A Gentle Introduction to Categorical Logic and Type Theory

Eric Schmid
DRAFT: March 6, 2025
Version 2

Contents

Preface

Category theory provides a unifying language to study mathematical structures in terms of their relationships (morphisms) rather than just their elements. *Categorical logic* uses category theory to study logical systems, offering a structural perspective on syntax and semantics, while *type theory* provides a framework connecting logic with computation via the Curry–Howard correspondence. These notes aim to introduce the basics of category theory and show how categories provide semantics for logic and type theories. We assume the reader has some familiarity with elementary set theory, algebra, and logic (propositional and first-order), but we proceed from first principles and include many examples and exercises for self-study.

These notes draw on several texts. The structure of presentation follows Abramsky and Tzevelekos's *Introduction to Categories and Categorical Logic*[1], while expanding on topics with material from Awodey's lecture notes on categorical logic[2] and Jacobs's comprehensive textbook *Categorical Logic and Type Theory*[3].

Chapter 1 introduces basic category theory: the language of objects and arrows, functors between categories, natural transformations, and fundamental categorical constructions (products, coproducts, limits, etc.). We emphasize how many familiar mathematical concepts can be reformulated in terms of arrows rather than elements, leading to a shift in perspective that underlies categorical logic.

Chapter 2 covers propositional logic and algebraic theories from a categorical perspective. We explain how algebraic structures (like groups, rings, etc.) can be described as categories with finite products (Lawvere theories), and how their models are functors to

[1] Samson Abramsky and Nikos Tzevelekos, "Introduction to Categories and Categorical Logic," arXiv:1102.1313 (2011), which provides a succinct introduction to category theory and its connections to logic.

[2] Steve Awodey (with input from Andrej Bauer), *Categorical Logic*, Autumn School on Proof and Computation, Fischbachau 2022 and the draft *Introduction to Categorical Logic* (September 15, 2024). These texts develop the idea of algebraic theories, propositional logic, and Stone duality in a categorical framework.

[3] Bart Jacobs, *Categorical Logic and Type Theory*, Studies in Logic and the Foundations of Mathematics 141, Elsevier, 1999. Jacobs's book systematically develops logic and type theory using category theory (in particular, fibrations) and serves as a reference for advanced topics beyond the scope of these introductory notes.

Set. We prove soundness and completeness for equational logic categorically. We then treat propositional logic: Boolean algebras and their dual Stone spaces provide semantics for classical propositional calculus, culminating in Stone's duality theorem. We also discuss intuitionistic propositional logic via Heyting algebras, introducing *frames* and *locales* as categorical generalizations of topological spaces.

Chapter 3 deals with first-order logic in categorical terms. We develop the notion of a *cartesian* or *finite limit logic* (also called coherent or regular logic for certain fragments) and show how quantifiers can be characterized as adjoint functors. The concept of a *classifying category* (or classifying topos in more advanced settings) for a first-order theory is introduced, providing a categorical semantics where models of the theory correspond to functors from this category. We define key properties of categories that capture logical axioms: regular categories (for $\exists$), coherent categories (for $\vee$), Heyting categories (for intuitionistic logic), and Boolean categories (for classical logic). We outline the internal logic of categories and give a glimpse of *Kripke–Joyal semantics* for logical formulas in a category (as used in topos theory).

Chapter 4 introduces type theory and its categorical semantics. We start with the simply-typed λ-calculus as a formal system of types and terms. We define *Cartesian closed categories* (CCC) and explain the Curry–Howard–Lambek correspondence: how CCCs are equivalent (as models) to the simply-typed lambda calculus. In particular, we show how a CCC provides a sound and complete semantics for propositional logic with implication (and how adding product types corresponds to conjunction). We then extend to dependent types and discuss how categories with additional structure (such as *locally Cartesian closed categories* and *categories with families*) provide semantics for dependent type theories (which correspond to first-order logic internally). Finally, we touch on polymorphism and universe types, connecting to advanced type theory: a *universe* in a category is introduced as an object that can be thought of as a type of all (small) types, allowing the interpretation of type-theoretic universes categorically.

Throughout, we include full proofs of propositions and theorems to make this text self-contained. In places where proofs are beyond the scope of an introduction (for example, Gödel's completeness theorem for first-order logic or a full proof of Stone's theorem), we give sketches or references for further reading.

It is our hope that this expanded textbook will serve students and researchers from both computer science and mathematics who are interested in the beautiful connections between category theory, logic, and type theory. The exercises should help the reader gain a working proficiency in these concepts. We encourage attempting the exercises, as many important ideas appear there.

Acknowledgments. We thank the authors of the original notes and textbooks. All mistakes or omissions in these notes are our own.

Chapter 1

Basic Category Theory

1.1 From Elements to Arrows: The Categorical Approach

Mathematics traditionally speaks in terms of elements: for example, we say a function $f : X \to Y$ is *injective* if "for all $x_1, x_2 \in X$, $f(x_1) = f(x_2)$ implies $x_1 = x_2$." In category theory, we shift focus from elements to **arrows**. Instead of quantifying over elements, we express properties via composition of arrows. For instance, the injectivity of f can be captured by the condition: whenever two arrows $g, h : Z \to X$ satisfy $f \circ g = f \circ h$, then $g = h$. In categorical terms, this says $f : X \to Y$ is a **monomorphism** if $f \circ g = f \circ h \implies g = h$. Dually, f is a **epimorphism** if $g \circ f = h \circ f \implies g = h$ (cancellable on the right).

This arrow-centric viewpoint reveals deep structural similarities across different mathematical contexts. A classic example: the concept of a "homomorphism" (structure-preserving map) between algebraic structures can be seen as a special case of a categorical *morphism* (arrow). By abstracting properties of these arrows (composition, identity, etc.), category theory provides a unifying framework to discuss disparate areas of math in a common language.

In category theory, we do not require that arrows are functions or that objects are sets; they can be any abstract entities as long as they satisfy certain axioms. This level of generality is powerful: for example, a partially ordered set (poset) can be viewed as a special category (with at most one arrow between any two objects), and a group can be viewed as a category with a single object (all group elements being arrows from that object to itself). These perspectives might seem unusual at first, but they allow us to apply categorical reasoning (like "universal properties") to a wide range of situations.

In summary, category theory encourages thinking about mathematical structures in terms of their relationships (morphisms) rather than just their constituents. This shift will

enable us to later describe logical systems and type systems in terms of categories, where logical formulas and types correspond to objects, and proofs and programs correspond to arrows.

1.2 Definition of Categories and Basic Examples

We begin with the formal definition of a category. Informally, a category $\mathcal{C}$ consists of a collection of objects and, for each pair of objects (A, B), a collection of arrows (also called *morphisms*) from A to B. These arrows can be composed, and there are identity arrows for each object, such that composition is associative and identities act as neutral elements. Let us state this precisely.

Definition 1.1 (Category)**.** A **category** $\mathcal{C}$ consists of:

- A class $\mathrm{Ob}(\mathcal{C})$ of **objects**, denoted $A, B, C, \ldots$
- For every ordered pair of objects (A, B), a class $\mathcal{C}(A, B)$ of **morphisms** (or **arrows**) from A to B. If f is a morphism in $\mathcal{C}(A, B)$, we write $f : A \to B$.

These data satisfy the following axioms:

- (**Composition**) For any $f : A \to B$ and $g : B \to C$, there is a morphism $g \circ f : A \to C$ called their **composite**. Composition is associative: if $f : A \to B$, $g : B \to C$, and $h : C \to D$, then $h \circ (g \circ f) = (h \circ g) \circ f$.
- (**Identity**) For every object A, there is a morphism $\mathrm{id}_A : A \to A$, called the **identity** on A, such that for every $f : A \to B$, we have $f \circ \mathrm{id}_A = f$, and for every $g : C \to A$, $\mathrm{id}_A \circ g = g$. In other words, id_A acts as a two-sided identity for composition.

We often denote the class of morphisms from A to B as $\hom_{\mathcal{C}}(A, B)$ or simply $\mathcal{C}(A, B)$. If a morphism $f : A \to B$ is an isomorphism (defined below), we may write $A \cong B$.

Proof. This definition is standard. Note that we allow the collections of objects and morphisms to be proper classes (not necessarily sets), but in practice many categories of interest are *locally small*, meaning $\mathcal{C}(A, B)$ is a set for all A, B. □

In the definition above, we used the word "class" to allow for the possibility that the collection of all objects or morphisms might be too large to be a set (for example, the category of all sets has a proper class of objects). A category is called **small** if its objects and morphisms form sets (rather than proper classes), and **locally small** if for each A, B, the hom-class $\mathcal{C}(A, B)$ is a set. Unless stated otherwise, we will not worry about set-theoretic size issues and will assume categories are locally small.

Let us unravel the definition with some immediate consequences and simple examples:

- Composition being associative means that given a composable triple of arrows $A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D$, it doesn't matter how we parenthesize the composite $h \circ g \circ f$. We can thus unambiguously write $h \circ g \circ f : A \to D$.

- Identities are unique for each object (because if $u : A \to A$ is any morphism with the identity property, $u = u \circ \mathrm{id}_A = \mathrm{id}_A \circ u = \mathrm{id}_A$). Often, we simply write 1_A for id_A when context is clear.

- The definition does not require that $\mathrm{Ob}(\mathcal{C})$ or $\mathcal{C}(A, B)$ are non-empty. It is perfectly valid (though not very interesting) to have a category with no objects (and hence no arrows), or a category with objects but no arrows between some pairs of objects. For instance, the **empty category** has no objects at all, and the **discrete category** on a set of objects has only identity arrows and no others.

Now we present a variety of examples. These illustrate how diverse categories can be, and how familiar structures fit into this abstract framework.

Example 1.2 (Sets and Functions)**.** Perhaps the most fundamental example is the category **Set**, whose objects are all (small) sets and whose morphisms are all functions between sets. Composition is the usual composition of functions, and identity morphisms are identity functions. It is straightforward to verify that **Set** satisfies the category axioms. This category plays a special role as the typical setting for the semantics of many theories (since models of theories are often sets with structure). We will often denote it by **Set**.

Other categories of mathematical structures and structure-preserving functions include:
- **Grp**: objects are groups, morphisms are group homomorphisms.
- **Mon**: monoids and monoid homomorphisms.
- **Ring**: rings (with unity) and ring homomorphisms.
- **Top**: topological spaces and continuous functions.
- $\mathbf{Vect}_k$: vector spaces over a field k and linear transformations.

In general, any class of structures with a suitable notion of homomorphism forms a category. Composition of morphisms is just composition of functions that respect the structures, and identities are the identity functions.

Example 1.3 (Posets as Categories)**.** A **partially ordered set** (poset) $(P, \leq)$ can be viewed as a category $\mathcal{C}$ where:

- The objects are the elements of P.
- There is a unique morphism $a \to b$ in $\mathcal{C}$ if and only if $a \leq b$ in the poset, and no morphism otherwise.

In this category interpretation, composition corresponds to transitivity of the order, and identity morphisms correspond to reflexivity ($a \leq a$). Associativity holds because if $a \leq b$ and $b \leq c$ and $c \leq d$, then $a \leq d$ (transitivity handles any way of associating the inequalities).

It's easy to see that verifying the category axioms reduces to checking that $\leq$ is reflexive and transitive (which is given). Anti-symmetry (the condition $a \leq b$ and $b \leq a$ implies $a = b$) means that the only isomorphisms in this category are identity morphisms; such a category is called **skeletal** (isomorphic objects are equal).

As a specific instance, consider any topological space X. Its open sets $O(X)$ form a poset under inclusion ($U \leq V$ iff $U \subseteq V$). Thus $O(X)$ can be regarded as a category: an arrow $U \to V$ exists iff $U \subseteq V$. This example will be important later when we discuss *frames* and *locales* in logic (since $O(X)$ is a frame, an example of a complete Heyting algebra).

Example 1.4 (Groups and Monoids as One-Object Categories)**.** Any **group** $(G, \cdot, e)$ can be seen as a category with a single object $*$. Define $\mathrm{Ob}(\mathcal{C}) = \{*\}$. Let $\mathcal{C}(*, *) = G$, i.e. each element $g \in G$ is considered a morphism $* \to *$. Composition of morphisms in $\mathcal{C}$ is given by the group operation: the composite of $f : * \to *$ and $g : * \to *$ is defined to be $g \circ f := g \cdot f$ (note the order: first f, then g). The group identity e serves as the identity morphism id_*. The group axioms (associativity of $\cdot$ and identity e) exactly ensure the category axioms hold. Every arrow in this category is invertible (with f^{-1} as the inverse of f), reflecting the fact that all morphisms are isomorphisms.

