Category Theory for Discrete Mathematics

A Companion to CS251

Supplementary Notes

Abstract

These notes provide a comprehensive introduction to category theory, developed in parallel with the CS251 discrete mathematics curriculum. Each section corresponds to a week of the main course and introduces categorical concepts that illuminate and generalize the core material. This companion is intended for motivated students who want to see deeper connections between discrete mathematics, programming, and abstract algebra.

Contents

1 Week 1–2: The Category of Sets

1.1 Categories: Objects and Morphisms

The central insight of category theory is that mathematical structures are best understood not in isolation, but through the *structure-preserving maps* between them. Instead of asking "what is a set?", we ask "what are the functions between sets?"

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

1. A collection $\mathrm{ob}(\mathcal{C})$ of **objects**
2. For each pair of objects A, B, a collection $\mathrm{Hom}(A, B)$ of **morphisms** (or **arrows**) from A to B
3. For each object A, an **identity morphism** $\mathrm{id}_A \in \mathrm{Hom}(A, A)$
4. For morphisms $f \in \mathrm{Hom}(A, B)$ and $g \in \mathrm{Hom}(B, C)$, a **composition** $g \circ f \in \mathrm{Hom}(A, C)$

subject to:

- **Associativity:** $(h \circ g) \circ f = h \circ (g \circ f)$
- **Identity:** $f \circ \mathrm{id}_A = f$ and $\mathrm{id}_B \circ f = f$

We write $f : A \to B$ to indicate $f \in \mathrm{Hom}(A, B)$.

Example 1.1 (The category **Set**)**.** The category **Set** has:

- Objects: all sets
- Morphisms: functions between sets
- Composition: function composition
- Identity: the identity function $\mathrm{id}_A(x) = x$

Example 1.2 (Counting morphisms)**.** In **Set**, how many morphisms are there from $A = \{1, 2\}$ to $B = \{a, b, c\}$?

Each element of A can map to any element of B, so there are $|B|^{|A|} = 3^2 = 9$ morphisms.

1.2 Commutative Diagrams

A powerful notation for expressing equations between morphisms.

Definition 1.2 (Commutative diagram)**.** A diagram of objects and morphisms **commutes** if all paths between any two objects give the same composite morphism.

Example 1.3. The triangle:

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ & \searrow^{h} & \downarrow{\scriptstyle g} \\ & & C \end{array}$$

commutes if and only if $g \circ f = h$.

Example 1.4. The square:

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

commutes if and only if $g \circ f = k \circ h$.

Programming Connection

Commutative diagrams are the "type signatures" of mathematics. In programming terms, they express that two different implementations compute the same result. The equation $g \circ f = h$ says: "composing f then g is the same as just doing h."

1.3 Monomorphisms, Epimorphisms, and Isomorphisms

The categorical approach characterizes injectivity and surjectivity *without mentioning elements.*

Definition 1.3 (Monomorphism)**.** A morphism $f : A \to B$ is a **monomorphism** (or **mono**) if it is left-cancellable:

$$\text{For all } g, h : X \to A, \quad f \circ g = f \circ h \implies g = h$$

Definition 1.4 (Epimorphism)**.** A morphism $f : A \to B$ is an **epimorphism** (or **epi**) if it is right-cancellable:

$$\text{For all } g, h : B \to X, \quad g \circ f = h \circ f \implies g = h$$

Definition 1.5 (Isomorphism)**.** A morphism $f : A \to B$ is an **isomorphism** if there exists $g : B \to A$ such that $g \circ f = \mathrm{id}_A$ and $f \circ g = \mathrm{id}_B$. We write $A \cong B$.

Theorem 1.1. *In* **Set***:*

1. *f is mono $\iff$ f is injective*
2. *f is epi $\iff$ f is surjective*
3. *f is iso $\iff$ f is bijective*

Proof. (Mono $\iff$ injective) Suppose f is mono and $f(a) = f(b)$. Define $g, h : \{*\} \to A$ by $g(*) = a$, $h(*) = b$. Then $f \circ g = f \circ h$, so $g = h$, hence $a = b$.

Conversely, suppose f is injective and $f \circ g = f \circ h$. For any x, $f(g(x)) = f(h(x))$, so $g(x) = h(x)$ by injectivity. Thus $g = h$. □

Warning

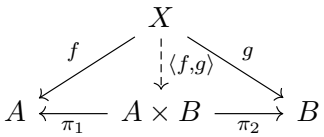
In other categories, mono $\neq$ injective and epi $\neq$ surjective! For example, in the category of rings, the inclusion $\mathbb{Z} \hookrightarrow \mathbb{Q}$ is epic but not surjective.

