘\( g \)’, ‘\( h \)’, \ldots).

Further,

1. For each arrow \( f \), there are unique associated objects \( \text{src}(f) \) and \( \text{tar}(f) \), respectively the **source** and **target** of \( F \). We write ‘\( f : A \to B \)’ or ‘\( A \xrightarrow{f} B \)’ to notate that \( F \) is an arrow with \( \text{src}(f) = A \) and \( \text{tar}(f) = B \).
2. For any two arrows \( f : A \to B \), \( g : B \to C \), where \( \text{src}(g) = \text{tar}(f) \), there exists an arrow \( g \circ f : A \to C \), ‘\( g \) following \( f \)’, which we call the **composite** of \( f \) with \( g \).
Given any object $A$, there is an arrow $1_A : A \to A$ called the identity arrow on $A$.

The axioms then require

**Identity.** The identity arrows do behave like identities! So for any $f : A \to B$ we have $f \circ 1_A = f = 1_B \circ f$.

**Associativity.** For any $f : A \to B$, $g : B \to C$, $h : C \to D$, we have $h \circ (g \circ f) = (h \circ g) \circ f$.

These axioms suffice to ensure our first mini-result:

**Theorem 1.** Identity arrows on a given object are unique; and the identity arrows on distinct objects are distinct.

**Proof.** Suppose $A$ has identity arrows $1_A$ and $1'_A$. Then $1_A = 1_A \circ 1'_A = 1'_A$.

For the second part, we simply note that $A \neq B$ entails $\text{src}(1_A) \neq \text{src}(1_B)$ which entails $1_A \neq 1_B$. \qed

(As you can see, I will cheerfully call the most trivial of lemmas, run-of-the-mill propositions, interesting corollaries, and the weightiest of results all ‘theorems’ without distinction.)

Four quick remarks on our terminology and notation:

(a) Borrowing familiar functional notation ‘$f : A \to B$’ for arrows in categories is very natural given that arrows in many categories indeed are functions. But as we’ll soon see, not all arrows are functions. Which means that not all arrows are morphisms either, in the usual sense of that term. Which is why I prefer the colourless ‘arrow’ to the about equally common term ‘morphism’ for the second sort of data in a category.

(b) In keeping with the functional paradigm, the source and target of an arrow are often called, respectively, the ‘domain’ and ‘codomain’ of the arrow. But again that usage has the potential to mislead, which is why I prefer our terminology.

(c) It should also be said that so-called objects in categories needn’t be objects either, in the usual logician’s sense of that term which contrasts objects with properties, relations or functions. In some categories, for example, the objects are in fact functions.

(d) Just occasionally, to reduce clutter, we may write simply ‘$gf$’ rather than $g \circ f$. (Some from a compsci background notate composition of $f$ with $g$ the other way about and write ‘$f;g$’, but we’ll stick to our much more usual convention.)

More substantively, we can also remark that since every object in a category is associated with one and only one identity arrow, and we can pick out identity arrows by the way they interact with other arrows, we can in principle define categories initially just in terms of arrows. For an account of how to do this, see Adámek et al. (2009, pp. 41–43). But I find this technical trickery unhelpful. (The central idea of category theory is that we should probe objects by considering the morphisms between them, not that we should write objects out of the story!)

### 1.2 Examples

In the preamble, we said that we are going to be interested in particular families of structures which have maps between them. Structures are nowadays very commonly thought of as being sets equipped with functions and/or relations on them. So we are going
to expect some paradigm examples of categories to have as objects sets-equipped-with-some-functions/relations, and then the arrows between such objects will then typically be suitable functions between the carrier sets which in a good sense ‘preserve structure’ (or perhaps we should say ‘respect structure’, for ‘preservation’ might sound like a matter of producing a full copy, which is more than we usually require).

Let’s start with what we can think of as the zero case.

(1) **Set** is a category with

**Objects:** all the sets (just plain sets, equipped with no extra structure).

**Arrows:** given sets \( X, Y \), any (total) set-function \( f : X \to Y \) is an arrow.

There’s an identity function on any set; set-functions \( f : A \to B, g : B \to C \) (where the domain of \( g \) is the codomain of \( f \)) compose; and the axioms for being a category are evidently satisfied.

Four comments on this initial case! (i) There is actually an issue about just what category we have in mind here in talking about **Set**. But whatever your favoured conception of the universe of sets – and remember that not everyone is a fan of ZFC, for example – it will constitute a category; so for the moment you can just interpret talk of sets in your favoured way. For some more introductory remarks, see here. And do eventually take a look at Leinster 2014, Ch. 3, ‘Interlude on sets’.

(ii) Note that arrows in **Set** come with determinate targets/codomains. But a common way of modelling functions set-theoretically is simply to identify a function with a certain set of ordered pairs (such that if \( \langle x, y \rangle \) and \( \langle x, y' \rangle \) are in the set, \( y = y' \)). That kind of definition is lop-sided in that it fixes a domain (the set of first elements in the pairs) but doesn’t determine the function’s codomain (the same set of pairs could model both \( f : A \to B \) and \( f' : A \to C \) where \( B \subseteq C \)). We then have two options in talking about arrows in **Set**. Either keep the usual lopsided definition of a set-function \( f \), but identify an arrow \( f : A \to B \) in **Set** with a triple \( \langle A, f, B \rangle \). Or revise the way we model the notion of a function in set theory, using such a triple there, and then identify arrows in the category with these revised modelings. There’s plainly nothing to choose. And henceforth, we’ll forget this wrinkle.

(iii) Perhaps we should remind ourselves why there is an identity arrow for \( \emptyset \) in **Set**. Vacuously, for any \( Y \), there is exactly one set-function \( f : \emptyset \to Y \) – i.e. one appropriate set of ordered pairs – namely the empty set, and hence in particular there is a function \( 1_{\emptyset} : \emptyset \to \emptyset \). Note that in **Set**, the empty set is in fact the only object such that there is exactly one arrow from it to any other object. This gives us a nice first example of how we can characterize a significant object in a category not by its internal constitution, so to speak, but by what arrows it has to and from other objects.

(iv) The function \( id_A \) defined on a set \( A \) by \( id_A(x) = x \) will evidently serve in the category **Set** as the (unique) identity arrow \( 1_A \). We can’t say that in pure category-speak. But we can do something that comes to the same. Glancing ahead to ideas which we’ll return to, note first that we can categorically define singletons in **Set** by relying on the observation that there is exactly one arrow from any object to a singleton. So fix on a singleton, call it simply 1. Then consider the possible arrows (i.e. set-functions) \( \bar{x} : 1 \to A \). There is exactly one such arrow for every \( x \in A \). So we can think of talk of arrows \( \bar{x} : 1 \to A \) as our category-speak surrogate for talking about elements \( x \) of \( A \). Then instead of saying \( id_A(x) = x \) for all members \( x \) of \( A \), we can say that for any \( \bar{x} : 1 \to A, 1_A \circ \bar{x} = \bar{x} \). (More on this sort of thing in due course: but it gives us another glimpse ahead of how we might trade in talk of sets-and-their-elements for categorial talk of sets-and-arrows-between-them.)
(2) To continue with examples of categories, there is also a category $\text{FinSet}$ of finite sets (meaning the hereditarily finite sets, i.e. sets with at most finite numbers of members, these members in turn having at most finite numbers of members, which in turn . . . ) and the functions between them.

(3) $\text{Set}$, is a category (of ‘pointed sets’) with

- **objects**: all the sets [apart from the empty set!], each set $X$ equipped with a zero-place function that picks out a distinguished object $\star_X$ in the set,
- **arrows**: for $X, Y$ among the sets, any (total) function $f: X \to Y$ which maps the distinguished object $\star_X$ to the distinguished object $\star_Y$ is an arrow.

Picking out a distinguished object is the least structure we can put on a set! We now consider cases with rather more structure:

(4) Recall the definition of a monoid $(M, \cdot, 1_M)$. This a set $M$ equipped with a two-place function mapping elements to elements and with a designated element, governed by the axioms (i) the function is associative, i.e. for all elements $a, b, c \in M$, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$, and (ii) the designated element $1_M$ acts as a unit, i.e. is such that for any $a \in M$, $1_M \cdot a = a = a \cdot 1_M$. Then $\text{Mon}$ is a category with

- **objects**: all the monoids,
- **arrows**: for $(M, \cdot, 1_M), (N, \times, 1_N)$ among the monoids, any monoid homomorphism $f: M \to N$ is an arrow. Here, $f: M \to N$ is a monoid homomorphism when it preserves monoidal structure, i.e. for any $x, y \in M$, $f(x \cdot y) = f(x) \times f(y)$, and $f$ preserves identity elements, i.e. $f(1_M) = 1_N$.

In this case, the identity arrows are the identity functions on the carrier sets for the monoids, and composition of arrows is composition of homomorphisms.

(5) $\text{Mon}$ is just the first of a family of similar algebraic cases, where the objects are sets equipped with some functions and the arrows are functions preserving that structure. We also have:

- (a) $\text{Grp}$, the category of groups, with
  - **objects**: the class of groups,
  - **arrows**: group homomorphisms (functions preserving group structure)

- (b) $\text{Ab}$, the category of abelian groups, with
  - **objects**: the class of abelian groups,
  - **arrows**: group homomorphisms.

- (c) $\text{Rng}$ the category of rings, with
  - **objects**: the class of rings,
  - **arrows**: ring homomorphisms.

And so it goes!

(6) The monoids as objects together with all the monoid homomorphisms form a (large!) category $\text{Mon}$. But any one particular monoid taken just by itself can be thought of as corresponding to a (perhaps very small!) category.

Thus take the monoid $(M, \cdot, 1_M)$. And now define $\mathcal{M}$ to be the corresponding category with data

- (i) the sole object of $\mathcal{M}$: just some object – choose whatever you like, and dub it ‘$\star$’.
- (ii) arrows of $\mathcal{M}$: the elements of the monoid $M$. 

iii. the identity arrow 1 of $\mathcal{M}$ is the identity element $1_M$ of the monoid $M$.

iv. the composition $m \circ n : \star \to \star$ of two arrows $m : \star \to \star$ and $n : \star \to \star$ (those arrows being just the elements $m, n \in M$), is $m \cdot n$.

It is trivial that the axioms for being a category are satisfied. So we can think of a monoid as a one-object category. (Conversely, then, we can think of categories as, in a sense, generalized monoids.)

Note in this case, unless the elements of the original monoid $M$ are themselves functions, the arrows of the associated category $\mathcal{M}$ are not themselves functions or morphisms in any natural sense.

(7) **Ord** is a category, with

- objects: the pre-ordered sets. Recall, the set $S$ is pre-ordered iff equipped with an order $\preceq$ where for all $x, y \in S$, $x \preceq x$, and $x \preceq y \land y \preceq z \to x \preceq z$. We represent the resulting pre-ordered set $(S, \preceq)$.
- arrows: monotone maps – i.e. maps $f : S \to T$, from the carrier set of $(S, \preceq)$ to the carrier set of $(T, \sqsubseteq)$, such that if $x \preceq y$ then $f(x) \sqsubseteq f(y)$.

(8) Note too that any pre-ordered set can also itself be regarded as a category (this time, a category with at most one arrow between objects). For corresponding to the pre-ordered set $(S, \preceq)$, we will have a category $\mathcal{S}$ with

- i. The objects of $\mathcal{S}$ are just the members of $S$.
- ii. An arrow from $\mathcal{S}$ from source $C$ to target $D$ is just an ordered pair of objects $(C, D)$ such that $C \preceq D$.
- iii. $1_C = (C, C)$.
- iv. Composition is defined by setting $(D, E) \circ (C, D) = (C, E)$.

It is easily checked that this satisfies the identity and associatively axioms. Conversely, any category $\mathcal{S}$ whose objects form a set $S$ and where there is at most one arrow between objects can be regarded as a pre-ordered set $(S, \preceq)$ where for $C, D \in S$, $C \preceq D$ just in case there is an arrow from $C$ to $D$ of $\mathcal{S}$.

We can thus call a category with at most one arrow between objects a **pre-order category**.

An aside on a point of notation. I’m planning to revert to old-school angle brackets, as in ‘$(C, D)$’, when we are indeed definitely talking about an ordered-pair treated as a single object. By contrast, we can in many contexts take parentheses as simply helpful punctuation, not as a constructor for a new object: for example, when talking informally of the pre-ordered set $(S, \preceq)$ as we just did, we are talking about the set $S$ and about the ordering $\preceq$ defined over $S$, and not – or at least, not yet – about some further pair-object.

The rule for the moment, then, will be that angle brackets are always used to denote ordered-pairs-as-objects. While the interpretation of expressions with parentheses may be up for grabs: case by case, read as noncommittally as possible. But we’ll have to see how this notational policy works out!

(9) A closely related case to Ex. (7): **Pos** is a category with

- objects: the posets – $S$ is a poset equipped with a partial order $\preceq$, i.e. with a pre-order which satisfies the additional constraint that for $x, y \in S$, $x \preceq y \land y \preceq x \to x = y$.
- arrows: monotone maps.
And exactly as each pre-ordered set can be regarded as category, so each individual poset can be regarded as a category, a poset category.

(10) **Top** is a category with

- objects: all the topological spaces,
- arrows: the continuous maps.

(11) **Met** is a category with

- objects: metric spaces, any set of points $S$ equipped with a real metric $d$,
- arrows: the non-expansive maps, where – in an obvious shorthand notation – $f : (S,d) \to (T,e)$ is non-expansive iff $d(x,y) \geq e(f(x),f(y))$.

For those who know about such beasts, we could also mention categories of sheaves, of schemes, of simplicial sets, and so on, in each case with the relevant objects equipped with predictable structure-preserving maps as arrows. But we won’t pause over those: instead let’s finish with a few more particularly elementary examples . . .

(12) We should mention at this early stage the **discrete category** on the set $M$, whose objects are just the members of the set, and which has as few arrows as possible, i.e. just the identity arrow for each object in $M$.

(13) The smallest discrete category is **1** which has exactly one object and one arrow (the identity arrow on that object). Let’s picture it in all its glory!

![Discrete Category](image.png)

But should we talk about the category **1**? Won’t a different choice of object make for different one-object categories?

Yes and no! We can have, in our mathematical universe, different cases of single objects equipped with an identity arrow – but they will be indiscernible from within category theory. We’ll need to say more on this sort of point in due course. But for the moment, compare a probably familiar sort of case from elsewhere in mathematics: we talk, for example, of the Klein four-group. Yes, there will be many concrete groups which have the right structure to be such a group. But they are group-theoretically indiscernible.

(14) And having mentioned **1** here’s another very small category, this time with two objects, the necessary identity arrows, and one further arrow between them. We can picture it like this:

![More Small Category](image.png)

Call this category **2**. (We can think of the von Neumann ordinal 2, i.e. the set $\{\emptyset, \{\emptyset\}\}$, as giving rise to this category when it considered as a poset with the subset relation as the ordering relation. And other von Neumann ordinals, finite and infinite, give rise to other poset categories.)

There’s no shortage of categories, then. We certainly have enough examples to make a start!
1.3 Diagrams

We can graphically represent categories – and in particular, represent facts about the equality of arrows – in a very natural way, using so-called commutative diagrams. We’ve just seen couple of mini examples. We’ll be using diagrams a great deal: so we’d better say something about them straight away.

Talk of diagrams is in fact used by category theorists in three different (but very closely related) ways. For the moment, we’ll mention two. First, we have the primary sense of diagrams-as-pictures, or representational diagrams, as we can call them.

**Definition 2.** A \((\text{representational})\) diagram has nodes representing objects from a given category \(\mathcal{C}\), and drawn arrows between nodes representing arrows of \(\mathcal{C}\). Nodes and drawn arrows are usually labelled.

Two nodes in a diagram can be joined by zero, one or more drawn arrows. A drawn arrow labelled ‘\(f\)’ from the node labeled ‘\(A\)’ to the node labeled ‘\(B\)’ of course represents the arrow \(f: A \rightarrow B\) of \(\mathcal{C}\).

We may also sometimes include drawn arrows looping from a node to itself, as in our diagrams above, which of course represent the identity arrow on the corresponding object.

**Definition 3.** A diagram in a category \(\mathcal{C}\) is what is represented by a representational diagram – i.e. some \(\mathcal{C}\)-objects and \(\mathcal{C}\)-arrows between some of them.

I’m being philosophically pernickety in distinguishing the two ideas here, the diagram-as-picture, and the diagram-as-what-is-pictured. Introductions to category theory tend to move between the two ideas without comment. And having made the distinction, we too needn’t fuss except when it matters, and will let context determine a sensible reading of claims about diagrams.

Now, within a diagram, we may be able to follow a directed path through more than two nodes, walking along the connecting drawn arrows as we go. So a path in a category diagram from nodes \(A\) to \(E\) (for example) might look like this

\[
A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D \xrightarrow{j} E
\]

And then we’ll call the composite arrow \(j \circ h \circ g \circ f\) the \textit{composite along the path}. (We know that, because of the associativity of composition, we needn’t fuss about bracketing here. And henceforth we insert or omit brackets just depending on whatever promotes local clarity.)

Then:

**Definition 4.** A category diagram \textit{commutes} if (i) for any two directed paths along edges in the diagram from a node \(X\) to node \(Y\), the composite arrow along the first path is equal to the composite arrow along the second path, and (ii) for any closed path that loops around from a node \(X\) to itself, the composite arrow along that path is equal to the identity arrow on the object at that node.

The second clause can be subsumed as a special case of the first if we suppose that diagrams are assumed to come with loops on each object representing identity arrows drawn in invisible ink!

Hence, for example, the associativity law can be represented by saying that the following diagram commutes:
So, in particular, the two outer paths from $A$ to $D$ give equal composites.

You can in fact develop the use of this sort of diagram into a formal language for making statements in category theory. For us, however, they will just be heuristic props – but quite extraordinarily helpful ones.

1.4 Duality

We have already noted a good initial range of examples of categories. And now here’s a cheap way of getting new categories from old: reverse all the arrows. More carefully:

**Definition 5.** Given a category $\mathcal{C}$, then $\mathcal{C}^{op}$ is the category with the data

1. The objects of $\mathcal{C}^{op}$ are just the objects of $\mathcal{C}$ again.
2. If $f$ is an arrow of $\mathcal{C}$ with source $A$ and target $B$, then $f$ is an arrow of $\mathcal{C}^{op}$ but with source $B$ and target $A$.
3. Identity arrows remain the same, i.e. $1^{op}_A = 1_A$.
4. Composition-in-$\mathcal{C}^{op}$ is defined terms of composition-in-$\mathcal{C}$ by putting $f \circ^{op} g = g \circ f$.

It is trivial to check that this definition is in good order and that $\mathcal{C}^{op}$ is indeed a category. And it is trivial to check that $((\mathcal{C}^{op})^{op})$ is $\mathcal{C}$. So every category is the opposite of some category!

Do be careful here. Take for example $\text{Set}^{op}$. An arrow $f: A \to B$ in $\text{Set}^{op}$ is the same thing as an arrow $f: B \to A$ in $\text{Set}$, which is of course a set-function from $B$ to $A$. This means that $f$ in $\text{Set}^{op}$ typically won’t be a function from its source to its target – it’s an arrow in that direction but actually a function in the other! (So this is one of those cases where talking of ‘domains’ and ‘codomains’ instead of ‘sources’ and ‘targets’ could initially encourage confusion.)

Let’s take $\mathcal{L}$ to be the elementary pure language of categories. This will be a two-sorted first-order language with identity, with one sort of variable for objects, $A, B, C, \ldots$, and another sort for arrows $f, g, h, \ldots$. It has built-in functions $\text{src}$ and $\text{tar}$, a built-in relation ‘…is the identity arrow for …’, and a composition function which will take two composable arrows to another arrow.

**Definition 6.** Suppose $\varphi$ is a wff of $\mathcal{L}$. Then its dual $\varphi^{op}$ is the result of swapping (i) ‘src’ and ‘tar’ and swapping (ii) ‘$f \circ g$’ for ‘$g \circ f$’, etc.

Now, the claim that $\mathcal{C}^{op}$ is a category just reflects the fact that the duals of the axioms for a category are also axioms. And that observation gives us the following **duality principle**:

**Theorem 2.** Suppose $\varphi$ is an $\mathcal{L}$-sentence with no free variables – so is a general claim about objects/functions in an arbitrary category. Then if the axioms of category theory entail $\varphi$, they also entail the dual claim $\varphi^{op}$.

**Proof.** If there’s a first-order proof of $\varphi$ from the axioms of category theory, then by taking the duals of every wff in the proof we’ll get a proof of $\varphi^{op}$ from the the duals of the axioms of category theory. But those duals of axioms are themselves axioms, so we have a proof of $\varphi^{op}$ from the axioms of category theory.

\qed
Putting the point semantically, if \( \varphi \) always holds, i.e. holds in every category \( \mathcal{C} \), then \( \varphi^{op} \) will hold in every \( \mathcal{C}^{op} \) – but the \( \mathcal{C}^{op} \)’s just comprise every category again, so \( \varphi^{op} \) too holds in every category.

Which is a very simple but also very labour-saving result, as we’ll see time and time again, starting in the next chapter.
This chapter characterizes a number of kinds of arrows in categories in terms of how they interact with other arrows. This will give us some elementary but characteristic examples of arrow-theoretic (re)definitions of familiar notions.

2.1 Monomorphisms, epimorphisms

A common-or-garden function $f$ is injective/one-to-one just if it sends different arguments to different values, i.e. for all relevant $x, y$, $f(x) = f(y)$ implies $x = y$.

How could we express this in category-speak about arrows, e.g. in $\text{Set}$? Well, we noted that we can think of elements $x$ of $f$’s domain $A$ as arrows $\vec{x} = \vec{x} : 1 \to A$ (where $1$ is some singleton), and then injectiveness comes to this: $f \circ \vec{x} = f \circ \vec{y}$ implies $\vec{x} = \vec{y}$. So if a function is ‘left-cancellable’ in $\text{Set}$ – in the sense that $f \circ g = f \circ h$ implies $g = h$ – then it is an injection.

Conversely, of course, if $f$ is injective then in particular for all $x$, $f(g(x)) = f(h(x))$ implies $g(x) = h(x)$ – which is to say that if $f \circ g = f \circ h$ then $g = h$, i.e. $f$ is left-cancellable.

So that motivates introducing a definition of the following shape (if not the new-fangled terminology):

**Definition 7.** An arrow $f : C \to D$ in the category $\mathcal{C}$ is a monomorphism (is monic) if and only if it is left-cancellable, i.e. for every pair of maps $g : B \to C$ and $h : B \to C$, if $f \circ g = f \circ h$ then $g = h$.

And we have just shown that

**Theorem 3.** The monomorphisms in $\text{Set}$ are exactly the injective functions.

The same applies in many, but not all, other categories where arrows are functions. For example, we have:

**Theorem 4.** The monomorphisms in $\text{Grp}$ are exactly the injective group homomorphisms.

*Proof.* We can use the same proof idea again to show that the injective group homomorphisms are monomorphisms in $\text{Grp}$.

For the other direction, suppose $f : C \to D$ is a group homomorphism between $(C, \cdot)$ and $(D, \star)$ and not an injection, so we must have $f(c) = f(c')$ for some $c, c' \in C$ where $c \neq c'$.
Note then that \( f(c^{-1} \cdot c') = f(c^{-1}) \ast f(c') = f(c^{-1}) \ast f(c) = f(c^{-1} \cdot c) = f(e_C) = e_D. \) So \( c^{-1} \cdot c' \) is an element in \( K \), the kernel of \( f \) (the set of elements that \( f \) sends to the unit \( e_D \) of \((D, \ast))\), where \( e' = c^{-1} \cdot c' \neq e_C. \)

Now define \( g: K \to C \) to be the inclusion map, while \( h: K \to C \) sends everything to \( e_C \). Since \( K \) has more than one element, \( g \neq h \). But obviously, \( f \circ g = f \circ h \) (both send everything in \( K \) to \( e_D \)). So \( f \) isn’t left-cancellable.

Hence, contrapositing, if \( f \) is monic in \( \text{Grp} \) it is injective.

Next, here is a companion definition:

**Definition 8.** An arrow \( f: C \to D \) in the category \( \mathcal{C} \) is an **epimorphism** (is **epic**) if and only if it is right-cancellable, i.e. for every pair of maps \( g, \) \( h: D \to E \) where \( g \neq h \). Then for some \( d \in D, g(d) \neq h(d). \) But by surjectivity, \( d = f(c) \) for some \( c \in C. \) So \( g(f(c)) \neq h(f(c)) \), whence \( g \circ f \neq h \circ f \). So the surjectivity of \( f \) in \( \text{Set} \) implies that if \( g \circ f = h \circ f \), then \( g = h \), i.e. \( f \) is epic.

Conversely, suppose \( f: C \to D \) is not surjective, so \( f[C] \neq D \). Consider two functions \( g, h: D \to E \) which agree on \( f[C] \subset D \) but disagree on the rest of \( D \). Then \( g \neq h \), even though by hypothesis \( g \circ f \) and \( h \circ f \) will agree everywhere on \( C \), so \( f \) is not epic. Contrapositing, if \( f \) is epic in \( \text{Set} \), it is surjective.

A similar result holds in many other categories, but we’ll see in a moment a case where we have an epic function which is not surjective.

As the very gentlest of exercises, let’s add for the record a mini-theorem:

**Theorem 5.** The epimorphisms in \( \text{Set} \) are exactly the surjective functions.

**Proof.** Suppose \( f: C \to D \) is surjective. And consider two functions \( g, h: D \to E \) where \( g \neq h \). Then for some \( d \in D, g(d) \neq h(d). \) But by surjectivity, \( d = f(c) \) for some \( c \in C. \) So \( g(f(c)) \neq h(f(c)) \), whence \( g \circ f \neq h \circ f \). So the surjectivity of \( f \) in \( \text{Set} \) implies that if \( g \circ f = h \circ f \), then \( g = h \), i.e. \( f \) is epic.

Conversely, suppose \( f: C \to D \) is not surjective, so \( f[C] \neq D \). Consider two functions \( g, h: D \to E \) which agree on \( f[C] \subset D \) but disagree on the rest of \( D \). Then \( g \neq h \), even though by hypothesis \( g \circ f \) and \( h \circ f \) will agree everywhere on \( C \), so \( f \) is not epic. Contrapositing, if \( f \) is epic in \( \text{Set} \), it is surjective.

A similar result holds in many other categories, but we’ll see in a moment a case where we have an epic function which is not surjective.

As the very gentlest of exercises, let’s add for the record a mini-theorem:

**Theorem 6.** (1) Identity arrows are always monic. Dually, they are always epic too.

(2) If \( f, g \) are monic, so is \( f \circ g \). If \( f, g \) are epic, so is \( f \circ g \).

(3) If \( f \circ g \) is monic, so is \( g \). If \( f \circ g \) is epic, so is \( f \).

**Proof.** (1) is trivial. For (2) suppose \( (f \circ g) \circ j = (f \circ g) \circ k \), then \( f \circ (g \circ j) = f \circ (g \circ k) \). Whence, assuming \( f \) is monic, \( g \circ j = g \circ k \). Whence, assuming \( g \) is monic, \( j = k \). So \( (f \circ g) \) is monic. The proof for epics is the dual.

For (3) assume \( f \circ g \) is monic. Temporarily suppose \( g \circ j = g \circ k \). Then \( f \circ (g \circ j) = f \circ (g \circ k) \), and hence \( (f \circ g) \circ j = (f \circ g) \circ k \), so \( j = k \). Therefore if \( g \circ j = g \circ k \) then \( j = k \); i.e. \( g \) is monic. Dually again for epics.

### 2.2 Sections, monics and their duals

We define some more types of arrow:

**Definition 9.** Given an arrow \( f: C \to D \) in the category \( \mathcal{C}, \)

(1) \( g: D \to C \) is a right-inverse or **section** of \( f \) (in \( \mathcal{C}, \) of course) iff \( f \circ g = 1_D \).

(2) \( g: D \to C \) is a left-inverse or **retraction** of \( f \) iff \( g \circ f = 1_C \).
(3) \( g : D \to C \) is an inverse of \( f \) iff it is both a section and a retraction of \( f \).

Note that \( g \circ f = 1_C \) in \( \mathcal{C} \) iff \( f \circ \text{op} \ g = 1_C \) in \( \mathcal{C}^{\text{op}} \). So a retraction in \( \mathcal{C} \) is a section in \( \mathcal{C}^{\text{op}} \). And vice versa. The ideas of a section and retraction are therefore dual to each other, and the idea of an inverse is dual to itself.

Another remark: If \( f \) has a section \( g \), then it is itself a retraction (of \( g \), of course!). Dually, if \( f \) has a retraction, then it is a section.

It is obvious that an arrow \( f \) need not have a retraction: just consider, for example, those arrows in \( \text{Set} \) which are many-one functions. An arrow \( f \) can also have many retractions: for a toy example in \( \text{Set} \) again, consider \( f : \{0, 1\} \to \{0, 1, 2\} \) where \( f(n) = n \).

Then the map \( g : \{0, 1, 2\} \to \{0, 1\} \) is a retraction so long as \( g(0) = 0 \) and \( g(1) = 1 \), which leaves us two choices for \( g(2) \), and hence we have two retractions. By the duality principle, an arrow can also have zero or many sections. However,

**Theorem 7.** If an arrow has both a section and a retraction, then these are the same and are the arrow’s unique inverse.

**Proof.** Suppose \( f : C \to D \) has section \( s \) and retraction \( r \). Then \( s = 1_C \circ s = (r \circ f) \circ s = r \circ (f \circ s) = r \circ 1_D = r \). Hence \( s \), i.e. \( r \), is an inverse.

Suppose now that \( f \) has inverses \( s \) and \( r \). By definition \( s \) will be a section and \( r \) a retraction, so as before \( s = r \). So in fact inverses are unique. \( \Box \)

Now, how does talk of an arrow as a section/retraction hook up to talk of an arrow as monic/epic?

**Theorem 8.**

1. In general, not every monomorphism is a section; and dually, not every epimorphism is a retraction.
2. But every section is monic, and every retraction is epic.

**Proof.** (1) can be shown by a toy example. Take the category \( 2 \) which we met back in §1.2, Ex. (14) – i.e. take as a category the two-element pre-ordered set which has just one non-identity arrow. The latter arrow is trivially monic and epic, but lacks either a left or a right inverse.

For (2), suppose \( f \) is a section for \( e \), which means that \( e \circ f = 1 \) [now suppressing unnecessary labellings of domains and codomains]. Now suppose \( f \circ g = f \circ h \). Then \( e \circ f \circ g = e \circ f \circ h \), and hence \( 1 \circ g = 1 \circ h \), i.e. \( g = h \), so indeed \( f \) is monic. Similarly for the dual. \( \Box \)

So monics need not in general be sections nor epics retractions: but how do things pan out in the particular case of the category \( \text{Set} \)? Here’s the answer:

**Theorem 9.** In \( \text{Set} \), every monomorphism is a section apart from arrows of the form \( \emptyset \to D \), while the claim that every epimorphism is a retraction is (a version of) the Axiom of Choice.

**Proof.** Suppose \( f : C \to D \) in \( \text{Set} \) is monic. It is therefore one-to-one between \( C \) and \( f[C] \), so consider a function \( g : D \to C \) that reverses \( f \) on \( f[C] \) and somehow or other maps \( D - f[C] \) into \( C \). Such a \( g \) is always possible to find in \( \text{Set} \) unless \( C \) is the empty set. So \( g \circ f = 1_C \), so \( f \) is a section.

Now suppose \( f : C \to D \) in \( \text{Set} \) is epic. Then it is a surjection. Suppose \( g : D \to C \) maps each \( d \in D \), to some chosen one of the elements \( c \) such that \( f(c) = d \). There will be such function assuming (but only assuming) the Axiom of Choice. Then \( f \circ g = 1_D \), so \( f \) is a retraction. \( \Box \)
Finally in this section, we should read into the record a couple of alternative bits of jargon:

**Definition 10.** A monomorphism which is a section (i.e. has a left-inverse) is said to be a *split monomorphism*. Dually, an epimorphism which is a retraction (i.e. has a right-inverse) is said to be a *split epimorphism*.

So, for example, we can say that the claim that every epimorphism splits in \( \textbf{Set} \) is the categorial version of the Axiom of Choice.

### 2.3 Isomorphisms

In familiar cases, we say that two structures ‘look just the same’ – are isomorphic – if there is a structure-preserving bijection between them. Well, in a category, the arrows typically do structure-preserving work between structured objects, and so we might expect the isomorphisms to be the bijective arrows. In the category \( \textbf{Set} \), the arrows which are both monic and epic are indeed the bijective functions. So shall we, more generally, define isomorphisms as arrows which are monic and epic?

No. Because if isomorphisms are to do the usual work of carving up structures into equivalence classes, they need to have inverses. But being monic and epic doesn’t always imply having an inverse. We can use again the toy case of \( 2 \), or here’s a generalized version of the same idea:

1. Take the category \( \mathcal{S} \) corresponding to the pre-ordered set \((S, \preceq)\). Then there is at most one arrow between any given objects of \( \mathcal{S} \). But if \( f \circ g = f \circ h \), then \( g \) and \( h \) must share the same object as domain and same object as codomain, hence \( g = h \), so \( f \) is monic. Similarly it is epic. But no arrows other than identities have inverses.

True, the arrows in that example aren’t functions, though we can easily construct toy categories with arrows which are set-functions and which share the key feature that there is at most one arrow between any pair of objects.

Here’s a more interesting case where the arrows *are* functions but being monic and epic still doesn’t imply being bijective, and so doesn’t imply having an inverse.

2. Consider the category \( \textbf{Rng} \) of rings. Among its objects are \( \mathbb{Z} \) and \( \mathbb{Q} \) (the integers and the rationals equipped with the usual functions). Let \( i: \mathbb{Z} \to \mathbb{Q} \) be the obvious inclusion map.

   Take some ring \( R \). First suppose we have two arrows (ring homomorphisms) \( g, h: R \to \mathbb{Z} \), where \( g \neq h \). So there is some element \( r \in R \), such that the integers \( g(r) \) and \( h(r) \) are different, which means that the corresponding rationals \( i(g(r)) \) and \( i(h(r)) \) are different, so \( i \circ g \neq i \circ h \). Contraposing, this means \( i \) is monic in the category.

   Now consider two other arrows \( j, k: \mathbb{Q} \to R \), where \( j \neq k \). That means there is some rational argument on which \( j \) and \( k \) disagree. But – and here we do use the fact that the maps \( j, k \) we are talking about are homomorphisms which preserve the structure of the rings – \( j \) and \( k \) can only disagree on some rational if they disagree on one of the integers the rational is a ratio of (why?). So \( j \) and \( k \) must disagree on \( i(p) \) for some integer \( p \). So \( j(i(p)) \) and \( k(i(p)) \) are different, and hence \( j \circ i \neq k \circ i \). Contraposing, this means \( i \) is epic in the category.

   Hence \( i \) is monic and epic. But it evidently lacks an inverse.
So: we can’t define an isomorphism as an epic monic if we want isomorphisms to have the essential feature of invertibility. Which all goes to motivate the following key definition:

**Definition 11.** An isomorphism (in category $\mathcal{C}$) is an arrow which has an inverse.

So isomorphisms are monic and epic but not always vice versa. However, we do have

**Theorem 10.** If $f$ is both monic and split epic (or both epic and split monic), then $f$ is an isomorphism.

*Proof.* If $f$ is a split epimorphism, it has a right inverse, i.e. there is a $g$ such that $f \circ g = 1$. Then $(f \circ g) \circ f = f$, whence $f \circ (g \circ f) = f \circ 1$. Hence, given that $f$ is also mono, $g \circ f = 1$, and $g$ is both a section and retraction for $f$, i.e. $f$ has an inverse. Dually for the other half of the theorem. \(\Box\)

**Convention.** It is standard to decorate arrows which are isomorphisms thus: $\sim\rightarrow$.

From what we have already seen, we know that/can readily check that

**Theorem 11.**

1. Identity arrows are isomorphisms.
2. An isomorphism $f: C \sim\rightarrow D$ has a unique inverse which we can call $f^{-1}: D \sim\rightarrow C$, such that $f^{-1} \circ f = 1_C$, $f \circ f^{-1} = 1_D$, $(f^{-1})^{-1} = f$, and $f^{-1}$ is also an isomorphism.
3. If $f$ and $g$ are isomorphisms, then $g \circ f$ is an isomorphism if it exists, whose inverse will be $f^{-1} \circ g^{-1}$.

**Definition 12.** If there is an isomorphism $f: C \sim\rightarrow D$ in $\mathcal{C}$ then $C, D$ are said to be isomorphic (in $\mathcal{C}$), and we write $C \cong D$.

Then, from the ingredients of Theorem 11, it is immediate that

**Theorem 12.** Isomorphism between objects in a category is an equivalence relation.

As you would expect, in the category $\textbf{Set}$ isomorphisms are bijections, in the category $\textbf{Grp}$ isomorphisms are group isomorphisms, in the category of vector spaces isomorphisms are invertible linear maps. And so it goes.
Our definition of an isomorphism characterizes a type of arrow not – as it were – ‘internally’, by how it operates on inputs (assuming it is indeed a function), but ‘externally’, by its interaction with other arrows. This is typical of a category-theoretic (re)definition of a familiar notion: we look for relations between arrows and/or structured objects, rather than digging ‘inside’ the relevant entities.

Here’s Awodey, offering some similarly arm-waving quick . . . remarks about category-theoretical definitions. By this I mean characterizations of properties of objects and arrows in a category in terms of other objects and arrows only, that is, in the language of category theory. Such definitions may be said to be abstract, structural, operational, relational, or external (as opposed to internal). The idea is that objects and arrows are determined by the role they play in the category via their relations to other objects and arrows, that is, by their position in a structure and not by what they ‘are’ or ‘are made of’ in some absolute sense. (Awodey, 2006, p. 25)

So a very natural way forward from where we’ve got to would be to proceed to give further ‘external’ category-theoretic definitions of familiar notions.

For example, take the usual ‘internal’ way of dealing with Cartesian products, sets of ordered pairs, by treating them as collections of (unordered!) Kuratowski-pairs of the kind \( \{\{x\}, \{x, y\}\} \). A familiar complaint is that this construction is arbitrary; there are endless alternatives. And the equally familiar riposte is: “Don’t worry. The construction works!” Yes; but what does ‘working’ come to here?

It means that there is a pairing function which takes us from sets \( X \) and \( Y \) to their Cartesian product, and a couple of matched unpairing functions which take a product and extract the original sets, and these pairing/upairing functions behave nicely (in ways we can and will spell out). But then, so long as we have suitable product-constructing and deconstructing functions, the ‘internal’ constitution of the product (e.g. as a collection of Kuratowski-pairs) doesn’t really matter at all. It’s the pattern of interrelations between various functions that matters. Which is just the sort of thing that category theory highlights.

For an approach to category-theory which starts by exploring categorial accounts of products and similar constructions, see e.g. Goldblatt (2006, Ch. 3). But in these notes, we will turn to consider products and the like rather later – indeed, not until Chapter 12. First, we go even more abstract and start considering not just the arrows/maps that can obtain between objects inside a category, but also maps between categories. There are pluses and minuses to taking this route via the stratosphere: but I’m following what
seems to be the Cambridge tradition in the Part III course (and also has its roots in the history of category theory). So let’s see how things work out.

### 3.1 Functors defined

The usual term for maps between categories is *functors*. A category $\mathcal{C}$ has two kinds of data, its objects and its arrows. So a functor $F$ from category $\mathcal{C}$ to category $\mathcal{D}$ will need to have two components, one that operates on objects, one that operates on arrows. Hence, to start our definition, we will say

**Definition 13.** Given categories $\mathcal{C}$ and $\mathcal{D}$, a functor $F : \mathcal{C} \to \mathcal{D}$ comprises the following data:

1. A mapping $\text{F}_{\text{ob}}$ whose value at the $\mathcal{C}$-object $A$ is some $\mathcal{D}$-object we can represent simply as $F(A)$ or indeed often just as $FA$.
2. A mapping $\text{F}_{\text{arw}}$ whose value at the $\mathcal{C}$-arrow $f : A \to B$ is a $\mathcal{D}$-arrow from $F(A)$ to $F(B)$ which we can represent as $F(f) : F(A) \to F(B)$, or indeed often just as $Ff : FA \to FB$.

But there’s more. If a functor is to preserve the most basic categorial structure, its component mappings must obey two obvious conditions. First they must map identity arrows to identity arrows. Second they should respect composition. That is to say, since the commutative diagram

\[
\begin{array}{ccc}
A & \xrightarrow{g \circ f} & C \\
\downarrow{g} & & \downarrow{F(g)} \\
B & \xrightarrow{F(f)} & F(C) \\
\end{array}
\]

the second diagram should also commute. Hence,

**Definition 13 continued.** The data in $F$ must satisfy the following axioms:

1. **Preserving identities:** for any $\mathcal{C}$-object $A$, $F(1_A) = 1_{FA}$;
2. **Respecting composition:** for any $\mathcal{C}$-arrows $f, g$ which compose, $F(g \circ f) = Fg \circ Ff$.

### 3.2 Some first examples of functors

Our first couple of examples illustrate two very broad types of case.

**F1** There is a functor $F : \text{Mon} \to \text{Set}$ with the following data:

1. $\text{F}_{\text{ob}}$ sends the monoid $(M, \cdot, 1_M)$ to its carrier set $M$.
2. $\text{F}_{\text{arw}}$ sends an arrow $f : (M, \cdot, 1_M) \to (N, \times, 1_N)$ (which is in fact a monoid homomorphism acting on elements on $M$) to the same function $f : M \to N$.

So defined, $F$ trivially obeys the axioms for being a functor. All it does is ‘forget’ about the structure carried by a monoid. It’s a *forgetful functor*, for short.

Similarly, there are other equally forgetful functors. For example, there is a functor $\text{Grp} \to \text{Set}$ that sends groups to their carrier sets, sends group homomorphisms to their underlying functions (but forgets about the group structure). And there is a functor $\text{Rng} \to \text{Grp}$ that sends a ring to the additive group it contains, forgetting the rest of the ring structure.

We are next going to construct a functor in the reverse direction to example (F1), i.e. a functor $F : \text{Set} \to \text{Mon}$. 
Let’s consider how to construct a functor making as few assumptions as we can about the monoid that a given set $M$ gets mapped to. Starting, then, from a set $M$, if we are going to get a monoid from it we need to equip it with a two-place associative function $\ast$. But we are assuming as little as we can about $\ast$, so we don’t even yet know that $\ast$ keeps us inside the original set $M$! So $M$ will need to get associated with a set $M^\ast$ that contains not only the original members of $M$, e.g. $x, y, z, \ldots$, but all the likes of $x \ast x, x \ast y, y \ast z, x \ast y \ast x, x \ast y \ast x \ast z, x \ast x \ast y \ast y \ast z, \ldots$, etc., etc. But even that’s not enough, for (in our assumption-free state) we don’t know that any of these elements of $M^\ast$ act as an identity for the $\ast$-function. So to get a monoid, we will have to throw in some identity element $1$.

Now, here’s a neat way to model the elements of $(M^\ast, \ast)$ [we’ll stop explicitly mentioning the monoid unit]. We model an element just as a finite sequence of members of $M^\ast$ that contains not only the original members of $M$, e.g. $x, y, z, \ldots$, but all the likes of $x \ast x, x \ast y, y \ast z, x \ast y \ast x, x \ast y \ast x \ast z, x \ast x \ast y \ast y \ast z, \ldots$, etc., etc. But we are assuming as little as we can about $\ast$, so we don’t even yet know that $1$ is just

Let’s consider how to construct a functor:

**(F2)** There is a functor $F : \textbf{Set} \to \textbf{Mon}$ with the following data:

i. $F_{\text{ob}}$ sends the set $M$ to the monoid $(M^\ast, \ast)$, where $M^\ast$ is the set of finite sequences of $M$-elements (including the null sequence) and $\ast$ is concatenation. $(M^\ast, \ast)$ is said to be the free monoid on $M$.

Now, if $F$ is to be a functor, what is it going to do with an arrow in $\textbf{Set}$, $f : M \to N$? $Ff$ must send elements of $M^\ast$ to elements of $N^\ast$. The obvious way to proceed is (a) for $Ff$ to send (the unit sequence of) an element $m$ of the original $M$ to (the unit sequence of) $f(m)$, and (b) for $Ff$ to respect concatenation. In other words

ii. $F_{\text{arr}}$ sends the arrow $f : M \to N$ to the arrow $Ff : M^\ast \to N^\ast$, defined by $Ff(x_1 \ast x_2 \ast x_3 \ast \ldots \ast x_n) = f(x_1) \ast f(x_2) \ast f(x_3) \ast \ldots \ast f(x_n)$.

As long as we remember about the null sequence, it is trivial to check that $Ff$ is a monoid homomorphism, and that $F$ is then a functor.

Similarly, there are other functors that send sets to freely generated structures on the set. For example there is a functor from $\textbf{Set}$ to $\textbf{Ab}$ which sends a set $X$ to the freely generated abelian group on $X$, i.e. the direct sum of $X$-many copies of $\mathbb{Z}$ (where we can think of $\mathbb{Z}$ as the free abelian group on one generator). [For more on freely generated abelian groups, see http://en.wikipedia.org/wiki/Free_abelian_group.]

Now for a third kind of example:

**(F3)** Take monoids $(M, \cdot)$ and $(N, \times)$ and considered the corresponding categories $\mathcal{M}$ and $\mathcal{N}$ in the sense of §1.2.

So $\mathcal{M}$ has a single object $\ast_{\mathcal{M}}$, and its arrows are elements of $M$, where the composition of the arrows $m_1$ and $m_2$ is just $m_1 \cdot m_2$, and the identity arrow is the identity element of the monoid, $1_M$.

Likewise $\mathcal{N}$ has a single object $\ast_{\mathcal{N}}$, and arrows are elements of $N$, where the composition of the arrows $n_1$ and $n_2$ is just $n_1 \times n_2$, and the identity arrow is the identity element of the monoid, $1_N$.

So now we see that a functor $F : \mathcal{M} \to \mathcal{N}$ will need to do the following:

i. $F$ must send $\ast_{\mathcal{M}}$ to $\ast_{\mathcal{N}}$.

ii. $F$ must send the identity arrow $1_M$ to the identity arrow $1_N$. 
iii. $F$ must send $m_1 \circ m_2$ (i.e. $m \cdot n$) to $Fm_1 \circ Fm_2$ (i.e. $Fm \times Fn$).

Apart from the trivial first condition, that just requires $F$ to be a monoid homomorphism. So any monoid homomorphism between two monoids induces a corresponding functor between the corresponding monoids-as-categories.

And for a similar kind of example to case (F3), consider pre-ordered sets as categories, or posets as categories. To take just the second case,

(F4) Take the posets $(S, \preceq)$ and $(T, \sqsubseteq)$ considered as categories $\mathcal{S}$ and $\mathcal{T}$. It is easy to check that a monotone function $f : S \to T$ induces a functor $F : (S, \preceq) \to (T, \sqsubseteq)$ which sends an $\mathcal{S}$-object $s$ to the $\mathcal{T}$-object $f(s)$, and sends an $\mathcal{S}$-arrow $\langle s, s' \rangle$ to the $\mathcal{T}$-arrow $\langle f(s), f(s') \rangle$.

Now, we have seen cases of functors that simply forget (some of the) structure put on structured sets. There will be other functors which – so to speak – collapse a category in on itself by forgetting distinctions between objects or arrows. Here’s an example.

(F5) Consider a category $\mathcal{C}$, and consider the corresponding slimmed-down category $\mathcal{S}$ which has the same objects but, whenever there are some $\mathcal{C}$-arrows from $A$ to $B$, $\mathcal{S}$ picks just one of them. (We’ll assume we have a strong enough choice principle to be able to do this.)

Evidently there is a functor $F : \mathcal{C} \to \mathcal{S}$ which sends an object to itself, and sends a $\mathcal{C}$-arrow from $A$ to $B$ to the unique $\mathcal{S}$-arrow from $A$ to $B$.

$\mathcal{S}$ is a pre-order category in the sense of §1.2, Ex. (8), and we could call $F$ the ‘pre-orderification’ functor on $\mathcal{C}$.

For a more extreme case, for any non-empty category $\mathcal{C}$ there is what we might call the ‘total collapse’ functor $F_\mathcal{C} : \mathcal{C} \to 1$ which sends every object of $\mathcal{C}$ to the sole object of 1, and sends every arrow in $\mathcal{C}$ to the sole arrow of 1.

And finally we should also mention the trivial case:

(F6) For any category $\mathcal{C}$ there is the trivial functor $1_\mathcal{C} : \mathcal{C} \to \mathcal{C}$ which sends each object in $\mathcal{C}$ to itself and sends each arrow in $\mathcal{C}$ to itself!

### 3.3 Subcategories and inclusion functors

Before we give another example of a functor we need an obvious definition:

**Definition 14.** Given a category $\mathcal{C}$, if $\mathcal{S}$ consists of the data

1. **Objects:** some or all of the $\mathcal{C}$-objects,
2. **Arrows:** some or all of the $\mathcal{C}$-arrows,

subject to the conditions

3. for each $\mathcal{S}$-object $C$, the $\mathcal{C}$-arrow $1_C$ is also an $\mathcal{S}$-arrow,
4. for any $\mathcal{S}$-arrows $f : C \to D$, $g : D \to E$, the arrow $g \circ f : C \to E$ is also an $\mathcal{S}$-arrow,

then $\mathcal{S}$ is a **subcategory** of $\mathcal{C}$.

Plainly, the conditions in the definition – containing identity arrows and being closed under composition – are there to ensure that $\mathcal{S}$ is indeed still a category.

Some examples:
(1) \( \mathcal{S} \) is a subcategory of \( \mathcal{C} \) if \( \mathcal{S} \) is the pre-orderification of \( \mathcal{C} \).

(2) The discrete category on the objects of \( \mathcal{C} \) is a subcategory of \( \mathcal{C} \) for any category.

(3) \( \text{FinSet} \) is a subcategory of \( \text{Set} \).

(4) \( \text{Ab} \) is a subcategory of \( \text{Grp} \).

So we can shed objects and/or arrows in moving from a category to a subcategory. The first two examples are cases where we shed some or all of the non-identity arrows while keeping all the objects. But the second two cases are ones where drop some objects while keeping all the arrows between those objects retained in the subcategory, and there is a standard label for such cases:

**Definition 15.** If \( \mathcal{S} \) is a subcategory of \( \mathcal{C} \) where, for all \( \mathcal{S} \)-objects \( A \) and \( B \), the \( \mathcal{S} \)-arrows from \( A \) to \( B \) are all the \( \mathcal{C} \)-arrows from \( A \) to \( B \), then \( \mathcal{S} \) is said to be a **full subcategory** of \( \mathcal{C} \).

And now note that

(F7) If \( \mathcal{S} \) is a subcategory of \( \mathcal{C} \), then there is an (obviously defined) inclusion functor \( F: \mathcal{S} \to \mathcal{C} \). (We reserve the notation \( F: \mathcal{S} \hookrightarrow \mathcal{C} \) for the case when \( \mathcal{S} \) is a full subcategory of \( \mathcal{C} \).)

### 3.4 What do functors preserve?

Here’s a first result about what functors can do for us:

**Theorem 13.** **Functors** preserve sections, retractions and isomorphisms.

**Proof.** Easy! Recall, \( f: C \to D \) is a section in the category \( \mathcal{C} \) iff for some arrow \( g \), \( g \circ f = 1_C \). Let \( F \) be a functor mapping the category \( \mathcal{C} \) to \( \mathcal{D} \). Then \( F(g \circ f) = F(1_C) \).

By the functorial nature of \( F \), that implies \( F(g) \circ F(f) = 1_D \). So \( F(f) \) is a section in the category \( \mathcal{D} \).

Duality gives the result for retractions. And putting the two results together gives us the result for isomorphisms.

Now we said, arm-wavingly, that functors respect basic categorial structure. But that doesn’t mean that they preserve **all** structural properties defined in categorial terms:

**Theorem 14.** **Functors do not necessarily** preserve monomorphism and epimorphisms.

**Proof.** Recall that in the category \( \text{Rng} \) the inclusion map \( i_R: \mathbb{Z} \to \mathbb{Q} \) is an epimorphism. But in the category \( \text{Set} \) the inclusion map \( i_S: \mathbb{Z} \to \mathbb{Q} \) is obviously **not** an epimorphism (for in general, maps that agree on their actions on rational integers need not agree on their action on other rationals). The forgetful functor \( F: \text{Rng} \to \text{Set} \) maps the epimorphism \( i_R \) to the non-epic \( i_S \). Hence functors do not necessarily preserve epimorphisms. The other half of the theorem follows by duality.

### 3.5 An example from algebraic topology

We’ll now briefly consider another example of a functor, from algebraic topology, which also allows us to apply Theorem 13. This example can readily be skipped – though in fact, to follow it will be enough to have read a few early pages from e.g. Allen Hatcher’s
**Algebraic Topology.** Even more briefly, to get a glimmer of what’s going on, you just need the idea of the fundamental group of a topological space (at a point).

Thus, given a space and a chosen base point in it, consider all directed paths that start at this base point, wander around and eventually loop back to their starting point. Such directed loops can be “added” together in an obvious way: you traverse the “sum” of two loops by going round the first loop, then round the second. Every loop has an “inverse” (you go round the same path in the opposite direction). Two loops are considered ‘homotopically’ equivalent if one can be continuously deformed into the other without breaking. Consider, then, the set of all such equivalence classes of loops – so-called homotopy equivalence classes – and define “addition” for these classes in the obviously derived way. This set, when equipped with addition, evidently forms a group: it is the fundamental group for that particular space, with the given basepoint. (Though for many spaces, the group is independent of the basepoint.)

Suppose, therefore, that **Top** is the category of pointed topological spaces: an object in the category is a topological space **X** equipped with a distinguished base point **x**. and the arrows in the category are continuous maps that preserve basepoints. Then:

(F8) There is a functor \( \pi_1 : \text{Top}_* \to \text{Grp} \), to use its standard label, with the following data

i. \( \pi_1 \) sends a pointed topological space \((X, x_0)\) – i.e. \( X \) with base point \( x_0 \) – to the fundamental group \( \pi_1(X, x_0) \) of \( X \) at \( x_0 \).

ii. \( \pi_1 \) sends a basepoint-preserving continuous map \( f : (X, x_0) \to (Y, y_0) \) to a corresponding group homomorphism \( f_* : \pi_1(X, x_0) \to \pi_1(Y, y_0) \). [For arm-waving motivation: \( f \) maps a continuous loop based at \( x_0 \) to a continuous loop based at \( y_0 \); and since \( f \) is continuous it maps a continuous deformation of a loop in \((X, x_0)\) to a continuous deformation of a loop \((Y, y_0)\) – and that induces a corresponding association \( f_* \) between the homotopy equivalence classes of \((X, x_0)\) and \((Y, y_0)\), and this will respect the group structure.]

Here’s an application. We’ll prove Brouwer’s Fixed Point Theorem (and here at last, you might say, we get something worth calling a *theorem!*):

**Theorem 15.** Any continuous map of the closed unit disc to itself has a fixed point.

**Proof.** Suppose that there is a continuous map \( f \) on the two-dimensional disc \( D \) (considered as a topological space) such that we always have \( f(x) \neq x \).

Let the boundary of the disc be the circle \( S \) (again considered as a topological space). Then we can define a map that sends the point \( x \) in \( D \) to the point in \( S \) at which the ray from \( f(x) \) through \( x \) intersects the boundary of the disc.

This map sends a point on boundary to itself. Pick a boundary point to the the base point of the pointed space \( D_* \) and also of the pointed space \( S_* \), then our map induces a map \( r : D_* \to S_* \). Moreover, this map is evidently continuous (intuitively: nudge a point \( x \) and since \( f \) is continuous that just nudges \( f(x) \), and hence the ray from \( f(x) \) through \( x \) is only nudged, and the point of intersection with the boundary is only nudged). And \( r \) is a retraction of inclusion map \( i : S_* \hookrightarrow D_* \) in **Top**., since \( r \circ i = 1 \).

Functors preserve retractions by Theorem 13, so \( \pi_1(r) \) will be a retraction of \( \pi_1(i) \) in the category **Grp**, which means that \( \pi_1(i) : \pi_1(S_*) \to \pi_1(D_*) \) is a section in **Grp**, hence by Theorem 8 is monic, and hence by Theorem 4 is an injection.

But that’s impossible. \( \pi_1(S_*) \), the fundamental group of \( S_* \), is [equivalent to] the group \( \mathbb{Z} \) of integers under addition (think of looping round a circle, one way or another, \( n \) times – each positive or negative integer corresponds to a different path); while \( \pi_1(D_*) \),
the fundamental group of $D_*$, is just a one element group (for every loop in $D_*$ can be smoothly shrunk to a point). And there is no injection between the integers and a one-element set!

What, if anything, do we gain from putting the proof in category theoretic terms? It might be said: the proof crucially depends on facts of algebraic topology – continuous maps preserve homotopic equivalences in a way that makes $\pi_1$ a functor (not entirely trivial), and the fundamental groups of $S^*$ and $D^*$ are respectively $\mathbb{Z}$ and the trivial group. And we could run the whole proof without actually mentioning categories at all. But what we’ve done is, so to speak, very clearly demarcate those bits of the proof that do specifically depend on facts of algebraic topology and those bits which depend on more general proof-ideas which are thoroughly portable to other contexts. And that surely counts as a gain in understanding.
More about functors and categories

4.1 Functors compose

**Theorem 16.** Suppose we have functors $F: \mathcal{C} \to \mathcal{D}$, $G: \mathcal{D} \to \mathcal{E}$, then there is a functor $G \circ F: \mathcal{C} \to \mathcal{E}$ with the following data:

1. A mapping $(G \circ F)_{\text{obj}}$ which sends a $\mathcal{C}$-object $A$ to the $\mathcal{E}$-object $GFA$ – i.e., if you prefer that with brackets, to $G(F(A))$.

2. A mapping $(G \circ F)_{\text{arr}}$ which sends a $\mathcal{C}$-arrow $f: A \to B$ to the $\mathcal{E}$-arrow $GFA \to GFB$ – i.e. to $G(F(f))$.

An elementary check confirms that the axioms for being a functor are satisfied. It is also elementary to check that we have:

**Theorem 17.** Composition of functors is associative.

4.2 Covariant vs contravariant functors

(a) How do functors interact with the operation of taking the opposite category?

Well, first we note:

**Theorem 18.** A functor $F: \mathcal{C} \to \mathcal{D}$ induces a functor $F^{\text{op}}: \mathcal{C}^{\text{op}} \to \mathcal{D}^{\text{op}}$.

**Proof.** Recall, the objects of $\mathcal{C}^{\text{op}}$ are exactly the same as the objects of $\mathcal{C}$. We can therefore define the object-mapping component of $F^{\text{op}}$ as acting on $\mathcal{C}^{\text{op}}$-objects exactly as the object-mapping component of $F$ acts on $\mathcal{C}$-objects. And then, allowing for the fact that taking opposites reverses arrows, we can define the arrow-mapping component of $F^{\text{op}}$ as acting on the $\mathcal{C}^{\text{op}}$-arrow $f: C \to D$ exactly as the arrow-mapping component of $F$ acts on the $\mathcal{C}$-arrow $f: C \to D$.

$F^{\text{op}}$ will evidently obey the axioms for being a functor because $F$ does. \qed

It is trivial, by the way, that $(F^{\text{op}})^{\text{op}} = F$.

But now for a new departure: we introduce a new kind of functor:

**Definition 16.** $F$ is a **contravariant** functor from $\mathcal{C}$ to $\mathcal{D}$ if $F: \mathcal{C}^{\text{op}} \to \mathcal{D}$ is a functor in the original sense. So it comprises the following data:

1. A mapping $F_{\text{obj}}$ whose value at the $\mathcal{C}$-object $A$ is some $\mathcal{D}$-object $F(A)$.  

22
A mapping $F_{\text{arw}}$ whose value at the $\mathcal{C}$-arrow $f : B \to A$ is the $\mathcal{D}$-arrow $F(f) : FA \to FB$.

And this data satisfies the two axioms:

Preserving identities: for any $\mathcal{C}$-object $A$, $F(1_A) = 1_{F(A)}$;

Respecting composition: for any $\mathcal{C}$-arrows $f, g$ which compose, $F(g \circ f) = Ff \circ Fg$.

In the light of the preceding theorem, it would of course be equivalent to define a contravariant functor from $\mathcal{C}$ to $\mathcal{D}$ to be a functor $F : \mathcal{C} \to \mathcal{D}^{\text{op}}$. Note: a functor in our original sense, when the contrast needs to be stressed, is also called a covariant functor.

Let’s have an example of a naturally arising contravariant functor. Take $\mathbb{V}$, the category whose objects are the finite dimension vector spaces over the reals, and whose arrows are linear maps between spaces. Now recall, the dual space of such a vector space $V$ is $V^*$, the set of all linear functions $f : V \to \mathbb{R}$ (where this set is equipped with vectorial structure in the obvious way). $V^*$ has the same dimension as $V$ (so, a fortiori, is also finite dimensional and belongs to $\mathbb{V}$). We’ll construct a dualizing functor-style map $D$ from the category $\mathbb{V}$ to itself where $D_{\text{ob}}$, acting on objects, sends a vector-space to its dual.