More generally, any **monoid** (a set with an associative binary operation and a unit, but not necessarily inverses) can be viewed as a category with one object. A monoid element m is a morphism $* \to *$; composition is given by the monoid operation $g \circ f = g \cdot f$. The lack of inverses in a monoid means that some arrows in the category will not have two-sided inverses (so it's not a "groupoid", which would be the category analog of group where every arrow is invertible).

This perspective "a category with one object is a monoid" is a correspondence: category theorists thus see monoids as very simple categories. It also suggests that studying properties of monoids can be done with categorical language if desired. For instance, a monoid homomorphism $h : M \to N$ is the same as a functor between the corresponding

one-object categories (more on functors soon).

Example 1.5 (The Zero and Unit Categories)**.** The **empty category** 0 has no objects (and hence no morphisms). It vacuously satisfies the category axioms. The empty category is the initial object in the category of (small) categories, meaning there is a unique functor from 0 to any category $\mathcal{C}$ (since to define a functor you have to specify images of objects and morphisms, but 0 has none to specify, so there's exactly one choice).

The **unit category** 1 (also sometimes denoted **1** or $*$) has a single object $*$ and only its identity morphism 1_*. In 1, for the single object $*$, $\hom(*,*) = \{1_*\}$. This is indeed a category (trivially satisfying axioms). The unit category serves as a terminal object in the category of categories: there is a unique functor from any category $\mathcal{C}$ to 1 (by sending all objects to $*$ and all morphisms to 1_*). We will later see that terminal objects in categories are an important notion and that 1 is a terminal category.

Example 1.6 (Morphisms as Generalized Relations)**.** There are interesting categories where morphisms are *not* functions but more general relations. For instance, **Rel** is a category where objects are sets and a morphism $R : X \to Y$ is a **binary relation** $R \subseteq X \times Y$. Composition of relations $R : X \to Y$ and $S : Y \to Z$ is given by the composite relation

$$S \circ R = \{(x, z) \mid \exists y \in Y, (x, y) \in R \text{ and } (y, z) \in S\}.$$

This is essentially the usual relational composition. The identity morphism on X is the diagonal (identity) relation $\{(x, x) \mid x \in X\}$. Verifying associativity and identity laws is an exercise. In **Rel**, unlike in **Set**, a morphism doesn't assign a unique output to each input; it can relate an element to multiple targets or none. **Rel** is an example of a category that is not *locally small* if we allow arbitrary (possibly proper class-sized) sets, but restricting to small sets it is locally small.

Another example of "non-functional" morphisms: consider a category **Mat** whose objects are natural numbers (or finite sets of given sizes) and a morphism $m \to n$ is an $m \times n$ matrix with real number entries. Composition is matrix multiplication. Identity morphisms are identity matrices. This forms a (small) category. It is closely related to linear algebra and can be seen as the category of finite-dimensional vector spaces with a fixed basis chosen for each (so that linear maps correspond to matrices). Morphisms in **Mat** are not literally functions between the objects (since objects are just numbers), but they can be interpreted as linear functions if we think each object n represents $\mathbb{R}^n$.

The above examples should convince you that the notion of morphism is very general. In category theory, *any* consistent rules of composing arrows qualify, not necessarily arrow-as-function. Thus, category theory can capture processes (like relations, or matrices

as linear transformations) that are more general than single-valued functions.

We also observe that in a category, the composition law determines a directed graph structure: objects are like vertices, and morphisms are like directed edges. In fact, if we forget composition and identities, a category is just a directed graph. Formally:

Definition 1.7 (Underlying Graph of a Category)**.** The **underlying graph** of a category $\mathcal{C}$ is the directed graph whose vertices are $\mathrm{Ob}(\mathcal{C})$ and which has an edge from A to B for every morphism $f : A \to B$. A **diagram** in a category is simply a directed graph (not necessarily the whole underlying graph, but any collection of objects and arrows closed under the incidence relation) drawn using objects of $\mathcal{C}$ as nodes and morphisms as edges. We say a diagram **commutes** if for any two parallel paths (paths with the same start and end object) in the diagram, the composed morphisms are equal in $\mathcal{C}$. Commutative diagrams are a convenient way to express equations between composed morphisms.

For example, the statement that $f : A \to B$ is a monomorphism (injective arrow) can be illustrated by a diagram:

$$C \underset{h}{\overset{g}{\rightrightarrows}} A \xrightarrow{f} B \ ,$$

commuting to $f \circ g = f \circ h \implies g = h$. A diagram with two different paths from C to B (one going through g then f, the other through h then f) indicates the equation $f \circ g = f \circ h$. If that implies the trivial path equality $g = h$, the diagram expresses the universal property of f being monic.

We often draw small diagrams to express such properties or to denote equalities of compositions. For instance, a rectangle

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle x}\downarrow & & \downarrow{\scriptstyle y} \\ C & \xrightarrow[g]{} & D \end{array}$$

is said to commute if $y \circ f = g \circ x$. In more complex diagrams, commutativity might impose several equations.

Example 1.8 (Commutative Diagram Example)**.** Consider the diagram (a portion of which was given in the textbooks & notes consulted):

$$\begin{array}{ccccc} A & \xrightarrow{f} & B & \xrightarrow{h} & C \\ & {\scriptstyle g}\searrow & \downarrow{\scriptstyle j} & & \\ & & D & \xrightarrow[k]{} & E \end{array}$$

This depicts objects A, B, C, D, E and arrows $f : A \to B$, $g : A \to D$, $h : B \to C$, $j : B \to D$, $k : D \to E$, and $m : C \to E$ (with m presumably an arrow $C \to E$ not drawn but can be inferred if the diagram commutes fully). The diagram is said to commute if $j \circ f = g$ and $k \circ j = m \circ h$ (and hence by chasing around, $k \circ g = m \circ h \circ f$). Commutativity imposes equations $g = j \circ f$ and $m \circ h = k \circ j$. From these one can derive $m \circ h \circ f = k \circ j \circ f = k \circ g$ (this kind of derivation is sometimes called a "diagram chase").

This shows how diagrams provide a visual and intuitive way to state algebraic conditions in a category. Many definitions in category theory (especially those of universal properties, limits, etc.) are conveniently phrased as the existence of a unique arrow making certain diagrams commute. We will see examples soon.

Before moving on, we define some special types of morphisms which generalize familiar concepts like injective, surjective, and bijective functions.

Definition 1.9 (Isomorphism, Monomorphism, Epimorphism)**.** Let $f : A \to B$ be a morphism in a category $\mathcal{C}$.

- We say f is an **isomorphism** (or **iso**) if there exists a morphism $f^{-1} : B \to A$ (called the inverse of f) such that $f^{-1} \circ f = \mathrm{id}_A$ and $f \circ f^{-1} = \mathrm{id}_B$. If such an f^{-1} exists, it is unique. If A and B are connected by an isomorphism, we write $A \cong B$ and say A and B are **isomorphic** objects.

- f is a **monomorphism** (or **mono**) if f is left-cancellative: for all morphisms $g, h : X \to A$,
$$f \circ g = f \circ h \implies g = h.$$
In other words, f has a trivial equalizer. Diagrammatically, f is mono if whenever two morphisms into A form the same composite with f, they must be the same morphism. In **Set**, monos are precisely injective functions.

- f is an **epimorphism** (or **epi**) if f is right-cancellative: for all morphisms $u, v : B \to Y$,
$$u \circ f = v \circ f \implies u = v.$$
So epimorphisms are arrows that can be "cancelled" on the right. In **Set**, epis are precisely surjective functions.

Proof. These definitions follow standard categorical terminology. Note that in a general category, mono + epi does *not* necessarily imply iso, unlike the situation in **Set**. A counterexample given in the source is a category with two objects and a single non-identity

arrow between them: that arrow is both mono and epi but not an iso.

□

An isomorphism is a morphism with a two-sided inverse. If $f : A \to B$ is an iso, then $f^{-1} : B \to A$ is also an iso (with $(f^{-1})^{-1} = f$). Isomorphisms represent the idea of "structure-preserving bijections" in category theory. One often regards isomorphic objects as essentially the same for many purposes, though formally they need not be equal. (Some categories enforce that any iso implies equality of objects, such categories are called skeletal, e.g., a poset viewed as a category is skeletal because the only isos are identities.)

Monomorphisms generalize injective maps. For example, in **Grp** (groups and homomorphisms), a homomorphism $f : G \to H$ is monic if and only if its kernel is trivial (since if $f(g) = e$ implies $g = e$ in G, then f is injective on elements). In **Top** (topological spaces), a continuous map may be monic (injective) but need not have a continuous inverse (hence it is not necessarily an isomorphism). Epimorphisms generalize surjections; in **Top**, a continuous surjection is an epi—indeed, every epi in **Top** is a surjective continuous map. (Note that while there are categories in which an epi is not given by surjectivity, in **Top** epis coincide with surjections. However, even in **Top** the combination of mono and epi does not guarantee an isomorphism since a continuous bijection may fail to be a homeomorphism due to the inverse lacking continuity.)

Example 1.10. In **Rel** (sets and relations), what are the monomorphisms and epimorphisms? A relation $R : X \to Y$ is monic if whenever $R \circ S_1 = R \circ S_2$ as relations (with $S_1, S_2 : Z \to X$), then $S_1 = S_2$. It can be shown that monomorphisms in **Rel** are precisely those relations R that are functional and injective as relations (i.e. each element of X is related to at most one element of Y, and if $x\,R\,y$ and $x'\,R\,y$ then $x = x'$). Epimorphisms in **Rel** turn out to be those relations that are right-total and surjective in the relational sense (each $y \in Y$ has at least one $x \in X$ with $x\,R\,y$, and if two relations out of Y agree on R then they must agree overall). These are a bit tricky to characterize succinctly, but it shows that mono/epi generalize injective/surjective but not always identically.

Finally, note that in a one-object category (monoid), a morphism is monic iff $f \cdot g = f \cdot h \implies g = h$ for all g, h in the monoid. If the monoid is like a group, every element is invertible so monic = always true (since you can cancel f). If the monoid is not a group, being monic might be a nontrivial condition on the element f. Similarly for epi. This is just a fun check of definitions in a simple setting.

We have now established the core definitions: categories, morphisms, and some special types of morphisms. Next, we discuss how one category can map to another via a

structure-preserving translation called a *functor*.

1.3 Functors and Natural Transformations

Having categories as mathematical objects, the appropriate notion of a mapping between categories is that of a **functor**. A functor plays the role analogous to a homomorphism between algebraic structures: it maps objects to objects and arrows to arrows in a way that respects all the structure (composition and identities).

Definition 1.11 (Functor)**.** Given two categories $\mathcal{C}$ and $\mathcal{D}$, a **functor** $F : \mathcal{C} \to \mathcal{D}$ consists of:

- For each object $A \in \mathrm{Ob}(\mathcal{C})$, an object $F(A) \in \mathrm{Ob}(\mathcal{D})$.
- For each morphism $f : A \to B$ in $\mathcal{C}$, a morphism $F(f) : F(A) \to F(B)$ in $\mathcal{D}$.

These assignments must satisfy two functoriality conditions:

- **Preservation of identities:** For each object A of $\mathcal{C}$, $F(\mathrm{id}_A) = \mathrm{id}_{F(A)}$.
- **Preservation of composition:** For any composable pair $A \xrightarrow{f} B \xrightarrow{g} C$ in $\mathcal{C}$, we have
$$F(g \circ f) = F(g) \circ F(f)$$
as morphisms $F(A) \to F(C)$ in $\mathcal{D}$.

A functor is often denoted by its action on objects and arrows. It is common to simply write $F : \mathcal{C} \to \mathcal{D}$ and Ff for $F(f)$.

Proof. See Awodey's notes. The conditions explicitly are $F1_A = 1_{FA}$ and $F(g \circ f) = Fg \circ Ff$. □

Functors thus compose (when categories line up) and have identities, making categories and functors into a category of categories. Indeed, one can define the category **Cat** whose objects are (small) categories and whose morphisms are functors. Composition of functors is defined by $(G \circ F)(-) := G(F(-))$ on both objects and arrows, which is associative and has an identity (the identity functor $1_{\mathcal{C}}$ that maps each object to itself and each arrow to itself).

A quick sanity check: a functor $F : \mathcal{C} \to \mathcal{D}$ must send identities to identities, and composition to composition. So F sends any commutative diagram in $\mathcal{C}$ to a commutative

diagram in $\mathcal{D}$ (because if some $h \circ g \circ f$ equals some other composition in $\mathcal{C}$, applying F yields $F(h) \circ F(g) \circ F(f)$ equals the corresponding other composition in $\mathcal{D}$). In particular, if a certain equation holds between morphisms in $\mathcal{C}$, the corresponding equation holds between their images in $\mathcal{D}$. In this sense, functors are structure-preserving maps between categories.