1.4 Universal Properties: Products and Coproducts

The deepest insights come from *universal properties*—characterizations that uniquely determine a construction up to isomorphism.

Definition 1.6 (Product)**.** A **product** of objects A and B is an object $A \times B$ together with morphisms $\pi_1 : A \times B \to A$ and $\pi_2 : A \times B \to B$ such that:

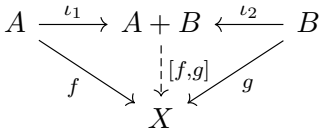
For any object X with morphisms $f : X \to A$ and $g : X \to B$, there exists a *unique* morphism $\langle f, g \rangle : X \to A \times B$ making this diagram commute:



In **Set**, the Cartesian product $A \times B = \{(a, b) : a \in A, b \in B\}$ with projections $\pi_1(a, b) = a$ and $\pi_2(a, b) = b$ satisfies this universal property.

Definition 1.7 (Coproduct)**.** A **coproduct** of objects A and B is an object $A + B$ together with morphisms $\iota_1 : A \to A + B$ and $\iota_2 : B \to A + B$ such that:

For any object X with morphisms $f : A \to X$ and $g : B \to X$, there exists a unique morphism $[f, g] : A + B \to X$ making this diagram commute:



Programming Connection

Products correspond to `pair` types or `structs`. Coproducts correspond to `Either` types or `unions`. The morphism $[f, g]$ is pattern matching:

```
case x of
  Left a  -> f(a)
  Right b -> g(b)
```

1.5 Exercises

Exercise 1.1. Let $A = \{1, 2, 3\}$ and $B = \{a, b\}$.

How many morphisms are there in $\mathrm{Hom}(A, B)$?

How many morphisms are there in $\mathrm{Hom}(B, A)$?

How many of the morphisms $A \to B$ are epimorphisms?

How many of the morphisms $B \to A$ are monomorphisms?

Exercise 1.2. Prove that every isomorphism is both a monomorphism and an epimorphism.

Exercise 1.3. Let $f : A \to B$ and $g : B \to C$. Prove:

(a) If $g \circ f$ is mono, then f is mono.

(b) If $g \circ f$ is epi, then g is epi.

Exercise 1.4 (Universal property verification)**.** Let $A = \{1, 2\}$, $B = \{a, b, c\}$, $X = \{x, y\}$. Define $f : X \to A$ by $f(x) = 1, f(y) = 2$ and $g : X \to B$ by $g(x) = a, g(y) = c$.

(a) Write out $A \times B$ explicitly.

(b) Construct the unique $\langle f, g \rangle : X \to A \times B$.

(c) Verify that $\pi_1 \circ \langle f, g \rangle = f$ and $\pi_2 \circ \langle f, g \rangle = g$.

Exercise 1.5. The **exponential** B^A in **Set** is the set of all functions from A to B, with evaluation $\varepsilon : B^A \times A \to B$ defined by $\varepsilon(f, a) = f(a)$.

The universal property says: for any $g : X \times A \to B$, there exists a unique $\tilde{g} : X \to B^A$ such that $\varepsilon \circ (\tilde{g} \times \mathrm{id}_A) = g$.

Interpret this in programming: what is $\tilde{g}$ doing? (Hint: currying.)

2 Week 3: Preorders as Categories and Functors

2.1 Preorders Are Categories

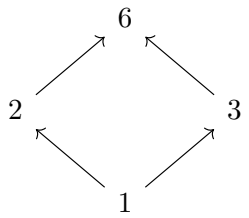
Definition 2.1 (Preorder)**.** A **preorder** $(P, \leq)$ is a set P with a relation $\leq$ that is reflexive and transitive.

Theorem 2.1. *Every preorder $(P, \leq)$ corresponds to a category:*

- *Objects: elements of P*
- *Morphisms: there is exactly one morphism $a \to b$ iff $a \leq b$*
- *Identity: $a \leq a$ (reflexivity)*
- *Composition: $a \leq b$ and $b \leq c$ imply $a \leq c$ (transitivity)*

*Such a category is called **thin** (at most one morphism between any two objects).*

Example 2.1. Consider divisibility on $\{1, 2, 3, 6\}$. As a category:



The arrow $1 \to 6$ exists (via composition $1 \to 2 \to 6$) but is not drawn separately.