So now, how is our functor $D$ going to act on arrows in the category $\mathbb{V}$? Take spaces $V, W$ and consider any linear map $g : W \to V$. Then, over on the dual spaces, there will be a naturally corresponding map $(- \circ g) : V^* \to W^*$ which maps $f : V \to \mathbb{R}$ to $f \circ g : W \to \mathbb{R}$ (note the direction that $g$ has to go in for a composition with $f$ to work). This defines the action of a component $D_{\text{arw}}$ for the dualizing map $D$: it sends a linear map $g$ to the map $(- \circ g)$.

And these components $D_{\text{ob}}$ and $D_{\text{arw}}$ evidently do give us a contravariant functor.

### 4.3 Product categories, arrow categories, and associated functors

‘Categories beget categories.’ We’ve already seen one way they do this: for every category there is an opposite category. In the rest of this chapter we meet some other ways of getting new categories from old (but this is all rather abstract for now and I hesitate to cover the material at this stage: so do feel very free to skim and skip until you really need these ideas in later applications).

**Definition 17.** Given categories $\mathcal{C}$ and $\mathcal{D}$, $\mathcal{C} \times \mathcal{D}$ is the derived product category with the following data:

1. Its objects are all the ordered pairs $\langle C, D \rangle$, for $C$ a $\mathcal{C}$-object and $D$ a $\mathcal{D}$-object.
2. For any two such pairs $\langle C, D \rangle, \langle C', D' \rangle$, the associated arrows from $\langle C, D \rangle$ to $\langle C', D' \rangle$ are the pairs $\langle f, f' \rangle$, where $f : C \to D$ and $f' : C' \to D'$.
3. For each $\langle C, D \rangle$, there is an identity arrow $1_{\langle C, D \rangle}$ which is the pair $\langle 1_C, 1_D \rangle$.
4. Composition of arrows is defined component-wise: $\langle f, f' \rangle \circ \langle g, g' \rangle = \langle f \circ g, f' \circ g' \rangle$.

It is immediate that this does indeed define a category.

Associated with a product category $\mathcal{C} \times \mathcal{D}$ there will be a pair of ‘projection’ functors $P^1 : \mathcal{C} \times \mathcal{D} \to \mathcal{C}, P^2 : \mathcal{C} \times \mathcal{D} \to \mathcal{D}$ defined in the obvious way. Thus

1. $P^1_{\text{ob}}$ maps $\langle C, D \rangle$ to $C$.
2. $P^1_{\text{arw}}$ maps a $\mathcal{C} \times \mathcal{D}$-arrow $\langle f, g \rangle$ to the $\mathcal{C}$-arrow $f$. 
And similarly, of course, for the projection functor $P^2$.

We then have a nice theorem which tells us that we can in a sense combine two functors from a given category into the two components of a product category to give us a new functor into the product category:

**Theorem 19.** Given categories $\mathcal{A}, \mathcal{C}, \mathcal{D}$, and functors $F : \mathcal{A} \to \mathcal{C}$, and $G : \mathcal{A} \to \mathcal{D}$, there is a unique functor $H : \mathcal{A} \to \mathcal{C} \times \mathcal{D}$ such that $P^1 \circ H = F$ and $P^2 \circ H = G$.

**Proof.** The only possibility is for $H$ to send an object $A$ in $\mathcal{A}$ to the object $\langle FA, GA \rangle$, and send an arrow $f$ in $\mathcal{A}$ to the arrow $\langle Ff, Gf \rangle$. But this is indeed a functor.

(b) Let’s also quickly mention another example of a derived category (if only to illustrate that the objects of category need not be object-like!):

**Definition 18.** Given a category $\mathcal{C}$, the derived arrow category $\mathcal{C} \to \to$ has the following data:

1. $\mathcal{C} \to \to$’s objects are simply the arrows of $\mathcal{C}$,
2. Given $\mathcal{C} \to \to$-objects $f_1, f_2$, i.e. $\mathcal{C}$-arrows $f_i : X_i \to Y_i$, an arrow $a : f_1 \to f_2$ is a pair $\langle j, k \rangle$ of $\mathcal{C}$-arrows such that the following diagram commutes:

$$
\begin{array}{ccc}
X_1 & \xrightarrow{f_1} & Y_1 \\
\downarrow{j} & & \downarrow{k} \\
X_2 & \xrightarrow{f_2} & Y_2
\end{array}
$$

Again it is easily checked that, with composition defined in the only sensible way, this does define a category. (Consider the operation of sending a category $\mathcal{C}$ to the corresponding arrow category $\mathcal{C} \to \to$. Question: is this functorial?)

### 4.4 Slice categories and associated functors

Suppose then that $\mathcal{C}$ is a category, and $I$ a particular $\mathcal{C}$-object. We will define a new category from $\mathcal{C}$, this time the so-called ‘slice’ category $\mathcal{C}/I$ where – as in an arrow category – the new category’s objects are in fact (some of) the original category’s arrows. (To keep things clear but brief, let’s use ‘arrow’ to refer to the old arrows and reserve plain ‘arrow’ for the new arrows to be found in $\mathcal{C}/I$.)

We can now give the following definition:

**Definition 19.** Let $\mathcal{C}$ be a category, and $I$ be a $\mathcal{C}$-object. Then the new category $\mathcal{C}/I$ – the slice category over $I$ – has the following data:

1. The $\mathcal{C}/I$-objects are all the arrows $\mathcal{C} f : A \to I$, for any $\mathcal{C}$-object $A$.
2. For $\mathcal{C}/I$-objects (yes, objects!) $f : A \to I$, $g : B \to I$, the associated arrows from $f$ to $g$ are just the arrows $\mathcal{C} j : A \to B$ such that $g \circ j = f$ in $\mathcal{C}$.
3. For every $\mathcal{C}/I$-object $f : A \to I$, there is an identity arrow, namely the arrow $1_A : A \to A$.
4. For any $\mathcal{C}/I$-objects $f : A \to I$, $g : B \to I$, $h : C \to I$, the composition of $j : f \to g$ and $k : g \to h$ is defined as follows: $k \circ j : f \to h$ is the arrow $k \circ j : A \to C$. 

Of course, we need to check that these data do indeed together satisfy the axioms for constituting a category. So let’s do that.

First, a diagram might help. Take the arrows $g: A \to C$ and $j: B \to C$. These will be objects $g$, $j$ of $\mathcal{C}/\mathcal{I}$. And the arrows of $\mathcal{C}/\mathcal{I}$ from $f$ to $g$ will be the arrows $\mathcal{C}/\mathcal{I}$ like $j: A \to B$ which make our first diagram commute. Note: the domain and co-domain of $j$ as an arrow $\mathcal{C}/\mathcal{I}$ are respectively $A$, $B$. But the domain and co-domain of $j$ as an arrow in the slice category $\mathcal{C}/\mathcal{I}$ are respectively $f$ and $g$.

We now need to confirm that our definition of $k \circ j$ for $\mathcal{C}/\mathcal{I}$ works. We are given in $\mathcal{C}$ that $j: A \to B$ is such that $g \circ j = f$, and also that $h \circ k = g$. Putting things together we get the second commutative diagram.

Or in equations, we have $(h \circ k) \circ j = f$ in $\mathcal{C}$, and therefore $h \circ (k \circ j) = f$.

So $(k \circ j)$ does indeed count as an arrow in $\mathcal{C}/\mathcal{I}$ from $f$ to $h$, as we require.

The remaining checks to confirm $\mathcal{C}/\mathcal{I}$ satisfies the axioms for being a category are then trivial.

Unsurprisingly, there’s a dual notion, the idea of a co-slice category, $I/\mathcal{C}$ whose objects are the arrows $\mathcal{C}/\mathcal{I}$, for any $\mathcal{C}$-object $A$ and the rest of the definition is as you would expect.

Here are a couple of quick examples of slice and co-slice categories, one of each kind:

1. First, we mentioned before the idea of a pointed topological space, i.e. a space $X$ equipped with a selected base-point $x$ (see §3.5). Now, we can think of a one-element set as trivial topological space, call it ‘1’. And we also mentioned before the idea that we can think of any element $x \in X$ as an arrow $\vec{x}: 1 \to X$ (a map from a singleton to $X$). So now think about the co-slice category $1/\text{Top}$. Its objects are all the morphisms $\vec{x}: 1 \to X$ (for any space $X$, one morphism for every point in $X$): we can think of each such morphism as equipping some space $X$ with a basepoint $x$. And the arrows in $1/\text{Top}$ from some $\vec{x}: 1 \to X$ to some $\vec{y}: 1 \to Y$s are all the maps $f: X \to Y$ from the original category such that $f \circ x = y$, and we can think of such maps as the continuous maps which preserve basepoints. So we can think of $1/\text{Top}$ as being (or at least, in some strong sense, as being ‘the same as’) the category $\text{Top}_*$ of pointed topological spaces.

2. For our second example, take an $n$-membered index set $I_n = \{c_1, c_2, c_3, \ldots, c_n\}$. Think of the members of the set as ‘colours’. Then a morphism $S \to I_n$ is an $n$-colouring of the set $S$. So we can think of $\text{FinSet}/I_n$ as the category of $n$-coloured finite sets, which is the sort of thing that combinatorialists might be interested in.

And now here a couple of examples of functors involving slice categories (and of course there will be corresponding constructions for co-slice categories):

1. There is another sort of forgetful functor, $F: \mathcal{C}/\mathcal{I} \to \mathcal{C}$ which just forgets about the object being sliced over! So $F$ sends a $\mathcal{C}/\mathcal{I}$-object $A \to I$ back to $A$, and $F$ sends an arrow $j$ in $\mathcal{C}/\mathcal{I}$ back to the original arrow $j$ in $\mathcal{C}$.

2. Next, let’s show how we can use an arrow $k: I \to J$ (for $I, J \in \mathcal{C}$) to generate a corresponding functor $K: \mathcal{C}/\mathcal{I} \to \mathcal{C}/\mathcal{J}$.

The functor needs to act on objects in $\mathcal{C}/\mathcal{I}$ and send them to objects in $\mathcal{C}/\mathcal{J}$. That is to say, $K_{\mathcal{I}}$ needs to send an arrow $\mathcal{C}/\mathcal{I}$ $f: X \to I$ to an arrow $\mathcal{C}/\mathcal{J}$ with codomain $J$. The obvious thing to do is to put $K_{\mathcal{I}}(f) = k \circ f$. 
And how will $K$ act on arrows of $\mathcal{C}/I$? Another diagram might help! The $\mathcal{C}/I$-arrows from $f: A \to I$ to $g: B \to I$, by definition, include any $j$ which makes the left-hand inner triangle commute. But then such a $j$ will also make the outer triangle commute, i.e. $j$ is an arrow from $k \circ f: A \to J$ to $k \circ g: B \to J$ (which is therefore an arrow from $K(f)$ to $K(g)$). So we can simply put $K(j)$ for $j: f \to g$ in $\mathcal{C}/I$ to be $j$ (i.e. $j: K(f) \to K(g)$ in $\mathcal{C}/J$).

It is then readily checked that $K$ is indeed a functor from $\mathcal{C}/I$ to $\mathcal{C}/J$.

### 4.5 Comma categories

The last section in this chapter again looks ahead. The notion of a so-called comma-category introduced in this section won’t re-appear until §11.4. This is as good a place as any to first introduce the ideas, as it explains yet another way of getting new categories from old; but (as we said before) by all means just skim through for the moment.

**Definition 20.** Given functors $S: \mathcal{A} \to \mathcal{C}$ and $T: \mathcal{B} \to \mathcal{C}$, then the ‘comma category’ $(S \downarrow T)$ is the category with the following data:

1. The objects of $(S \downarrow T)$ are triples $\langle A, f, B \rangle$ where $A$ is an $\mathcal{A}$-object, $B$ is a $\mathcal{B}$-object, and $f: SA \to TB$ is an arrow in $\mathcal{C}$.

2. An arrow of $(S \downarrow T)$ from $\langle A, f, B \rangle$ to $\langle A', f', B' \rangle$ is a pair $\langle a, b \rangle$, where $a: A \to A'$ is an $\mathcal{A}$-arrow, $b: B \to B'$ is an $\mathcal{B}$-arrow, and the following diagram commutes:

$$
\begin{bmatrix}
SA & \rightarrow & TB \\
\downarrow^{Sa} & & \downarrow^{Tb} \\
SA' & \rightarrow & TB'
\end{bmatrix}
$$

3. The identity arrow on the object $\langle A, f, B \rangle$ is the pair $\langle 1_A, 1_B \rangle$.

4. Composition in $(S \downarrow T)$ is induced by the composition laws of $\mathcal{A}$ and $\mathcal{B}$, thus: $\langle a', b' \rangle \circ \langle a, b \rangle = \langle a' \circ_{\mathcal{A}} a, b' \circ_{\mathcal{B}} b \rangle$.

It is easily checked that, so defined, $(S \downarrow T)$ is indeed a category. The standard label ‘comma category’ arises from an earlier notation ‘$(S,T)$’. And our use of ‘$S$’ and ‘$T$’ for the two functors here is supposed to be helpfully indicate ‘source’ and ‘target’.

The notion of a comma category generalizes a number of simpler constructions. And indeed, we have already met two comma categories in thin disguise.

1. Take the minimal case where $\mathcal{A} = \mathcal{B} = \mathcal{C}$, and where both $S$ and $T$ are the identity functor on that category, $1_\mathcal{C}$.

Then the objects in this category $(1_\mathcal{C} \downarrow 1_\mathcal{C})$ are triples $\langle X, X \xrightarrow{f} Y, Y \rangle$ for $X, Y$ both $\mathcal{C}$-objects. And an arrow from $\langle X, X \xrightarrow{f} Y, Y \rangle$ to $\langle X', X' \xrightarrow{f'} Y', Y' \rangle$ is a pair of $\mathcal{C}$-arrows $a: X \to X'$, $b: Y \to Y'$ such that the following diagram commutes:
So the only difference between \((1_{\mathcal{C}} \downarrow 1_{\mathcal{C}})\) and the arrow category \(\mathcal{C} \rightarrow\) is that we have now ‘decorated’ the objects of \(\mathcal{C} \rightarrow\), i.e. \(\mathcal{C}\)-arrows \(f: X \rightarrow Y\), with explicit assignments of their sources and targets as \(\mathcal{C}\)-arrows, to give triples \((X, X \xrightarrow{f} Y, Y)\). Hence \((1_{\mathcal{C}} \downarrow 1_{\mathcal{C}})\) and \(\mathcal{C} \rightarrow\), although not identical, come to the same. (Later, we’ll have an official notion of isomorphic categories, and these categories are trivially isomorphic.)

(2) Take secondly the special case where \(\mathcal{A} = \mathcal{C}\) with \(S\) the identity functor \(1_{\mathcal{C}}\), and where \(\mathcal{B} = 1\) (the category with a single object \(\star\) and the single arrow \(1_{\star}\)). Now, a functor \(1 \rightarrow \mathcal{C}\) will send \(\star\) to some individual \(\mathcal{C}\)-object \(I\) (and send the identity arrow on \(\star\) to the identity arrow \(1_{I}\)); let’s label this functor \(I\) too.

Applying the definition, the objects of the category \((1_{\mathcal{C}} \downarrow I)\) are therefore triples \(\langle A, A \xrightarrow{f} I, \star \rangle\), and an arrow between \(\langle A, A \xrightarrow{f} I, \star \rangle\) and \(\langle B, B \xrightarrow{g} I, \star \rangle\) will be a pair \(\langle j, 1_{\star} \rangle\), with \(j: A \rightarrow B\) an arrow such the diagram on the left commutes:

![Diagram](image)

But the diagram on the left is equivalent to that on the right – which should look familiar! We’ve ended up with something tantamount to the slice category \(\mathcal{C}/I\), the only differences being that (i) instead of the slice category’s objects \(f: A \rightarrow I\) (for \(A \in \mathcal{C}\) we now have ‘decorated’ objects \(\langle A, A \xrightarrow{f} I, \star \rangle\) which correspond one-to-one with them, and (ii) instead of the slice category’s arrows \(j: A \rightarrow B\) we have decorated arrows \(\langle j, 1_{\star} \rangle\) which correspond one-to-one with them.

Again later, in §6.3, we’ll be able to do better than saying we have ‘something tantamount’ to a slice category here: we will be able to say that the categories \((1_{\mathcal{C}} \downarrow I)\) and \(\mathcal{C}/I\) are isomorphic categories.
Natural transformations

Category theory is an embodiment of Klein’s dictum that it is the maps that count in mathematics. If the dictum is true, then it is the functors between categories that are important, not the categories. And such is the case. Indeed, the notion of category is best excused as that which is necessary in order to have the notion of functor. But the progression does not stop here. There are maps between functors, and they are called natural transformations. (Freyd 1965, quoted in Marquis 2008.)

Natural transformations, as special morphisms-between-functors, were there from the very start. The founding document of category theory is the paper by Samuel Eilenberg and Saunders Mac Lane ‘General theory of natural equivalences’ (Eilenberg and Mac Lane, 1945). But the key idea there had already been introduced, three years previously, in a paper on ‘Natural isomorphisms in group theory’, before the categorial framework was invented precisely to provide a general setting for the account (Eilenberg and MacLane, 1942). So natural transformations are going to be central to our story.

5.1 Motivation

It might be helpful scene-setting, then, to pause to reflect in some detail about the kind of pre-category-theoretic problem that the account of natural transformations aims to solve. Let’s take in stages an example used by Eilenberg and Mac Lane:

(a) Consider a finite dimensional vector space over the reals, $V$, and the corresponding dual space $V^*$ of linear functions $f : V \to \mathbb{R}$. It is elementary to show that $V$ is isomorphic to $V^*$ (there’s a bijective linear map between the spaces).

Proof sketch: Take a basis $B = \{v_1, v_2, \ldots, v_n\}$ for $V$. Define the functions $v_i^* : V \to \mathbb{R}$ by putting $v_i^*(v_j) = 1$ if $i = j$ and $v_i^*(v_j) = 0$ otherwise. Then $B^* = \{v_1^*, v_2^*, \ldots, v_n^*\}$ is a basis for $V^*$, and the linear function $\varphi^B : V \to V^*$ generated by putting $\varphi^B(v_i) = v_i^*$ is an isomorphism.

Note, however, that the isomorphism we have arrived at here depends on the initial choice of basis $B$. (Look at it like this: our map $\varphi^B$ in fact defines an inner product for the space $V$, where for $e, e' \in V$, $\langle e, e' \rangle = \varphi^B(e)(e')$. Now recall that inner products need not be preserved by a change of basis.) But no choice of basis is more ‘natural’ than any other; none of the isomorphisms from $V$ to $V^*$ of the kind we’ve just defined is to be especially preferred.
To get a contrasting case, now consider the double dual of $V$, i.e. $V^{**}$ the space of functionals $g: V^* \rightarrow \mathbb{R}$. We can again get an isomorphism by selecting a basis $B$ for $V$, defining a derived basis $B^*$ for $V^*$ as we just did, and then using that basis in turn to define a basis $B^{**}$ for $V^{**}$ by repeating the same construction. And then we can define an isomorphism from $V$ to $V^{**}$ by mapping the elements of $B$ to the corresponding elements of $B^{**}$.

However, we don’t have to go through the palaver of choosing bases. Suppose we define $\psi_V$ as acting on an element $v \in V$ to give as output the functional $g: V^* \rightarrow \mathbb{R}$ which sends a function $f: V \rightarrow \mathbb{R}$ to the value $f(v)$. So, in short, $\psi_V(v)(f) = f(v)$. It is readily checked that this is an isomorphism (we rely on the fact that $V$ is finite-dimensional). And we get this isomorphism independently of any arbitrary choice of basis: so it is in a sense intrinsically ‘natural’.

Interim summary: it is very natural(!) to say that the isomorphisms of the kind we just described between $V$ and $V^*$ are not intrinsic, are not ‘natural’ to the spaces involved. By contrast there is a ‘natural’ isomorphism between $V$ and $V^{**}$, generated by a general procedure that applies to any vector space.

Now, this is far from being the only case where we might want to contrast intuitively ‘natural’ from more arbitrarily cooked-up maps between structured objects. The story goes that such talk was already bandied about quite a bit e.g. by topologists in the 1930s, but without any account of what it meant. Eilenberg and Mac Lane were aiming to provide the missing story.

(b) To continue with our example, the idea is that the isomorphism $\psi_V: V \rightarrow V^{**}$ which we constructed is natural because the only information about $V$ it relies on is that $V$ has the structure of a finite dimensional vector space. So the construction will work the same way on other such vector spaces, so we get a corresponding isomorphism $\psi_W: W \rightarrow W^{**}$, and we will expect such naturally constructed isomorphisms to be respected by maps between the spaces $V$ and $W$ (and the induced maps from $V^{**}$ to $W^{**}$).

We can clarify that last requirement snappily by saying that we want the following diagram to commute, whatever vector spaces we take and for any structure-preserving $f: V \rightarrow W$ (i.e. any linear map),

\[
\begin{array}{ccc}
V & \xrightarrow{f} & W \\
\downarrow{\psi_V} & & \downarrow{\psi_W} \\
V^{**} & \xrightarrow{DD(f)} & W^{**}
\end{array}
\]

where $DD(f)$ is the double-dual correlate of $f$.

Recall, back in §4.2, we saw that the correlate $Df$ of $f: V \rightarrow W$ is the functional $(- \circ f): W^* \rightarrow V^*$; and then moving to the double dual, the correlate $DDf$ will be the functional we can notate $(- \circ (- \circ f)): V^{**} \rightarrow W^{**}$. And then our diagram does indeed commute, both paths sending an element $v \in V$ to the functional that maps a function $k: V \rightarrow \mathbb{R}$ to the value $k(f(x))$. Think about it!

(c) So far, so good. Now let’s think about why there can’t be a similarly ‘natural’ isomorphism from $V$ to $V^*$. (The isomorphisms based on an arbitrary choice of basis aren’t natural: but we want to show that there is no other ‘natural’ isomorphism either.)

Suppose that there was a construction which gave us an isomorphism $\varphi_V: V \rightarrow V^*$ which again does not depend on information about $V$ other than that it has the structure of a finite dimensional vector space. So again we want the construction to work the same
way on other such vector spaces, and to be preserved by structure-preserving maps between the spaces. This time, then, we want the following diagram to commute for any structure-preserving $f$ between vector spaces (note we have to reverse an arrow for things to make sense, given our definition of the functional $D$):

$$
\begin{array}{ccc}
V & \overset{f}{\longrightarrow} & W \\
\downarrow{\varphi_V} & & \downarrow{\varphi_W} \\
V^* & \overset{D(f)}{\longleftarrow} & W^*
\end{array}
$$

Now, remember the $\varphi$s are supposed to be isomorphisms, so the diagram commutes just so long as $\varphi^{-1}_V \circ D(f) \circ \varphi_W \circ f = 1_V$. [Recall, we built into the idea of a diagram commuting that the composition of arrows along a closed path must be equal to the identity arrow.] So – in category-speak – $f$ has a retraction. But it is obvious that in general, a linear map $f: V \to W$ need not have a retraction. So there can’t in general be isomorphisms $\varphi_V, \varphi_W: V \to V^*$ making that diagram commute.

(d) ‘Ah! Perhaps what the last argument shows is that we were asking too much. Perhaps intuitive naturality requires only that the diagram commutes for any mapping which fully preserves structure, i.e. it requires that we get a commuting diagram only for arrows $f: V \xrightarrow{\sim} W$ which are isomorphisms.’ Nice try! But this revised proposal doesn’t rescue the situation. For any isomorphism $f$, we will still have $\varphi_V(v_1) = D(f) \circ ((\varphi_W \circ f)(v_1)) = ((\varphi_W \circ f)(v_1)) \circ f$, for $v_1 \in V$. The left and right sides here are functions from $V$ to $\mathbb{R}$. Evaluate them both at some $v_2 \in V$ and we get

$$\varphi_V(v_1)(v_2) = ((\varphi_W \circ f)(v_1)) \circ f(v_2).$$

But while the left-hand side is constant, the right-hand side can vary as the isomorphism $f$ varies. Take the two dimensional case, and suppose $v_1$ and $v_2$ form a basis for $V$. Then if $f$ is an isomorphism, so is $f'$, the mapping generated by setting $f'(v_1) = f(v_1), f'(v_2) = 2f(v_2)$. But then substituting $f'$ for $f$, the l.h.s. of the equation stays fixed and the value of the r.h.s. doubles. Impossible! So even the proposed revised criterion of naturalness can’t make an isomorphism between $V$ and $V^*$ natural.

(e) Here’s another interim summary. We started off by saying that, intuitively, there’s a ‘natural’, intrinsic, isomorphism between a (finite dimensional) vector space and its double dual, one that depends only on their structures are vector spaces. And we’ve now suggested that this intuitive idea can be formally captured by saying that a certain diagram always commutes, for any choice of vector spaces and structure-preserving maps between them. We’ve also seen that we can’t get analogous always-commuting diagrams for the case of isomorphisms between a vector space and its dual – which chimes with the intuition that (at least the obvious examples) are not ‘natural’ isomorphisms. Promising!

So let’s now try to generalize. First, the claim that the diagram

$$
\begin{array}{ccc}
V & \overset{f}{\longrightarrow} & W \\
\downarrow{\psi_V} & & \downarrow{\psi_W} \\
V^{**} & \overset{DD(f)}{\longleftarrow} & W^{**}
\end{array}
$$

always commutes can be put a slightly different way, using category-speak. For we have in effect been talking about the category we called $\mathbf{V}$ (of finite spaces and the structure-preserving maps between them), and about a functor we can call $DD: \mathbf{V} \to \mathbf{V}$ which
takes a vector space to its double dual, and maps each arrow between vector spaces to its double-dual correlate as explained. So, the claim that the previous diagram commutes is just the claim that, for every arrow \( f : V \to W \) in \( \mathbf{V} \), there are isomorphisms \( \psi_V \) and \( \psi_W \) in \( \mathbf{V} \) such the following commutes:

\[
\begin{array}{ccc}
V & \xrightarrow{f} & W \\
& \downarrow{\psi_V} & \downarrow{\psi_W} \\
DD(V) & \xrightarrow{DD(f)} & DD(W)
\end{array}
\]

NB: the isomorphisms depend only on the domain and codomain of \( f \), and are independent of what \( f \) actually does.

Now, consider the trivial identity functor \( 1 : \mathbf{V} \to \mathbf{V} \) that maps each vector space to itself and each \( \mathbf{V} \)-arrow to itself. Then we can re-express that last claim as follows. For every arrow \( f : V \to W \) in \( \mathbf{V} \), there are isomorphisms \( \psi_V \) and \( \psi_W \) in \( \mathbf{V} \) such this diagram commutes:

\[
\begin{array}{ccc}
1(V) & \xrightarrow{1(f)} & 1(W) \\
& \downarrow{\psi_V} & \downarrow{\psi_W} \\
DD(V) & \xrightarrow{DD(f)} & DD(W)
\end{array}
\]

And what’s nice about this way of putting things is that it transmutes the claim about the availability of a ‘natural’ isomorphism between a space and its double dual into a claim about the naturalness of a certain map between functors on the category of spaces – in this case, between the functors \( 1 \) and \( DD \). And this seems a natural(!) way to go: for talking about functors is our way of category-theoretic way of talking at once both about relations between objects and about relations between arrows.

The two functors in our example so far relate a category to itself. But there is an obvious (and again ‘natural’!) generalization:

**Definition 21.** Let \( \mathcal{C} \) and \( \mathcal{D} \) be categories, let \( \mathcal{C} \xrightarrow{F} \mathcal{D} \xleftarrow{G} \mathcal{C} \) be functors, and let \( \psi \) be a whole family of \( \mathcal{D} \)-isomorphisms \( \psi_C : F(C) \to G(C) \) indexed by the objects \( \mathcal{C} \)-objects. Then \( \psi \) is said to be a natural isomorphism between \( F \) and \( G \) if for every \( f : A \to B \) in \( \mathcal{C} \), the following diagram commutes in \( \mathcal{D} \):

\[
\begin{array}{ccc}
F(A) & \xrightarrow{F(f)} & F(B) \\
& \downarrow{\psi_A} & \downarrow{\psi_B} \\
G(A) & \xrightarrow{G(f)} & G(B)
\end{array}
\]

So our intuitive claim that there is a natural isomorphism between a vector space and its double dual becomes reflected in the claim that there is a natural isomorphism in our official sense between the identity and double-dual functors from the category \( \mathbf{V} \) to itself.

### 5.2 Natural transformations defined

We’ve now introduced one kind of mapping between functors. But just as isomorphisms are just one kind of morphism between objects within categories, so natural isomorphisms
are just one kind of natural morphism between functors. We are next going to introduce the more general idea of a natural transformation (to use the standard term).

We can motivate this as an extension of the ideas we’ve already met. But here’s a slightly different way of thinking about things (following e.g. some remarks of Goldblatt 2006, pp. 198-199). Take a pair of functors $F: \mathcal{C} \to \mathcal{D}$ and $G: \mathcal{C} \to \mathcal{D}$. Each projects the objects and arrows of $\mathcal{C}$ into $\mathcal{D}$ giving, so to speak, two ‘pictures’ or ‘diagrams’ of $\mathcal{C}$ within $\mathcal{D}$. Now, the thought is that we can think of a smooth or natural transformation from $F$ to $G$ as a way of smoothly ‘superimposing’ in $\mathcal{D}$ the picture of $\mathcal{C}$ which results from $F$ onto the picture which results from $G$. This ‘superimposing’ will have to be done, then, using morphisms available in $\mathcal{D}$. So diagrammatically, the situation should in part look as below. On the left we have a couple of objects and an arrow between them in $\mathcal{C}$; and the two functors $F$ and $G$ yield two pictures of this situation in $\mathcal{D}$, one at the top and one at the bottom, with the squiggly arrows then representing local ‘superimposing’ maps between the objects:

$$
\begin{array}{ccc}
A & \xrightarrow{f} & B \\
\downarrow{\alpha_A} & & \downarrow{\alpha_B} \\
G(A) & \xrightarrow{G(f)} & G(B)
\end{array}
$$

And if the ‘pictures’ are to indeed to properly ‘superimpose’, we need that diagram to commute.

So, all this motivates the following key definition (making one just crucial change from the previous definition, in moving from natural isomorphisms to the more general case). We revert to our earlier bracket-free notation:

**Definition 22.** Let $\mathcal{C}$ and $\mathcal{D}$ be categories, let $\mathcal{C} \xrightarrow{F} \mathcal{D}$ be (covariant) functors, and let $\alpha$ be a family of $\mathcal{D}$-arrows $\alpha_C: FC \to GC$ indexed by the $\mathcal{C}$-objects $C$. Then $\alpha$ is said to be a natural transformation between $F$ and $G$, written $\alpha: F \Rightarrow G$, if for every $f: A \to B$ in $\mathcal{C}$, the following diagram commutes in $\mathcal{D}$:

$$
\begin{array}{ccc}
FA & \xrightarrow{F(f)} & FB \\
\downarrow{\alpha_A} & & \downarrow{\alpha_B} \\
GA & \xrightarrow{G(f)} & GB
\end{array}
$$

Arrows $\alpha_A, \alpha_B$ etc. are said to be components of the natural transformation $\alpha$, and we call $\alpha_A$ the component of $\alpha$ at $A$.

Alternative notation to indicate a natural transformation is $\alpha: F \Rightarrow G$ (with a dotted arrow) or simply $\alpha: F \to G$: but, whatever the arrow used, a Greek letter label is the conventionally reliable give-away, signalling a natural transformation. A further bit of notation is very useful. When we have functors $F: \mathcal{C} \to \mathcal{D}$, $G: \mathcal{C} \to \mathcal{D}$, together with a natural transformation $\alpha: F \Rightarrow G$, we can neatly represent the whole situation thus:

$$
\begin{array}{ccc}
\mathcal{C} & \xrightarrow{\alpha} & \mathcal{D} \\
\downarrow{G} & & \\
\end{array}
$$
We will also use the such a diagram just to denote the natural transformation: context will tell.

Note: Defn. 22 defines a natural transformation between covariant functors. Since a contravariant functor $F : C \to D$ is a covariant functor $F : C^{op} \to D$, we get for free a definition of a natural transformation between two contravariant functors. But can we also talk about some kind of naturality of a transformation between a covariant functor $F$ and a contravariant functor $G$ from $C$ to $D$? The diagram in §5.1 (c) gives us a pointer of where we might try to go with this idea. But it turns out not to be important at this stage, so we won’t pause over it.

5.3 Examples of natural transformations

We have already mentioned the natural transformation between a vector space and its double dual. Now for a few more examples:

(1) Given a group $G = (G, \ast)$ [to abuse notation in a familiar way] we can define its opposite $G^{op} = (G, \ast^{op})$, where $a \ast^{op} b = b \ast a$.

We can now define a functor $Op : Grp \to Grp$ which sends an object in the category, i.e. a group $G$, to its opposite $G^{op}$, and sends an arrow $f$ in the category, i.e. a group homomorphism $f : G \to H$, to $f^{op} : G^{op} \to H^{op}$ where $f^{op}(a) = f(a)$. $f^{op}$ so defined is indeed a group homomorphism, since

$$f^{op}(a \ast^{op} b) = f(b \ast a) = f(b) \ast f(a) = f^{op}(a) \ast^{op} f^{op}(b)$$

Claim: there is a natural transformation $\alpha : 1 \Rightarrow Op$ (where 1 is the trivial identity functor in $Grp$).

We need to find a family of arrows $\alpha$ in $Grp$ such that the following diagram always commutes. [Careful! ‘$G$', ‘$H$' here are now groups, not functors, . . .]

$$\begin{array}{ccc}
G & \xrightarrow{f} & H \\
\downarrow{\alpha_G} & & \downarrow{\alpha_H} \\
G^{op} & \xrightarrow{f^{op}} & H^{op}
\end{array}$$

Now, since taking the opposite between groups involves reversing the order of multiplication and taking inverses inside a group in effect does the same, a good bet is to choose $\alpha_G(a) = a^{-1}$ for every $G$. And it’s easy to check that with this choice of components, $\alpha$ is a natural transformation.

Indeed, in this case, we have another natural isomorphism – reflecting the intuition that there a group and its opposite are ‘really the same’.

(2) To get an interesting example of a natural-transformation-which-isn’t-an-isomorphism, recall the idea of the abelianization of a group $G$. Officially, this is the quotient of a group by its commutator subgroup $[G, G]$ (but you can think of it as sending a group to the ‘biggest’ Abelian group $A$ for which there is a surjective homomorphism from $G$ onto $A$). There is a functor $Ab$ which sends a group to its abelianization, and sends an arrow $f : G \to H$ to the arrow $Ab(f) : Ab(G) \to Ab(H)$ in the obvious way.

So we have functors, $Grp \xrightarrow{1} Grp$, and this diagram always commutes,
where $\alpha_G = G/[G,G]$. So we have a natural transformation, but not usually a natural isomorphism, between the functors 1 and $Ab$.

(3) Recall the functor $F : \text{Set} \to \text{Mon}$ that we met in §3.2, which sends a set $X$ to the free monoid on $X$. We also mentioned there the functor $G : \text{Set} \to \text{Ab}$ that sends a set $X$ to the freely generated abelian group with generators in $X$; and by ‘forgetting’ the group structure in order to leave us just a monoid, we can get a derived functor $G : \text{Set} \to \text{Mon}$.

[Aside If we are pressing the metaphor of a functor from $C$ to $D$ giving us a ‘picture’ of $C$ in $D$, we’d have to say that the functors $F$ and $G$ give us much decorated pictures of sets!]

There is then a natural transformation from $F$ to $G$:

$$\begin{array}{ccc}
\text{Free monoid on } X & \xrightarrow{Ff} & \text{Free monoid on } Y \\
\alpha_X & & \alpha_B \\
\text{Abelian group on } X & \xrightarrow{Gf} & \text{Abelian group on } Y
\end{array}$$

where $\alpha_X$ sends a string $x_1 x_2 x_3 \ldots x_n$ to the group element $x_1 + x_2 + x_3 + \ldots + x_n$.

(4) Here are the bare bones of an example for topologists. We met in §3.5 a functor $\pi_1$ which sends pointed topological spaces $\text{Top}^\ast$ to $\text{Grp}$. There’s another functor $H_1$ from $\text{Top}$ to $\text{Ab}$ which sends a space to its first homology group. Making those functors a bit forgetful (about base points and about the fact that homology groups are abelian), we can get a derived pair of functors from $\text{Top}$ to $\text{Grp}$. It is a significant fact of topology that there is a natural transformation between them.

### 5.4 Functor categories

Suppose $F : \mathcal{C} \to \mathcal{D}$. Then quite trivially, the following diagram commutes for every $f : A \to B$ in $\mathcal{C}$:

$$\begin{array}{ccc}
F(A) & \xrightarrow{F(f)} & F(B) \\
1_{FA} & & 1_{FB} \\
F(A) & \xrightarrow{F(f)} & F(B)
\end{array}$$

So we have a natural transformation

$$\begin{array}{ccc}
\mathcal{C} & \xrightarrow{F} & \mathcal{D} \\
1_F & \downarrow & \downarrow 1_F \\
\mathcal{C} & \xrightarrow{F} & \mathcal{D}
\end{array}$$

where the components $(1_F)_A$ of the transformation are the identity morphisms $1_{(FA)}$.

Next, suppose we have functors $F$, $G$ and $H$ between the categories $\mathcal{C}$ and $\mathcal{D}$, with a natural transformation between $F$ and $G$, and another between $G$ and $H$. In a diagram, we have:
Then we can compose the natural transformations to give us a natural transformation between $F$ and $H$: i.e. we get

We simply compose component-wise, i.e. put $(\beta \circ \alpha)_A = \beta_A \circ \alpha_A$, and we are done.

Right: we now have identity transformations and composition of transformations (with composition evidently defined in such a way that it is associative). So, lo and behold, the following definition must be in good order!

**Definition 23.** The functor category from $\mathcal{C}$ to $\mathcal{D}$, denoted $[\mathcal{C}, \mathcal{D}]$ (alternative: $\mathcal{D}^{\mathcal{C}}$) is the category whose objects are the functors $F: \mathcal{C} \to \mathcal{D}$, with the natural transformations between them as arrows.

Let’s have a couple of easy examples, which illustrate how functor categories can be (equivalent to) interesting structures.

1. Recall the category $\mathcal{2}$. Omitting identity arrows, we can diagram this as $\bullet \rightarrow \star$. Now ask: what is the functor category $[\mathcal{2}, \mathcal{C}]$?

   An object in this category is a functor $F: \mathcal{2} \to \mathcal{C}$, where (i) $F_{ob}$ will send $\bullet$ to some $\mathcal{C}$-object $X$ and send $\star$ to an object $Y$, and (ii) $F$ will map the arrow from $\bullet$ to $\star$ to an arrow, $f: X \to Y$.

   And what about the arrows in our new category? A natural transformations between two such functors $\mathcal{2} \xrightarrow{F_1} \mathcal{C} \xleftarrow{F_2} \mathcal{C}$ can have as components any two $\mathcal{C}$-arrows, $j, k$, which makes this a commutative square:

   $\begin{array}{ccc}
   X_1 & \xrightarrow{f_1} & Y_1 \\
   \downarrow{j} & & \downarrow{k} \\
   X_2 & \xrightarrow{f_2} & Y_2
   \end{array}$

   Hence in sum, objects in $[\mathcal{2}, \mathcal{C}]$ correspond exactly to the $\mathcal{C}$-arrows $f: X \to Y$, and the arrows of the new category between two such objects correspond exactly to pairs of $\mathcal{C}$-arrows which make the relevant diagram commute. So $[\mathcal{2}, \mathcal{C}]$ corresponds exactly to the arrow category $\mathcal{C} \to$ (see §4.3).

2. Consider the twosome-category $\mathcal{T}$ which we can diagram as $\bullet \rightarrow \star$ (i.e. the category with just two objects, and two parallel arrows in addition to the identity arrows).

   So now let’s think about the functor category $[\mathcal{T}, \text{Set}]$. An object in this category is a functor $F: \mathcal{T} \to \text{Set}$. So $F_{ob}$ will send $\bullet$ to some set $E_F$, and send $\star$ to a set $V_F$.

   And $F$ will map the two arrows from $\bullet$ to $\star$ to two functions, $f_1^F, f_2^F: E_F \to V_F$.

   However, a pair of sets $E, V$ together with a pair of arrows $f^1, f^2: E \to V$ can be diagrammed as a directed graph, if we think of $E$ as the set of edges, and the
two functions mapping as an edge to its beginning and end vertices. So a functor $F: \mathcal{T} \to \textbf{Set}$ ‘pictures’ $\mathcal{T}$ by a directed graph.

Now, what about natural transformations $\alpha$ between functors $\mathcal{T} \xrightarrow{F} \textbf{Set}$? By definition they make the following diagram commute for any $f$:

\[
\begin{array}{ccc}
E_F & \xrightarrow{f} & V_F \\
\downarrow{\alpha_E} & & \downarrow{\alpha_V} \\
E_G & \xrightarrow{f_G} & V_G
\end{array}
\]

So a natural transformation is a pair of functions which preserves the assignments of vertices to edges. It is thus a homomorphism for directed graphs. Thus, in short, the functor category $[\mathcal{T}, \textbf{Set}]$ is or (in some good sense – we’ll need to return to this) is ‘the same as’ the category of directed graphs.

### 5.5 Horizontal composition of natural transformations

Composing two natural transformations $\mathcal{C} \xrightarrow{\mathcal{F}} \mathcal{D}$ to get $\mathcal{C} \xrightarrow{\mathcal{F}} \mathcal{D}$ naturally enough called vertical composition. But we can also put things together horizontally in various ways.

First, there is so-called whiskering(!) where we combine a functor with a natural transformation between functors to get a new natural transformation. Thus, what happens when we ‘add a whisker’ on the left of a diagram for a natural transformation?

The situation $\mathcal{C} \xrightarrow{F} \mathcal{D} \xrightarrow{\beta} \mathcal{E}$ gives rise to $\mathcal{C} \xrightarrow{J} \mathcal{E}$ where the component of $\beta F$ at $A$ is the component of $\beta$ at $FA$. Why so? Consider the function $Ff: FA \to FB$ in $\mathcal{D}$ (where $f: A \to B$ is in $\mathcal{C}$). Now apply the functors $J$ and $K$, and since $\beta$ is a natural transformation we get the commutative ‘naturality square’

\[
\begin{array}{ccc}
J(FA) & \xrightarrow{J(Ff)} & J(FB) \\
\downarrow{\beta_{FA}} & & \downarrow{\beta_{FB}} \\
K(FA) & \xrightarrow{K(Ff)} & K(FB)
\end{array}
\]

and we can read that as giving a natural transformation between $J \circ F$ and $K \circ F$. Symmetrically, adding a whisker on the right,

the situation $\mathcal{C} \xrightarrow{F} \mathcal{D} \xrightarrow{J} \mathcal{E}$ gives rise to $\mathcal{C} \xrightarrow{J \circ F} \mathcal{E}$ where the component of $J \alpha$ at $X$ is the component of $\alpha$ is $J(\alpha_X)$.

Second, we can horizontally compose two natural transformations:
CHAPTER 5. NATURAL TRANSFORMATIONS

We take \( \mathcal{C} \xrightarrow{\alpha} \mathcal{D} \xrightarrow{\beta} \mathcal{E} \) and get \( \mathcal{C} \xrightarrow{\beta \circ \alpha} \mathcal{E} \).

How do we define \( \beta \circ \alpha \)? Consider any arrow \( f: A \to B \) in \( \mathcal{C} \). Given \( J \) is a functor,

\[
\begin{align*}
FA &\xrightarrow{Ff} FB \\
\alpha_A &\xrightarrow{\alpha_B} \alpha_B \\
GA &\xrightarrow{Gf} GB \\
J(FA) &\xrightarrow{J(Ff)} J(FB) \\
J(FA) &\xrightarrow{J(\alpha_A)} J(FA) \\
J(FG) &\xrightarrow{J(\alpha_B)} J(GB)
\end{align*}
\]

then since \( GF: GA \to GB \) is a map in \( \mathcal{D} \), and \( \beta \) is a natural transformation between \( \mathcal{D} \xrightarrow{\beta} \mathcal{F} \xrightarrow{J} \mathcal{E} \), we have

\[
\begin{align*}
J(GA) &\xrightarrow{J(Gf)} J(GB) \\
\beta_{GA} &\xrightarrow{\beta_{GB}} \beta_{GB} \\
K(GA) &\xrightarrow{K(Gf)} K(GB)
\end{align*}
\]

And since \( J \circ F \) to \( K \circ G \), if we set the component of \( \beta \circ \alpha \) at \( X \) to be \( \beta_{GX} \circ J\alpha_X \).

Three remarks:

1. That definition for \( \beta \circ \alpha \) looks surprisingly asymmetric. But note that instead of first applying \( J \) and then using the fact that \( \beta \) is natural, we could have similarly first used the fact that \( \beta \) is natural and then applied \( K \), showing that we could alternatively have defined the natural transformation as having the components \( K\alpha_X \circ \beta_{FX} \). So we get do at least get two accounts which mirror each other.

2. We can think of whiskering as a special case of the horizontal composition of two natural transformations where one of them is the identity natural transformation. For example \( \mathcal{C} \xrightarrow{\alpha} \mathcal{D} \xrightarrow{1_J} \mathcal{E} \) produces \( \mathcal{C} \xrightarrow{1_J \circ \alpha} \mathcal{E} \), and the component of \( 1_J \circ \alpha \) at \( X \) is an identity composed with \( J\alpha_X \). So this is the same as taking the left-hand natural transformation and simply whiskering with \( J \) on the right.

3. We could now go on to consider the case of horizontally composing a couple of pairs of vertical compositions – and show that it comes to the same if we construe the resulting diagram as the result of vertically composing a couple of horizontal compositions. But we won’t now pause over this, but return to the point if and when we need the construction. (Or see Leinster 2014, p. 38.)

5.6 Natural isomorphisms (again)

Suppose \( [\mathcal{C}, \mathcal{D}] \) is a functor category, then there will be isomorphisms in the category (if only the identity arrows). What are these isomorphisms?

**Theorem 20.** The isomorphisms in the functor category \( [\mathcal{C}, \mathcal{D}] \) are the natural isomorphisms \( \psi: F \Rightarrow G \), where \( \mathcal{C} \xrightarrow{F} \mathcal{D} \).
Proof. Suppose $\psi: F \Rightarrow G$ is a natural isomorphism between $\mathcal{C} \xrightarrow{F} \mathcal{D}$ in the sense of Defn. 21. Then, by definition, all the components $\psi_A, \psi_B$ are isomorphisms, and hence have inverses $\psi_A^{-1}, \psi_B^{-1}$, and these will therefore are available as components of a natural isomorphism $\psi^{-1}: G \Rightarrow F$. And this is readily seen to be an inverse of $\psi$ in the category $[\mathcal{C}, \mathcal{D}]$.

Conversely, suppose $\psi: F \Rightarrow G$ has an inverse $\psi^{-1}$ in the category $[\mathcal{C}, \mathcal{D}]$, i.e. $\psi^{-1} \circ \psi = 1_F$. But composition is defined component-wise, so this requires for each component $\psi_X^{-1} \circ \psi_X = 1_{FX}$, i.e. each component of $\psi$ has an inverse, so is an isomorphism, and hence $\psi$ is a natural isomorphism. \hfill \Box

Next, suppose we have $A$ a $\mathcal{C}$-object, and functors $F,G: \mathcal{C} \to \mathcal{D}$. Then there will be objects $FA$ and $GA$ in category $\mathcal{D}$. And there may perhaps be some isomorphism (some $\mathcal{D}$-arrow with an inverse) between these objects, so that we have $FA \cong GA$. But we are going to most interested in a special case, when the relevant isomorphism belongs to a natural family.

Definition 24. Given functors $F,G: \mathcal{C} \to \mathcal{D}$, we say that $FA \cong GA$ naturally in $A$ (or naturally in $A$ in $\mathcal{C}$) just if $F$ and $G$ are naturally isomorphic.

Note, to have $FA \cong GA$ naturally in $A$ depends on a lot more than what happens at the object $A$. It requires a natural isomorphism between $F$ and $G$, and that’s a condition which depends on how the functors $F$ and $G$ behave right across the category they act on.

It is important to see that there are cases of categories and functors where we have $F;G: \mathcal{C} \to \mathcal{D}$, and for all relevant $A$, $FA \cong GA$, but not naturally in $A$. (A slogan: pointwise isomorphism doesn’t entail natural isomorphism.) Let’s have a couple of examples.

(1) (A toy example, to make the point of principle.) Suppose $\mathcal{C}$ is a category with exactly one object $A$, and two arrows, the identity arrow $1_A$, and another arrow $f$, where $f \circ f = f$. And now consider two functors, the identity functor $1_\mathcal{C}: \mathcal{C} \to \mathcal{C}$, and the functor $F: \mathcal{C} \to \mathcal{C}$ which sends the only object to itself, and sends both arrows to the identity arrow. Then, quite trivially, we have $1_\mathcal{C}(A) = FA(A)$ for the one and only object in $\mathcal{C}$. But there isn’t a natural isomorphism between the functors, because by hypothesis $1_A \neq f$, and the square

$$
\begin{array}{ccc}
F(A) & \xrightarrow{F(f)} & F(A) \\
\downarrow {id_A} & & \downarrow {id_A} \\
1_\mathcal{C}(A) & \xrightarrow{1_\mathcal{C}(f)} & 1_\mathcal{C}(A)
\end{array}
$$

which is none other than

$$
\begin{array}{ccc}
A & \xrightarrow{id_A} & A \\
\downarrow {id_A} & & \downarrow {id_A} \\
A & \xrightarrow{f} & A
\end{array}
$$

cannot commute.

(2) (A standard illustrative example which is more interesting.) We’ll work in the category $\mathcal{F}$ of finite sets and bijections between them.

There is a functor $\text{Sym}: \mathcal{F} \to \mathcal{F}$ which (i) sends a set $A$ in $\mathcal{F}$ to the set of permutations on $A$ (treating permutation functions as sets, this is a finite set), and (ii) sends a bijection $f: A \to B$ in $\mathcal{F}$ to the bijection that sends the permutation $p$ on $A$ to the permutation $f \circ p \circ f^{-1}$ on $B$. Note: if $A$ has $n$ members, there are $n!$ members of the set of permutations on $A$.

There is also a functor $\text{Ord}: \mathcal{F} \to \mathcal{F}$ which (i) sends a set $A$ in $\mathcal{F}$ to the set of linear orderings on $A$ (you can identify an order-relation with a set, so we can
think of this too as a finite set), and (ii) sends a bijection \( f: A \to B \) in \( \mathcal{F} \) to the bijection \( \text{Ord}(f) \) which sends a total order on \( A \) to the total order on \( B \) where \( x <_A y \) iff \( f(x) <_B f(y) \). Again, if \( A \) has \( n \) members, there are also \( n! \) members of the set of linear orderings on \( A \).

Now, for any object \( A \) of \( \mathcal{F} \), \( \text{Sym}(A) \cong \text{Ord}(A) \) (since they are equinumerous finite sets). But there cannot be a natural transformation \( \alpha \) between the functors \( \text{Sym} \) and \( \text{Ord} \). For suppose otherwise, and consider the functors acting on a map \( f: A \to A \). Then the following square would commute:

\[
\begin{array}{ccc}
\text{Sym}(A) & \xrightarrow{\text{Sym}(f)} & \text{Sym}(A) \\
\downarrow{\alpha_A} & & \downarrow{\alpha_A} \\
\text{Ord}(A) & \xrightarrow{\text{Ord}(f)} & \text{Ord}(A)
\end{array}
\]

But consider what happens to the identity permutation \( i \) in \( \text{Sym}(A) \). It gets sent by \( \text{Sym}(f) \) to \( f \circ i \circ f^{-1} \), i.e. to itself. So the square tells us that the ordering \( \alpha_A(i) \) (whatever) it is, gets sent by \( \text{Ord}(f) \) to itself. But in general that won’t be so – suppose \( f \) swaps around elements.
6.1 Full, faithful, and other kinds of functors

Definition 25. A functor \( F : \mathcal{C} \to \mathcal{D} \) is full if given any \( \mathcal{C} \)-objects \( C, C' \), then for any arrow \( g : FC \to FC' \) there is an arrow \( f : C \to C' \) such that \( g = Ff \).

A functor is faithful if given any \( \mathcal{C} \)-objects \( C, C' \), and any pair of parallel arrows \( f, g : C \to C' \), then if \( F(f) = F(g) \), then \( f = g \).

Think of it like this. \( F \) puts a picture of \( \mathcal{C} \) inside \( \mathcal{D} \). It’s a faithful picture if it has enough arrows: different arrows get sent to different arrows. It’s a full picture if it doesn’t have too many arrows: every arrow between pictures of \( \mathcal{C} \)-objects is indeed a picture of a \( \mathcal{C} \)-arrow.

Some examples:

(1) The forgetful functor \( F : \text{Mon} \to \text{Set} \) is faithful, as \( F \) sends a set-function which happens to be a monoid homorphism to itself, so different arrows in \( \text{Mon} \) get sent to different arrows in \( \text{Set} \). But the functor is not full: there will be lots of arrows in \( \text{Set} \) that don’t correspond to a monoid homomorphism.

(2) The ‘pre-orderification’ functor from §3.2 (F5), \( F : \mathcal{C} \to \mathcal{S} \), is full but not faithful unless \( \mathcal{C} \) is already a pre-order category.

(3) Suppose \( \mathcal{M} \) and \( \mathcal{N} \) are the categories that correspond to the monoids \( (M, \cdot) \) and \( (N, \times) \). And let \( f \) be a monoid homomorphism which is surjective but not injective. Then the functor \( F \) corresponding to \( f \) is full but not faithful.

(4) You might be tempted to say that e.g. the ‘total collapse’ functor \( F : \text{Set} \to 1 \) is full but not faithful. But it isn’t full. Take \( C, C' \) in \( \text{Set} \) to be respectively the singleton of the empty set and the empty set. Then there is a trivial identity map in \( 1, 1 : FC \to FC' \); but there is no arrow in \( \text{Set} \) from \( C \) to \( C' \).

(5) An inclusion functor \( F : \mathcal{S} \to \mathcal{C} \) is faithful; if \( \mathcal{S} \) is a full subcategory of \( \mathcal{C} \), then the inclusion map is fully faithful.

We evidently have

Theorem 21. The composition of full functors is full and the composition of faithful functors is faithful.

More significantly, we have:
Theorem 22. A faithful functor $F: \mathcal{C} \to \mathcal{D}$ reflects monomorphisms and epimorphisms. That is to say, if $Ff$ is monic (epic) then $f$ is monic (epic).

Proof. Suppose $f \circ g = f \circ h$, then $F(f \circ g) = F(f \circ h)$, so $Ff \circ Fg = Ff \circ Fh$. Since $Ff$ is monic, $Fg = Fh$. And since $F$ is faithful (so injective), $g = h$. Hence $f$ is monic. Dually for epics. □

Definition 26. A functor $F: \mathcal{C} \to \mathcal{D}$ is essentially surjective on objects (e.s.o.) iff every $\mathcal{D}$-object $D$ is isomorphic in $\mathcal{D}$ to some object $FC$, for some $\mathcal{C}$-object $C$.

Definition 27. A functor $F: \mathcal{C} \to \mathcal{D}$ is conservative iff it is ‘isomorphism-reflecting’, i.e. if $f$ is a $\mathcal{C}$-arrow such that $Ff$ is an isomorphism in $\mathcal{D}$, then $f$ is an isomorphism in $\mathcal{C}$.

Continuing with our examples:

(6) The forgetful functor $F: \text{Grp} \to \text{Set}$ (which, recall, sends a group $G$ to its underlying set, and sends a group homomorphism to its underlying set-function) is conservative – for a group homomorphism is an isomorphism if and only if its underlying function is.

(7) On the other hand, the forgetful functor $F': \text{Top} \to \text{Set}$ – while also faithful but not full – is not conservative. Consider the continuous bijection from the half-open interval $[0, 1)$ to $S^1$. Treated as a topological map $f$, it is not a homeomorphism in $\text{Top}$; however, treated as a set-function $F'f$, it is an isomorphism in $\text{Set}$.

(8) The forgetful functor $F'': \text{Ab} \to \text{Grp}$ is faithful, full but not e.s.o.

(9) On the other hand, the abelianization functor $Ab: \text{Grp} \to \text{Ab}$ is e.s.o. [Exercise for those who know about groups.]

Theorem 23. If a functor is fully faithful it is conservative.

Proof. Suppose $F: \mathcal{C} \to \mathcal{D}$, and $Ff$ is an isomorphism, with $f$ an arrow in $\mathcal{C}$ with source $A$. Then $Ff$ has an inverse, and since $F$ is full, this inverse must be $Fg$ for some arrow $g$ in $\mathcal{C}$. Hence $Fg \circ Ff = 1_{FA}$, so $F(g \circ f) = 1_{FA}$. But $F$ is faithful so can send nothing other than $1_A$ to $1_{FA}$. Hence $g \circ f = 1_A$, and $f$ has a left inverse. Dually, it has a right inverse. So $f$ is an isomorphism. □

6.2 The categories $\text{Set}^*$ and $\text{Set}_*$

A logico-philosophical digression to motivate a claim about the ‘equivalence’ of the categories $\text{Set}^*$ and $\text{Set}_*$

There is no getting away from the central importance of the notion of a partial function from $\mathbb{N}$ to $\mathbb{N}$ in the general theory of computation (thus, the computable function $\varphi_e$ computed by the $e$-th Turing machine in a standard enumeration is typically partial).

How should we treat partial functions in logic? Suppose the partial computable function $\varphi: \mathbb{N} \to \mathbb{N}$ takes no value for $n$ (the algorithm defining $\varphi$ doesn’t terminate gracefully for input $n$). Then the term ‘$\varphi(n)$’ apparently lacks a denotation. But in standard first-order logic, all terms are assumed to denote. Two-valued logic requires every sentence to be determinately either true or false (truth-value gaps aren’t allowed): but a sentence with a non-denoting term, on the standard semantics, will lack a truth-value. What to do?
Historically, there are three options on the market for dealing with empty terms in a regimented logical language, and hence for dealing with the partial functions which give rise to them:

1. **Frege’s proposal.** Treat apparently empty terms as not really empty, by providing a default ‘rogue’ object for them to denote. So (apparently) empty terms are still genuine terms, but with a deviant denotation.

   How does this work for functional terms? Well, given what we naively think of as a partial function \( \varphi : \mathbb{N} \rightarrow \mathbb{N} \), treat this as officially being a total function \( f : \mathbb{N} \rightarrow \mathbb{N} \cup \{ \star \} \), where \( \star \) is any convenient non-number, and where \( f(n) = \varphi(n) \) when \( \varphi \) takes a numerical value, and \( f(n) = \star \) otherwise. Or, loving symmetry as we do, we could perhaps better talk about a function \( f' : \mathbb{N} \cup \{ \star \} \rightarrow \mathbb{N} \cup \{ \star \} \) where \( f'(n) = f(n) \) on numbers, and \( f'(\star) = \star \). If you like, you can think of \( \star \) as coding ‘not numerically defined’.

   So, on this approach there are no genuinely partial functions. All functions are total. And what we really meet in the theory of computation are total functions which are only partially numerical (not all their values are numbers).

2. **Russell’s proposal** Treat apparently empty terms as not really terms at all, by translating them away. Thus, to use his famous example, ‘The present King of France is bald’ (which has an apparently non-denoting term creating a truth-value gap) really says ‘There is at least one present king of France, there is at most one present king of France, and whoever is a present king of France is bald’ (which is plain false because the first conjunct is false). The apparently denoting expression ‘the present King of France’ disappears when we analyse the true logical form of the statement (we are left are left with a predicate ‘(is a) present king of France’ and quantifiers).

   For a mathematical example, ‘The rational square root of two is greater than one’ really says ‘At least one rational is a square root of 2, at most one rational is a square root of 2, and any rational which is a square root of 2 is greater than one’ – which again is plain false.

   The suggestion, then, is that we translate away empty terms, and that we can thereby avoid the truth-value gaps which are anathema to standard two-valued logic.

   But you’ll immediately spot a snag. It isn’t a matter of pure logic that ‘the present King of France’ fails to denote (that’s an empirical fact), nor indeed that ‘the rational square root of two’ fails to denote (that’s a mathematical fact). And in other cases, we may not know whether a term ‘the \( F \)’ denotes or not. But surely the intrinsic logical form of a statement should not depend on extraneous non-logical facts which are perhaps beyond our ken. It seems, then, that if we plan to avoid empty denoting terms of the kind ‘the \( F \)’ by translating them away, we’ll have to adopt a more general policy of translating away all such terms.