Example 1.12 (Examples of Functors)**.** We list some basic examples of functors:

- The **identity functor** $1_{\mathcal{C}} : \mathcal{C} \to \mathcal{C}$ maps each object to itself and each morphism to itself. Clearly, it satisfies $1_{\mathcal{C}}(g \circ f) = (1_{\mathcal{C}} g) \circ (1_{\mathcal{C}} f)$ since it's just $g \circ f = g \circ f$, and identities map to identities trivially.

- For any two categories $\mathcal{C}$ and $\mathcal{D}$, any mapping that sends objects to objects and arrows to arrows and respects source/target can be checked if it's a functor. For instance, if we have $\mathcal{C}$ as a discrete category (only identity morphisms), then giving a functor $F : \mathcal{C} \to \mathcal{D}$ is equivalent to choosing an object of $\mathcal{D}$ for each object of $\mathcal{C}$; there is no further condition because the only morphisms are identities which must go to identities. So functors subsume the notion of indexing or picking out objects.

- A **constant functor**: Fix a category $\mathcal{D}$ and an object $D \in \mathcal{D}$. For any category $\mathcal{C}$, define a functor $F : \mathcal{C} \to \mathcal{D}$ by $F(A) = D$ for all objects A, and $F(f) = \mathrm{id}_D$ for all morphisms f. This sends everything to D and only identity morphisms. It is indeed a functor (composition and identities all map to 1_D which works). This is called a constant functor picking the object D. It is analogous to a constant function in set theory.

- If $\mathcal{C}$ is a one-object category corresponding to a monoid M (Example 1.4), then a functor $F : \mathcal{C} \to \mathcal{D}$ amounts to an object $D = F(*)$ in $\mathcal{D}$ together with a morphism $F(m) : D \to D$ for each $m \in M$, satisfying $F(1) = \mathrm{id}_D$ and $F(m_2 \cdot m_1) = F(m_2) \circ F(m_1)$. So F is precisely a monoid homomorphism from M to the endomorphism monoid $\mathrm{End}_{\mathcal{D}}(D)$ (the set of all $D \to D$ arrows under composition). In particular, if $\mathcal{D} = \mathbf{Set}$, F corresponds to an action of M on the set $F(*)$. Thus, functors from one-object categories recover the notion of a group or monoid action on some object.

- Consider the inclusion of a subcategory as a functor: If $\mathcal{C}$ is a category and $\mathcal{C}'$ is a **subcategory** (meaning $\mathrm{Ob}(\mathcal{C}') \subseteq \mathrm{Ob}(\mathcal{C})$ and for any two objects in $\mathcal{C}'$, $\mathcal{C}'(A, B) \subseteq \mathcal{C}(A, B)$, and composition in $\mathcal{C}'$ is the same as in $\mathcal{C}$), then the inclusion map $I : \mathcal{C}' \hookrightarrow \mathcal{C}$ that sends each object and morphism to itself (viewed as an object or morphism in $\mathcal{C}$) is a functor (identity on each homset, preserving composition by definition). For example, the functor $I : \mathbf{Grp} \to \mathbf{Mon}$ (viewing every group as a monoid by forgetting inverses) is an inclusion functor (not full or essentially surjective, but faithful—terms defined below).

- If $\mathcal{C}$ and $\mathcal{D}$ are poset categories (Example 1.3), then a functor $F : \mathcal{C} \to \mathcal{D}$ is exactly a monotone function between posets: $a \leq b$ in $\mathcal{C}$ implies $F(a) \leq F(b)$ in $\mathcal{D}$. The functor conditions reduce to that single condition since the only morphisms are given by the order. For instance, if $\mathcal{C} = (P, \leq)$ and $\mathcal{D} = (Q, \leq)$ are posets, then giving $F : P \to Q$ such that $x \leq y \implies F(x) \leq F(y)$ is equivalent to giving a functor between them.

We now introduce two important properties of functors: **faithfulness** and **fullness**. These generalize the idea of an injective or surjective function, but in the context of mapping hom-sets.

Definition 1.13 (Full and Faithful Functors)**.** A functor $F : \mathcal{C} \to \mathcal{D}$ is said to be:

- **Faithful** if for every pair of objects A, B in $\mathcal{C}$, the function

$$F : \mathcal{C}(A, B) \to \mathcal{D}(F(A), F(B))$$

(mapping each $f : A \to B$ to $F(f) : F(A) \to F(B)$) is injective. Equivalently, different morphisms in $\mathcal{C}$ are never mapped to the same morphism in $\mathcal{D}$.

- **Full** if for every A, B in $\mathcal{C}$, the function

$$F : \mathcal{C}(A, B) \to \mathcal{D}(F(A), F(B))$$

is surjective onto $\mathcal{D}(F(A), F(B))$. That is, for every morphism $g : F(A) \to F(B)$ in $\mathcal{D}$, there exists at least one morphism $f : A \to B$ in $\mathcal{C}$ with $F(f) = g$.

If F is both full and faithful, we say F is an **embedding** of $\mathcal{C}$ into $\mathcal{D}$ (also sometimes called a fully faithful functor, which essentially identifies $\mathcal{C}$ as a subcategory of $\mathcal{D}$ up to isomorphism).

Intuitively, a faithful functor doesn't "identify" distinct arrows from $\mathcal{C}$ (no information about morphisms is lost), and a full functor doesn't "miss" any arrows between the images (any potential arrow in $\mathcal{D}$ between images actually comes from an arrow in $\mathcal{C}$). An easy example: the inclusion functor $I : \mathbf{Grp} \to \mathbf{Mon}$ (from groups to monoids) is faithful (a group homomorphism is uniquely determined by itself as a monoid homomorphism, since I is injective on each hom-set), but not full (e.g., a monoid homomorphism between two groups that is not a group homomorphism wouldn't come from **Grp**). The inclusion of a subcategory is always full and faithful by definition.

At this point, we have categories and functors, giving us a category-of-categories viewpoint. The next level of structure is to consider transformations between functors, known

as **natural transformations**. These will be crucial for capturing the idea of equivalence of categories (two categories being "essentially the same") and in many constructions like limits, adjoints, etc.

Definition 1.14 (Natural Transformation)**.** Let $F, G : \mathcal{C} \to \mathcal{D}$ be two functors between the same categories. A **natural transformation** $\eta : F \Rightarrow G$ is a family of morphisms in $\mathcal{D}$, one for each object of $\mathcal{C}$, satisfying a compatibility condition:

- For each object $A \in \mathcal{C}$, there is a morphism (called the **component** of η at A) $\eta_A : F(A) \to G(A)$ in $\mathcal{D}$.

- For every morphism $f : A \to B$ in $\mathcal{C}$, the following **naturality square** commutes in $\mathcal{D}$:

$$\begin{array}{ccc} F(A) & \xrightarrow{F(f)} & F(B) \\ {\scriptstyle \eta_A}\downarrow & & \downarrow{\scriptstyle \eta_B} \\ G(A) & \xrightarrow[G(f)]{} & G(B) \end{array}$$

  That is, $G(f) \circ \eta_A = \eta_B \circ F(f)$.

If such an η exists, we say $\eta : F \Rightarrow G$ is a natural transformation from F to G. If each component η_A is an isomorphism in $\mathcal{D}$, then η is called a **natural isomorphism**, and F and G are said to be **naturally isomorphic** (written $F \cong G$).

Diagrammatically, a natural transformation is a "vertical arrow" connecting two parallel functors, such that for every arrow in the source category, a certain square commutes. The commutative square means η respects the action of each $f : A \to B$, ensuring the transformation is in some sense uniform or "natural" across all morphisms of $\mathcal{C}$ (hence the term natural). We can visualize η as a collection of arrows $\eta_A : F(A) \to G(A)$ for each A, making all these squares commute.

In the diagram above, the equation $G(f) \circ \eta_A = \eta_B \circ F(f)$ expresses exactly that commutativity. It can be remembered as: first apply F to f then η at B, or first apply η at A then apply G to f — both ways give the same result from $F(A)$ to $G(B)$.

Example 1.15 (Natural Transformations)**.** Here are some examples to solidify the concept of natural transformations:

- **Identity transformation:** For any functor $F : \mathcal{C} \to \mathcal{D}$, there is an identity natural transformation $1_F : F \Rightarrow F$, whose components are $1_{F(A)} : F(A) \to F(A)$ for each object A. The naturality square trivially commutes because $G = F$ and $\eta_A = 1_{F(A)}$ so both sides of the equation are just $F(f)$. This is analogous to the identity morphism but in the functor category.

- **Twist transformation:** Consider the functors $F, G : \mathbf{Set} \times \mathbf{Set} \to \mathbf{Set}$ defined by $F(X, Y) = X \times Y$ and $G(X, Y) = Y \times X$, the Cartesian product functor and the "twisted" product. There is a natural transformation $t : F \Rightarrow G$ whose component at each pair (X, Y) is the "swap" isomorphism $t_{(X,Y)} : X \times Y \to Y \times X$, $(x, y) \mapsto (y, x)$. This t is natural because given any pair of functions $f : X \to X'$, $g : Y \to Y'$, the square

$$\begin{array}{ccc} X \times Y & \xrightarrow{f \times g} & X' \times Y' \\ {\scriptstyle t_{(X,Y)}}\downarrow & & \downarrow{\scriptstyle t_{(X',Y')}} \\ Y \times X & \xrightarrow[g \times f]{} & Y' \times X' \end{array}$$

  commutes. Indeed $t_{(X',Y')} \circ (f \times g) = (g \times f) \circ t_{(X,Y)}$ because both send (x, y) to $(g(y), f(x))$. This shows the twist is a natural iso between the product functor and its opposite-order version.

- **Conjugation in group functors:** Let G be a fixed group seen as a one-object category $\mathbf{G}$. Consider two functors $F, H : \mathbf{G} \to \mathbf{G}$ corresponding to two homomorphisms $f, h : G \to G$ (each given by what they do on the single object $*$). A natural transformation $\eta : F \Rightarrow H$ consists of a single component $\eta_* : * \to *$ which is just an element $g \in G$ (since $\hom(*, *) = G$). The naturality condition for the unique (nontrivial) $a \in \hom(*, *)$ (an element of G) says $H(a) \circ \eta_* = \eta_* \circ F(a)$. Writing this in group terms: $h(a) \cdot g = g \cdot f(a)$ for all $a \in G$. This is exactly the condition that g *conjugates* f into h, i.e. $h(a) = gf(a)g^{-1}$. So a natural transformation between two group homomorphisms is a group element g implementing a conjugation between them. In particular, f and h are naturally isomorphic functors iff they are conjugate homomorphisms.

This last example is quite illustrative: it shows that "naturally isomorphic functors" capture the idea of equivalence up to internal symmetry. Two functors $f, h : G \to G$ being conjugate is an example of natural isomorphism. In category theory, we often regard two functors or even two categories as "essentially the same" if there is a natural isomorphism or an equivalence between them.

Now, with natural transformations, we can form a new category called a **functor category**. Given two categories $\mathcal{C}$ and $\mathcal{D}$, the category $[\mathcal{C}, \mathcal{D}]$ (or $\mathcal{D}^{\mathcal{C}}$) has as objects all functors $\mathcal{C} \to \mathcal{D}$, and as morphisms all natural transformations between such functors. Composition in $[\mathcal{C}, \mathcal{D}]$ is defined by "vertical composition" of natural transformations: if $\eta : F \Rightarrow G$ and $\theta : G \Rightarrow H$, then $(\theta \circ \eta)_A = \theta_A \circ \eta_A$ for each object A. One must check this is again natural (which it is, by an exercise in diagram chasing) and associative. The identity in this functor category is 1_F for each functor F.

The functor category $[\mathcal{C}, \mathcal{D}]$ is sometimes denoted $\mathcal{D}^{\mathcal{C}}$. If $\mathcal{C}$ is small and $\mathcal{D}$ is locally small, then $\mathcal{D}^{\mathcal{C}}$ is locally small (in fact, $\mathrm{Hom}(F, G)$ is the set of all natural transformations $F \Rightarrow G$). This functor category concept is powerful: it allows us to consider "metacategories" of functors and reason about functors themselves using categorical notions like limits or adjoints. We will encounter $\mathbf{Set}^{\mathcal{C}}$ which is the category of all set-valued functors on $\mathcal{C}$, important in the Yoneda lemma.

One particularly important functor category is $\mathbf{Set}^{\mathcal{C}^{op}}$, the category of all contravariant set-valued functors on $\mathcal{C}$. These are called **presheaves** on $\mathcal{C}$. The Yoneda lemma, which we will see later, relates $\mathcal{C}$ to a full subcategory of $\mathbf{Set}^{\mathcal{C}^{op}}$. But before we reach Yoneda, we need to cover the notion of *universal properties* like initial objects, products, etc., which will be useful across logic and type theory.

