1
Category Theory

Elementary category theory is concerned with categories, functors, and natural transformations. As described in Mac Lane (1998):

“category” has been defined in order to be able to define “functor” and “functor” has been defined in order to be able to define “natural transformation.”

We shall consider each notion in turn, whilst simultaneously preparing the grounds for string diagrams to be introduced in Chapter 2.

1.1 Categories

A *category* consists of objects and arrows between objects. The letters $\mathcal{C}$, $\mathcal{D}$, … range over categories, and the uppercase letters A,B, … over objects. We write $A : \mathcal{C}$ to express that A is an object of the category $\mathcal{C}$. Lowercase letters f, g, … range over arrows, and we write $f : A \to B : \mathcal{C}$ to express that f is an arrow from A to B in the category $\mathcal{C}$. The object A is called the *source* of f and B its *target*. If $\mathcal{C}$ is obvious from the context, we abbreviate $f : A \to B : \mathcal{C}$ by $f : A \to B$.

For every object $A : \mathcal{C}$ there is an arrow $id_A : A \to A$, called the *identity*. Two arrows can be *composed* if their types match: if $f : A \to B$ and $g : B \to C$, then $g \cdot f : A \to C$ (pronounced “g after f”). We require composition to be unital and associative, with identity as its neutral element:

$$id_B \cdot f = f = f \cdot id_A, \tag{1.1a}$$

$$(h \cdot g) \cdot f = h \cdot (g \cdot f). \tag{1.1b}$$

1.1.1 Examples of Categories. To make the abstract notion of category more tangible, we introduce several examples, many of which will accompany us throughout the monograph. We begin with two trivial but useful categories:

Example 1.1 (**0** and **1**)**.** There is a category, denoted **0**, with no objects and no arrows. There is also a category **1**, with one object and one arrow, the identity on the single object. □

Categories can be seen as generalizations of possibly more familiar mathematical objects.

Example 1.2 (Monoids and Preorders)**.** Two extreme classes of categories are worth singling out.

A monoid $(A, e, \bullet)$ can be seen as a category that has exactly one object. The arrows are the elements of the monoid: e serves as the identity and $\bullet$ as composition.

A preorder $(A, \leqslant)$ can be seen as a category with at most one arrow between any two objects, which are the elements of the preorder. There exists an arrow of type $a \to b$ if and only if $a \leqslant b$; reflexivity ensures the existence of identities and transitivity the existence of composites. □

A category is often identified with its collection of objects: we loosely say that **Set** is the category of sets. However, equally if not more important are the arrows of a category. So, **Set** is really the category of sets and total functions. There is also **Rel**, the category of sets and relations.

Remark 1.3 (Preservation and Reflection of Structure)**.** An arrow*preserves* structure if features of the source allow us to deduce features of the target. For example, if $h : (A, 0, +) \to (B, 1, \times)$ is a monoid homomorphism, and $a + a' = 0$ holds in the source monoid, then $h\,a \times h\,a' = 1$ holds in the target monoid. This is exactly the motivation for homomorphisms between algebraic structures: they *preserve equations*.

An arrow *reflects* structure if we can infer properties of the source from properties of the target. Notice the backward direction of travel.

To illustrate this, let us first establish some useful notation that we need time and again. For a function $f : A \to B$ there is a *direct image function* taking subsets of A to subsets of B:

$$f^{\blacktriangleright}\, X := \{\, y \in B \mid \exists x \in X \,.\, f\,x = y \,\}.$$

There is also an *inverse image function*, mapping subsets in the opposite direction:

$$f^{\blacktriangleleft}\, Y := \{\, x \in A \mid \exists y \in Y \,.\, f\,x = y \,\}.$$

With this notation in place, if $h : A \to B$ is a continuous map of topological spaces, $Y \subseteq B$ being an open subset of B implies $f^{\blacktriangleleft}\, Y \subseteq A$ is an open set in A. So, structure in the target implies structure in the source, and these topological arrows reflect structure. $\square$

Example 1.4 (Sets and Structures)**.** Many examples of categories used in practice are sets with additional structure, and functions that preserve or reflect this structure.