   And this is exactly what Russell ended up doing. Roughly, given a sentence \( G(t) \) where \( t \) is a complex term, the general proposal is that we first treat \( t \), if it isn’t already obviously in that form, as being short for a so-called definite description of the kind ‘the \( F \)’, and then we render ‘\( G(\text{the } F) \)’ as really having the logical form ‘There is at least one \( F \), there is at most one \( F \), and whatever is \( F \) is \( G \)’. The original term \( t \) has disappeared from the analysis, via what’s called Russell’s Theory of Descriptions.

   How does this pan out for functional terms? Well, Russell (i) trades in functions for relations, so (ii) a functional term becomes a definite description involving a relation, and then (iii) this gets translated away using the same general policy.
Taking this in stages, the idea (i) that a one-place function ϕ, for example, is just a special sort of binary relation \( R \) (one where if \( Rxy \) and \( Rxy' \) then \( y = y' \)) is now a commonplace. Which is not to say it is right, but let’s not pause over that. The present point is that this treatment applies equally smoothly to total and to partial functions. Then the idea (ii) is that we replace an occurrence of e.g. ‘\( ϕ(n) \)’, for a (perhaps partial) function \( ϕ \), with an occurrence of ‘the \( y \) such that \( Rny \)’: and we are left with something involving new quantifiers and the new relation \( R \), but without the original possibly empty term. So we’ve again eliminated those pesky truth-value gaps. And this time we have kept partial functions alongside the total ones by transmuting them into certain relations.

(3) Logical revisionism Frege’s proposal for avoiding truth-value gaps is simple but artificial. Russell’s proposal for avoiding truth-value gaps is arguably more principled but gets horribly messy with all those introduced quantifiers. So a third option is to bite the bullet and learn to live with truth-value gaps, by adopting a logic which is free from the assumption that all terms denote, by adopting a free logic for short. Then we can take partial functions and the empty terms they give rise to at face value.

However, this policy too has its own costs (logical revisionism has its complications) . . .

But we won’t continue the story any further now. The debate about the best logical treatment of partial functions is the sort of thing that might grip some philosophically-minded logicians but really seems of little more general mathematical interest.

And that’s exactly the point of this section! We’ll set aside the Russelian option for the moment, but from a mathematical point of view there surely isn’t anything much to choose between options (1) and (3). On the small scale, we can think of a world of partial numerical functions \( ϕ: \mathbb{N} \to \mathbb{N} \) (genuinely partial because not everywhere defined), or we can equally think of a corresponding world of total functions \( f: \mathbb{N} \to \mathbb{N} \cup \{\star\} \), with \( \star \notin \mathbb{N} \), and \( f(\star) = \star \). Take your pick!

More generally, on the large scale, we can think of other sets with partial functions between them, or of corresponding pointed sets (sets with a distinguished object as base point) and base-point preserving total functions between them. What’s to choose, apart from familiarity? Mathematically, surely both approaches come to the same.

And so back, at last, to categories! There is a category \( \text{Set}^\star \) whose objects are sets and whose arrows are (possibly) partial functions between them. And there is also the category \( \text{Set}_\star \) of pointed sets which we’ve already met, whose objects are sets with a distinguished base point, and whose arrows are (total) set-functions which preserve base points. So, putting the upshot of our reflections in this section in categorial terms, we get the following attractive

Desideratum An account of what it is for two categories to be equivalent should surely count \( \text{Set}_\star \) and \( \text{Set}^\star \) as being so, for mathematically they come to the same.

And in the next section we will see that a natural first-shot account of what it is for categories to be equivalent fails to meet this desideratum.

6.3 Isomorphic categories

In this section we introduce the obvious definition of isomorphic categories.
Objects are isomorphic if they have an isomorphism between them, i.e. an arrow which has a two-sided inverse. Functors between categories are like arrows between objects. So it is natural to say:

**Definition 28.** Categories $\mathcal{C}$ and $\mathcal{D}$ are isomorphic, in symbols $\mathcal{C} \cong \mathcal{D}$, iff there is an isomorphism $F: \mathcal{C} \rightarrow \mathcal{D}$, where a functor $F$ is an isomorphism iff it has an inverse, i.e. there is a functor $G: \mathcal{D} \rightarrow \mathcal{C}$ where $GF = 1_\mathcal{C}$ and $FG = 1_\mathcal{D}$.

Recall, $1_\mathcal{C}$ is the trivial functor that sends every bit of $\mathcal{C}$’s data to itself (see §3.2, (F6)).

Let’s have a few examples:

1. Take the toy two-object categories with different pairs of objects which we can diagram as

   $\begin{array}{ccc}
   \mathcal{C} & \bullet & \rightarrow & * \\
   \mathcal{D} & a & \rightarrow & b
   \end{array}$

   Plainly they are isomorphic (and indeed there is unique isomorphic functor that sends the first to the second)! If we don’t care about distinguishing copies of structures that are related by a unique isomorphism that we’ll count these as the same in a strong sense. Which to that extent warrants our earlier talk about the category $2$ (e.g. in §1.2, Ex (14)).

2. There is a category $\text{Bool}$ whose objects are Boolean algebras $(B, \neg, \wedge, \vee, 0, 1)$ obeying the familiar constraints, and whose arrows are homomorphisms that preserve algebraic structure. There is also a category $\text{BoolR}$ whose objects are Boolean rings, i.e. rings $(R, +, \times, 0, 1)$ where $x^2 = x$ for all $x \in R$, and whose arrows are ring homomorphisms.

   There is a familiar way of marrying up Boolean algebras with corresponding rings, and vice versa. Thus if we start from $(B, \neg, \wedge, \vee, 0, 1)$, take the same carrier set and distinguished objects, put

   (i) $x \times y \overset{\text{def}}{=} x \wedge y$,
   (ii) $x + y \overset{\text{def}}{=} (x \vee y) \wedge \neg(x \wedge y)$ (exclusive ‘or’),

   then we get a Boolean ring. And if we apply the same process to two algebras $B_1$ and $B_2$, it is elementary to check that this will carry a homomorphism of algebras $f_a: B_1 \rightarrow B_2$ to a corresponding homomorphism of rings $f_r: R_1 \rightarrow R_2$. We can equally easily go from rings to algebras, by putting

   (i) $x \wedge y \overset{\text{def}}{=} x \times y$,
   (ii) $x \vee y \overset{\text{def}}{=} x + y + (x \times y)$
   (iii) $\neg x \overset{\text{def}}{=} 1 + x$.

   Note that going from a algebra to the associated ring and back again takes us back to where we started.

   In summary, without going into more details, we can in this way define a functor $F: \text{Bool} \rightarrow \text{BoolR}$, and a functor $G: \text{BoolR} \rightarrow \text{Bool}$ which are inverses to each other. So, as we’d surely expect, $\text{Bool}$ is isomorphic to $\text{BoolR}$.

3. Revisit the example in §4.4 of the coslice category $1/\text{Top}$. This category has as objects all the arrows $\vec{x}: 1 \rightarrow X$ for any $X \in \text{Top}$. And the arrows from $\vec{x}: 1 \rightarrow X$ to $\vec{y}: 1 \rightarrow Y$ are just the arrows $j: X \rightarrow Y$ from $\text{Top}$ such that $j \circ \vec{x} = \vec{y}$.

   Now we said before that this is in some strong sense ‘the same as’ the category $\text{Top}_* \overset{\text{Top}}{\rightarrow}$ of pointed topological spaces. And indeed the categories are isomorphic. For take the function $F_{ob}$ from objects in $1/\text{Top}$ to objects $\text{Top}_*$ which sends an
object \( \bar{x} : 1 \to X \) to the pointed topological space \((X, x)\), i.e. \(X\)-equipped-with-the-basepoint-\(x\), where \(x\) is the value of the function \( \bar{x} \) for its sole argument. And take \( F_{\text{ruu}} \) to send a \( j : X \to Y \) such that \( j \circ \bar{x} = \bar{y} \) to \( j : (X, x) \to (Y, y) \), which preserves base points. Then it is easy to check that \( F \) is a functor \( F : 1/\text{Top} \to \text{Top} \).

In the other direction, we can define a functor \( G : \text{Top} \to 1/\text{Top} \) which sends \((X, x)\) to the function \( \bar{x} : 1 \to X \) which sends the sole object in 1 to the point \( x \), and sends a basepoint-preserving continuous function from \( X \) to \( Y \) to itself.

And it is immediate that these two functors \( F \) and \( G \) are inverse to each other. Hence, as claimed, \( \text{Top} \cong 1/\text{Top} \).

(4) For two more easy examples, we noted in §4.5 two closely linked pairs (starting from a base category \( \mathcal{C} \) and a \( \mathcal{C} \)-object \( I \):

(a) the comma category \((1_{\mathcal{C}} \downarrow 1_{\mathcal{C}})\) and the arrow category \( \mathcal{C}^\to \),

(b) the comma category we called \((1_{\mathcal{C}} \downarrow I)\) and the slice category \( \mathcal{C}/I \).

It is easily seen that in each case the mentioned categories are isomorphic to each other.

So far, so good then. We’ve given examples of pairs of categories which, intuitively, ‘come to just the same’ and are indeed isomorphic by our definition. However, we also have the following result:

**Theorem 24.** \( \text{Set}_* \) is not isomorphic to \( \text{Set}^* \).

We can remark that there is an obvious functor \( F : \text{Set}_* \to \text{Set}^* \). \( F \) sends a pointed set \((X, x)\) to the set \( X \setminus \{x\} \), and sends a base-point preserving total function \( f : (X, x) \to (Y, y) \) to the partial function \( \varphi : X \setminus \{x\} \to Y \setminus \{y\} \), where \( \varphi(x) = f(x) \) if \( f(x) \in Y \setminus \{y\} \), and is undefined otherwise. But, nice though this is, \( F \) isn’t an isomorphism (it could send distinct \((X, x)\) and \((X', x')\) to the same target object).

Again, there is a whole family of functors from \( \text{Set}^* \) to \( \text{Set}_* \) which take any set \( X \) and add an element not yet in \( X \) to give as an expanded set with the new object as a basepoint. Here’s a way of doing this in a uniform way without making arbitrary choices for each \( X \). Define \( G : \text{Set}^* \to \text{Set}_* \) as sending a \( X \) to the pointed set \( X_* = \text{def} (X \cup \{X\}, X) \), remembering that in standard set theories \( X \notin X \)! And then let \( G \) send a partial function \( \varphi : X \to Y \) to the total basepoint-preserving function \( f : X_* \to Y_* \), where \( f(x) = \varphi(x) \) if \( \varphi(x) \) is defined and \( f(x) = \{Y\} \) otherwise. \( G \) is a natural choice, but isn’t an isomorphism (it isn’t surjective on objects).

Still, those observations don’t yet rule out there being some pair of functors between \( \text{Set}_* \) and \( \text{Set}^* \) which are mutually inverse. But there can’t be any such pair.

**Proof.** A functor which is an isomorphism from \( \text{Set}^* \) to \( \text{Set}_* \) must send objects in \( \text{Set}^* \) one-to-one to objects in \( \text{Set}_* \), and must send isomorphisms to isomorphisms, so should preserve the cardinality of isomorphism classes. But the isomorphism class of the empty set in \( \text{Set}^* \) has just one member, while there is no one-membered isomorphism class in \( \text{Set}_* \). So there can’t be an isomorphism between the categories. \( \square \)

### 6.4 Equivalent categories

(a) The last two sections have together shown that there are categories \( \text{Set}^* \) and \( \text{Set}_* \) which to all intents and purposes are mathematically equivalent but which aren’t isomorphic (according to the obvious definition of isomorphism for categories).
We did, however, note an obvious choice of functors $F: \mathbf{Set} \to \mathbf{Set}^*$ and $G: \mathbf{Set}^* \to \mathbf{Set}$. And while $GF$ isn’t the identity on $\mathbf{Set}$, it does map $\mathbf{Set}$ to itself in a rather natural way (without arbitrary choices).

Reflecting on this case a bit more suggests that the following weakening of the definition of isomorphism between categories:

**Definition 29.** Categories $\mathcal{C}$ and $\mathcal{D}$ are equivalent iff there are functors $F: \mathcal{C} \to \mathcal{D}$ and $G: \mathcal{D} \to \mathcal{C}$, together with a pair of natural isomorphisms $\alpha: 1_\mathcal{C} \Rightarrow GF$ and $\beta: FG \Rightarrow 1_\mathcal{D}$. If $\mathcal{C}$ is equivalent to $\mathcal{D}$, we write $\mathcal{C} \simeq \mathcal{D}$.

And yes, we can now give a direct proof that $\mathbf{Set}$ and $\mathbf{Set}^*$ are indeed equivalent in this way. But in fact we won’t do that. Rather we’ll first prove a result which yields an alternative characterization of equivalence which is easier to apply.

**Theorem 25.** Assuming a sufficiently strong choice principle, a functor $F: \mathcal{C} \to \mathcal{D}$ is part of an equivalence between $\mathcal{C}$ and $\mathcal{D}$ iff $F$ is faithful, full and e.s.o.

**Proof.** First suppose $F$ is part of an equivalence between $\mathcal{C}$ and $\mathcal{D}$, so that there is a functor $G: \mathcal{D} \to \mathcal{C}$, where $GF \cong 1_\mathcal{C}$ and $FG \cong 1_\mathcal{D}$. Then:

(i) Given an arrows $f,g: A \to B$ in $\mathcal{C}$, then by hypothesis, the following square commutes for $f$ ($\alpha$ is the required natural isomorphism between the identity functor and the composite $GF$),

\[
\begin{array}{ccc}
A & \xrightarrow{f} & B \\
\downarrow{\alpha_A} & & \downarrow{\alpha_B} \\
GFA & \xrightarrow{GFf} & GB
\end{array}
\]

and hence $\alpha_B^{-1} \circ GFf \circ \alpha_A = f$. And of course $\alpha_B^{-1} \circ GFg \circ \alpha_A = g$. It immediately follows that if $Ff = Fg$ then $f = g$, i.e. $F$ is faithful. A companion argument, interchanging the roles of $\mathcal{C}$ and $\mathcal{D}$, shows that $G$ too is faithful.

(ii) Suppose we are given an arrow $h: FA \to FB$, then put $f = \alpha_B^{-1} \circ Gh \circ \alpha_A$. But we know that $f = \alpha_B^{-1} \circ GFf \circ \alpha_A$. So it follows that $GFf = Gh$, and since $G$ is faithful, $h = Ff$. So every such $h$ in $\mathcal{D}$ is the image under $F$ of some arrow in $\mathcal{C}$. So $F$ is full.

(iii) Recall, $F: \mathcal{C} \to \mathcal{D}$ is e.s.o. iff for any $D \in \mathcal{D}$ we can find some isomorphic object $FC$, for $C \in \mathcal{C}$. But we know that our natural isomorphism between $1_\mathcal{D}$ and $FG$ means that that there is an isomorphism from $D$ to $FGD$, so putting $C = GD$ gives the desired result that $F$ is e.s.o.

Now for the argument in the other direction. Suppose, then, that $F: \mathcal{C} \to \mathcal{D}$ is faithful, full and e.s.o. We need to construct (i) a corresponding functor $G: \mathcal{D} \to \mathcal{C}$, and then a pair of natural isomorphisms (ii) $\beta: FG \Rightarrow 1_\mathcal{D}$ and (iii) $\alpha: 1_\mathcal{C} \Rightarrow GF$:

(i) By hypothesis, $F$ is e.s.o., so by definition every object $D \in \mathcal{D}$ is isomorphic in $\mathcal{D}$ to some object $FC$, for $C \in \mathcal{C}$. Hence – and here we are invoking an appropriate choice principle – for a given $D \in \mathcal{D}$, we can choose a pair $(C, \beta_D)$, with $C \in \mathcal{C}$ and $\beta_D: FC \to D$ an isomorphism in $\mathcal{D}$. Now define $G_{arw}$ as sending an object $D \in \mathcal{D}$ to the chosen $C \in \mathcal{C}$ (so $FC = D$, and $\beta_D: FGD \to D$).

To get a functor, we need the component $G_{arw}$ to act suitably on an arrow $g: D \to E$. Now, note

$$FGD \xrightarrow{\beta_D} D \xrightarrow{g} E \xrightarrow{\beta_E^{-1}} FGE$$
and since $F$ is full and faithful, there must be some unique $f: GD \to GE$ which $F$ sends to the composite $\beta_{F}^{-1} \circ g \circ \beta_{D}$. Put $G_{\text{arw}}g = f$.

Claim: $G$, with components $G_{\text{ob}}, G_{\text{arw}}$, is indeed a functor. We need to show that $G$ (a) preserves identities and (b) respects composition:

For (a), note that $G1_{D} = e$ where $e$ is the unique arrow from $GD$ to $GD$ such that $Fe = \beta_{D}^{-1} \circ 1_{D} \circ \beta_{D} = 1_{D}$. So $e = 1_{GD}$.

For (b) we need to show that, given $\mathcal{D}$-arrows $g: D \to E$ and $h: E \to F$, $G(h \circ g) = Gh \circ Gg$. But note that

$$FG(h \circ g) = \beta_{F}^{-1} \circ h \circ g \circ \beta_{D} = (\beta_{F}^{-1} \circ h \circ \beta_{E}) \circ (\beta_{E}^{-1} \circ g \circ \beta_{D}) = FG(h) \circ FG(g) = F(G(h) \circ G(g))$$

Hence, since $FG(h \circ g) = F(G(h) \circ G(g))$ and $F$ is faithful, $G(h \circ g) = G(h) \circ G(g)$, so $G$ is indeed a functor. Phew!

(ii) By construction, $\beta$ is natural isomorphism from $FG$ to $1_{\mathcal{D}}$.

(iii) Note next that we have an isomorphism $\beta_{F,A}^{-1}: FA \to FGFA$. As $F$ is full and faithful, $\beta_{F,A}^{-1} = F(\alpha_{A})$ for some unique $\alpha_{A}: A \to GFA$. Since $F$ is fully faithful it is conservative, i.e. reflects isomorphisms (by Theorem 23), hence $\alpha_{A}$ is also an isomorphism. Also, the naturality diagram

$$\begin{array}{ccc}
A & \xrightarrow{f} & B \\
\downarrow{\alpha_{A}} & & \downarrow{\alpha_{B}} \\
GFA & \xrightarrow{Gf} & GFB
\end{array}$$

always commutes for any arrow $f: A \to B$ in $\mathcal{D}$. Why? Because

$$F(\alpha_{B} \circ f) = F\alpha_{B} \circ Ff = \beta_{F}^{-1} \circ Ff = FGf \circ \beta_{F,A}^{-1} = FGf \circ F\alpha_{A} = F(Gf \circ F\alpha_{A})$$

relying on the naturality of $\beta^{-1}$. But if $F(\alpha_{B} \circ f) = (GFf \circ F\alpha_{A})$ then since $F$ is faithful, $\alpha_{B} \circ f = GFf \circ F\alpha_{A}$. Hence the $\alpha_{A}$ are the components of our desired natural isomorphism $\alpha: 1_{\mathcal{D}} \Rightarrow GF$.

So we are done! $\square$

Our theorem enables us now to quickly show

**Theorem 26.** $\text{Set}^{*} \simeq \text{Set}_{*}$

*Proof.* Choose the particular functor $G: \text{Set}^{*} \to \text{Set}_{*}$, which sends a set $X$ to a set $X_{*} = \text{def} X \cup \{X\}$ with basepoint $X$, and sends a partial function $f: X \to Y$ to the total function $f_{*}: X_{*} \to Y_{*}$, where for $f_{*}(x) = f(x)$ if $f(x)$ is defined and $f_{*}(x) = Y$ otherwise.

$G$ is faithful, as it is easily checked that it sends distinct functions to distinct functions. And it is equally easy to check that $G$ is full, i.e. given any basepoint preserving function between sets $X_{*}$ and $Y_{*}$, there is a partial function $f$ which $G$ sends to it.

But $G$ is essentially surjective on objects. For every pointed set in $\text{Set}_{*}$ – i.e. every set which can be thought of as the union of a set $X$ with $\{x\}$ where $x$ is an additional basepoint element (not in $X$) – is isomorphic in $\text{Set}_{*}$ to the set $X \cup \{X\}$ with $X$ as basepoint. Hence $G$ is part of an equivalence between $\text{Set}^{*}$ and $\text{Set}_{*}$. $\square$
(b) Now for another example. Recall \textsf{FinSet} is the category of finite sets and functions between them. Let \textsf{Finordn} be its full subcategory containing the empty set and all sets of the form \{0, 1, 2, \ldots, n-1\} and all functions between them. It doesn’t really matter for present purposes how you think of the natural numbers; but to fix ideas, think of them set-theoretically as von Neumann ordinals, so the objects of \textsf{Finordn} are then the finite ordinals – hence the label for the category. We then have:

**Theorem 27.** \textsf{Finordn} \simeq \textsf{Finset}  

*Proof.* \textsf{Finordn} is a full subcategory of \textsf{Finset}, so the inclusion functor \( F \) is fully faithful. \( F \) is also essentially surjective on objects: for take any object in \textsf{Finset}, which is some \( n \)-membered set: that is in bijective correspondence (and hence isomorphic in \textsf{Finset}) with the finite ordinal \( n \). Hence \( F \) is part of an equivalence, and \textsf{Finordn} \simeq \textsf{Finset}.  

How should we regard this last result? We saw that defining equivalence of categories in terms of isomorphism would be too strong, as it rules out our treating \textsf{Set} and \textsf{Set} \star as in effect equivalent. But now we’ve seen that defining equivalence of categories as in Defn. 6.4 makes the seemingly very sparse category \textsf{Finordn} equivalent to the seemingly much more abundant \textsf{Finset}. Is that a strike against the definition of equivalence, showing it to be too weak?  

It might help to think of a toy example. Consider the two categories which we can diagram respectively as

\[
\bullet \xrightarrow{\sim} \quad \circ \xleftarrow{\sim} \ast \xrightarrow{\sim}
\]

On the left, we have the category 1; on the right we have a two-object category 2! with arrows in both directions between the objects (in addition, of course, to the required identity arrows). These two categories are also equivalent. For the obvious inclusion functor 1 \hookrightarrow 2! is full and faithful, and it is trivially essentially surjective on objects as each object in the two-object category is isomorphic to the other.  

What this toy example highlights is that our equivalence criterion counts categories as amounting to the same when (putting it very roughly) one is just the same as the other padded out with new objects and just enough arrows to make the new objects isomorphic to some old objects.  

But on reflection that’s fine. Taking a little bit of the mathematical world and adding indiscernable copies of structures it already contains won’t, for many (most? nearly all?) purposes, give us a real enrichment. Therefore a criterion of equivalence of categories-as-mathematical-universes that doesn’t care about surplus isomorphic copies is what we typically need. Hence the results that 1 \simeq 2! and \textsf{Finord} \simeq \textsf{Finset} are arguably welcome features, not bugs, of our account of equivalence.

### 6.5 Skeleton categories

Still, even if we treat categories as equivalent when one just has additional objects isomorphic to those already there in the other, shouldn’t the usual sort of concern for Bauhaus elegance and lack of redundancy lead us to privilege categories which are as skeletal as possible? Let’s say:

**Definition 30.** The category \( \mathcal{J} \) is a skeleton of the category \( \mathcal{C} \) if \( \mathcal{J} \) is a full subcategory of \( \mathcal{C} \) which contains exactly one object from each class of isomorphic objects of \( \mathcal{C} \). A category is skeletal if it is a skeleton of some category.
For a toy example, suppose \( \mathcal{C} \) is a category arising from a pre-order (as in §1.2, Ex. (8)). Then any skeleton of \( \mathcal{C} \) will be a poset category. (Check that!)

**Theorem 28.** If \( \mathcal{I} \) is a skeleton of the category \( \mathcal{C} \) then \( \mathcal{I} \cong \mathcal{C} \).

*Proof.* The inclusion functor \( \mathcal{I} \hookrightarrow \mathcal{C} \) is fully faithful, and by the definition of \( \mathcal{I} \) is essentially surjective on objects. So we can apply Theorem 25.

So how about this for a programme: Take the usual universe of categories. But now slim it down by taking skeletons. Then work with these. And we can now forget bloated non-skeletal categories (and forget too about the notion of equivalence and revert to using the simpler notion of isomorphism, because equivalent skeletal categories are in fact isomorphic). What’s not to like?

The trouble is that hardly any categories that occur in the wild (so to speak) are skeletal. And slimming down to be done by appeal to an axiom of choice. Indeed the following statements are each equivalent to a version of the axiom of choice:

1. Any category has a skeleton.
2. A category is equivalent to any of its skeletons
3. Any two skeletons of a given category are isomorphic.

The choice of a skeleton is usually quite artificial – there typically won’t be a canonical choice. So any gain in simplicity from concentrating on skeletal categories would be bought at the cost of having to adopt ‘unnatural’, non-canonical, choices of skeletons. Given that category theory is supposed to be all about natural patterns already occurring in mathematics, this perhaps isn’t going to be such a good trade-off after all.
7.1 Categories of categories

(a) We noted in §3.2 that there is an identity functor $1_C : C \to C$ for every category. We also noted in §4.1 that composable functors $F : C \to D$ and $G : D \to E$ (with the target of the first the source of the second) always do have a composite $G \circ F$, and moreover composition of functors is associative.

So the following definition looks to be in good order:

Definition 31. Suppose $\mathcal{X}$ comprises two sorts of data:

1. Objects: some categories, $C, D, E, \ldots$,
2. Arrows: some functors, $F, G, H, \ldots$, between those categories,

where the arrows include the identity functor on each category, and include $G \circ F$ for each included composable pair $F$ and $G$: then $\mathcal{X}$ is a category of categories.

Two trivial examples. First, take the category which has the single category $C$ as its sole object and the identity functor $1_C$ as its sole arrow. Less boringly, consider the category whose objects are the finite categories, and whose arrows are all the functors between finite categories.

So there certainly seem to be some categories of categories. But there are limitations. Suppose we next say:

Definition 32. A category is normal iff it is not one of its own objects.

The categories which we have met in previous chapters have all been normal, and hence all the functors we have met have been normal functors too, i.e. functors between normal categories. Now ask: can all the normal categories be gathered together as the objects of one really big category whose arrows are some or all the normal functors between finite categories?

So there certainly seem to be some categories of categories. But there are limitations. Suppose we next say:

Theorem 29. There is no category of normal categories.

Proof. Suppose there is a category $\mathcal{N}$ whose objects are all the normal categories. Now ask, is $\mathcal{N}$ normal? If it is, then it is one of the objects of $\mathcal{N}$, so $\mathcal{N}$ is non-normal. So $\mathcal{N}$ must be non-normal. But then it is not one of the objects of $\mathcal{N}$, so $\mathcal{N}$ is normal after all. Contradiction.
That argument just re-runs, in our new environment, the very familiar argument from Russell’s Paradox to the conclusion that there is no set of all the normal sets (where a set is normal if it is not a member of itself). It is worth remarking, however, that the argument is not especially to do with sets. For at its core is a simple, purely logical, observation. Take any two-place relation \( R \) defined over some objects; then there can be no object \( r \) which is related by \( R \) to all and only those objects which are not \( R \)-related to themselves. In other words, it is a simple first-order logical theorem that \( \neg \exists r \forall x (Rxr \leftrightarrow \neg Rxr) \). Russell’s original argument applies this elementary logical theorem to the set-theoretic relation \( R_1 \), ‘... is a set which is a member of the set ...’, to show that there is no set of all normal (i.e. non-self-membered) sets. Our argument above applies the same logical theorem to the category-theoretic relation \( R_2 \), ‘... is a category which is an object of the category ...’, to show that there is no category of all normal categories. There is no lurking presumption here that categories are sets.

(b) Russell’s original argument that there is no set of all normal sets is usually taken to entail that, a fortiori, there is no universal set, no set of all sets. The reasoning being that if there were a universal set then we should be able carve out of it (via a separation principle) a subset containing just those sets which are normal, which we know can’t be done.

To keep ourselves honest, however, we should note that this further argument can be, and has been, resisted. There are cogent set theories on the market which work by restricting separation and blocking the idea that, if there is a universal set of all sets, we should be able to carve out from it a set of all normal sets: see Forster (1995). But we can’t explore deviant set theories here: henceforth in these notes we’ll have to assume our set theory is a standard one – e.g. Zermelo-Fraenkel set theory with or without ur-elements.

Similarly, our Russelian argument that there is no category of all normal categories is naturally taken to entail that there is no universal category in the obvious sense:

**Definition 33.** A category \( \mathcal{U} \) is universal if it is a category of categories such that every category is an object of \( \mathcal{U} \).

**Theorem 30?** There is no universal category.

The argument goes: suppose a universal category \( \mathcal{U} \) exists. Then we could take the full subcategory whose objects are just the normal categories, to get a category of all normal categories. But we have shown there can be no such category.

Can this argument be resisted? Well, how could we justify saying that even if there is a category of all categories, it needn’t have a full subcategory of normal categories? Perhaps some themes in debates about set theories with a universal set could be carried over to this case. But again, it would take us too far away from mainstream concerns in category theory to try to explore this here.

### 7.2 ‘Small’ and ‘locally small’ categories

(a) When we talk about sets, then, we are assuming we are working in a reasonably standard set theory according to which there can be entities (e.g. the sets themselves) which are too many to form a set. Likewise, the normal categories are too many to form a category. Still, there can it seems be some pretty capacious categories of categories.

Let’s say

**Definition 34.** A category \( \mathcal{C} \) is small iff it has only a ‘set’s worth’ of arrows – i.e. the arrows of \( \mathcal{C} \) can be put into one-one correspondence with the members of some set. We
use $\mathbf{Cat}$ to denote the category, assuming there is one, whose objects are small categories and whose arrows are the functors between them.

A category $\mathcal{C}$ is locally small iff for every pair of $\mathcal{C}$-objects $C, D$ there is only a ‘set’s worth’ of arrows from $C$ to $D$, i.e. those arrows can be put into one-one correspondence with the members of some set. We use $\mathbf{CAT}$ to denote the category, assuming there is one, whose objects are locally small categories and whose arrows are the functors between them.

Some comments and examples:

(1) The terms ‘small’ and ‘locally small’ are standard.

(2) It would be more usual to say that in a small category the arrows themselves form a set. But if our favoured set theory is a theory like ZFC where sets only have other sets as members, that would presuppose that arrows are themselves sets, and we might not want to make that assumption just yet. So, for smallness, let’s only require that the arrows aren’t too many to be represented by a set. Similarly for local smallness.

(3) Since for every object in $\mathcal{C}$ there is at least one arrow, namely the identity arrow on $\mathcal{C}$, if the objects of $\mathcal{C}$ are too many to be mapped into a set, then $\mathcal{C}$ has too many arrows to be small. Contrapositing, if $\mathcal{C}$ is small, not only its arrows but its objects can be put into one-one correspondence with the members of some set.

(4) Among our examples in §1.2, tiny finite categories like $1$ and $2$ are of course small. But so too are the categories corresponding to an infinite but set-sized monoid or to an infinite pre-ordered set. Categories such as $\mathbf{Set}$ or $\mathbf{Mon}$, however, have too many objects and arrows to be small.

(5) A discrete category only has as many arrows as objects, so the discrete category on any set is small. But that means, the category $\mathbf{Cat}$ of small categories has at least as many objects as there are sets, and hence is itself determinately not small. So no Russellian paradox arises for $\mathbf{Cat}$ as it did for the putative category of normal categories.

(6) But while not small, categories such as $\mathbf{Set}$ or $\mathbf{Mon}$ and indeed all our other examples so far are locally small (some authors even build local smallness into the very definition of a category). In $\mathbf{Set}$, for example, the arrows between objects $C$ to $D$ are members of a certain subset of the powerset of $C \times D$: which makes $\mathbf{Set}$ locally small.

(7) Take a one-element category $1$, which is certainly locally small. Then a functor from $1$ to $\mathbf{Set}$ will just map the object of $1$ to some particular set: and there will be as many distinct functors $F: 1 \to \mathbf{Set}$ as there are sets. In other words, arrows from $1$ to $\mathbf{Set}$ in $\mathbf{CAT}$ are too many to be mapped one-to-one to a set. Hence $\mathbf{CAT}$ is determinately not locally small. So again no Russellian paradox arises for $\mathbf{CAT}$.

Since no Russellian paradox arises for either $\mathbf{Cat}$ or $\mathbf{CAT}$, it seems that we can indeed countenance these categories, and will henceforth assume that they do exist.

(b) Note however that we have the following negative result:

**Theorem 31.** Smallness is not preserved by categorial equivalence.

In other words, we can have $\mathcal{C}$ a small category, $\mathcal{C} \simeq \mathcal{D}$, yet $\mathcal{D}$ not small. This is a simple corollary of our observation in §6.4 that if we take a category, inflate it by adding lots of objects and just enough arrows to ensure that these objects are isomorphic to the
original objects, then the augmented category is equivalent to the one we started with. For an extreme example, start with the one-object category \( 1 \), i.e. \( \bullet \xrightarrow{\approx} \) (that’s small)! Now add as new objects every set, and as new arrows an identity arrow for each set, and also for every set \( X \) a pair of arrows \( \bullet \xrightarrow{\approx} X \) which composed to give identities. Then we get a new pumped-up category \( 1^+ \) (which is certainly not small). But \( 1^+ \simeq 1 \).

Categorial notions that are not invariant under equivalence are sometimes said to be ‘evil’. That makes smallness an evil notion! If you fuss about these things, you can highlight a neighbouring notion which evidently is virtuous:

\textbf{Definition 35.} A category is \textit{essentially small} if it is equivalent to category with a set’s worth of arrows.

But we aren’t going to fuss here.

There is, by the way, a companion positive result

\textbf{Theorem 32.} Local smallness is preserved by categorial equivalence.

\textit{Proof.} An equivalence \( \mathcal{C} \xrightarrow{F} \mathcal{D} \xleftarrow{G} \) requires \( F \) and \( G \) to be full and faithful functors. So in particular, for any \( \mathcal{D} \)-objects \( D, D' \), there are the same number of arrows between them as between the \( \mathcal{C} \)-objects \( GD, GD' \). So that ensures that if \( \mathcal{C} \) has only a set’s worth of arrows between any pair of objects, the same goes of \( \mathcal{D} \). \( \square \)

7.3 Classes, virtual and real

We are going need to make use in the next section and onwards of a notion of a class of entities, where this notion is to be distinguished from that of a set. This section is therefore an aside which does some preliminary explaining: the points will probably be familiar from an introductory set theory course.

Recall the usual story about the cumulative hierarchy of sets. We start at level zero with some objects – or indeed, in pure ZFC set theory, we start with nothing at all. Then at each succeeding level, we add to whatever entities we have so far accumulated all the possible sets which can be formed with those entities as members. Every set, the story goes, is formed at some level. And we keep on going: at no level in the set-building process completed. We can always go up a level to form new sets – however many sets we’ve formed so far, we can always collect them together into new sets at the next level. Which means that the whole universe of sets never becomes available at any level to itself be collected into a set. Which is why there is no set of all sets.

However, even if there is no set of all sets (given that we are indeed conceiving of sets as living in the cumulative hierarchy), can’t we still think of the universe of sets as in some sense forming a super-collection, albeit one too big to be a set, call it – to use the standard term – a (proper) \textit{class}? But what does that really mean? There is a spectrum of positions available here, taking classes with increasing seriousness:

\textit{Virtual classes} We begin by noting that in many contexts class-talk involves no more than a convenient logical fiction; i.e. the apparent reference to classes can simply be parsed away. In such a usage, to say that \( a \) belongs to the class of \( F \)s is to say no more than that \( a \) is one of the \( F \)s, which it turn says no more than that \( a \) is an \( F \); to say that \( a \) belongs to the intersection of the class of \( F \)s with the class of \( G \)s is to say no more than that \( a \) is both an \( F \) and a \( G \); to say that the class of \( F \)s is identical to the class of \( G \)s is to say no more than that for all \( x \), \( x \) is an \( F \) iff \( x \) is a \( G \). And so it goes.

We can formalize such talk of \textit{virtual classes}, to use the standard term introduced by Quine (1963, §2). Extend a standard first-order theory \( T \) with what look like set-abstractions
of the form ‘\{z \mid C(z)\}’ and with what looks like a membership sign ‘∈’ (perhaps we ought to use different symbolism here – but mostly authors recycle familiar set notation).

We then explain the role of these class-abtraits by stipulating that \(x \in \{z \mid C(z)\}\) is to be understood as a whole, as just equivalent to \(C(x)\), for all contexts \(C(\cdot)\) of the original unaugmented language (avoiding clash of variables); the first formula is just long-winded for the second formula. We can then define the further abstract \(\alpha \cap \beta\) where \(\alpha\) and \(\beta\) are class abstracts by stipulating that \(x \in \alpha \cap \beta\) iff \(x \in \alpha \land x \in \beta\); \(\emptyset\) is introduced as shorthand for the class abstract \(\{z \mid z \neq z\}\); the wff \(\alpha \subseteq \beta\) is stipulated to be equivalent to \(\forall x(x \in \alpha \rightarrow x \in \beta)\); we define \(\alpha = \beta\) by stipulating it is equivalent to \(\alpha \subseteq \beta \land \beta \subseteq \alpha\). And so it goes. Crucially, the class-abstracts are not genuine terms which can be unrestrictedly substituted in arbitrary contexts: they can only appear in (shorthand for) expressions of the form \(x \in \{z \mid C(z)\}\). And note that class ‘membership’ always goes that way around. No sense is ascribed to a formula \(\{z \mid C(z)\} \in x\): indeed, that is deemed ill-formed.

We can now add ‘substitutional’ quantification over our virtual classes: so if \(C\) is a class variable, ‘\(\forall C(\ldots z \in C \ldots)\)’ is just treated as a compendious way of asserting every instance of a schema of the corresponding form ‘\(\ldots C(z) \ldots\)’.

In this way, we get something of the convenience of talking about classes as if they are set-like entities – indeed we can arguably reconstruct all or nearly all that elementary talk of sets or classes which was fashionable in ‘new math’ – but without taking on any ontological commitment to new entities over and above those postulated by our original theory \(T\). “We can explain the sham [classes] away as a mere manner of speaking, by contextual definition, when the ontological reckoning comes” (Quine, 1970, p. 69).

Now, we can in particular add such eliminable talk of classes to a given theory of sets like ZFC. Indeed, this is exactly what happens in some classic texts. Of course, in this context we already have genuine set abstracts \(\{z \mid C(z)\}\) to play with: but of course not every context \(C\) gives us a legitimate set abstract (e.g. there is no Russellian set \(\{z \mid z /\in z\}\)). The proposal is, in headline terms, that we can make up for those missing set abstracts by now introducing proper class abstracts for talking in the singular about objects which are too many to be a set (these class abstracts are standardly made to look just the same as set abstracts – so care is needed here!).

Thus there is no set of sets or set of ordinals in ZFC: but we can introduce virtual classes of sets or ordinals as does Kunen (1980, p. 24), putting

\[
V = \{x \mid x = x\} \\
ON = \{x \mid x \text{ is an ordinal}\}
\]

where ‘\(x\) is an ordinal’ abbreviates a suitable wff of the language of ZFC. Then Kunen adds:

Formally, proper classes do not exist, and expressions involving them must be thought of as abbreviations for expressions not involving them. Thus, \(x \in ON\) abbreviates the formula expressing that \(x\) is an ordinal, and \(ON = V\) abbreviates the (false) sentence (abbreviated by)

\[
\forall x(x \text{ is an ordinal} \leftrightarrow x = x).
\]

There is, in fact, no formal distinction between a formula and a class; the distinction is only in the informal presentation.

And what does adding these virtual classes buy us? Convenience. Kunen writes
The abbreviations obtained with the class become very useful when discussing general properties of classes. Asserting that a statement is true of all classes is equivalent to asserting a theorem schema.

His example is the principle of transfinite induction for classes of ordinals, in the form ‘For any class \( C \), if \( C \subseteq \text{ON} \) and \( C \neq \emptyset \), then \( C \) has a least member’. This helpfully wraps up into a single claim the effect of asserting every instance of the schema

\[
(\forall x(Cx \rightarrow x \text{ is an ordinal}) \land \exists xCx) \rightarrow \exists x(Cx \land \forall y(Cy \rightarrow y \geq x)),
\]

with the order relation appropriately defined, and where \( C \) involves only expressions of the unaugmented language of ZFC. The class talk here earns its keep by being brisk and perspicuous. But as Kunen reiterates (p. 26), while

it is rarely necessary in mathematical arguments about classes to translate away the classes . . . it is, however, important to know that this can be done in principle.

And since we can translate away the class talk from the theory ZFC\(_v\) – i.e. ZFC with added virtual classes – this theory can prove no more about sets than was already provable in good old ZFC.

VNB: almost virtual classes There are, however, set theories which introduce talk of classes seemingly to handle pluralities of objects which are too many to form a set and where such talk is not straightforwardly eliminable. But it is quite a subtle question – and certainly not one we can explore in depth here – how far such theories are really committed to classes as genuine entities over and above sets (for a classic but still very useful discussion, see Fraenkel et al. 1973, Ch. 2 §7, written in fact by Azriel Levy).

Consider, for example, the set theory VNB (due to von Neumann and Bernays, a version of which has been popularized by the very widely used textbook Mendelson 1964 which has gone through numerous editions). This theory is most perspicuously presented as a two-sorted theory with variables for both sets and classes, and which allows quantification over both. VNB’s comprehension principle for classes allows us to form class abstracts \( \{z \mid C(z)\} \) for the class of sets satisfying \( C \), where \( C \) may now contain free class variables as parameters (that wasn’t the case in ZFC\(_v\), the theory with Quinean merely virtual classes). However, and this is crucial, we still don’t allow such conditions \( C \) to contain bound class variables.

Now, as Levy points out in Fraenkel et al. (1973, p. 124))

the conditions \( C(z) \) that we allow in [the comprehension principle for classes] differ essentially from pure conditions [i.e. conditions of the original language of ZFC] only in that they may also contain class parameters; once these parameters are given values which are classes determined by pure conditions [the only way in end to make use of the parameters!], the condition \( C(z) \) itself becomes equivalent to a pure condition.

Yet Kunen’s theory with virtual classes already allows classes determined by pure conditions: so we might reasonably expect that VNB comes to little more than ZFC\(_v\). And indeed, as with ZFC\(_v\), VNB too is conservative over ZFC: i.e. VNB can prove no more about sets than was already provable in ZFC.

There are differences however between VNB and ZFC with or without virtual classes. First, unlike either version of ZFC, VNB is finitely axiomatizable. And second, perhaps surprisingly, VNB exhibits ‘speed-up’: that is to say, there are infinitely many proofs of
ZFC theorems which are exponentially shorter in VNB than they are in ZFC or ZFC\(_v\). But although those are not just gains in convenience, arguably these features still don’t make VNB’s use of classes more than a helpful technical dodge, and so don’t really seem to count against Levy’s verdict (Fraenkel et al., 1973, p. 131)):

[O]ne should regard ZF and VNB as essentially the same theory, and the differences between those theories as mere technical matters.

Or here is Potter (2004, p. 307, with omissions) taking a similar line:

[It] was Hilbert who conceived the general project of instrumentally justifying our use of an extension of an already accepted theory by proving finitistically that it is conservative. . . .

According to Quine’s dictum that to be is to be the value of a variable, by adopting the quantified theory of classes we express a commitment to an ontology of classes. But Hilbert’s programme suggests an alternative view, according to which the objects of the new theory may be regarded as no more than convenient fictions whose use is justified by the conservativeness proof: it is useful to argue as if such entities exist because doing so sometimes permits shorter proofs, but in every such case we know that if required we could in principle purge the proofs of all mention of the ideal elements [in our example, classes] so as to be left with only unobjectionable reference to the real objects [sets] to which our prior theory already committed us.

**MK: real classes?** Consider next Morse-Kelly set theory – which is what you get if you take VNB but now allow class abstracts for *any* condition \(C\) which can be expressed in the language, including conditions which quantify over classes. This theory is no longer conservative over ZFC: the postulation of classes is therefore now more than just a device to make a prior set theory technically smoother in certain respects. Hence it seems that here we really are committed to a two layer universe of collections: old-style sets of sets, then added classes of sets, classes which are too big to themselves be identified with sets. (That fount of all knowledge, Wikipedia, pronounces that the ontology of NBG and of MK is the same: but this seems wrong – as Fraenkel et al. 1973 put it, the former theory doesn’t take classes seriously, the latter theory does.)

Now, as with ZFC\(_v\) and VNB, in MK too classes have members but are not themselves members (of either sets or of classes). MK therefore avoids an analogue of Russell’s paradox appearing at the level of classes: there is no class of all the self-membered classes because there is no class containing any classes. So far, so good.

The trouble is however that if we are allowed to gather sets into collections, some of which are too big to be a set, then – once we are treating classes as genuine entities in their own right – exactly why are we not allowed to gather classes into collections too, even if some are too big to be classes themselves? Is it a stably justifiable position both to countenance classes as real entities, and yet also to ban ourselves from considering any collections of them at all? What is the peculiar ontological status of classes which makes them more than fictions, more than the shadows cast by a technical trick, yet also makes such that we can’t form even finite collections such as ordered pairs of them? This is puzzling, to say the least.

**Classes as big sets** Recall that, roughly speaking, an *inaccessible cardinal* is an uncountable cardinal number \(\kappa\) which cannot be ‘accessed’ from below by applying the basic operations of cardinal arithmetic to smaller cardinals. ZFC cannot prove the existence
of inaccessible cardinals. For if $\kappa$ is indeed an inaccessible cardinal, then the sets introduced below the level $V_\kappa$ in a cumulative model of set theory will already be a model of ZFC, so if ZFC could prove $\kappa$ exists, it would prove there is a level $V_\kappa$ and hence prove itself consistent, which it can’t (assuming it is consistent).

So consider now the theory ZFC + there is an inaccessible cardinal. This is a proper extension of ZFC, proves ZFC is consistent, and a model of the theory will have ‘small’ sets which live below level $V_\kappa$ (for the first inaccessible $\kappa$), and then ‘big’ sets living higher up. (Perhaps we should call the big sets ‘potentially big’, as e.g. the singleton of $\kappa$ is small in terms of cardinality, though it first appears at a rank beyond the inaccessible $\kappa$).

The resulting universe of sets will be vast: but there is nothing intrinsically mysterious about the status of the ‘big’ sets – in particular we can collect even big sets living up to a given level $\kappa + \alpha$ into sets living one level yet higher up the hierarchy.

Suppose we call the small sets of ZFC + there is an inaccessible cardinal – i.e. the sets which give a model of the intended sort for ZFC – simply sets and call the big sets classes (some of which will be proper classes, i.e. too big to be put in bijective correspondence with a set). Now at last, it seems, we do have a coherent story which takes classes seriously: our world includes sets and classes of sets which are too big to be sets and classes of classes too – though of course no class of all classes. (Observation: the sets up to $V_{\kappa+1}$ provide a model of MK: but once you countenance $V_{\kappa+1}$ it is difficult to conceive of any reason not to like $V_{\kappa+2}$ etc. Which makes again the point that it is difficult to fix on a sensible conception of a universe of collections which underpins exactly MK and no more.)

### 7.4 Categories and classes: back to basics

(a) Back to categories. And having now raised issues of ‘size’ and of collections-too-big-to-be-sets, it is worth briefly backtracking right to the very outset. So recall our definition of the very idea of a category. It began like this:

**Definition 1** The data for a category $\mathcal{C}$ comes in two distinct, non-overlapping, sorts:

1. **Objects** (which we will typically notate by ‘$A$', ‘$B$', ‘$C$', . . .).
2. **Arrows** (which we typically notate by ‘$f$', ‘$g$', ‘$h$', . . .). . .

That is pretty much in accord with e.g. Awodey (2006, p. 4) and Lawvere and Schanuel (2009, p. 21).

But this is not the most common way of putting things. Thus Simmons (2011, p. 2), for example, starts

**Definition 1** A category $\mathcal{C}$ consists of

1. A collection $\text{Obj}$ of entities called objects.
2. A collection $\text{Arw}$ of entities called arrows. . .

Leinster (2014, p. 10) also talks of categories consisting in collections, as does Goedecke (2013). Others, for example Borceux (1994, p. 4) and Adámek et al. (2009, p. 18), talk of a category consisting of a class of objects, etc.

We probably shouldn’t read very much into the choice of wording, ‘collections’ vs ‘classes’. The real question is: what, if anything, is the difference between talking of a category as having as data some objects (plural) and some arrows (plural), and saying that a category consists in a collection/class (singular) of objects and a collection/class (singular) of arrows?
It obviously depends what we mean here by ‘classes’. We know, because many paradigm categories have too many objects to form a set, that some notion of class must be intended here that contrasts with the notion of a set (or with the notion of a set in the usual cumulative hierarchy, at any rate). But that still leaves options. When Defn. 1* is given, are we just employing a fiction of virtual classes and saying no more than is said by Defn. 1 which doesn’t talk of collections or classes at all? Or are we buying into some overall theory of sets-and-classes which allows large collections, too big to be ‘ordinary’ sets: and if so, how seriously are we to take these? Two observations:

(1) It is clear that most of those who propose a definition along the lines of Defn. 1* do in fact intend the talk of collections or classes to be construed substantively, as they typically soon go on at least to raise the question of the kind of theory sets-and-classes which is being presupposed (even if they then may do little more than arm-wave or shelve the question).

(2) On the other hand, note that the first six chapters of this introduction to category theory happily worked with Defn. 1, without making any assumption that the objects or the arrows of a category do fall into collections or classes which are distinct entities over and above the objects and arrows. We could have used virtual-class talk to make a few statements along the way marginally snappier: but we didn’t, and we lost nothing by eschewing class-talk.

So, it seems that we don’t need to buy in to a substantive theory of classes at least at the outset of our theorizing about categories. The issue then arises: just when, if at all, does the category theorist need to take on the further commitment? And how weighty is this further commitment?

I frankly don’t know the detailed answer to this. We have, for example, in this present chapter started talking about categories of categories. Now, small categories can be thought of as living in (or at least as being modelled in) the world of sets, and the collections that constitute a category $\text{Cat}$ of small categories can perhaps comfortably be thought of as virtual classes. But a locally small category can have too many objects to form a set. So, it seems, the category $\text{CAT}$ of locally small categories has a class of objects which may themselves comprise classes of objects; this apparently takes us beyond the merely virtual, beyond the reach of VBN or even of MK. But it remains to be seen when $\text{CAT}$ becomes a load-bearing part of much category-theoretic theorizing which we need to accommodate with a substantive theory of classes.

At this point, then, I can only propose that we keep our wits about us and try to note if and when we do get seriously entangled with a commitment to real-collections-too-big-to-be-sets. Perhaps much the best thing to do is just to continue to develop category theory, talking in a relaxed way about sets, classes (and perhaps even bigger collections) on an as-needed basis. We can then plan to return later to see in retrospect what kind of background theory about sets and classes a category theorist really needs at various stages of the development of her theory if the load-bearing parts of her actual mathematical practice are indeed to be in good order.

If you do, however, insist on more up-front discussion of foundations-in-the-theory-of-classes before going any further, then two options are to be found in e.g. Borceux (1994, §1.1) and Adámek et al. (2009, §2). (A comment re Borceux’s discussion. His universe $\mathcal{U}$ is a so-called Grothendieck universe; and it can be shown that the level $V_\kappa$ of the usual cumulative hierarchy for inaccessible $\kappa$ constitutes a Grothendieck universe. So we are swimming here in the same conceptual waters as in the fourth line on classes which we identified.) For a much more sophisticated considerations of the options looking back
from the vantage point of advanced knowledge of category theory, you will perhaps one day want to tackle Shulman (2008).

7.5 Another definition of categories

We often find, instead of Definition 1*, the following:

Definition 1** The data for a category \( \mathcal{C} \) comprises:

1. A class \( \text{ob}(\mathcal{C}) \), whose elements we will call **objects**.
2. For every \( A, B \in \text{ob}(\mathcal{C}) \), a class \( \mathcal{C}(A, B) \), whose elements \( f \) we will call **arrows** from \( A \) to \( B \). We signify that the arrow \( f \) belongs to \( \mathcal{C}(A, B) \) by writing \( f : A \to B \) or \( A \xrightarrow{f} B \).
3. For every \( A \in \text{ob}(\mathcal{C}) \), an arrow \( 1_A \in \mathcal{C}(A, A) \) called the **identity** on \( A \).
4. For any \( A, B, C \in \text{ob}(\mathcal{C}) \), a two-place **composition** operation, which takes arrows \( f, g \), where \( f : A \to B \) and \( g : B \to C \), to an arrow \( g \circ f : A \to C \), the composite of \( f \) and \( g \).

The axioms are as before.

So the difference is that Defn. 1* has one all-in class of arrows, Defn. 1** has lots of different classes of arrows, one (perhaps empty) for every pair of objects in the category. Obviously if we start from Defn. 1* we can define the class \( \mathcal{C} \) different classes of arrows, one (perhaps empty) for every pair of objects in the category. The axioms are as before.

Now note that on Defn. 1*, the arrows \( f : A \to B \) and \( f' : A' \to B' \) cannot be identical items if \( A \neq A' \) or \( B \neq B' \). For if \( \text{src}(f) \neq \text{src}(f') \), \( f \neq f' \); likewise, of course, if \( \text{tar}(f) \neq \text{tar}(f') \), \( f \neq f' \). Hence if \( A \neq A' \) or \( B \neq B' \), \( \mathcal{C}(A, B) \neq \mathcal{C}(A', B') \). On the other hand, there’s nothing in Defn. 1** that ensures that \( \mathcal{C}(A, B) \) and \( \mathcal{C}(A', B') \) are disjoint when \( A \neq A' \) or \( B \neq B' \). That is to say, on the new definition, there’s nothing (yet) to stop the arrows \( f : A \to B \) and \( f' : A' \to B' \) being the same item even if \( A \neq A' \) or \( B \neq B' \). That means that our two definitions don’t quite line up. What to do?

(i) Some authors simply add to Defn. 1** the requirement that the classes \( \mathcal{C}(A, B) \) for different pairs of objects are disjoint.

(ii) Alternatively, we can associate with every category in the sense of the new definition an associated category in the sense of the previous one by putting \( \text{arw}(\mathcal{C}) = \coprod_{A, B \in \text{ob}(\mathcal{C})} \mathcal{C}(A, B) \) i.e. take the disjoint union of classes of arrows (assume we can make that construction!). Though notice that the round trip – starting with a category in the new sense, constructing (as suggested) the associated category in the original sense and then going back to a category in the new sense using our definition of \( \mathcal{C}(A, B) \) from \( \text{arw}(\mathcal{C}) \) – doesn’t quite return us to our starting point. We get a category which, in a good sense, ‘looks just like’ the one we started off with; but it’s an isomorphic copy (in a sense to be made clear), not the original. Still, the difference really shouldn’t matter.

Among familiar books, Borceux (1994) gives the new definition; so does Adámek et al. (2009), adding ‘for technical convenience’ the requirement that hom-sets are disjoint, and Leinster (2014) does the same. While Awodey (2006), Goldblatt (2006), Lawvere and Schanuel (2009), Mac Lane (1997), and Simmons (2011) use either Defn. 1 or Defn. 1*. So both styles of definition are current, and hence it is worth explicitly noting the difference. For later convenience, however, we stick to our original definition that requires that classes \( \mathcal{C}(A, B) \) for distinct objects are disjoint.
Where do we go next? Here are three large families of topics to explore:

**Limits** Back in the preamble to Ch. 3, we noted that what’s important e.g. about Cartesian products is not how they are implemented, not what a product-object ‘consists in’, but rather how a product $A \times B$ is related to $A$ and $B$ by pair-forming and pair-unforming morphisms – an observation that is crying out for category-theoretic treatment.

So how should we deal with products (and indeed some similar constructions like disjoint sums) now we have some category theory to hand? We will find that these sorts of constructions can all be thought of as involving limits (or colimits) in a certain sense we need to explore.

**Adjoints** Recall the remark from Tom Leinster which we quoted at the outset: ‘Category theory takes a bird’s eye view of mathematics. From high in the sky … we can spot patterns that were impossible to detect from ground level.’ One of the patterns that is, if not impossible, at least difficult to detect from ground level if you are not imbued with categorial ideas is the ubiquity of so-called adjunctions (think of these as a generalization of Galois Connections, if you know what they are). We will want to explore these too.

**Representables/Yoneda Lemma** Cayley’s Theorem in group theory tells us that every group is isomorphic to a permutation group shuffling around the members of a set. In other words, there’s a good sense in which any group can given a (relatively!) concrete set-representation. There are other similar classical representation theorems. Now we can reflect such results in category theory by investigating functors of a special form from various categories into the category of sets which can be thought of as giving representations of those categories in the world of sets. Key results in this area concern the so-called Yoneda embedding and the associated Yoneda Lemma.

Now, category theory being an enquiry into how mathematical things (in a broad sense) hang together (in a broad sense), it’s not surprising to learn that these topics too hang together, being interrelated in complex ways. Which means that there is no obviously best order in which to tackle them. Thus Leinster (2014) has chapters dealing with adjoints, representables, and limits, in that order, before knitting everything together in a final chapter ‘Adjoints, representable and limits’. Awodey (2006) reverses this order, and tackles limits first, followed by scattered material on representables with
the Yoneda Lemma not appearing until Ch. 8, and then only in his penultimate Ch. 9
do we meet adjoints. In his more elementary text, Simmons (2011) also tackles limits be-
fore adjunction, and comparatively downplays representables (and doesn’t mention the
Yoneda Lemma as such). By contrast Adámek et al. (2009) prove the Yoneda Lemma
very early in their book, before discussing limits or adjunctions: Borceux (1994) does the
same. Finally, to take just one more example, Goldblatt (2006) starts his classic book
talking about limits. And it is only four hundred pages later – after we’ve done some
quite detailed exploration of those categories which are toposes (or topoi, if you prefer) –
that we get a little on adjoints and on representables.

So we have to make a presentational choice here about the order of exposition. The
Cambridge Way has been (briefly) to discuss representables, the Yoneda embedding and
Yoneda’s Lemma, early on, before tackling limits and adjoints (see the transcription of
Peter Johnstone’s lectures at http://www.math.cornell.edu/~bfontain/cattheory.pdf or
Julia Goedecke’s https://www.dpmms.cam.ac.uk/~jg352/pdf/CategoryTheoryNotes.pdf,
and this was also the ordering followed by RL-W’s lectures). We’ll take the same path
here. Which doesn’t mean that I’m convinced at the outset that this is the best policy,
but let’s see how things go – and perhaps consider later where we have profited from
doing things this way about.

8.1 Hom-sets

(a) Let’s start by recycling a bit of symbolism which got introduced in passing at
the end of the last chapter (see § 7.5, Defn. 1**). We there used \( \mathcal{C}(A, B) \) for the class of
\( \mathcal{C} \)-arrows from \( A \) to \( B \). Now, if \( \mathcal{C} \) is locally small, then by definition there is only a set’s
worth of arrows from \( A \) to \( B \) for any choice of \( \mathcal{C} \)-objects \( A, B \). So in this case, we can
think of \( \mathcal{C}(A, B) \) as forming a set-sized collection. Which motivates

**Definition 36.** If \( \mathcal{C} \) is locally small, then the hom-set \( \mathcal{C}(A, B) \) is the set of \( \mathcal{C} \)-arrows
from \( A \) to \( B \).

An alternative notation is ‘Hom\((A, B)\)’: the terminology comes from that paradigm case
where the set of arrows in question is in fact a set of homomorphisms between two
algebraic structures.

But is \( \mathcal{C}(A, B) \) a set that lives in \textbf{Set}? We haven’t fixed which category \textbf{Set} is exactly
(see the remark back in §1.2); but if you think of that as comprising the sets you know
and love from a basic set theory course in the delights of ZFC, then that is a category
of pure sets, i.e. of sets whose members, if any, are sets whose members, if any, are sets
\ldots all the way down. So if we think of \( \mathcal{C}(A, B) \) as living in a category \textbf{Set} of pure sets,
then the arrows which are members of \( \mathcal{C}(A, B) \) will have to be pure sets too. But do we
want to be committed to thinking of the arrows of a category \( \mathcal{C} \) as always being pure
sets?

However, it is absolutely standard in category theory to treat \( \mathcal{C}(A, B) \) as an object of
\textbf{Set}, whatever the category \( \mathcal{C} \). So it seems that we have three options. (i) Continue
to regard \textbf{Set} as a theory of pure sets, and take \( \mathcal{C}(A, B) \) to be a pure set, so arrows of
any category (or at least, of any category we are interested in) to be pure sets – which is
to build in from the start a thorough-going set-theoretic reductionism about categories.
(ii) Allow impure sets in \textbf{Set} where the non-set elements can be arrows in any category.
Or (iii) we can take \textbf{Set} to be a universe of pure sets, and \( \mathcal{C}(A, B) \) to be, in general, an
impure set whose members are arrows (which needn’t be themselves sets), but then say
that it isn’t strictly speaking \( \mathcal{C}(A, B) \) as originally defined which lives in \textbf{Set} but rather
a set which represents or models it, whose elements are in a one-to-one correspondence
with the members of \( \mathcal{C}(A, B) \) – call this representing set \( \mathcal{C}(A, B) \) too, with context deciding when we are talking about the original hom-set and when we are talking about its pure-set representation. We will have to shelve the question of which is the best way to go. (Note that in his canonical 1997, Saunders Mac Lane initially gives a definition like our Defn. 1 as a definition of what he calls metacategories, and then for him a category proper “will mean any interpretation of the category axioms within set theory” – so for Mac Lane and those who follow him, the hom-sets of a category, like all the other gadgets of category theory, will unproblematically live in the universe of set theory, and they can take option (i). But do we want or need to suppose that categories are always and everywhere sets? Not perhaps if we have ambitions for category theory as a more democratic way of organizing the mathematical universe, which provides an alternative to set-theoretic imperialism.)