1.4 Universal Constructions: Initial Objects, Products, and Limits

A central idea in category theory is to characterize mathematical constructions by universal properties. These are often phrased as existence and uniqueness of certain morphisms making a diagram commute. Universal properties lead to definitions of **limits** and **colimits**, generalizing concepts like products, coproducts, intersections, unions, inverse limits, etc. In logic, these correspond to logical connectives (e.g. categorical product corresponds to logical conjunction, as we will see) and in type theory to type constructors (product types, sum types). Here we introduce a few basic universal constructions: initial/terminal objects and binary products, which will be directly relevant to categorical semantics of logic and type theory.

Definition 1.16 (Terminal and Initial Objects)**.** In a category $\mathcal{C}$:

- A **terminal object** is an object T such that for every object A in $\mathcal{C}$, there exists a unique morphism $A \to T$. (In other words, a terminal object can be seen as an empty **limit**.) It is sometimes denoted by 1.

- An **initial object** is an object I such that for every object A, there exists a unique morphism $I \to A$. (Thus, an initial object may be regarded as an empty **colimit**.) It is often denoted by 0.

By definition, if T is terminal, for each A there is exactly one arrow $!_A : A \to T$ (some texts use this exclamation mark notation). Terminal objects are "universal receivers" of arrows. Dually, initial objects are "universal sources" of arrows (unique arrow out of the initial object).

Terminal and initial objects, if they exist, are unique up to isomorphism. For example, in **Set**, any singleton set is a terminal object: given any set A, there is a unique function from A to $\{*\}$ (sending every element of A to the single element $*$). All singletons are isomorphic (indeed equal in **Set** if we pick a particular one, say $\{0\}$, as the distinguished terminal). Dually, the empty set $\emptyset$ is an initial object in **Set**: for any set A, there is exactly one function $\emptyset \to A$ (the empty function). In **Set**, $\emptyset$ is initial and any singleton is terminal.

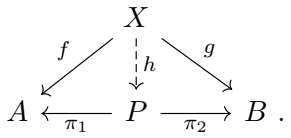
In a poset category (viewing a poset as a category with arrows as order relations), a terminal object is an object that every other object has exactly one arrow to: that means an element T such that for all x, $x \leq T$ and the arrow is unique which implies the order is antisymmetric so T is a top element (maximum). So "terminal" in a poset is the same as having a greatest element. Similarly, initial is a least element (minimum). Many order examples correspond to algebraic notions like 1 being a unit (top element) and 0 being a zero element (bottom element) in lattices.

Initial and terminal objects are special cases of limits and colimits (specifically, an initial object is an empty limit, a terminal object is an empty colimit). These objects are especially significant in logic: for instance, an initial object in a category of models can represent an "inconsistent theory" because there's a unique arrow from it to any object (meaning a false proposition implies anything). But let's not get ahead of ourselves.

Next, consider **binary products**. The categorical product generalizes the Cartesian product of sets, the direct product of groups, etc., by a universal property. It also will later correspond to conjunction in logic and product types in type theory.

Definition 1.17 (Binary Product)**.** Given two objects A and B in a category $\mathcal{C}$, a **product** of A and B is an object P equipped with two morphisms (projections) $\pi_1 : P \to A$ and $\pi_2 : P \to B$, such that for any object X with morphisms $f : X \to A$ and $g : X \to B$, there exists a *unique* morphism $h : X \to P$ (called the **mediating morphism** or pairing of f

and g, often denoted $\langle f, g \rangle$) making the following diagram commute:



In other words, $\pi_1 \circ h = f$ and $\pi_2 \circ h = g$, and h is the unique arrow from X to P with that property. We often denote the product as $A \times B$ (especially in **Set** or in a category with a set-like flavor) and write $\pi_1 : A \times B \to A$, $\pi_2 : A \times B \to B$.

This is a **universal property**: the product (P, π_1, π_2) is such that any "cone" of morphisms $X \to A$ and $X \to B$ factors uniquely through P. The dashed arrow in the diagram above is determined uniquely by f and g. One often says (P, π_1, π_2) is the **universal cone** from (A, B). More formally, it is the limit of the diagram consisting of A and B with no arrows between them (just two objects) – but we'll stick to this concrete definition for binary products.

From this definition, one can prove that if a product exists, it is unique up to unique isomorphism. Essentially, if (P, π_1, π_2) and (P', π_1', π_2') are two products of A and B, by universality a unique iso between P and P' can be constructed. So we can speak of *the* product of A and B in a category (assuming it exists).

In **Set**, the product is the usual Cartesian product: given sets A and B, $A \times B$ with projection maps to each coordinate is a product in the categorical sense. Indeed, given any set X and functions $f : X \to A$, $g : X \to B$, the universal property is satisfied by the function $h : X \to A \times B$ defined by $h(x) = (f(x), g(x))$. This h is unique with the property $\pi_1(h(x)) = f(x)$, $\pi_2(h(x)) = g(x)$, so $A \times B$ with projections is a product in **Set**.

In **Grp**, the product of two groups is the direct product group $G \times H$ with componentwise operations. The projections are the group homomorphisms sending (g, h) to g and to h respectively. The universal property holds: given homomorphisms $f : X \to G$ and $g : X \to H$, the unique homomorphism $h : X \to G \times H$ is $h(x) = (f(x), g(x))$, which is a homomorphism if f, g are.

In any category with a terminal object 1, the product of an object A with 1 (if it exists) is isomorphic to A (since a terminal object 1 satisfies a similar universal property, the mediating map from any X to $A \times 1$ corresponds to giving maps to A and 1, but there's a unique map to 1, so it's essentially the same as a map to A). So $A \times 1 \cong A$. In **Set**, $A \times \{*\} \cong A$ obviously.

One can also define **binary coproducts** dually: an object C with injections $i_1 : A \to C$, $i_2 : A \to C$ universal for any object X with arrows from A and B into X. That is a *pushout* of two initial maps or simply a co-product. In **Set**, the coproduct is disjoint union $A \amalg B$ (with inclusion maps). In **Grp**, coproduct is the free product of groups. In logic and type theory, coproduct corresponds to disjunction or sum types.

Now, the power of categorical definitions is that they yield theorems like: if a certain universal property is satisfied, then certain consequences follow automatically. For example, one can prove:

Proposition 1.18 (Uniqueness of Products Up to Isomorphism)**.** *If (P, π_1, π_2) is a product of A and B and (P', π_1', π_2') is another product of A and B in the same category, then there is a unique isomorphism $\varphi : P \to P'$ such that $\pi_1' \circ \varphi = \pi_1$ and $\pi_2' \circ \varphi = \pi_2$.*

Proof. By the universal property of (P, π_1, π_2), since $\pi_1' : P' \to A$ and $\pi_2' : P' \to B$ are maps from P' into A and B, there is a unique map $u : P' \to P$ with $\pi_1 \circ u = \pi_1'$ and $\pi_2 \circ u = \pi_2'$. Similarly, by universality of (P', π_1', π_2') given $\pi_1 : P \to A$, $\pi_2 : P \to B$, there is a unique map $v : P \to P'$ with $\pi_1' \circ v = \pi_1$ and $\pi_2' \circ v = \pi_2$. Now consider $v \circ u : P' \to P'$. This arrow satisfies $\pi_1' \circ (v \circ u) = \pi_1'$ and $\pi_2' \circ (v \circ u) = \pi_2'$ (because $\pi_1' \circ v = \pi_1$, then $\pi_1' \circ (v \circ u) = (\pi_1' \circ v) \circ u = \pi_1 \circ u = \pi_1'$, and similarly for π_2'). But the identity $\mathrm{id}_{P'} : P' \to P'$ also satisfies $\pi_1' \circ \mathrm{id}_{P'} = \pi_1'$, $\pi_2' \circ \mathrm{id}_{P'} = \pi_2'$. By the uniqueness in the definition of product applied to P', we must have $v \circ u = \mathrm{id}_{P'}$. Similarly $u \circ v = \mathrm{id}_P$. Thus u and v are inverses, giving the isomorphism. The equalities $\pi_i' \circ v = \pi_i$ ensure $\pi_i' \circ (v \circ u) = \pi_i'$ hold as used. Therefore $\varphi := v$ is the desired isomorphism (its inverse is u) and it is unique by the same kind of reasoning (since any such φ must satisfy those projection equalities and thus must equal v by uniqueness).

□

This formal argument mirrors a common approach in category theory: use universality twice to get mutual inverses. It highlights that a product, if it exists, is essentially determined by A and B.

Given the product (P, π_1, π_2), sometimes we write $P = A \times B$ and denote the unique arrow from any X by $\langle f, g \rangle : X \to A \times B$ such that $\pi_1 \circ \langle f, g \rangle = f$ and $\pi_2 \circ \langle f, g \rangle = g$. This notation $\langle f, g \rangle$ is called the **pairing** of f and g.

One can derive a more concise definition of product from this: P is a product iff there is a bijection

$$\hom(X, P) \cong \hom(X, A) \times \hom(X, B)$$

naturally in X. This means each map $X \to P$ corresponds exactly to a pair of maps $X \to A$ and $X \to B$. The bijection is given by $h \mapsto (\pi_1 \circ h, \pi_2 \circ h)$, and its inverse is $(f, g) \mapsto \langle f, g \rangle$. Naturality in X ensures this bijection is functorial in X (like an isomorphism of functors

in X), which holds by uniqueness property.

Products are a special case of *limits*. There are other limits like equalizers, pullbacks, etc., but products and terminal objects are enough to define what is known as a **Cartesian category** (one with all finite products). If a category has a terminal object and binary products for every pair of objects, it is a **Cartesian monoidal category** essentially, which in logical terms means it can interpret conjunction and truth (the unit for conjunction).

In logic, if we think of $\mathcal{C}$ as a category of "domains" (like sets) and morphisms as "interpretations of proofs or functions", a terminal object corresponds to a **truth value** $\top$ (since from any context there's a unique way to derive truth, reflecting the idea that truth is always true), and a product corresponds to a logical conjunction (to give a map into $A \times B$ is to give a proof of A and a proof of B simultaneously, echoing the meaning of $A \wedge B$). We will formalize this correspondence later.

For now, let's practice with a short exercise:

Exercise 1.19. Verify in the category **Set** that:

1. A terminal object exists and is a singleton set. Prove that if T and T' are two singleton sets, there is a unique bijection $T \to T'$ (hence they are isomorphic in **Set**).

2. The product of two sets A and B as per Definition 1.17 is given by the Cartesian product $A \times B$ with the usual projections. Formally show the uniqueness of the mediating arrow $h : X \to A \times B$ given $f : X \to A$ and $g : X \to B$.

3. Dually, describe the coproduct of two sets A and B in **Set** (this is the disjoint union $A \amalg B$). What are the injection morphisms, and why is the mediating morphism (out of $A \amalg B$) unique?

We will not dive deeper into general limits and colimits (like equalizers, coequalizers, pullbacks, pushouts) right now, though they are defined similarly by universal properties. Instead, having established some categorical fundamentals, we proceed to see how these concepts play out in *categorical logic*. We will start with algebraic theories and propositional logic in the next chapter, where finite products and certain special objects (like truth values) come into play.

Chapter 2

Algebraic Theories and Propositional Logic

In this chapter, we bridge category theory with logic by examining **algebraic theories** (also known as *Lawvere theories*) and categorical semantics of propositional logic. Algebraic theories provide a categorical framework for equational reasoning about algebraic structures (like groups, rings, etc.), while propositional logic provides the simplest case of logical deduction, which we can analyze categorically via Boolean or Heyting algebras and their dual spaces. We will see that both can be subsumed under the idea of categories with certain universal properties: finite product categories for algebraic theories, and (for propositional logic) Boolean algebras as certain categorical structures.

2.1 Algebraic Theories and Functorial Semantics

An **algebraic theory** in the sense of Lawvere is a presentation of an algebraic structure by operations and equations, with the restriction that it only deals with operations of finitely many inputs and equations that are universally quantified (no relations or quantifiers beyond $\forall$ for equations). Examples include the theory of groups, rings, lattices, etc. Each such theory can be represented by a category whose objects are finite lists of variables (or equivalently natural numbers representing arity) and whose morphisms are equivalence classes of terms (operations) in those variables.

The approach we take here follows Lawvere's insight: *a theory can be identified with a category with finite products in which certain objects (arities) and arrows (operations) are specified, and models of the theory correspond to product-preserving functors from that category to* **Set**.

