Дискретная математика

Теория категорий

Константин Чухарев

Осень 2026

Теория категорий

> *Я придумал категории не для того, чтобы изучать функторы. Я придумал их, чтобы изучать естественные преобразования.*
>
> ***— Сондерс Мак Лейн***

Зачем нужен ещё один язык?

Три конструкции из разных глав курса несут одно устройство:

1. **НОД**: каждый общий делитель чисел 12 и 18 делит 6
2. **произведение**: пару функций $f : X \to A$ и $g : X \to B$ можно собрать в одну $x \mapsto (f(x), g(x))$ в $A \times B$, причём только одну
3. **конъюнкция**: любое утверждение, из которого следуют и p, и q, следует из $p \wedge q$

Свойство везде про *положение среди других объектов*, и формулируется оно без вычислений.

> Материал трёх примеров разный, устройство одно и то же. Теория категорий — язык, который делает это совпадение видимым до вычислений.

1945 год и слово «естественный»

Топологи 1930-х называли одни изоморфизмы *естественными*, а другие — *зависящими от выбора базиса*.

Точного определения за словом не стояло.

Пример (Дважды двойственное пространство)

Вложение пространства V в его дважды двойственное V^{**} строится одним правилом, без базиса. Изоморфизм пространств $V \to V^*$ требует выбрать базис и меняется при его смене.

Сэмюэл Эйленберг и Сондерс Мак Лейн формализовали «естественность» в 1945 году.

Для этого понадобились сразу три понятия: категория, функтор, естественное преобразование.

Три знакомых мира

В каждом из трёх миров предыдущих модулей есть объекты, стрелки между ними и способ складывать стрелки. Таблица показывает, что именно играет эти роли.

Мир	Стрелки	Композиция	Тождество
множества	функции $f : A \to B$	$g \circ f$	$\mathrm{id}_A(x) = x$
моноид	элементы $m : M \to M$	умножение $a \cdot b$	единица e
порядок	$a \to b$ при $a \leq b$	транзитивность	рефлексивность

Определение категории фиксирует общий узор всех трёх строк.

Категория

Определение 1 (Категория)

Категория $\mathcal{C}$ — это алгебра из трёх частей.

1. Совокупность **объектов** $A, B, C, \ldots$
2. Совокупность **стрелок** $f : A \to B$ с началом A и концом B
3. Операция **композиции**, сопоставляющая паре $f : A \to B$ и $g : B \to C$ стрелку $g \circ f : A \to C$

Композиция ассоциативна, а у каждого объекта есть тождественная стрелка id_A с $\mathrm{id}_B \circ f = f = f \circ \mathrm{id}_A$.

Композиция читается справа налево, как $g(f(x))$, а тождественная стрелка единственна: для кандидатов e, e' выполнено $e = e \circ e' = e'$ — как с нейтральным элементом моноида.

Как объект устроен внутри, язык не описывает: об объекте он говорит только через стрелки, входящие в него и выходящие из него.

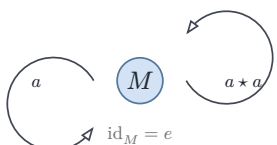
Множества и моноиды как категории

Пример (Категория Set**)**

Объекты — множества, стрелки — функции, композиция — обычная. Ассоциативность — теорема о композиции функций.

Пример (Моноид — категория с одним объектом)

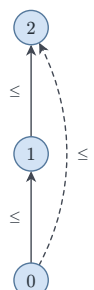
Элементы моноида получают **вторую роль** — роль стрелок $M \to M$, операция — композиция. Таблица Кэли становится таблицей композиции стрелок.



Порядок как тонкая категория

Пример (Порядок — категория без параллельных стрелок)
Объекты — элементы, стрелка из a в b есть ровно при $a \leq b$. Композиция — транзитивность, тождество — рефлексивность. Между любыми двумя объектами не более одной стрелки.

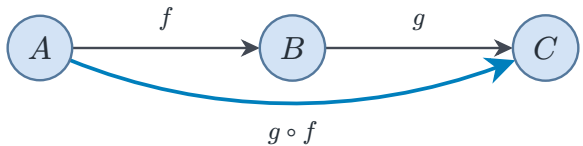
Пунктирная стрелка $0 \to 2$ — та же единственная, равная композиции сплошных; тождественные петли на рисунках не изображаются.



Свободная категория графа

Пример (Пути как морфизмы)

Стрелки — пути в графе, композиция — склейка путей. Из A в C появляется новый морфизм $g \circ f$.