### 8.2 Hom-functors

(a) Take \( \mathcal{C}(A, B) \), the hom-set of \( \mathcal{C} \)-arrows from \( A \) to \( B \). Keep \( A \) fixed. Then as we vary \( X \) through the objects in \( \mathcal{C} \), we get varying \( \mathcal{C}(A, X) \).

So: consider the resulting function which sends an object \( X \) in \( \mathcal{C} \) to the set \( \mathcal{C}(A, X) \), a set which we follow standard practice as taking as living in \( \text{Set} \).

Can we now treat this function on \( \mathcal{C} \)-objects as the first component of a functor, call it \( \mathcal{C}(A,-) \), from \( \mathcal{C} \) to \( \text{Set} \)? Well, how could we find a component of the functor to deal with the \( \mathcal{C} \)-arrows? Such a component is going to need to send an arrow \( f : X \to Y \) in \( \mathcal{C} \) to a \( \text{Set} \)-function from \( \mathcal{C}(A, X) \) to \( \mathcal{C}(A, Y) \). The obvious candidate for the latter function is the one we can notate as \( f \circ - \) that maps any \( g : A \to X \) in \( \mathcal{C}(A, X) \) to \( f \circ g : A \to Y \) in \( \mathcal{C}(A, Y) \). (Note, \( f \circ g : A \to Y \) has to be in \( \mathcal{C}(A, Y) \) because \( \mathcal{C} \) is a category which by hypothesis contains \( g : A \to X \) and \( f : X \to Y \) and hence must contain their composition.)

It is easy to check that these components add up to a genuine covariant functor. So far, so good!

Now, start again from the hom-set \( \mathcal{C}(A, B) \) but now keep \( B \) fixed: then as we vary \( X \) through the objects in \( \mathcal{C} \), we again get varying hom-sets \( \mathcal{C}(X, B) \). Which generates a function which sends an object \( X \) in \( \mathcal{C} \) to an object \( \mathcal{C}(X, B) \) in \( \text{Set} \). To turn this into a functor \( \mathcal{C}(-, B) \), we need again to add a component to deal with \( \mathcal{C} \)-arrows. That will need to send \( f : X \to Y \) in \( \mathcal{C} \) to some function between \( \mathcal{C}(X, B) \) to \( \mathcal{C}(Y, B) \). But this time, to get functions to compose properly, things will have to go the other way about, i.e. the associated functor will have to send a function \( g : Y \to B \) in \( \mathcal{C}(Y, B) \) to \( g \circ f : X \to B \) in \( \mathcal{C}(X, B) \). So this means that the resulting functor \( \mathcal{C}(-, B) \) is a contravariant hom-functor.

So, to summarize:

**Definition 37.** Given a locally small category \( \mathcal{C} \), then the associated **covariant hom-functor** \( \mathcal{C}(A,-) : \mathcal{C} \to \text{Set} \) is the functor with the following data:

1. A mapping \( \mathcal{C}(A,-)_{\text{ob}} \) whose value at the object \( X \) in \( \mathcal{C} \) is the hom-set \( \mathcal{C}(A, X) \).
2. A mapping \( \mathcal{C}(A,-)_{\text{arw}} \), whose value at the \( \mathcal{C} \)-arrow \( f : X \to Y \) is the set function \( f \circ - \) from \( \mathcal{C}(A, X) \) to \( \mathcal{C}(A, Y) \) which sends an element \( g : A \to X \) to \( f \circ g : A \to Y \).

And the associated **contravariant hom-functor** \( \mathcal{C}(-, B) : \mathcal{C} \to \text{Set} \) is the functor with the following data:

3. A mapping \( \mathcal{C}(-, B)_{\text{ob}} \) whose value at the object \( X \) in \( \mathcal{C} \) is the hom-set \( \mathcal{C}(X, B) \).
(4) A mapping $\mathcal{C}(-, B)_{arw}$, whose value at the $\mathcal{C}$-arrow $f: Y \to X$ is the set function $- \circ f$ from $\mathcal{C}(X, B)$ to $\mathcal{C}(Y, B)$ which sends an element $g: X \to B$ to the map $g \circ f: Y \to B$.

The use of a blank in the notion ‘$\mathcal{C}(A, -)$’ invites an obvious shorthand: instead of writing ‘$\mathcal{C}(A, -)_{arw}(f)$’ to indicate the result of the component of the functor which acts arrows applied to the function $f$, we will write simply ‘$\mathcal{C}(A, f)$’. Similarly for the dual.

For the record, we can also talk about a ‘bi-functor’ $\mathcal{C}(\cdot, \cdot): \mathcal{C}^{op} \times \mathcal{C} \to \textbf{Set}$, contravariant in the first place and covariant in the second. The headline news is that we can think of this as acting on a product category mapping the pair object $(A, B)$ in $\mathcal{C}$ to the hom-set $\mathcal{C}(A, B)$, and the pair of morphisms $(f: X' \to X, g: Y \to Y')$ to the morphism between $\mathcal{C}(X, Y)$ and $\mathcal{C}(X', Y')$ that sends $h: X \to Y$ to $h \circ g \circ f: X' \to Y'$.

We will return to this if/when we need to say more.

(b) Take a (locally small) category $\mathcal{C}$, and fix on a $\mathcal{C}$-arrow $f: B \to A$. We’ll describe how to construct from $f$ a corresponding natural transformation $\alpha$ between the hom-functors $\mathcal{C}(A, -)$ and $\mathcal{C}(B, -)$.

If $\alpha$ is to be a natural transformation, its components must be such that the following diagram commutes, given any arrow $j: X \to Y$:

$$
\begin{array}{ccc}
\mathcal{C}(A, X) & \xrightarrow{\mathcal{C}(A, j)} & \mathcal{C}(A, Y) \\
\downarrow{\alpha_X} & & \downarrow{\alpha_Y} \\
\mathcal{C}(B, X) & \xrightarrow{\mathcal{C}(B, j)} & \mathcal{C}(B, Y)
\end{array}
$$

where $\mathcal{C}(C, j)$ is (as just defined) the map which sends an arrow $g: C \to X$ to the map $j \circ g: C \to Y$.

Suppose we set a component $\alpha_Z: \mathcal{C}(A, Z) \to \mathcal{C}(B, Z)$ to be the function that sends an arrow $g: A \to Z$ to the composite $g \circ f: B \to Z$ (so we could also write $\alpha_Z$ as $- \circ f$).

Then our diagram will indeed commute. For going round the top-route takes us from $g: A \to X$ to $j \circ g: A \to Y$ to $(j \circ g) \circ f: B \to Y$; and going round the bottom route takes us from $g: A \to X$ to $g \circ f: A \to Y$ to $j \circ (g \circ f): B \to Y$.

So in sum, if there is a morphism $f: B \to A$, then there is a corresponding natural transformation $\alpha: \mathcal{C}(A, -) \Rightarrow \mathcal{C}(B, -)$ with components $\alpha_Z$ as defined.

And note: if $f$ is an isomorphism, then each component $\alpha_Z$ (i.e. $- \circ f$) has an inverse (i.e. $- \circ f^{-1}$), so is an isomorphism. Therefore $\alpha$ is then a natural isomorphism.

To sum up this result and introduce some slightly more perspicuous notation:

**Theorem 33.** Suppose $\mathcal{C}$ is a locally small category, and consider the hom-functors $\mathcal{C}(A, -)$ and $\mathcal{C}(B, -)$, for objects $A, B$ in $\mathcal{C}$. Then if there exists an arrow $f: B \to A$, there is a corresponding natural transformation $\mathcal{C}(f, -): \mathcal{C}(A, -) \Rightarrow \mathcal{C}(B, -)$, where for each $Z$, the component $\mathcal{C}(f, -)_Z: \mathcal{C}(A, Z) \to \mathcal{C}(B, Z)$ sends an arrow $j: A \to Z$ to $j \circ f: B \to Z$. Furthermore, if $f$ is an isomorphism, then $\mathcal{C}(f, -)$ is a natural isomorphism.

(c) As a quick reality-check, let’s for the record show

**Theorem 34.** Given a locally small category $\mathcal{C}$ including objects $A, B, C$, and arrows $f: B \to A$ and $g: C \to B$, then

1. $\mathcal{C}(f \circ g, -) = \mathcal{C}(g, -) \circ \mathcal{C}(f, -)$.
2. $\mathcal{C}(f, -)_A 1_A = f$.
3. $\mathcal{C}(1_A, -) = 1_{\mathcal{C}(A, -)}$. 

Proof. (1) \( \mathcal{C}(f \circ g, -)_Z \) sends any arrow \( e: A \to Z \) to \( e \circ (f \circ g): C \to Z \). But \( \mathcal{C}(f, -)_Z(e) = e \circ f \), so \( \mathcal{C}(g, -)_Z(\mathcal{C}(f, -)_Z(e)) = (e \circ f) \circ g \). Hence \( \mathcal{C}(f \circ g, -) \) and \( \mathcal{C}(g, -) \circ \mathcal{C}(f, -) \) agree on all components, so are identical natural transformations.

(2) \( \mathcal{C}(f, -)_A \) sends any arrow \( j: A \to A \) to \( j \circ f: B \to A \). So in particular it sends \( 1_A \) to \( f \).

(3) \( \mathcal{C}(1_A, -)_Z \) sends any arrow \( j: A \to Z \) to itself. While \( 1_{\mathcal{C}(A, -)} \) is the identity arrow on the object \( \mathcal{C}(A, -) \) in the functor category \( [\mathcal{C}, \textbf{Set}] \). Arrows in that functor category are natural transformation: so the component \( (1_{\mathcal{C}(A, -)})_Z \) is the identity on \( \mathcal{C}(A, Z) \), i.e. sends any arrow \( j: A \to Z \) to itself. Which shows that \( \mathcal{C}(1_A, -) \) and \( 1_{\mathcal{C}(A, -)} \) agree on all components and therefore are identical. \( \square \)

(d) The theorems so far have been about covariant hom-functors. We have corresponding duals to Theorems 33 and 34 for contravariant hom-functors. We’ll state the first theorem and also the part of the second theorem that we’ll need, leaving proofs as routine exercises in dualizing.

**Theorem 35.** Suppose \( \mathcal{C} \) is a locally small category, and consider the contravariant hom-functors \( \mathcal{C}(\cdot, A) \) and \( \mathcal{C}(\cdot, B) \), for objects \( A, B \) in \( \mathcal{C} \). Then if there exists an arrow \( f: A \to B \), there is a corresponding natural transformation \( \mathcal{C}(\cdot, f): \mathcal{C}(\cdot, A) \Rightarrow \mathcal{C}(\cdot, B) \), where for each \( Z \), the component \( \mathcal{C}(\cdot, f)_Z: \mathcal{C}(Z, A) \to \mathcal{C}(Z, B) \) sends an arrow \( j: Z \to A \) to \( f \circ j: Z \to B \).

**Theorem 36.** Given a locally small category \( \mathcal{C} \) including objects \( A, B, C \), and arrows \( f: A \to B \) and \( g: B \to C \), then \( \mathcal{C}(\cdot, g \circ f) = \mathcal{C}(\cdot, g) \circ \mathcal{C}(\cdot, f) \).

### 8.3 Generating natural transformations

The obvious next question to ask is: are all possible natural transformations between the hom-functors \( \mathcal{C}(A, -) \) and \( \mathcal{C}(B, -) \) generated from arrows \( f: B \to A \) in the way described in Theorem 33.

The answer is give by:

**Theorem 37.** Suppose \( \mathcal{C} \) is a locally small category, and consider the hom-functors \( \mathcal{C}(A, -) \) and \( \mathcal{C}(B, -) \), for objects \( A, B \) in \( \mathcal{C} \). Then if there is a natural transformation \( \alpha: \mathcal{C}(A, -) \Rightarrow \mathcal{C}(B, -) \), there is a unique arrow \( f: B \to A \) such that \( \alpha = \mathcal{C}(f, -) \).

**Proof.** Since \( \alpha \) is a natural transformation, the following diagram must commute, for any \( Z \) and any \( g: A \to Z \),

\[
\begin{array}{ccc}
\mathcal{C}(A, A) & \xrightarrow{\mathcal{C}(A, g)} & \mathcal{C}(A, Z) \\
\downarrow{\alpha_A} & & \downarrow{\alpha_Z} \\
\mathcal{C}(B, A) & \xrightarrow{\mathcal{C}(B, g)} & \mathcal{C}(B, Z)
\end{array}
\]

where, recalling the definition, \( \mathcal{C}(A, g) \) is a map that (among other things) sends an arrow \( h: A \to A \) to the arrow \( g \circ h: A \to Z \), and \( \mathcal{C}(B, g) \) is a map which sends an arrow \( j: B \to A \) to the arrow \( g \circ j: B \to Z \).

Why start with \( \mathcal{C}(A, A) \) at the top left? Because we know that such hom-sets must be populated, if only by the identity arrow \( 1_A: A \to A \).

Chase that identity arrow round the diagram from the top left to bottom right nodes. The top route gives us \( \alpha_Z(g) \), and the bottom route gives us \( \mathcal{C}(B, g)(\alpha_A(1_A)) \), and since the diagram commutes these must be equal. So put \( f: B \to A = \text{def} \ \alpha_A(1_A) \).
That means \( \alpha_Z(g) = \mathcal{C}(B,g)(f) = g \circ f = \mathcal{C}(f,\_)_Z(g) \). Therefore since \( Z \) and \( g \) were arbitrary, we get \( \alpha = \mathcal{C}(f,\_) \) as required.

Now suppose both \( f \) and \( f' \) are such that for each \( Z \), \( \mathcal{C}(f,\_)_Z = \mathcal{C}(f',\_)_Z \). Then by Theorem 34 (2)
\[
f = \mathcal{C}(f,\_)(1_A) = \mathcal{C}(f',\_)(1_A) = f'
\]
which shows \( f \)'s uniqueness.

Needless to say, there’s a dual version of the theorem about contravariant hom-functors. It is another routine exercise in dualizing to prove it:

**Theorem 38.** Suppose \( \mathcal{C} \) is a locally small category, and consider the hom-functors \( \mathcal{C}(\_,A) \) and \( \mathcal{C}(\_,B) \), for objects \( A,B \) in \( \mathcal{C} \). Then if there is a natural transformation \( \alpha: \mathcal{C}(\_,A) \Rightarrow \mathcal{C}(\_,B) \), there is a unique arrow \( f: A \rightarrow B \) such that \( \alpha = \mathcal{C}(\_,f) \).

### 8.4 The Restricted Yoneda Lemma

For brevity’s sake, let’s introduce some shorthand:

**Definition 38.** We use the standard notation ‘\( \text{Nat}(F,G) \)’ to denote the set of natural transformations from \( F \) to \( G \) in \( \mathcal{C} \). And just in this section and the next, we will further abbreviate ‘\( \text{Nat}(\mathcal{C}(A,\_),\mathcal{C}(B,\_)) \)’ by simply ‘\( \text{Nat}_{AB} \)’.

Since the ambient category \( \mathcal{C} \) is being assumed to be locally small, \( \text{Nat}(F,G) \) is indeed a set.

Now note that a \( \mathcal{C} \)-arrow \( f: B \rightarrow A \) is of course a member of the hom-set \( \mathcal{C}(B,A) \). So, in the proofs of our Theorems 33 and 37 we have in effect defined two families of functions \( \mathcal{X}_{AB} \) and \( \mathcal{E}_{AB} \), i.e. families of arrows in \textbf{Set} indexed by \( A,B \in \mathcal{C} \), where

i) \( \mathcal{X}_{AB}: \mathcal{C}(B,A) \rightarrow \text{Nat}_{AB} \) sends a function \( f: B \rightarrow A \) to the natural transformation \( \mathcal{C}(f,\_). \)

ii) \( \mathcal{E}_{AB}: \text{Nat}_{AB} \rightarrow \mathcal{C}(B,A) \) sends a natural transformation \( \alpha: \mathcal{C}(\_,A) \Rightarrow \mathcal{C}(\_,B) \) to \( \alpha_A(1_A) \).

And again, the next question to ask is obvious: are \( \mathcal{X}_{AB} \) and \( \mathcal{E}_{AB} \) inverses of each other in \textbf{Set}? We might expect so.

Let’s fix on some particular \( A \) and \( B \), and – to improve readability – temporarily drop subscripts on \( \mathcal{X} \) and \( \mathcal{E} \). Then we note:

1. Given an arbitrary \( f: B \rightarrow A \),
\[
(\mathcal{E} \circ \mathcal{X}) f = \mathcal{E}(\mathcal{C}(f,\_)) = \mathcal{C}(f,\_)(1_A) = f
\]
whence \( \mathcal{E} \circ \mathcal{X} = 1_{\mathcal{C}(B,A)} \).

2. Given an arbitrary \( \alpha: \mathcal{C}(\_,A) \Rightarrow \mathcal{C}(\_,B) \),
\[
(\mathcal{X} \circ \mathcal{E}) \alpha = \mathcal{X}(\alpha_A(1_A)) = \mathcal{C}(\alpha_A(1_A),\_)
\]
But, by definition, \( \mathcal{C}(\alpha_A(1_A),\_)_Z \) sends an arrow \( g: A \rightarrow Z \) to \( g \circ \alpha_A(1_A): B \rightarrow Z \),
and looking at the last commutative diagram, \( g \circ \alpha_A(1_A) = \alpha_Z(g \circ 1_A) = \alpha_Z(g) \).
So for each \( Z \), \( \mathcal{C}(\alpha_A(1_A),\_)_Z = \alpha_Z \). Hence, having identical components, the natural transformations are identical and \( (\mathcal{X} \circ \mathcal{E}) \alpha = \alpha \). Since that holds for any \( \alpha \), \( \mathcal{X} \circ \mathcal{E} = 1_{\text{Nat}} \).
So $\mathcal{X}_{AB}$ and $\mathcal{E}_{AB}$ are mutual inverses, and hence the following is immediate:

**Theorem 39** (Restricted Yoneda Lemma). Suppose $\mathcal{C}$ is a locally small category, and $A, B$ are objects of $\mathcal{C}$. Then $\text{Nat}(\mathcal{C}(A, -), \mathcal{C}(B, -)) \cong \mathcal{C}(B, A)$.

There are, needless to say, dual versions of all this. In particular, for each $A, B$ in $\mathcal{C}$, there is an isomorphism $\mathcal{Y}_{AB} : \mathcal{C}(A, B) \rightarrow \text{Nat}(\mathcal{C}(-, A), \mathcal{C}(-, B))$ which sends a function $f : A \rightarrow B$ to the natural transformation $\mathcal{C}(-, f)$; and $\mathcal{Y}_{AB}$ has an inverse. Consequently,

**Theorem 40** (Restricted Yoneda Lemma). Suppose $\mathcal{C}$ is a locally small category, and $A, B$ are objects of $\mathcal{C}$. Then $\text{Nat}(\mathcal{C}(-, A), \mathcal{C}(-, B)) \cong \mathcal{C}(A, B)$.

The shared label we’ve given this dual pair of theorems is not standard, but the reason for it will become clear when we meet the full Yoneda Lemma in Ch. 10. That full version has a reputation for being the first result in category theory whose proof takes some effort to understand. Be that as it may, our cut-down version at least should seem unproblematic: once we have Theorem 33 (which just restated an early result), it’s just a matter of asking the obvious questions and finding the easy answers. And the Restricted Yoneda Lemma is all we need for the main result in this chapter.

### 8.5 A new pair of fully faithful functors

Continuing to assume that we are working in a category $\mathcal{C}$ which is locally small, then falling into the convenient shorthand of writing ‘$A \in \mathcal{C}$’ for ‘$A$ is an object of $\mathcal{C}$’ –

i) We’ve associated with each object $A \in \mathcal{C}$ a hom-functor $\mathcal{C}(A, -)$.

ii) For each $A, B \in \mathcal{C}$, we’ve a bijection $\mathcal{X}_{AB}$ which sends any $f : B \rightarrow A$ to the natural transformation $\mathcal{C}(f, -) : \mathcal{C}(A, -) \Rightarrow \mathcal{C}(B, -)$.

Now, the hom-functors $\mathcal{C}(A, -)$ and $\mathcal{C}(B, -)$ are objects of the functor category $\mathcal{C}^{\text{op}}$, and each natural transformation $\mathcal{C}(f, -) : \mathcal{C}(A, -) \Rightarrow \mathcal{C}(B, -)$ is an arrow in that category. So, given the way the $A/B$ order gets reversed in (ii), the functional associations we’ve just noted look as if they should add up to a contravariant functor from $\mathcal{C}$ to $\mathcal{C}^{\text{op}}$. Equivalently, it looks as if

**Theorem 41.** For any locally small category $\mathcal{C}$, there is a functor we’ll label simply $\mathcal{X} : \mathcal{C}^{\text{op}} \rightarrow [\mathcal{C}, \text{Set}]$ with components $\mathcal{X}_{ob}$ and (for all $A, B \in \mathcal{C}^{\text{op}}$) $\mathcal{X}_{AB}$ such that

1. for any $A \in \text{ob}(\mathcal{C}^{\text{op}})$, $\mathcal{X}_{ob}(A) = \mathcal{C}(A, -)$,
2. for any arrow $f \in \mathcal{C}^{\text{op}}(A, B)$, i.e. arrow $f : B \rightarrow A$ in $\mathcal{C}$, $\mathcal{X}_{\text{arw}}(f) = \mathcal{C}(f, -)$.

*Proof.* The values of $\mathcal{X}_{ob}$ and $\mathcal{X}_{\text{arw}}$ are indeed respectively objects and arrows in $[\mathcal{C}, \text{Set}]$. So we just need to check the two functorial axioms are indeed satisfied. First, identities are preserved:

$\mathcal{X}(1_A) = \mathcal{C}(1_A, -) = 1_{\mathcal{C}(A, -)} = 1_{\mathcal{X}(A)}$

(where the central equation holds by Theorem 34(3). And second, composition is respected. In other words, for any composable $f, g$ in $\mathcal{C}^{\text{op}}$,

$\mathcal{X}(g \circ f) = \mathcal{C}(f \circ g, -) = \mathcal{C}(g, -) \circ \mathcal{C}(f, -) = \mathcal{X}(g) \circ \mathcal{X}(f)$

(where the central equation holds by Theorem 34(1)). □
Once more there is a dual result. Just recall that the contravariant functor \( \mathcal{C}(-, A) \) lives as an object in the functor category \([\mathcal{C}^{op}, \text{Set}]\). So the dual result is this:

**Theorem 42.** For any locally small category \( \mathcal{C} \), there is a functor \( \mathcal{Y} : \mathcal{C} \to [\mathcal{C}^{op}, \text{Set}] \) with components \( \mathcal{Y}_{ob} \) and \( \mathcal{Y}_{arw} \) such that

1. for any \( A \in \text{ob}(\mathcal{C}) \), \( \mathcal{Y}_{ob}(A) = \mathcal{C}(-, A) \).
2. For any arrow \( f \), \( \mathcal{Y}_{arw}(f) = \mathcal{C}(-, f) \).

**Theorem 43.** \( \mathcal{X} : \mathcal{C}^{op} \to [\mathcal{C}, \text{Set}] \) and \( \mathcal{Y} : \mathcal{C} \to [\mathcal{C}^{op}, \text{Set}] \) are fully faithful functors which are injective on objects.

**Proof.** By definition, \( \mathcal{X} : \mathcal{C}^{op} \to [\mathcal{C}, \text{Set}] \) is fully faithful since each component \( \mathcal{X}_{AB} \) is a bijection.

To say \( \mathcal{X} \) is injective on objects is to say, of course, that if \( A \neq B \), then \( \mathcal{X}(A) \neq \mathcal{X}(B) \). Suppose \( \mathcal{X}(A) = \mathcal{X}(B) \), i.e. \( \mathcal{C}(A, -) = \mathcal{C}(B, -) \). Then \( \mathcal{C}(A, -)(C) = \mathcal{C}(B, -)(C) \), i.e. \( \mathcal{C}(A, C) = \mathcal{C}(B, C) \). But that can’t be so if \( A \neq B \), given our assumptions that hom-sets on different pairs of objects are disjoint (see the last sentence of §7.5).

The proof for \( \mathcal{Y} : \mathcal{C} \to [\mathcal{C}^{op}, \text{Set}] \) is dual. \( \square \)

Here’s a quick corollary:

**Theorem 44.** For any objects \( A, B \) in the locally small category \( \mathcal{C} \), \( A \cong B \) iff \( \mathcal{X}A \cong \mathcal{X}B \), and likewise \( A \cong B \) iff \( \mathcal{Y}A \cong \mathcal{Y}B \).

**Proof.** Suppose \( A \cong B \). Then for some isomorphism \( f, f : B \to A \). So there is a natural transformation \( \mathcal{C}(f, -) : \mathcal{C}(A, -) \Rightarrow \mathcal{C}(B, -) \), which by Theorem 33 is an isomorphism. In other words, \( \mathcal{C}(f, -) : \mathcal{X}A \Rightarrow \mathcal{X}B \) is an isomorphism. Hence \( \mathcal{X}A \cong \mathcal{X}B \).

For the converse, suppose \( \mathcal{X}A \cong \mathcal{X}B \). So there exists a natural transformation \( \alpha : \mathcal{C}(A, -) \Rightarrow \mathcal{C}(B, -) \). By Theorem 37, \( \alpha \) is \( \mathcal{C}(f, -) \) from some \( f : B \to A \), i.e. is \( \mathcal{X}f \). But \( \mathcal{X} \) is fully faithful, by Theorem 23, hence since \( \mathcal{X}f \) is an isomorphism, so is \( f \). Hence \( A \cong B \).

That shows \( A \cong B \) iff \( \mathcal{X}A \cong \mathcal{X}B \). The argument for the functor \( \mathcal{Y} \) is parallel. \( \square \)

### 8.6 The Yoneda embedding defined

So the situation is this. The functor \( \mathcal{Y} \), for example, injects a copy of the \( \mathcal{C} \)-objects one-to-one into the objects of the functor category \([\mathcal{C}^{op}, \text{Set}]\); and then it faithfully matches up the arrows between \( \mathcal{C} \)-objects with arrows between the corresponding objects in \([\mathcal{C}^{op}, \text{Set}]\). In other words, \( \mathcal{Y} \) yields an isomorphic copy of \( \mathcal{C} \) sitting inside the functor category as a full sub-category.

Why full? Because we know that for any pair of objects \( \mathcal{Y}A \) and \( \mathcal{Y}B \) in the surrounding category \([\mathcal{C}^{op}, \text{Set}]\), i.e. for any \( \mathcal{C}(-, A) \) and \( \mathcal{C}(-, B) \), every arrow (natural transformation between them) is of the form \( \mathcal{C}(-, f) \), for some \( f : A \to B \) in \( \mathcal{C} \). But the functor \( \mathcal{Y} \) acting on \( \mathcal{C} \) provides all these natural transformations.

So, in a phrase, \( \mathcal{Y} \) embeds a copy of \( \mathcal{C} \) in \([\mathcal{C}^{op}, \text{Set}]\). Hence the terminology (in honour of its discoverer):

**Definition 39.** The full and faithful functor \( \mathcal{Y} : \mathcal{C} \to [\mathcal{C}^{op}, \text{Set}] \) is the Yoneda embedding of \( \mathcal{C} \).
There was a reason, then, behind our use of ‘\( \mathcal{Y} \)’ for this functor! And indeed the ‘\( \mathcal{Y} \)’ notation – in upper or lower case, in one font or another – is pretty standard for the Yoneda embedding. However, ‘\( \mathcal{X} \)’ is just our label for the dual embedding, which doesn’t seem to have a standard name or notation, though we’ll can usefully call it a Yoneda embedding too. (Alternative notations include Leinster’s ‘\( \mathcal{H}_s \)’ for the Yoneda embedding \( \mathcal{Y} \), with ‘\( \mathcal{H}^* \)’ available for the dual.)
An aside on Cayley’s Theorem

Let’s pause for breath and ask: what makes the Restricted Yoneda Lemma and the related Yoneda embedding interesting?

Here’s one rather general thought. Take any locally small category you like. Then the Yoneda embedding tells us how to find a category built from functors-into-$\textbf{Set}$-and-arrows-between-them which looks just like the category we started off with.

Now, this is surely reminiscent of some classical representation theorems which tell us how, given a mathematical structure of a certain type, we can find another structure which lives in the universe of sets and is isomorphic to it. At the simple end of the spectrum there is an observation that we can attribute to Dedekind: any given partially ordered objects are isomorphic to certain corresponding sets ordered by set-inclusion. A significantly more sophisticated result of the same flavour is the Stone Representation Theorem: any Boolean algebra is isomorphic to a field of sets (where a field of sets is a sub-algebra of a canonical power-set algebra $(\mathcal{P}(X), \overline{\cdot}, \cap, \cup, \emptyset, X)$, where $X$ is some set and of course $\overline{A}$ is $X - A$).

In this short chapter, however, we’ll concentrate on just one such classical representation theorem, namely . . .

\section{Cayley’s Theorem}

\textbf{Theorem 45.} Any group $(G, \cdot)$ is isomorphic to a subgroup of the group $\text{Sym}(G)$, i.e. the group of permutations on the set $G$.

\textit{Proof.} (The usual one, just rehearsed as a reminder!) Given any object $g \in G$, we define the set-function $g: G \to G$ by setting $g(x) = g \cdot x$ (i.e. $g = \{ (x, y) \mid x, y \in G \land y = g \cdot x \}$).

Evidently any such $g$ is surjective: for any $x \in G$, there’s an object which $g$ sends to $x$, namely $g^{-1} \cdot x$. And if $g(x) = g(y)$, then $g \cdot x = g \cdot y$ whence $g^{-1} \cdot g \cdot x = g^{-1} \cdot g \cdot y$, therefore $x = y$. Hence $g$ is also an injection and is therefore a bijection on $G$, i.e. is a permutation of the group objects.

Put $K = \{ g \mid g \in G \}$. It is now routine to confirm $(K, \circ)$ is a group, and hence a subgroup of $\text{Sym}(G)$, where the group operation is composition of functions:

i. Any two functions $f, g$ have a product $f \circ g$, where $(f \circ g)(x) = f \cdot g \cdot x$.

ii. The function $e$ is a group identity, where $e$ is the identity in $(G, \cdot)$.

iii. $f \circ (g \circ h) = (f \circ g) \circ h$ because $f \cdot (g \cdot h) = (f \cdot g) \cdot h$. 

69
iv. We note that \((g^{-1} \circ g)(x) = g^{-1}(g \cdot x) = x = e(x)\). So \(g^{-1} \circ g = e\), and similarly \(g \circ g^{-1} = e\). So each \(g\) has an inverse.

It remains to check that the map \(F\) defined by \(g \mapsto \overline{g}\) is a group isomorphism from \((G, \cdot) \to (K, \circ)\). \(F\) is injective. For if \(f = g\), then \(f(e) = g(e)\), so \(f \cdot e = g \cdot e\), so \(f = g\). Since \(F\) is also a surjection just by the definition of \(K\), \(F\) (as a map on the carries sets) is an isomorphism.

Also, for any \(x\), \(F(f \cdot g)(x) = (f \cdot g)(x) = f \cdot g \cdot x = f(g(x)) = f(g \cdot x) = (f \cdot g)(x) = (Ff \circ Fg)(x)\), so \(F\) indeed respects group structure.

Now, the modern way is – at least at least officially – to think of a group \((G, \cdot)\) as a set-theoretic structure from the outset; so Cayley’s theorem might seem just to tell us that, given one set theoretic structure, we can find another isomorphic one. Big deal! However, that rather disguises what’s actually going on.

For various reasons – some good, some rather disreputable – it has become absolutely standard in mathematics to trade in a lot of plural talk (referring to many objects at once) for singular talk (referring to a set of those many objects). For example, we’ve learnt to slide easily e.g. from talk of the natural numbers (plural) to talk of the set \(\mathbb{N}\) (singular). And so instead of stating the Least Number Principle as e.g. ‘Given any natural numbers, one of them will be the least’ we say ‘Any set \(S\), where \(S \subseteq \mathbb{N}\), has a least member’. But note that the singular talk about a set here is not yet doing any real work. Indeed, quite a lot of informal set talk is in fact similarly low-level, non-committal stuff – i.e. is talk about merely virtual classes – so is a familiar way of speaking which can however be readily translated away, most naturally into a plural idiom. That applies here, to part of the statement of Cayley’s Theorem. Instead of starting ‘Any group \((G, \cdot), \ldots\)’ and reading this as referring to a set-theoretic object (e.g. an ordered pair of a set and a set-function), we can capture the core of the theorem like this:

Suppose we are given some objects and a group operation on them. Then there will be always also be some sets (in particular, some set-functions) with a group structure on them which form a group isomorphic to the one we started with.

Put this way, stripped of a layer of unnecessary set-idiom, we have (in an intuitive sense) a ‘cross-category’ result which says that objects with a group structure on them (whatever objects they are) can always be represented by an isomorphic structure living in the world of sets.

### 9.2 Yoneda meets Cayley

Recall from §1.2 ex.(6) that a monoid can be considered as a category in its own right. This will apply, of course, to those monoids which are groups.

Let’s spell this out. Take a group \((G, \cdot)\), where we assume that \(G\), the collection of group elements, is indeed set-sized. And then consider the corresponding category \(\mathcal{G}\) with the following data:

i. the sole object of \(\mathcal{G}\): choose whatever object you like, and dub it ‘\(*\)’.

ii. the arrows of \(\mathcal{G}\) are the elements of the group \((G, \cdot)\).

iii. the identity arrow \(1_\star\) of \(\mathcal{G}\) is the identity element \(e\) of the group \(G\).

iv. the composite \(g \circ f: \star \to \star\) of the two arrows \(g, f: \star \to \star\) is just \(g \cdot f\).
Now, suppose that the element \( a^{-1} \) is the inverse of \( a \) in the group \((G, \cdot)\). Then the arrow \( a^{-1} \) is the inverse of the arrow \( a \) in the category \( \mathcal{C} \). Therefore, since every group element in the \((G, \cdot)\) has an inverse, every \( \mathcal{C} \)-arrow has an inverse. Which proves

**Theorem 46.** A group considered as a category is a one-object category all of whose arrows are isomorphisms.

\( \mathcal{G} \) is locally small since its sole potential hom-set \( \mathcal{G}(\star, \star) \) is none other than \( G \), which we assumed is indeed set-sized. We can therefore apply the Restricted Yoneda Lemma in one version or the other. And there’s only one possible application of each version. Consider the version which tells us that

\[
\text{Nat}(\mathcal{G}(-, \star), \mathcal{G}(-, \star)) \cong \mathcal{G}(\star, \star).
\]

So: what are the natural transformations \( \alpha : \mathcal{G}(-, \star) \to \mathcal{G}(-, \star) \)? We can apply Theorem 38: every such \( \alpha \) is \( \mathcal{G}(-, a) \) for some arrow \( a \) in \( \mathcal{G} \).

Now, by definition: \( \mathcal{G}(-, a) \) sends an arrow \( j : \star \to \star \) to \( a \circ j : \star \to \star \). But \( \mathcal{G}(\star, \star) \) is just \( G \), and arrows are \( G \)-elements, so \( \mathcal{G}(-, a) \) acts on \( G \) by sending an element \( j \) to the element \( a \circ j \), which is the function we earlier called \( g(a) \). But as before, that’s a bijective map on \( G \), i.e. a permutation on \( G \). Therefore our natural transformations are permutations on \( G \). Hence, so far, we know that \( G \) (as a set) is isomorphic to some set of permutations on \( G \).

Moreover, our proof of the Lemma gives us the isomorphism \( \mathcal{Y} \), which sends the arrow \( a : \star \to \star \) to \( \mathcal{G}(-, a) \). So by Theorem 36,

\[
\mathcal{Y}(a \cdot b) = \mathcal{Y}(a \circ b) = \mathcal{G}(-, a \circ b) = \mathcal{G}(-, a) \circ \mathcal{G}(-, b) = \mathcal{Y}(a) \circ \mathcal{Y}(b).
\]

So if we put a group structure on the natural transformations \( \mathcal{G}(-, a) \) by using composition as the group operation (check it is a group operation in this case!), our isomorphism \( \mathcal{Y} \) preserves group structure.

So in short, we can more or less immediately read off from the proof of the Restricted Yoneda Lemma that a group \((G, \cdot)\) is isomorphic as a group to a group of permutations on \( G \) with composition as the group operation.

Which is why it is often said that the Yoneda Lemma is a generalization of Cayley’s Theorem.
In Chapter 8 we proved that the Yoneda embedding and its dual are indeed embeddings (in a very natural sense) by appeal to a couple of fairly easy results, Theorems 39 and 40. We called the combination of these theorems, for want of a standard label, the Restricted Yoneda Lemma.

Which invites the question: what’s the unrestricted Yoneda Lemma? This chapter explains.

10.1 Towards the full Yoneda Lemma

Let $F$ be the functor $\mathcal{C}(B,-)$. Then one half of the Restricted Yoneda Lemma tells us that there is an isomorphism between $\text{Nat}(\mathcal{C}(A,-), F)$ and $F(A)$. (The other half of the Restricted Lemma is the dual of course: for the moment, we’ll let it look after itself!)

Now, to get from where we are to the Yoneda Lemma proper we need two (again easy) observations:

1. Look again at the ingredients of the proof of the restricted version and ask ‘Where did we depend on the fact that the second functor, now notated simply ‘$F’$, actually had the form $\mathcal{C}(B,-)$ for some $B$?’ Inspection shows that we didn’t! So we in fact have the more general result that for any locally small category $\mathcal{C}$, any functor $F: \mathcal{C} \to \text{Set}$, and any object $A \in \mathcal{C}$, there is an isomorphism $\mathcal{E}$ between $\text{Nat}(\mathcal{C}(A,-), F)$ and $F(A)$.

2. Note too that our proof provided a general recipe for constructing the required isomorphism. Take a locally small category $\mathcal{C}$ and any $\mathcal{C}$-object $A$, then, without having to invoke any arbitrary choices, our proof fixes inverse isomorphisms $\mathcal{X}_{AF}$ and $\mathcal{E}_{AF}$ between $\text{Nat}(\mathcal{C}(A,-), F)$ and $F(A)$ (at least for the case where $F$ is a functor of the form $\mathcal{C}(B,-)$). In an intuitive sense, we’ve constructed a natural isomorphism. And so we should be able to show that there is a natural isomorphism in the official, categorial, sense between some relevant functors.

In sum, we will get from the Restricted Yoneda Lemma to the full-dress Yoneda Lemma in two steps, one that involves a generalizing move, and one that involves recasting in category-theoretic terms an intuitive judgement of the naturality of our construction. Neither step involves anything conceptually difficult: we just need to nail down the tiresome details. (So you could indeed perhaps, if you want, just skim the details of the proofs in rest of this chapter on a first reading)
10.2 The generalizing move

We continue working in a locally small category $\mathcal{C}$. Let’s restate what we know so far, temporarily using ‘$F$’ to abbreviate ‘$\mathcal{C}(B, -)$’.

Theorems 33 and 37 tell us that there’s a bijection between $\mathcal{C}$-arrows $f: B \to A$ and natural transformation $\mathcal{C}(f, -); \mathcal{C}(A, -) \Rightarrow F$, where for each $Z$, the component $\mathcal{C}(f, -)_Z$ maps an arrow $g: A \to Z$ to $g \circ f: B \to Z$.

Now by definition, the functor $F$ maps an arrow $g: A \to Z$ to a function which sends an arrow $f: B \to A$ to the arrow $g \circ f: B \to Z$. Hence $F(g)(f) = g \circ f$.

Putting those two remarks together, we get this: there’s a bijection which sends $\mathcal{C}(f, -)_Z: \mathcal{C}(A, Z) \to \mathcal{C}(B, Z)$, for any $Z$, the component $\mathcal{C}(f, -)_Z$ maps $g: A \to Z$ to $F(g)(f)$.

OK, with that preliminary, here’s the announced generalization of the Restricted Yoneda Lemma where we free up the interpretation of $F$ to encompass any functor from $\mathcal{C}$ to $\text{Set}$:

**Theorem 47.** For any locally small category $\mathcal{C}$, object $A \in \mathcal{C}$ and functor $F: \mathcal{C} \to \text{Set}$, $\text{Nat}(\mathcal{C}(A, -), F) \cong F(A)$.

**Proof.** Following the construction in the proof leading up to Theorem 39, define $\theta$ by requiring $\theta(\alpha) = \alpha_A(1_A)$, for any $\alpha: \mathcal{C}(A, -) \Rightarrow F$.

And define $\psi$ by requiring that, for an element $f \in F(A)$, the value of $\psi(f)$ is the natural transformation whose component $\psi(f)_Z$ sends a map $g: A \to Z$ to $F(g)(f)$. (In the light of our preliminary observation above, $\psi(f)$ is just a generalized version of $\mathcal{C}(f, -)$ and is readily checked to still be a natural transformation.)

We show that $\psi$ is a two-sided inverse of $\theta$. First, given an arbitrary element $f \in F(A)$,

$$\theta \circ \psi) f = \psi(f)_{A}(1_A) = F(1_A)(f) = 1_{F(A)}(f) = f,$$

and therefore $\theta \circ \psi = 1_{F(A)}$.

Secondly, for $\alpha: \mathcal{C}(A, -) \Rightarrow F$, we have $\psi \circ \theta(\alpha)_Z = \psi(\alpha_A(1_A))_Z$, which sends a map $g: A \to Z$ to $F(g)(\alpha_A(1_A))$. But since $\alpha$ is a natural transformation, this next diagram must commute:

$$
\begin{array}{ccc}
\mathcal{C}(A, A) & \xrightarrow{\mathcal{C}(A,g)} & \mathcal{C}(A, Z) \\
\downarrow{\alpha_A} & & \downarrow{\alpha_Z} \\
F(A) & \xrightarrow{F(g)} & FZ
\end{array}
$$

So $F(g(\alpha_A(1_A))) = \alpha_Z(\mathcal{C}(A,g)(1_A)) = \alpha_Z(g)$.

Which shows that $\psi \circ \theta(\alpha)_Z$ is the map which sends any $g$ to $\alpha_Z(g)$, i.e. is $\alpha_Z$. So $\psi \circ \theta = 1_{Nat}$.

We have shown, then, that $\theta: \text{Nat}(\mathcal{C}(A, -), F) \to F(A)$ has two-sided inverse, i.e. is an isomorphism, and so we are done.

\[\square\]

10.3 Making it all natural

A second step takes us to the full Yoneda Lemma. Not only is there an isomorphism we’ll now label $\theta_{AF}$ from $\text{Nat}(\mathcal{C}(A, -), F)$ to $F(A)$, but $\theta_{AF}$ is intuitively ‘natural’ (constructed in a uniform way given $A$ and $F$, without arbitrary choices). We now want to capture this intuitive remark using our official categorial account of a natural isomorphism.

Here’s a reminder:
Definition 24 Given functors $F, G : \mathcal{C} \to \mathcal{D}$, we say that $FA \cong GA$ naturally in $A$ (or naturally in $A$ in $\mathcal{C}$) just if $F$ and $G$ are naturally isomorphic.

What we want to prove first, keeping $F$ fixed, is that $Nat(\mathcal{C}(A, -), F) \cong F(A)$ naturally in $A \in \mathcal{C}$. Which, by our definition, means we have to establish something along these lines:

Theorem 48. Let $\mathcal{C}$ be a locally small category, and $F$ a functor $F : \mathcal{C} \to \text{Set}$. Then the functors $Nat(\mathcal{C}(-, -), F)$ and $F$ are naturally isomorphic.

But hold on! We haven’t yet met a functor $Nat(\mathcal{C}(-, -), F) : \mathcal{C} \to \text{Set}$! What is it? It’s the functor $N$ (for short) with the following components:

\begin{enumerate}
  \item $N$ sends any $\mathcal{C}$-object $A$ to the set $Nat(\mathcal{C}(A, -), F)$.
  \item $N$ sends any $\mathcal{C}$-arrow $f : A \to B$ to an arrow between the sets $Nat(\mathcal{C}(A, -), F)$ and $Nat(\mathcal{C}(B, -), F)$. Which arrow? The obvious candidate is the one that sends $\alpha : \mathcal{C}(A, -) \Rightarrow F$ to $\alpha \circ \mathcal{C}(f, -) : \mathcal{C}(B, -) \Rightarrow F$.
\end{enumerate}

Let’s check that this definition works! Given $f : A \to B$ (NB the order of $A$ and $B$ here!), we have $\mathcal{C}(f, -) : \mathcal{C}(B, -) \Rightarrow \mathcal{C}(A, -)$. So yes, $\alpha \circ \mathcal{C}(f, -)$ is a natural transformation (by vertical composition). So now to check $N$ is functorial.

\begin{enumerate}
  \item For any $A \in \text{ob}(\mathcal{C})$, $N(1_A)$ is the arrow that sends any $\alpha : \mathcal{C}(A, -) \Rightarrow F$ to $\alpha \circ \mathcal{C}(1_A, _) : \mathcal{C}(A, -) \Rightarrow F$. But $\mathcal{C}(1_A, -) = 1_{\mathcal{C}(A, -)}$ (by Theorem 34). So $N(1_A)$ is the identity arrow on the the set $Nat(\mathcal{C}(A, -), F)$, i.e. $1_{N(A)}$.
  \item For any $\mathcal{C}$-arrows $f : A \to B$ and $g : B \to C$, $(Ng \circ Nf)\alpha = Ng(\alpha \circ \mathcal{C}(f, -)) = \alpha \circ \mathcal{C}(g \circ f, -) = (Ng \circ f)\alpha$ (by Theorem 34 at the penultimate step). Hence $N(g \circ f) = Ng \circ Nf$.
\end{enumerate}

Hence there really is functor $N = Nat(\mathcal{C}(-, -), F)$ and at least the statement of our theorem is in good order. So now we need a …

Proof. Given any $f : A \to B$, consider the following diagram,

$$
\begin{array}{ccc}
Nat(\mathcal{C}(A, -), F) & \xrightarrow{Nf} & Nat(\mathcal{C}(B, -), F) \\
\downarrow{\theta_{AF}} & & \downarrow{\theta_{BF}} \\
F(A) & \xrightarrow{F(f)} & F(B)
\end{array}
$$

Now take any $\alpha : \mathcal{C}(A, -) \Rightarrow F$. Then we have:

\begin{enumerate}
  \item $\theta_{BF} \circ Nf(\alpha) = \theta_{BF}(\alpha \circ \mathcal{C}(f, -)) = (\alpha \circ \mathcal{C}(f, -))B(1_B) = \alpha_B(f)$ (for the last equation, compare the end of the proof of Theorem 37).
  \item But also $F(f) \circ \theta_{AF}(\alpha) = F(f)(\alpha_A(1_A)) = \alpha_B \circ \mathcal{C}(A, f)(1_A) = \alpha_B(f)$ (for the middle equation look at the commutative diagram at the end of the proof of Theorem 47, with trivial relabelling).
\end{enumerate}

So our diagram will always commute, and hence there is a natural isomorphism $\theta_F : N \Rightarrow F$ with components $(\theta_F)_A = \theta_{AF}$ for each $A \in \mathcal{C}$, and our theorem is proved.

That captures in categorial terms the intuition that the construction of $\theta_{AF}$ depends in a natural way on $A$: now for the companion intuition that it depends in a natural way on $F$ too. Keeping $A$ fixed, we want to prove $Nat(\mathcal{C}(A, -), F) \cong F(A)$ naturally in $F$ (where $F$ belongs to the functor category $[\mathcal{C}, \text{Set}]$). Which, by definition, means we want to prove something of this shape:
Theorem 49. Let \( \mathcal{C} \) be a locally small category. Then \( \text{Nat}(\mathcal{C}(A,-), \cdot) \) and \( \cdot(A) \) are naturally isomorphic, where these are appropriate functors from \( \mathcal{C}, \mathbf{Set} \) to \( \mathbf{Set} \).

So of course, our first task, again, is to define the appropriate functors to work with.

We need \( \text{Nat}(\mathcal{C}(A,-), \cdot) \), or \( K \) for short, to act on an object in the functor category \( [\mathcal{C}, \mathbf{Set}] \), i.e. on a functor \( F : \mathcal{C} \to \mathbf{Set} \) and send it to \( \text{Nat}(\mathcal{C}(A,-), F) \). And how should \( K \) act on arrows in the functor category, i.e. on natural transformations \( \gamma : F \Rightarrow G \) where \( \mathcal{C} \xrightarrow{\cdot} \mathbf{Set} \)? Well, the obvious option is to set the value of \( K(\gamma) \) to be the map that sends \( \alpha : \mathcal{C}(A,-) \Rightarrow F \) to \( \gamma \circ \alpha : \mathcal{C}(A,-) \Rightarrow G \). It is easily checked that \( K \) is then functorial.

Likewise, we need the functor \( \cdot(A) \) to act on an object in the functor category, i.e. on a functor \( F : \mathcal{C} \to \mathbf{Set} \), by sending it to \( F(A) \). So we can think of \( \cdot(A) \) as a functor which evaluates a functor \( F \) at the object \( A \), so let’s write it more helpfully as \( \text{ev}_{A} \).

Then \( \text{ev}_{A}(F) = F(A) \), and we can put \( \text{ev}_{A}(\gamma) = \gamma_{A} \) to get a functor, as again is easily checked.

So, having explained our notation in stating our theorem, we now need another . . .

**Proof.** Given any \( \gamma : F \Rightarrow G \), consider the following diagram,

\[
\begin{array}{ccc}
\text{Nat}(\mathcal{C}(A,-), F) & \xrightarrow{K(\gamma)} & \text{Nat}(\mathcal{C}(A,-), G) \\
\downarrow{\theta_{AF}} & & \downarrow{\theta_{AG}} \\
F(A) & \xrightarrow{\text{ev}_{A}(\gamma)} & G(A)
\end{array}
\]

Then again this diagram commutes. For take any \( \alpha : \mathcal{C}(A,-) \Rightarrow F \). Then we have:

1. \( \theta_{AG} \circ K(\gamma)(\alpha) = \theta_{AG}(\gamma \circ \alpha) = (\gamma \circ \alpha)_{A}(1_{A}) = \gamma_{A}(\alpha_{A}(1_{A})) \).
2. But also \( \text{ev}_{A}(\gamma) \circ \theta_{AF}(\alpha) = \gamma_{A}(\alpha_{A}(1_{A})) \).

Since the diagram always commutes, there is a natural isomorphism \( \theta_{A} : K \Rightarrow \text{ev}_{A} \) with components \( (\theta_{A})_{F} = \theta_{AF} \) for each \( F \in [\mathcal{C}, \mathbf{Set}] \). So we are done. \( \square \)

**10.4 Putting everything together**

So now combine all the ingredients from the last three theorems . . .

Cue drum-roll!

. . . and we at last have the full-dress result:

**Theorem 50** (Yoneda Lemma). For any locally small category \( \mathcal{C} \), object \( A \in \mathcal{C} \), and functor \( F : \mathcal{C} \to \mathbf{Set} \), \( \text{Nat}(\mathcal{C}(A,-), F) \cong F(A) \), both naturally in \( A \in \mathcal{C} \) and naturally in \( F \in [\mathcal{C}, \mathbf{Set}] \).

There will evidently be a dual version too (involving contravariant functors in \( \mathcal{C} \), i.e. functors in \( \mathcal{C}^{\text{op}} \)):

**Theorem 51** (Yoneda Lemma). For any locally small category \( \mathcal{C} \), object \( A \in \mathcal{C} \), and functor \( F : \mathcal{C}^{\text{op}} \to \mathbf{Set} \), \( \text{Nat}(\mathcal{C}(-,A), F) \cong F(A) \), both naturally in \( A \in \mathcal{C} \) and naturally in \( F \in [\mathcal{C}^{\text{op}}, \mathbf{Set}] \).
Some authors call only the second version the Yoneda Lemma: we’ll use the label for both, talking of the covariant and contravariant versions if we need to mark the distinction.

And having done all this work, we see that a further generalization is in principle possible. We’ve so far been working with locally small categories, i.e. categories whose classes of arrows between pairs of objects are indeed sets which live in \textbf{Set}. Suppose we turn our attention to larger categories whose hom-classes (as we could naturally call them) are some bigger collections which live in a suitably well-behaved category \textbf{Set} which allows bigger collections. Then we can re-run our arguments to show that for a category $\mathcal{C}$ with hom-classes in \textbf{Set}, $A \in \mathcal{C}$, and a functor $F: \mathcal{C} \to \text{set}$, then the $\textbf{Set}$ of natural transformations from $\mathcal{C}(A,-)$ to $F$ is in natural isomorphism with $F(A)$.

But we won’t delay over this further generalization – indeed, will we have occasion to use it?
11.1 Presheaves, diagrams, representables

Definition 40. Given a category $\mathcal{C}$ then we say

i. A contravariant functor from $\mathcal{C}$ to $\textbf{Set}$, i.e. a functor $F: \mathcal{C}^{\text{op}} \to \textbf{Set}$, is a \textit{presheaf} on $\mathcal{C}$.

ii. The presheaves on $\mathcal{C}$ (as objects) and the natural transformations between them (as arrows) form the \textit{presheaf category on} $\mathcal{C}$, denoted $\hat{\mathcal{C}}$.

Four quick comments on these definitions:

(1) The terminology ‘presheaf’ comes from an example in topology. But here we can just take it as an arbitrary (though now standard) label.

(2) We can also talk more generally about a $\mathcal{V}$-valued presheaf on $\mathcal{C}$, i.e. a functor $F: \mathcal{C}^{\text{op}} \to \mathcal{V}$. An interesting case might be when $\mathcal{V} = \textbf{Set}$, i.e. is a category with larger collections than sets (see the end of §10.4). Then by talking of $\textbf{Set}$-valued presheaves we could then extend results about locally small categories and presheaves in our initial sense. But we won’t need to pursue this.

(3) $\hat{\mathcal{C}}$ is just a relabelling of the functor category we met in §8.5 and called $[\mathcal{C}, \textbf{Set}]$. And so the Yoneda embedding $\mathcal{Y}$ we met there is a functor $\mathcal{Y}: \mathcal{C} \to \hat{\mathcal{C}}$; and in our new notation, assuming the Axiom of Choice, we can say that $\mathcal{C}$ is isomorphic to a full subcategory of $\hat{\mathcal{C}}$.

(4) Recall $\mathcal{Y}A = \mathcal{C}(\cdot, A)$. Hence $\mathcal{C}(\mathcal{Y}A, F)$ is the hom-class of the presheaf category $\hat{\mathcal{C}}$ which comprises the arrows of that functor category from $\mathcal{C}(\cdot, A)$ to $F$, i.e. it is $\text{Nat}(\mathcal{C}(\cdot, A), F)$. That’s why (one version) of the Yoneda Lemma can be presented like this: on the usual assumptions, $\hat{\mathcal{C}}(\mathcal{Y}A, F) \cong FA$, naturally in both $A \in \mathcal{C}$ and $F$ in $\hat{\mathcal{C}}$.

As you would expect, alongside the alternative terminology of ‘presheaves’ for contravariant functors, there’s terminology for the dual covariant functors. Back in §5.2 we said, informally, a functor from one category $\mathcal{J}$ to another category $\mathcal{C}$ projects a ‘picture’ or ‘diagram’ of $\mathcal{J}$ into $\mathcal{C}$. That informal way of speaking prefigures some more standard jargon:

Definition 41. Given a category $\mathcal{C}$, we say that a functor into it, i.e. a functor $D: \mathcal{J} \to \mathcal{C}$, is a \textit{diagram} – or more explicitly, is a \textit{diagram of shape} $\mathcal{J}$ \textit{in} $\mathcal{C}$.
(Our definitions reflect that we are going to be particularly interested in contravariant functors from a given category \( \mathcal{C} \) and covariant functors into \( \mathcal{C} \).

How does this last notion, of diagrams-as-functors, relate to the idea(s) of a diagram as originally introduced in §1.3? Think of the case where \( \mathcal{J} \) is a very small category, a few objects and arrows. Then the functor \( D: \mathcal{J} \to \mathcal{C} \) will send \( \mathcal{J} \) to a corresponding handful of objects and arrows sitting inside \( \mathcal{C} \), i.e. the sort of fragment that we called a diagram in \( \mathcal{C} \). Diagrams-as-functors deliver diagrams-in-categories.

**Definition 42.** If \( \mathcal{C} \) is a locally small category, then

i. A (set-valued) presheaf \( F: \mathcal{C}^{\text{op}} \to \text{Set} \) which is naturally isomorphic to the contravariant hom-functor \( \mathcal{C}(–,A) \) for some \( A \in \mathcal{C} \) is said to be representable, and \( A \) is said to represent it.

ii. Dually, a set-valued diagram \( F: \mathcal{C} \to \text{Set} \) which is naturally isomorphic to the covariant hom-functor \( \mathcal{C}(A,–) \) is also said to be representable, and again \( A \) represents it.

iii. Representable functors are often just called representables.

We immediately note an important result about representing objects (we state one version, leaving the dual as then obvious):

**Theorem 52.** Suppose \( F: \mathcal{C} \to \text{Set} \) is represented by both \( A,B \in \mathcal{C} \). Then \( A \cong B \).

**Proof.** By our supposition, we have both \( F \cong \mathcal{C}(A,–) \) and \( F \cong \mathcal{C}(B,–) \), hence \( \mathcal{C}(A,–) \cong \mathcal{C}(B,–) \), so \( \mathcal{X}A \cong \mathcal{X}B \). Apply Theorem 44. \( \square \)

### 11.2 Is the forgetful functor \( \text{Mon} \to \text{Set} \) representable?

Trivially, hom-functors themselves are representables. But now let’s find some other examples.

Start with a simple example, indeed the very first functor we met back in §3.2. The forgetful functor \( F: \text{Mon} \to \text{Set} \) sends any monoid \( (M,\cdot) \) to its underlying set \( M \), and sends a monoid homomorphism \( f: (M,\cdot) \to (N,\times) \) to the same function \( f: M \to N \) in Set.

Question: is there a representing monoid \( R \) such that the hom-functor \( \text{Mon}(R,–) \) is naturally isomorphic to the forgetful \( F \)?

Applying the usual definition, \( \text{Mon}(R,–) \) sends a monoid \( M \) in \( \text{Mon} \) to \( \text{Mon}(R,M) \), i.e. to the set of monoid homomorphisms from \( R \) to \( M \). And it sends a monoid homomorphism \( f: M \to N \) to the set-function \( f \circ – \) which sends a function \( g: R \to M \) to the function \( f \circ g: R \to N \).

And if this functor \( \text{Mon}(R,–) \) is to be naturally isomorphic with the forgetful functor \( F \), there will have to be an isomorphism \( \psi \) with components such that, for any \( f: M \to N \) in \( \text{Mon} \), the following diagram commutes in Set:

\[
\begin{array}{ccc}
M & \xrightarrow{f} & N \\
\downarrow{\psi_M} & & \downarrow{\psi_N} \\
\text{Mon}(R,M) & \xrightarrow{f \circ –} & \text{Mon}(R,N)
\end{array}
\]
For this to work, we certainly need to choose an $R$ such that (for any monoid $M$) there is a bijection between $M$ and $\text{Mon}(R, M)$. And presumably, for the needed generality, $R$ will have to be a monoid without too much distinctive structure.

Now, the simplest such ‘boring’ monoid is the one-element monoid we can suggestively notate $\theta$ consisting of just an identity element (thinking of the monoid operation as addition, it just has a zero element). But that plainly isn’t going to work as a candidate for $R$ (there is only ever a unique member of $\text{Mon}(\theta, M)$, i.e. the homomorphism that sends the identity element of $\theta$ to the identity element of $M$, so in general there is no bijection between $\text{Mon}(\theta, M)$ and $M$).

The next simplest monoid is the free monoid with a single generator. This is the monoid we can suggestively label $\mathbb{N}$, which has an identity element ‘0’, a single generator we’ll call ‘1’, and all the freely generated ‘sums’ of these elements. Now consider a homomorphism from $\mathbb{N}$ to $M$. 0 $\in \mathbb{N}$ has to map to the identity element in $M$; and once we fix that 1 $\in \mathbb{N}$ goes to some $m \in M$ that fixes where every element of $\mathbb{N}$ (since every non-zero $\mathbb{N}$ element $1+1+1+\ldots+1$ goes to a corresponding $M$-element $m \cdot m \cdot m \cdot \ldots \cdot m$).

So consider the map $\psi_M: M \to \text{Mon}(\mathbb{N}, \text{Set})$ which maps $m$ to the homomorphism $\pi: \mathbb{N} \to M$ which sends 1 $\in \mathbb{N}$ to $m$. That’s well-defined. $\psi_M$ is evidently injective. But $\psi_M$ is also surjective on homomorphisms from $\mathbb{N} \to M$ since every such homomorphism must send 1 $\in \mathbb{N}$ to some $m$ or other, thereby determining the whole homomorphism.