2.1.1 Syntax and Theories as Categories

Let's outline how to go from a conventional algebraic theory (with operations and equations) to a category:
- Start with a signature Σ: a set of operation symbols, each with a given finite arity. For example, in the theory of groups, $\Sigma = \{e : 0\text{-ary},\ m : 2\text{-ary},\ i : 1\text{-ary}\}$ containing a constant e, a binary operation m (multiplication), and a unary operation i (inverse).
- Consider the **term algebra** built from this signature: for each natural number n (think of n as a set of n distinct variables $x_1, \ldots, x_n$), one can form terms in those n variables using the operations. Equations are pairs of terms of the same arity (like $m(x, e) = x$).
- The **Theory category** T_Σ has as objects the natural numbers (or equivalently, finite sets $\{x_1, \ldots, x_n\}$ of variables). A morphism $n \to m$ in T_Σ is given by m terms in n variables: essentially an m-tuple of Σ-terms each using n input variables. Composition of morphisms is substitution of terms into each other.
- To enforce the equations of the theory, we take a quotient of this term category by the congruence generated by those equations. That is, identify morphisms (tuples of terms) if the corresponding equations can prove they are equal. The result is a category (indeed a *Lawvere theory*) in which morphisms represent equivalence classes of terms modulo the axioms.

Crucially, T_Σ is a category with finite products: the product of objects n and m is just $n + m$ (because a context with n variables and a context with m variables together give $n + m$ variables). The terminal object is 0 (no variables, the context of a closed term). The projection maps in this category correspond to projection terms. This category can be seen as the **classifying category** of the theory: models of the theory correspond to product-preserving functors from T_Σ to **Set**.

Let's illustrate with a simple algebraic theory:

Example 2.1 (Theory of Monoids as a Category)**.** Consider the theory of monoids. The signature Σ has a constant e (unit) and a binary operation $\cdot$ (multiplication). The axioms are:

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z, \quad x \cdot e = x, \quad e \cdot x = x.$$

In the theory category T_{Mon}:

- Objects are the natural numbers $0, 1, 2, \ldots$, where n represents a context with n free variables.
- A morphism $f : n \to 1$ is given by a single term in n variables modulo the monoid axioms. For example, the term

$$x_1 \cdot (x_2 \cdot e)$$

represents a morphism $2 \to 1$ and, using the axiom $x_2 \cdot e = x_2$, is identified with $x_1 \cdot x_2$.

- More generally, a morphism $n \to m$ is an m-tuple of terms in n variables.

To illustrate composition, consider the following concrete instance. Let

$$f : 2 \to 2 \quad \text{be given by} \quad f(x_1, x_2) = (x_1^2, x_2)$$

(where x_1^2 denotes $x_1 \cdot x_1$), and let

$$g : 1 \to 2 \quad \text{be given by} \quad g(z) = (z \cdot z, e).$$

Composing g with f means substituting in f the term $z \cdot z$ for x_1 and e for x_2. Thus, the composite morphism

$$f \circ g : 1 \to 2$$

is given by

$$f \circ g(z) = \big((z \cdot z)^2,\, e\big),$$

and since $(z \cdot z)^2$ equals $z \cdot z \cdot z \cdot z$ (i.e. z^4), the result is

$$(z^4,\, e) : 1 \to 2.$$

This cleanly demonstrates substitution and the use of the axioms.

T_{Mon} has finite products: the product of object m and n is $m + n$. The projection $\pi_1 : m+n \to m$ is given by the tuple of terms that picks out the first m inputs and ignores the rest (like $\pi_1(x_1, \ldots, x_{m+n}) = (x_1, \ldots, x_m)$, π_2 picks $x_{m+1}, \ldots, x_{m+n}$). These satisfy the universal property by term-wise combination.

A functor $F : T_{\text{Mon}} \to \mathbf{Set}$ that preserves finite products will assign $F(1)$ a set (call it M) and interpret the constant e as an element $F(e) \in M = F(1)$, and the binary operation $\cdot$ as a function $M \times M = F(1) \times F(1) \to F(1)$ (since F takes the arrow $2 \to 1$ corresponding to multiplication term to a function $F(2) = M \times M \to M$). Preservation of products ensures $F(2) \cong F(1) \times F(1) = M \times M$. F will automatically satisfy the axioms, because those correspond to equalities of morphisms in T_{Mon} which F will preserve, yielding equalities of functions on M. Thus $(M, F(\cdot), F(e))$ is a monoid in **Set**. Conversely any monoid $(M, \cdot, e)$ yields such a functor (called the **forgetful functor** from the theory to **Set** picking out that model). Therefore, $\text{Funct}_{\text{prod}}(T_{\text{Mon}}, \mathbf{Set})$ (the category of finite product preserving functors and natural transformations between them) is equivalent to the category of monoids and monoid homomorphisms.

The above example demonstrates the core idea: there is a category T (sometimes called the **theory category** or **classifying category**) whose product-preserving set-valued functors correspond exactly to models of the theory T in **Set**. This result is often referred to as **functorial semantics** or the *duality between theories and categories of models*:

Theorem 2.2 (Soundness and Completeness for Equational Logic (Functorial Semantics))**.** *Let T be an algebraic theory (presented as a category with finite products, as above). Then:*

1. *(**Soundness**) Any equation $s = t$ derivable from the axioms of T holds in every model of T. Equivalently, if s and t are represented by the* same *morphism $u : n \to 1$ in the theory category T (meaning s and t are proved equal), then for any product-preserving functor (model) $M : T \to \mathbf{Set}$, the interpretations $M(s), M(t) : M^n \to M^1$ are the same function.*

2. *(**Completeness**) If an equation $s = t$ (with s, t terms in n variables) holds in every set-based model of T, then it is derivable from the axioms of T (i.e., becomes equal as morphisms in T). Equivalently, if for all product-preserving functors $M : T \to \mathbf{Set}$ we have $M(s) = M(t)$ as functions $M^n \to M$, then $s = t$ can be proved from T's axioms.*

*Thus, the category T together with the forgetful functor $U : \mathrm{Funct}_{prod}(T, \mathbf{Set}) \to \mathbf{Set}$ (selecting the underlying set of the interpretation of one generator) forms a **classifying category** for the theory, in the sense that T is initial among categories receiving a product-preserving functor from T.*

Proof. This is essentially a restatement of the fact that T captures exactly the provable equalities. For soundness: if $s = t$ in T, then $M(s) = M(t)$ for any functor M because M is a functor (so it preserves equalities of morphisms in T). In other words, provable equality implies semantic equality in all models. This is just saying the functor M cannot distinguish s and t if T itself doesn't.

For completeness: Suppose $s \neq t$ in T (not provably equal). Then as distinct morphisms $s, t : n \to 1$, by the nature of categories of this form, we can find a model M (a functor) that separates them. Concretely, one standard argument is to take a model in **Set** that is the *term model* (or free model) of T on n generators. In that free model, the interpretations of s and t will be distinct elements (because if they were equal in the free model, that would already impose an equation derivable in T by the principle of free objects). This M can be constructed by taking $M(1)$ as the set of T-terms in n variables modulo provable equality, and interpreting each operation symbol as the induced operation on those equivalence classes. By construction $M(s) \neq M(t)$ in this model, yet M is a valid

product-preserving functor (since the term model obviously satisfies the axioms), contradicting the assumption. So if no model distinguishes s and t, they must already be equal in T. This establishes that semantic equality (holding in all models) implies provable equality.

In categorical terms, T is initial among all categories with finite products equipped with a model of the theory, which is a general statement of completeness: given any model $M : T \to \mathbf{Set}$, there is a unique morphism from T to the category "generated" by M. But perhaps the simplest explanation is the term algebra argument given above. □

This theorem connects syntactic and semantic notions: it's essentially the completeness of equational logic (as every equation true in all models is derivable). There is a precise contravariant equivalence between the category of algebraic theories (i.e. small categories with finite products, known as Lawvere theories) and the category of their Set-models. More precisely, for a Lawvere theory T, one associates the category T-Mod of product-preserving functors $T \to \mathbf{Set}$. This assignment establishes a duality: informally,

$$\mathsf{Th} \simeq (\mathsf{Mod})^{op},$$

meaning that the syntactic category of a theory is (up to equivalence) dual to the category of its models. This is often referred to as **Lawvere duality**.

In short, categories allow us to algebraically encode theories, and then use category-theoretic constructs (like functor categories) to reason about models. This viewpoint will extend when we add more logical constructs beyond plain equations (like adding $\vee, \exists$, etc., requiring more structure like coequalizers or left adjoints).

2.1.2 Examples of Algebraic Categories and Functors

To ground the above in more examples:
- If T is the theory of groups, then T is a category with finite products, and $\mathrm{Funct}_{\mathrm{prod}}(T, \mathbf{Set}) \simeq \mathbf{Grp}$, the category of groups.
- A homomorphism of models corresponds to a natural transformation between the corresponding functors (one can show that product-preserving functors $T \to \mathbf{Set}$ are in fact determined by the set $X = F(1)$ with the structure of an algebra, and a natural transformation between F and G corresponds to a function $F(1) \to G(1)$ commuting with the interpretations of operations, i.e., an algebra homomorphism).
- There is a slight generalization: if instead of **Set** we consider functors $T \to \mathcal{C}$ for some other category $\mathcal{C}$ with finite products, those correspond to $\mathcal{C}$*-models* of the theory. For example, a product-preserving functor $T_{\mathrm{Grp}} \to \mathbf{Set}$ yields a group (set-based model), while a product-preserving functor $T_{\mathrm{Grp}} \to \mathbf{Cat}$ would yield a group object in **Cat**, which is essentially a group in the category of categories (which might be just a one-object group-like

category), etc.

The concept of **classifying category** T for a theory is analogous to a classifying space in topology or classifying topos in logic: it is an entity that encapsulates the theory fully so that (in this case) set-based models correspond to functors from it. This notion will reappear when we consider first-order logic, where instead of a category with finite products, we will need categories with additional structure (like finite limits, or a topos with subobject classifier, etc.) to classify more complex theories.

We have focused on set-valued models. One can show a more general statement: T not only classifies set models, but also T itself as a functor $T \to T$ is the *generic model* (the identity functor considered as a model in its own category), and any model $M : T \to \mathcal{C}$ in a different category $\mathcal{C}$ factors through a unique interpretation from the generic model. This is part of the universal property: T is an initial object in the category of categories with finite products and a chosen model of the theory. We won't formalize that further here, but conceptually it's important: the theory category yields a **generic model** such that all others are its images.

Now, having established how categories capture algebraic (essentially universally quantified equation) logic, we move to propositional logic, which introduces connectives that are not just operations on one set, but truth-functional operations, and often involves a two-valued truth notion in classical logic or multi-valued in intuitionistic logic. We will first treat classical propositional logic via Boolean algebras and Stone duality, then intuitionistic logic via Heyting algebras and locales.

2.2 Boolean Algebras and Classical Propositional Logic

Classical propositional logic can be seen as an algebraic theory, but not quite an algebraic theory in the finitary sense, because it has not just operations $\wedge, \vee, \neg$ which are finitary, but the laws involve also the nullary operations **true** ($\top$) and **false** ($\bot$), and critically, the law of double negation or excluded middle is non-algebraic if considered as a scheme. However, propositional logic can indeed be captured by the notion of a **Boolean algebra**: an algebraic structure with operations $\wedge$, $\vee$, $\neg$, 0, 1 satisfying certain axioms (those of a complemented distributive lattice).

Definition 2.3 (Boolean Algebra)**.** A **Boolean algebra** is a set B equipped with binary operations $\wedge$ (meet, logical "and") and $\vee$ (join, logical "or"), a unary operation $\neg$ (complement, logical "not"), and distinguished elements 0 (bottom, false) and 1 (top, true), such that for all $a, b, c \in B$ the following axioms hold:

- $(B, \wedge, \vee)$ is a distributive lattice: - $a \wedge (b \wedge c) = (a \wedge b) \wedge c$ and $a \vee (b \vee c) = (a \vee b) \vee c$ (associativity);
  - $a \wedge b = b \wedge a$, $a \vee b = b \vee a$ (commutativity);
  - $a \wedge (a \vee b) = a$, $a \vee (a \wedge b) = a$ (absorption laws);
  - $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$, $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ (distributivity of $\wedge$ over $\vee$ and vice versa).

- Complement laws: for each $a \in B$, - $a \wedge \neg a = 0$ and $a \vee \neg a = 1$.

- 0 and 1 are the identity elements for $\vee$ and $\wedge$ respectively: $a \vee 0 = a$, $a \wedge 1 = a$.

This definition is algebraic: all axioms are universally quantified equations in the language $\{\wedge, \vee, \neg, 0, 1\}$ (the complement law $a \vee \neg a = 1$ and $a \wedge \neg a = 0$ are equations, not implications). So one could indeed form a theory category for Boolean algebras and get a similar correspondence with Boolean algebras as product-preserving functors into **Set**. Boolean algebras capture exactly the equational part of propositional logic: a Boolean algebra homomorphism is a function $h : B_1 \to B_2$ preserving $\wedge, \vee, \neg, 0, 1$. The category **Bool** of Boolean algebras and homomorphisms is algebraic.