Example 2.2. The power set $(\mathcal{P}(X), \subseteq)$ is a preorder category. Products are intersections (greatest lower bounds) and coproducts are unions (least upper bounds).

2.2 Functors

Definition 2.2 (Functor)**.** A **functor** $F : \mathcal{C} \to \mathcal{D}$ between categories consists of:

1. A mapping $F : \mathrm{ob}(\mathcal{C}) \to \mathrm{ob}(\mathcal{D})$ on objects
2. For each $f : A \to B$ in $\mathcal{C}$, a morphism $F(f) : F(A) \to F(B)$ in $\mathcal{D}$

such that:

- $F(\mathrm{id}_A) = \mathrm{id}_{F(A)}$ (preserves identities)
- $F(g \circ f) = F(g) \circ F(f)$ (preserves composition)

Theorem 2.2. *A functor between preorder categories is exactly a* ***monotone function****:* $a \leq b \implies F(a) \leq F(b)$.

Example 2.3 (Power set functor)**.** Define $\mathcal{P} : \mathbf{Set} \to \mathbf{Set}$ by:

- On objects: $\mathcal{P}(A) = \{S : S \subseteq A\}$
- On morphisms: $\mathcal{P}(f)(S) = \{f(x) : x \in S\}$ (direct image)

Verify: $\mathcal{P}(\mathrm{id}_A)(S) = S$ and $\mathcal{P}(g \circ f) = \mathcal{P}(g) \circ \mathcal{P}(f)$.

Example 2.4 (Forgetful functor)**.** Many "forgetful" functors forget structure:

- $U : \mathbf{Grp} \to \mathbf{Set}$ forgets the group operation
- $U : \mathbf{Top} \to \mathbf{Set}$ forgets the topology

2.3 The Category Rel

Definition 2.3 (The category **Rel**)**.** The category **Rel** has:

- Objects: sets
- Morphisms $R : A \to B$: relations $R \subseteq A \times B$
- Composition: $S \circ R = \{(a, c) : \exists b.\, (a, b) \in R \wedge (b, c) \in S\}$
- Identity: $\Delta_A = \{(a, a) : a \in A\}$

There is a functor $\mathbf{Set} \to \mathbf{Rel}$ sending each function to its graph (a functional relation).

Example 2.5. Let $R = \{(1, a), (1, b), (2, c)\} \subseteq \{1, 2\} \times \{a, b, c\}$ and $S = \{(a, x), (b, x), (c, y)\} \subseteq \{a, b, c\} \times \{x, y\}$.

Then $S \circ R = \{(1, x), (2, y)\}$.

2.4 Galois Connections (Adjunctions for Preorders)

Definition 2.4 (Galois connection)**.** Let $(P, \leq)$ and $(Q, \leq)$ be preorders. A **Galois connection** is a pair of monotone functions $F : P \to Q$ and $G : Q \to P$ such that:

$$F(p) \leq q \iff p \leq G(q)$$

We write $F \dashv G$ ("F is left adjoint to G").

Example 2.6 (Floor and ceiling)**.** Between $(\mathbb{R}, \leq)$ and $(\mathbb{Z}, \leq)$:

- $\iota \dashv \lfloor \cdot \rfloor$: the inclusion $\iota : \mathbb{Z} \to \mathbb{R}$ is left adjoint to floor
- $\lceil \cdot \rceil \dashv \iota$: ceiling is left adjoint to inclusion

This means: $\iota(n) \leq x \iff n \leq \lfloor x \rfloor$ and $x \leq \iota(n) \iff \lceil x \rceil \leq n$.

Key Result

Galois connections are the "preorder version" of adjunctions, one of the most important concepts in category theory. They appear throughout computer science: in abstract interpretation, type inference, and database query optimization.

2.5 Exercises

Exercise 2.1. Show that the divisibility preorder on $\{1, 2, 4, 8\}$ forms a total order when viewed as a category.

Exercise 2.2. Let $P = (\{1, 2, 3, 4\}, \leq)$ and $Q = (\{a, b, c\}, \leq)$ where $a \leq b \leq c$.

(a) How many functors (monotone functions) are there from P to Q?

(b) Give an example of a non-monotone function $\{1, 2, 3, 4\} \to \{a, b, c\}$.