Hooray! $\psi_M$ is an isomorphism in $\text{Set}$, for any $M$. And now it is easily seen that our diagram commutes. Chase an element $m \in M$ round the diagram. The route via the north-east node takes gives us $m \mapsto f(m) \mapsto f(\pi)$, the other route gives us $m \mapsto \pi \mapsto f \circ \pi = f(m)$.

Which, in summary, delivers

**Theorem 53.** The forgetful functor $F: \text{Mon} \to \text{Set}$ is representable, and is represented by $\mathbb{N}$.

### 11.3 More examples of representables

Unsurprisingly, there are analogous representation theorems for other forgetful functors. For instance:

**Theorem 54.**

1. The forgetful functor $F: \text{Grp} \to \text{Set}$ is representable, and is represented by $\mathbb{Z}$.

2. The forgetful functor $F: \text{Ab} \to \text{Set}$ is representable, and is also represented by $\mathbb{Z}$.

3. The forgetful functor $F: \text{Vect} \to \text{Set}$ (where $\text{Vect}$ is the category of vector spaces over the reals) is representable, and is represented by $\mathbb{R}$.

4. The forgetful functor $F: \text{Top} \to \text{Set}$ is representable, and is represented by the one-point topological space, call it $S_0$.

To comment on the only last of these, we simply note that a trivial continuous functions with domain $S_0$ into a space $S$ in effect picks out a single point of $S$, so the set of arrows $\text{Top}(S_0, S)$ is indeed in bijective correspondence with the set of points of $S$.

Given such examples, you might be tempted to conjecture that all such forgetful functors into $\text{Set}$ are representable. But not so. Consider $\text{FinGrp}$, the category of finite groups. Then

**Theorem 55.** The forgetful functor $F: \text{FinGrp} \to \text{Set}$ is not representable,
Proof. Suppose a putative representing group \( R \) has \( r \) members, and take any group \( G \) with \( g > 1 \) members, where \( g \) is coprime with \( r \). Then it is a known result that the only group homomorphism from \( R \) to \( G \) is the trivial one that sends everything to the identity in \( G \). But then the underlying set of \( G \) can’t be in bijective correspondence with \( \text{FinGrp}(R, G) \) as would be needed to get the analogue of the last commutative diagram for a case involving the group \( G \).

Let’s take another couple of examples. First, a pair of definitions:

**Definition 43.**

i. The (covariant) powerset functor \( P : \text{Set} \to \text{Set} \) maps a set \( X \) to its powerset \( \mathcal{P}(X) \) and maps a set-function \( f : X \to Y \) to the function which sends \( U \in \mathcal{P}(X) \) to its image \( f[U] \in \mathcal{P}(Y) \).

ii. The contravariant powerset functor \( \overline{P} : \text{Set} \to \text{Set} \) again maps a set to its powerset, and maps a set-function \( f : Y \to X \) to the function which sends \( U \in \mathcal{P}(X) \) to its inverse image \( f^{-1}[U] \in \mathcal{P}(Y) \).

We won’t pause to check the elementary fact that these are indeed functors, but rather will immediately state

**Theorem 56.** The contravariant powerset functor \( \overline{P} \) is represented by the set \( 2 = \{0, 1\} \); but the covariant powerset functor \( P \) is not representable.

Proof. As yet, we don’t have any general principles about representables and non-representables which we can invoke to prove theorems such us this. So again we need to labour through by applying definitions and seeing what we get.

If the contravariant functor \( \overline{P} \) is to be representable, then there must be a representing set \( R \) and a natural isomorphism \( \psi \) with components such that, for all set functions \( f : Y \to X \), the following diagram always commutes:

\[
\begin{array}{ccc}
\mathcal{P}X & \xrightarrow{\mathcal{P}f} & \mathcal{P}Y \\
\psi_X \downarrow & & \downarrow \psi_Y \\
\text{Set}(X, R) & \xrightarrow{\text{Set}(f,R)} & \text{Set}(Y, R)
\end{array}
\]

Now \( \text{Set}(X, R) \) is the set of set-functions from \( X \) to \( R \), whose cardinality is \(|R|^{|X|}\); and the cardinality of \( \mathcal{P}X \), i.e. \( \mathcal{P}(X) \), is \( 2^{|X|} \). So that forces \( R \) to be a two-membered set: so we pick the set \( 2 = \{0, 1\} \).

\( \text{Set}(X, 2) \) is then the set of characteristic functions for subsets of \( X \), i.e. the set of functions \( c_U : X \to \{0, 1\} \) where \( c_U(x) = 1 \) iff \( x \in U \subseteq X \). So the obvious next move is to take \( \psi_X : \mathcal{P}X \to \text{Set}(X, R) \) to be the isomorphism that sends a set \( U \in \mathcal{P}(X) \) to its characteristic function \( c_U \).

And with this choice, the diagram always commutes. Chase the element \( U \in \mathcal{P}X \) around. The route via the north-east node takes us from \( U \subseteq X \) to \( f^{-1}[U] \subseteq Y \) to its characteristic function, i.e. the function which maps \( y \in Y \) to 1 iff \( f(y) \in U \). Meanwhile, the route via the south-west node takes us first from \( U \subseteq X \) to \( c_U \). And then we apply \( \text{Set}(f, 2) \), which maps \( c_U : X \to 2 \) to \( c_U \circ f : Y \to 2 \), which again is the function which maps \( y \in Y \) to 1 iff \( f(y) \in U \). Which establishes the first half of the theorem.

For the second half of the theorem, we just note that if we try to run a similar argument for the covariant functor \( P \), we’d need to find a representing set \( R' \) such that \( \mathcal{P}X \) and \( \text{Set}(R', X) \) are always in bijective correspondence. But \( \text{Set}(R', X) \) is the set of set-functions from \( R' \) to \( X \), whose cardinality is \(|X|^{|R'|}\), while the cardinality of \( \mathcal{P}X \) is \( 2^{|X|} \). And there is no choice of \( R' \) which will make these equal for varying \( X \).
11.4 Representations, or universal elements

In §10.2 we defined the isomorphism \( \psi \) between \( FA \) and \( \text{Nat}(\mathcal{C}(A, -), F) \). Recall, for \( a \in FA \), \( \psi(a) \) is the natural transformation whose component \( \psi(a)Z \) sends a map \( f: A \to Z \) to \( F(f)(a) \).

So if \( F: \mathcal{C} \to \text{Set} \) is to be representable, then for some \( a \in FA \), \( \psi(a): \mathcal{C}(A, -) \Rightarrow F \). But \( \psi(a) \) must be not just a natural transformation but a natural isomorphism. When will this be the case? Just when each component \( \psi(a)Z: \mathcal{C}(A, Z) \Rightarrow FZ \) is a bijection. That is to say, just when for each \( z \in FZ \) there is a unique \( f: A \to Z \) such that \( \psi(a)(f) = z \), i.e. such that \( F(f)(a) = z \).

This makes the following notion of interest (even if it doesn’t yet explain the label):

**Definition 44.** A universal element (alternatively, a representation) of the functor \( F: \mathcal{C} \to \text{Set} \) is a pair \( \langle A, a \rangle \), where \( A \in \mathcal{C} \) and \( a \in FA \), where for each \( Z \in \mathcal{C} \) and \( z \in FZ \), there is a unique map \( f: A \to Z \) such that \( F(f)(a) = z \).

Dually, a universal element of the contravariant functor \( F: \mathcal{C}^{\text{op}} \to \text{Set} \) is a pair \( \langle A, a \rangle \), where \( A \in \mathcal{C} \) and \( a \in FA \), where for each \( Z \in \mathcal{C} \) and \( z \in FZ \), there is a unique map \( f: Z \to A \) such that \( F(f)(a) = z \).

We then have the following result (and its obvious dual):

**Theorem 57.** A functor \( F: \mathcal{C} \to \text{Set} \) is representable iff it has a representation (a universal element).

**Proof.** Our preceding remarks show that if \( F: \mathcal{C} \to \text{Set} \) is representable then it has a universal element. So it remains to prove the converse.

Suppose \( \langle A, a \rangle \) is a universal element for \( F \). Then, \( a \in FA \), and by definition \( \psi(a) \) is the natural transformation from \( \mathcal{C}(A, -) \) to \( F \) whose component \( \psi(a)Z: \mathcal{C}(A, Z) \to FZ \) sends a map \( f: A \to Z \) to \( F(f)(a) \).

But then, by the condition on universality, there’s a function \( \delta(a)_Z \) which sends \( z \in FZ \) to the unique \( f: A \to Z \) in \( \mathcal{C}(A, Z) \) where \( F(f)(a) = z \). And we can immediately see that \( \psi(a)_Z \) and \( \delta(a)_Z \) are inverses. So each component \( \psi(a)_Z \) is an isomorphism, and hence \( \psi(a) \) is a natural isomorphism from \( \mathcal{C}(A, -) \) to \( F \) witnessing that \( F \) is representable. \( \Box \)

11.5 Categories of elements

Next, we note that the universal elements of \( F \) are objects in a broader category:

**Definition 45.** Elts\(_\mathcal{C}(F)\), the category of elements of the functor \( F: \mathcal{C} \to \text{Set} \), has the following data:

1. Objects are the pairs \( \langle A, a \rangle \), where \( A \in \mathcal{C} \) and \( a \in FA \).
2. An arrow from \( \langle A, a \rangle \) to \( \langle B, b \rangle \) is a \( \mathcal{C} \)-arrow \( f: A \to B \) such that \( F(f)(a) = b \).
3. The identity arrow on \( \langle A, a \rangle \) is \( 1_A \).
4. Composition of arrows is induced by composition of \( \mathcal{C} \)-arrows.

It is easily checked that this is a category. (Alternative symbolism for the category includes variations on ‘\( \text{f} F \)’.)

Now, the standard name ‘category of elements of \( F \)’ is initially puzzling – after all, functors don’t in a straightforward sense have elements. But we can perhaps throw some light on the name as follows.
(i) Suppose we are given a category $\mathcal{C}$ whose objects are sets (perhaps with some additional structure on them) and whose arrows are functions between sets. Then there will be a derived category – or rather, some derived categories – whose objects are (or involve) elements of $\mathcal{C}$’s objects, and whose arrows between these elements are induced by the arrows between the containing sets.

Now such a category can be constructed in more than one way. But if we don’t want the derived category to forget about which elements belong to which sets, then a natural way to go would be to say that the objects of the derived category – which could be called the category of elements of $\mathcal{C}$ – are all the pairs $\langle A, a \rangle$ for $A \in \mathcal{C}$, $a \in A$. And then given elements $a \in A$, $b \in B$, whenever there is a $\mathcal{C}$-arrow $f: A \rightarrow B$ such that $f(a) = b$, we’ll say that $f$ is also an arrow from $\langle A, a \rangle$ to $\langle B, b \rangle$ in our new category. This derived category of elements in a sense unpacks what’s going on inside the original category $\mathcal{C}$.

(ii) However, in the general case, $\mathcal{C}$’s objects need not be sets so need not have elements. But a functor $F: \mathcal{C} \rightarrow \text{Set}$ gives us a diagram of $\mathcal{C}$ inside Set, and of course the objects in the resulting diagram of $\mathcal{C}$ do have elements. So we can consider the category of elements of $F$’s-diagram-of-$\mathcal{C}$, which – following the template in (i) – has as objects all the pairs $\langle FA, a \rangle$ for $A \in \mathcal{C}$, $a \in FA$. And then given elements $a \in FA$, $b \in FB$, whenever there is a Set-arrow $Ff: FA \rightarrow FB$ such that $Ff(a) = b$, we’ll say that $Ff$ is also an arrow from $\langle FA, a \rangle$ to $\langle FB, b \rangle$ in our new category.

Now, we can streamline that. Instead of taking the objects to be pairs $\langle FA, a \rangle$ take them simply to be pairs $\langle A, a \rangle$ (but where, still, $a \in FA$). And instead of talking of the arrow $Ff: FA \rightarrow FB$ we can instead talk more simply of $f: A \rightarrow B$ (but where, still, $Ff(a) = b$). And with that streamlining – lo and behold! – we are back with the category $\text{Elts}_\mathcal{C}(F)$, which is isomorphic to category of elements of $F$’s-diagram-of-$\mathcal{C}$, and which – as convention has it – we’ll call the category of elements of $F$, for short.

So the construction of $\text{Elts}_\mathcal{C}(F)$ is tolerably natural.

And it is worth noting too that this category is (isomorphic to) a certain simply characterised comma category. To explain, first let’s define another functor into Set. So let $1$ be the one-object category whose sole object is $\star$, and consider the functor $1: 1 \rightarrow \text{Set}$ which sends $\star$ to some singleton set $\ast$ and sends the identity arrow on $\star$ to the identity function on $\ast$. And then, overloading notation, let $\ast: 1 \rightarrow \text{Set}$ be the obvious inclusion functor. We then have

**Theorem 58.** For a given functor $F: \mathcal{C} \rightarrow \text{Set}$, the category $\text{Elts}_\mathcal{C}(F)$ is isomorphic to the comma category $(\star \downarrow F)$.

**Proof.** By the definition in §4.5, but changing some notation,

1. The objects of $(\star \downarrow F)$ are triples $\langle \ast, a, A \rangle$, where $A \in \mathcal{C}$ and $a: \ast \rightarrow FA$ is an arrow in Set.
2. An arrow of $(\star \downarrow F)$ from $\langle \ast, a, A \rangle$ to $\langle \ast, b, B \rangle$ is a pair $(1_\ast, f)$, where $f: A \rightarrow B$ is an $\mathcal{C}$-arrow, and the following diagram commutes:

```
    *  \rightarrow  FA
     \downarrow 1_\ast
      \downarrow
     *  \rightarrow  FB
```

$F(f)$
So $F(f)(a) = b$.

Now, an arrow $a: * \to FA$ in $\textbf{Set}$ is a function from a singleton which in effect picks out a single object $a \in FA$. Hence there is an obvious bijection between the objects $\langle *, a, A \rangle$ in $(* \downarrow F)$, where $a: * \to FA$, and the objects $\langle A, a \rangle$ in $\textbf{Elts}_F(F)$, with $a \in FA$. This bijection in turn induces an obvious isomorphism between the categories.

Having defined a category $\textbf{Elts}_F(F)$ for universal elements of $F: \mathcal{C} \to \textbf{Set}$ to live in, we can finish by asking: how do we distinguish universal elements from other elements categorically? The answer is immediate from Defn. 44:

**Theorem 59.** An object $I = \langle A, a \rangle \in \textbf{Elts}_F(F)$ is a universal element iff, for every object $E \in \textbf{Elts}_F(F)$ there is exactly one morphism $f: I \to E$.

(We can leave it as an exercise to dualize all the propositions in the last two sections.)
After the high-level abstractions of recent chapters where we have been talking about maps between categories (or even maps between these maps!), we now get back to ground-level, and introduce three pairs of categorial notions that apply inside a category. These notions will have a family resemblance to each other, and they will in fact all turn out to be our first instances of limits (or dually, colimits) in a technical sense to be defined: but that unifying idea will have to wait to Chapter 13.

12.1 Terminal and initial objects

Start with the simplest pair of cases:

**Definition 46.** The object \( I \) is an initial object of the category \( \mathcal{C} \) iff, for every \( X \in \mathcal{C} \), there is a unique arrow \( I \to X \).

Dually, the object \( T \) is a terminal object of \( \mathcal{C} \) iff, for every \( X \in \mathcal{C} \), there is a unique arrow \( X \to T \).

Some examples:

1. In the poset \((\mathbb{N}, \leq)\) thought of as a category, zero is trivially the unique initial object and there is no terminal object. The poset \((\mathbb{Z}, \leq)\) has neither initial nor terminal objects.

2. In \( \textbf{Set} \), the empty set is an initial object (cf. the comment on Ex. 1 in §1.2). Any singleton set \( \{\star\} \) is a terminal object. (For if \( X \) has members, there’s a unique \( \textbf{Set} \)-arrow which sends all the members to \( \star \); while if \( X \) is empty, then there’s a unique \( \textbf{Set} \)-arrow to any set, including \( \{\star\} \).

3. In \( \textbf{Set}_* \), the category of pointed sets (i.e. non-empty sets equipped with a distinguished member), each singleton is both initial and terminal.

4. In \( \textbf{Top} \), the empty set (considered as a trivial topological space) is the initial object. Any one-point singleton space is a terminal object.

5. In \( \textbf{Grp} \), the trivial one-element group is an initial object (just recall that a group homomorphism sends identity elements to identity elements; so there is one and only one homomorphism from the trivial group to any given group \( G \)). The same one-element group is also terminal.
(6) In the category \textbf{Bool}, the trivial one-object algebra is terminal. While the two-object algebra on \{0, 1\} familiar from propositional logic is initial – for a homomorphism of Boolean algebras from \{0, 1\} to \(B\) must send 0 to the bottom object of \(B\) and 1 to the top object, and there’s a unique map that does that.

(7) Recall: in the slice category \(\mathcal{C}/X\) an object is a \(\mathcal{C}\)-arrow like \(f: A \to X\), and an arrow from \(f: A \to X\) to \(g: B \to X\) is a \(\mathcal{C}\)-arrow \(j: A \to B\) such that \(g \circ j = f\) in \(\mathcal{C}\). Consider the \(\mathcal{C}/X\)-object \(1_X: X \to X\). A \(\mathcal{C}/X\)-arrow from \(f: A \to X\) to \(1_X\) is a \(\mathcal{C}\)-arrow \(j: A \to X\) such that \(1_X \circ j = f\), i.e. such that \(j = f\) – which exists and is unique! So \(1_X\) is terminal in \(\mathcal{C}/X\).

(8) A universal element is initial in a category of elements \(\text{Elts}_\mathcal{C}(F)\) (by Theorem 59).

Such various cases show that a category may have zero, one or many initial objects, and (independently) may have zero, one or many terminal objects. Further, an object can be both initial and terminal. There is, incidentally, a standard bit of jargon for the last case:

**Definition 47.** An object \(O \in \mathcal{C}\) is a **null object** of the category \(\mathcal{C}\) iff it is both initial and terminal.

Now, suppose \(I\) and \(J\) are both initial objects in \(\mathcal{C}\). By definition there must be unique arrows \(f: I \to J\), and \(g: J \to I\). But then \(g \circ f\) is an arrow from \(I\) to itself. Another arrow from \(I\) to itself is \(1_I\). But since \(I\) is initial, there can only be one arrow from \(I\) to itself, so \(g \circ f = 1_I\). Likewise \(f \circ g = 1_J\). Hence the unique arrow \(f\) is an isomorphism.

Relatedly, suppose \(I\) is initial, and that there is an isomorphism \(i: I \to J\). Then for any \(X\), there is a unique arrow \(f: I \to X\), and hence there is an arrow \(f \circ i^{-1}: J \to X\). Now suppose we also have \(g: J \to X\), and so \(g \circ i = f\), hence \((g \circ i) \circ i^{-1} = f \circ i^{-1}\), hence \(g = f \circ i^{-1}\). In sum, for any \(X\) there is a unique arrow from \(J\) to \(X\), thus \(J\) is also initial.

These arguments, and their duals, deliver

**Theorem 60.** Initial objects are ‘unique up to unique isomorphism’: i.e. if \(I, J \in \mathcal{C}\) are both initial, then there is a unique isomorphism \(f: I \iso J\). Dually for terminal objects.

Further, if \(I\) is initial and \(I \iso J\), then \(J\) is initial. Dually for terminal objects.

There is some sense in which category theory doesn’t care about distinguishing isomorphic objects (though what this remark comes to we’ll need to reflect on further at some point). Hence it is common to introduce notation for an arbitrary initial and terminal objects, as follows:

**Definition 48.** We use ‘0’ to denote an initial object of \(\mathcal{C}\) (assuming one exists), and likewise ‘1’ to denote a terminal object.

Three comments on this terminology:

(i) In \(\text{Set}\), 0 is \(\emptyset\), the only initial object – and \(\emptyset\) is also the von Neumann ordinal 0. While the von Neumann ordinal 1 is \(\{\emptyset\}\), i.e. a singleton, i.e. a terminal object 1. Which perhaps excuses the recycling of the notation.

(ii) We’ve also already used ‘1’ in other ways – e.g. back in §4.4, it denoted a one-point topological space. But since such one-point spaces are terminal objects in \(\text{Top}\) our notation remains consistent.
(iii) Often, null objects (objects which are both initial and terminal) are alternatively called ‘zero’ objects: but that perhaps doesn’t sit happily with using ‘0’ for an initial object: for 0 (in the sense of an initial object) typically isn’t a zero (in the sense of null) object.

12.2 Elements and generalized elements

There’s an obvious intuitive sense in which initial and terminal objects are ‘limit cases’. Before turning to consider our second introductory pair of ‘limit cases’ in this chapter, let’s pause over our initial examples a moment longer.

Take \text{Set}. For every \( X \in \text{Set} \) there is a unique arrow from \( X \) to 1. What about arrows from 1 to \( X \)? An arrow \( \vec{x} : 1 \to X \) is a set-function sending the member of the singleton 1 to some member \( x \in X \), and trivially there is exactly one such arrow for any \( x \in X \). So, in \text{Set}, we can think of talk of arrows \( \vec{x} : 1 \to X \) as the categorial version of talking of elements of \( X \). (We’ve been here before – see §§1.2, 2.2.)

Which motivates the following:

Definition 49. An element or point of \( X \in \mathcal{C} \) is an arrow \( \vec{x} : 1 \to X \) where 1 is a terminal object of \( \mathcal{C} \).

(In fact, the standard terminology for such an element is ‘global element’, picking up from a paradigm example in topology – but we won’t fuss about that.) Note, elements \( \vec{x} : 1 \to X \) are monic. For suppose \( \vec{x} \circ f = \vec{x} \circ g \); then both \( f \) and \( g \) must be morphisms \( Y \to 1 \), for the same \( Y \), hence must be the identical since 1 is terminal.

Definition 50. Suppose the category \( \mathcal{C} \) has a terminal object, and for any \( X,Y \in \mathcal{C} \), and arrows \( f,g : X \to Y \), \( f = g \) iff for all \( \vec{x} : 1 \to X \), \( f \circ \vec{x} = g \circ \vec{x} \); then \( \mathcal{C} \) is said to be well-pointed.

Think of it this way: in a well-pointed category, there are enough elements (points) to ensure that arrows which act identically on all relevant elements are indeed identical.

Theorem 61. \text{Set} is well-pointed. \text{Grp}, for example, is not.

Proof. The claim about \text{Set} is immediate.

In \text{Grp}, for example, the only homomorphism from 1 (the one-element group) to a group \( X \) sends the only element of 1 to the identity element \( e \) of \( X \); call this homomorphism \( \vec{e} \). So, take two group homomorphisms \( f,g : X \to Y \) where \( f \neq g \); still, for the only possible \( \vec{e} \), both \( f \circ \vec{e} \) and \( g \circ \vec{e} \) send the sole element of 1 to the identity element of \( Y \), so are equal.

If we like to think of arrows in categories as modelled on functions, however, we’d perhaps still like a version of the principle that these ‘generalized functions’ are identical if and only if they act identically on ‘generalized elements’ (in some sense of that term). The following definition delivers this principle:

Definition 51. A generalized element of \( X \in \mathcal{C} \) (of shape \( S \)) is an arrow \( e : S \to X \).

Trivially, for relevant \( f,g : X \to S \), if for all \( e : S \to X \) of any shape \( f \circ e = g \circ e \), then in particular \( f \circ I_X = g \circ I_X \), so indeed \( f = g \).

In terms of our new jargon, the Yoneda \( \mathcal{Y} \) embedding from (a locally small) \( \mathcal{C} \) into \([\mathcal{C}, \text{Set}]\) sends an object \( X \in \mathcal{C} \) to \( \mathcal{C}(\cdot, X) \), which is the functor which sends an object \( S \) to the set of generalized elements of \( X \) of shape \( S \). So we can think of the embedding as taking us from an object to something that encodes all the information about the (generalized) elements of that object.
12.3 Pairing schemes

Our next main topic will be a categorial treatment of products (as in Cartesian products) – so the paradigm construction we are interested in takes elements from sets \(X\) and \(Y\) and forms their ordered pairs. But what are ordered pairs?

(a) Suppose we are working in a theory of arithmetic and want to start talking about ordered pairs of natural numbers – perhaps we want to go on to use such pairs in constructing integers or rationals. Well, we can represent such pairs without taking on any new commitments by using \textit{code-numbers}. For example, if we want a bijective coding between pairs of naturals and all the numbers, we could adopt the scheme of coding the ordered pair \(\langle m, n \rangle\) by the single number \(\left\{ (m + n)^2 + 3m + n \right\}/2\). Or, if we don’t insist on every number coding a pair, we could adopt the simpler policy of putting \(\langle m, n \rangle =_{\text{def}} 2^m3^n\). Relative to a given coding scheme, we can call such code-numbers \(\langle m, n \rangle\) \textit{pair-numbers} or, by a slight abuse of terminology, simply \textit{pairs}: and we can refer to \(m\) as the first element of the pair, and \(n\) as the second element.

Why should this way of handling ordered pairs of natural numbers be regarded as somehow inferior to other, albeit more familiar, coding devices such as explicitly set-theoretic ones?

It might be said that (i) a single pair-number is really neither ordered nor a twosome; (ii) while a number \(m\) is a member of (or is one of) the pair \(m, n\), a number can’t be a genuine member of a pair-number \(\langle m, n \rangle\); and in any case (iii) coding schemes are pretty arbitrary (e.g. we could equally well have used \(3m5^n\) as a code for the pair \(m, n\)).

Which is all true. But of course we can lay \textit{exactly} analogous complaints against e.g. the familiar Kuratowski definition of ordered pairs that we all know and love. This treats the ordered pair \(m, n\) as the set \(\{\{m\}, \{m, n\}\}\). But (i) that set is not intrinsically ordered (after all, it is a \textit{set}!), nor is it always two-membered (consider the case where \(m = n\)). (ii) Even when it is a twosome, its members are not the members of the pair: in standard set theories, \(m\) cannot be a member of \(\{\{m\}, \{m, n\}\}\). And (iii) the construction again involves pretty arbitrary choices: thus \(\{n\}, \{m, n\}\) or \(\{\{m\}\}, \{\{m, n\}\}\) etc., etc., would have done just as well. On these counts, at any rate, coding pairs of numbers by using pair-numbers involves no worse a trick than coding them using Kuratowski’s standard gadget.

There is indeed a rather neat symmetry between the adoption of pair numbers as representing ordered pairs of numbers and a familiar procedure adopted by the enthusiast for working in standard ZFC. For remember that pure ZFC knows only about pure sets. So to get natural numbers into the story at all – and hence to get Kuratowski pair-sets of natural numbers – the enthusiast for sets has to choose some convenient sequence of sets to implement the numbers (or to ‘stand proxy’ for numbers, ‘simulate’ them, ‘play the role’ of numbers, or even ‘define’ them – whatever your favourite way of describing the situation is). But someone who, for her purposes, has opted to play the game this way and is dealing with natural numbers by selecting some convenient sets to implement them, is hardly in a position to complain about someone else who, for his purposes, does the opposite and treats numbers as basic, and deals with ordered pairs of them by choosing some convenient code-numbers to implement \textit{them}. Both are in the implementation game.

It might be retorted that the Kuratowski trick at least has the virtue of being an all-purpose device, available not just when you want to talk about pairs of \textit{numbers}, while e.g. the powers-of-primes coding is of more limited use. True. Similarly you can use sledgehammers to crack all sorts of things, while you can only use nutcrackers for nuts. But that’s not particularly to the point if it happens to be nuts you currently want
to crack. If we want to implement pairs of numbers without ontological inflation – e.g. in pursuing the project of ‘reverse mathematics’ – then pair-numbers are just the kind of thing we need.

(b) Let’s now think a bit more formally about what it takes to have a way of pairing-up an object \(x \in X\) with an object \(y \in Y\).

We need some objects \(O\) to serve as ordered pairs, a pairing function that sends a given \(x\) and \(y\) to a pair \(o \in O\), and a couple of functions which allow us to recover \(x\) and \(y\) from \(o\). And the point we’ve just been making is that maybe we shouldn’t care too much about the ‘internal’ nature of the objects \(O\), so long as we do have suitable associated pairing and unpairing functions. Which motivates:

**Definition 52.** Suppose \(X, Y\) and \(O\) are sets of objects (these can be the same or different). Let \(pr : X, Y \rightarrow O\) be a two-place function, while \(\pi_1 : O \rightarrow X\), and \(\pi_2 : O \rightarrow Y\), are one-place functions. Then \([O, pr, \pi_1, \pi_2]\) form a pairing scheme for \(X\) with \(Y\) iff

(a) \((\forall x \in X)(\forall y \in Y)(\pi_1(pr(x, y)) = x \land \pi_2(pr(x, y)) = y)\),

(b) \((\forall o \in O) pr(\pi_1 o, \pi_2 o) = o\).

\(O\) will be said to comprise the pair-objects of the pairing scheme, with \(pr\) the associated pairing function, while \(\pi_1\) and \(\pi_2\) are unpairing or projection functions.

Evidently, if \(O\) is the set of naturals of the form \(2^m3^n\) and \(pr(m, n) = 2^m3^n\), with \(\pi_1 o (\pi_2 o)\) returning the exponent of 2 (3) in the factorization of \(o\), then \([O, pr, \pi_1, \pi_2]\) form a pairing scheme for \(\mathbb{N} \rightarrow \mathbb{N}\).

Three initial remarks about this idea:

i. Don’t read too much into the square brackets: they need be read as no more than punctuation!

ii. Different pairs of objects are sent by \(pr\) to different pair-objects. For suppose \(pr(x, y) = pr(x', y')\). Then by (a) \(x = \pi_1(pr(x, y)) = \pi_1(pr(x', y')) = x'\), and likewise \(y = y'\).

iii. Note that by (b), \(pr\) is surjective. The ‘unpairing’ or ‘projection’ functions \(\pi_1\) and \(\pi_2\) are also surjective. For given \(x \in X\), take any \(y \in Y\) and put \(o = pr(x, y)\). Then by (a), \(x = \pi_1 o\). Likewise, given \(y \in Y\) there is an \(o \in O\) such that \(y = \pi_2 o\).

So pairing schemes basically work as you would expect.

As we’d also expect, a given pairing function fixes the two corresponding projection functions, and vice versa, in the following sense:

**Theorem 62.** (1) If \([O, pr, \pi_1, \pi_2]\) and \([O, pr', \pi_1', \pi_2']\) are both pairing schemes for \(X\) with \(Y\), then \(\pi_1 = \pi_1'\) and \(\pi_2 = \pi_2'\).

(2) If \([O, pr, \pi_1, \pi_2]\) and \([O, pr', \pi_1, \pi_2]\) are both pairing schemes, then \(pr = pr'\).

**Proof.** For (1), take any \(o \in O\). There is some (unique) \(x, y\) such that \(o = pr(x, y)\). Hence, applying (a) to both schemes, \(\pi_1 o = x = \pi_1'o\). Hence \(\pi_1 = \pi_1'\), and similarly \(\pi_2 = \pi_2'\).

For (2), take any \(x \in X\), \(y \in Y\), and let \(pr(x, y) = o\), so \(\pi_1 o = x\) and \(\pi_2 o = y\). Then by (b) applied to the second scheme, \(pr'(\pi_1 o, \pi_2 o) = o\). Whence \(pr'(x, y) = pr(x, y)\). \(\square\)

Further, there is a sense in which all schemes for pairing \(X\) with \(Y\) are equivalent. More carefully,
Theorem 63. If \([O, pr, π₁, π₂]\) and \([O', pr', π'_₁, π'_₂]\) are both schemes for pairing \(X\) with \(Y\), then there is a unique bijection \(f: O \to O'\) such that the following diagram commutes:

\[
\begin{array}{ccc}
X \times Y & \xrightarrow{pr} & O \\
\downarrow{pr'} & & \downarrow{f} \\
O' & & 
\end{array}
\]

Putting it another way, there is a unique bijection \(f\) such that, if we pair \(x\) with \(y\) using \(pr\), use \(f\) to send the resulting pair-object \(o\) to \(o'\), and then retrieve elements using \(π'_1\) and \(π'_2\), we get back to the original \(x\) and \(y\).

**Proof.** Define \(f: O \to O'\) by putting \(f(o) = pr'(π₁o, π₂o)\). Then it is immediate that \(f(pr(x, y)) = pr'(x, y)\) as required to make the diagram commute.

To show that \(f\) is injective, suppose \(f(o) = f(o')\), for \(o, o' \in O\). Then we have \(pr'(π₁o, π₂o) = pr'(π₁o', π₂o')\). Apply \(π'_1\) to each side and then use principle (a), and it follows that \(π₁o = π₁o'\). And likewise \(π₂o = π₂o'\). Therefore \(pr(π₁o, π₂o) = pr(π₁o', π₂o')\). Whence by condition (b), \(o = o'\).

To show that \(f\) is surjective, take any \(o' \in O'\). Then put \(o = pr(π'_1o', π'_2o')\). By the definition of \(f\), \(f(o) = pr'(π₁o, π₂o)\); plugging the definition of \(o\) twice into the right hand side and simplifying using rules (a) and (b) confirms that \(f(o) = o'\). Hence \(f\) is a bijection with the right properties. And since every \(o \in O\) is \(pr(x, y)\) for some \(x, y\), the requirement that \(f(pr(x, y)) = pr'(x, y)\) fixes \(f\) uniquely.

(c) Here’s another simple theorem, to motivate the final definition in this section:

**Theorem 64.** Suppose \(X, Y, O\) are sets of objects, and the functions \(π₁: O \to X\), \(π₂: O \to Y\) are such that there is a unique function \(pr: X, Y \to O\) such that (a) \((∀x \in X)(∀y \in Y)(π₁(pr(x, y)) = x ∧ π₂(pr(x, y)) = y)\). Then \([O, pr, π₁, π₂]\) form a pairing scheme.

**Proof.** We argue that the uniqueness of \(pr\) ensures that the pairing function is surjective, and then that its surjectivity implies that condition (b) from Defn. 52 holds as well as the given condition (a).

Suppose \(pr\) is not surjective. Then for some \(o \in O\), there is no \(x \in X, y \in Y\) such that \(pr(x, y) = o\). So \(pr(π₁o, π₂o) = o' \neq o\). Consider then function \(pr'\) which agrees with \(pr\) on all inputs except that \(pr'(π₁o, π₂o) = o\). Then for all cases other than \(x = π₁o, y = π₂o\) we still have \(π₁(pr'(x, y)) = x ∧ π₂(pr'(x, y)) = y\), and by construction for the remaining case \(π₁(pr'(π₁o, π₂o)) = π₁o ∧ π₂(pr'(π₁o, π₂o)) = π₂o\). So condition (a) holds for \(pr'\), where \(pr' \neq pr\). Contrapositing, if \(pr\) uniquely satisfies the condition, it is surjective.

Because \(pr\) is surjective, every \(o \in O\) is \(pr(x, y)\) for some \(x, y\). But then by (a) \(π₁o = x ∧ π₂o = y\), and hence \(pr(π₁o, π₂o) = pr(x, y) = o\). Generalizing gives us (b).

Now, informally, pairing up \(X\) with \(Y\) through a pairing scheme, gives us a product of \(X\) with \(Y\). But we don’t want to identify the resulting product simply with the set \(O\) (for it depends on the rest of the pairing scheme what role \(O\) plays). Our last theorem, however, makes the following an appropriate definition:

**Definition 53.** If \(X, Y\) are sets, then \([O, π₁, π₂]\) form a product of \(X\) with \(Y\), where \(O\) is a set, and \(π₁: X \to O, π₂: Y \to O\) are functions, so long as there is a unique two-place function \(pr: X, Y \to O\) such that (a) \((∀x \in X)(∀y \in Y)(π₁(pr(x, y)) = x ∧ π₂(pr(x, y)) = y)\). Again, don’t read too much into the brackets – though if you really insist you could take the notation as an alternative to \(\langle O, \langle π₁, π₂\rangle \rangle\) – and cf. Theorem 68.
12.4 Binary products, categorially

We have characterized pairing schemes and the resulting products they create in terms of a set of objects $O$ being the source and target of some morphisms. Which all looks highly categorial in spirit (see the preamble to Ch. 3).

But our natural story isn’t categorial as it stands. For a crucial ingredient, namely the pairing function $pr : X, Y \to O$, is a two-place function, while the arrows in a category always have just a single domain. What to do?

Suppose for a moment we are working in a well-pointed category like $\text{Set}$, where ‘elements’ in the sense of Defn. 49 do behave sufficiently like how elements intuitively should behave. In this case, instead of talking of elements $x \in X$ and $y \in Y$, we can talk of two arrows $\vec{x} : 1 \to X$ and $\vec{y} : 1 \to Y$. And suppose that for every such $\vec{x}$ and $\vec{y}$ there is a unique arrow $u : 1 \to O$ such the following commutes:

\[
\begin{array}{ccc}
1 & \xrightarrow{\vec{x}} & X \\
\downarrow{u} & & \downarrow{\pi_1} \\
O & \xrightarrow{\vec{y}} & Y \\
\end{array}
\]

Our supposition ensures that there is a unique two-place function $\vec{x}, \vec{y} \mapsto u$ such that we always have $\pi_1 \circ u = \vec{x}$ and $\pi_2 \circ u = \vec{y}$. So in the light of our well-motivated Defn. 53, we can naturally say that $[O, \pi_1, \pi_2]$ form a product of $X$ with $Y$.

So far so good. But this only works as we want in well-pointed categories with ‘enough’ elements. However, we know how to generalize to other categories – i.e. replace talk about elements (points) with talk of generalized elements. Which motivates the following definition:

**Definition 54.** In any category $\mathcal{C}$, a (binary) product $[O, \pi_1, \pi_2]$ for the objects $X$ with $Y$ is an object $O$ together with ‘projection’ arrows $\pi_1 : O \to X, \pi_2 : O \to Y$, such that for any object $S$ and arrows $f_1 : S \to X$ and $f_2 : S \to Y$ there is always a unique arrow $u : S \to O$ such that the following diagram commutes:

\[
\begin{array}{ccc}
S & \xleftarrow{u} & O \\
\downarrow{f_1} & & \downarrow{\pi_1} \\
X & \xleftarrow{\pi_2} & Y \\
\end{array}
\]

The unique mediating arrow $u$ in that diagram is also very often labelled ‘$(f_1, f_2)$’ or ‘$(f_1, f_2)$’ – but be careful not to over-interpret that notation. And note, by the way, that we are falling into the following common

**Convention.** In a category diagram, we use a dashed arrow $\longrightarrow$ to indicate an arrow which is uniquely fixed by the requirement that the diagram commutes.

Here’s very slightly different way of putting things (sometimes just geometrically distorting/rotating/reflecting a diagram can be illuminating!). Let’s say

**Definition 55.** A wedge to $X$ and $Y$ (in category $\mathcal{C}$) is an object $S \in \mathcal{C}$ and a pair of arrows $f_1 : S \to X, f_2 : S \to Y$. 
Then a wedge \( \begin{array}{c}
O \\
\pi_1
\end{array} \rightarrow X \\
\pi_2 \rightarrow Y \)

is a product for \( X \) with \( Y \) iff for any other wedge \( \begin{array}{c}
S \\
f_1
\end{array} \rightarrow X \\
\pi_2 \rightarrow Y \)

\( X \) and \( Y \), there exists a unique morphism \( u \) such that the following diagram commutes:

\[
\begin{array}{ccc}
S & \xrightarrow{u} & O \\
\pi_1 & \downarrow & \pi_2 \\
X & \xrightarrow{f_1} & X \\
\pi_2 & \downarrow & \pi_2 \\
Y & \xrightarrow{f_2} & Y \\
\end{array}
\]

We can say \( f_1 \) ‘factors’ as \( \pi_1 \circ u \) and \( f_2 \) as \( \pi_2 \circ u \), and so the whole wedge from \( S \) into \( X \) and \( Y \) (uniquely) factors through the product via \( u \). And another way of putting the definition of a product, now using the notion of wedges, is captured like this:

\textbf{Definition 56.} Given a category \( \mathcal{C} \) and objects \( X, Y \in \mathcal{C} \), then the derived wedge category \( \mathcal{C}_{W(XY)} \) has as objects all wedges \( [O, f_1, f_2] \) to \( X, Y \). And an arrow from \( [O, f_1, f_2] \) to \( [O', f'_1, f'_2] \) is a \( \mathcal{C} \)-arrow \( g: O \to O' \) such that the resulting triangles commute: i.e. \( f_j = f'_j \circ g \). A product of \( X, Y \) is then a terminal object of \( \mathcal{C}_{W(XY)} \).

Let’s have some examples:

1. In \textbf{Set}, as you would certainly hope, the usual Cartesian product, the set \( X \times Y \) of Kuratowski pairs \( (x, y) \) of elements from \( X \) and \( Y \), together with the obvious projection functions \( (x, y) \overset{\pi_1}{\mapsto} x \) and \( (x, y) \overset{\pi_2}{\mapsto} y \) form a binary product.

   Let’s just confirm this. Suppose we are given any set \( S \) and functions \( f_1: S \to X \) and \( f_2: S \to Y \). Then if, for \( s \in S \), we put \( u(s) = (f_1(s), f_2(s)) \), the diagram evidently commutes. Now trivially, for any pair \( p \in X \times Y \), \( p = (\pi_1 p, \pi_2 p) \).

   Hence if \( u': S \to X \times Y \) is another candidate for completing the diagram, \( u(s) = (f_1(s), f_2(s)) = (\pi_1 u'(s), \pi_2 u'(s)) = u'(s) \). So \( u \) is unique.

That paradigm case motivates the following

\textbf{Convention.} We will often use the notation \( X \times Y \) for the object \( O \) in a binary product \([O, \pi_1, \pi_2] \) for \( X \) with \( Y \).

Continuing our examples:

2. In \textbf{Grp}, you can construct a product of the groups \((G, \cdot)\) and \((H, \circ)\) by taking the group \((G \times H, \circ)\) where \( G \times H \) is the usual Cartesian product of the underlying sets and the group operation is defined component-wise, so that \((g, h) \circ (g', h') = (g \cdot g', h \circ h')\), and then equipping this group with the obvious projection functions from \( G \times H \) to \( G \) (resp. \( H \)) which send \((g, h)\) to \( g \) (resp. \( h \)).

3. Take a poset \((P, \preceq)\) considered as a category (so there is an arrow \( p \to q \) iff \( p \preceq q \)). Then a product of \( p \) and \( q \) would be an object \( c \) such that \( c \preceq p, c \preceq q \) and such that for any object \( d \) with arrows from it to \( p \) and \( q \), i.e. any \( d \) such that \( d \preceq p, d \preceq q \), there is a unique arrow from \( d \) to \( c \), i.e. \( d \preceq c \). That means the product of \( p \) and \( q \) must be their supremum (equipped with the obvious two arrows).

   Since pairs of objects in posets need not in general have greater lower bounds, that goes to show that a category in general need not have products.
(4) Here’s a new example of a category, call it \( \text{Prop}_L \) – its objects are propositions, wffs of a given first-order language \( L \), and there is a unique arrow from \( X \) to \( Y \) iff \( X \vdash Y \), i.e. iff \( X \) semantically entails \( Y \). The reflexivity and transitivity of semantic entailment means we get the identity and composition laws which ensure that this is a category.

In this case, one product of \( X \) with \( Y \) will be the conjunction \( X \land Y \) (with the obvious projections \( X \land Y \to X \), \( X \land Y \to Y \)).

12.5 Results about binary products

(a) We say:

**Definition 57.** A category \( \mathcal{C} \) has binary products if for all objects \( X, Y \in \mathcal{C} \), there exists a product of \( X \) with \( Y \).

But of course, products need not exist for arbitrary objects \( X \) and \( Y \) in a given category; and when they do, they need not be strictly unique. However, when they exist, they are ‘unique up to unique isomorphism’. That is to say,

**Theorem 65.** If both \( [O, \pi_1, \pi_2] \) and \( [O', \pi'_1, \pi'_2] \) are products for \( X \) with \( Y \) in the category \( \mathcal{C} \), then there is a unique isomorphism \( f : O \cong O' \) (which also fixes what \( \pi'_1 \), \( \pi'_2 \) have to be, given \( \pi_1 \), \( \pi_2 \)).

**Proof.** Start with a simple observation. Since \( [O, \pi_1, \pi_2] \) is a product, every wedge factors uniquely through it, including itself. In other words, there is a unique \( u \) such that this diagram commutes:

\[
\begin{array}{ccc}
O & \xrightarrow{\pi_1} & X \\
\downarrow{u} & & \downarrow{\pi_1} \\
O & \xrightarrow{\pi_2} & Y
\end{array}
\]

But evidently putting \( 1_O \) for the central arrow trivially makes the diagram commute. So by the uniqueness requirement we know that

(i) Given an arrow \( u : O \to O \), if \( \pi_1 \circ u = \pi_1 \) and \( \pi_2 \circ u = \pi_2 \), then \( u = 1_O \).

Now, since \( [O', \pi'_1, \pi'_2] \) is a product, \( [O, \pi_1, \pi_2] \) has to uniquely factor through it. That is to say, there is a unique \( f : O \to O' \) such that

(ii) \( \pi'_1 \circ f = \pi_1 \) and \( \pi'_2 \circ f = \pi_2 \).

And since \( [O, \pi_1, \pi_2] \) is also a product, the other wedge has to uniquely factor through it. That is to say, there is a unique \( g : O' \to O \) such that

(iii) \( \pi_1 \circ g = \pi'_1 \) and \( \pi_2 \circ g = \pi'_2 \).

Whence,

(iv) \( \pi_1 \circ g \circ f = \pi'_1 \circ f = \pi_1 \) and \( \pi_2 \circ g \circ f = \pi_2 \).

But \( g \circ f : O \to O \), from which it follows – given our initial observation (i) – that

(v) \( g \circ f = 1_O \)

The situation with the wedges is symmetric so we also have

(vi) \( f \circ g = 1_{O'} \)

Hence \( f \) is an isomorphism. It also follows that \( \pi'_1 = \pi_1 \circ f^{-1} \), and \( \pi'_2 = \pi_2 \circ f^{-1} \). \( \square \)
(b) We next verify some more respects in which binary products behave just as you would expect. First, equal components of a product mean equal products. Or categorically, if arrows into a product factor through the product the same way, they are equal. In other words,

**Theorem 66.** Given a product \([X \times Y, \pi_1, \pi_2]\) and parallel arrows \(S \xrightarrow{f} X \times Y, \xrightarrow{g} X \times Y\), then, if \(\pi_1 \circ f = \pi_1 \circ g\) and \(\pi_2 \circ f = \pi_2 \circ g\), it follows that \(f = g\).

**Proof.** A diagram says it all!

\[
\begin{array}{c}
\pi_1 f / \pi_1 g & \pi_2 f / \pi_2 g \\
\downarrow f & \downarrow g \\
X & X \times Y & Y \\
\pi_1 & \pi_2 \\
& X \\
\end{array}
\]

The wedge \(X \leftarrow S \rightarrow Y\) factors uniquely thought \(X \times Y\), and hence \(f = g\). \(\square\)

**Theorem 67.** In a category with a terminal object 1 and with the relevant products,

1. \(1 \times X \cong X \cong X \times 1\)
2. \(X \times Y \cong Y \times X\)
3. \(X \times (Y \times Z) \cong (X \times Y) \times Z\)

**Proof** (1) Evidently the wedge \(\xrightarrow{1_X} X\) exists for some unique map \(f\) since 1 is terminal. Hence if \(X \times 1\) is a product, there is a unique \(u\) such that this commutes,

\[
\begin{array}{c}
\ X \\
\pi_1 \\
\pi_2 \\
\downarrow f \\
1 \\
\end{array}
\]

Hence \(\pi_1 \circ u = 1_X\). But \(1_X\) is an inverse of itself and we can go counterclockwise round the top triangle to get the arrow \(u \circ 1_X \circ \pi_1\) mapping \(X \times 1\) to itself, which must equal the undrawn identity arrow on \(X \times 1\); so \(u \times \pi_1 = 1_{X \times 1}\). Hence \(u\) has a two-sided inverse, i.e. is an isomorphism, so \(X \cong X \times 1\). Similarly for the other half of (1).

(2) If \([X \times Y, \pi_1 : X \times Y \rightarrow X, \pi_2 : X \times Y \rightarrow Y]\) is a product of \(X\) with \(Y\), then \([X \times Y, \pi_2, \pi_1]\) is obviously a product of \(Y\) with \(X\), and hence – by Theorem 65 – there is a unique isomorphism between the object in that product and the object \(Y \times X\) of any other product of \(Y\) with \(X\).

(3) For the last part, it is (perhaps!) a just-about useful exercise to show that it can be proved by appeal to our definition of a product, using brute force: in the next section we give a slicker proof.

We are given the following wedges are products:
CHAPTER 12. FIRST EXAMPLES OF LIMITS

A \leftarrow^f A \times (B \times C) \rightarrow^g B \times C

A \times B \leftarrow^j (A \times B) \times C \rightarrow^k C

A \leftarrow^m A \times B \rightarrow^n B

B \leftarrow^r B \times C \rightarrow^s C

Which means that there are wedges

A \leftarrow^f A \times (B \times C) \rightarrow^{rog} B

B \leftarrow^{noj} (A \times B) \times C \rightarrow^k C

The first, a wedge to A and B, must factor through the product A \times B; and the second must factor through B \times C, via some u and v respectively. Which gives us:

A \times (B \times C) \rightarrow^u A \times B, \quad \text{where } f = m \circ u, \text{ and } r \circ g = n \circ u

(A \times B) \times C \rightarrow^v B \times C, \quad \text{where } k = s \circ v, \text{ and } n \circ j = r \circ v

So we now also have wedges

A \times B \leftarrow^u A \times (B \times C) \rightarrow^{sog} C

A \leftarrow^{moj} (A \times B) \times C \rightarrow^v B \times C

These wedges must in turn factor through, respectively, the products (A \times B) \times C and A \times (B \times C), via some x and y, which gives us:

A \times (B \times C) \rightarrow^x (A \times B) \times C, \quad \text{where } u = j \circ x, \text{ and } s \circ g = k \circ x

(A \times B) \times C \rightarrow^y (A \times B) \times C, \quad \text{where } v = g \circ y, \text{ and } m \circ j = f \circ y

If we can show x and y are mutually inverse, we are done. So here goes!

Note we have another wedge

A \leftarrow^{moj} (A \times B) \times C \rightarrow^{noj} B

which, being a wedge to A and B, must uniquely factor through

A \leftarrow^m A \times B \rightarrow^n B

That is to say, there is a unique z such that m \circ z = m \circ j and n \circ z = n \circ j, which is of course z = j.

But now put w = j \circ x \circ y. Then we have, appealing to various of the equations above,

m \circ w = m \circ j \circ x \circ y = m \circ u \circ y = f \circ y = m \circ j

n \circ w = n \circ j \circ x \circ y = n \circ u \circ y = r \circ g \circ y = r \circ v = n \circ j

Hence w is also a map such that m \circ w = m \circ j and n \circ w = n \circ j, so by the uniqueness of candidates

j \circ x \circ y = w = j

We also have

k \circ x \circ y = s \circ g \circ y = s \circ v = k

So finally consider again the wedge
This wedge must of course factor uniquely through itself. So there is a unique map \( q \) from \((A \times B) \times C\) to itself such that \( j \circ q = k \) and \( k \circ q = k \). But trivially \( 1_{(A \times B) \times C} \) satisfies that last condition. But we have also just shown that \( x \circ y \) satisfies the condition. Hence

\[
x \circ y = 1_{(A \times B) \times C}
\]

An exactly similar argument shows that we also have

\[
y \circ x = 1_{A \times (B \times C)}
\]

And so, at last, we are indeed done! □

(c) We remark in passing on a special case:

**Definition 58.** In a category with binary products, the wedge \( X \xleftarrow{1_X} X \xrightarrow{1_X} X \) must factor uniquely through the product \( X \times X \) via an arrow \( u: X \to X \times X \). That unique arrow \( u \) is the diagonal morphism on \( X \), and so we will notate it \( \delta_X \).

In \( \text{Set} \), \( \delta_X \) is the function that sends an element \( x \in X \) to \( \langle x, x \rangle \) (hence the label ‘diagonal’).

(d) Lastly in this section, we prove a theorem which gives us another characterization of products (at least for locally small categories). First we need a definition.

**Definition 59.** Suppose \( \mathcal{C} \) is locally small, and \( O, X, Y \in \mathcal{C} \), with arrows \( x: O \to X \) and \( y: O \to Y \). Then \( \mathcal{C}(\cdot, X) \times \mathcal{C}(\cdot, Y): \mathcal{C}^{\text{op}} \to \text{Set} \) is the functor which sends \( O \) to the set \( \mathcal{C}(O, X) \times \mathcal{C}(O, Y) \) (invoking the usual cartesian product of sets), and sends an arrow \( f: O' \to O \) to the function \( \langle x, y \rangle \mapsto \langle x \circ f, y \circ f \rangle \).

It is readily checked that this does indeed well-define a functor. Then

**Theorem 68.** \([O, \pi_1, \pi_2]\) is a product of \( X \) with \( Y \) in the locally small \( \mathcal{C} \) iff \( \langle O, \langle \pi_1, \pi_2 \rangle \rangle \) is a universal element of the functor \( F = \mathcal{C}(\cdot, X) \times \mathcal{C}(\cdot, Y) \).

*Proof.* This is just a straightforward application of Defn. 44. \( \langle O, \langle \pi_1, \pi_2 \rangle \rangle \) is a universal element of that functor so long as, for each \( S \in \mathcal{C}^{\text{op}} \) and each object in \( FS \), i.e. each \( \langle f_1, f_2 \rangle \) such that \( f_1 \in \mathcal{C}(S, X), f_2 \in \mathcal{C}(S, Y) \), there is a unique map \( u: Z \to O \) (remember the functor is from \( \mathcal{C}^{\text{op}} \)) such that the map \( \langle x, y \rangle \mapsto \langle x \circ u, y \circ u \rangle \) sends \( \langle \pi_1, \pi_2 \rangle \) to \( \langle f_1, f_2 \rangle \), i.e. so long as \( \pi_1 \circ u = f_1, \pi_2 \circ u = f_2 \). Which is exactly the condition for \([O, \pi_1, \pi_2]\) being a product of \( X \) with \( Y \). □

(If we really wanted to, we could apply this to bigger categories by using the device of §10.4 and considering functors from \( \mathcal{C}^{\text{op}} \) to a big category of collections \( \text{set} \). But again, we won’t follow up this idea.)

### 12.6 Two-place functions and arrows from products

(a) Let’s step back from category theory just for a moment, and think about a familiar line on multi-place functions.

A one-place total function from numbers to numbers, like the successor function or the square-of function, is a function \( f: \mathbb{N} \to \mathbb{N} \). What about a two-place total function from numbers to numbers, like addition or multiplication? “Easy-peasy: that is a function of the type \( f: \mathbb{N}^2 \to \mathbb{N} \).”
CHAPTER 12. FIRST EXAMPLES OF LIMITS

But hold on! \( \mathbb{N}^2 \), i.e. \( \mathbb{N} \times \mathbb{N} \), is the cartesian product of \( \mathbb{N} \) with itself, i.e. is the set of ordered pairs of numbers: and an ordered pair is one thing not two things. So a function from \( \mathbb{N}^2 \) to \( \mathbb{N} \) – according to the set-theoretic orthodoxy – is in fact a unary function that maps one argument, an ordered pair object, to a value, and is not (as we initially wanted) a binary function mapping two arguments to a value.

“Ah, don’t be so pernickety! Given two objects, we can find a pair-object that codes for them – we usually choose a Kuratowski pair – and we can without loss trade in a function from two objects to a value to a related function from the corresponding pair-object to the same value.”

Yes, sure, we can do that. And standard notational choices can make the trade invisible. For suppose we adopt the modern convention of using ‘\((m, n)\)’ as our notation for the ordered pair of \( m \) with \( n \), then ‘\( f(m, n) \)’ can be parsed either way, as representing a two-place function \( f(\cdot, \cdot) \) with arguments \( m \) and \( n \) or as a corresponding one-place function \( f \cdot \) with the single argument \( (m, n) \). But the fact that trade between the two-place and the one-place function is notationally glossed over doesn’t mean that it isn’t being made. And the fact that the trade can be made (even staying within arithmetic, using an arithmetic pairing function) is a result and not quite a triviality. So if we are doing things from scratch – including proving that there is a pairing scheme that matches two things with one thing in such a way that we can then extract the two objects we started with – then we do need to at least start by talking about two-place functions proper. For example, if we want to confine ourselves to pure arithmetic, we show how to construct a pairing function from the ordinary school-room two-place addition and multiplication functions, not from some surrogate one-place functions!

Versions of type theory also trade in two-place functions for unary functions, treating addition for example as a function of the type \( \mathbb{N} \to (\mathbb{N} \to \mathbb{N}) \); this is a unary function which takes one number (of type \( \mathbb{N} \)) to output another unary function (of type \( \mathbb{N} \to \mathbb{N} \)). So this time we get from two numbers as input to a numerical output by feeding the first number to a function which delivers another function as output and then feeding the second number to the second function. This so-called ‘currying’ trick of course works to eliminate two-place functions: that is to say, it is adequate for certain particular formal purposes. But is the type-shifting supposed to reveal what two-place functions are ‘really’?

Heres a revealing quote from A Gentle Introduction to Haskell on the haskell.org site (Haskell being one those programming languages where what we might think of naturally as binary functions are curried):

Consider this definition of a function which adds its two arguments:

\[
\text{add :: Integer } \to \text{ Integer } \to \text{ Integer}
\]

\[
\text{add } x \ y = x + y
\]

So we have the declaration of type – we are told that \text{add} sends a number to a function from numbers to numbers – and we are then told how the curried function acts ... but how? By appeal, of course, to our prior understanding of the familiar school-room two-place function! Ordinary two-place functions are essential rungs on the ladder by which we can climb to an understanding of what’s going on in the likes of Haskell.

(b) How does category theory stand on two-place functions (and multi-place functions more generally)? As we noted before, at ground level, all arrows are from an object (one of them) to an object. So we don’t get ‘native’ binary functions. Nor do we get currying within a category, at least in the sense that we don’t have arrows inside a category from objects to arrows of that category – arrows are always from objects to
the set-theoretic one. We can have an arrow of the kind objects (we’ll meet a different sort of currying in due course). The obvious trick which allows us to model or represent two-place functions categorically is, rather, a version of the set-theoretic one. We can have an arrow of the kind \( f: X \times Y \to S \) where, equipping the object with suitable projection arrows, \([X \times Y, \pi_1, \pi_2]\) is a product of \( X \) with \( Y \), and such an \( f \) will do duty for a two-place function from an object in \( X \) and an object in \( Y \) to a value in \( S \).

**12.7 Products generalized**

(a) So far we have talked of binary products. But we can generalize in obvious ways. For example,

**Definition 60.** In any category \( \mathcal{C} \), a ternary product \([O, \pi_1, \pi_2, \pi_3]\) for the objects \( X_1, X_2, X_3 \) is an object \( O \) together with projection arrows \( \pi_i: O \to X_i \) (i = 1, 2, 3) such that for any object \( S \) and arrows \( f_i: S \to X_i \) there is always a unique arrow arrow \( u: S \to O \) such that \( f_i = \pi_i \circ u \).

And then, exactly as we would expect, using the same proof ideas as in the binary case, we can prove

**Theorem 69.** If both the ternary products \([O, \pi_1, \pi_2, \pi_3]\) and \([O', \pi_1', \pi_2', \pi_3']\) exist for \( X_1, X_2, X_3 \) in the category \( \mathcal{C} \), then there is a unique isomorphism \( f: O \cong O' \) (which also fixes what the \( \pi_i \) have to be given the \( \pi_i' \)).

We now note that if \( \mathcal{C} \) has binary products for all pairs of objects, then it has ternary products too, for

**Theorem 70.** \((X_1 \times X_2) \times X_3\) together with the obvious projection arrows forms a ternary product of \( X_1, X_2, X_3 \).

**Proof.** Assume \([X_1 \times X_2, \pi_1, \pi_2]\) is a product of \( X_1 \) with \( X_2 \), and \([(X_1 \times X_2) \times X_3, \rho_1, \rho_2]\) is a product of \( X_1 \times X_2 \) with \( X_3 \).

Take any object \( S \) and arrows \( f_i: S \to X_i \). By our first assumption, \( a \) there is a unique \( u: S \to X_1 \times X_2 \) such that \( f_1 = \pi_1 \circ u \), \( f_2 = \pi_2 \circ u \). So by our second assumption (b) there is then a unique \( v: S \to (X_1 \times X_2) \times X_3 \) such that \( u = \rho_1 \circ v \), \( f_3 = \rho_2 \circ v \).

Therefore \( f_1 = \pi_1 \circ \rho_1 \circ v \), \( f_2 = \pi_2 \circ \rho_1 \circ v \), \( f_3 = \rho_2 \circ v \).