But propositional logic usually is presented as a deductive system with propositional variables, formulas built via logical connectives, and inference rules. There is a straightforward connection:
- If you take the free Boolean algebra on a set of generators (propositional variables), its elements correspond to propositional formulas modulo logical equivalence.
- In particular, the free Boolean algebra on no generators (just constants) has exactly two elements and corresponds to classical propositional calculus with no propositional variables, which can only produce either tautology (True) or contradiction (False). That algebra is the two-element Boolean algebra $\mathbf{2} = \{0, 1\}$ with 0 and 1.
- The two-element Boolean algebra **2** plays the role of the set of truth values. In fact, any Boolean algebra can be seen as the algebra of propositional values of some logical theory: an element $a \in B$ is like the truth value of some proposition under some maximal consistent assignment of truth values to atoms. (This can be made precise: any Boolean algebra is isomorphic to a field of sets, via Stone's representation theorem; each element corresponds to a set of "possible worlds" or ultrafilters where that proposition is true. More on that soon.)

The key result bridging Boolean algebras and propositional logic is that *the equational theory of Boolean algebras is equivalent to classical propositional logic in the sense that a formula is a tautology iff it equals* 1 *in every Boolean algebra interpretation* (soundness and

completeness of Boolean algebra semantics for propositional logic). This can be stated as:

Theorem 2.4 (Soundness and Completeness of Propositional Calculus)**.** *Let Φ be a set of propositional formulas (assumptions) and φ a formula. Then the following are equivalent:*

1. *$\Phi \vdash \varphi$ (formula φ is provable from premises Φ in classical propositional calculus).*
2. *Every Boolean algebra valuation that makes all formulas in Φ equal 1 (true) also makes $\varphi = 1$.*
3. *In every Boolean algebra B and every homomorphism $v : \mathbf{Form} \to B$ sending each assumption in Φ to 1, we have $v(\varphi) = 1$ in B. Here **Form** is the free Boolean algebra on the set of propositional variables (so a formula is interpreted as an element of B under v).*

Consequently, a formula φ is a tautology (provable with no assumptions) if and only if φ evaluates to 1 in all Boolean algebra valuations (which includes the usual two-valued truth table evaluation as a special case with $B = \mathbf{2}$).

Proof. $(1 \Rightarrow 2)$ is soundness: if $\Phi \vdash \varphi$, then any interpretation making Φ true must make φ true. This is proved by induction on the length of the proof; it holds for each axiom (which are tautologies) and is preserved by inference rules (Modus Ponens corresponds to a certain condition on valuations preserving implication truth). Boolean algebra semantics is equivalent to truth-table semantics for propositional logic (by Stone's representation, but even easier: if something holds in all Boolean algebras, in particular it holds in $\mathbf{2}$, the standard truth values), so this is the usual soundness of propositional logic.

$(2 \Rightarrow 1)$ is completeness: if $\Phi \nvdash \varphi$, then $\Phi \cup \{\neg\varphi\}$ is consistent, extend it to a maximal consistent set Γ. Γ corresponds to an ultrafilter of formulas (all formulas that Γ proves). One then defines a Boolean algebra B as the set of all equivalence classes of formulas modulo Γ-provable equivalence. This B is a Boolean algebra model where each formula ψ is mapped to its equivalence class $[\psi]$. In B, each element of Φ maps to 1 (since each $\psi \in \Phi$ is in Γ, hence provably true in Γ). However, φ maps to 0 in B because $\neg\varphi \in \Gamma$. Thus we have a Boolean algebra (in fact $\mathbf{2}$ can be taken by Lindenbaum–Tarski if we just do truth assignments) that satisfies Φ but not φ. This shows if not $\Phi \vdash \varphi$, there is a counter-model, proving completeness.

(3) is just a restatement of (2) in algebraic terms, since "Boolean algebra valuation" can be formalized as a homomorphism from the free Boolean algebra (the algebra of all propositions) to B that sends each assumption (which is a specific element of the free algebra) to 1.

Hence, $\Phi \vdash \varphi$ iff in every B and every homomorphism v with $v(\Phi) = \{1\}$, we have $v(\varphi) = 1$. In particular, taking Φ empty, $\vdash \varphi$ (tautology) iff for every Boolean algebra B and homomorphism v, $v(\varphi) = 1$. Taking specifically $B = \mathbf{2}$ recovers the usual notion: φ is true under every truth assignment. □

This is basically the algebraic completeness (Lindenbaum algebra argument) for propositional logic, which is a special case of the general equational completeness theorem we proved for algebraic theories, instantiated to the theory of Boolean algebras. Indeed, the above is an example of how one would approach it using the category of Boolean algebras and its functors to **Set**. The free Boolean algebra on n generators is the algebra of all formulas in those n variables modulo provable equivalence, and a homomorphism from that free algebra to $\mathbf{2}$ corresponds to a truth assignment. Since $\mathbf{2}$ is a Boolean algebra, the existence of a homomorphism sending formulas to $1/0$ such that assumptions go to 1 and conclusion to 0 yields an actual Boolean valuation counterexample.

Thus, Boolean algebra semantics is equivalent to the usual semantics.

Now, category theory enters through the observation: the category of Boolean algebras **Bool** is not just an ordinary algebraic category, it has some additional structure (it's a variety of algebras). But more interestingly, **Bool** is dual (in the sense of a contravariant equivalence) to the category of certain topological spaces. This is **Stone duality**:

Theorem 2.5 (Stone Representation and Duality)**.** *Every Boolean algebra B is isomorphic to a field of sets (an algebra of subsets of some set, with union, intersection, complement). Concretely, B is isomorphic to the algebra of all clopen (simultaneously closed and open) subsets of a certain compact Hausdorff zero-dimensional topological space* X *(the space of ultrafilters of B). In addition:*

- *The points of* X *correspond to ultrafilters u of B, and the clopen set corresponding to $b \in B$ is $\{u \in \mathrm{X} \mid b \in u\}$.*

- *This assignment extends to a dual equivalence (contravariant isomorphism of categories) between* **Bool** *and the category* ***Stone*** *of Stone spaces (compact Hausdorff zero-dimensional topological spaces) and continuous maps.*

- *In particular, the two-element Boolean algebra* $\mathbf{2}$ *corresponds to a one-point space. A homomorphism of Boolean algebras $h : B_1 \to B_2$ corresponds to a continuous function $\mathrm{X}_{B_2} \to \mathrm{X}_{B_1}$ between their Stone spaces (reversing arrows, hence duality).*

Proof. (Sketch) Stone's representation theorem constructs $\mathrm{X} = \mathrm{Ult}(B)$, the set of all ultrafilters (maximal filters) of B. Each $b \in B$ defines a subset $\hat{b} = \{u \in \mathrm{Ult}(B) \mid b \in u\}$. One checks that $\hat{b}$ is both open and closed in a certain topology (the Stone topology) on X, where basic open sets are of form $\hat{b}$. The map $b \mapsto \hat{b}$ is a Boolean algebra homomorphism from B to the power set Boolean algebra $\mathcal{P}(\mathrm{Ult}(B))$, and is injective (because distinct b differ on some ultrafilter). Surjectivity onto clopen sets follows because finite Boolean combinations of basic opens are still clopen and generate the clopen algebra. So $B \cong \mathrm{Clopen}(\mathrm{Ult}(B))$ as Boolean algebras. $\mathrm{Ult}(B)$ is compact and zero-dimensional by construction (clopens form a basis), and Hausdorff because ultrafilters are distinct enough to be separated by clopens (using an element in one ultrafilter but not another yields disjoint opens containing each ultrafilter).

This extends to a categorical duality: Given a Boolean homomorphism $h : B_1 \to B_2$, the inverse image of an ultrafilter along h is an ultrafilter on B_1. Thus to each ultrafilter u of B_2 we assign $h^{-1}(u)$ which is an ultrafilter of B_1. This gives a map $\mathrm{Ult}(B_2) \to \mathrm{Ult}(B_1)$ that is continuous in the Stone topology. This assignment is functorial and contravariant, and it yields an equivalence of categories $\mathbf{Bool}^{op} \simeq \mathbf{Stone}$.

□

Stone duality is very significant: it shows that Boolean logic (via Boolean algebras) is equivalent to studying certain topological spaces. The Stone space $\mathrm{Ult}(B)$ of ultrafilters can be seen as the space of all "possible worlds" or models of the propositional theory represented by B. Each ultrafilter is like a consistent assignment of truth values (a maximally consistent set of propositions), and $\hat{b}$ is the set of models where proposition b is true. So Stone duality provides a topological semantics for propositional logic: the clopen subsets of the Stone space form the algebra of propositions. In particular:
- If B is the Boolean algebra of all formulas modulo equivalence, then $\mathrm{Ult}(B)$ is essentially the space of all truth assignments or models of the propositional theory, and a formula b corresponds to the set of assignments satisfying it (this is a Stone representation of the Lindenbaum algebra).
- Stone's theorem thus gives a robust formulation of the completeness theorem: the logical content (Boolean algebra of formulas) is captured by the structure of the space of models.

From a categorical logic viewpoint, Stone duality is the first example of a logic-space duality: algebraic logic (Boolean algebra) vs. geometric space (Stone space). This heralds the idea of the *duality between syntax and semantics*: syntax (formulas, proofs) forms an algebra or category, semantics (models) forms spaces or categories of models, and there is a duality or equivalence bridging them.

We will see a similar pattern for intuitionistic logic: Heyting algebras vs. topological spaces called locales, and a duality there too (Stone duality generalizes to the so-called

Priestley or Esakia duality for distributive lattices or Heyting algebras, but with some conditions one gets locales).

2.2.1 Heyting Algebras and Frames: Intuitionistic Propositional Logic

Definition 2.6 (Heyting Algebra)**.** A **Heyting algebra** is a bounded distributive lattice $(H, \wedge, \vee, 0, 1)$ equipped with a binary operation $\Rightarrow$ (called *implication*) such that for all $a, b, x \in H$,

$$a \wedge x \leq b \quad \text{if and only if} \quad x \leq (a \Rightarrow b).$$

In a Heyting algebra, one defines the *negation* of an element a by

$$\neg a := (a \Rightarrow 0).$$

If additionally the law of excluded middle holds (i.e. $a \vee \neg a = 1$ for every $a \in H$), then H is a Boolean algebra.

In categorical logic terms, a Heyting algebra is a lattice that is also a *Cartesian closed poset*: finite meets (including top) serve as products, and $\Rightarrow$ is an internal hom making it cartesian closed. Actually, each finite meet with some a yields an adjoint $a \wedge - \dashv (a \Rightarrow -)$, capturing quantification: $a \Rightarrow b$ plays the role of $a \to b$ or logically $a \implies b$ which is also $\neg a \vee b$ in a Boolean algebra but not necessarily in a Heyting.

A key point: Intuitionistic propositional logic is complete with respect to models in Heyting algebras (or equivalently in Kripke models, or topological models as we mention soon). There is a version of Stone duality for Heyting algebras, but since a Heyting algebra might not have enough complements, its dual space is not a general Stone space but a **locale** (a space in point-free topology). A **frame** is a complete Heyting algebra (one that has arbitrary joins and finite meets), and a **locale** is defined as a frame considered as an abstract "lattice of open sets" of some space. Indeed, frames and continuous maps (which preserve finite meets and arbitrary joins in opposite direction) form a category **Frame**, and locales are defined as objects of $\mathbf{Frame}^{op}$ (so a locale is really an equivalence class of frames under isomorphism, but conceptually a locale is "a space given by its lattice of opens").

There is a duality (in fact an equivalence of categories) between spatial locales (those coming from actual topological spaces via their open set lattices) and sober topological spaces, but in general locales generalize spaces (allowing "point-free" constructions). The slogan is "Topology = Locale = Frame^{op}."

In particular, the category **Stone** we saw is replaced in intuitionistic logic by the category **Loc** of locales (or perhaps specifically coherent spaces for coherent logic). Stone

duality generalizes to **Stone–Čech duality** or **Priestley duality** or in geometric form to **Stone duality for distributive lattices and frames**. The pattern remains: the algebra of propositions (like a Heyting algebra) corresponds contravariantly to a locale (its spectrum or space of "pseudo-characters").

We don't dive deeply here, but we mention:

Definition 2.7 (Frame and Locale)**.** A **frame** $(L, \leq)$ is a complete lattice (arbitrary joins and finite meets exist, with bottom 0 and top 1) satisfying the infinite distributive law:

$$a \wedge \bigvee_{i \in I} b_i = \bigvee_{i \in I} (a \wedge b_i) \quad \text{for any } a \in L, \{b_i\}_{i \in I} \subseteq L.$$

A frame homomorphism is a function $f : L \to M$ preserving finite meets and arbitrary joins (including 0 and 1). The category of frames is denoted **Frame**.