Exercise 2.3. Verify that relational composition in **Rel** is associative.

Exercise 2.4 (Application: Type systems)**.** In a programming language with subtyping, we have a preorder on types. The function type constructor is:

- Contravariant in the argument: if $A' \leq A$, then $(A \to B) \leq (A' \to B)$
- Covariant in the result: if $B \leq B'$, then $(A \to B) \leq (A \to B')$

Explain why the argument is contravariant using the substitution principle.

3 Weeks 4–5: Polynomial Functors and Counting

3.1 Types Have Sizes

A surprising connection between algebra and type theory:

Type	**Notation**	**Inhabitants**
Void (empty)	0	0
Unit	1	1
Bool	2	2
Sum $A + B$	$A + B$	$\lvert A\rvert + \lvert B\rvert$
Product $A \times B$	$A \times B$	$\lvert A\rvert \times \lvert B\rvert$
Function $A \to B$	B^A	$\lvert B\rvert^{\lvert A\rvert}$

This is not coincidence—it reflects deep structure.

3.2 Polynomial Functors

Definition 3.1 (Polynomial functor)**.** A **polynomial functor** $F : \mathbf{Set} \to \mathbf{Set}$ has the form:

$$F(X) = A_0 + A_1 \times X + A_2 \times X^2 + A_3 \times X^3 + \cdots$$

where the A_i are fixed sets and $X^n = X \times X \times \cdots \times X$ (n times).

The **generating function** of F is $\sum_i |A_i| x^i$.

Example 3.1 (Maybe/Option)**.** $\text{Maybe}(X) = 1 + X$ with generating function $1 + x$.

- $\text{Maybe}(\emptyset) = 1 + \emptyset = 1 = \{\text{Nothing}\}$
- $\text{Maybe}(\{a\}) = 1 + \{a\} = \{\text{Nothing}, \text{Just } a\}$

Example 3.2 (Lists (up to length 2))**.** $\text{List}_2(X) = 1 + X + X^2$ with generating function $1 + x + x^2$.
For $X = \{a, b\}$: $|\text{List}_2(\{a, b\})| = 1 + 2 + 4 = 7$.
These are: $[]$, $[a]$, $[b]$, $[a, a]$, $[a, b]$, $[b, a]$, $[b, b]$.

Example 3.3 (Full lists)**.** $\text{List}(X) = 1 + X + X^2 + X^3 + \cdots = \frac{1}{1-X}$ (formal power series).
The recursive definition $\text{List}(X) = 1 + X \times \text{List}(X)$ encodes this.

3.3 Functors on Morphisms

For a polynomial functor to be a true functor, we need its action on morphisms.

Example 3.4 (Maybe as a functor)**.** Given $f : A \to B$, define $\text{Maybe}(f) : \text{Maybe}(A) \to \text{Maybe}(B)$ by:

$$\text{Maybe}(f)(\text{Nothing}) = \text{Nothing} \qquad \text{Maybe}(f)(\text{Just } a) = \text{Just } f(a)$$

This is `fmap` for the Maybe functor!

Programming Connection

The functor laws $F(\text{id}) = \text{id}$ and $F(g \circ f) = F(g) \circ F(f)$ are exactly the laws that `fmap` must satisfy in Haskell:

```
fmap id = id
fmap (g . f) = fmap g . fmap f
```

3.4 Combinatorial Species (Advanced)

Definition 3.2 (Species)**.** A **species** is a functor $F : \mathcal{B} \to \mathbf{Set}$ where $\mathcal{B}$ is the category of finite sets and bijections.
$F(A)$ = "F-structures on the set A"
$F(\sigma)$ = "relabeling by bijection σ"

Example 3.5 (Species of linear orders)**.** $L(A) = \{\text{linear orderings of } A\}$. We have $|L(\{1, 2, 3\})| = 3! = 6$.

Example 3.6 (Species of graphs)**.** $G(A) = \{\text{simple graphs with vertex set } A\}$. We have $|G(\{1, 2, 3\})| = 2^3 = 8$.

The theory of species provides a categorified approach to generating functions.

3.5 Exercises

Exercise 3.1. What polynomial functor represents "a pair where the first component is from a 3-element set"? What is its generating function?

Exercise 3.2. Express the type `Either (a, b) c` as a polynomial in a, b, c. How many inhabitants does `Either (Bool, Bool) ()` have?