Sets with additional structure include monoids, groups, preorders, lattices, graphs, and so on. In each of these cases the arrows are structure-preserving maps. For example, **Mon** is the category of monoids and monoid homomorphisms. Notice the difference as compared to Example 1.2. There we considered a single monoid; here we consider the collection of all monoids and homomorphisms between them. Likewise, we can form the category **Pre**, whose objects are preorders and whose arrows are monotone or order-preserving functions.

Further examples include **Bool**, **Sup**, and **CompLat**, which are respectively the categories of Boolean lattices, complete join-semilattices, and complete lattices, with homomorphisms preserving the algebraic structure. Note that, although every complete join-semilattice is automatically a lattice, the categories **Sup** and **CompLat** are different, as the arrows preserve different structure.

As well as these examples with structure-preserving maps, there are examples where the arrows *reflect* structure, such as the categories **Top** and **Met** of topological spaces and metric spaces, with continuous maps as arrows. $\square$

The following category, which will accompany us as a running example, is perhaps slightly more unusual.

Example 1.5 (Category of Actions)**.** Let $(M, e, \bullet)$ be a fixed monoid. The objects of the category M-**Act** are pairs $(A, \triangleleft)$, where A is a set and $(\triangleleft) : M \times A \to A$ is an operation that respects the monoid structure:

$$e \triangleleft a = a, \tag{1.2a}$$
$$(m \bullet n) \triangleleft a = m \triangleleft (n \triangleleft a). \tag{1.2b}$$

The operation is also called a *left action of* M. An arrow $f : (A, \triangleleft) \to (B, \blacktriangleleft)$ in M-**Act** is a function of type $A \to B$ that preserves actions:

$$f\,(m \triangleleft a) = m \blacktriangleleft f\,a, \tag{1.3}$$

also known as an *equivariant function*. $\square$

There are many ways of constructing new categories from old, as we will see in later sections. For now, we consider three useful cases.

Definition 1.6 (Subcategories)**.** A *subcategory* $\mathcal{S}$ of a category $\mathcal{C}$ is a collection of some of the objects and some of the arrows of $\mathcal{C}$, such that identity and composition are preserved to ensure $\mathcal{S}$ constitutes a valid category. For example, **Set** is a subcategory of **Rel** as every function is a binary relation. Commutative monoids **CMon** and commutative, idempotent monoids **CIMon** form subcategories of **Mon**.

In a *full subcategory* the collection of arrows is maximal: if $f : A \to B : \mathcal{C}$ and $A, B : \mathcal{S}$, then $f : A \to B : \mathcal{S}$. The category **Fin** of finite sets and total functions is a full subcategory of **Set**. □

Definition 1.7 (Opposite Categories)**.** For any category $\mathcal{C}$ we can consider its *opposite category* $\mathcal{C}^{op}$. This has the same objects as $\mathcal{C}$, but an arrow of type $A \to B$ in $\mathcal{C}^{op}$ is an arrow of type $B \to A$ in $\mathcal{C}$. Identities in $\mathcal{C}^{op}$ are as in $\mathcal{C}$, and composition in $\mathcal{C}^{op}$ is given by forming the reverse composite in $\mathcal{C}$. The process of swapping source and target is purely bureaucratic; it does not do anything to the arrows. □

Definition 1.8 (Product Categories)**.** For any pair of categories $\mathcal{C}$ and $\mathcal{D}$ we can form their product $\mathcal{C} \times \mathcal{D}$. An object of the *product category* is a pair of objects (A, B) with $A : \mathcal{C}$ and $B : \mathcal{D}$; an arrow of type $(A, C) \to (B, D) : \mathcal{C} \times \mathcal{D}$ is a pair of arrows (f, g) with $f : A \to B : \mathcal{C}$ and $g : C \to D : \mathcal{D}$. Identity and composition are defined componentwise,