So now consider \([(X_1 \times X_2) \times X_3, \pi_1 \circ \rho_1, \pi_2 \circ \rho_1, \rho_2]\). This, we claim, is indeed a ternary product of \( X_1, X_2, X_3 \). We’ve just proved that \( S \) and arrows \( f_i: S \to X_i \) factor through the product via the arrow \( v \). It remains to confirm \( v \)’s uniqueness in this new role.

Suppose we have \( w: S \to (X_1 \times X_2) \times X_3 \) where \( f_1 = \pi_1 \circ \rho_1 \circ w \), \( f_2 = \pi_2 \circ \rho_1 \circ w \), \( f_3 = \rho_2 \circ w \). Then \( \rho_1 \circ w: S \to X_1 \times X_2 \) is such that \( f_1 = \pi_1 \circ (\rho_1 \circ w) \), \( f_2 = \pi_2 \circ (\rho_1 \circ w) \). Hence by (a), \( u = \rho_1 \circ w \). But now invoking (b), that together with \( f_3 = \rho_2 \circ w \) entails \( w = v \).

Evidently, an exactly similar argument will show that \( X_1 \times (X_2 \times X_3) \) together with the obvious projection arrows forms a ternary product of \( X_1, X_2, X_3 \). And that, combined with Theorem 69, tells us that \( X_1 \times (X_2 \times X_3) \cong (X_1 \times X_2) \times X_3 \), so giving us a very much neater proof of the last part of Theorem 67.
CHAPTER 12. FIRST EXAMPLES OF LIMITS

(b) What goes for ternary products goes for \( n \)-ary products defined in a way exactly analogous to Defn. 60. And if \( C \) has binary products for all pairs of objects it will have quaternary products such as \( (X_1 \times X_2) \times X_3 \times X_4 \), quinary products, and \( n \)-ary products more generally, for any finite \( n \geq 2 \).

To round things out, how do things go for the nullary and unary cases?

Following the same pattern of definition, a nullary product in \( C \) would be an object \( O \) together with no projection arrows, such that for any object \( S \) there is a unique arrow \( u: S \to O \). Which is just to say that a nullary product is a terminal object of the category.

And a unary product of \( X \) would be an object \( O \) and a single projection arrow \( \pi_1: O \to X \) such that for any object \( S \) and arrow \( f: S \to X \) there is a unique arrow \( u: S \to O \) such that \( \pi \circ u = f \). Putting \( O = X \) and \( \pi = 1_X \) evidently fits the bill. So the basic case of a unary product of \( X \) is not quite \( X \) itself, but rather \( X \) equipped with its identity arrow (and like any product, this is unique up to unique isomorphism).

Trivially, all unary products exist in all categories.

In sum, if we say

**Definition 61.** A category \( C \) has all finite products iff \( C \) has \( n \)-ary products for any \( n \) objects for all \( n \geq 0 \),

then our preceding remarks establish

**Theorem 71.** A category \( C \) has all finite products iff \( C \) has a terminal object and a binary product for any pair of objects.

(c) We can generalize further in the obvious way, beyond finite products to infinite cases.

**Definition 62.** In any category \( C \), the product of the \( C \)-objects \( X_i \) (for indices \( i \in I \)), if it exists, is an object \( O \) together with projection arrows \( \pi_i: O \to X_i \) (for \( i \in I \)) such that for any object \( S \) and arrows \( f_i: S \to X_i \) (again for \( i \in I \)), there is always a unique arrow arrow \( u: S \to O \) such that \( f_i = \pi_i \circ u \). (We can use \( \prod_{i \in I} \) to notate such an \( O \).)

As before, these generalized products will be unique up to unique isomorphism. And we will say

**Definition 63.** A category \( C \) has all small products iff for any set’s worth of \( C \)-objects – i.e. any objects \( C \)-objects \( X_i \), for \( i \in I \), where \( I \) is some set – the objects in the set have a product.

Obviously, a category which has all small products will a fortiori have all finite products; and equally obviously the reverse does not hold (think, for example, of the category \( \text{Finset} \) of finite sets).

### 12.8 Coproducts

Let’s note a common terminological device:

**Convention.** For many kinds of categorially defined widget, a co-widget of \( C \) is a widget of \( C^{\text{op}} \): co-widgets are dual to widgets.

For example, we could (and some do) call initial objects ‘co-terminal’. Likewise we could (and some do) call sections ‘co-retractions’. True, there is a limit to this sort of thing –
no one, as far as I know, talks e.g. of ‘co-monomorphisms’ (instead of ‘epimorphisms’). But the convention is used quite widely. In particular, it is absolutely standard to talk of ‘co-products’, or indeed more commonly ‘coproducts’.

The definition of a coproduct is immediately obtained, then, by reversing all the arrows in our definition of products. Thus:

**Definition 64.** In any category \( \mathcal{C} \), a (binary) coproduct \([O, \iota_1, \iota_2]\) for the objects \( X \) with \( Y \) is an object \( O \) together with ‘injection’ arrows \( \iota_1 : X \rightarrow O, \iota_2 : Y \rightarrow O \), such that for any object \( S \) and arrows \( f_1 : X \rightarrow S \) and \( f_2 : Y \rightarrow S \) there is always a unique arrow \( v : O \rightarrow S \) such that the following diagram commutes:

\[
\begin{array}{ccc}
X & \xrightarrow{f_1} & S \\
\downarrow{\iota_1} & & \downarrow{v} \\
O & \xleftarrow{v} & Y \\
\downarrow{\iota_2} & & \downarrow{f_2}
\end{array}
\]

Note, ‘injections’ in this sense need not be injective or even monic. The object \( O \) in a coproduct for \( X \) with \( Y \) is often notated ‘\( X \oplus Y \)’ or ‘\( X \amalg Y \)’.

Let’s say that objects and arrows arranged as \( X \xrightarrow{\iota_1} O \leftarrow \xleftarrow{\iota_2} Y \) form a **corner** (or we could say ‘co-wedge’) from \( X \) and \( Y \) with vertex \( O \). Then a co-product of \( X \) with \( Y \) is a corner from \( X \) and \( Y \) which factors through any other corner from \( X \) and \( Y \) via a unique map between the vertices of the corners.

Before we ever meet any coproducts, we immediately know that we have the following composite theorem, just by duality:

**Theorem 72.** If both the coproducts \([O, \iota_1, \iota_2]\) and \([O', \iota'_1, \iota'_2]\) exist for \( X \) and \( Y \) in the category \( \mathcal{C} \), then there is a unique isomorphism \( f : O \cong O' \) (which also fixes what \( \iota'_1, \iota'_2 \) have to be, given \( \iota_1, \iota_2 \)).

And in a category with an initial object \( 0 \) and where the coproducts exist,

1. \( 0 \oplus X \cong X \cong X \oplus 0 \)
2. \( X \oplus Y \cong Y \oplus X \)
3. \( X \oplus (Y \oplus Z) \cong (X \oplus Y) \oplus Z \)

But now let’s now have some examples of coproducts. Start with easy cases:

1. For **Set**, the headline news is that disjoint unions are coproducts.

   Given sets \( X \) and \( Y \), take the set \( X \oplus Y \) with members \( \langle x, 0 \rangle \) for \( x \in X \) and \( \langle y, 1 \rangle \) for \( y \in Y \). And let \( \iota_1 : X \rightarrow X \oplus Y \) be the function \( x \mapsto \langle x, 0 \rangle \), and similarly let \( \iota_2 : Y \rightarrow X \oplus Y \) be the function \( y \mapsto \langle y, 1 \rangle \). Then \([X \oplus Y, \iota_1, \iota_2]\) is a coproduct for \( X \) with \( Y \).

   For take any object \( S \) and arrows \( f_1 : X \rightarrow S \) and \( f_2 : Y \rightarrow S \), then define the function \( v : X \oplus Y \rightarrow S \) as sending an element \( \langle x, 0 \rangle \) to \( f_1(x) \) and an element \( \langle y, 1 \rangle \) to \( f_2(y) \). By construction, this will make the diagram commute. Moreover, if \( v' \) is another candidate for completing the diagram, \( v'(\langle x, 0 \rangle) = v' \circ \iota_1(x) = f_1(x) = v(\langle x, 0 \rangle) \), and likewise \( v'(\langle y, 1 \rangle) = v(\langle y, 1 \rangle) \), whence \( v' = v \), which gives us the desired uniqueness.

2. Take a poset \((P, \preceq)\) considered as a category (so there is an arrow \( p \rightarrow q \) iff \( p \preceq q \)).

   Then a coproduct of \( p \) and \( q \) would be an object \( c \) such that \( p \preceq c, q \preceq c \) and such that for any object \( d \) such that \( p \preceq d, q \preceq d \) there is a unique arrow from \( c \) to \( d \), i.e. \( c \preceq d \). Which means that the coproduct of \( p \) and \( q \), if it exists, must be their least upper bound (equipped with the obvious two arrows).
(3) In Prop, the disjunction $X \lor Y$ (with the obvious injections $X \to X \lor Y$, $Y \to X \lor Y$) is a coproduct of $X$ with $Y$.

However things soon get rather less obvious (the details here aren’t going to matter at least for now, so by all means skip, but we’ll spell out a couple of cases):

(4) In the category Grp, coproducts are the so-called ‘free products’ of groups.

Take the groups $G = (G, \cdot)$, $H = (H, \odot)$ [again, we abuse notation in a familiar way]. If necessary, doctor these to equate their identity elements while ensuring the sets $G$ and $H$ are otherwise disjoint. Form all the finite ‘reduced words’ $G \ast H$ you get by concatenating elements from $G \cup H$, and then multiplying out neighbouring $G$-elements by $\cdot$ and neighbouring $H$-elements by $\odot$ as far as you can. Equip $G \ast H$ with the operation $\circ$ of concatenation-of-words-followed-by-reduction. Then $G \ast H = (G \ast H, \odot)$ is a group – the free product of the two groups we started with – and there are obvious ‘injection’ group homomorphisms $\iota_1 : G \to G \ast H$, $\iota_2 : H \to G \ast H$. Claim: $[G \ast H, \iota_1, \iota_2]$ is a coproduct for the groups $G$ and $H$. That is to say, for any group $K = (K, *)$ and morphisms $f_1 : G \to K$, $f_2 : H \to K$, there is a unique $v$ such that this commutes:

$$
\begin{array}{ccc}
G & \xrightarrow{\iota_1} & G \ast H & \xleftarrow{\iota_2} & H \\
\uparrow & & \uparrow v & & \downarrow \\
\downarrow f_1 & & & & \downarrow f_2 \\
& & K & & \\
\end{array}
$$

Put $v : G \ast H \to K$ to be the morphism that sends a word like $g_1 h_1 g_2 h_2 \cdots g_r$ ($g_i \in G, h_i \in H$) to $j(g_1) \ast k(h_1) \ast j(g_2) \ast k(h_2) \ast \cdots \ast j(g_r)$. By construction, $v \circ \iota_1 = j$, $v \circ \iota_2 = k$. So that makes the diagram commute.

Let $v'$ be any other candidate group homomorphism to make the diagram commute. Then, to take a simple example, consider $gh \in G \ast H$. Then $v'(gh) = v'(g) \ast v'(h) = v'(g) \ast v'(i_1(g)) \ast v'(i_2(h)) = f_1(g) \ast f_2(h) = v(i_1(g)) \ast v(i_2(h)) = v(i_1(g) \ast i_1(h)) = v(g h)$. And by induction over the length of words we’ll get $v' = v$. So, as required, $v$ is unique.

(5) So what about coproducts in Ab, the category of abelian groups? Since the free product of two abelian groups need not be abelian, the same construction won’t work again as it stands.

OK: hit the construction with the extra requirement that words in $G \ast H$ be treated as the same if one can be shuffled into the other (in effect, further reduce $G \ast H$ by quotienting out with the obvious equivalence relation). But that means that we can take a word other than the identity, bring all the $G$-elements to the front, followed by all the $H$ elements: but now multiply out the $G$-elements and the $H$-elements and we are left with two-element word $gh$. So we can equivalently treat the members of our further reduced $G \ast H$ as pairs $(g, h)$ belonging to $G \times H$. Equip this with the group operation $\circ$ defined component-wise as before (in §12.4): this gives us an abelian group if $G$ and $H$ are. Take the obvious injections, $g \xrightarrow{i_1} (g, 1)$ and $h \xrightarrow{i_2} (1, h)$. Then we claim $[G \times H, i_1, i_2]$ is a coproduct for the abelian groups $G$ and $H$.

Take any abelian group $K = (K, *)$ and morphisms $f_1 : G \to K$, $f_2 : H \to K$. Put $v : G \times H \to K$ to be the morphism that sends $(g, h)$ to $f_1(g) * f_2(h)$. This evidently makes the coproduct diagram (with $G \times H$ for $G \ast H$) commute. And a similar argument to before shows that it is unique.
So, in the case of abelian groups, the same objects can serve as both products and coproducts, when equipped with (respectively) appropriate projections and injections.

Finally in this section, let’s note that the notion of a coproduct generalizes beyond the binary case, just as with products. Thus, exactly as you would expect, we have:

**Definition 65.** In any category $\mathcal{C}$, the coproduct of the $\mathcal{C}$-objects $X_i$ (for indices $i \in I$) is an object $O$ together with injection arrows $\iota_i: X_i \to O$ (for $i \in I$) such that for any object $S$ and suite of arrows $f_i: X_i \to O$, there is always a unique arrow arrow $v: O \to S$ such that $f_i = v \circ \pi_i$. (We can use $\bigsqcup_{i \in I}$ to notate such an $O$.)

By dual arguments to those we’ve met for products, nullary coproducts are initial objects and unary coproducts are objects equipped with its identity arrow. And we can define what it is for a category to have all finite/small coproducts in the obvious ways.

### 12.9 Equalizers

We’ve so far met terminal and initial objects, products and coproducts. Now for our third pair of examples in this long chapter, equalizers and co-equalizers. This third pair of cases is perhaps not so intrinsically interesting; but together with products will play an important role in our general theory of ‘limits’.

It was useful, when defining products, to introduce the idea of a ‘wedge’ (Defn. 55) for a certain small configuration of objects and arrows in a category. Here’s a similar definition that is going to be useful in defining the equalisers:

**Definition 66.** A fork (from $S$ through $X$ to $Y$) consists of arrows $k: S \to X$ with $f: X \to Y$ and $g: X \to Y$, such that $f \circ k = g \circ k$.

So diagrammatically, a fork looks like this: $S \xrightarrow{k} X \xrightarrow{f} Y$, with the composite arrows from $S$ to $Y$ being equal.

Now, a product wedge from $O$ to $X$ and $Y$ is a ‘limit’ wedge such that any other wedge from $S$ to $X$ and $Y$ uniquely factors through it. Likewise, an equalizing fork from $E$ through $X$ to $Y$ is a ‘limit’ fork such that any other fork from an object $S$ through $X$ to $Y$ uniquely factors through it. That is to say

**Definition 67.** Let $\mathcal{C}$ be a category and $f: X \to Y$ and $g: X \to Y$ be a pair of ‘parallel’ arrows in $\mathcal{C}$. Then the object $E$ and arrow $e: E \to X$ form an equalizer in $\mathcal{C}$ for those arrows iff $f \circ e = g \circ e$, and for any fork $S \xrightarrow{k} X \xrightarrow{f} Y$ there is a unique arrow $u: S \to E$ such the following diagram commutes:

\[
\begin{array}{ccc}
S & \xrightarrow{k} & X \\
\downarrow{u} & & \downarrow{f} \\
E & \xrightarrow{e} & Y
\end{array}
\]

Note that, just as with products (see Defn. 56), we can give an alternative definition which defines equalizers in terms of a terminal object in a category of forks:
**Definition 68.** Given a category \( \mathcal{C} \) and parallel arrows \( f, g : X \to Y \), then the derived fork category \( \mathcal{C}_{F(\mathcal{C})} \) has as objects all forks \( S \xrightarrow{k} X \xrightarrow{f} Y \). An arrow from \( S \xrightarrow{k} \cdots \to S' \xrightarrow{k'} \cdots \) in \( \mathcal{C}_{F(\mathcal{C})} \) is a \( \mathcal{C} \)-arrow \( g : S \to S' \) such that the resulting triangle commutes: i.e. such that \( k = k' \circ g \). An equalizer of \( X, Y \) is then some \( [E, e] \) such the fork that \( E \xrightarrow{e} X \xrightarrow{f} Y \) is terminal in \( \mathcal{C}_{F(\mathcal{C})} \).

Before giving some examples let's immediately note that, just as products are unique up to unique isomorphism, equalizers are too. That is to say,

**Theorem 73.** If both the equalisers \( [E, e] \) and \( [E', e'] \) exist for \( X \xrightarrow{f} Y \), then there is a unique isomorphism \( j : E \xrightarrow{\sim} E' \) (which also fixes what \( e' \) has to be, given \( e \)).

**Proof.** We can use an argument that goes along exactly the same lines as the one we used to prove the uniqueness of products and equalisers. This is of course no accident, given the similarity of the definitions. To spell things out . . .

Assume \( [E, e] \) equalizes \( f \) and \( g \), and suppose \( e \circ h = e \). Then observe that the following diagram will commute.

\[
\begin{array}{ccc}
E & \xrightarrow{e} & X \xrightarrow{f} Y \\
\downarrow{h} & & \downarrow{g} \\
E & &
\end{array}
\]

But \( h = 1_E \) makes that diagram commute, and by hypothesis there is a unique arrow \( E \to E \) which makes the diagram commute. So we can conclude that if \( e \circ h = e \), then \( h = 1_E \).

Now suppose \( [E', e'] \) is also a product. Then \( [E, e] \) must factor uniquely through it. That is to say, there is a (unique) \( v : E \to E' \) such that \( e' \circ v = e \). And since \( [E, e] \) must factor uniquely though \( [E', e'] \) there is a unique \( w \) such that \( e \circ w = e' \). So \( e \circ w \circ v = e \), and hence by our initial conclusion, \( w \circ v = 1_E \). Likewise \( v \circ w = 1_{E'} \). Which makes the unique \( v \) an isomorphism. Which fixes that \( e' = e \circ v^{-1} \).

**Theorem 74.** If \( [E, e] \) constitute an equalizer, then \( e \) is a monomorphism.

**Proof.** Assume \( [E, e] \) equalizes \( X \xrightarrow{f} Y \), and suppose \( e \circ g = e \circ h \), where \( D \xrightarrow{g} E \). Then the following diagram commutes,

\[
\begin{array}{ccc}
D & \xrightarrow{e \circ g = e \circ h} & X \xrightarrow{f} Y \\
\downarrow{g} & & \downarrow{g} \\
E & &
\end{array}
\]

So \( D \xrightarrow{e \circ g} X \xrightarrow{f} Y \) is a fork factoring uniquely through the equalizer, and hence there is only one arrow to do the factoring, i.e. \( g = h \). So \( e \) is left-cancellable in the equation \( e \circ g = e \circ h \); i.e. \( e \) is monic.

Let’s now have some actual examples of equalizers.
(1) Suppose in \( \text{Set} \) we have the functions \( X \xrightarrow{f} Y \). Then consider the set \( E \subseteq X \) such that \( x \in E \) implies \( fx = gx \), and let \( e : E \hookrightarrow X \) be the obvious inclusion map. Then \([E, e]\) is an equalizer for \( f \) and \( g \).

By construction, \( f \circ e = g \circ e \). So suppose \( S \xrightarrow{k} X \xrightarrow{f} Y \) is any fork. Since \( f(k(s)) = g(k(s)) \) for each \( s \in S \), the set \( k[S] = E \) so defining \( u : S \to k[S] \) to agree with \( k : S \to X \) on all inputs will make the diagram for equalizers commute.

It remains to show that this is the unique candidate for the function \( u \). But note \( k = e \circ u \), and \( e \) doesn’t change the values of the function (only its codomain), so \( k \) and \( u \) must indeed agree on all inputs.

(2) Other cases of categories of sets-with-structure are similar. For a simple case, take the category \( \text{Mon} \). Given a pair of monoid homomorphisms \( (X, \cdot) \xrightarrow{f} (Y, \ast) \), take the subset \( E \) of \( X \) on which the functions agree. Then \((E, \cdot)\) is indeed a monoid. Evidently \( E \) must contain the identity element of \( X \) (since \( f \) and \( g \), being homomorphisms must both send it to the identity element of \( Y \)). And suppose \( e, e' \in E \): then \( f(e \cdot e') = f(e) * f(e') = g(e) * g(e') = f(e \cdot e') \), which means that \( E \) is closed under products of members – and so will inherit the behaviour of identity element and the required associativity law.

So take the monoid \((E, \cdot)\) and equip it with the injection map into \((X, \cdot)\): and this will give us an equalizer for \( (X, \cdot) \xrightarrow{f} (Y, \ast) \).

(3) Similarly, take \( \text{Top} \) where we have a set of points equipped with a topology. The obvious equalizer for a pair of continuous maps \( X \xrightarrow{f} Y \) is the subset of the underlying set of \( X \) on which the functions agree with the subspace topology, equipped with the injection into \( X \). (This works because of the way that the subspace topology is defined – we won’t go into details).

(4) A special case. Suppose we are in \( \text{Grp} \) and have two group homomorphisms, \( f : X \to Y \), and the trivial homomorphism \( o \) which sends \( X \) to the identity element in \( Y \), i.e the composite \( X \to 1 \to Y \) of the only possible homomorphisms. Then consider what would constitute an equaliser for \( f \) and \( o \).

Suppose \( E \) is the kernel of \( f \), i.e. the subgroup of \( X \) whose objects are the elements which \( f \) sends to the identity element of \( Y \), and let \( i : E \hookrightarrow X \) be the inclusion map. Then \( E \xrightarrow{f} Y \) is a fork since \( f \circ e = o \circ e \).

Let \( S \xrightarrow{k} X \xrightarrow{f} Y \) be another fork. Now, \( o \circ k \) sends every element of \( S \) to the unit of \( Y \). Since \( f \circ k = o \circ k \), \( k \) must send any element of \( S \) to some element in the kernel \( E \). So let \( k' : S \to E \) agree with \( k : S \to X \) on all arguments.

Then the following commutes:

\[
\begin{array}{c}
S \\
| k' \downarrow \\
X \\
| i \downarrow \\
E \\
\end{array}
\xrightarrow{f} Y
\]

And evidently \( k' \) is the only possible homomorphism to make the diagram commute.
So the equaliser of \( f \) and \( o \) is \( f \)'s kernel \( E \) equipped with the inclusion map into the domain of \( X \). Or putting it the other way about, we can define kernels of group homorphisms categorically in terms of equalizers.

(5) Finally we remark that the equalizer of a pair of maps \( X \xrightarrow{f} Y \) where in fact \( f = g \) is simply \([X, 1_X]\).

Consider then a poset \((P, \leq)\) considered as a category whose objects are the members of \( P \) and where there is a unique arrow \( X \to Y \) (for \( X, Y \in P \)) iff \( X \leq Y \). So the only cases of parallel arrows from \( X \) to \( Y \) are cases of equal arrows which then, as remarked, have equalizers. So in sum, a poset category has all possible equalizers.

We could now generalize to define equalizers for more-than-two parallel arrows in obvious ways, and indeed define the trivial unary and null cases. But we don’t need to, so we won’t.

Instead, we’ll make a different point. We saw that in \( \textbf{Set} \) the equalizer of two parallel arrows from an object \( X \) is a certain subset of \( X \) equipped with the trivial inclusion function. We now note that the reverse holds too:

**Theorem 75.** A subset \( S \) of \( X \), equipped with the inclusion map \( i: S \hookrightarrow X \), is an equalizer in \( \textbf{Set} \) of certain parallel arrows from \( X \).

**Proof.** Given \( S \subseteq X \), there is a corresponding ‘characteristic function’ \( s: X \to \{0, 1\} \) which sends \( x \in X \) to \( 1 \) (‘true’).

Another function from \( X \) to \( \{0, 1\} \) is the cheerfully indiscriminate map \( t \) which sends everything in \( X \) to \( 1 \).

We show that \([S, i]\) is an equalizer for \( X \xrightarrow{s} \{0, 1\} \). First, it is trivial that \( s \circ i = t \circ i \). So we now show that any upper fork in this diagram factors through the lower fork via a unique suitable \( u \):

\[
\begin{array}{c}
P \xrightarrow{f} X \xrightarrow{s} \{0, 1\} \\
\text{u} \\
S \xrightarrow{i} X \xrightarrow{t}
\end{array}
\]

Since \( s \circ f = t \circ f \), \( f[Z] \subseteq S \subseteq S \). So if we put \( u: P \to S \) to agree with \( f: P \to X \) on all inputs, then the diagram commutes. And this \( u \) is evidently the only possible candidate.

Which gives us a nice categorial way of dealing with subobjects in \( \textbf{Set} \). Later we’ll find a different way of talking about subobjects in categories much more generally, and we will then have to investigate its relation to the sort of construction for \( \textbf{Set} \) that we’ve just met.

**12.10 Co-equalizers**

We dualize our definition of an equalizer to get the notion of a co-equalizer. So, as a preliminary, we say

**Definition 69.** A co-fork (from \( X \) through \( Y \) to \( S \) ) consists of parallel arrows \( f: X \to Y \), \( g: X \to Y \) and an arrow \( k: Y \to S \), such that \( k \circ f = k \circ g \).
(Actually, plain ‘fork’ is used for the dual too; but the barbarous ‘co-fork’ keeps things clear.) So diagrammatically, a co-fork looks like this: \( X \xrightarrow{f,g} Y \xrightarrow{k} S \), with the composite arrows from \( X \) to \( S \) being equal.

**Definition 70.** Let \( \mathcal{C} \) be a category and \( f: X \to Y \) and \( g: X \to Y \) be a pair of parallel arrows in \( \mathcal{C} \). Then the object \( C \) and arrow \( c: Y \to S \) form a co-equalizer in \( \mathcal{C} \) for those arrows iff \( c \circ f = c \circ g \), and for any co-fork from \( X \) through \( Y \) to \( S \) there is a unique arrow \( u: C \to S \) such the following diagram commutes:

\[
\begin{array}{ccc}
X & \xrightarrow{f,g} & Y \\
& k \downarrow & \downarrow c \\
S & & C
\end{array}
\]

As we would now expect, by a dual argument, co-equalizers are unique up to a unique isomorphism; and since equalizers are monic, co-equalizers are epic. We won’t pause to spell out or prove those results, but turn immediately to give just one central example.

Work in \( \textbf{Set} \). The parallel arrows \( f, g: X \to Y \) determine a relation \( R \) on the members of \( Y \) where \( y_1Ry_2 \) holds when there is an \( x \in X \) such that \( f(x) = y_1 \land g(x) = y_2 \).

Now if there is a co-fork \( X \xrightarrow{f,g} Y \xrightarrow{k} S \), then if \( y_1Ry_2 \) then \( k(y_1) = k(y_2) \) (and of course, being mapped to the same value by \( k \) is an equivalence relation). So, for every co-fork, there is a corresponding equivalence relation such that being \( R \)-related implies being in that equivalence relation. What’s the limiting case of such an equivalence relation? It will have to be \( R^{\sim} \), the smallest equivalence relation containing \( R \).

That observation gives us the clue about how to proceed.

**Theorem 76.** Given functions \( f, g: X \to Y \) in \( \textbf{Set} \), let \( R^{\sim} \) be the smallest equivalence relation containing \( R \) – where \( y_1Ry_2 \iff (\exists x \in X) (f(x) = y_1 \land g(x) = y_2) \).

Let \( C \) be \( Y/R^{\sim} \), i.e. the set of \( R^{\sim} \)-equivalence classes of \( Y \); and let \( c \) map \( y \in Y \) to the \( R^{\sim} \)-equivalence class containing \( y \). Then \([C, c]\), so defined, is a co-equalizer for \( f \) and \( g \).

**Proof.** First we need to check that \( c \circ f = c \circ g \). But the left-hand side sends \( x \in X \) to the \( R^{\sim} \)-equivalence class containing \( f(x) \) and the right-hand side sends \( x \) to the \( R^{\sim} \)-equivalence class containing \( g(x) \). However, \( f(x) \) and \( g(x) \) are by definition \( R \)-related, and hence are \( R^{\sim} \)-related: so by construction they belong to the same \( R^{\sim} \)equivalence class.

Now suppose again there is a co-fork \( X \xrightarrow{f,g} Y \xrightarrow{k} S \), so \( k \circ f = f \circ g \). We next outline a proof that if \( y_1R^{\sim}y_2 \) then \( k(y_1) = k(y_2) \).

Start with \( R \) and let \( R' \) be its reflexive closure. Obviously we’ll still have that if \( y_1R'y_2 \) then \( k(y_1) = k(y_2) \). Now consider \( R'' \) the symmetric closure of \( R' \); again obviously, we’ll still have that \( y_1R''y_2 \) then \( k(y_1) = k(y_2) \). Now note that if \( y_1R''y_2 \) and \( y_2R''y_3 \), then \( k(y_1) = k(y_3) \). So if we take the transitive closure of \( R'' \), we’ll still have a relation which, when it holds between some \( y_1 \) and \( y_2 \), implies that \( k(y_1) = k(y_2) \). But the transitive closure of \( R'' \) is \( R^{\sim} \).

We have shown, then, that \( k \) is constant on members of a \( R^{\sim} \)-equivalence class, and so we can well-define a function \( u: C \to S \) which sends an equivalence class to the value of \( k \) on a member of that class, which makes the diagram above commute. Moreover, since \( c \) is surjective and \( C \) only contains \( R^{\sim} \)-equivalence classes, \( u \) is the only function for which \( u \circ c = k \).
In this chapter, we first define an official class of \textit{limits}, and the corresponding dual class of \textit{co-limits}. Our examples from the previous chapter will indeed all fall into these classes. We then give a new pair of illustrations, so-called pullbacks and pushouts.

\section{Defining limits}

There’s obviously a family resemblance between terminal objects, products, equalizers: roughly, each is a kind of limiting case with unique arrows pointing towards it.

As we’ve already seen, we can sharpen that up:

1. A terminal object in $\mathcal{C}$ is . . . wait for it! . . . terminal in the given category $\mathcal{C}$.
2. The product of $X$ with $Y$ in $\mathcal{C}$ is a terminal object in the derived category $\mathcal{C}_W(X,Y)$ of wedges to $X$ and $Y$.
3. The equalizer of parallel arrows through $X$ to $Y$ in $\mathcal{C}$ are (parts of) terminal objects in the derived category $\mathcal{C}_F(XY)$ of forks through $X$ to $Y$.

And that gives us a clue about how to generalize.

Way back in Defn. 3, we characterized a diagram $D$ in a category as being simply a bunch of objects $D_j$ with some arrows between some of them; and recall we allow the limiting cases(!) where there are no arrows, or even no objects.

\begin{definition}
Let $D$ be a diagram in category $\mathcal{C}$. Then a \textit{cone} over $D$ is an object $C \in \mathcal{C}$ (the vertex of the cone), together with an arrow $c_j : C \to D_j$ for each object $D_j$ in $D$, such that whenever there is an arrow $d : D_k \to D_l$ in $D$, the following diagram commutes:

\begin{equation}
\begin{array}{ccc}
D_k & \xrightarrow{d} & D_l \\
\downarrow^{c_k} & & \downarrow_{c_l} \\
C & \xleftarrow{c_j}
\end{array}
\end{equation}

We use $[C, c_j]$ as our notation for such a cone.

Think of it diagrammatically like this: arrange the objects in the diagram $D$ in a plane with whatever arrows there are between them. Now sit the object $C$ above the plane, with a quiverful of arrows from $C$ zinging down, one to each object in the plane. Those arrows form a skeletal cone. And any triangles thus formed with $C$ at the apex must commute.
CHAPTER 13. LIMITS AND COLIMITS DEFINED

Now, the cones \([C, c_j]\) over a given diagram \(D\) in \(\mathcal{C}\) form a category in a very natural way:

**Definition 72.** Given a diagram \(D\) in category \(\mathcal{C}\), the derived category \(\mathcal{C}(D)\) — the category of cones over \(D\) — has the following data:

1. Its objects are the cones \([C, c_j]\) over \(D\).
2. An arrow from \([C, c_j]\) to \([C', c'_j]\) is any \(\mathcal{C}\)-arrow \(f: C \to C'\) such that \(c'_j \circ f = c_j\) for all \(j\). That is to say, in a diagram like

\[
\begin{array}{ccc}
C & \rightarrow^f & C' \\
\downarrow^{c_j} & & \downarrow^{c'_j} \\
D_j & \rightarrow^d & D_k
\end{array}
\]

(Imagine a triangle for each object \(D_i\) in \(D\)) all the triangles commute.

We can leave it as a routine exercise to confirm that \(\mathcal{C}(D)\) is indeed a category, and instead move on to the obvious next definition:

**Definition 73.** A limit for \(D\) in \(\mathcal{C}\) is a terminal object in \(\mathcal{C}(D)\).

Being a terminal objects, we immediately have

**Theorem 77.** Limit cones over a given diagram \(D\) are unique up to unique isomorphism.

And now let’s immediately confirm that our three examples of limits so far are indeed limit cones.

1. Let’s get the null case out of the way. Start with the empty diagram in \(\mathcal{C}\) — zero objects and so, necessarily, no arrows. Then a cone over the empty diagram is simply an object \(C\) (there is no further condition to fulfill), and an arrow between such cones is just an arrow between objects in \(\mathcal{C}\). So the category of cones over the empty diagram is just the category \(\mathcal{C}\) we started with, and a limit cone is just a terminal object in \(\mathcal{C}\)!

2. Consider now a diagram which is just two objects we’ll call \(D_1\), \(D_2\), still with no arrow between them. Then a cone over such a diagram is just a wedge into \(D_1, D_2\); and a limit cone in the category of such cones (wedges) is simply a product.

3. Next consider a diagram which is again two objects, but now with two parallel arrows between them, which we can represent \(D_1 \rightarrow^d D_2\). Then a cone over this diagram is a commuting diagram like this:

\[
D_1 \rightarrow^d D_2
\]

If there is such a diagram, then we must have \(d \circ c_1 = d' \circ c_1\); and vice versa, if that identity holds, then we can put \(c_2 = d \circ c_1 = d' \circ c_1\) to complete the
commutative diagram. Hence we have a cone from the vertex $C$ to our diagram iff
\[
C \xrightarrow{c_1} D_1 \xrightarrow{d} D_2
\]
is a fork.

Since $c_1$ fixes what $c_2$ has to be to complete the cone, we can focus on the cut-down cone consisting of just $[C,c_1]$. In this case, the corresponding cut-down terminal cone is the object and arrow $[E,e]$ such that $E \xrightarrow{e} D_1 \xrightarrow{d} D_2$ is the terminal fork in the category of forks through $D_1$ to $D_2$. Hence $[E,e]$ is the equalizer of our diagram.

Now, we initially defined products without talking of categories of wedges, and first defined equalizers without talking of categories of forks. Similarly, we can of course recast our definition of limits – i.e. limit cones – without going via the notion of a category of cones. As you would expect, we could more directly say

**Definition 74** (Alternative). A cone $[L,\pi_j]$ over a diagram $D$ in $\mathcal{C}$ is a limit (cone) iff for any cone $[C,c_j]$ over $D$ there is a unique arrow $u: C \rightarrow L$ such that, for all indexing objects $j$ in $D$, $\pi_j \circ u = c_j$.

We can then give a direct proof from this alternative definition of Theorem 77, that limit cones over a given diagram $D$ are unique up to unique isomorphism. The line of argument is the one we met before for the special case of products. But just for the record:

**Proof.** First we note that $[L,\pi_j]$ factors through itself via the identity $1_L: L \rightarrow L$. But by definition, cones over $D$ uniquely factors through the limit, so that means that if

(i) $\pi_j \circ u = \pi_j$ for all $j$, $u = 1_L$.

Now suppose $[L',\pi'_j]$ is another limit cone over $D$. Then $[L',\pi'_j]$ uniquely factors through $[L,\pi_j]$, via some $f$, so

(ii) $\pi_j \circ f = \pi'_j$ for all $j$.

And likewise $[L,\pi_j]$ uniquely factors through $[L',\pi'_j]$ via some $g$, so

(iii) $\pi'_j \circ g = \pi_j$ for all $j$.

Whence

(iv) $\pi_j \circ f \circ g = 1_L$ for all $j$.

Therefore

(v) $f \circ g = 1_L$.

And symmetrically

(vi) $g \circ f = 1_{L'}$.

Whence $f$ is not just unique (by hypothesis, the only way of completing the relevant diagrams) but an isomorphism. \[\Box\]

**Theorem 78.** Suppose $[L,\pi_j]$ is a limit cone over a diagram $D$ in $\mathcal{C}$, and $[L',\pi'_j]$ is another cone over $D$ which factors through $[L,\pi_j]$ via an isomorphism $f$. Then $[L',\pi'_j]$ is also a limit cone.
Proof. Take any cone \([C, c_j]\) over \(D\). We need to show that (i) there is an arrow \(v: C \to L'\) such that for all indexing objects \(j\) in \(D\), \(c_j = \pi'_j \circ v\), and (ii) it is unique.

But we know that there is a unique arrow \(u: C \to L\) such that for \(j\), \(c_j = \pi_j \circ u\).

And we know that \(f: L' \to L\) and \(\pi'_j = \pi_j \circ f\) (so \(\pi'_j = \pi_j \circ f^{-1}\)).

Therefore put \(v = f^{-1} \circ u\), and that satisfies (i).

Now suppose there is another arrow \(v': C \to L'\) such that \(c_j = \pi'_j \circ v'\). Then we have \(f \circ v': C \to L\), and also \(c_j = \pi_j \circ f \circ v'\). Therefore \([C, c_j]\) factors through \([L, \pi_j]\) via \(f \circ v'\), so \(f \circ v' = u\). Whence \(v' = f^{-1} \circ u = v\). Which proves (ii).

Finally, we note some standard

**Notation.** We denote the object at the vertex of a limit cone for the diagram \(D\) with objects \(D_j\) by \(\lim_{\leftarrow j} D_j\).

The ‘projection’ arrows from this limit object to the various objects \(D_j\) can naturally be denoted \(\pi_i: \lim_{\leftarrow j} D_j \to D_i\) (though many use \(p_i\) rather than \(\pi_i\)).

The limit cone could therefore be represented by \([\lim_{\leftarrow j} D_j, \pi_{j}]\).

The direction of the arrow under ‘lim’ in this notation is perhaps unexpected, but we just have to learn to live with it.

Note, too, that some prefer to say more austerely that a cone is (not an object-with-a-family-of-arrows but just) a family of arrows. On such a usage, \(\pi_i: \lim_{\leftarrow j} D_j \to D_i\) will be said to represent the limit cone itself. Since we can read off the vertex of the cone as the common domain of those arrows, there’s no loss of information here. It is merely a matter of convenience whether we speak austerely or instead, more verbosely, mention the vertex. So sometimes, to avoid long-windedness, it will be convenient to read our notation \([C, c_j]\) austerely too, as strictly speaking just picking out a bunch of arrows from which we can read off a common vertex.

### 13.2 Limits and diagrams-as-functors

So far, our story about limit cones has been presented in terms of the idea of a diagram-in-a-category - \(\mathcal{C}\) thought of as a (possibly very small) fragment of a category – i.e. just some objects and arrows.

Note, though, that in our three concrete examples so far – cones over the null diagram, over discrete two-object diagrams, over two parallel arrows – if would have made no difference to the story at all if we’d taken the diagrams to be three miniature subcategories, i.e. the empty category, a discrete two object category (just add identity arrows), a category with two parallel arrows (just add identity arrows again).

And from now on we are going to concentrate on this kind of case, where diagrams have the identity arrows and composites required to form a sub-category. And recall that we have already noted before another way of thinking about such a diagram: i.e. as a ‘picture’-in-\(\mathcal{C}\) of another (possibly very small) category \(\mathcal{J}\), with the picture projected onto \(\mathcal{C}\) by a functor \(D: \mathcal{J} \to \mathcal{C}\). See the beginning of §5.2, and then §11.1, Defn. 41 – except that now we follow a common convention and use the notation \(\mathcal{J}\) rather than \(\mathcal{F}\) when a small – typically very small – category is likely to be in focus.

So using this notion of diagrams-as-functors, the corresponding definitions for cones and limit cones over such diagrams go as follows (entirely as you would predict):

**Definition 75.** Suppose we are given a category \(\mathcal{C}\), together with \(\mathcal{J}\) a small (possibly very small) category, and a diagram-as-functor \(D: \mathcal{J} \to \mathcal{C}\). Then:
A cone over $D$ is an object $C \in \mathcal{C}$, together with an arrow $c_J : C \to D(J)$ for each $J \in \text{ob}(\mathcal{J})$, such that for any arrow $d : K \to L$ in $\text{arr}(\mathcal{J})$, $c_L = D(d) \circ c_K$. We use $[C, c_J]$ (where ‘$J$‘ is understood to run over objects in $\mathcal{J}$) for such a cone.

(2) A limit cone over $D$ is a cone we can notate $\lim_J D, \pi_J$ such that for every cone $[C, c_J]$ over $D$, there is a unique arrow $u : C \to \lim_J D$ such that, for all $J \in \text{ob}(\mathcal{J})$, $\pi_J \circ u = c_J$.

(3) $\mathcal{C}$ has limits of shape $\mathcal{J}$ iff for every diagram $D : \mathcal{J} \to \mathcal{C}$, there is a limit cone over $D$.

Two comments:

(i) To re-iterate. Not all diagrams in the original sense of fragments-of-a-category $\mathcal{C}$ are diagrams in the sense of pictures of a category $\mathcal{J}$ in $\mathcal{C}$. So not all (limit) cones over a diagram in the more general first sense are (limit) cones over diagrams in the second narrower sense. You need to be aware that different presentations in fact use the different definitions: e.g. compare Borceux (1994) and Leinster (2014). Still, the distinction is one that doesn’t seem to weigh much in practice. There’s a neatness to treating diagrams the second way, so that is what we mostly follow here from now on.

(ii) On our initial definition, if we start with a discrete sub-category $D$ consisting in two objects $D_0$ and $D_1$ with no arrows between them, then a limit cone over $D$ will be a binary product of $D_0$ with $D_1$. If we correspondingly take $\mathcal{J}$ to be the two object discrete category $\mathcal{T}$ with two objects 0, 1, then a limit cone over a diagram-as-functor $D : \mathcal{T} \to \mathcal{C}$ will be a product of $D(0)$ with $D(1)$ – but that may not be a strictly binary product as we’ve not required the functor $D$ to be faithful, and we could have $D(0) = D(1)$.

13.3 Another characterization of limits

(a) Recall Theorem 68 which told us that a product can be identified with a universal element for a certain functor. We can now prove a much more general result with the same flavour.

Let $D : \mathcal{J} \to \mathcal{C}$ be a diagram as usual. Define the functor $\text{Cone}(-, D) : \mathcal{C}^{\text{op}} \to \text{Set}$ as follows. $\text{Cone}(-, D)$ sends an object $C \in \mathcal{C}$ to $\text{Cone}(C, D)$, the set of cones over $D$ with vertex $C$. And $\text{Cone}(-, D)$ sends a $\mathcal{C}$-arrow $f : C' \to C$ to the arrow $\text{Cone}(f) : \text{Cone}(C, D) \to \text{Cone}(C', D)$, which takes a cone $[C, \pi_J]$ and sends it to $[C', \pi_J \circ f]$. It is easily checked that this is a functor.

Now simply apply Defn. 44. Then a universal element of $\text{Cone}(-, D) : \mathcal{C}^{\text{op}} \to \text{Set}$ is a pair $(L, [L, \pi_J])$, where $L \in \mathcal{C}$ and $[L, \pi_J]$ is in $\text{Cone}(L, D)$, the set of cones over $D$ with vertex $L$. And moreover, we require that for each $C \in \mathcal{C}$ and each cone $[C, c_J]$, there is a unique map $f : Z \to L$ such that $\text{Cone}(f)[L, \pi_J] = [C, c_J]$, which requires $\pi_J \circ f = c_J$ for each $J$. But that’s just to say that $[L, \pi_J]$ is a limit cone. Hence

**Theorem 79.** A limit cone over $D$ [together with its vertex] is a universal element for $\text{Cone}(-, D)$.

(b) Suppose we take some suitable small category $\mathcal{J}$. Consider the functor category $[\mathcal{J}, \mathcal{C}]$ whose objects are diagrams-as-functors $D : \mathcal{J} \to \mathcal{C}$ and whose arrows are natural transformations between such functors.

One particular kind of object in $[\mathcal{J}, \mathcal{C}]$ is the trivial constant functor $\Delta_C : \mathcal{J} \to \mathcal{C}$ that sends every object in $\mathcal{J}$ to the object $C$ and every arrow in $\mathcal{J}$ to 1$_C$. 
Now, what would be a natural transformation from $\Delta_C$ to another diagram-as-functor $D$? Applying the definition, it would be a family $\alpha$ of $J$ arrows $\alpha_J : \Delta_C(J) \to D(J)$ indexed by $J \in J$, i.e. $\alpha_J : C \to D(J)$, such that for every $d : K \to L$ in $J$, the diagram on the left commutes in $C$ and hence so does the one on the right:

$$
\begin{array}{ccc}
C & \xrightarrow{1_C} & C \\
\downarrow & & \downarrow \\
D(K) & \xrightarrow{D(d)} & D(L)
\end{array}
\Rightarrow
\begin{array}{ccc}
C & \xrightarrow{\alpha_K} & D(K) \\
\downarrow & & \downarrow \\
\alpha_L : C & \xrightarrow{\alpha_J} & D(L)
\end{array}
$$

But we now recognize that! So – for convenience thinking of cones in this context in the austere way – a cone over $D$ with vertex $C$ is simply a natural transformation from the trivial functor $\Delta_C$ to $D$.

Hence $\text{Cone}(C, D)$, the set of cones over $D$ with vertex $C$ is the hom-set of arrows in the functor category $[J, C]$ from $\Delta_C$ to $D$. And hence the functor $\text{Cone}(-, D)$ can also be thought of as a functor $[J, C](\Delta_C, D)$ that takes an element $C$ of $C$ to the hom-set $[J, C](\Delta_C, D)$. A limit cone over $D$, we saw, is a universal element for $\text{Cone}(–, D)$, so can also therefore be thought of as a universal element for $[J, C](\Delta_C, D)$.

### 13.4 Colimits defined

The headline, and thoroughly predictable, story is: reverse the relevant arrows and you get a definition of colimits.

So, proceeding in the style of §13.1, and wrapping everything together, we get:

**Definition 76.** Let $D$ be a diagram in category $C$. Then a cocone under $D$ is an object $C \in C$, together with an arrow $c_j : D_j \to C$ for each object $D_j$ in $D$, such that whenever there is an arrow $d : D_k \to D_l$ in $D$, the following diagram commutes:

$$
\begin{array}{ccc}
D_k & \xrightarrow{d} & D_l \\
\downarrow & & \downarrow \\
C & \xrightarrow{c_j} & C
\end{array}
$$

The cocones under $D$ form a category with objects the cocones $[C, c_j]$ and an arrow from $[C, c_j]$ to $[C', c'_j]$ being any $C'$-arrow $f : C \to C'$ such that $c'_j = f \circ c_j$ for all indexes $j$. A colimit for $D$ is a terminal object in the category of cocones under $D$. It is standard to denote the object at the vertex of the colimit cocone for the diagram $D$ by $\text{lim} \rightarrow_{j} D_j$.

And it is now routine to confirm that our earlier examples of initial objects, coproducts and co-equalizers do count as colimits.

1. The null case where we start with the empty diagram in $C$ gives rise to a cocone which is simply an object in $C$. So the category of cocones over the empty diagram is just the category $C$ we started with, and a limit cocone is just an initial object in $C$!

2. Consider now a diagram which is just two objects we’ll call ‘$D_1$, ‘$D_2$', still with no arrow between them. Then a cocone over such a diagram is just a corner from $D_1, D_2$; and a limit cocone in the category of such cocones is simply a coproduct.

3. And if we start with the diagram $D_1 \xrightarrow{d \neq d'} D_2$. Then a limit cocone over this diagram gives rise to a co-equalizer.

The arguments for the last two cases are exactly dual to the arguments in §13.1, and we need not pause to spell them out.
13.5 More examples: pullbacks and pushouts

(a) We will briefly explore in this chapter just one more kind of limit and its dual. Take a co-wedge or, as I prefer to say, a corner $D$ in category $\mathcal{C}$, i.e. a diagram like this:

\[
\begin{array}{c}
D_3 \\
\downarrow^e \\
D_1 \xrightarrow{d} D_2
\end{array}
\]

You can think of this, if you want, as the picture in $\mathcal{C}$ of the category which looks like this:

\[
\begin{array}{c}
\cup \\
\bullet \\
\downarrow
\end{array}
\]

Check that adding identity arrows to our original diagram won’t make any difference to the ensuing arguments.

A cone over our diagram has a rather familiar shape, i.e. it is a commutative square:

\[
\begin{array}{c}
C \\
\downarrow^{c_1} \\
D_1 \xrightarrow{d} D_2 \\
\downarrow^{c_2} \\
D_3 \xrightarrow{\mathcal{C}} e
\end{array}
\]

Though note, we needn’t really draw the diagonal here, for if the sides of the square commute thus ensuring $d \circ c_1 = e \circ c_3$, then we know the diagonal $c_2$ exists making the triangles commute.

And a limit cone of this type will be a cone with apex $L = \lim_{\rightarrow} D_j$ and projections $\pi_j: L \rightarrow D_j$ such that for any cone $[C, c_j]$ over $D$, there is a unique $u: C \rightarrow L$ such that this whole diagram commutes:

\[
\begin{array}{c}
C \\
\downarrow^{c_1} \\
D_1 \xrightarrow{d} D_2 \\
\downarrow^{c_2} \\
D_3 \xrightarrow{\mathcal{C}} e
\end{array}
\]

(And note that if this commutes, there’s just one possible $\pi_2$ and $c_2$ we can draw in which makes it still commute.)

**Definition 77.** A limit for a corner diagram is a *pullback*. The whole square formed by the original corner and its limit is a *pullback square*.

(b) Let’s immediately have some examples from $\textbf{Set}$. Here, a corner comprises (changing the labelling) three sets $X, Y, Z$ and a pair of functions which share the same codomain, $f: X \rightarrow Z, \ g: Y \rightarrow Z$.

We know from the top left of the diagram above that $L$ must be product-like; and to get the other part of the diagram to commute, the pullback square must have at its apex
L (something isomorphic to) \{\langle x, y \rangle \in X \times Y \mid f(x) = g(y)\} with the obvious projection maps to X and Y.

So suppose that X and Y are subsets of Z, and f and g are the corresponding inclusion maps. Then \(L \cong \{\langle x, y \rangle \in X \times Y \mid x = y\} = \{\langle z, z \rangle \mid z \in X \cap Y\} \cong X \cap Y\). In short: in Set, the intersection of a pair of sets is their pullback object.

Take another case. Suppose in the corner formed by \(X, Y, Z\), and functions \(f: X \to Z\), \(g: Y \to Z\) we in fact have \(Y = Z\) and \(g = 1_Z\). Then the pullback object \(L \cong \{\langle x, z \rangle \in X \times Z \mid f(x) = z\} \cong f^{-1}[Z]\), the inverse image of Z: the associated projection maps are just the familiar projection maps for ordered pairs. So we have

\[
\begin{array}{ccc}
X & \overset{f}{\longrightarrow} & Z \\
\downarrow{\pi_1} & & \downarrow{1_Z} \\
Y & \overset{g}{\longrightarrow} & Z
\end{array}
\]

In short, in Set, the inverse image of a function is also a pullback object.

(c) Why ‘pullback’? Look at that last diagram. We can say that we get to \(f^{-1}[Z]\) from Z by pulling back along \(f\) – or more accurately, we get to \(\pi_1: f^{-1}[Z] \to X\) by pulling back the identity arrow on Z along \(f\). In this sense,

**Theorem 80.** Pulling back a monomorphism yields a monomorphism

In other words, if we start with a corner or corner

\[
\begin{array}{ccc}
Y & \overset{g}{\longrightarrow} & Z \\
\downarrow{f} & & \downarrow{g} \\
X & \overset{j}{\longrightarrow} & Z
\end{array}
\]

with \(g\) monic, and can pullback \(g\) along \(f\) to give a pullback square

\[
\begin{array}{ccc}
L & \overset{b}{\longrightarrow} & Y \\
\downarrow{a} & & \downarrow{g} \\
X & \overset{f}{\longrightarrow} & Z
\end{array}
\]

then the resulting arrow \(a\) is monic. (This does not depend on the character of \(f\).)

**Proof.** Suppose, for some arrows \(C \overset{j}{\longrightarrow} L\), \(a \circ j = a \circ k\). Then \(g \circ b \circ j = f \circ a \circ j = f \circ a \circ k = g \circ b \circ k\). Hence, given that \(g\) is monic, \(b \circ j = b \circ k\).

It follows, therefore, that the two cones over the original corner, \(X \overset{a \circ j}{\longleftarrow} C \overset{b \circ j}{\longrightarrow} Y\) and \(X \overset{a \circ k}{\longleftarrow} C \overset{b \circ k}{\longrightarrow} Y\) are in fact the same cone, and hence must factor through the limit L via the same unique arrow \(C \to L\). Which means \(j = k\).

In sum, \(a \circ j = a \circ k\) implies \(j = k\), so \(a\) is monic. \(\square\)

Here’s another result about monomorphisms and pullbacks:

**Theorem 81.** The arrow \(f: X \to Y\) is a monomorphism in \(\mathcal{C}\) if and only if the following is a pullback square:
Proof. Suppose this is pullback diagram. That means that any cone $X \xleftarrow{a} C \xrightarrow{b} Y$ over the corner $\xrightarrow{f} Y \xleftarrow{f'} X$ must uniquely factor through the limit with vertex $X$. That is to say, if $f \circ a = f \circ b$, then there is a $u$ such that $a = 1_X \circ u$ and $b = 1_X \circ u$, hence $a = b$ — so $f$ is monic.

Conversely, if $f$ is monic, then given any cone $X \xleftarrow{a} C \xrightarrow{b} Y$ over the original corner, $f \circ a = f \circ b$, whence $a = b$. But that means the cone factors through the cone $X \xleftarrow{1_X} X \xrightarrow{1_X} X$ via the unique $a$, making that cone a limit and the square a pullback square.

(d) We’ve explained, up to a point, the label ‘pullback’. It should now be noted in passing that a pullback is sometimes called a fibered product (or fibre product) because of a construction of this kind on fibre bundles in topology. Those who know some topology can chase up the details.

But here’s a way of getting products into the story, using an idea that we already know about. Recall the definition of the slice category $\mathcal{C}/Z$. Its objects are arrows $f: C \to Z$ for $C \in \mathcal{C}$, and an arrow from $f: X \to Z$ to $g: Y \to Z$ is an arrow $h: X \to Y$ such that $f = g \circ h$ in $\mathcal{C}$.

Now the pullback of the corner formed by $f$ and $g$ in $\mathcal{C}$ is a pair of arrows $a: L \to X$ and $b: L \to Y$ such that $f \circ a = g \circ b$ ($= k$) and which form a wedge such that any other wedge $a': L' \to X$, $b': L' \to Y$ such that $f \circ a' = g \circ b'$ ($= k'$) factors uniquely through it.

Looked at as a construction in $\mathcal{C}/Z$, this means taking two $\mathcal{C}/Z$-objects $f$ and $g$ and getting a pair of $\mathcal{C}/Z$-arrows $a: k \to f$, $b: k \to g$ (check that $a$ is indeed a $\mathcal{C}/Z$-arrow from $f \circ a$ as $\mathcal{C}$-arrow to $f$!). And this pair of arrows forms a wedge such that any other wedge $a': k' \to f$, $b': k' \to g$ factors uniquely through it. In other words, the pullback in $\mathcal{C}$ is a product in $\mathcal{C}/Z$.

Product notation is often used for pullbacks, thus:

$$X \times_Z Y \to Y$$

with the little corner conventionally indicating it is indeed a pullback square.

(e) As you would expect by now, all this dualizes. Suppose we take a wedge $D$, i.e. a diagram that, with appropriate labelling, looks like this: $D_1 \xleftarrow{d} D_2 \xrightarrow{e} D_3$. A cocone under this diagram is another commutative square (omitting again the diagonal arrow which is fixed by the others).

$$D_2 \xrightarrow{e} D_3$$

$$D_1 \xrightarrow{c_1} C$$
And a limit cocone of this type will be a cocone with apex \( L = \lim_{\to j} D_j \) and projections \( \pi_j: L \to D_j \) such that for any cocone \([C, c_j]\) under \( D \), there is a unique \( u: L \to C \) such that the obvious dual of the whole pullback diagram above commutes.

**Definition 78.** A limit for a wedge diagram is a pushout.

Now, in \( \text{Set} \), we get the limit object for a corner diagram \( X \xrightarrow{f} Z \xleftarrow{g} Y \) by taking a certain subset of a product \( X \times Y \). Likewise we get the colimit object for a wedge diagram \( X \xleftarrow{f} Z \xrightarrow{g} Y \) by taking a certain quotient of a coproduct \( X \amalg Y \). (Recall from our discussion of (co)-equalizers that quotients are categorially dual to subsets in \( \text{Set} \).) In fact, we need to quotient out by the smallest equivalence relation containing the relation \( R \) where \( \langle x, 0 \rangle R \langle y, 1 \rangle \) iff there is a \( z \) such that \( f(z) = x \land g(z) = y \). We won’t pause over this. Though it does illustrate how colimits can tend to seem messier than limits.
We have seen, then, that a whole range of familiar constructions from various areas of ordinary mathematics can be regarded as instances of taking limits or colimits of (very small) diagrams in appropriate categories. Examples so far include: forming cartesian products or logical conjunctions, taking disjoint unions or free products, quotienting out by an equivalence relation, taking inverse images.

Now, not every familiar kind of construction involves taking (co)limits. We’ll later see that exponentiation involves a different idea. But plainly we are mining a very rich seam here.

It would get tedious, however, to explore what it takes for a category to have limits for various further kinds of diagram case-by-case, even if we just stick to very small examples. But fortunately we don’t need to do a case-by-case examination. It turns out that if a category has all finite products and has equalizers, then it has all finite limits – and the same goes if we replace ‘finite’ by ‘small’.

This chapter explains and proves that important claim. We could go straight to the abstract general argument (it isn’t difficult): but it will hopefully be illuminating if we take things in three gentle stages.

14.1 Pullbacks, products and equalizers related

As a warm-up exercise, in this section we prove a miniaturized version of our later main result. We will show that, if a category has binary products and equalizers for any pair of parallel arrows then it has pullbacks for any corner.

First, we pause to note an almost-converse of that:

Theorem 82. Suppose $\mathcal{C}$ has a pullback for any corner and also a terminal object: then $\mathcal{C}$ has (i) binary products, and hence (ii) equalizers.

Proof. For (i), note that the pullback of the corner $\begin{array}{ccc} X & \xrightarrow{f} & Y \\ \downarrow{g} & & \downarrow{\delta_Y} \\ 1 & \xleftarrow{} & Y \end{array}$ is just the product $X \times Y$ with the usual projection functions.

For (ii), think of parallel arrows $\begin{array}{ccc} X & \xrightarrow{g} & Y \\ \downarrow{f} & & \downarrow{\delta_Y} \\ \end{array}$ as a wedge $\begin{array}{ccc} Y & \xleftarrow{f} & X \\ \downarrow{g} & & \downarrow{\delta_Y} \\ \end{array}$, which factors uniquely via $u = (f, g)$ through the product $Y \times Y$.

So now consider the pullback for the corner $\begin{array}{ccc} X & \xrightarrow{u} & Y \times Y \\ & & \xleftarrow{\delta_Y} Y \end{array}$, where $\delta_Y$ is the ‘diagonal’ arrow (see Defn. 58):
Then \([E,e]\) form an equalizer for the parallel pair \(f,g\).

For \(E \xrightarrow{e} X \xrightarrow{g} Y\) is a fork. Take any other fork \(C \xrightarrow{c} X \xrightarrow{g} Y\). Then the wedge \(X \xleftarrow{c} C \xrightarrow{f \circ c} Y\) must factor through \(E\) (because \(E\) is the vertex of the pullback) via a unique \(v\). Which is to say that there is a unique arrow \(v: C \to E\) through which the second fork factors through the first one, making the fork a limiting case, and \([E,e]\) an equalizer.

And now for the announced warm-up theorem:

**Theorem 83.** If a category \(\mathcal{C}\) has binary products and equalizers, then it has pullbacks.

**Proof.** Suppose we start with an arbitrary corner \(X \xrightarrow{f} Z \xleftarrow{g} Y\). Because \(\mathcal{C}\) has binary products, there will in particular be a product \(X \times Y\) with projections \(\pi_1: X \times Y \to X\) and \(\pi_1: X \times Y \to Y\). This gives us parallel arrows \(X \times Y \xrightarrow{\pi_1 \circ e} Z\). Because \(\mathcal{C}\) has equalizers, that parallel pair must have an equalizer \([E,e]\). So we get the square in the following commutative diagram:

\[
\begin{array}{ccc}
  & & E \\
  & & \downarrow{e} \\
 X & \xrightarrow{\pi_1} & X \times Y \\
  & \nearrow{\pi_2} & \\
  & & Y \\
\end{array}
\]

Claim: the wedge formed by \(E\) with the projections \(\pi_1 \circ e, \pi_2 \circ e\) is a pullback of the original corner.

