An Introduction to Category Theory

The Mathematics of Structure

Said BOUDJELDA

Paris pantheon-assas University

May 4, 2025

Outline

Why Category Theory?

- Originally developed in algebraic topology (1940s)
- Provides a unifying language for mathematics
- Reveals deep connections between seemingly different areas
- Abstracts common patterns across mathematical structures
- Has become increasingly important in:
  - Computer science
  - Physics
  - Logic and foundations

Historical Background: Before Categories

- Mathematics was becoming increasingly abstract (1900-1940)
- Emmy Noether's work on abstract algebra (1920s)
  - Shifted focus from specific instances to structural properties
  - Emphasized isomorphisms over concrete representations
- Topology developing rapidly
  - New methods needed to classify topological spaces
  - Algebraic invariants being developed (homology, cohomology)
- Need for precise language to describe "naturality"
  - Mathematicians using informal notions of "natural" mappings
  - No formal definition of what "natural" meant

Birth of Category Theory

- Samuel Eilenberg (topologist) and Saunders Mac Lane (algebraist)
- Collaboration began in 1941 at University of Michigan
- Working on algebraic topology problems
- Needed a way to formalize "natural isomorphisms"
- First paper: "General Theory of Natural Equivalences" (1945)
  - Introduced categories, functors, and natural transformations
  - Initially seen as a language for discussing existing concepts
- Quote from Mac Lane (1996):

  "We needed to understand natural transformations. In order to do that, we were led to formulate functors. In order to formulate functors, we needed categories."

Early Development (1945-1957)

- Initially received with skepticism
  - Derisively called "abstract nonsense" or "general abstract nonsense"
  - Term later embraced by category theorists
- Henri Cartan and Samuel Eilenberg
  - Book "Homological Algebra" (1956)
  - Used categorical language to unify and clarify concepts
- Daniel Kan (1958)
  - Introduced adjoint functors
  - Significant advancement in categorical thinking
- Initial applications mainly in algebraic topology and homological algebra

The Grothendieck Revolution (1957-1970)

- Alexander Grothendieck transformed algebraic geometry
  - Introduced the concept of "scheme" via categorical methods
  - Formulated functorial approach to algebraic geometry
- Tohoku paper (1957): "Sur quelques points d'algèbre homologique"
  - Revolutionized homological algebra using categorical methods
  - Introduced abelian categories
- Grothendieck's seminar notes (1960s)
  - Expanded category theory from a language to a foundation
  - Developed theory of topoi and descent theory
- Showed category theory could solve concrete mathematical problems

Category Theory Matures (1970s-1980s)

- Saunders Mac Lane's "Categories for the Working Mathematician" (1971)
  - First comprehensive textbook on category theory
  - Established category theory as independent discipline
- Peter Freyd and Max Kelly's contributions
  - Formalization of enriched categories
  - Development of categorical universal algebra
- William Lawvere's work
  - Functorial semantics for algebraic theories
  - Elementary topos theory as foundation for mathematics
- Jean Bénabou's work on bicategories and fibrations
- Application to logic through categorical semantics

Recent Developments (1990s-Present)

- Expansion into computer science
  - Eugenio Moggi's work on monads for programming languages
  - Practical applications in functional programming (Haskell)
- Higher category theory
  - André Joyal, Ross Street, John Baez contributions
  - Development of n-categories and ∞-categories
- Homotopy Type Theory
  - Vladimir Voevodsky's univalent foundations program
  - Synthesis of type theory and homotopy theory
- Applied category theory
  - David Spivak, Brendan Fong, John Baez
  - Applications to networks, databases, and complex systems