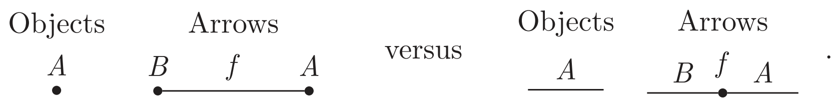
$$id_{(A,B)} \coloneqq (id_A, id_B), \tag{1.4a}$$

$$(g_1, g_2) \cdot (f_1, f_2) \coloneqq (g_1 \cdot f_1, g_2 \cdot f_2), \tag{1.4b}$$

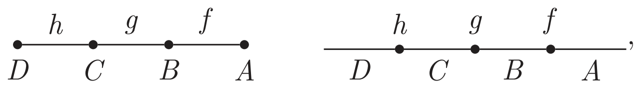
in terms of identity and composition of the underlying categories. □

1.1.2 Graphical Representation of Objects and Arrows. We have noted in the prologue that notation matters, so a brief discussion of the syntax is certainly not amiss. Composition of arrows is a binary operation. Applications of binary operations or 2-ary functions are variably written prefix $op\ a\ b$, infix $a\ op\ b$, or postfix $a\ b\ op$, often with additional syntactic ornaments such as parentheses or commas. We have opted to write composition infix as $g \cdot f$. Why? Infix notation has a distinct advantage over the alternatives when expressions are nested as in $h \cdot g \cdot f$. At the outset, nested infix expressions are ambiguous, consider for example $a - b - c$. Do we intend to say $(a - b) - c$ or $a - (b - c)$? Convention has it that $a - b - c$ is resolved to $(a - b) - c$. For composition, however, the problem of ambiguity dissolves into thin air as composition is associative (1.1b). Here a bug has been turned into a feature: in calculations we do not have to invoke the associative law explicitly; it is built into the notation. By contrast, say we wrote composition prefix; then we are forced to express $h \cdot g \cdot f$ as either $comp\,(h, comp\,(g, f))$ or as $comp\,(comp\,(h, g), f)$. The syntax forces us to make an unwelcome distinction.

Composition of arrows in categories lends itself well to a graphical representation using vertices and edges. There are basically two options: objects can be represented by vertices, and arrows by edges between them, or vice versa:



The two types of diagrams are related by *topological* or *Poincaré duality*, where vertices become edges and edges become vertices. There are many variations of the two schemes. Vertices are often drawn as boxes or are not drawn at all, being replaced by their labels. Edges are often directed to allow for a more flexible arrangement of vertices. We avoid arrowheads by agreeing that the flow is from *right to left*. This choice blends well with the symbolic notation in that the graphical direction of composition,



follows the direction in the term $h \cdot g \cdot f$. For reasons of consistency, we should also write the types backwards: if $g : C \leftarrow B$ and $f : B \leftarrow A$, then $g \cdot f : C \leftarrow A$. We stick to the customary notation, however, and use right-to-left types only for emphasis. (An alternative is to change the order of composition: forward composition $f \mathbin{;} g \mathbin{;} h$ blends well with left-to-right types. We use both forward and backward composition.)

Like the symbolic notation, the diagrammatic representations have associativity (1.1b) built in, as we are simply threading beads on a necklace. We can further obviate the need for invoking unitality (1.1a) explicitly by agreeing that the identity arrow on an object A is represented by the rendition of A. The same convention is also used in symbolic notation: the identity on A is often written $A : A \rightarrow A$. A distinctive advantage of diagrams over terms is that they add vital type information. For a monoid $a \cdot b$ is always defined. However, as composition is in general partial, our notation should prevent us from joining arrows together incorrectly.

We have two graphical representations to choose from. But which one to pick? Different communities have different preferences: theoreticians seem to prefer the diagrams on the left above (e.g. as parts of commuting diagrams; see Section 1.7.2), while hardware people seem to prefer the diagrams on the right (e.g. in the form of circuit diagrams). We favor the latter notation for reasons that will become clear later.

1.2 Properties of Arrows

We now consider categorical generalizations of injective, surjective, and bijective functions.