Exercise 3.3. Verify the functor laws for Maybe: Maybe(id) = id and Maybe$(g \circ f)$ = Maybe$(g) \circ$ Maybe(f).

Exercise 3.4. The type of binary trees with data at leaves is Tree$(A) = A + \text{Tree}(A)^2$. Expand the first few terms of this as a power series in A. What is the coefficient of A^3? (It counts binary tree shapes with 3 leaves.)

4 Weeks 6–7: Graphs and Free Categories

4.1 Graphs Generate Categories

Definition 4.1 (Quiver/Directed graph)**.** A **quiver** (or directed graph) Q consists of:

- A set Q_0 of **vertices**
- A set Q_1 of **edges**
- Source and target functions $s, t : Q_1 \to Q_0$

Definition 4.2 (Free category on a graph)**.** The **free category** Path(Q) on a quiver Q has:

- Objects: vertices of Q
- Morphisms: directed paths in Q (including empty paths)
- Composition: path concatenation
- Identity: empty path at each vertex

Example 4.1. For the graph $1 \xrightarrow{a} 2 \xrightarrow{b} 3$:
Path(Q) has morphisms: $\text{id}_1, \text{id}_2, \text{id}_3, a, b, b \circ a$.

Example 4.2. For a cycle $1 \xrightarrow{a} 2 \xrightarrow{b} 3 \xrightarrow{c} 1$:
Path(Q) is infinite! From 1 to 1 we have: $\text{id}_1, cba, (cba)^2, (cba)^3, \ldots$

4.2 Diagrams as Functors

Definition 4.3. A **diagram of shape J in category $\mathcal{C}$** is a functor $D : J \to \mathcal{C}$.

The shape J is often a small category like:

- $\bullet \to \bullet$: picks out a morphism
- $\bullet \rightrightarrows \bullet$: picks out a parallel pair
- A commutative square: picks out a commuting square in $\mathcal{C}$

Application (Database schemas). A database schema is a quiver where vertices are tables and edges are foreign key relationships. A database instance is a functor from Path(Schema) to **Set**:

- Each table T maps to a set $F(T)$ of rows
- Each foreign key $e : T \to T'$ maps to a function $F(e) : F(T) \to F(T')$

4.3 Adjacency Matrices Count Paths

Theorem 4.1. *For the free category Path(Q), if A is the adjacency matrix of Q, then:*

$$(A^n)_{ij} = \textit{number of paths of length } n \textit{ from } i \textit{ to } j$$

This explains the graph-theoretic interpretation of matrix powers!

4.4 Exercises

Exercise 4.1. For the graph $a \xrightarrow{f} b \xrightarrow{g} c$, $c \xrightarrow{h} b$:

(a) List all morphisms from a to c in Path(Q).

(b) List all morphisms from b to b.

(c) Is Path(Q) finite or infinite?

Exercise 4.2. Draw a database schema for: Users, Posts, Comments, where Posts have authors (Users) and Comments reference both a Post and a User. Write down what a functor from this schema to **Set** looks like.

5 Week 8: Initial Algebras and Catamorphisms

5.1 F-Algebras

Definition 5.1 (F-algebra)**.** Let $F : \mathcal{C} \to \mathcal{C}$ be an endofunctor. An **F-algebra** is a pair (A, α) where:

- A is an object (the **carrier**)
- $\alpha : F(A) \to A$ is a morphism (the **structure map**)

Definition 5.2 (Algebra homomorphism)**.** A **homomorphism** $h : (A, \alpha) \to (B, \beta)$ of F-algebras is a morphism $h : A \to B$ such that:

$$\begin{array}{ccc} F(A) & \xrightarrow{\alpha} & A \\ {\scriptstyle F(h)}\downarrow & & \downarrow{\scriptstyle h} \\ F(B) & \xrightarrow[\beta]{} & B \end{array}$$

commutes, i.e., $h \circ \alpha = \beta \circ F(h)$.

Example 5.1 (Algebras for $F(X) = 1 + X$)**.** An F-algebra (A, α) consists of:

- A set A
- A function $\alpha : 1 + A \to A$, i.e., we specify:
  - $z = \alpha(*) \in A$ (the "zero")
  - $s(a) = \alpha(a)$ for each $a \in A$ (the "successor")

So an F-algebra is (A, z, s) with $z \in A$ and $s : A \to A$.

