1. Category Theory Basics

1.1. Overview

Category theory provides a unifying language for mathematics, abstracting the notion of structure-preserving mappings between mathematical objects. It reveals deep connections between seemingly disparate areas of mathematics and computer science. This module introduces categories, functors, natural transformations, and fundamental constructions.

1.2. Categories

1.2.1. Definition

A **category** $\mathcal{C}$ consists of:
1. A collection of **objects** $\mathrm{Ob}(\mathcal{C})$
2. For each pair of objects X, Y, a collection of **morphisms** $\mathrm{Hom}(X, Y)$
3. For each object X, an **identity morphism** $\mathrm{id}_X \in \mathrm{Hom}(X, X)$
4. **Composition**: For morphisms $f : X \to Y$ and $g : Y \to Z$, a morphism $g \circ f : X \to Z$

Satisfying:
- **Associativity**: $(h \circ g) \circ f = h \circ (g \circ f)$
- **Identity laws**: $f \circ \mathrm{id}_X = f$ and $\mathrm{id}_Y \circ f = f$

1.2.2. Examples

Set: Objects are sets, morphisms are functions

Grp: Objects are groups, morphisms are group homomorphisms

Top: Objects are topological spaces, morphisms are continuous maps

Vect_k: Objects are k-vector spaces, morphisms are linear maps

Pos: Objects are partially ordered sets, morphisms are monotone functions

Cat: Objects are categories, morphisms are functors

1.3. Morphisms

1.3.1. Special Types

Monomorphism (mono): $f : X \to Y$ where $f \circ g_1 = f \circ g_2 \Rightarrow g_1 = g_2$
- Generalizes injections

Epimorphism (epi): $f : X \to Y$ where $g_1 \circ f = g_2 \circ f \Rightarrow g_1 = g_2$
- Generalizes surjections

Isomorphism: $f : X \to Y$ with inverse $g : Y \to X$ where $g \circ f = \mathrm{id}_X$ and $f \circ g = \mathrm{id}_Y$
- Objects $X \cong Y$ are isomorphic

Endomorphism: Morphism $f : X \to X$

Automorphism: Isomorphism $f : X \to X$

1.3.2. Sections and Retractions

A morphism $s : X \to Y$ is a **section** of $r : Y \to X$ if $r \circ s = \mathrm{id}_X$.

A morphism $r : Y \to X$ is a **retraction** of $s : X \to Y$ if $r \circ s = \mathrm{id}_X$.

If both exist, X is a **retract** of Y.

1.4. Functors

1.4.1. Definition

A **functor** $F : \mathcal{C} \to \mathcal{D}$ consists of:
1. Object mapping: $X \mapsto F(X)$ for objects
2. Morphism mapping: $f \mapsto F(f)$ for morphisms

Preserving:
- **Identity**: $F(\mathrm{id}_X) = \mathrm{id}_{\{F(X)\}}$
- **Composition**: $F(g \circ f) = F(g) \circ F(f)$

1.4.2. Types of Functors

Covariant functor: Preserves direction of morphisms (standard)

Contravariant functor: Reverses morphisms, $F : \mathcal{C}^{\mathrm{op}} \to \mathcal{D}$

Faithful functor: Injective on hom-sets

Full functor: Surjective on hom-sets

Fully faithful: Bijective on hom-sets (embedding)

Essentially surjective: Every object in $\mathcal{D}$ is isomorphic to $F(X)$ for some X

1.4.3. Examples

Forgetful functors: Grp $\longrightarrow$ Set forgets group structure

Free functors: Set $\longrightarrow$ Grp constructs free groups

Hom functors: $\mathrm{Hom}(A, -) : \mathcal{C} \to \mathrm{Set}$

Power set functor: $\mathcal{P} : \mathrm{Set} \to \mathrm{Set}$

Fundamental group: $\pi_1 : \mathrm{Top}_* \to \mathrm{Grp}$

1.5. Natural Transformations

1.5.1. Definition

A **natural transformation** $\alpha : F \Rightarrow G$ between functors $F, G : \mathcal{C} \to \mathcal{D}$ is:
- For each object $X \in \mathcal{C}$, a morphism $\alpha_X : F(X) \to G(X)$

Such that for every morphism $f : X \to Y$ in $\mathcal{C}$: G(f) ∘ α_X = α_Y ∘ F(f)

(The naturality square commutes)

1.5.2. Natural Isomorphism

A natural transformation is a **natural isomorphism** if each component α_X is an isomorphism.