Consider any other cone over the original corner, i.e. any wedge \(X \xleftarrow{c_1} C \xrightarrow{c_2} Y\) with \(f c_1 = g c_2\). We need to show that this factors uniquely through \(E\).

Now, that wedge certainly uniquely factors through the product \(X \times Y\), so there is a unique \(u: C \to X \times Y\) such that \(c_1 = \pi_1 \circ u, c_2 = \pi_2 \circ u\). Hence \(f \circ \pi_1 \circ u = g \circ \pi_2 \circ u\).

Therefore \(C \xrightarrow{u} X \times Y \xrightarrow{f \circ \pi_1, g \circ \pi_2} Z\) is a fork, which must factor uniquely through the equalizer \(E\) via some \(v\).

That is to say, there is a \(v: C \to E\) such that \(e \circ v = u\). Hence \(\pi_1 \circ e \circ v = \pi_1 \circ u = c_1\). In other words, completing the diagram with \(u: C \to E\) makes the left triangle commute. Symmetrically, it makes the top triangle commute.

Finally, we need to prove the uniqueness of such a diagram-completing arrow. Suppose \(v': C \to E\) also makes \(\pi_1 \circ e \circ v' = c_1, \pi_2 \circ e \circ v' = c_2\). But \(e \circ v': C \to X \times Y\), and we know the wedge \(X \xleftarrow{c_1} C \xrightarrow{c_2} Y\) factors uniquely through \(X \times Y\), so \(e \circ v' = u = e \circ v\). But equalizers are monic, so \(v' = v\). \(\square\)
14.2 Set has all finite/small limits

(a) We fix some standard terminology:

**Definition 79.** Let \( \mathcal{C} \) be a category, and \( \mathbf{J} \) be a small category. Then:

1. \( \mathcal{C} \) has limits of shape \( \mathbf{J} \) if for all functors \( D: \mathbf{J} \to \mathcal{C} \) there exists a limit cone over \( D \) in \( \mathcal{C} \).
2. \( \mathcal{C} \) has all finite limits if for any finite category \( \mathbf{J} \), \( \mathcal{C} \) has limits of shape \( \mathbf{J} \). A category with all finite limits is also said to be finitely complete.
3. \( \mathcal{C} \) has all small limits if for any small category \( \mathbf{J} \), \( \mathcal{C} \) has limits of shape \( \mathbf{J} \). A category with all limits is also said to be complete.

(I guess that talk about having ‘limits of shape \( \mathbf{J} \)’, though entirely conventional, is rather misplaced. After all, the limits won’t look like \( \mathbf{J} \), rather it is the diagrams that the limits are over which will look like \( \mathbf{J} \).)

It terms of this jargon, and given what we’ve seen before, a category’s having terminal objects is a matter of having limits of shape \( \mathbf{0} \) (where that’s the empty category), having binary products is a matter of having limits of shape \( \mathbf{T} \) (where that is ‘the’ two-element discrete category), having equalizers is a matter of having limits of shape \( \mathcal{C} \bullet \quad \ast \quad \emptyset \), and having pullbacks is having limits of shape \( \bullet \longrightarrow \ast \leftarrow \circ \). (omitting the identity arrows).

(b) Now revisit the proof of the last theorem where we showed that having binary products and equalizers entails having pullbacks.

We first constructed the product of objects from a corner. But that didn’t give us the vertex of the limit cone over the corner. We got to that by taking an equalizer, which supplied us with the limit object and a monic arrow into the product we took. Now in \( \text{Set} \), monics are injective and – up to isomorphism – a set-with-an-injection-into-a-product can be thought of as a subset-of-that-product. That gives us the clue to how to prove the following in a rather brute-force way:

**Theorem 84.** Set has all finite limits.

**Proof.** Take a finite category \( \mathbf{J} \), and a diagram-as-functor \( D: \mathbf{J} \to \text{Set} \).

First construct the product \( [P, p_{\mathbf{J}}] \) of all the objects \( D(J) \) for \( J \in \mathbf{J} \) (which of course we can do in \( \text{Set} \)). But that’s not (in general) going to give us the required vertex \( L \) of a limit cone over \( D \). We need again to take a subset of \( P \) – i.e. take some subset of the tuples \( \vec{x} = \langle x_I \rangle_{I \in \mathbf{J}} \), where each tuple-component \( x_J \) is of course a member of the corresponding \( D(J) \). But which subset of \( P \) do we want?

Whenever there is an arrow \( f: J \to K \) in \( \mathbf{J} \) there will be a corresponding arrow \( D(f): D(J) \to D(K) \) in \( \text{Set} \). Hence to get a cone with vertex \( L \) (let alone a limit cone) we need \( L \) to be equipped with projections \( \pi_J: L \to D(J) \), \( \pi_K: L \to D(K) \) that commute with \( D(f): D(J) \to D(K) \) – i.e. we require \( \pi_K = D(f) \circ \pi_J \).

Now, there’s a ‘natural’ projection from \( L \) to \( D_J \) for each \( J \). Just have \( \pi_J \) agree with \( p_J \) in picking the \( J \)-th component of a tuple \( \vec{x} \) (a component which is already a member of \( D_J \)) and send it to itself. But then, to now get \( \pi_K = D(f) \circ \pi_J \), for any tuple \( \vec{x} \) in \( L \), we require that \( D(f)(x_J) = x_K \).

So all that tells us one way of getting a cone \( [L, \pi_J] \) over \( D: \mathbf{J} \to \text{Set} \):

1. Put \( L \) to be the set of tuples \( \vec{x} = \langle x_I \rangle_{I \in \mathbf{J}} \) such that \( x_J \in D(J) \) for all \( J \in \mathbf{J} \) and where \( D(f)(x_J) = x_K \) for all \( f: J \to K \) in \( \mathbf{J} \).
(2) Put $\pi_J(\bar{x}) = x_J$.

And since we’ve constructed this cone in the most natural and economical way possible, it morally ought to be a limit cone.

Which it is! Suppose $[C, c_J]$ is any cone over $D$. Define the map $u : C \rightarrow L$ as sending $c \in C$ to the tuple $(c_J(c))_{J \in J}$. Then evidently $[C, c_J]$ factors through $[L, \pi_J]$ via $u$. And $u$ is unique. For if $[C, c_J]$ also factors through $[L, \pi_J]$ via $u'$, then $\pi_J \circ u' = c_J$, therefore for any $c$ the $J$-th component of $u'(c) = c_J(c)$, hence $u' = u$.

Since everything in that proof is finite it also establishes that $\text{FinSet}$ also has all finite limits. But in $\text{Set}$ we can of course form Cartesian products for set-indexed collection of sets of arbitrary size. Hence, so long as $J$ remains set-sized exactly the same proof will establish the stronger result that, unlike with $\text{FinSet}$,

**Theorem 85.** $\text{Set}$ has all small limits.

### 14.3 The existence of limits, more generally

(a) The proof in the last section almost wrote itself once we had the idea of taking a product $[P, \pi_J]$ of the objects $D(J)$, for $J \in J$ and extracting a subset containing just enough tuples. But it talked about subsets etc. in an old-school way. Now let’s see if we can rewrite the proof, still working for the moment in $\text{Set}$ but putting things in more purely categorial terms.

So, instead of talking of a subset of $P$ we’ll need to talk of a monic arrow into $P$. And – as in the proof of Theorem 83 – we know that a way of getting such a monic into the story as an equalizer of a couple of parallel arrows from $P$. But what parallel arrows to use?

Well, the only equalities we have hanging around in the proof of our last theorem arise when we take any arrow $f : J \rightarrow K$ in $J$ and then require $D(f)(x_J) = x_K$. So let’s work with these equalities.

So first take any particular arrow $f : J \rightarrow K$; then there are parallel arrows in $\text{Set}$

$$
\begin{array}{ccc}
P & \overset{p_f}{\longrightarrow} & D(K) \\
\downarrow^{\pi_J} & & \downarrow^{\rho_K} \\
D(J) & \overset{D(f)}{\longrightarrow} & D(K)
\end{array}
$$

where $p_f$ sends a product tuple $\bar{x}$ to $D(f)(x_J)$ and $q_f$ sends $\bar{x}$ to $x_K$.

But we want to wrap all those separate facts, one for each $f$, into one big fact. So take the product $[Q, \rho_K]$ of all the $D(K)$, i.e. the product of all codomains of arrows $D(f)$ for $f$ in $J$, which we can do in $\text{Set}$ given $J$ is small. Then there will be a unique $p$ such that each $p_f$ factors through $Q$ via $p$, and a unique $q$ such that each $q_f$ factors through $Q$ via $q$. So now we get parallel arrows

$$
\begin{array}{ccc}
P & \overset{p}{\longrightarrow} & Q \\
\downarrow^{\pi_J} & & \downarrow^{\rho_K} \\
D(J) & \overset{D(f)}{\longrightarrow} & D(K)
\end{array}
$$

where $p, q$ make these diagram commute for all $f : J \rightarrow K$ in $J$:

$$
\begin{array}{ccc}
P & \overset{p}{\longrightarrow} & Q \\
\downarrow^{\pi_J} & & \downarrow^{\rho_K} \\
D(J) & \overset{D(f)}{\longrightarrow} & D(K)
\end{array}
\quad \quad \quad
\begin{array}{ccc}
P & \overset{q}{\longrightarrow} & Q \\
\downarrow^{\pi_K} & & \downarrow^{\rho_K} \\
D(K) & \overset{D(f)}{\longrightarrow} & D(K)
\end{array}
$$

Since equalizers always exist in $\text{Set}$, we can take the equalizer $[E, e]$ of $P \overset{p}{\longrightarrow} Q$. Claim: $[E, \pi_J \circ e]$ form the desired limit cone over $D$. 
CHAPTER 14. THE EXISTENCE OF LIMITS

(1) \([E, \pi_J \circ e]\) is a cone. For suppose we have an arrow \(D(f) : D(J) \to D(K)\). Then we require \(D(f) \circ \pi_J \circ e = \pi_K \circ e\).

But indeed \(D(f) \circ \pi_J \circ e = \rho_K \circ p \circ e = \rho_K \circ q \circ e = \pi_K \circ e\), where the inner equation holds because \(e\) is an equalizer.

(2) Suppose \([C, c_J]\) is any other cone over \(D\). Then there must be a unique \(u : C \to P\) such that every \(c_J\) factors through the product and we have \(c_J = \pi_J \circ u\).

Since \([C, c_J]\) is a cone, for any \(f : J \to K\) in \(J\), \(D(f) \circ c_J = c_K\). Hence \(D(f) \circ \pi_J \circ u = \pi_K \circ u\), and hence (from the last diagrams) for each \(\rho_K, \rho_K \circ p \circ u = \rho_K \circ q \circ u\).

But then we can apply a generalized version of Theorem 66, and conclude that \(p \circ u = q \circ u\). Which means that

\[
\begin{array}{ccc}
C & \xrightarrow{u} & P \\
\downarrow & & \downarrow \rho \\
Q & \xleftarrow{q} & Q
\end{array}
\]

is a fork, which must therefore uniquely factor through the equalizer \([E, e]\). That is to say, there is a unique \(v : C \to E\) such that \(u = e \circ v\), and hence for all \(J\), \(c_J = \pi_J \circ u = \pi_J \circ e \circ v\). That is to say, \([C, c_J]\) factors uniquely through \([E, \pi_J \circ e]\) via \(v\). So \([E, \pi_J \circ e]\) is indeed a limit cone.

(b) That gives us a categorial proof, without mentioning subsets, of our result that \(\text{Set}\) has all (finite) limits. But now note that we didn’t depend on anything but the fact that \(\text{Set}\) has (finite) products and equalizers. Hence, generalizing, we have the originally promised sweeping result:

**Theorem 86.** If \(\mathcal{C}\) has terminal objects, binary products and equalizers, then it has all finite limits, i.e. is finitely complete.

**Proof.** Since \(\mathcal{C}\) has terminal objects and binary products it has all finite products, by Theorem 71.

So given a functor \(D : J \to \mathcal{C}\) with \(J\) finite, \(\mathcal{C}\) will have the products \([P, \pi_J]\) of all \(D(J)\) for \(J \in J\), and \([Q, \rho_K]\) of all \(D(K)\) such there is an arrow \(f : J \to K\) in \(J\).

The cone \([P, \rho_K]\) (with an arrow for each \(f : J \to K\)) must factor via a unique \(p\) through \(Q\). And the cone \([P, \pi_K]\) (with an arrow for each \(K\)) must there is an arrow \(f : J \to K\) must factor via a unique \(q\) through \(Q\). So that gives us \(p, q\) as parallel arrows from \(P\) to \(Q\).

Since \(\mathcal{C}\) has equalizers for all pairs of parallel arrows, we can take an equalizer \([E, e]\) for this pair. And then the argument goes exactly as before.

And similarly, if \(J\) is (not assumed finite) but small, then the argument will again go through so long as this time \(\mathcal{C}\) has products for all set-sized collections of objects, or as we say, has all small products. Hence

**Theorem 87.** If \(\mathcal{C}\) has all small products and has equalizers, then it has all small limits, i.e. is complete.

Given ingredients from our previous discussions, we can easily show that

**Theorem 88.** The categories of structured sets \(\text{Mon, Grp, Ab, Rng}\) (among others) are all complete.

\(\text{Top}\) too is complete. We have also met categories that are, by contrast, finitely complete but not complete like \(\text{Finset}\); while e.g. a poset-as-a-category may lack all many products and hence not be even finitely complete.
14.4 Dualizing again

Needless to say by this stage, our results in the last two sections dualize in obvious ways. Thus we need not delay over the proofs of

**Theorem 89.** If $\mathcal{C}$ has initial objects, binary coproducts and co-equalizers, then it has all finite colimits, i.e. is finitely cocomplete. If $\mathcal{C}$ has all coproducts and has co-equalizers, then it has all small colimits, i.e. is cocomplete.

**Theorem 90.** Set is cocomplete – as are the categories of structured sets Mon, Grp, Ab, Rng.

But note that a category can of course be (finitely) complete without being (finitely) cocomplete and vice versa. For a generic source of examples, take a poset $(P, \preceq)$ considered as a category. This automatically has all equalizers (and coequalizers) – see §12.9 Ex. (5). But it will have other limits (colimits) depending on which products (coproducts) exists, i.e. which sets of elements have supreme (inflame). For a simple case, take a poset with a maximum element and such that every pair of elements has a supremum: then considered as a category it has all finite limits (but maybe not infinite ones). But it need not have a minimal element and/or infima for all pairs of objects: hence it can lack some finite colimits despite having all finite limits.
A functor $F: \mathcal{C} \to \mathcal{D}$ must, just in virtue of its functoriality, preserve some aspects of the categorial structure of $\mathcal{C}$ as it sends objects and arrows into $\mathcal{D}$. The ‘some’ is essential, though. Recall, for example, that functors preserve sections and restrictions but don’t necessarily preserve monomorphisms and epimorphisms. So how do things stand with respect to preserving limits and colimits? And how do functors interact with limits more generally?

15.1 Preserving limits

Start with the natural definition: a functor preserves limits if it sends limits to limits and preserves colimits if it sends colimits to colimits. More carefully,

**Definition 80.** A functor $F: \mathcal{C} \to \mathcal{D}$ preserves limits of shape $\mathcal{J}$ iff, for any diagram $D: \mathcal{J} \to \mathcal{C}$, if $[C,c,J]$ is a limit cone over $D$, then $[FC,Fc,J]$ is a limit cone over $F \circ D: \mathcal{J} \to \mathcal{D}$.

Likewise $F$ preserves colimits of shape $\mathcal{J}$ iff, for any diagram $D: \mathcal{J} \to \mathcal{C}$, if $[C,c,J]$ is a limit cocone under $D$, then $[FC,Fc,J]$ is a limit cocone under $F \circ D: \mathcal{J} \to \mathcal{D}$.

A functor which preserves (co)limits of shape $\mathcal{J}$ for all finite categories $\mathcal{J}$ is said to preserve all finite (co)limits.

Suppose $\lim_{\leftarrow J} D$ is any limit object in $\mathcal{C}$ for a diagram $D$ of shape $\mathcal{J}$ and $\lim_{\leftarrow J}(F \circ D)$ is a limit object for a diagram $F \circ D$ of the same shape. Then if $F$ preserves limits of shape $\mathcal{J}$, then we get the crucial relation

$$F(\lim_{\leftarrow J} D) \cong \lim_{\leftarrow J}(F \circ D),$$

that is to say $F$ commutes with taking limits of shape $\mathcal{J}$. (Commuting with limits doesn’t, conversely, strictly entail preserving limits – for preservation, the isomorphism between $F(\lim_{\leftarrow J} D)$ and $\lim_{\leftarrow J}(F \circ D)$ has to be of the right kind – though often we don’t need to worry about that.)

Now recall that slogan from elementary analysis: ‘continuous functions commute with limits’. Which explains another bit of standard terminology:

**Definition 81.** A functor which preserves (co)limits of shape $\mathcal{J}$ for all small categories $\mathcal{J}$ is said to be (co)continuous.
Let’s have some simple examples. Our first two cases might seem a little artificial, but they will serve to illustrate an important point in an elementary way.

(1) Consider the functor $P: \mathbf{Set} \rightarrow \mathbf{Set}$ which sends the empty set $\emptyset$ to itself and sends every other set to the singleton 1, and acts on arrows in the only possible way if it is to be a functor (i.e. for $A \neq \emptyset$, it sends any arrow $\emptyset \rightarrow A$ to the unique arrow $\emptyset \rightarrow 1$, it sends the arrow $\emptyset \rightarrow \emptyset$ to itself and sends all other arrows to the identity arrow 1). Claim: $P$ preserves binary products but not equalizers.

For the first half of this claim, we simply consider cases. If neither $A$ nor $B$ is the empty set, then $A \times B$ is not empty either. $P$ then sends the limit wedge $A \leftarrow A \times B \rightarrow B$ to $1 \leftarrow 1 \rightarrow 1$, and it is obvious that any other wedge $1 \leftarrow L \rightarrow 1$ factors uniquely through the latter. So $P$ sends non-empty products to products.

If $A$ is the empty set and $B$ isn’t, $C$ is the empty set too. Then $P$ sends the limit wedge $A \leftarrow A \times B \rightarrow B$ to $\emptyset \leftarrow \emptyset \rightarrow 1$. Since the only arrows in $\mathbf{Set}$ with the empty set as target have the empty set as source, the only wedges $\emptyset \leftarrow L \rightarrow 1$ have $L = \emptyset$, so trivially factor uniquely through $\emptyset \leftarrow \emptyset \rightarrow 1$. So $P$ sends products of the empty set with non-empty sets to products.

Likewise for products of non-empty sets with the empty set, and the product of the empty set with itself. So, taking all the cases together, $P$ sends products to products.

Now consider the equalizer in $\mathbf{Set}$ of the two different maps $1 \xrightarrow{f} 2$, where 2 is a two-membered set. Since $f$ and $g$ never agree, their equalizer is the empty set (with the empty inclusion map). But since $P$ sends both the maps $f$ and $g$ to the identity map on 1, the equalizer of $Pf$ and $Pg$ is therefore the set 1 (with the identity map). Which means that the equalizer of $P(f)$ and $P(g)$ is not the result of applying $P$ to the equalizer of $f$ and $g$.

(2) Consider next the functor $Q: \mathbf{Set} \rightarrow \mathbf{Set}$ which sends any set $A$ to the set $\langle X, 2 \rangle$, and sends any arrow $f: X \rightarrow Y$ to $\langle f, 1_2 \rangle: \langle X, 2 \rangle \rightarrow \langle Y, 2 \rangle$. Claim: $Q$ preserves equalizers but not binary products

Concerning products, if a functor $F$ preserves binary products in $\mathbf{Set}$, then by definition $F\langle X, Y \rangle \cong \langle FX, FY \rangle$. However we have $Q\langle X, Y \rangle = \langle \langle X, Y \rangle, 2 \rangle \ncong \langle \langle X, 2 \rangle, \langle Y, 2 \rangle \rangle = \langle QX, QY \rangle$.

Now note that the equalizer of parallel arrows $X \xrightarrow{\langle f, g \rangle} Y$ is $E$, the subset of $X$ on which $f$ and $g$ take the same value (together with inclusion map from $E$ to $X$). And the equalizer of the parallel arrows $QX \xrightarrow{Qf} QY$ is the subset of $\langle X, 2 \rangle$ on which $\langle f, 1_2 \rangle$ and $\langle g, 1_2 \rangle$ take the same value, which will be $\langle E, 2 \rangle$, i.e. $QE$. So indeed $Q$ preserves equalizers.

What these first two toy examples illustrate is that a functor may preserve some limits but not others – it isn’t an all or nothing business.

However, consider now another example:

(3) The forgetful functor $F: \mathbf{Mon} \rightarrow \mathbf{Set}$ sends a terminal object in $\mathbf{Mon}$, a one-object monoid, to its underlying singleton set, which is terminal in $\mathbf{Set}$. So $F$ preserves limits of shape $\emptyset \bullet$.

The same functor sends a product $(M, \cdot) \times (N, \ast)$ in $\mathbf{Mon}$ to its underlying set of pairs of objects from $M$ and $N$, which is a product in $\mathbf{Set}$. So the forgetful $F$ also preserves limits of the shape of the discrete two object category $\emptyset \bullet \rightarrow \emptyset \ast$. 
Likewise for equalizers. As we saw in §12.9, Ex. (2), the equalizer of two parallel monoid homomorphisms \((M, \cdot) \xrightarrow{f} (N, \ast)\) is \((E, \cdot)\) equipped with the inclusion map \(E \hookrightarrow M\), where \(E\) is the set on which \(f\) and \(g\) agree. Which means that the forgetful functor takes the equalizer of \(f\) and \(g\) as monoid homomorphisms to their equalizer as set functions. So \(F\) preserves equalizers, i.e. preserves limits of shape \(\bullet \xrightarrow{\sim} \ast \xleftarrow{\sim} \bullet\).

Further, we have the following result:

**Theorem 91.** If \(\mathcal{C}\) is finitely complete, and a functor \(F: \mathcal{C} \to \mathcal{D}\) preserves terminal objects, binary products and equalizers, then \(F\) preserves all finite limits. If \(\mathcal{C}\) is complete and \(F\) preserves all small products and equalizers, then \(F\) preserves all (small) limits.

**Proof.** Suppose \(\mathcal{C}\) is finitely complete. Then any limit cone \([C, c_j]\) over a diagram \(D: J \to \mathcal{C}\) in \(\mathcal{C}\) is uniquely isomorphic to some limit cone \([C', c'_j]\) constructed from equalizers and finite products built in turn from terminal objects and binary products, by Theorem 86. Since \(F\) preserves terminal objects, binary products and equalizers, it sends the construction for \([C', c'_j]\) to a construction for a limit cone \([FC', FC'_j]\) over \(F \circ D: J \to \mathcal{D}\). But \(F\) preserves isomorphisms, so \([FC, FC_j]\) will be isomorphic to \([FC', FC'_j]\) and hence is also a limit cone over \(F \circ D: J \to \mathcal{D}\).

The proof for the case where \(\mathcal{C}\) is complete is exactly similar. \(\square\)

Therefore, to return to our last example,

(3) (Continued) Since \(\text{Mon}\) has all finite limits, the forgetful functor \(F: \text{Mon} \to \text{Set}\) preserves all finite limits. We can similarly show that the functor in fact preserves all limits.

(4) On the other hand, the same forgetful \(F\) does not preserve colimits with the ‘shape’ of the empty category, i.e. initial objects. For a one-object monoid is initial in \(\text{Mon}\) but its underlying singleton set is not initial in \(\text{Set}\).

\(F\) doesn’t preserve coproducts either – essentially because coproducts in \(\text{Mon}\) can be larger than coproducts in \(\text{Set}\). Cf. our discussion in §12.8 of coproducts in \(\text{Grp}\). Similarly, \(F(M \oplus N)\), the underlying set of a coproduct of monoids \(M\) and \(N\), is (isomorphic to) the set of finite sequences of alternating non-identity elements from \(M\) and \(N\). While \(FM \oplus FN\) is just the disjoint union of the underlying sets.

The example generalizes. A forgetful functor from a category of structured sets to \(\text{Set}\) typically preserves limits but not all colimits.

Finally, let’s remark on a simple result (which has an obvious dual which we leave unstated for once):

**Theorem 92.** If the functor \(F: \mathcal{C} \to \mathcal{D}\) preserves pullbacks it preserves monomorphisms.

**Proof.** By Theorem 81, if \(f: X \to Y\) in \(\mathcal{C}\) is monic then it is part of the pullback square on the left:

\[
\begin{array}{ccc}
X & \xrightarrow{1_X} & X \\
\downarrow{1_X} & & \downarrow{f} \\
X & \xrightarrow{f} & Y
\end{array}
\quad
\begin{array}{ccc}
FX & \xrightarrow{1_{FX}} & FX \\
\downarrow{1_{FX}} & & \downarrow{f} \\
FX & \xrightarrow{Ff} & FY
\end{array}
\]

By assumption \(F\) sends a pullback squares to pullback squares, so the square on the right is also a pullback square. So by Theorem 81 again, \(Ff\) is monic too. \(\square\)
15.2 Reflecting limits

Definition 82. A functor $F: \mathcal{C} \to \mathcal{D}$ reflects limits of shape $J$ iff, given a cone $[C, c_J]$ over a diagram $D: J \to \mathcal{C}$, then if $[FC, Fc_J]$ is a limit cone over $F \circ D: J \to \mathcal{D}$, $[C, c_J]$ is already a limit cone over $D$.

Reflecting colimits is defined dually.

Let’s again immediately turn to some examples. We can cook up more toy cases to show, e.g., that reflecting products needn’t go along with reflecting equalizers and vice versa (do it!). But we have:

1. The forgetful functor $F: \textbf{Mon} \to \textbf{Set}$ reflects all limits. Similarly for some other forgetful functors from familiar categories of structured sets to $\textbf{Set}$.

However, be careful! For we also have . . .

2. The forgetful functor $F: \textbf{Top} \to \textbf{Set}$ which sends topological space to its underlying set preserves all limits but does not reflect all limits. Here’s a case involving binary products. Suppose $X$ and $Y$ are a couple of spaces with a coarse topology, and let $Z$ be the space $FX \times FY$ equipped with a finer topology. Then, with the obvious arrows, $X \leftarrow Z \rightarrow Y$ is a wedge to $X$, $Y$ but not the limit wedge in $\textbf{Top}$: but $FX \leftarrow FX \times FY \rightarrow FY$ is a limit wedge in $\textbf{Set}$.

3. Neither of the forgetful functors $F: \textbf{Mon} \to \textbf{Set}$ or $F: \textbf{Top} \to \textbf{Set}$ preserves all finite colimits.

Here, though, a couple of theorems that tell more us about functors that do reflect limits:

Theorem 93. Suppose $F: \mathcal{C} \to \mathcal{D}$ preserves limits. Then if $\mathcal{C}$ is complete and $F$ reflects isomorphisms, then $F$ reflects limits.

Proof. Since $\mathcal{C}$ is complete there exists a limit cone $[B, b_J]$ over a diagram $D: J \to \mathcal{C}$, and so – since $F$ preserves limits – $[FB, Fb]$ is a limit cone over $F \circ D: J \to \mathcal{D}$.

Now suppose that there is a cone $[C, c_J]$ over a diagram $D$ such that $[FC, Fc_J]$ is another limit cone over $F \circ D$. Now $[C, c_J]$ must uniquely factor through $[B, b_J]$ via a map $f: C \to B$. Which means that $[FC, Fc_J]$ factors through $[FB, Fb]$ via $Ff$. However, since these are by hypothesis both limit cones over $F \circ D$, $Ff$ must be an isomorphism. Hence, since $F$ reflects isomorphisms, $f$ must be an isomorphism. So $[C, c_J]$ must be a limit cone by Theorem 78.

Theorem 94. Suppose $F: \mathcal{C} \to \mathcal{D}$ is full and faithful. Then $F$ reflects limits.

Proof. Suppose $[C, c_J]$ is a cone over a diagram $D: J \to \mathcal{C}$, and $[FC, Fc_J]$ is a limit cone over $F \circ D: J \to \mathcal{D}$.

Now take any other cone $[B, b_J]$ over $D$. Then $F$ sends this to a cone $[FB, Fb_J]$ which must uniquely factor through the limit cone $[FC, Fc_J]$ via some $u: FB \to FC$ which makes $Fb_J = Fc_J \circ u$ for each $J \in J$. Since $F$ is full and faithful, $u = Fv$ for some unique $v: B \to C$ such that $b_J = Fc_J \circ u$ for each $J$. So $[FB, Fb_J]$ factors uniquely through $[C, c_J]$. Which confirms that $[C, c_J]$ is a limit cone.
15.3 Creating limits

Alongside the rather natural notions of preserving and reflecting limits, we meet a third notion:

**Definition 83.** A functor \( F : \mathcal{C} \to \mathcal{D} \) creates limits of shape \( \mathcal{J} \) iff, for any diagram \( D : \mathcal{J} \to \mathcal{C} \), if \([M, m, j]\) is a limit cone over \( F \circ D \), there is a unique cone \([C, c, j]\) over \( D \) such that \([FC, Fc, j]\) = \([M, M, m, j]\), and moreover \([C, c, j]\) is a limit cone. Creating colimits is defined dually.

[Note: some define creation of limits by only requiring that \([FC, Fc, j]\) is isomorphic to \([M, m, j]\) in the obvious sense.]

Roughly speaking: reflection is a condition on those limit cones over \( F \circ D \) which take the form \([FC, Fc, j]\), creation is a similar condition on any limit cone over \( F \circ D \). So as you would predict,

**Theorem 95.** If the functor \( F : \mathcal{C} \to \mathcal{D} \) creates limits of shape \( \mathcal{J} \), it reflects them.

**Proof.** Suppose \([FC, Fc, j]\) is a limit cone over \( F \circ D : \mathcal{J} \to \mathcal{C} \). Then for some unique cone \([C', c', j']\), \([FC', Fc', j']\) = \([FC, Fc, j]\), and also \([C', c', j']\) is a limit cone. By the uniqueness requirement, \([C', c', j'] = [C, c, j] \) (since the latter certainly fits the condition!). Hence \([C, c, j]\) is a limit cone.

**Theorem 96.** Suppose \( F : \mathcal{C} \to \mathcal{D} \) is a functor, that \( \mathcal{D} \) has limits of shape \( \mathcal{J} \) and \( F \) creates such limits. Then \( \mathcal{C} \) has limits of shape \( \mathcal{J} \) and \( F \) preserves them.

**Proof.** Take any diagram \( D : \mathcal{J} \to \mathcal{C} \). Then there is a limit \([M, m, j]\) over \( F \circ D \) (since \( \mathcal{D} \) has all limits). Hence (since \( F \) creates limits), there is a limit cone \([C, c, j]\) over \( D \) where this is such that \([FC, Fc, j]\) is \([M, m, j]\) and hence is a limit cone too.

Take again the example of the forgetful functor \( F : \text{Mon} \to \text{Set} \). \( \text{Set} \) has all limits. And \( F \) creates all limits (we won’t here pause to check that!). So it follows again from the last theorem that, as we already know, \( F \) preserves all limits.

15.4 Hom-functors preserve limits

For the rest of this chapter \( \mathcal{C} \) is a locally small category, and as usual \( \mathcal{J} \) is small. The next main theorem locates one rich source of continuous, i.e. limit-preserving, functors:

**Theorem 97.** The covariant hom-functor \( \mathcal{C}(A, -) : \mathcal{C} \to \text{Set} \), for any \( A \) in the \( \mathcal{C} \), preserves all limits that exist in \( \mathcal{C} \).

And let’s go first for a straight just-apply-the-definitions-and-see-what-happens demonstration (it’s a useful reality check to run through the details):

**Proof.** We’ll first check that \( \mathcal{C}(A, -) : \mathcal{C} \to \text{Set} \) sends a cone over the diagram \( D : \mathcal{J} \to \mathcal{C} \) to a cone over \( \mathcal{C}(A, -) \circ D : \mathcal{J} \to \text{Set} \).

A cone has a vertex \( C \), and arrows \( C_J : C \to DJ \) for each \( J \in \mathcal{J} \), where for any \( f : J \to K \) in \( \mathcal{J} \), so for any \( DF : DJ \to DK \), \( c_K = DF \circ c_J \).

Now, acting on objects, \( \mathcal{C}(A, -) \) sends \( C \) to \( \mathcal{C}(A, C) \) and sends \( DJ \) to \( \mathcal{C}(A, DJ) \). And acting on arrows, \( \mathcal{C}(A, -) \) sends \( C_J : C \to DJ \) to the set function \( c_J \circ - \) which takes \( g : A \to C \) and outputs \( c_J \circ g : A \to DJ \); and it sends \( DF : DJ \to DK \) to the set-function \( DF \circ - \) which takes \( h : A \to DJ \) and outputs \( DF \circ h : A \to DK \).

Diagrammatically, then, the functor sends the triangle on the left to the one on the right:
And assuming $c_K = Df \circ c_J$, we have $c_K \circ - = (Df \circ c_J) \circ - = Df \circ (c_J \circ -)$: hence, if the triangle on the left commutes, so does the triangle on the right. Likewise for other such triangles. Which means that if $[C, c_J]$ is a cone over $D$, then $[\mathcal{C}(A, C), c_J \circ -]$ is indeed a cone over $\mathcal{C}(A, -) \circ D$.

So far, so good! It remains, then, to show that in particular $\mathcal{C}(A, -)$ sends limit cones to limit cones.

Suppose then that $[L, \pi_J]$ is a limit cone in $\mathcal{C}$ over $D$. The functor $\mathcal{C}(A, -) : \mathcal{C} \to \textbf{Set}$ sends the commuting triangle on the left in the next diagram to the commuting triangle at the bottom of the right triangle in the next diagram. And we now suppose that $[M, m_J]$ is any other cone over the image of $D$:

Hence $m_K = (Df \circ -) \circ m_J$. Now remember that $M$ lives in $\textbf{Set}$: so take a member $x$. Then $m_J(x)$ is a particular arrow in $(A, DJ)$, so we have $m_J(x) : A \to DJ$. Likewise we have $m_K(x) : A \to DK$. But $m_K(x) = Df \circ m_J(x)$. Which means that for all $f$ the outer triangles on the left below commute and so $[A, m_J(x)]$ is a cone over $D$. And this must factor uniquely through an arrow $u(x)$ as follows:

So $u(x)$ is an arrow from $A$ to $L$, i.e. an element of $\mathcal{C}(A, L)$. So consider the map $u : M \to \mathcal{C}(A, L)$ which sends $x$ to $u(x)$. Since $m_J(x) = \pi_J \circ u(x)$ for each $x$, $m_J = (\pi_J \circ -) \circ u$. Since this applies for each $J$, So $[M, m_J]$ factors through the image of the cone $[L, \pi_J]$ via $u$.

Suppose there is another map $v : M \to \mathcal{C}(A, L)$ such that we also have each $m_J = (\pi_J \circ -) \circ v$. Then again take an element $x \in M$: then $m_J(x) = \pi_J \circ v(x)$. So again, $[A, m_J(x)]$ factorizes through $[L, \pi_J]$ via $v(x) −$ which, by the uniqueness of factorization through limits, means that $v(x) = u(x)$. Since that obtains for all $x \in M$, $v = u$. Hence $[M, m_J]$ factors uniquely through the image of $[L, \pi_J]$. Since $[M, m_J]$ was an arbitrary cone, we have therefore proved that the image of the limit cone $[L, \pi_J]$ is also a limit cone. \qed
15.5 More on hom-functors and limits

Our last proof, as we announced, involved working straight from definitions. We could get to the same result more snappily by remarking that if there is a limit cone over \( D: J \to \mathcal{C} \), then it can be constructed from suitable products and equalizers (as indicated by the proof of Theorem 87). So to show hom-functors preserve those limits which exists it is enough to show that they preserve products and equalizers which exist. See e.g. Awodey (2006, p. 106) for this line of proof.

But we won’t pause over this, but rather consider another way of using the proof of the existence of limits in \( \mathbf{Set} \) to get at least the implication that hom-functors commute with limits. Note, then, two preliminary results which flow easily from previous theorems:

**Theorem 98.** Suppose \( D: J \to \mathcal{C} \) is a diagram and \( A \) an object in \( \mathcal{C} \). Then

\[
\text{Cone}(A, D) \cong \lim_{\leftarrow J} (\mathcal{C}(A, -) \circ D).
\]

*Proof.* We have already shown that \( \mathbf{Set} \) has all limits, and so in particular it will have a limit for the diagram \( \mathcal{C}(A, -) \circ D: J \to \mathbf{Set} \).

And we know from the proof of Theorem 84 what limits for a functor \( F: J \to \mathbf{Set} \) look like. The vertex of the limit cone is the set of tuples \( \langle x_J \rangle_{J \in J} \) such that \( x_J \in FJ \) for all \( J \in J \) and where \((Ff)x_J = x_K\) for all \( f: J \to K \) in \( J \).

So the vertex of the limit cone for \( \mathcal{C}(A, -) \circ D \) is the set of tuples \( \langle x_J \rangle_{J \in J} \) such that \( x_J \in \mathcal{C}(A, DJ) \) and \( Df \circ x_J = x_K \). But those last conditions hold just when the \( x_J \) are arrows forming a cone with vertex \( A \) over \( D \). So each tuple is in a trivial (and obviously natural) isomorphism with a cone with vertex \( A \) over \( D \).

Hence \( \lim_{\leftarrow J} (\mathcal{C}(A, -) \circ D) \), the set of such tuples, is isomorphic (indeed naturally isomorphic) to the set of cones \( \text{Cone}(A, D) \).

**Theorem 99.** Suppose \( D: J \to \mathcal{C} \) is a diagram and \( A \) an object in \( \mathcal{C} \). Then if \( D \) has a limit,

\[
\text{Cone}(A, D) \cong \mathcal{C}(A, \lim_{\leftarrow J} D).
\]

*Proof.* Let \( [\lim_{\leftarrow J} D, \pi_J] \) be a limit cone over \( D \). Then there is a one-one correspondence between arrows \( f \in \mathcal{C}(A, \lim_{\leftarrow J} D) \), i.e. arrows \( f: A \to \lim_{\leftarrow J} D \) and cones \( [A, \pi_J \circ f] \in \text{Cone}(A, D) \). For if \( f \neq f' \), the corresponding cones \( [A, \pi_J \circ f] \) and \( [A, \pi_J \circ f'] \) must be distinct, otherwise there would be a single cone with vertex \( A \) which factors through the limit cone in two different ways. And every cone over \( D \) with vertex \( A \) is of that form for a unique \( f \), again since it factors uniquely through the limit. \( \square \)

Hence we have the following result:

**Theorem 100.** Suppose \( D: J \to \mathcal{C} \) is a diagram and \( A \) an object in \( \mathcal{C} \). Then if \( D \) has a limit,

\[
\mathcal{C}(A, \lim_{\leftarrow J} D) \cong \lim_{\leftarrow J} (\mathcal{C}(A, -) \circ D).
\]

Put \( F = \mathcal{C}(A, -) \), and we have \( F(\lim D) \cong \lim_{\leftarrow J} (F \circ D) \) – so the hom-functor \( F \) indeed commutes with limits. Indeed, the isomorphism is a natural one – an observation which takes us back to the claim that the covariant hom-functor \( F \) preserves the limit over \( D \), but we’ll hang fire for the moment in saying more about this.
16

Galois connections

We will return to say more about functors and limits after we have tackled the next Big Idea from category theory that we need to get our heads around – namely, the notion of adjoints and adjunction.

Rather than dive straight in to the general story, however, we are going to look first at a restricted class of cases – i.e. we are going to be saying something in this chapter about so-called Galois connections, which are in effect adjunctions between two categories which are posets. These cases are of considerable stand-alone interest; but they will also usefully introduce us to some key themes in a relatively uncluttered context. In the next chapter, we will pause to see the idea of Galois connection at work in a famous special case of interest to logicians. The general story about adjoints starts in Chapter 18.

16.1 (Probably unnecessary) reminders about posets

Recall: The set $P$ equipped with the binary relation $\leq$, which we denote $(P, \leq)$, is a poset just in case $\leq$ is a partial order – i.e., for all $x, y, z \in S$, (i) $x \leq x$, (ii) if $x \leq y$ and $y \leq z$ then $x \leq z$, (iii) if $x \leq y$ and $y \leq x$ then $x = y$. (We will, as appropriate, recruit ‘$\leq$', ‘$\subseteq$', ‘$\subseteq$’ as other symbols for partial orders.)

Reversing a partial order gives us another partial order. Hence reversing the order in a poset $\mathcal{P} = (P, \leq)$ gives us a dual poset $\mathcal{P}^{op} = (P, \geq)$ defined in the obvious way.

There is the related notion of a strict poset defined in terms of the notion of a strict partial order $\prec$, where $x \prec y$ iff $x \leq y \land x \neq y$ (and so $x \leq y$ iff $x \prec y \lor x = y$). It is just a matter of convenience whether we concentrate on the one flavour of poset or the other. And we can take it that you are already familiar with a variety of examples of ‘naturally occurring’ posets of both flavours.

The following notions will also be familiar, in one terminology or another:

**Definition 84.** Suppose that $(P, \preceq)$ and $(Q, \sqsubseteq)$ are two posets. Let $f : P \to Q$ be a map between their carrier sets. Then

1. $f$ is monotone just in case, for all $x, y \in P$, if $x \preceq y$ then $f(x) \sqsubseteq f(y)$;
2. $f$ is an order-embedding just in case, for all $x, y \in P$, $x \preceq y$ iff $f(x) \sqsubseteq f(y)$;
3. $f$ is an order-isomorphism iff $f$ is a surjective order-embedding.

Some obvious remarks:

i. Monotone maps compose to give monotone maps and composition is associative.
Likewise for order-embeddings and order-isomorphisms.
ii. As you’d expect from the label, order-embeddings are injective. Keeping the same notation, suppose \( f(x) = f(y) \) and hence both \( f(x) \subseteq f(y) \) and \( f(y) \subseteq f(x) \). Then, if \( f \) is an embedding, \( x \preceq y \) and \( y \preceq x \), and hence \( x = y \).

iii. If \( f[P] \) is \( P \)'s image under \( f \), an order-embedding \( f \) is an order-isomorphism from \((P, \preceq)\) to \((f[P], \subseteq)\).

iv. An order-isomorphism is bijective, and therefore is an isomorphism as a set-function.

Order-isomorphisms have unique inverses which are also order-isomorphisms.

v. Posets are deemed isomorphic if there is an order-isomorphism between them.

If \((P, \preceq)\) is a poset and \( X \subseteq P \), then a maximum of \( X \) is defined in the entirely obvious way. Maxima are unique – for if \( m, m' \in X \) are both maxima, \( m' \preceq m \) (since \( m \) is a maximum) and similarly \( m \preceq m' \) and hence \( m = m' \). And if \( X \subseteq P \) we say that \((X, \preceq)\) is a sub-poset of \((P, \preceq)\) – we will not routinely fuss to distinguish a relation defined over \( P \) from the restriction of that relation to \( X \).

**Definition 85.** Suppose \( S \) is a set and \( \Pi \subseteq \mathcal{P}(S) \) is some set of subsets of \( S \). Then \( \Pi \) ordered by inclusion, i.e. \((\Pi, \subseteq)\), is an inclusion poset.

**Theorem 101.** Every poset is isomorphic to an inclusion poset.

**Proof.** Take the poset \((P, \preceq)\). For each \( p \in P \), now form the set containing it and its \( \preceq \)-predecessors \( \pi_p = \{ x \in P \mid x \preceq p \} \). Let \( \Pi \) the set of all \( \pi_p \) for \( p \in P \). Then \((\Pi, \subseteq)\) is an inclusion poset.

Define \( f : P \to \Pi \) by putting \( f(p) = \pi_p \). Then \( f \) is very easily seen to be a bijection, and also \( p \preceq q \) iff \( \pi_p \preceq \pi_q \). So \( f \) is an order-isomorphism. \( \square \)

### 16.2 Galois connections defined

We now define a relation between posets which is weaker than isomorphism but strong enough to have a wealth of interesting implications.

**Definition 86.** Suppose that \( \mathcal{P} = (P, \preceq) \) and \( \mathcal{Q} = (Q, \subseteq) \) are two posets, and let \( f_* : P \to Q \) and \( f^* : Q \to P \) be a pair of functions such that for all \( p \in P \), \( q \in Q \),

\[
(G) \quad f_*(p) \subseteq q \iff p \preceq f^*(q).
\]

Then \((f_*, f^*)\) form a Galois connection between \( \mathcal{P} \) and \( \mathcal{Q} \). And if \((f_*, f^*)\) form such a connection, \( f_* \) is said to be the left adjoint of \( f^* \), and \( f^* \) the right adjoint of \( f_* \).

To remember which way round the stars go in our convention, think that the left adjoint \( f_* \) with the lower star appears on the left or ‘lower’ side of the ordering sign in \((G)\), and \( f^* \) appears on the upper side. (Though just to be annoying, at least as many authors use \( f_* \) and \( f^* \) the other way about.) Talk of adjoints here seems to have been originally borrowed from the old theory of Hermitian operators, where in e.g. a Hilbert space with inner product \( \langle \cdot, \cdot \rangle \) the operators \( A \) and \( A^* \) are said to be adjoint when we have, generally, \( \langle Ax, y \rangle = \langle x, A^*y \rangle \). The formal analogy is evident.

The first discussion of such a connection \((f_*, f^*)\) – and hence the name – is to be found in Evariste Galois’s work in what has come to be known as Galois theory, a topic beyond our purview here. And there are plenty of serious mathematical examples (e.g. from number theory, abstract algebra and topology) of two posets with a Galois connection between them. But we really don’t want to get bogged down in unnecessary mathematics at this early stage; so for the moment let’s just give some very simple cases, including a couple of logical ones:
(1) Suppose \( f \) is an order isomorphism between \((P, \preceq)\) and \((Q, \subseteq)\): then \( f^{-1} \) is an order-isomorphism in the reverse direction. Take \( p \in P, q \in Q \): then trivially \( f(p) \subseteq q \iff f^{-1}(f(p)) \subseteq f^{-1}(q) \) if \( p \subseteq f^{-1}(q) \), so \((f, f^{-1})\) form a Galois connection.

(2) Take \( \mathcal{N} = (\mathbb{N}, \leq) \) and \( \mathcal{Q} = (\mathbb{Q}^+, \leq) \), i.e. the naturals and the non-negative rationals in their standard orders. Let \( i : \mathbb{N} \rightarrow \mathbb{Q}^+ \) be the injection function which maps a natural number to the corresponding rational integer, and let \( f : \mathbb{Q}^+ \rightarrow \mathbb{N} \) be the ‘floor’ function which maps a rational to the natural corresponding to its integral part. Then \((i, f)\) is a Galois connection between the posets \( \mathcal{N} \) and \( \mathcal{Q} \).

(3) Let \( f : X \rightarrow Y \) be some function between two sets \( X \) and \( Y \). It induces a function \( F : \mathcal{P}(X) \rightarrow \mathcal{P}(Y) \) between their powersets which sends \( A \subseteq X \) to \( f[A] \), and another function \( F^{-1} : \mathcal{P}(Y) \rightarrow \mathcal{P}(X) \) which sends \( B \subseteq Y \) to its pre-image under \( f \), \( F^{-1}[B] = \{x \in X \mid f(x) \in B\} \). Then \((F, F^{-1})\) is a Galois connection between the inclusion posets \((\mathcal{P}(X), \subseteq)\) and \((\mathcal{P}(Y), \subseteq)\).

(4) Take any poset \((P, \leq)\), and let \((1, =)\) be a one-object poset with the only possible partial order relation. Let \( t : P \rightarrow 1 \) be the only possible function, the trivial one which sends every element of \( P \) to the single element 0 of 1. Suppose \( t^* : 1 \rightarrow P \) is its right adjoint. Then for any \( p \), \( t(p) = 0 \) if \( p \preceq t^*(0) \). So \( t \) has a right adjoint just in case \( P \) has a maximum, and then \( t^* \) sends 1’s only element to it. Dually, \( t \) has a left adjoint just in case \( P \) has a minimum, and then the left adjoint \( t_* \) sends 1’s only element to \( that \).

(5) Our next example is from elementary logic. Let \( \mathcal{L} \) be a formal language equipped with a proof-system (it could be classical or intuitionistic, or indeed any other logic with normally-behaved conjunction and conditional connectives and a sensible deducibility relation): \( \alpha \vdash \beta \) notates that there is an \( \mathcal{L} \)-proof from premiss \( \alpha \) to conclusion \( \beta \). Then let \( |\alpha| \) be the equivalence class of \( \mathcal{L} \)-wffs interderivable with \( \alpha \). Take \( E \) to be set of all such equivalence classes, and put \(|\alpha| \leq |\beta| \) if \( \alpha \vdash_L \beta \). Then it is easily checked that \((E, \leq)\) is a poset.

Now consider the following two functions between \((E, \leq)\) and itself. Fix \( \gamma \) to be some \( \mathcal{L} \)-wff. Then let \( f_\gamma \) send the equivalence class \(|\alpha|\) to the class \(|(\gamma \land \alpha)|\), and let \( f_\gamma^* \) send \(|\alpha|\) to the class \(|(\gamma \rightarrow \alpha)|\).

Given our normality assumption, \((\gamma \land \alpha) \vdash_L \beta\) if and only if \( \alpha \vdash_L (\gamma \rightarrow \beta) \). Hence \(|(\gamma \land \alpha)| \leq |\beta| \iff |\alpha| \leq |(\gamma \rightarrow \beta)|\). That is to say \( f_\gamma(|\alpha|) \leq |\beta| \iff |\alpha| \leq f_\gamma^*(|\beta|)\). Hence \((f_\gamma, f_\gamma^*)\) form a Galois connection between \((E, \leq)\) and itself. So, in a slogan, ‘Conjunction is left adjoint to conditionalization’.

(6) Our last example for the moment is again from elementary logic. Let \( \mathcal{L} \) now be a first-order language, and consider the set of \( \mathcal{L} \)-wffs with at most the variables \( \bar{x} \) free.

We will write \( \varphi(\bar{x}) \) for a formula in in this class, \(|\varphi(\bar{x})|\) for the class of formulae interderivable with \( \varphi(\bar{x}) \), and \( E_\bar{x} \) for the set of such equivalence classes of formulae with at most \( \bar{x} \) free. Using \( \leq \) as in the last example, \((E_\bar{x}, \leq)\) is a poset for any choice of variables \( \bar{x} \).

We now consider two maps between the posets \((E_\bar{x}, \leq)\) and \((E_{\bar{x},y}, \leq)\). In other words, we are going to be moving between (equivalence classes of) formulae with at most \( \bar{x} \) free, and (equivalence classes of) formulae with at most \( \bar{x}, y \) free – where \( y \) is a new variable not among the \( \bar{x} \).

First, since every wff with at most the variables \( \bar{x} \) free also has at most the variables \( \bar{x}, y \) free, there is a trivial map \( f_* : E_\bar{x} \rightarrow E_{\bar{x},y} \) that sends the class of formulas \(|\varphi(\bar{x})|\) in \( E_\bar{x} \) to the same class of formulas which is also in \( E_{\bar{x},y} \).
Second, we define the companion map \( f^*: E_{\vec{x},y} \rightarrow E_\vec{x} \) that sends \(|\varphi(\vec{x}, y)|\) in \( E_{\vec{x},y} \) to \(|\forall y \varphi(\vec{x}, y)|\) in \( E_\vec{x} \).

Then \((f_*, f^*)\) form a Galois connection. For that is just to say

\[
\varphi(\vec{x}) \vdash_\varphi \psi(\vec{x}, y) \iff |\varphi(\vec{x})| \leq f^*(|\psi(\vec{x}, y)|).
\]

Which just reflects the familiar logical rule that \( \varphi(\vec{x}) \vdash_\varphi \psi(\vec{x}, y) \) iff \( \varphi(\vec{x}) \vdash_\varphi \forall y \psi(\vec{x}, y) \), so long as \( y \) is not free in \( \varphi(\vec{x}) \). Hence, in another slogan, universal quantification is right-adjoint to a certain trivial operation of adding a dummy variable.

And we can exactly similarly show that existential quantification is left-adjoint to the same operation.

Our first example, then, shows that Galois connections exist and are at least as plentiful as order-isomorphisms: such an isomorphism has a right and left adjoint which are the same (i.e. are the inverse). The second and fourth cases show that posets that aren’t order-isomorphic can in fact still be Galois connected. The third case shows that posets can have many Galois connections between them (as any \( f: X \rightarrow Y \) generates a connection between the inclusion posets on the powersets of \( X \) and \( Y \)). The fourth example gives a case where a function has both a left and a right adjoint which are different. The fourth and sixth cases give a couple of illustrations of how a significant construction (taking maxima, forming a universal quantification respectively) can be regarded as adjoint to some quite trivial operation. The fifth example, like the third, shows that even when the Galois-connected posets are isomorphic (in the fifth case trivially so, because they are identical!), there can be a pair of functions which aren’t isomorphisms but which also go to make up a connection between the posets. And the fifth and sixth examples already show that Galois connections are of interest to logicians.

### 16.3 Galois connections re-defined

Here is a common alternative definition for a Galois connection:

**Definition 87** (Alternative). Suppose that \( \mathcal{P} = (P, \preceq) \) and \( \mathcal{Q} = (Q, \sqsubseteq) \) are posets, and let \( f_*: P \rightarrow Q \) and \( f^*: Q \rightarrow P \) be a pair of functions such that for all \( p \in P, q \in Q, \)

1. \( f_* \) and \( f^* \) are both monotone,
2. for all \( p \in P, q \in Q, p \preceq f^*(f_*(p)) \) and \( f_*(f^*(q)) \subseteq q. \)

Then \((f_*, f^*)\) is a Galois connection between \( \mathcal{P} \) and \( \mathcal{Q}. \)

This alternative definition coincides with our original one, for we have the following simple theorem:

**Theorem 102.** \((f_*, f^*)\) is a Galois connection by our original definition if and only if it is a Galois connection by the alternative definition.

**Proof.** (If) Assume conditions (1) and (2) of the alternative definition both hold. And suppose \( f_*(p) \subseteq q \). Since by (1) \( f^* \) is monotone, \( f^*(f_*(p)) \preceq f^*(q) \). But by (2) \( p \preceq f^*(f_*(p)) \). So, since \( \preceq \) is transitive, \( p \preceq f^*(q) \).

That establishes one half of the biconditional \((G)\) \( f_*(p) \subseteq q \) iff \( p \preceq f^*(q) \). The proof of the other half is dual.
(Only if) Suppose (G) is true. Then in particular, \( f_*(p) \subseteq f_*(q) \iff p \preceq f^*(f_*(p)) \).
Since \( \subseteq \) is reflexive, \( p \preceq f^*(f_*(p)) \). Similarly for the other half of (2).

Now, suppose also that \( p \preceq p' \). Then since we’ve just shown \( p' \preceq f^*(f_*(p')) \), we have \( p \preceq f^*(f_*(p')) \). But by (G) we have \( f_*(p) \subseteq f_*(p') \iff p \preceq f^*(f_*(p')) \). Whence, \( f_*(p) \subseteq f_*(p') \) and \( f_* \) is monotone. Similarly for the other half of (1).

In the light of condition (1) of our equivalent alternative, Galois connections as we have now twice defined them are sometimes called **monotone** connections. Note now that if \( (f_*, f^*) \) is a (monotone) connection between \( \mathcal{P} = (P, \leq) \) and \( \mathcal{Q} = (Q, \subseteq) \), so that \( f_*(p) \subseteq q \iff p \preceq f^*(q) \) then there is an **antitone** connection between \( \mathcal{P} \) and the dual poset \( \mathcal{Q}^\text{op} = (Q, \subseteq) \), meaning a pair of functions \( (f_*, f^*) \) such that \( q \leq f_*(p) \iff p \preceq f^*(q) \) (since, trivially, \( x \leq y \iff y \leq x \)).

The only point of mentioning antitone connections here is to note that (a) you might sometimes encounter the idea of a connection defined the antitone way (which indeed is how it appears in Galois’s work), but (b) since an antitone connection between two posets is just a monotone connection between the first poset and the second’s dual, we just don’t need separate stories about the two types of connection, and without loss of generality we can concentrate entirely on connections presented the monotone way.

### 16.4 Some basic results about Galois connections

In this section, we prove that Galois connections behave as we would hope. We start by proving that such connections compose in an obvious way to produce new connections:

**Theorem 103.** Let \( (f_*, f^*) \) be a Galois connection between the posets \( \mathcal{P} = (P, \leq) \) and \( \mathcal{Q} = (Q, \subseteq) \), and let \( (g_*, g^*) \) be a Galois connection between the posets \( \mathcal{Q} \) and \( \mathcal{R} = (R, \subseteq) \). Then \( (g_* \circ f_*, f^* \circ g^*) \) is a Galois connection between \( \mathcal{P} \) and \( \mathcal{R} \).

**Proof.** By assumption, (G) for any \( p \in P, q \in Q \) we have \( f_*(p) \subseteq q \iff p \preceq f^*(q) \), and (G') for any \( q \in Q, r \in R \) we have \( g_*(q) \subseteq r \iff q \preceq g^*(r) \).

So take any \( p \in P, r \in R \). (G) implies that \( f_*(p) \subseteq g^*(r) \iff p \preceq f^*(g^*(r)) \) while (G') implies that \( g_*(f_*(p)) \subseteq r \iff f_*(p) \preceq g^*(r) \) – whence \( g_*(f_*(p)) \subseteq r \iff p \preceq f^*(g^*(r)) \), which is what we needed to show.

Next we show that the components of a Galois connection are indeed connected in the sense that if a Galois connection \( (f, \cdot) \) between two posets exists, then \( f \) uniquely fixes what its right adjoint is; and if a connection \( (\cdot, f) \) exists, \( f \) uniquely fixes its left adjoint.

**Theorem 104.** If \( (f_*, f^1) \) and \( (f_*, f^2) \) are both Galois connections between the posets \( (P, \leq) \) and \( (Q, \subseteq) \), then \( f^1 = f^2 \). Likewise, if \( (f_1, f^*) \) and \( (f_2, f^*) \) are both Galois connections between the same posets, then \( f_1 = f_2 \).

**Proof.** Take the basic principle (G) applied to the connection \( (f_*, f^1) \), put \( p = f^2(q) \), and we get \( f_*(f^2(q)) \subseteq q \iff f^2(q) \preceq f^1(q) \).

But by Theorem 102, applied to the connection \( (f_*, f^2) \), we know that \( f_*(f^2(q)) \subseteq q \).

So we can infer that, indeed, \( f^2(q) \preceq f^1(q) \).

Interchanging the two connections we can similarly show that \( f^1(q) \preceq f^2(q) \). And hence \( f^1(q) = f^2(q) \). But \( q \) was arbitrary, so indeed \( f^1 = f^2 \) as was to be shown.

The other part of the theorem is proved similarly.
Careful, though! This theorem does not say that, for any \( f_* \) which maps between \( (P, \preceq) \) and \( (Q, \sqsubseteq) \), there must actually exist a unique corresponding \( f^* \) in the reverse direction such that \( (f_*, f^*) \) form a Galois connection (we’ll note counterexamples in a moment). Nor does it say that when there is a Galois connection between the posets, it is unique (our toy examples have already shown that that is false too). The claim is only that, if you are given a possible left adjoint – or a possible right adjoint – there can be at most one candidate for its companion to complete a connection.

Given that adjoint functions determine each other, we naturally seek an explicit definition of one in terms of the other. Here it is:

**Theorem 105.** If \((f_*, f^*)\) is a Galois connection between the posets \((P, \preceq)\) and \((Q, \sqsubseteq)\), then

1. \( f^*(q) = \) the maximum of \( \{p \in P \mid f_*(p) \sqsubseteq q\} \),
2. \( f_*(p) = \) the minimum of \( \{q \in Q \mid p \preceq f^*(q)\} \).

**Proof.** To show (1), note that Theorem 102 yields (a) \( p \preceq f^*(f_*(p)) \), (b) \( f_*(f^*(q)) \sqsubseteq q \), and (c) \( f^* \) is monotone.

Suppose \( u \in \{p \in P \mid f_*(p) \sqsubseteq q\} \). Then \( f_*(u) \sqsubseteq q \). By (c), \( f^*(f_*(u)) \preceq f^*(q) \). Whence from (a), \( u \preceq f^*(q) \).

That shows \( f^*(q) \) is an upper bound for the set \( \{p \in P \mid f_*(p) \sqsubseteq q\} \). But by (b) we have \( f^*(q) \in \{p \in P \mid f_*(p) \sqsubseteq q\} \). So \( f^*(q) \) is indeed the maximum of the set.

The proof of (2) is simply dual. \( \Box \)

It follows that if \( f_* : P \to Q \) is a function such that the set \( \{p \in P \mid f_*(p) \sqsubseteq q\} \) doesn’t have a maximum for each \( q \in Q \), then there can’t be a corresponding right adjoint \( f^* : Q \to P \). To revert to a previous example, if we take the posets \((P, \preceq)\) and \((1, =)\) and let \( f_* : P \to 1 \) be only possible such function, then \( \{p \in P \mid f_*(p) = 1\} = P \), so if \( P \) has no maximum, \( f_* \) has no right adjoint.