A **locale** is by definition an object of the category $\mathbf{Loc} := \mathbf{Frame}^{op}$. Concretely, a locale can be thought of as "a space" whose lattice of opens is a given frame. If X is a topological space, $\mathcal{O}(X)$ (the set of open sets of X) is a frame (with $\wedge = \cap$ and $\vee = \bigcup$). Not every frame is of that form for some X unless X is "sober", but one treats frames as generalized topologies.

A frame that is also a Boolean algebra (so each element has complement) is essentially the lattice of open sets of a Stone space, and that recovers our earlier Stone duality scenario as a special case (Boolean algebra = Stone locale).

The connection to logic: an intuitionistic propositional theory can be associated with a Lindenbaum–Tarski algebra which is a Heyting algebra (not necessarily complete). The space of all prime filters of that Heyting algebra can be given a topology (the Alexandroff topology or subbasic sets of form $\{p : a \in p\}$), yielding a topological space whose open sets corresponds to the theory's propositions. This is essentially a Stone-style representation for Heyting algebra, known as **Esakia duality** (if one includes an order on the space to represent $\leq$). Another approach is via **Kripke models**, which can be seen as particular frames (the frame of upward closed sets of a Kripke model yields a Heyting algebra semantics).

One result: the category of Heyting algebras is dual to the category of certain structured spaces called *Esakia spaces* (which are Stone spaces with a certain order, basically the duals of Heyting algebras that reflect the $\to$ structure). And frames are dual to *sober topological spaces*.

So summarizing logic vs. geometry:
- Classical Prop Logic: Boolean algebra $\leftrightarrow$ Stone space (set of models).
- Intuitionistic Prop Logic: Heyting algebra (frame of opens) $\leftrightarrow$ Locale (space of "points" or models possibly not all actual points).

We have now covered categories and logic for propositional fragments. Next, we move on to first-order logic and type theory, which require more categorical machinery (like adjoints for quantifiers, and dependent types or objects). Before that, we provide a couple of exercises to practice these concepts:

Exercise 2.8. 1. Show that the category of Boolean algebras **Bool** has all small products. (Hint: The product of a family of Boolean algebras $\{B_i\}$ can be taken as the algebra of tuples $\prod B_i$ with componentwise operations. One must show the resulting algebra satisfies the Boolean axioms and is the categorical product.)

2. Prove that a homomorphism $h : B \to 2$ from a Boolean algebra B to the 2-element algebra corresponds exactly to an ultrafilter of B. (Hint: $h^{-1}(1)$ is a filter that is maximal because the homomorphism can't map both b and $\neg b$ to 1.)

3. Verify the infinite distributive law in a powerset lattice: Given a set X, show that for any subset $A \subseteq X$ and any family $\{B_i \subseteq X \mid i \in I\}$, we have $A \cap \left(\bigcup_{i\in I} B_i\right) = \bigcup_{i\in I}(A \cap B_i)$. This shows $\mathcal{P}(X)$ is a frame.

4. Consider an infinite set of propositional variables $\{p_i \mid i \in \mathbb{N}\}$. Characterize the Stone space (dual space) of the free Boolean algebra on these generators. (Hint: points are ultrafilters = truth assignments; the space is homeomorphic to $\{0,1\}^{\mathbb{N}}$ with the product topology, a Cantor space.)

With propositional logic under our belt, we move on to considering predicates and type theory. This will involve categories that can capture not just finite products but also other structures such as exponentials (for implication or function types), and perhaps even more for full dependent types. That is our next chapter.

Chapter 3

First-Order Logic in Categories

First-order logic extends propositional logic by adding quantifiers that allow statements about elements of some domain of discourse. Categorical semantics of first-order logic requires categories that can interpret: - Conjunction and disjunction (like before, via products or other limits/colimits).
- Existential and universal quantifiers, which in category theory correspond to the existence of left and right adjoint functors to the pullback along projection maps (or existence of certain limits/colimits, specifically coequalizers, etc., to handle equivalence relations for equality).
- Equality, which is a special predicate requiring certain properties like diagonal morphisms and their mono nature.

In algebraic terms, first-order theories are no longer simply equational (if they have $\exists$ or $\forall$ axioms, or non-algebraic axioms), but we can often break them into sequents and consider categories called **classifying categories** or **syntactic categories** that capture their deductive structure.

There are two main categorical approaches to first-order logic: 1. The use of **syntactic categories (or syntactic sites)** as in the work of Makkai or the concept of *classifying topos* for a theory. Jacobs's approach using fibrations is one approach: interpret contexts and predicates in a fibered category (the category of contexts and substitutions is base, with a fiber of propositions above each context).
2. The approach using **adjoint functor semantics (Lawvere's hyperdoctrines)**: The idea that $\exists$ and $\forall$ quantifiers are adjoints to pullback along projections: specifically, in a category with finite limits (for context and equality), a functor Π_X that is right adjoint to pullback along $\pi_X : X \times Y \to Y$ can serve as $\forall x : X$, and a functor Σ_X left adjoint to that pullback can serve as $\exists x : X$.

The simplest categorical structure to consider is a **regular category**, which is a category with finite limits and images (stable under pullback). Regular categories correspond to first-order theories with only $\exists$ and $\wedge$ (and equality) but no $\vee$ or $\forall$ beyond the ones that can be expressed via $\exists$ (like a coherent theory restricts allowed formulas). A **coherent category** adds finite unions (disjunctions). A **Heyting category** (or arithmetic category) adds an internal $\Rightarrow$ or subobject classifier.

However, rather than define each of those from scratch, let's conceptualize: - The simplest semantics of first-order logic is given by **relational structures** in **Set**, i.e. models in the usual Tarskian sense. But category theory can treat these as functors from a syntactic category to **Set** that preserve certain limits (like product and equalizer).
- Instead of **Set**, one can consider models in arbitrary categories with similar structure. E.g., models in any topos, etc.

Lawvere (1963) introduced **functorial semantics of algebraic theories** (which we covered) and suggested one can do similar for more general theories via categories with certain structure.

The notion of **classifying topos** is the most powerful: for any first-order (even higher-order or type-theoretic) theory T satisfying some conditions, there is a topos $\mathcal{E}_T$ such that for any topos $\mathcal{E}$, models of T in $\mathcal{E}$ correspond to logical functors $\mathcal{E}_T \to \mathcal{E}$ (functors preserving the relevant structure, i.e., finite limits and whatever else needed). For example, the classifying topos of the theory of groups is a topos $\mathcal{E}$ such that giving a group in any topos $\mathcal{F}$ is equivalent to giving a logical functor $\mathcal{E} \to \mathcal{F}$. When the topos is **Set**, such a functor picks out a group in **Set**, recovering ordinary models.

Although classifying toposes are conceptually elegant, constructing them directly can be heavy. Instead, we can proceed stepwise: - Interpret first-order logic in a fixed topos (like **Set**) using the internal logic of the topos. This internal logic approach says: a topos can serve as a universe in which logical formulas can be interpreted, with $\wedge, \vee, \exists, \forall$ having meaning via categorical operations (subobjects, products, etc., and a subobject classifier Ω for truth values). If the topos is boolean (like **Set**) this internal logic is classical; if it's an arbitrary topos, it's intuitionistic.
- But we want an abstract notion of "category that behaves like formulas and contexts."

We introduced in Jacobs: - **Fibred category semantics**: e.g., a **hyperdoctrine** is a functor $\mathcal{P} : \mathcal{C}^{op} \to$ **Poset** that gives for each object A (a context) a poset of predicates on A, with appropriate structure (like left and right adjoints for each substitution functor) capturing $\exists$ and $\forall$.

One can show equivalently that giving such a structure is giving a first-order logical

calculus semantics. Jacobs defines things like *regular categories* and shows the subobject functor of a regular category is a hyperdoctrine with left adjoints to pullbacks (for $\exists$). If also right adjoints exist, it's a *coherent category* (with $\forall$ for certain monos) or Heyting category (with all $\forall$). This ties to Π and Σ in type theory as well.

However, due to the complexity, let's outline a simpler vantage:

Classifying category construction: For a first-order theory T (with function and relation symbols and axioms), one can form a category $\mathcal{C}_T$ whose: - objects are contexts (finite lists of variables, or possibly formulas considered as types in dependent type parlance).
- morphisms are interpretation of sequents or substitutions between contexts (like a morphism from context Γ to Δ could be given by a tuple of terms Δ in context Γ that is *provably* making Δ imply something? Actually, simpler: a morphism from Γ to Δ can be taken as an equivalence class of Δ-indexed tuples of terms or formulas in Γ, capturing how to get a Δ instance from each Γ instance.
One can do this systematically by taking the opposite of the category of theories as in syntactic fibration:
Jacobs had something like: **Subobject fibrations**: For each context Γ, consider the set of formulas (predicates) with free variables in Γ; they form a poset of subobjects of Γ in some classifying category.
So one defines $\mathcal{C}_T$ (the category of contexts and provable substitutions) and a functor $p : \mathcal{E}_T \to \mathcal{C}_T$ such that each fiber is the poset of "provable predicates" on that context. That is basically the syntactic fibration which is a hyperdoctrine.
Then $\mathcal{E}_T$ might turn out to be a topos if T is geometric or something.

Given the time, let's articulate results: - If T is a coherent theory (allowing $\wedge, \vee, \exists$ but not full $\forall$ except maybe for equality), then the classifying category of T is a coherent category (regular plus with some finite colimits).
- If T is a first-order theory (with possibly all quantifiers), and say it is essentially equational or something, we might need to go to a topos with a subobject classifier (the classifying topos that ensures existence of $\forall$ as well).
- There's the notion of **regular category** for regular logic and **coherent category** for coherent logic, **Heyting category** or **logos** for full first-order (with all finite limits and power object maybe). Jacobs calls a coherent category with a certain property a *logos* (which is basically a pretopos).

Anyway, to ground with an example:

Example 3.1 (Classifying category of groups (sketch))**.** Consider the theory of groups in

first-order logic:
- Sort: one sort (the domain of group elements).
- Function symbols: $m(x, y)$ (multiplication), $e()$ (unit), $(-)^{-1}$ (inverse).
- Axioms: $\forall x, y, z.\ (x \cdot y) \cdot z = x \cdot (y \cdot z)$, $\forall x.\ x \cdot e = x$, $e \cdot x = x$, $\forall x.\ x^{-1} \cdot x = e = x \cdot x^{-1}$.
The classifying category $\mathcal{C}_T$ will have one object **1** for a single variable context (or think of objects as 1 for one generator, 2 for two generators, etc. effectively like Lawvere theory but now with axioms). However, the difference from Lawvere theory: those axioms are $\forall$ statements, which means in the category they impose that certain diagrams commute or certain subobjects are equivalences.
One can construct $\mathcal{C}_T$ as follows:
- Objects: natural numbers n (for n distinct variables).
- Morphisms: an arrow $n \to m$ is given by an m-tuple of terms in n variables (just like Lawvere theory morphism), *but* under an equivalence relation generated by the axioms not just equational but also some involving existence of inverses, etc.
However, because the axioms are all equations with universal quantifiers, they translate to the statement that certain parallel pairs of arrows in $\mathcal{C}_T$ coequalize. For example, the axiom $\forall x.(e \cdot x = x)$ presumably means the two morphisms from 1 to 1 given by $e \cdot x$ and x (seen as terms in one variable) are equal in the category. Similarly, the inverse axiom means: the composite $1 \xrightarrow{x} 1 \xrightarrow{m(x^{-1},x)} 1$ equals the morphism $1 \xrightarrow{e()} 1$.
So effectively $\mathcal{C}_T$ is like the Lawvere theory of groups (which is a category with finite products) but now with the requirement that those morphisms become equal, which forces some coequalizers. In fact, the classifying category of groups is more precisely a **pretopos** (finite limits + effective equivalence relations + coequalizers) because quotienting by the normal subgroup generated by those relations is needed.
The classifying topos of groups would be a further completion of that category into a topos (adding a subobject classifier).
But one can already reason in $\mathcal{C}_T$: models of the group theory in any regular category correspond to product-preserving functors $\mathcal{C}_T \to \mathcal{C}$ that also take those coequalizer diagrams to coequalizers in $\mathcal{C}$.
This example is complex but key insight: $\mathcal{C}_T$ has one object 1 (for one variable) and the structure makes it a *group object in the topos* $\widehat{\mathcal{C}_T}$ (the topos of presheaves on $\mathcal{C}_T$). That representability yields the classifying topos property.