5.2 Initial Algebras

Definition 5.3 (Initial algebra)**.** An **initial F-algebra** $(\mu F, \text{in})$ is an F-algebra such that for every F-algebra (A, α), there exists a *unique* homomorphism $\alpha : \mu F \to A$.

This unique morphism is called a **catamorphism** (or **fold**).

Key Result

[Lambek's Lemma] If $(\mu F, \text{in})$ is an initial F-algebra, then $\text{in} : F(\mu F) \to \mu F$ is an **isomorphism**.
Therefore: $\mu F \cong F(\mu F)$.

This is exactly what recursive type definitions mean!

Example 5.2 (Natural numbers)**.** For $F(X) = 1 + X$, the initial algebra is $(\mathbb{N}, [\text{zero}, \text{succ}])$ where:

- $\text{zero} : 1 \to \mathbb{N}$ picks out 0
- $\text{succ} : \mathbb{N} \to \mathbb{N}$ is the successor function

For any (A, z, s), the unique catamorphism $z, s : \mathbb{N} \to A$ is:

$$z, s(0) = z \qquad z, s(n+1) = s(z, s(n))$$

This is primitive recursion!

Example 5.3 (Lists)**.** For $F_A(X) = 1 + A \times X$, an algebra gives "nil" and "cons."
The initial algebra is $(List(A), [\text{nil}, \text{cons}])$.
The catamorphism is `foldr`:

```
foldr f z []     = z
foldr f z (x:xs) = f x (foldr f z xs)
```

Example 5.4 (Binary trees)**.** For $F_A(X) = A + X \times X$, the initial algebra is binary trees with data at leaves.

Examples of catamorphisms:

- Sum: $\text{leaf}(a) = a$, $\text{node}(x, y) = x + y$
- Height: $\text{leaf}(a) = 0$, $\text{node}(x, y) = 1 + \max(x, y)$
- Flatten: $\text{leaf}(a) = [a]$, $\text{node}(x, y) = x +\!\!+ y$

5.3 Why Structural Recursion Terminates

The initiality property guarantees:

1. **Existence:** A recursive definition gives a function
2. **Uniqueness:** The function is well-defined
3. **Correctness:** It satisfies the recursive equations

If recursion didn't terminate, no function would exist (violating existence).

5.4 Catamorphism Fusion

Theorem 5.1 (Fusion)**.** *If $h : (A, \alpha) \to (B, \beta)$ is an algebra homomorphism, then:*

$$h \circ \alpha = \beta$$

Programming Connection

This is the basis for **deforestation** and **fusion** optimizations:

```
map f . map g = map (f . g)   -- fuse two traversals into one
sum . map f   = foldr (\x a -> f x + a) 0
```

5.5 Exercises

Exercise 5.1. For $F(X) = 1 + X$:

(a) Define an F-algebra on $\{\text{even}, \text{odd}\}$ with $z = \text{even}$ and $s = \text{flip}$.

(b) Compute the unique catamorphism from $\mathbb{N}$ to this algebra.

(c) What function does this catamorphism compute?

Exercise 5.2. For $F_{\mathbb{N}}(X) = 1 + \mathbb{N} \times X$:

(a) Define an algebra on $\mathbb{N}$ that computes the sum of a list.

(b) Define an algebra on $\mathbb{N}$ that computes the length of a list.

(c) Define an algebra on $\mathbb{N}$ that computes the product of a list.

Exercise 5.3. Verify Lambek's Lemma for natural numbers: show explicitly that $[\text{zero}, \text{succ}] : 1 + \mathbb{N} \to \mathbb{N}$ is an isomorphism by constructing its inverse.

Exercise 5.4. Express each tree traversal (preorder, inorder, postorder) as a catamorphism by specifying the algebra.

6 Week 9: Coalgebras and Automata

6.1 F-Coalgebras

Coalgebras are **dual** to algebras—arrows reversed.

Definition 6.1 (F-coalgebra)**.** Let $F : \mathcal{C} \to \mathcal{C}$ be an endofunctor. An **F-coalgebra** is a pair (A, α) where:

- A is an object (the **state space**)
- $\alpha : A \to F(A)$ is a morphism (the **observation/transition map**)

Definition 6.2 (Coalgebra homomorphism)**.** A homomorphism $h : (A, \alpha) \to (B, \beta)$ is a morphism $h : A \to B$ such that:

$$\begin{array}{ccc} A & \xrightarrow{\alpha} & F(A) \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle F(h)} \\ B & \xrightarrow[\beta]{} & F(B) \end{array}$$

commutes, i.e., $\beta \circ h = F(h) \circ \alpha$.