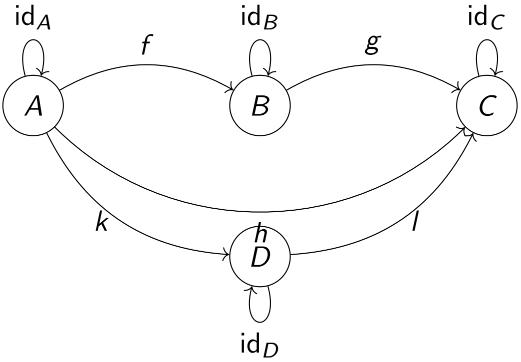
From Sets to Categories: A Shift in Perspective

- Traditional mathematics: Focus on objects and their internal structure
  - Sets with elements
  - Groups with operations
  - Spaces with points
- Categorical perspective: Focus on relationships between objects
  - How objects relate to each other
  - Properties defined by patterns of relationships
  - Internal structure becomes secondary
- Shift in thinking:
  - From "What is it?" to "How does it behave?"
  - From intrinsic properties to extrinsic relationships
  - From elements to morphisms

Philosophical Significance of Categories

- Categories provide a "structural" view of mathematics
  - Objects defined by their relationships, not internal constituents
  - Isomorphic objects are essentially "the same"
  - Structure preserved under transformation is what matters
- Reflects structuralist philosophy in mathematics
  - Mathematics studies patterns and relationships
  - Content is secondary to structure
- Beyond set theory:
  - Alternative to set-theoretic foundations
  - Captures mathematical practice more faithfully
  - Avoids paradoxes of naive set theory

Visualizing Categories



- Categories can be visualized as directed graphs with identity loops
- Composition is represented by paths: $g \circ f$ is a path from A to C
- Associativity ensures different path traversals yield the same result
- Identity morphisms are loops that don't change paths when composed

What is a Category?

Definition

A **category** $\mathcal{C}$ consists of:

- A collection of **objects**: $\mathrm{Ob}(\mathcal{C})$
- For each pair of objects A, B, a collection of **morphisms** (or arrows): $\mathrm{Hom}_{\mathcal{C}}(A, B)$
- For each object A, an **identity morphism**: $\mathrm{id}_A : A \to A$
- A **composition operation** for morphisms: $\circ$

satisfying:

- **Associativity**: $(h \circ g) \circ f = h \circ (g \circ f)$
- **Identity laws**: $\mathrm{id}_B \circ f = f = f \circ \mathrm{id}_A$ for $f : A \to B$

Examples of Categories - Part 1

Set - Category of sets

- Objects: Sets
- Morphisms: Functions
- Composition: Function composition
- Identity: Identity function

Grp - Category of groups

- Objects: Groups
- Morphisms: Group homomorphisms
- Composition: Function composition
- Identity: Identity homomorphism

Top - Category of topological spaces

- Objects: Topological spaces
- Morphisms: Continuous maps
- Composition: Function composition
- Identity: Identity map

$\mathbf{Vect}_K$ - Category of vector spaces

- Objects: Vector spaces over field K
- Morphisms: Linear transformations
- Composition: Function composition
- Identity: Identity transformation

Examples of Categories - Part 2

Ring - Category of rings

- Objects: Rings
- Morphisms: Ring homomorphisms
- Composition: Function composition

Cat - Category of small categories

- Objects: Small categories
- Morphisms: Functors
- Composition: Functor composition

Pos - Category of partially ordered sets

- Objects: Posets
- Morphisms: Order-preserving functions
- Composition: Function composition

Hask - Category of Haskell types

- Objects: Haskell types
- Morphisms: Functions between types
- Composition: Function composition

More Examples of Categories

- **Discrete categories**:
  - Only identity morphisms
  - Example: Sets with only identity functions
- **Monoids as categories**:
  - A monoid can be seen as a category with just one object
  - Morphisms correspond to monoid elements
  - Composition corresponds to monoid operation
- **Preorders as categories**:
  - Objects are elements of the preorder
  - Single morphism $a \to b$ exists iff $a \leq b$
  - Composition follows from transitivity
- ***n* as a category**:
  - Objects are numbers $0, 1, 2, \ldots, n-1$
  - Morphism $i \to j$ exists iff $i \leq j$