1.2.1 Mono and Epi Arrows. An arrow $f : A \to B$ is called *mono* if it is *left-cancelable*:

$$f \cdot x_1 = f \cdot x_2 \implies x_1 = x_2, \tag{1.5a}$$

for all objects X and all arrows $x_1, x_2 : X \to A$. In **Set** these are the injective functions. Dually, an arrow $f : A \to B$ is called *epi* if it is *right-cancelable*:

$$x_1 \cdot f = x_2 \cdot f \implies x_1 = x_2, \tag{1.5b}$$

for all objects X and all arrows $x_1, x_2 : B \to X$. In **Set** these are the surjective functions. The inverse directions of the cancellation properties (1.5a) and (1.5b) are Leibniz's context rules,

$$x_1 = x_2 \implies f \cdot x_1 = f \cdot x_2, \tag{1.5c}$$
$$x_1 = x_2 \implies x_1 \cdot f = x_2 \cdot f, \tag{1.5d}$$

so implications (1.5a) and (1.5b) can both be strengthened to equivalences.

1.2.2 Split Mono and Split Epi Arrows. For an arrow $f : A \to B$, a *post-inverse* of f is an arrow $k : A \leftarrow B$ such that

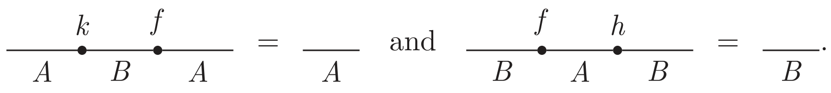
$$k \cdot f = id_A.$$

In this case, f is referred to as a *split mono*. Dually, a *pre-inverse* of f is an arrow $h : A \leftarrow B$ such that

$$f \cdot h = id_B.$$

Such an f is referred to as a *split epi*.

In pictures, these are arrows that annihilate each other if they touch in the right order:



Observe that the identity arrows are rendered by edges.

In **Set**, almost every injective function has a post-inverse. The only exceptions are functions of type $\emptyset \to A$ with $A \neq \emptyset$, simply because there are no functions of type $A \to \emptyset$. However, every surjective function has a pre-inverse.

Occasionally it is useful to reinterpret categorical notions using set-theoretic spectacles. If we partially apply the composition operator, $- \cdot f$ or $g \cdot -$, we obtain maps over collections of arrows. Using these maps we can reinterpret the notions of mono and epi. Property (1.5a) captures that $f \cdot -$ is injective; likewise, (1.5b) states that $- \cdot f$ is injective:

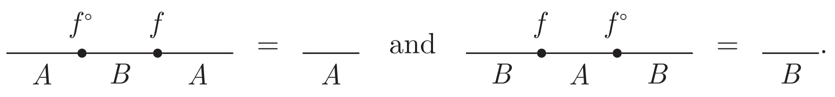
$$(f \cdot -)\, x_1 = (f \cdot -)\, x_2 \implies x_1 = x_2,$$
$$(- \cdot f)\, x_1 = (- \cdot f)\, x_2 \implies x_1 = x_2.$$

While cancellation properties are related to injectivity, existence of a pre- or a post-inverse are related to surjectivity:

$$g \cdot - \text{ injective} \iff g \text{ mono}, \tag{1.6a}$$
$$- \cdot f \text{ injective} \iff f \text{ epi}, \tag{1.6b}$$
$$g \cdot - \text{ surjective} \iff g \text{ split epi}, \tag{1.6c}$$
$$- \cdot f \text{ surjective} \iff f \text{ split mono}. \tag{1.6d}$$

The proofs of (1.6c) and (1.6d) are relegated to Exercise 1.8. The preceding list of equivalences partially explains why there are four different notions, rather than only two as in **Set**.

1.2.3 Isomorphisms. Two objects A and B are isomorphic, written $A \cong B$, if there is a pair of functions $f : A \to B$ and $g : A \leftarrow B$ such that $f \cdot g = id_B$ and $id_A = g \cdot f$. If an arrow $f : A \to B$ has both a pre- and a post-inverse, then they coincide, and we denote them f°. In this case f is an *isomorphism*, *iso* for short, with *inverse* f°, written $f : A \cong B : f^\circ$:



In **Set**, the isos are exactly the bijective functions.