To formalize the presence of quantifiers:
- $\exists x.\Phi(x, y)$ corresponds to a left adjoint to pullback along projection $\pi : X \times Y \to Y$: $\Sigma_X : \mathrm{Sub}(X \times Y) \to \mathrm{Sub}(Y)$ sending a subobject $U \rightarrowtail X \times Y$ to its direct image $\exists X.U$ in Y. A regular category ensures existence of these left adjoints (for predicates, indeed images of monos).
- $\forall x.\Phi(x, y)$ corresponds to a right adjoint Π_X to that pullback (this often requires a condition like the subobject is "definable"), which is available in a coherent category with a

Heyting structure or in a topos with a subobject classifier (where Π can be constructed using Σ and $\neg$ since $\forall x.\Psi$ is $\neg\exists x.\neg\Psi$ in classical logic).
- Up to issues with intuitionistic logic, one might only have $\neg$ as pseudo-complement.

In Jacobs (chapter on first-order logic) he introduces categories called **regular**, **coherent**, and **(geometric) theory categories** and shows results like:
"If $\mathcal{C}$ is a regular category and $f : X \to Y$ a morphism, then pulling back along f has a left adjoint $\exists_f$ (the direct image functor), and these satisfy the expected properties for existential quantifiers."

And:
"In a coherent category (regular + finite unions) one can also interpret $\vee$ and $\top$ properly." Similarly, "In a Heyting category (coherent + exponentials for subobjects), one can interpret $\forall$." Jacobs basically builds the semantics of first-order logic in any Heyting category, culminating in completeness results like if a sequent is valid in all models in all Heyting categories (or all toposes), it is provable.

One remarkable fact:
Gödel's completeness theorem (for classical first-order logic) can be seen as a special case of the existence of a classifying topos (or category): If a sentence is true in all set-based models, it is provable. We already saw a part of that for propositional. The first-order version uses ultraproducts or syntactic category like constructing a term model.

We have to be careful with $\forall$: For completeness one uses Henkin's trick of adding constants to witness existential assumptions, effectively turning it into an equational theory or rather a coherent theory (this ensures we can find a model when consistent).

From the category side, one often restricts to **geometric logic** (which allows $\exists, \wedge, \vee$ but not $\forall$ except over finite sets of equations) because geometric logic is preserved under the inverse image part of geometric morphisms between toposes, and a classifying topos exists for geometric theories. For full first-order (with $\forall$), completeness is not constructively valid but classically we still have it.

We have gone quite theoretical. Let's conclude this chapter with a summary:

- Categories with finite limits can interpret conjunction and equality.
- Add stable images (regular category) to interpret existential quantifiers.
- Add finite joins (coherent category) for disjunction.
- Add a subobject classifier (topos, or at least Heyting operations) to interpret universal quantifiers.
- The internal logic of a topos is an intuitionistic higher-order logic in which these inter-

pretations make the usual logical rules sound and complete.

Thus, category theory provides a language to talk about logic where: - Objects of a category are like types or domains.
- Morphisms are like terms or functions.
- Special objects and morphisms (like subobjects, product projections) represent predicates and logical connectives.
- Functors between categories represent interpretations or models.

The Curry–Howard correspondence is one step up: between logic and type theory, which we consider next.

3.1 Categorical Semantics of Type Theory

We now turn to **type theory**. There are various type theories; the simplest is the simply-typed λ-calculus with function types (arrow types) and possibly product types. More complex ones include dependent types (e.g., a type of all vectors of length n depends on a term n of type $\mathbb{N}$) and polymorphic types.

A fundamental result by Lambek (and Howard earlier informally) is:
- The category of contexts and provable function terms in simply-typed lambda calculus is a Cartesian closed category (CCC).
- Conversely, any CCC gives rise to a simply-typed lambda calculus as its internal language.

This is often called the **Curry–Howard–Lambek correspondence**:

Simply-typed λ-calculus $\equiv$ Cartesian Closed Category $\equiv$ Propositional logic with implication.

In that correspondence: - A type A corresponds to an object in a CCC.
- A term of type B in context $x : A$ corresponds to a morphism $A \to B$ in the category.
- The function type $A \Rightarrow B$ corresponds to an exponential object B^A in the category (the internal hom).
- Application corresponds to evaluation morphism $B^A \times A \to B$; abstraction (lambda) corresponds to the curry (adjunct) of evaluation: a morphism $X \times A \to B$ corresponds bijectively to a morphism $X \to B^A$.

Thus, in this correspondence:
- A type A corresponds to an object in a CCC.

- A term of type B in context $x : A$ corresponds to a morphism $A \to B$ in the category.
- The function type $A \Rightarrow B$ corresponds to an exponential object B^A in the category (the internal hom).
- Application corresponds to evaluation morphism $B^A \times A \to B$; abstraction (lambda) corresponds to the curry (adjunct) of evaluation: a morphism $X \times A \to B$ corresponds bijectively to a morphism $X \to B^A$.

Theorem (Curry–Howard–Lambek). The simply-typed lambda calculus with product types and a base type corresponds exactly to the theory of Cartesian closed categories with a distinguished object. More precisely, there is an equivalence between:

- The syntactic category of the lambda calculus (with contexts as objects, terms as morphisms).
- The free Cartesian closed category generated by the base types (and operations if any).

And models of the lambda calculus correspond to functors from this free CCC to any CCC (much like the algebraic theories case).

A consequence: If a formula (type) is inhabited (i.e., there is a term of that type) in the type theory, then it is true in all CCC-models (soundness), and if it's true in all models, it's inhabited (completeness). This is analogous to completeness for logic, but in type theory it's often called *strong normalization implies consistency, etc.*

Let's break it down:
- A **Cartesian Closed Category (CCC)** is a category with finite products and exponentials for each pair of objects. Finite products give a terminal object (type of trivially true proposition) and binary products (pair types). Exponentials give function types.
- The **internal language** of a CCC is a simply-typed lambda calculus: each morphism $f : A \to B$ can be seen as a lambda term $\lambda(x : A).t$ of type B using one free variable of type A, and composition corresponds to substitution/application.

For example, in **Set** (which is CCC), an exponential B^A is the set of all functions $A \to B$. A specific function $g : X \to B^A$ is equivalently a function $X \times A \to B$ (by currying). That is exactly like having a term $f(x, a)$ with x from X and a from A yielding a B; currying means we regard it as the function $x \mapsto (a \mapsto f(x, a))$.

So, in **Set**, a lambda term $\lambda x^A.t(x)$ is interpreted as the actual function mapping each $a \in A$ to the interpretation of $t(a)$ in B. The beta-reduction corresponds to plugging in

that element.

Simply-typed lambda calculus as initial model: We can treat the syntax of lambda calculus as a category (the term model) and any CCC as a model. Then:

Theorem 3.2. *The term model (syntactic CCC) is the free CCC on the set of base types (and constants). Thus, any equation between terms that holds in all CCC models is provable in the lambda calculus (completeness), and if it's provable, it holds in all models (soundness). In particular, if there is no term of type $\bot$ (an empty type) in the calculus, then no CCC can give an arrow $1 \to \bot$ (which means the calculus is consistent: can't derive an inhabitant of the empty type).*

The proof is analogous to the algebraic one: construct the syntactic category and show it is initial among CCCs. A consequence is the **lambda calculus is consistent** because a term of type $\bot$ in the term model would mean a morphism $1 \to \bot$ in the initial CCC, which then would exist in **Set** under any interpretation, implying a set of 0 elements has an element, contradiction.

The above covers simply typed (no polymorphism, no dependent types, no higher-order).
Polymorphism (System F) leads to models in categories with additional structure (like *dinatural transformations*, not obviously just an ordinary category property, but something like a universe object etc).
Dependent types lead to **categories with families (CwF)** or **categories with fibrations** that correspond to the type theory (these generalize CCCs: - A dependent type $B(x)$ over $x : A$ is like an object in a slice category $\mathcal{C}/A$. A category with families provides such a thing plus substitution operations).
The simplest such structure is a **Locally Cartesian Closed Category (LCCC)** which has nice interpretations for dependent Π and Σ (the rules for these correspond to adjointness of pullbacks, as mentioned for quantifiers). For example: - A locally cartesian closed category (LCCC) is one where every slice category $\mathcal{C}/X$ is cartesian closed. That corresponds to having Π types (dependent function types) and Σ types (dependent sum types) and is the categorical model of dependent type theory without universes or higher inductive types.
Martin-Löf type theory with universes is modeled in a **category with a universe object** or a **Grothendieck fibration with some additional structure**. Jacobs covers universes as well, likely showing something like: "If a category has a universe (an object U with a surjection $El \to U$ that classifies small objects) then one can interpret an extra type of types."

The interaction of logic and type theory:
- Propositions-as-Types: in type theory with a type Prop of propositions, each proposition is a type whose inhabitants are proofs. In a category, this is like having a subobject classifier Ω (the object of propositions), and subobjects correspond to monomorphisms or characteristic functions into Ω.
In categories:
- A Ω as subobject classifier in a topos is analogous to a type of propositions where an element of Ω is a truth value.
- A predicate P on X is a morphism $X \to \Omega$.
So higher-order logic (where we quantify over predicates) corresponds to a type theory with an Ω type that one can quantify over—that essentially pushes to the realm of toposes or at least logical categories with power-objects.

Finally, summarizing Curry-Howard:
- Simply typed with $\to, \times, 1$ corresponds to CCC.
- Add a base type $\mathbf{0}$ (empty type) or a type for false, that would correspond to an initial object or subobject classifier phenomenon.
- Dependent types (Π, Σ) corresponds to categories that are locally cartesian closed (for Π) and have sums (for Σ), which are basically toposes (a topos has these plus a subobject classifier, making it a model of full higher-order logic).

So the progression:
Algebraic theories (no logic, just equations) $\to$ categories with finite products.
Propositional logic (Boolean) $\to$ Boolean algebra or Stone spaces.
First-order logic $\to$ regular, coherent categories or toposes.
Type theory $\to$ cartesian closed (for simple) $\to$ locally cartesian closed (for dependent) $\to$ toposes with natural number object etc (for full Martin-Löf with inductive types).

And at each stage, there is a completeness: if something is true in all such categories (models), it's derivable in the logic.

We conclude with one more exercise:

Exercise 3.3. In a Cartesian closed category $\mathcal{C}$, consider an object A and an element (morphism from 1) $a : 1 \to A$. Show that evaluation at a gives a natural isomorphism $\mathcal{C}(1, B^A) \cong \mathcal{C}(A, B)$ by mapping a morphism $f : 1 \to B^A$ to $eval \circ (f \times a) : 1 \times A \to B$. Use this to argue that a term t of type B with a free variable of type A corresponds uniquely to a lambda term $\lambda a.t$ of type $A \Rightarrow B$. (This reflects the currying bijection.)

With that, we have outlined how category theory and logic/type theory inform each other: categories provide models and structural insights into logic, while logic provides

syntax and computational content for categories. The unity of these fields is one of the beautiful aspects of categorical logic and type theory.

Appendix A

Further Reading

Due to the breadth of categorical logic and type theory, we can only scratch the surface in one textbook. For readers seeking more:
- Samson Abramsky and Nikos Tzevelekos's introductory chapter provides a concise starting point.
- Bart Jacobs's *Categorical Logic and Type Theory* covers advanced topics like fibrations for predicate logic, internal category theory, and topos semantics in depth.
- Steve Awodey's lecture notes offer accessible treatments of algebraic and categorical logic, including Lawvere theories and Stone duality.
- On type theory and categories, see *Categories for Types* by Crole and *Practical Foundations of Mathematics* by Johnstone for connections between type theory and topos theory.
- For Stone duality and topology related to logic, Johnstone's *Stone Spaces* and *Sketches of an Elephant* are comprehensive references.

We hope this textbook has equipped the reader with a foundational understanding to approach both classical texts and research literature in categorical logic and type theory.

Bibliography

[1] Abramsky, Samson, and Nikos Tzevelekos. 2011. "Introduction to Categories and Categorical Logic," arXiv:1102.1313.

[2] Awodey, Steve. 2022. *Categorical Logic* (Autumn School Lecture Notes).

[3] Awodey, Steve, and Andrej Bauer. 2024. "Introduction to Categorical Logic (Draft)."

[4] Jacobs, Bart. 1999. *Categorical Logic and Type Theory*. Elsevier Science.

[5] Lawvere, F. William. 1963. "Functorial Semantics of Algebraic Theories." *Proceedings of the National Academy of Sciences* 50(5): 869–872.

[6] Mac Lane, Saunders, and Ieke Moerdijk. 1992. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. Springer.

[7] Johnstone, Peter T. 1982. *Stone Spaces*. Cambridge Studies in Advanced Mathematics.