Special Types of Categories

- **Small category**: Objects and morphisms form sets (not proper classes)
  - Example: Any finite category is small
- **Locally small category**: For any objects A, B, the morphisms $\mathrm{Hom}(A, B)$ form a set
  - Example: Most familiar categories (Set, Grp, Top)
- **Discrete category**: Only identity morphisms
  - Example: Sets with only identity functions
- **Indiscrete/Codiscrete category**: Exactly one morphism between any two objects
  - Example: Sets where all elements are related
- **Thin category**: At most one morphism between any two objects
  - Example: Posets as categories
- **Groupoid**: Every morphism is invertible
  - Example: Fundamental groupoid of a topological space

Special Types of Morphisms - Part 1

Monomorphism (Mono)

- Left-cancellative: $f \circ g_1 = f \circ g_2 \Rightarrow g_1 = g_2$
- Generalizes injective functions
- Visual: No merging of elements
- In Set: Precisely the injective functions

Epimorphism (Epi)

- Right-cancellative: $g_1 \circ f = g_2 \circ f \Rightarrow g_1 = g_2$
- Generalizes surjective functions
- Visual: Covers all elements
- In Set: Precisely the surjective functions

Note: Monomorphism and epimorphism are dual concepts

Special Types of Morphisms - Part 2

Isomorphism (Iso)

- Has an inverse: $f \circ g = \text{id}$ and $g \circ f = \text{id}$
- Generalizes bijective functions
- Objects related by an isomorphism are "essentially the same" from a categorical perspective
- In Set: Precisely the bijective functions

Endomorphism

- Morphism from an object to itself: $f : X \to X$
- Generalizes functions from a set to itself
- Example: Linear operators on a vector space
- Set of all endomorphisms forms a monoid under composition

Automorphism

- Isomorphism from an object to itself
- The set of all automorphisms of an object forms a group
- Example: Group of symmetries

More Special Types of Morphisms

- **Section/Right Inverse**:
  - Morphism $s : B \to A$ such that $f \circ s = \mathrm{id}_B$
  - In Set: Right inverse exists iff f is surjective
- **Retraction/Left Inverse**:
  - Morphism $r : B \to A$ such that $r \circ f = \mathrm{id}_A$
  - In Set: Left inverse exists iff f is injective
- **Bimorphism**:
  - Both a monomorphism and an epimorphism
  - Not necessarily an isomorphism (unlike in Set)
- **Zero morphism**:
  - In categories with zero objects, a morphism that factors through the zero object
  - Example: Zero matrix in Vect_K

Natural Transformations

Definition

A **natural transformation** $\eta : F \Rightarrow G$ between functors $F, G : \mathcal{C} \to \mathcal{D}$ consists of:

- For each object X in $\mathcal{C}$, a morphism $\eta_X : F(X) \to G(X)$ in $\mathcal{D}$

such that for every morphism $f : X \to Y$ in $\mathcal{C}$, the following diagram commutes:

$$\begin{array}{ccc} F(X) & \xrightarrow{F(f)} & F(Y) \\ {\scriptstyle \eta_X}\downarrow & & \downarrow{\scriptstyle \eta_Y} \\ G(X) & \xrightarrow[G(f)]{} & G(Y) \end{array}$$

(i.e., $\eta_Y \circ F(f) = G(f) \circ \eta_X$)

Natural Isomorphisms

Definition

A **natural isomorphism** is a natural transformation $\eta : F \Rightarrow G$ where each component η_X is an isomorphism.

$$\begin{tikzcd} F(X) \arrow[r, "F(f)"] \arrow[d, "\eta_X"', "\cong"] & F(Y) \arrow[d, "\eta_Y", "\cong"'] \\ G(X) \arrow[r, "G(f)"'] & G(Y) \end{tikzcd}$$