Notation: $F \cong G$

1.5.3. Examples

Double dual: For finite-dimensional vector spaces, $V \cong V^{\star\star}$ naturally

Determinant: $\det : \mathrm{GL}_n \to \mathcal{U}(1)$ is a natural transformation

Abelianization: Natural transformation from identity to abelianization functor

1.6. Universal Properties

1.6.1. Initial and Terminal Objects

An object I is **initial** if for every object X, there exists a unique morphism $I \to X$.

An object T is **terminal** if for every object X, there exists a unique morphism $X \to T$.

Examples:
- Initial in Set: $\emptyset$
- Terminal in Set: any singleton
- Initial and terminal in Grp: trivial group

1.6.2. Limits and Colimits

A **limit** of a diagram $D : \mathcal{J} \to \mathcal{C}$ is a universal cone over D.

A **colimit** is a universal cocone under D.

Special cases:
- **Product**: Limit of discrete diagram
- **Coproduct**: Colimit of discrete diagram
- **Equalizer**: Limit of parallel pair
- **Coequalizer**: Colimit of parallel pair
- **Pullback**: Limit of cospan
- **Pushout**: Colimit of span

1.7. Products and Coproducts

1.7.1. Categorical Product

The **product** $X \times Y$ comes with projections $\pi_1 : X \times Y \to X$ and $\pi_2 : X \times Y \to Y$ such that:

For any object Z with morphisms $f : Z \to X$ and $g : Z \to Y$, there exists unique $h : Z \to X \times Y$ with $\pi_1 \circ h = f$ and $\pi_2 \circ h = g$.

1.7.2. Categorical Coproduct

The **coproduct** $X + Y$ comes with injections $i_1 : X \to X + Y$ and $i_2 : Y \to X + Y$ such that:

For any object Z with morphisms $f : X \to Z$ and $g : Y \to Z$, there exists unique $h : X + Y \to Z$ with $h \circ i_1 = f$ and $h \circ i_2 = g$.

1.7.3. Examples

In Set:
- Product is Cartesian product
- Coproduct is disjoint union

In Grp:
- Product is direct product
- Coproduct is free product

In Top:
- Product is product topology
- Coproduct is disjoint union topology

1.8. Adjunctions

1.8.1. Definition

An **adjunction** between functors $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{D} \to \mathcal{C}$ consists of a natural isomorphism: “Hom”*cal(D)(F(X), Y) ≅ “Hom”*cal(C)(X, G(Y))

We say F is **left adjoint** to G, and G is **right adjoint** to F.

Notation: $F \dashv G$

1.8.2. Unit and Counit

Equivalently, an adjunction consists of:

- **Unit**: $\eta : \mathrm{id}_{\mathtt{cal}}(C) \Rightarrow G \circ F$
- **Counit**: $\varepsilon : F \circ G \Rightarrow \mathrm{id}_{\mathtt{cal}}(D)$

Satisfying triangle identities.

1.8.3. Examples

Free-Forgetful: Free functor $F : \mathrm{Set} \to \mathrm{Grp}$ is left adjoint to forgetful $U : \mathrm{Grp} \to \mathrm{Set}$

Exponential: In cartesian closed categories, $(- \times A) \dashv (A \to -)$

Galois connections: Order-preserving maps between posets

1.9. Equivalence of Categories

1.9.1. Definition

An **equivalence** between categories $\mathcal{C}$ and $\mathcal{D}$ consists of functors:

- $F : \mathcal{C} \to \mathcal{D}$
- $G : \mathcal{D} \to \mathcal{C}$

With natural isomorphisms:

- $\eta : \mathrm{id}_{\mathtt{cal}}(C) \cong G \circ F$
- $\varepsilon : F \circ G \cong \mathrm{id}_{\mathtt{cal}}(D)$

1.9.2. Skeleton

A **skeletal category** has no distinct isomorphic objects.

Every category is equivalent to its skeleton.