Now recall again the posets \( \mathcal{N} = (\mathbb{N}, \leq) \) and \( \mathcal{Z} = (\mathbb{Q}^+, \leq) \) with the injection map \( i : \mathbb{N} \to \mathbb{Q}^+ \) and floor function \( f : \mathbb{Q}^+ \to \mathbb{N} \) which maps a rational to the natural corresponding to its integral part. Then we remarked before that \((i, f)\) is a Galois connection between \( \mathcal{N} \) and \( \mathcal{Z} \). Now we note that \((f, i)\) is not a Galois connection between \( \mathcal{Z} \) and \( \mathcal{N} \). Indeed, there can be no such connection of the form \((f, \cdot)\). For \( f(p) = 1 \) iff \( 1 \leq p < 2 \), and hence there is no maximum member of \( \{p \in \mathbb{Q}^+ \mid f(p) \leq 1\} \). Hence there is no right adjoint to \( f \).

Generalizing, we have the following:

**Theorem 106.** Galois connections are not necessarily symmetric. That is to say, given \((f_*, f^*)\) is a Galois connection between the posets \( \mathcal{P} \) and \( \mathcal{Z} \), it does not follow that \((f^*, f_*)\) is a connection between \( \mathcal{Z} \) and \( \mathcal{P} \).

### 16.5 Fixed points and closures

We continue with a portmanteau theorem:

**Theorem 107.** If \((f_*, f^*)\) is a Galois connection between the posets \((P, \preceq)\) and \((Q, \sqsubseteq)\), then

1. \( f_* \circ f^* \circ f_* = f_* \) and \( f^* \circ f_* \circ f^* = f^* \),
2. \( p \in f^*[Q] \) if and only if \( p \) is a fixed point of \( f^* \circ f_* \), and \( q \in f_*[P] \) if and only if \( q \) is a fixed point of \( f_* \circ f^* \).
Suppose more than just the one-object posets which are trivially isomorphic.

We know that a pair of posets which have a Galois connection between them needn’t be monotone. So \( f_*(p) \subseteq f_*(f_*(p))) \).

But an instance of the fundamental relation (G) yields \( f_*(f_*(p))) \subseteq (f_*(p) \iff f_*(f_*(p))) \). The r.h.s. is trivially true, so indeed \( f_*(f_*(p))) \subseteq (f_*(p). \)

The antisymmetry of \( \subseteq \) then entails \( f_*(f_*(p))) = (f_*(p). \) Since \( p \) was arbitrary, that gives us one half of (1), and the other half is dual.

(2) Suppose \( p \in f_*(Q). \) Then for some \( q \in Q, p = f_*(q) \) and hence \( f_*(f_*(p)) = f_*(f_*(f_*(q))) = f_*(q) = p, \) so \( p \) is a fixed point of \( f_* \).

Conversely suppose \( p \) is a fixed point of \( f_* \), so \( f_*(f_*(p)) = p. \) Then \( p \) is the value of \( f_*(q) \) for \( q = f_*(p), \) and hence \( p \in f_*(Q). \)

So \( p \in f_*(Q) \) if and only if \( p \) is a fixed point of \( f_* \), and that’s half of (2). The other half is dual.

(3) We have just seen that if \( p \in f_*(Q) \) then \( p \in f_*(f_*(p))) \) so \( p \in (f_*(f_*)[P]. \)

Therefore \( f_*(Q) \subseteq (f_*(f_*)[P). \)

Conversely, suppose \( p \in (f_*(f_*)[P), \) then as we saw before \( p \in f_*(Q). \) Therefore \( (f_*(f_*)[P) \subseteq f_*(Q). \)

Hence \( f_*(Q) = (f_*(f_*)[P), \) and that’s half of (3). The other half is dual. \( \square \)

Now let’s introduce a new bit of abbreviatory notation:

Definition 88. Suppose \( \hat{f} = (f_*, f_*) \) is a Galois connection between the posets \( \mathcal{P} = (P, \preceq) \) and \( \mathcal{D} = (Q, \preceq) \). Put \( \mathcal{P}^\dagger = (f_*(f_*)[P) \) and \( \mathcal{Q} = (f_*(f_*)[Q). \) Then we define \( \mathcal{P}^\dagger = (P, \preceq) \) and \( \mathcal{Q} = (Q, \preceq) \) – so these are sub-posets of \( \mathcal{P} \) and \( \mathcal{D} \) respectively.

Theorem 108. If \( \hat{f} = (f_*, f_*) \) is a Galois connection between the posets \( \mathcal{P} = (P, \preceq) \) and \( \mathcal{D} = (Q, \preceq) \), then \( \mathcal{P}^\dagger \) and \( \mathcal{Q} \) are order-isomorphic.

Proof. By the previous theorem, \( \mathcal{P}^\dagger = (f_*(Q), \preceq) \) and \( \mathcal{Q} = (f_*(P), \preceq) \). We show that \( f_* \) restricted to \( f_*(Q) \) provides the desired order isomorphism.

Note first that if \( p \in f_*(Q) \) then, for some \( q, p = f_*(q) \) and hence \( f_*(p) = f_*(f_*(q))) \) in \( Q = f_*(P). \) So \( f_* \) as required sends elements of \( f_*(Q) \) to elements of \( f_*(P). \) Moreover every element of \( f_*(P) \) is \( f_*(u) \) for some \( u \in f_*(Q). \) For if \( q \in f_*(P), \) then for some \( p, q = f_*(p) = f_*(f_*(f_*(p))) = f_*(u) \) where \( u = f_*(f_*(p)) \) in \( f_*(Q). \)

So \( f_* \) restricted to \( f_*(Q) \) is onto \( f_*(P). \) It remains to show that it is an order-embedding. We know that \( f_\) will be monotone, so we need to prove is that, if \( p, p' \in f_*(Q) \) and \( f_*(p) \subseteq f_*(p'), \) then \( p \preceq p'. \)

But if \( f_*(p) \subseteq f_*(p'), \) then by the monotonicity of \( f_\), \( f_*(f_*(p)) \preceq f_*(f_*(p')) \). Recall, though, that \( p, p' \in f_*(Q) \) are fixed points of \( f_\). Hence \( p \preceq p' \) as we want. \( \square \)

We know that a pair of posets which have a Galois connection between them needn’t be isomorphic overall, and can – so to speak – be lop-sidedly related. But this nice theorem says that they must, for all that, contain a pair of isomorphic sub-posets (and typically, more than just the one-object posets which are trivially isomorphic).

Definition 89. Suppose \( \mathcal{P} = (P, \preceq) \) is a poset; then a closure function on \( \mathcal{P} \) is a function \( c: P \rightarrow P \) such that, for all \( p, p' \in P, \)

(1) \( p \preceq c(p); \)

(2) if \( p \preceq p', \) then \( c(p) \preceq c(p'), \) i.e. \( c \) is monotone;

(3) \( c(c(p)) = c(p) \) i.e. \( c \) is idempotent.
Roughly speaking, then, a closure function \( c \) maps a poset ‘upwards’ to a subposet which then stays fixed under further applications of \( c \).

**Theorem 109.** If \( \langle f_*, f^* \rangle \) is a Galois connection between \( \mathcal{P} = (P, \leq) \) and some poset, then \( f^* \circ f_* \) is a closure function for \( P \).

**Proof.** We quickly check that the three conditions for closure apply. (i) is given by Theorem 102. (ii) is immediate as \( f^* \circ f_* \) is a composition of monotone functions. And for (iii), we know that \( f_* \circ f^* \circ f_* = f_* \), and hence \( (f^* \circ f_*) \circ (f^* \circ f_*) = f^* \circ f_* \). \( \square \)

### 16.6 One way a Galois connection can arise

The last three sections have been about Galois connections in general, and reveal that they have a perhaps surprisingly rich structure. In this final section, we note one particular way in which connections can arise.

**Theorem 110.** Let \( R \) be a binary relation between members of \( X \) and members of \( Y \). Define \( f_R: \mathcal{P}(X) \to \mathcal{P}(Y) \) by putting \( f_R(A) = \{ b \mid \forall a \in A aRb \} \), and similarly define \( f^R: \mathcal{P}(Y) \to \mathcal{P}(X) \) by putting \( f^R(B) = \{ a \mid \forall b \in B aRb \} \).

Then \( \langle f_R, f^R \rangle \) is a Galois connection between the inclusion posets \( (\mathcal{P}(X), \subseteq) \) and \( (\mathcal{P}(Y), \supseteq) \) – note the order reversal.

**Proof.** We just have to prove that principle (G) holds, i.e. for any \( A \subseteq X, B \subseteq Y, f_R(A) \supseteq B \) iff \( A \subseteq f^R(B) \).

But simply by applying definitions we see \( f_R(A) \supseteq B \) iff \( (\forall b \in B)(\forall a \in A) aRb \) iff \( (\forall a \in A)(\forall b \in B) aRb \) iff \( A \subseteq f^R(B) \). \( \square \)

Let’s say that Galois connection produced in this way is *relation-generated*. Galois’s original classic example was of this kind. And the modern classic example which is the topic of the next chapter is relation-generated too.
In his famous Dialectica paper ‘Adjointness in foundations’ (1969), F. William Lawvere writes of ‘the familiar Galois connection between sets of axioms and classes of models, for a fixed [signature]’. We are now in a position to explain this connection and draw out some consequences.

17.1 Structures and models

First, in this section, some quick reminders. Let $\mathcal{L}$ be a formal language. Then a set of $\mathcal{L}$-axioms in the wide sense that Lawvere is using is just any old set of $\mathcal{L}$-sentences. And by talk of ‘models’, Lawvere in fact just means structures apt for interpreting a language with $\mathcal{L}$’s signature. So the claim is that we can make a Galois connection between sets of sentences and classes of structures.

Take a familiar type of example. Suppose $\mathcal{L}$ is a classical first-order language. $\mathcal{L}$ will have its distinctive non-logical vocabulary. The particular choice of symbolism is unimportant, of course – what really matters is $\mathcal{L}$’s signature, which specifies $\mathcal{L}$ as having a certain fixed number of constants, a certain fixed number of predicates (each with a given arity), and a certain fixed number of function-symbols (again each with a given arity). A structure apt for interpreting $\mathcal{L}$ will have a domain, with distinguished elements assigned to each constant, sets of $n$-tuples from the domain assigned to the $n$-ary predicates, and appropriate sets of $n+1$-tuples from the domain assigned to the $n$-ary functions. We can then define what it is for an $\mathcal{L}$-sentence $\varphi$ to be true with respect to the $\mathcal{L}$-structure $\sigma$ in a now entirely familiar way.

But nothing that follows will in fact depend on $\mathcal{L}$’s being specifically classical or being first-order. We just need there to be some conception of a class of $\mathcal{L}$-structures apt for interpreting $\mathcal{L}$-sentences, and the idea of a particular $\mathcal{L}$-sentence being true with respect to a structure (and later we make some minimal assumptions about consistency and negation).

For the record, we recall some standard notation:

**Definition 90.** Let $\mathcal{L}$ be a given formal language, $\varphi$ be an $\mathcal{L}$-sentence, $\Gamma$ be a set of $\mathcal{L}$-sentences, and let $\sigma$ be $\mathcal{L}$-structure, i.e. a structure apt for interpreting $\mathcal{L}$. Then

1. If $\varphi$ is true w.r.t. $\sigma$, we say $\sigma$ is a model for $\varphi$ and write ‘$\sigma \models \varphi$’.
2. $\sigma$ is a model for $\Gamma$ – notated ‘$\sigma \models \Gamma$’ – iff, for every $\varphi \in \Gamma$, $\sigma \models \varphi$.
3. If every model for $\Gamma$ is a model for $\varphi$, we say $\Gamma$ semantically entails $\varphi$ and write ‘$\Gamma \models \varphi$’.
(4) If, in particular, if every model of \( \varphi \) makes \( \psi \) true too, we write \( \varphi \models \psi \) (rather than \( \{ \varphi \} \models \psi \)).

Note the second definition – the modern conventional one – contrasts with Lawvere’s wider usage of ‘model’ for structures more generally. The overloading of the symbol \( \models \) is of course entirely standard.

### 17.2 The Galois connection between syntax and semantics

Now we make the Galois connection between syntax and semantics which Lawvere refers to. We continue to assume that \( \mathcal{L} \) is a formal language, and \( \mathcal{L} \)-structures are those which are apt for interpreting it. Then we offer the following further definitions:

**Definition 91.** Let \( \mathcal{L} \) be the set of sentences of the language \( \mathcal{L} \) and let \( \mathcal{S} \) be the set of \( \mathcal{L} \)-structures. Then

1. \( \mathcal{L} \) is the poset \( (\mathcal{P}(L), \subseteq) \),
2. \( \mathcal{S} \) is the poset \( (\mathcal{P}(S), \supseteq) \),
3. \( f^*: \mathcal{P}(L) \to \mathcal{P}(S) \) is such that, for \( \Gamma \subseteq L \), \( f^*(\Gamma) = \{ \sigma \mid (\forall \varphi \in \Gamma) \sigma \models \varphi \} \),
4. \( f_*: \mathcal{P}(S) \to \mathcal{P}(L) \) is such that, for \( \Sigma \subseteq S \), \( f_*(\Sigma) = \{ \varphi \mid (\forall \sigma \in \Sigma) \sigma \models \varphi \} \).

(Ok, there’s a bug lurking here which will probably immediately strike you: but bear with me, and I’ll return to the point.)

The definitions here should all strike you as entirely natural ones. \( \mathcal{L} \) is the inclusion poset on sets-of-\( \mathcal{L} \)-sentences – an obvious syntactic construction to think about. \( \mathcal{S} \) is the dual inclusion poset on sets-of-\( \mathcal{L} \)-structures – apart from the reversing the order, an equally obvious semantic construction to think about. Then \( f_* \) is the obvious ‘find the models’ function. It takes a bunch of \( \mathcal{L} \)-sentences \( \Gamma \) and returns all the models of \( \Gamma \). In the other direction, \( f^* \) is the equally obvious natural ‘find all the true sentences’ function. It takes a bunch of \( \mathcal{L} \)-structures and returns the set of \( \mathcal{L} \)-sentences that are true in all of those structures.

We then have

**Theorem 111.** \((f_*, f^*)\) is a Galois connection between \( \mathcal{L} \) and \( \mathcal{S} \).

**Proof.** Use Theorem 110 – putting the generating relation \( R \) between a sentence and a structure to be the converse of \( \models \).

### 17.3 Drawing some consequences

Terrific! Now we can just turn the handle, and apply all those general theorems about Galois connections from the last chapter to our special case of the connection between ‘syntax’ \( \mathcal{L} \) and ‘ semantics’ \( \mathcal{S} \). So let’s collect together some of the implications:

**Theorem 112.** Let \( \mathcal{L} = (\mathcal{P}(L), \subseteq), \mathcal{S} = (\mathcal{P}(S), \supseteq) \) and the Galois connection \((f_*, f^*)\) between them be as defined in Defn. 91. Then

1. \( f_* \) is monotone, i.e. if \( \Gamma \subseteq \Delta \) then \( f_*(\Gamma) \supseteq f_*(\Delta) \),
2. for any \( \Gamma \subseteq L \), \( \Gamma \subseteq (f^* \circ f_*)(\Gamma) \),
3. \( f_* \circ f^* \circ f_* = f_* \),
4. \( \Gamma \in f^*[\mathcal{P}(S)] \) iff \( \Gamma \) is a fixed point of \( f^* \circ f_* \).
And dually,

(5) $f^*$ is monotone, i.e. if $\Sigma \supseteq \Xi$ then $f^*(\Sigma) \subseteq f^*(\Xi)$,

(6) for any $\Sigma \subseteq S$, $(f_\ast \circ f^*)[\Sigma] \supseteq \Sigma$,

(7) $f^* \circ f_\ast \circ f^* = f^*$,

(8) $\Sigma \in f_\ast[P(L)]$ iff $\Sigma$ is a fixed point of $f_\ast \circ f^*$.

The proofs of all these claims are already to hand from the last chapter (see Theorems 102 and 107).

But what do they really mean? Well, result (1) just reminds us that if the set of sentences $\Gamma$ is contained in the set of sentences $\Delta$ then the set of models of $\Gamma$ contains the models of $\Delta$. Which is simply the point that, as we expand a set of sentences, we can’t increase the number of ways of making them all true together.

What about results (2) to (4)? To get a handle on these, let’s consider the significance of the composite map $f^* \circ f_\ast$. By definition, $f_*[\Gamma]$ is the set of all structures which are models for $\Gamma$. So $(f^* \circ f_\ast)[\Gamma]$ returns the set of all sentences which are true in all those structures which are models for $\Gamma$. That is to say, $\varphi \in (f^* \circ f_\ast)[\Gamma]$ iff every interpretation which makes all of $\Gamma$ true makes $\varphi$ true too. Hence $(f^* \circ f_\ast)[\Gamma]$ is exactly the set of semantic logical consequences of $\Gamma$.

This means we are in very familiar territory. Recalling some standard definitions, we have

**Definition 92.** (1) A set of $L$-sentences $\Gamma$ is closed under (semantic) logical consequence just in case, if $\Gamma \models \varphi$, then $\varphi \in \Gamma$.

(2) An $L$-theory $\Theta$ is a set of $L$-sentences closed under logical consequence.

(3) The theory $\Theta$ with axioms $\Gamma$ is the smallest theory containing $\Gamma$.

‘Theories’ are, of course, equally often defined as sets of sentences closed under some syntactic deducibility relation rather than as sets closed under semantic consequence (though in a first-order context the definitions come to the same because of the completeness theorem). So let’s emphasize: it is the semantic relation that is in play in our definition here.

And in these terms, then, our previous remarks show that if $\Theta$ is a theory, then $(f^* \circ f_\ast)[\Theta]$ is $\Theta$ (closing under logical consequence a set of sentences already closed under logical consequences yields the same set of sentences). And hence

**Theorem 113.** $(f^* \circ f_\ast)[\Gamma]$ is the theory with axioms $\Gamma$.

*Proof.* We’ve seen that $(f^* \circ f_\ast)[\Gamma]$ is closed under logical consequence and is hence a theory. So we just need to confirm that $(f^* \circ f_\ast)[\Gamma]$ is the smallest theory containing $\Gamma$. But suppose $\Theta$ is a theory and $\Gamma \subseteq \Theta$. Then by the monotonicity of the composite function, $(f^* \circ f_\ast)[\Gamma] \subseteq (f^* \circ f_\ast)[\Theta] = \Theta$.  

Looked at through the lens of this result, parts (2) to (4) of our previous theorem now become familiar near-trivia. Thus (2) just says again that, given a set of sentences as axioms, the theory with those axioms includes the axioms! (3) says that if you first round out a bunch of sentences $\Gamma$ by taking their consequences to get the theory $f^*(f_\ast(\Gamma))$ and then look for the theory’s models, you’ll get the same upshot as if you’d just looked for the models of $\Gamma$ straight off. And (4) tells us that if $\Gamma$ is a set of just those sentences made true across a bunch of structures, then it is closed under logical consequence.

And what about the other parts of Theorem 112? (5) just confirms our expectations again: it says that if the set of structures $\Sigma$ contains $\Xi$, then the set of truths verified
by all the structures in $\Sigma$ is contained in the set of truths verified by all the structures in $\Xi$. But as for the other results, what’s their significance? What does the map $f_s \circ f^*$ do for us?

Well, $f^*[\Sigma]$ is the set of sentences from $L$ made true by every structure in $\Sigma$. So $f^*[\Sigma]$ is a theory (that’s because the logical consequences of any sentences in $f^*[\Sigma]$ will also be made true by every structure in $\Sigma$, so $f^*[\Sigma]$ is closed under logical consequence). And hence $(f_s \circ f^*)[\Sigma]$ is in turn the set of all structures which are models for that theory. This links up with another familiar pair of ideas:

**Definition 93.** (1) A set of $L$-structures $\Sigma$ is **axiomatized** by the $L$-theory $\Theta$ just in case $\sigma \in \Sigma$ if and only if $\sigma$ is a model of $\Theta$.

(2) A set of $L$-structures $\Sigma$ is **axiomatizable** iff there is an $L$-theory which axiomatizes it.

And by a proof dual to that of the previous theorem

**Theorem 114.** $(f_s \circ f^*)[\Sigma]$ is the smallest axiomatizable set of structures containing $\Sigma$.

And looked at through the lens of this result, the last three parts of Theorem 112 again turn into near-trivia. Thus, (6) just says that the smallest axiomatizable class containing $\Sigma$ contains $\Sigma$; (7) says, in effect, that if you first round out a bunch of models $\Sigma$ to get the minimal axiomatizable class containing it and then look an axiomatization for those models, you get to the same place as if you’d just looked for the theory of $\Sigma$ straight off; and (8) tells us that if $\Sigma$ is a set of just those structures that make true some bunch of sentences, then it is axiomatizable set.

### 17.4 Is this all trivial, then?

‘Hold on! Is that it? Those implications of Theorem 112 really are, as frankly acknowledged, familiar near-trivialities! Have we laboured so hard to bring forth such a mouse?’

An understandable first reaction, perhaps, but one that entirely misses the point.

We are not in the business here of aiming to prove exciting new results about theories and the structures they are about: rather we are trying to fit very familiar old thoughts (at least, old thoughts well known to logicians) into a very much more general order-theoretic framework (less familiar, perhaps, to logicians). Look at it this way. Start from the fundamental true-of relation which can obtain between an $L$-sentence and an $L$-structure. This, as we’ve seen, immediately generates a certain Galois connection $(f_s, f^*)$ between two naturally ordered posets whose elements are sets of sentences and sets of structures. And this by itself already dictates that e.g. the composite map $f^* \circ f_s$ has to have a special significance as a closure function. So in this way, the notion of the theory $f^* \circ f_s[\Gamma]$ with axioms $\Gamma$, with all the properties we now expect of that notion, emerges as generated by a basic construction that appears all over the place in mathematics.

This setting of familiar ideas into a wider abstract framework – so we get to see local results in one domain as in fact exemplifying a very general pattern that can be found across many domains – is, as we are seeing, characteristic of modern mathematics in general and of category theory in particular. And showing in this way how the local fits into a general pattern is one kind of explanatory exercise, revealing how the particular case exemplifies a ‘natural kind’ of phenomenon in perhaps surprising ways.
17.5 Another implication

There is perhaps not a great deal more juice we can usefully squeeze out of the idea of a Galois connection applied to the simple observation that the ‘true of’ relation generates such a connection between sets of $\mathcal{L}$-sentences and sets of $\mathcal{L}$-structures. But let’s make one more observation.

Applying Theorem 108 and using the same notation, we have

**Theorem 115.** If $f = (f_*, f^*)$ is the defined Galois connection between the posets $\mathcal{L}$ and $\mathcal{J}$, then $\mathcal{L}^f$ and $\mathcal{J}^f$ are order-isomorphic, and (the restriction of) $f_*$ provides an order-isomorphism between them.

$\mathcal{L}^f$ is the poset $((f^* \circ f_*)[\mathcal{P}(\mathcal{L})], \subseteq)$, which is the poset of theories built in the language $\mathcal{L}$, partially ordered by inclusion. This poset evidently has a maximum, namely the inconsistent theory containing all $\mathcal{L}$-sentences. Then, at the top of the poset in the ordering, just under the maximum, will be those theories $\Theta$ such that adding even one more new axiom to $\Theta$ which isn’t already in the set takes us back to the maximal, inconsistent, theory. Such a $\Theta$ is consistent at least in Post’s sense of not containing all $\mathcal{L}$-sentences. And, assuming the language has a negation operator obeying minimal normal constraints, $\Theta$ is also negation-complete, i.e. for every $\mathcal{L}$-sentence $\varphi$, either $\Theta \models \varphi$ or $\Theta \models \neg \varphi$. (For suppose otherwise. Then since $\Theta \not\models \varphi$, $\varphi \notin \Theta$. And since $\Theta \not\models \neg \varphi$, $\Theta \cup \{\varphi\}$ is consistent, i.e. we can add a new axiom to $\Theta$ without getting inconsistency, contrary to assumption.) And as we go down the poset $\mathcal{L}^f$ further in the ordering, then we move from the inconsistent theory, through the negation-complete theories, on to more and more partial theories, till we get down to the theory consisting of just logical truths of $\mathcal{L}$ (which belong to any $\mathcal{L}$-theory) as the minimum.

Now, our theorem tells us that $f_*$ is an order-isomorphism between $\mathcal{L}^f$, our poset of theories, and a corresponding poset of axiomatizable-sets-of-structures, $\mathcal{J}^f$. And how is that second poset built up? Well, start with the maximum of $\mathcal{L}^f$, the inconsistent theory: then $f_*$ will map that across to the maximum of $\mathcal{J}^f$, namely the empty set of structures (remember which way up the ordering is on the structures side of the Galois connection). And then, just below the maximum of $\mathcal{L}^f$, $f_*$ maps each consistent negation-complete theory to the corresponding set of its models. But the models of a negation-complete theory $\Theta$ agree on the truth-value of every $\mathcal{L}$-sentence (for they make $\varphi$ true if $\Theta \models \varphi$ and $\varphi$ false if $\Theta \models \neg \varphi$ and one of those must hold). So $f_*$ maps the negation-complete theories to sets of elementarily equivalent structures, in the model-theorist’s sense (and the map is onto and injective).

Then, as we go further down the poset $\mathcal{L}^f$ we get more and more partial theories which settle the truth-values of more and more limited classes of sentences. And $f_*$ maps these partial theories to more inclusive bunches of structures, i.e. elements of $\mathcal{J}^f$, which need agree on less and less. Again, a familiar story really: but its contours are rather neatly brought out by thinking in terms of the Galois connection between syntax and semantics.

17.6 A bug?

Suppose that $\mathcal{L}$ is, for example, a standard first-order language. Reflect that any non-empty set can in principle be made into a structure for interpreting $\mathcal{L}$. Just take the set as the domain, select out enough elements (repetitions are allowed!) to assign to constants as denotations, and similarly define such $n$-ary relations and functions over the set as are needed to interpret a language with the given signature. And now think again about
our definition of the poset $\mathcal{S}$. Its carrier ‘set’ is supposed to be the powerset of the set $S$ of $\mathcal{L}$-structures. But $S$ is in this case as big as the universe of all sets, period. So the carrier ‘set’ for $\mathcal{S}$, on standard views, is then too big to be a kosher set. It’s a collection whose members are proper classes of structures – which is no doubt why Lawvere did here talk of ‘classes of models’. So $\mathcal{S}$ can’t, it seems, really be a poset. Here then is the potential bug I alluded to earlier. What to do?

The short answer, for our purposes, is not worry! We looked before at some options for dealing with sets that are seemingly ‘too big’ (see §7.3). Let’s mention just two possible lines again, at different ends of the spectrum.

1. At the non-committal end, maybe we can think of the posets we seem to have been talking about as no more than virtual collections. For it seems that, for most of our discussion, we might have used a plural idiom instead and simply talked about having some objects and a partial ordering defined over them (and then about selections from these objects, also thought of plurally, and partial orderings over them). The suggestion is, then, that to get the elements of the theory of order under way, and to talk about Galois connections in particular, we don’t really need to think of the various objects we are putting into an order as themselves constituting a new object, the set of them. (Of course, the set-idiom is utterly familiar, and it would indeed be distractingly perverse to go out of our way to avoid it. And the implicit extra commitment to many-objects-as-one-set which comes with our set-talk is mostly entirely harmless. But, the hope is, when it does cause trouble, we can rephrase to avoid it.)

2. Going to the other extreme, we could buy into a big set ontology, but take it that the structures talked about by logicians normally involve sets-smaller-than-some-inaccessible (surely that’s enough for most purposes!), and then the posets we’ve been dealing with are just yet bigger sets.

Let’s be cheerfully agnostic about whether either of these, or some other line, indicates the best direction to go. However things come out in the wash, we don’t want the neat insight that there is a Galois connection between syntax and semantics to get swept away on a technicality!
Recall that quotation from Tom Leinster which we gave at the very outset:

Category theory takes a bird’s eye view of mathematics. From high in the sky, details become invisible, but we can spot patterns that were impossible to detect from ground level. (Leinster, 2014, p. 1)

Perhaps the most dramatic patterns that category theory newly reveals involve adjunctions. As Mac Lane famously puts it (1997, p. vii) the slogan is “Adjoint functors arise everywhere.” In the last two chapters, we have seen a restricted version of the phenomenon (well known before category theory). But category theory enables us to generalize radically.

18.1 Adjoint functors: a first definition

(a) Let \( P \) now be (not the poset itself but) the category corresponding to the poset \((P, \preceq)\). So the objects of \( P \) are the members of \( P \), and there is a \( P \)-arrow \( p \to p' \) (for \( p, p' \in P \)), which we can identify with the pair \( \langle p, p' \rangle \), if and only if \( p \preceq p' \). Similarly let \( \mathcal{Q} \) be the category corresponding to the poset \((Q, \sqsubseteq)\).

Now, a Galois connection between the posets \((P, \preceq)\) and \((Q, \sqsubseteq)\) is a pair of functions \( f: P \to Q \) and \( g: Q \to P \) such that

(i) \( f \) and \( g \) are monotone, and

(ii) \( f(p) \sqsubseteq q \) iff \( p \preceq g(q) \) for all \( p \in P, q \in Q \).

(Well, we know condition (ii) implies condition (i), but it is helpful now to make it explicit.) However, monotone functions \( f, g \) between posets give rise to functors \( F, G \) between the corresponding categories – see §3.2, Ex. (F4). Thus the monotone function \( f: P \to Q \) gives rise to the functor \( F: \mathcal{P} \to \mathcal{Q} \) which sends the object \( p \) in \( \mathcal{P} \) to \( f(p) \) in \( \mathcal{Q} \), and sends an arrow \( p \to p' \) in \( \mathcal{P} \), i.e. the pair \( \langle p, p' \rangle \), to the pair \( \langle f(p), f(p') \rangle \) which is an arrow in \( \mathcal{Q} \). Similarly, \( g: Q \to P \) gives rise to a functor \( G: \mathcal{Q} \to \mathcal{P} \).

So (ii) means that our adjoint functions, i.e. the Galois connection \( (f, g) \) between the posets \((P, \preceq)\) and \((Q, \sqsubseteq)\), gives rise to a pair of functors \( (F, G) \) between the poset categories \( \mathcal{P} \) and \( \mathcal{Q} \), one in each direction, such that there is a (unique) arrow \( F p \to q \) in \( \mathcal{Q} \) iff there is a corresponding (unique) arrow \( p \to G q \) in \( \mathcal{P} \). This sets up an isomorphism between the hom-sets \( \mathcal{Q}(Fp, q) \) and \( \mathcal{P}(p, Fq) \), for each \( p \in \mathcal{P}, q \in \mathcal{Q} \).

Of course, for a particular choice of \( p, q \), this will be a rather trivial isomorphism, as the homsets are either both empty or both single-membered. But what isn’t trivial
is that the isomorphism arises systematically from the Galois connection, in a uniform way independently of our choice of \( p \) and \( q \). Informally, it is a natural isomorphism. And we now know how to put that informal claim into more formal category-theoretic terms: we have \( \mathcal{P}(Fp, q) \cong \mathcal{P}(p, Fq) \) naturally in \( p \in \mathcal{P}, q \in \mathcal{Q} \).

(b) Now we generalize in the obvious way, and also introduce some absolutely standard notation:

**Definition 94.** Suppose \( \mathcal{A} \) and \( \mathcal{B} \) are categories and \( F: \mathcal{A} \to \mathcal{B} \) and \( G: \mathcal{B} \to \mathcal{A} \) are functors. Then \( F \) is left adjoint to \( G \) and \( G \) is right adjoint to \( F \), notated \( F \dashv G \), iff

\[
\mathcal{B}(F(A), B) \cong \mathcal{A}(A, G(B))
\]

naturally in \( A \in \mathcal{A}, B \in \mathcal{B} \). We also write \( \mathcal{A} \xrightarrow{F} \mathcal{B} \) when this situation obtains.

There is an additional fairly standard bit of notation to indicate the action of the natural isomorphism between the hom-sets:

**Definition 95.** Given the situation just described, and an arrow \( f: F(A) \to B \), then one direction of the natural bijection between the hom-sets sends that arrow to its transpose \( f: A \to G(B) \); likewise the inverse bijection associates an arrow \( g: A \to G(B) \) to its transpose \( g: F(A) \to B \).

Evidently, transposing twice takes us back to where we started: \( \overline{\overline{f}} = f \) and \( \overline{\overline{g}} = g \).

As we’d expect from our discussion of the special case of Galois connections, given an adjoint connection (or adjunction) \( F \dashv G \), we can deduce a range of additional properties of the adjoint functors and of the operation of transposition. But before exploring this any further in the abstract, let’s have some more examples of adjunctions.

### 18.2 Examples

As a warm-up exercise, we start with a particularly easy case:

1. Consider any (non-empty!) category \( \mathcal{A} \) and the one object category \( \mathbf{1} \) (comprising just the object \( \bullet \) and its identity arrow). There is a unique functor \( F: \mathcal{A} \to \mathbf{1} \). Questions: when does \( F \) have a right adjoint \( G: \mathbf{1} \to \mathcal{A} \)? what about a left adjoint?

   If \( G \) is to be a right adjoint, remembering that \( \mathcal{P}(\mathbf{1}, \mathbf{1}) = \{\text{identity arrow}\} \), we require

\[
\mathcal{A}(A, G(B)) \cong \mathcal{P}(\mathbf{1}, \mathbf{1})
\]

for any \( A \). The hom-set on the left contains just the identity arrow. So that can only be in bijection to the hom-set on the right, for each \( A \), if there is always a *unique* arrow \( A \to G\bullet \), i.e. if \( G\bullet \) is terminal in \( \mathcal{A} \).

In sum, \( F \) has a right adjoint \( G: \mathbf{1} \to \mathcal{A} \) just in case \( G \) sends \( \mathbf{1} \)'s unique object to \( \mathcal{A} \)'s terminal object: no terminal object, no right adjoint.

Dually, \( F \) has a left adjoint if and only if \( \mathcal{A} \) has an initial object.

This toy example reminds of what we have already seen in the special case of Galois connections, namely that a functor may or may not have a right adjoint, and independently may or may not have a left adjoint, and if both adjoints exist they may be different. But let’s also note that we have here a first indication that adjunctions and limits can interact in interesting way: in this case, indeed, we could *define* terminal and initial objects for a category \( \mathcal{A} \) in terms of the existence of right and left adjoints to the functor \( F: \mathcal{A} \to \mathbf{1} \). We will return to this theme.
Now for a couple of more substantive examples. And by the way, to speed things along, we won’t in this chapter prove that the relevant hom-sets in our various examples are naturally isomorphic in the official formal sense: we will take it as enough to find a bijection which can be set up in a systematic and intuitively natural way, without arbitrary choices.

(2) Let’s next consider the forgetful functor \( U : \text{Top} \to \text{Set} \) which sends each topological space to its underlying set of points, and sends any continuous function between topological spaces to the same function thought of as a set-function. Questions: does this have a left adjoint? a right adjoint?

If \( U \) is to have a left adjoint \( F : \text{Set} \to \text{Top} \), then for any set \( S \) and for any topological space \( (T,O) \) – with \( T \) a set of points and \( O \) a topology (a suitable collection of open sets) – we require

\[
\text{Top}(F(S), (T,O)) \cong \text{Set}(S, U(T,O)) = \text{Set}(S, T),
\]

where the bijection here needs to be a natural one.

Now, on the right we have the set of all functions \( f : S \to T \). So that needs to be in bijection with the set of all continuous functions from \( FS \) to \( (T,O) \). How can we ensure this holds in a systematic way, for any \( S \) and \( (T,O) \)? Well, suppose that for any \( S, F \) sends \( S \) to the topological space \( (S,D) \) which has the discrete topology (i.e. all subsets of \( S \) count as open). It is a simple exercise to show that every function \( f : S \to T \) then counts as a continuous function \( f : (S,D) \to (T,O) \). So the functor \( F \) which assigns a set the discrete topology will indeed be left adjoint to the forgetful functor.

Similarly, the functor \( G : \text{Set} \to \text{Top} \) which assigns a set the indiscrete topology (the only open sets are the empty set and \( S \) itself) is right adjoint to the forgetful functor \( U \).

(3) Let’s now take another case of a forgetful functor, this time the functor \( U : \text{Mon} \to \text{Set} \) which forgets about monoidal structure. Does \( U \) have a left adjoint \( F : \text{Set} \to \text{Mon} \). If \( (M, \cdot) \) is a monoid and \( S \) some set, we need

\[
\text{Mon}(FS, (M, \cdot)) \cong \text{Set}(S, U(M, \cdot)) = \text{Set}(S, M).
\]

The hom-set on the right contains all possible functions \( f : S \to M \). How can these be in one-one correspondence with the monoid homomorphisms from \( FS \) to \( (M,\cdot) \)?

Arm-waving for a moment, suppose \( FS \) is some monoid with a lot of structure (over and above the minimum required to be a monoid). Then there may be few if any monoid homomorphisms from \( FS \) to \( (M,\cdot) \). Therefore, if there are potentially to be lots of such monoid homomorphisms, one for each \( f : S \to M \), then \( FS \) will surely need to have minimal structure. Which suggests going for broke and considering the limiting case, i.e. the functor \( F \) which sends a set \( S \) to \( (S^*,\ast) \), the free monoid on \( S \) which we met back in §3.2, Ex. (F2). Recall, the objects of \( (S^*,\ast) \) are sequences of \( S \)-elements (including the null sequence) and its monoid operation is concatenation.

There is an obvious map \( \alpha \) which takes an arrow \( f : S \to M \) and sends it to \( \overline{f} : (S^*,\ast) \to (M,\cdot) \), where \( \overline{f} \) maps the empty sequence of \( S \)-elements to the unit of \( M \), and maps the finite sequence \( x_1 \ast x_2 \ast x_3 \ast \ldots \ast x_n \) to the \( M \)-element \( f_{x_1} \cdot f_{x_2} \cdot f_{x_3} \cdot \ldots \cdot f_{x_n} \). So defined, \( \overline{f} \) respects the unit and the monoid operation and so is a monoid homomorphism.

There is an equally obvious map \( \beta \) which takes an arrow \( g : (S^*,\ast) \to (M,\cdot) \) to the function \( \overline{g} : S \to M \) which sends an element \( x \in S \) to \( g(x) \) (i.e. to \( g \) applied to the one-element list containing \( x \)).
Evidently $\alpha$ and $\beta$ are inverses, so form a bijection, and their construction is quite general (i.e. can be applied to any set $S$ and monoid $(M, \cdot)$). Which establishes that, as required $\text{Mon}(FS, (M, \cdot)) \cong \text{Set}(S, M)$.

So in sum, the free functor $F$ which takes a set to the free monoid on that set is left adjoint to the forgetful function $U$ which sends a monoid to its underlying set.

18.3 The uniqueness of adjoints

We have just shown e.g. that, on monoids, the free functor is left adjoint to the forgetful functor. But is it the unique left adjoint?

Theorem 104 proved a uniqueness result for a special case: there we saw if a function has a left (right) adjoint in a Galois connection, then that adjoint is unique. The corresponding generalized result is this:

**Theorem 116.** Assume we are dealing with locally small categories. Given functors $F: \mathcal{A} \to \mathcal{B}$ and $G, G': \mathcal{B} \to \mathcal{A}$ such that $F \dashv G$ and $F \dashv G'$, then $G \cong G'$.

Likewise, given functors $F, F': \mathcal{A} \to \mathcal{B}$ and $G: \mathcal{B} \to \mathcal{A}$ such that $F \dashv G$ and $F' \dashv G$, then $F \cong F'$.

So, adjoints (when they exist) are determined uniquely up to natural isomorphism.

**Proof sketch.** It is enough to establish the first case, as the second follows by duality. Because of the two adjunctions, we have

$$\mathcal{A}(A, GB) \cong \mathcal{B}(FA, B) \cong \mathcal{A}(A, G'B),$$

naturally in $A \in \mathcal{A}$, $B \in \mathcal{B}$. The naturalness in $A$ by definition means that (a) the hom-functors $\mathcal{A}(-, GB)$ and $\mathcal{A}(-, G'B)$ are naturally isomorphic. And (a) holds for all $B$ and naturally so.

But (a) in turn means that $\mathcal{Y}GB \cong \mathcal{Y}G'B$, where $\mathcal{Y}$ is the Yoneda embedding (see Theorem 42 and Defn. 39). So we can invoke Theorem 44 to conclude (b) $GB \cong G'B$. This still holds for all $B$, and naturally so. Which implies that $G \cong G'$. $\square$

We have rushed a bit at the end here, and will give a more plodding proof later. But taking the result as established, we are now justified in the talking of the left or right adjoint of a given functor, meaning (as so often in category theory) that we have uniqueness up to isomorphism.

18.4 Examples, continued

Our previous example involving monoids is actually typical of a whole cluster of cases. The left adjoint of the trivial forgetful functor from some class of algebraic structures to their underlying sets is characteristically the non-trivial functor that takes us from a set to a free structure of that algebraic kind. Thus we have, for example,

(4) The forgetful functor $U: \text{Grp} \to \text{Set}$ has as left adjoint the functor $F: \text{Set} \to \text{Grp}$ which sends a set to the free group on that set (i.e. the group obtained from a set $S$ by adding just enough elements for it to become a group while imposing no constraints other than those required to ensure we indeed have a group).

What about right adjoints to our last two forgetful functors?
(5) Consider again the forgetful functor $U: \text{Mon} \to \text{Set}$. It would have a right adjoint $G: \text{Set} \to \text{Mon}$ just in case $\text{Set}(M, S) = \text{Set}(U(M, \cdot), S) \cong \text{Mon}(M, \cdot), GS)$, for all monoids $(M, \cdot)$ and sets $S$. But this requires the monoid homomorphisms from $(M, \cdot)$ to $GS$ always to be in bijection with the set-functions from $M$ to $S$. But that’s not possible (consider keeping $M$ and $S$ fixed, but spinning the possible monoid operations on $M$ governed by different constraining equations). Which shows that $U: \text{Mon} \to \text{Set}$ has no right adjoint. Similarly for the forgetful functor $U: \text{Grp} \to \text{Set}$.

(6) There are however examples of less forgetful algebraic functors which have both left and right adjoints. Take the functor $U: \text{Grp} \to \text{Mon}$ which forgets about group inverses but keeps the monodical structure. This has a left adjoint $F: \text{Mon} \to \text{Grp}$ which converts a monoid to a group by adding inverses for elements (and otherwise making no more assumptions that are needed to get a group). $U$ also has a right adjoint $G: \text{Mon} \to \text{Grp}$ which rather than adding elements subtracts them by mapping a monoid to the submonoid of its invertible elements (which can be interpreted as a group).

Let’s quickly check just the second of those claims. We have $U \dashv G$ so long as

$$\text{Mon}(U(K, \times), (M, \cdot)) \cong \text{Grp}((K, \times), G(M, \cdot)),$$

for any monoid $(M, \cdot)$ and group $(K, \times)$. Now we just remark that every element of $(K, \times)$-as-a-monoid is invertible and a monoid homomorphism sends invertible elements to invertible elements. Hence a monoid homomorphism from $(K, \times)$-as-a-monoid to $(M, \cdot)$ will in fact also be a group homomorphism from $(K, \times)$ to the submonoid-as-a-group $G(M, \cdot)$.

And now for some cases not involving forgetful functors:

(7) Consider the functor $- \times B: \text{Set} \to \text{Set}$ which acts on object by sending a set $A$ to the set $A \times B$ (and does the obvious thing on arrows). Then $- \times B$ has a right adjoint, namely the functor $(-)^B: \text{Set} \to \text{Set}$ which sends a set $C$ to the set of set-functions from $B$ to $C$. For we have

$$\text{Set}(A \times B, C) \cong \text{Set}(A, C^B).$$

since a function $f$ acting on a pair of objects $a, b$ can evidently be associated one-to-one (naturally!) with a corresponding function $\overline{f}$ which acts on $a$ and outputs the function whose action on $b$ outputs $f(a, b)$ (compare our earlier remarks on ‘currying’ in §12.6).

Of course, $C^B$ is none other than $\text{Set}(B, C)$: but the old exponential notation for the set of functions from $B$ to $C$ is standard before we ever get to category theory and has a familiar motivation. What we have now seen, though, is that in shorthand categorial terms, exponentiation-in-$\text{Set}$ is right-adjoint to Cartesian product. And this invites generalization: we can say more generally that a category has exponentiation if it has right adjoints to product functors. We will return to this point.

(8) Recall Defn. 17 which defined the product of two categories. Given a category $\mathcal{C}$ there is a trivial diagonal functor $\Delta: \mathcal{C} \to \mathcal{C} \times \mathcal{C}$ which sends a $\mathcal{C}$-object $A$ to the pair $\langle A, A \rangle$, and sends a $\mathcal{C}$-arrow $f$ to the pair of arrows $\langle f, f \rangle$. What would it take for this functor to have a right adjoint $G: \mathcal{C} \times \mathcal{C} \to \mathcal{C}$? We’d need

$$\mathcal{C} \times \mathcal{C}(\langle A, A \rangle, \langle B, C \rangle) \cong \mathcal{C}(A, G(B, C)).$$
naturally in \( A \in \mathcal{C} \) and in \( \langle B, C \rangle \in \mathcal{C} \times \mathcal{C} \). But by definition the left hand hom-set is \( \mathcal{C}(A, B) \times \mathcal{C}(A, C) \). But then if we can take \( G \) to be the product functor that sends \( \langle B, C \rangle \) to the product object \( B \times C \) in \( \mathcal{C} \) we’ll get the desired natural transformation

\[
\mathcal{C}(A, B) \times \mathcal{C}(A, C) \cong \mathcal{C}(A, B \times C).
\]

So in sum, \( \Delta : \mathcal{C} \to \mathcal{C} \times \mathcal{C} \) has a right adjoint if \( \mathcal{C} \) has binary products. We will return to this point too.

(9) For topologists, let’s have another example of a case where the adjoint of a trivial functor is something much more substantial. The inclusion functor from \( \text{KHaus} \), the category of compact Hausdorff spaces, into \( \text{Top} \) has a left adjoint, namely the Stone-Čech compactification functor.

18.5 Naturality

We said: \( F \dashv G \) just in case

\[
\mathscr{B}(F(A), B) \cong \mathscr{A}(A, G(B))
\]

holds naturally in \( A \in \mathscr{A}, B \in \mathscr{B} \). Let’s now pause to be more explicit about what the official naturality requirement comes to.

Take first naturality in \( B \). We have in play two hom-functors from \( \mathscr{B} \) to \( \text{Set} \), namely \( \mathscr{B}(F(A), -) \) and \( \mathscr{A}(A, G(-)) \). To reduce clutter, temporarily denote those functors \( P(-) \) and \( Q(-) \) respectively.

Applying Defn. 21 and Defn. 24, the displayed isomorphism holds naturally in \( B \in \mathscr{B} \) just in case there’s a family of isomorphisms \( \psi_B : P(B) \to Q(B) \) such that for every \( h : B \to B' \), the usual naturality square always commutes: i.e.

\[
\begin{array}{ccc}
P(B) & \xrightarrow{P(h)} & P(B') \\
\downarrow{\psi_B} & & \downarrow{\psi_{B'}} \\
Q(B) & \xrightarrow{Q(h)} & Q(B')
\end{array}
\]

But how does the covariant hom-functor \( P \) operate on \( h \)? \( P \) sends \( B \) and \( B' \) to the hom-sets \( \mathscr{B}(F(A), B) \) and \( \mathscr{B}(F(A), B') \): and as usual – see §8.2 – \( P \) will to send \( h : B \to B' \) to \( h \circ - \), i.e. to the function which composes \( h \) with an arrow from \( \mathscr{B}(F(A), B) \) to give an arrow in \( \mathscr{B}(F(A), B') \). Similarly, \( Q \) will send \( h \) to \( Gh \circ - \).

So consider an arrow \( f : F(A) \to B \) living in \( \mathscr{B}(F(A), B) \) i.e. in \( \mathscr{B}(F(A), B) \). The naturality square now tells us that for any \( h : B \to B' \), \( \psi_{B'}(h \circ f) = Gh \circ \psi_B(f) \). So since the components of \( \psi \) send an arrow in \( \mathscr{B}(F(A), B) \) to its transpose, we can write that as \( \overline{h \circ f} = Gh \circ \overline{f} \).

Dually, consider an arrow \( g : A \to G(B) \) living in \( \mathscr{A}(A, G(B)) \). Then for any \( k : A' \to A \) (note the reversal because the relevant hom-functors are now contravariant!\), \( g \circ k = \overline{g} \circ Fk \) – so, for future use, \( \overline{g} \circ Fk = g \circ k \).

18.6 A second definition of adjoints

(a) We now know what it takes for a pair of functors to be adjoint to each other, and we have given various examples of adjoint pairs (to add to the special cases from the previous two chapters where the adjunctions are Galois connections).
Now, our first definition of adjunctions was inspired by our original definition of Galois connections in §16.2. But we gave an alternative definition of such connections in §16.3. This too can be generalized to give a second definition of adjunctions. In this section we show how, and prove that the new definition is equivalent to our first one. (This alternative definition will turn out to look somewhat more complicated, but it is useful in practice – though for the moment our prime aim to bring out is something of the structural richness of adjunctions.)

A Galois connection between the posets $(P, \preceq)$, $(Q, \sqsubseteq)$, according to the alternative definition, comprises a pair of functions $f: P \to Q$ and $g: Q \to P$ such that

(i) $f$ and $g$ are monotone,
(ii) $p \preceq g(f(p))$ for all $p \in P$, and
(iii) $f(g(q)) \sqsubseteq q$ for all $q \in Q$.

Since the composition of monotone functions is monotone, (ii) and (iii) are in fact easily seen to be equivalent to

(ii′) if $p \preceq p'$, then $p \preceq p' \preceq g(f(p'))$ and $p \preceq g(f(p)) \preceq g(f(p'))$,

(iii′) if $q \sqsubseteq q'$, then $f(g(q)) \sqsubseteq q \sqsubseteq q'$ and $f(g(q)) \sqsubseteq f(g(q')) \sqsubseteq q'$.

As before, let $\mathcal{P}$ be the category corresponding to the poset $(P, \preceq)$, and recall that there is an arrow $p \to p'$ in $\mathcal{P}$ just when $p \preceq p'$ in the poset $(P, \preceq)$. Likewise for $\mathcal{Q}$ corresponding to $(Q, \sqsubseteq)$. And again as before, note that the monotone functions $f$, $g$ between the posets give rise to functors $F$, $G$ between the corresponding categories. Hence, in particular, the composite monotone function $g \circ f$ gives rise to a functor $G \circ F: \mathcal{P} \to \mathcal{Q}$, and likewise $f \circ g$ gives rise to a functor $F \circ G: \mathcal{Q} \to \mathcal{P}$.

Now, (ii′) corresponds in $\mathcal{P}$ to the claim that the following diagram always commutes:

\[
\begin{array}{ccc}
p & \rightarrow & p' \\
\downarrow & & \downarrow \\
(G \circ F)p & \rightarrow & (G \circ F)p'
\end{array}
\]

(We needn’t label the arrows as in the poset category $\mathcal{P}$ arrows between objects are unique when they exist.)

Dropping the explicit sign for composition of functors for brevity’s sake, let’s define $\eta: 1_\mathcal{P} \to GF$ to be the family of arrows $p \to GFp$, one for each $p \in \mathcal{P}$. Then our commutative diagram version of (ii′) can be revealingly redrawn as follows:

\[
\begin{array}{ccc}
1_\mathcal{P}p & \rightarrow & 1_\mathcal{P}p' \\
\downarrow & & \downarrow \\
GFp & \rightarrow & GFp'
\end{array}
\]

This commutes for all $p, p'$. So applying Defn. 22, this is just to say that $\eta: 1_\mathcal{P} \to GF$ is a natural transformation in $\mathcal{P}$.

Likewise, (iii) and hence (iii′) correspond to the claim that $\varepsilon: FG \to 1_\mathcal{Q}$ is a natural transformation in $\mathcal{Q}$.
(b) So far so good. We have here the initial ingredients for an alternative definition for an adjunction between functors \( F: \mathcal{A} \to \mathcal{B} \) and \( G: \mathcal{B} \to \mathcal{A} \): we require there to be a pair of natural transformations \( \eta: 1_{\mathcal{A}} \to GF \) and \( \varepsilon: FG \to 1_{\mathcal{B}} \).

However, as we’ll see, this isn’t yet enough. But the additional ingredients we want are again suggested by our earlier treatment of Galois connections. Recall from Theorem 107 that if \((f, g)\) is a Galois connection, then we immediately have the key identities

(iv) \( f \circ g \circ f = f \), and
(v) \( g \circ f \circ g = g \).

By (iv), \( fp \preceq (f \circ g \circ f)p \preceq f(p) \) in \( (P, \preceq) \). Hence in \( \mathcal{P} \) the following diagram commutes for each \( p \):

\[
\begin{array}{ccc}
Fp & \xrightarrow{F\eta} & FGp \\
\downarrow & & \downarrow \\
Fp & & Fp
\end{array}
\]

Here, the diagonal arrow is the identity \( 1_{Fp} \). The downward arrow is \( \varepsilon_{Fp} \) (the component of \( \varepsilon \) at \( Fp \)). And the horizontal arrow is \( F\eta_p \) – for recall that by definition the functor \( F \) will indeed send an arrow \( p \to GFp \) to \( Fp \to FGp \).

Or what comes to the same, in the functor category \([\mathcal{P}, \mathcal{Q}]\) this diagram commutes:

\[
\begin{array}{ccc}
F & \xrightarrow{F\eta} & FG \\
\downarrow & & \downarrow \\
F & & F
\end{array}
\]

Here, remember our convention of using double arrows for those arrows which are natural transformations. And remember whiskering(!), discussed in §5.5: the components \( F\eta_p \) assemble into the natural transformation \( F\eta \), and the components \( \varepsilon_{Fp} \) assemble into the natural transformation \( \varepsilon F \).

Exactly similarly, from (v) we infer that the following diagram commutes in \([\mathcal{Q}, \mathcal{P}]\):

\[
\begin{array}{ccc}
G & \xrightarrow{G\eta} & GFG \\
\downarrow & & \downarrow \\
G & & G
\end{array}
\]

(c) And now we can put everything together to give us our new definition:

**Definition 96** (Alternative). Suppose \( \mathcal{A} \) and \( \mathcal{B} \) are categories and \( F: \mathcal{A} \to \mathcal{B} \) and \( G: \mathcal{B} \to \mathcal{A} \) are functors. Then \( F \) is left adjoint to \( G \) and \( G \) is right adjoint to \( F \), notated \( F \dashv G \) and \( G \dashv F \) such that the following triangle identities hold in the functor categories \([\mathcal{A}, \mathcal{B}]\) and \([\mathcal{B}, \mathcal{A}]\) respectively:

\[
\begin{array}{ccc}
F & \xrightarrow{F\eta} & FG \\
\downarrow & & \downarrow \\
F & & F
\end{array}
\]

\[
\begin{array}{ccc}
G & \xrightarrow{G\eta} & GFG \\
\downarrow & & \downarrow \\
G & & G
\end{array}
\]
Note, \( \eta \) and \( \varepsilon \) are standardly called the unit and counit of the adjunction.

It remains to show that Defn. 94 and Defn. 96 are equivalent:

**Theorem 117.** For given functors \( F : \mathcal{A} \to \mathcal{B} \) and \( G : \mathcal{B} \to \mathcal{A}' \), \( F \dashv G \) holds by our original definition iff it holds by the alternative definition.

**Proof (If).** Suppose there are natural transformations \( \eta : 1_{\mathcal{A}} \to GF \) and \( \varepsilon : FG \to 1_{\mathcal{B}} \) for which the triangle identities hold.

Define \( \varphi_{AB} : \mathcal{B}(F(A), B) \to \mathcal{A}(A, G(B)) \) by putting \( \varphi_{AB}(f) = G(f) \circ \eta_A \) for any \( f : F(A) \to B \).

Define \( \psi_{AB} : \mathcal{A}(A, G(B)) \to \mathcal{B}(F(A), B) \) by putting \( \psi_{AB}(g) = \varepsilon_B \circ F(g) \) for any \( g : A \to G(B) \).

Since \( \eta_A : A \to GF(A) \) and \( G(f) : GF(A) \to GB \), their composition \( \varphi_{AB} \) is indeed in \( \mathcal{A}(A, G(B)) \). Likewise \( \psi_{AB}(g) \) is in \( \mathcal{B}(F(A), B) \). So our definitions are in good order.

Keep \( B \) fixed: then the various components of \( \varphi_{AB} \) assemble into a natural transformation \( \varphi_B \) as we vary \( A \). Similarly keep \( A \) fixed, and the various components of \( \varphi_{AB} \) assemble into a natural transformation \( \varphi_A \) as we vary \( B \). So \( \varphi_{AB} \) is a natural transformation from \( \mathcal{B}(F(A), B) \to \mathcal{A}(A, G(B)) \) natural in \( A \in \mathcal{A} \), \( B \in \mathcal{B} \). Similarly \( \psi_{AB} \) is a natural transformation.

Dropping some subscripts, we now show that \( \varphi \) and \( \psi \) are mutual inverses, so are isomorphisms, from which the desired result follows: i.e. \( \mathcal{B}(F(A), B) \cong \mathcal{A}(A, G(B)) \) naturally in \( A \in \mathcal{A} \) and in \( B \in \mathcal{B} \).

Take any \( f : FA \to B \). Then

\[
\psi(\varphi(f)) = \psi(G(f) \circ \eta_A)
\]

by definition of \( \phi \)

\[
= \varepsilon_B \circ F(G(f) \circ \eta_A)
\]

by definition of \( \psi \)

\[
= \varepsilon_B \circ FGf \circ F\eta_A
\]

by functoriality of \( F \)

\[
= f \circ \varepsilon_{FA} \circ F\eta_A
\]

by naturality square for \( \varepsilon \)

\[
= f \circ 1_{FA}
\]

by triangle equality

\[
= f
\]

Hence \( \psi \circ \varphi = 1 \) (note how we did need to appeal to the added triangle equality, not just definitions and the naturality of \( \varepsilon \)). Likewise \( \psi \circ \varphi = 1 \).

**Proof (Only if).** Suppose \( \mathcal{B}(F(A), B) \cong \mathcal{A}(A, G(B)) \) naturally in \( A \in \mathcal{A} \) and in \( B \in \mathcal{B} \). We need to define a unit and counit for the adjunction, and show they satisfy the triangle equalities.

Take the identity map \( 1_{FA} \) in \( \mathcal{B}(FA, FA) \). The natural isomorphism defining the adjunction sends \( 1_{FA} \) to a map \( \eta_A : A \to GF(A) \).

We first show that the components \( \eta_A \) assemble into a natural transformation from \( 1_{\mathcal{A}} \) to \( GF \). So consider the following two diagrams:

\[
\begin{array}{ccc}
FA & \xrightarrow{Ff} & FA' \\
\downarrow 1_{FA} & & \downarrow 1_{FA'} \\
FA & \xrightarrow{Ff} & FA'
\end{array}
\]

\[
\begin{array}{ccc}
A & \xrightarrow{f} & A' \\
\downarrow \eta_A & & \downarrow \eta_{A'} \\
GFA & \xrightarrow{GFf} & GFA'
\end{array}
\]

Trivially, the diagram on the left commutes for all \( f : A \to A' \). \( Ff \circ 1_{FA} = 1_{FA'} \circ Ff \).

Transposition must evidently preserve identities. So \( Ff \circ 1_{FA} = 1_{FA'} \circ Ff \). But by the first of the naturality requirements in §18.5, \( Ff \circ 1_{FA} = GFf \circ 1_{FA} = GFf \circ \eta_A \). And by the other naturality requirement, \( 1_{FA'} \circ Ff = \eta_{A'} \circ Ff = \eta_{A'} \circ f \). So we have \( GFf \circ \eta_A = \)
\( \eta_{A'} \circ f \) and the diagram on the right commutes for all \( f \). Hence the components \( \eta_A \) do indeed assemble into a natural transformation.

Similarly the same natural isomorphism in the opposite direction sends \( 1_{GB} \) to its transpose \( \varepsilon_B : FG(B) \to B \), and the components \( \varepsilon_B \) assemble into a natural transformation from \( FG \) to \( 1_B \).

Now consider these three diagrams:

![Diagram](image)

The diagram on the left trivially commutes. Transpose it via the natural isomorphism that defines the adjunction and use the naturality requirements again; we find that the middle diagram must also commute. Using the fact the middle diagram (like the first) in fact commutes for all \( A \), and deleting the redundant corner, the final diagram must commute – which gives us one of the triangle identities. The other identity we get dually.

We are done. But although the strategies for proving the equivalence of our definitions are entirely straightforward, checking the details was a bit tedious and required keeping our wits about us. So let’s take a breather before resuming the general exploration of adjunctions in the next chapter.