- Represents a structure-preserving equivalence between functors
- Captures when two functorial constructions are "essentially the same"
- Example: For finite-dimensional vector spaces V, the natural isomorphism $V \cong (V^*)^*$

Universal Properties

- Define objects by how they relate to other objects
- Characterize objects up to unique isomorphism
- Examples:
  - Products and coproducts
  - Equalizers and coequalizers
  - Pullbacks and pushouts
  - Initial and terminal objects
- Unify constructions across different categories

Products and Coproducts

Product of A and B:

$$\begin{tikzcd}
 & P \arrow[dl, "p_1"'] \arrow[dr, "p_2"] & \\
A & & B
\end{tikzcd}$$

With universal property:

$$\begin{tikzcd}
 & X \arrow[dl, "f"'] \arrow[d, dashed, "\exists! u"] \arrow[dr, "g"] & \\
A & P \arrow[l, "p_1"] \arrow[r, "p_2"'] & B
\end{tikzcd}$$

Coproduct of A and B:

$$\begin{tikzcd}
 & C & \\
A \arrow[ur, "i_1"] & & B \arrow[ul, "i_2"']
\end{tikzcd}$$

With universal property:

$$\begin{tikzcd}
 & C \arrow[d, dashed, "\exists! u"] & \\
A \arrow[ur, "i_1"] \arrow[dr, "f"'] & {} & B \arrow[ul, "i_2"'] \arrow[dl, "g"] \\
 & X &
\end{tikzcd}$$

Examples of Products and Coproducts

Products

- **Set**: Cartesian product with projections
- **Grp**: Direct product with projections
- **Top**: Product space with projections
- $\textbf{Vect}_K$: Direct sum with projections

Coproducts

- **Set**: Disjoint union with inclusions
- **Grp**: Free product with inclusions
- **Top**: Disjoint union with inclusions
- $\textbf{Vect}_K$: Direct sum with inclusions

Limits and Colimits

- **Diagram**: A functor $D : \mathcal{J} \to \mathcal{C}$ from a small index category
- **Cone**: An object X with morphisms to each object in the diagram, commuting with diagram arrows
- **Limit**: The universal cone
- **Colimit**: The universal cocone (dual notion)

Examples of limits:

- Terminal object: Limit of empty diagram
- Product: Limit of discrete diagram
- Equalizer: Limit of parallel arrows
- Pullback: Limit of span diagram

Adjunctions

Definition

An **adjunction** between categories $\mathcal{C}$ and $\mathcal{D}$ consists of functors $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{D} \to \mathcal{C}$, together with a natural bijection:

$$\mathrm{Hom}_{\mathcal{D}}(F(A), B) \cong \mathrm{Hom}_{\mathcal{C}}(A, G(B))$$

for all objects A in $\mathcal{C}$ and B in $\mathcal{D}$.
We write $F \dashv G$ and say "F is left adjoint to G" or "G is right adjoint to F".

Adjunctions: Unit and Counit

Equivalently, an adjunction can be defined by two natural transformations:

- **Unit**: $\eta : 1_{\mathcal{C}} \Rightarrow G \circ F$
- **Counit**: $\varepsilon : F \circ G \Rightarrow 1_{\mathcal{D}}$

Satisfying the triangle identities:

$$\begin{tikzcd} F \arrow[r, "F\eta"] \arrow[dr, equal] & FGF \arrow[d, "\varepsilon F"] \\ & F \end{tikzcd} \qquad \begin{tikzcd} G \arrow[r, "\eta G"] \arrow[dr, equal] & GFG \arrow[d, "G\varepsilon"] \\ & G \end{tikzcd}$$