The relation $\cong$ is an equivalence relation: it is reflexive, symmetric, and transitive. Furthermore, it is compatible with most constructions on objects. Reflexivity is established by identity arrows:

$$id_A : A \cong A : id_A.$$

Symmetry is shown by exchanging the isomorphisms:

$$f : A \cong B : f^\circ \implies f^\circ : B \cong A : f.$$

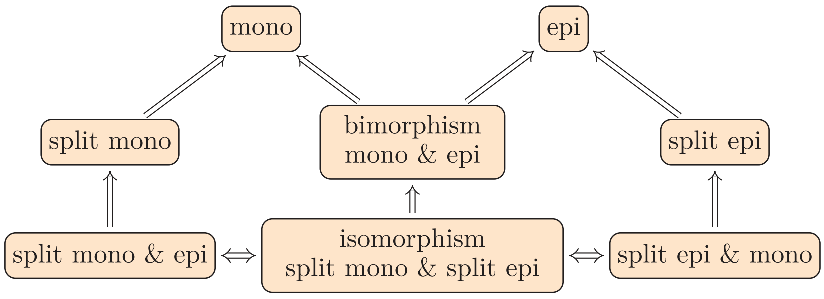
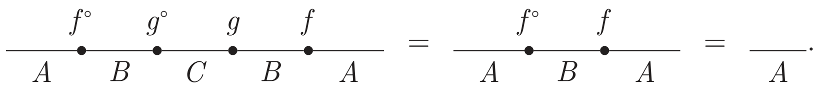


Figure 1.1 Properties of arrows.

Transitivity is established by suitably composing the witnesses:

$$f : A \cong B : f^\circ \;\wedge\; g : B \cong C : g^\circ \quad\Longrightarrow\quad g \cdot f : A \cong C : f^\circ \cdot g^\circ .$$

To prove that the composites are isomorphisms, we first annihilate the inner arrows and then the outer ones.



The proof for the reverse direction is entirely analogous.

Figure 1.1 relates the various properties of arrows – an *isomorphism* is an arrow that is both split mono and split epi; an arrow that is both mono and epi is called a *bimorphism*. The identity has all the properties, and all the properties are preserved by composition. Exercise 1.11 asks you to establish the relations and to show that the inclusions are proper.

The attentive reader may have noted that categorical concepts come in pairs. An epi in $\mathcal{C}$ is a mono in $\mathcal{C}^{\mathsf{op}}$; a split epi in $\mathcal{C}$ is a split mono in $\mathcal{C}^{\mathsf{op}}$; the concept of an iso is self-dual; an iso in $\mathcal{C}$ is an iso in $\mathcal{C}^{\mathsf{op}}$. Duality means that we get two concepts for the price of one. The next section provides further evidence for the economy of expression afforded by duality.

1.3 Thinking in Terms of Arrows

A category consists of objects and arrows. However, these entities are not treated on an equal footing: category theory puts the conceptual emphasis on arrows. Indeed, to master the subject one has to learn to think in terms of arrows. To illustrate, let us define some additional infrastructure: initial and final objects, products and coproducts, and exponentials. In each case, the defined object is characterized in terms of its relationship to other objects. In a sense, category theory is the most social of all mathematical foundations.

1.3.1 Initial and Final Objects. Let $\mathcal{C}$ be a category. An object $0 : \mathcal{C}$ is called *initial in* $\mathcal{C}$ if, for each object $A : \mathcal{C}$, there is exactly one arrow from the initial object 0 to A. This property is referred to as the *universal property* of initial objects.

Dually, $1 : \mathcal{C}$ is a *final* or *terminal object in* $\mathcal{C}$ if it satisfies the *universal property* that, for each object $A : \mathcal{C}$, there is a unique arrow from A to 1. A final object in $\mathcal{C}$ is an initial object in $\mathcal{C}^{\mathsf{op}}$.

An object that is simultaneously initial and final in $\mathcal{C}$ is called a *zero object*.