1.9.3. Examples

FinSet and FinOrd: Finite sets and finite ordinals

Vect_k and Mat_k: Finite-dimensional vector spaces and matrices

CompHaus and StoneBool: Compact Hausdorff spaces and Boolean algebras (Stone duality)

1.10. Monads

1.10.1. Definition

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

- Functor $T : \mathcal{C} \to \mathcal{C}$
- Natural transformation $\eta : \mathrm{id} \to T$ (unit)
- Natural transformation $\mu : T^2 \to T$ (multiplication)

Satisfying:

- Associativity: $\mu \circ T\mu = \mu \circ \mu T$
- Unit laws: $\mu \circ T\eta = \mu \circ \eta T = \mathrm{id}$

1.10.2. Examples

List monad: In programming, handles non-determinism

Maybe monad: Handles partial functions

Power set monad: $\mathcal{P} : \mathrm{Set} \to \mathrm{Set}$

Closure operators: In topology and order theory

1.10.3. Kleisli Category

The **Kleisli category** $\mathcal{C}_T$ of monad T has:

- Objects: Same as $\mathcal{C}$
- Morphisms: $X \underset{T}{\to} Y$ are morphisms $X \to T(Y)$ in $\mathcal{C}$

1.11. Topos Theory

1.11.1. Elementary Topos

An **elementary topos** is a category with:
1. Finite limits
2. Exponentials
3. Subobject classifier Ω

1.11.2. Examples

Set: The prototypical topos

Sh(X): Sheaves on topological space X

Set^cal(C)^"op": Presheaves on $\mathcal{C}$

1.11.3. Internal Logic

Toposes have internal logic:
- Objects as types
- Morphisms as terms
- Subobjects as propositions
- Subobject classifier as truth values

1.12. Applications

1.12.1. Computer Science

Type theory: Types as objects, programs as morphisms

Functional programming: Monads for effects, functors for mapping

Database theory: Categories for schema and queries

Concurrency: Categories for processes and communication

1.12.2. Physics

Quantum mechanics: Hilbert spaces form a category

Topological quantum field theory: Functors from cobordisms to vector spaces

General relativity: Categories of manifolds and smooth maps

1.12.3. Homotopy Theory

Model categories: Framework for homotopy theory

Infinity-categories: Higher morphisms and coherence

Homotopy type theory: Univalent foundations

1.13. Categorical Constructions

1.13.1. Comma Categories

For functors $F : \mathcal{A} \to \mathcal{C}$ and $G : \mathcal{B} \to \mathcal{C}$, the **comma category** $(F \downarrow G)$ has:
- Objects: Triples $(A, B, f : F(A) \to G(B))$
- Morphisms: Commuting squares

1.13.2. Slice Categories

The **slice category** $\frac{\mathcal{C}}{X}$ has:

- Objects: Morphisms $f : Y \to X$ in $\mathcal{C}$
- Morphisms: Triangles commuting over X

1.13.3. Functor Categories

The **functor category** $[\mathcal{C}, \mathcal{D}]$ has:
- Objects: Functors $\mathcal{C} \to \mathcal{D}$
- Morphisms: Natural transformations

1.14. Implementation Notes

1.14.1. Mathlib Structure

```
class Category (C : Type u) where
  Hom : C → C → Type v
  id : ∀ X : C, Hom X X
  comp : ∀ {X Y Z : C}, Hom Y Z → Hom X Y → Hom X Z
  id_comp : ∀ {X Y : C} (f : Hom X Y), comp (id Y) f = f
  comp_id : ∀ {X Y : C} (f : Hom X Y), comp f (id X) = f
  assoc : ∀ {W X Y Z : C} (f : Hom W X) (g : Hom X Y) (h : Hom Y Z),
    comp h (comp g f) = comp (comp h g) f
```

1.14.2. Size Issues

Categories can be:
- **Small**: Objects and morphisms form sets
- **Locally small**: Hom-sets are sets
- **Large**: Objects form a proper class

Mathlib uses universe levels to handle size.

1.15. Historical Context

Category theory developed through:

- **Eilenberg & Mac Lane (1945)**: Introduced categories for algebraic topology
- **Kan (1958)**: Adjoint functors
- **Lawvere (1963)**: Categorical foundations for mathematics
- **Grothendieck (1960s)**: Topos theory
- **Mac Lane (1971)**: "Categories for the Working Mathematician"
- **Moggi (1989)**: Monads in computer science

Modern developments:
- Higher category theory
- Homotopy type theory
- Applied category theory
- Categorical quantum mechanics