Examples of Adjunctions

- **Free-Forgetful adjunction**:
  - Free functor $F : \mathbf{Set} \to \mathbf{Grp}$
  - Forgetful functor $U : \mathbf{Grp} \to \mathbf{Set}$
  - $F \dashv U$
- **Product-Diagonal adjunction**:
  - Diagonal functor $\Delta : \mathbf{Set} \to \mathbf{Set} \times \mathbf{Set}$
  - $- \times A \dashv \Delta$
- **Currying adjunction**:
  - $- \times B \dashv (-)^B$ in cartesian closed categories
  - Captures currying in lambda calculus
- **Stone-Čech compactification**:
  - Left adjoint to the forgetful functor from compact Hausdorff spaces to topological spaces
- **Galois connections**:
  - Special case of adjunctions between poset categories

Monads

Definition

A **monad** on a category $\mathcal{C}$ consists of:

- An endofunctor $T : \mathcal{C} \to \mathcal{C}$
- A unit natural transformation $\eta : 1_{\mathcal{C}} \Rightarrow T$
- A multiplication natural transformation $\mu : T^2 \Rightarrow T$

satisfying the following coherence conditions:

(Associativity)
$$\begin{tikzcd} T^3 \arrow[r, "T\mu"] \arrow[d, "\mu T"] & T^2 \arrow[d, "\mu"] \\ T^2 \arrow[r, "\mu"] & T \end{tikzcd}$$

(Unit laws)
$$\begin{tikzcd} T \arrow[r, "T\eta"] \arrow[dr, equal] & T^2 \arrow[d, "\mu"] \\ & T \end{tikzcd}$$

$$\begin{tikzcd} T \arrow[r, "\eta T"] \arrow[dr, equal] & T^2 \arrow[d, "\mu"] \\ & T \end{tikzcd}$$

Examples of Monads

- **List monad**:
  - $T(X) =$ lists of elements from X
  - $\eta(x) = [x]$ (singleton list)
  - μ concatenates lists of lists
- **Maybe monad**:
  - $T(X) = X \cup \{\text{Nothing}\}$
  - Models computations that might fail
- **State monad**:
  - $T(X) = (S \times X)^S$
  - Models stateful computations
- **Continuation monad**:
  - $T(X) = R^{(R^X)}$
  - Models continuation-passing style
- **Free monad**:
  - Generates free algebraic structures

Applications in Mathematics

- **Algebraic topology**:
  - Fundamental groups, homology, cohomology
  - Spectral sequences
- **Algebraic geometry**:
  - Schemes, sheaves, stacks
  - Grothendieck topologies
- **Homological algebra**:
  - Derived functors, Ext and Tor
  - Abelian categories
- **Logic**:
  - Categorical semantics
  - Topos theory as foundation

Applications in Computer Science

- **Functional programming**:
  - Monads for effects
  - Functors, applicatives
- **Type theory**:
  - Categorical semantics of types
  - Adjunctions in type constructors
- **Databases**:
  - Categorical query languages
  - Functorial data migration
- **Concurrency**:
  - Process algebras as categories
  - Monoidal categories for resources

Applications in Physics

- **Quantum mechanics**:
  - Monoidal categories for tensor products
  - String diagrams for quantum processes
- **Quantum field theory**:
  - Topological quantum field theories
  - Cobordism categories
- **General relativity**:
  - Categories of spacetimes
  - Categorical formulation of observables

Summary

- Category theory provides a powerful language for mathematical structures
- Focuses on relationships rather than internal details
- Unifies concepts across different mathematical fields
- Provides tools for abstraction and generalization
- Has found applications beyond pure mathematics

Further Reading

- **Introductory**:
  - "Category Theory for Programmers" - Bartosz Milewski
  - "Conceptual Mathematics" - Lawvere and Schanuel
- **Intermediate**:
  - "Categories for the Working Mathematician" - Mac Lane
  - "Basic Category Theory" - Tom Leinster
- **Advanced**:
  - "Sheaves in Geometry and Logic" - Mac Lane and Moerdijk
  - "Higher Topos Theory" - Jacob Lurie
- **Applications**:
  - "Physics, Topology, Logic and Computation" - Baez and Stay
  - "Seven Sketches in Compositionality" - Fong and Spivak