Example 1.9 (Preorders)**.** An initial object in a preorder category is a least element. Dually, a final object is a greatest element. If the preorder is a partial order, meaning the relation $\leqslant$ is also antisymmetric, then initial and final objects are unique. □

Example 1.10 (Sets and Structures)**.** In **Set**, the empty set is initial and *any* singleton set is final. In **Mon**, the singleton monoid $(\{()\}, (), \bullet)$ with $() \bullet () = ()$ is both initial and final, as homomorphisms have to preserve the neutral element: the singleton monoid is a zero object. □

The examples demonstrate that, in general, initial and final objects are not unique. They are, however, *unique up to a unique isomorphism*. If A and B are both initial, then there are unique arrows of type $A \to B$ and $B \to A$, whose compositions are necessarily identities.

1.3.2 Products and Coproducts. A *product* of two objects B_1 and B_2 consists of an object written as $B_1 \times B_2$ and a pair of *projection arrows*:

$$outl : B_1 \times B_2 \to B_1 \quad \text{and} \quad outr : B_1 \times B_2 \to B_2.$$

These three entities have to satisfy the following *universal property*: for each object A and for each pair of arrows $f_1 : A \to B_1$ and $f_2 : A \to B_2$, there exists an arrow $f_1 \vartriangle f_2 : A \to B_1 \times B_2$ (pronounced "f_1 *split* f_2") such that

$$f_1 = outl \cdot g \;\wedge\; f_2 = outr \cdot g \iff f_1 \vartriangle f_2 = g, \tag{1.7}$$

for all $g : A \to B_1 \times B_2$. The equivalence captures the existence of an arrow satisfying the property on the left and furthermore states that $f_1 \vartriangle f_2$ is the *unique* such arrow. The following commutative diagram (see Section 1.7.2) summarizes the type information:

$$\begin{tikzcd}
B_1 & & \\
& B_1 \times B_2 \arrow[ul, "outl"'] \arrow[dl, "outr"] & A \arrow[l, dotted, "f_1 \vartriangle f_2"'] \arrow[ull, bend right, "f_1"'] \arrow[dll, bend left, "f_2"] \cdot \\
B_2 & &
\end{tikzcd}$$

The dotted arrow indicates that $f_1 \vartriangle f_2$ is the unique arrow from A to $B_1 \times B_2$ that makes the diagram commute. Informally, the universal property states that for anything that "looks like" a product there is a unique arrow that factorizes the look-alike product in terms of a "real" product. Section 5.2 makes the notion of a universal construction precise.

The construction of products dualizes to coproducts, which are products in the opposite category. The *coproduct* of two objects A_1 and A_2 consists of an object written as $A_1 + A_2$ and a pair of *injection arrows*:

$$inl : A_1 \to A_1 + A_2 \quad \text{and} \quad inr : A_2 \to A_1 + A_2.$$

These three entities have to satisfy the following *universal property*: for each object B and for each pair of arrows $g_1 : A_1 \to B$ and $g_2 : A_2 \to B$, there exists an arrow $g_1 \triangledown g_2 : A_1 + A_2 \to B$ (pronounced "g_1 *join* g_2") such that

$$f = g_1 \triangledown g_2 \iff f \cdot inl = g_1 \;\wedge\; f \cdot inr = g_2, \tag{1.8}$$

for all $f : A_1 + A_2 \to B$. Reversing the arrows in the previous product diagram, we obtain the corresponding diagram for coproducts:

$$\begin{tikzcd}
A_1 \arrow[dr, "inl"'] \arrow[drr, bend left, "g_1"] & & \\
& A_1 + A_2 \arrow[r, dotted, "g_1 \triangledown g_2"] & B \cdot \\
A_2 \arrow[ur, "inr"] \arrow[urr, bend right, "g_2"'] & &
\end{tikzcd}$$

Remark 1.11 (Bigger Products and Coproducts)**.** We have introduced binary products. Clearly, we can also define ternary products, with three projection arrows, and a corresponding universal property. These can be built by nesting binary products, with the order of composition unimportant, as