6.2 Deterministic Finite Automata as Coalgebras

Theorem 6.1. *A DFA over alphabet Σ is exactly a coalgebra for the functor:*

$$F(X) = 2 \times X^{\Sigma}$$

where $2 = \{accept, reject\}$ and X^{Σ} denotes functions $\Sigma \to X$.

A coalgebra $(Q, \langle o, \delta \rangle)$ consists of:

- Q: set of states
- $o : Q \to 2$: output function (is this state accepting?)
- $\delta : Q \to Q^{\Sigma}$: transition function

6.3 Final Coalgebras

Definition 6.3 (Final coalgebra)**.** A **final F-coalgebra** $(\nu F, \text{out})$ is an F-coalgebra such that for every F-coalgebra (A, α), there exists a unique homomorphism $\alpha : A \to \nu F$.

This unique morphism is called an **anamorphism** (or **unfold**).

Theorem 6.2. *For $F(X) = 2 \times X^{\Sigma}$, the final coalgebra is:*

$$(\mathcal{P}(\Sigma^*), \langle \varepsilon?, \textit{derivatives} \rangle)$$

where:

- $\mathcal{P}(\Sigma^*)$ *is the set of all languages over* Σ
- $\varepsilon?(L) =$ *accept iff* $\varepsilon \in L$
- $\partial_a(L) = \{w : aw \in L\}$ *(Brzozowski derivative)*

The unique coalgebra morphism from any DFA to this final coalgebra maps each state to **the language it accepts**!

6.4 Bisimulation

Definition 6.4 (Bisimulation)**.** Two states (possibly in different automata) are **bisimilar** if they map to the same element of the final coalgebra.

For DFAs: bisimilar states accept the same language.

6.5 Streams as a Final Coalgebra

Theorem 6.3. *For $F(X) = A \times X$, the final coalgebra is:*

$$(A^{\omega}, \langle \textit{head}, \textit{tail} \rangle)$$

where A^{ω} is the set of infinite streams over A.

The isomorphism $A^{\omega} \cong A \times A^{\omega}$ says: "a stream is a head plus a tail."

Example 6.1 (Generating streams)**.** The Fibonacci sequence as an anamorphism:

```
fibs = unfold (\(a,b) -> (a, (b, a+b))) (0, 1)
     = [0, 1, 1, 2, 3, 5, 8, 13, ...]
```

6.6 Algebra vs Coalgebra: Summary

Concept	Algebra	Coalgebra
Structure map	$F(A) \to A$	$A \to F(A)$
Universal object	Initial	Final
Universal morphism	Catamorphism (fold)	Anamorphism (unfold)
Canonical example	Natural numbers, lists, trees	Streams, automata
Data	Finite (inductive)	Potentially infinite (coinductive)
Perspective	Constructors	Observers/destructors

6.7 Exercises

Exercise 6.1. Write the DFA accepting "strings ending in 01" as an F-coalgebra for $F(X) = 2 \times X^{\{0,1\}}$.

Exercise 6.2. Compute the Brzozowski derivatives $\partial_0(L)$ and $\partial_1(L)$ for $L = \{w : w \text{ has even number of 1s}\}$.

Exercise 6.3. Define the stream of powers of 2 as an anamorphism.

Exercise 6.4. Two DFAs have state sets $Q_1 = \{a, b\}$ and $Q_2 = \{x, y, z\}$. State a accepts $\{0^n 1^n : n \geq 0\}$ (note: not regular, but pretend). State x accepts all strings. Are a and x bisimilar? Why or why not?

7 Week 10: Monoids and Cost

7.1 Monoids as One-Object Categories

Definition 7.1 (Monoid)**.** A **monoid** $(M, \cdot, e)$ is:

- A set M
- An associative binary operation $\cdot : M \times M \to M$
- An identity element $e \in M$ with $e \cdot m = m \cdot e = m$

Theorem 7.1. *A monoid is exactly a category with one object.*

- *The single object: call it* $*$
- *Morphisms* $* \to *$*: elements of* M
- *Composition: the monoid operation*
- *Identity:* e

Example 7.1 (Common monoids)**.**

Monoid	Set	Operation	Identity
$(\mathbb{N}, +, 0)$	natural numbers	addition	0
$(\mathbb{N}, \times, 1)$	natural numbers	multiplication	1
$(\Sigma^*, \cdot, \varepsilon)$	strings	concatenation	empty string
$(\mathbb{N} \cup \{\infty\}, \max, 0)$	extended naturals	maximum	0
$(\mathbb{N} \cup \{\infty\}, \min, \infty)$	extended naturals	minimum	∞

7.2 Free Monoids

Definition 7.2 (Free monoid)**.** The **free monoid** on a set Σ is $(\Sigma^*, \cdot, \varepsilon)$—strings with concatenation.

Theorem 7.2 (Universal property)**.** *For any monoid M and function $f : \Sigma \to M$, there exists a unique monoid homomorphism $f^* : \Sigma^* \to M$ extending f.*

7.3 Regular Languages and Syntactic Monoids

Theorem 7.3. *A language $L \subseteq \Sigma^*$ is regular if and only if there exists:*

1. *A finite monoid M*
2. *A monoid homomorphism $h : \Sigma^* \to M$*
3. *A subset $F \subseteq M$*

such that $L = h^{-1}(F)$.

The minimal such M is the **syntactic monoid** of L.

Example 7.2. For $L = \{w : |w|_1 \equiv 0 \pmod 3\}$ (number of 1s divisible by 3):
The syntactic monoid is $\mathbb{Z}_3$ with $h(0) = 0$, $h(1) = 1$.

7.4 Cost Monoids

Definition 7.3 (Cost monoid)**.** A **cost monoid** is a monoid used to track computational resources:

- $(\mathbb{N}, +, 0)$: counting operations (sequential)
- $(\mathbb{N}, \max, 0)$: parallel time (max of parallel branches)
- $(\mathbb{N} \times \mathbb{N}, +, (0, 0))$: tracking time AND space

Example 7.3 (Merge sort cost)**.** $T(n) = 2T(n/2) + \Theta(n)$
Using $(\mathbb{N}, +, 0)$: compose costs of subproblems and merge step.
Using $(\mathbb{N}, \max, 0)$ for parallel: $T(n) = T(n/2) + \Theta(\log n) = \Theta(\log^2 n)$.

7.5 Enriched Categories (Brief)

A **category enriched over a monoidal category** V replaces hom-sets with objects of V.

Example 7.4 (Weighted graphs)**.** A weighted graph with weights in $(\mathbb{R}_{\geq 0} \cup \{\infty\}, \min, \infty)$ is an enriched category:

- Objects: vertices
- $\mathrm{Hom}(u, v)$: edge weight (∞ if no edge)
- Composition: $d(u, w) = \min_v(d(u, v) + d(v, w))$

This is exactly the structure that shortest-path algorithms exploit!

7.6 Exercises

Exercise 7.1. Verify that $(\mathbb{N}, \max, 0)$ is a monoid.

Exercise 7.2. How many monoid structures are there on the set $\{0, 1\}$? List them all.

Exercise 7.3. Show that the length function $|\cdot| : \Sigma^* \to \mathbb{N}$ is a monoid homomorphism.

Exercise 7.4. Find the syntactic monoid of $L = \{w \in \{a, b\}^* : w \text{ contains } ab\}$.

Exercise 7.5. Model the cost of linear search and binary search using the cost monoid $(\mathbb{N}, +, 0)$.

8 Conclusion: The Categorical Perspective

8.1 Recurring Themes

Throughout this companion, we've seen several recurring ideas:

1. Universal Properties. Many constructions are characterized by being "the best" solution to a problem: products, coproducts, free constructions, initial and final objects.

2. Functors Preserve Structure. Structure-preserving maps (functors, homomorphisms) are often more important than the structures themselves.

3. Duality. Many concepts come in pairs: products/coproducts, monos/epis, algebras/coalgebras, initial/final. Understanding one often gives the other "for free."

4. Diagrams as Specifications. Commutative diagrams express equations. When we say "this diagram commutes," we're stating an invariant that implementations must satisfy.

8.2 Where to Go From Here

- **Seven Sketches in Compositionality** by Fong & Spivak: applied category theory, free online
- **Category Theory for Programmers** by Milewski: Haskell-focused, free online
- **Algebra of Programming** by Bird & de Moor: folds, unfolds, and program calculation
- **Categories for the Working Mathematician** by Mac Lane: the classic reference