Замкнутость против свободы

В $(\mathbb{Z}_4, +, 0)$ стрелок ровно четыре: композиция — сложение по модулю 4, результат — снова элемент моноида.

Композиция не рождает новых стрелок.

> **Пример (Пути — наоборот)**
> В свободной категории графа пути растут: обходя цикл снова и снова, получаем всё новые стрелки. Композиция — склейка путей, и верхнего предела у неё нет.

Разница — в таблице: у моноида она *замкнута*, у путей — конкатенация без границ.

Двойственность

Категория $\mathcal{C}^{\mathrm{op}}$ — те же объекты и стрелки, начала и концы обращены.

Доказанное для всех категорий верно и для $\mathcal{C}^{\mathrm{op}}$ — значит, верно и двойственное утверждение.

Двойник своими руками

У каждого понятия есть двойник:

1. мономорфизм $\leftrightarrow$ эпиморфизм
2. терминальный объект $\leftrightarrow$ начальный
3. предел $\leftrightarrow$ копредел

Каждая доказанная теорема автоматически даёт двойственную.

Стрелки вместо элементов

Определение 2 (Изоморфизм)

Стрелка $f : A \to B$ — **изоморфизм**, если есть обратная $g : B \to A$ с $g \circ f = \mathrm{id}_A$ и $f \circ g = \mathrm{id}_B$.

Определение 3 (Мономорфизм и эпиморфизм)

Стрелка f — **мономорфизм**, если из $f \circ g_1 = f \circ g_2$ следует $g_1 = g_2$. Стрелка f — **эпиморфизм**, если из $g_1 \circ f = g_2 \circ f$ следует $g_1 = g_2$.

Пример (Знакомые соответствия)

В категории Set это в точности биекции, инъекции и сюръекции. В категории моноида изоморфизмы — **обратимые элементы**: $g \circ f = \mathrm{id}$ читается как $g \star f = e$. В тонкой категории каждая стрелка — и моно, и эпи, но $0 \to 1$ не изоморфизм: обратной стрелки нет.

Функтор

Определение 4 (Функтор)

Функтор $F : \mathcal{C} \to \mathcal{D}$ переводит объекты в объекты и стрелки в стрелки так, что

$$F(g \circ f) = F(g) \circ F(f), \quad F(\mathrm{id}_A) = \mathrm{id}_{F(A)}.$$

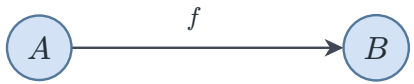
Это отображение структур, сохраняющее композицию.

Коммутирующие диаграммы переходят в коммутирующие.

В каждом из трёх знакомых миров сохранение структуры — функтор:

1. гомоморфизм моноида — функтор между однообъектными категориями
2. монотонное отображение — функтор между тонкими категориями порядка
3. отображение графа, сохраняющее рёбра, — функтор между свободными категориями путей

Булеан в обе стороны



Пример (Образ и прообраз)

Функция $f : A \to B$ поднимается на подмножества дважды: образ идёт вперёд (ковариантность), прообраз — назад (контравариантность: стрелки переворачиваются). Для $f(1) = a$, $f(2) = b$: образ $\{1\} = \{a\}$, прообраз $\{a, c\} = \{1\}$, прообраз $\{c\} = \varnothing$.

Категория типов и код

В программистской категории объекты — типы, стрелки — функции. Функтор в ней — контейнер с `map`.

```
// Функтор "список": map уважает композицию и тождество.
fn map<T, U>(xs: &[T], f: impl Fn(&T) -> U) -> Vec<U> {
    xs.iter().map(f).collect()
}
// map(xs, |x| g(f(x))) == map(map(xs, f), g)
// map(xs, |x| x)             == xs
```

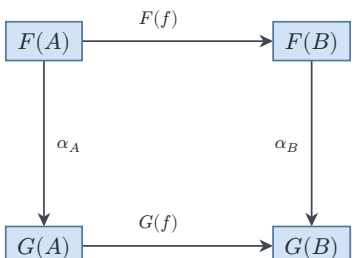
Естественное преобразование

Определение 5 (Естественное преобразование)

Семейство стрелок $\alpha_A : F(A) \to G(A)$ называется **естественным преобразованием** $F \Rightarrow G$, если для каждой стрелки $f : A \to B$ коммутирует квадрат

$$G(f) \circ \alpha_A = \alpha_B \circ F(f).$$

Семейство согласовано со всеми стрелками сразу — «одно правило, без выбора».



Полиморфизм — это естественность

Пример (Безопасный первый элемент)

Семейство `head` переводит список в `Option`: пустой — в `None`, непустой — в первый элемент. Естественность — это равенство

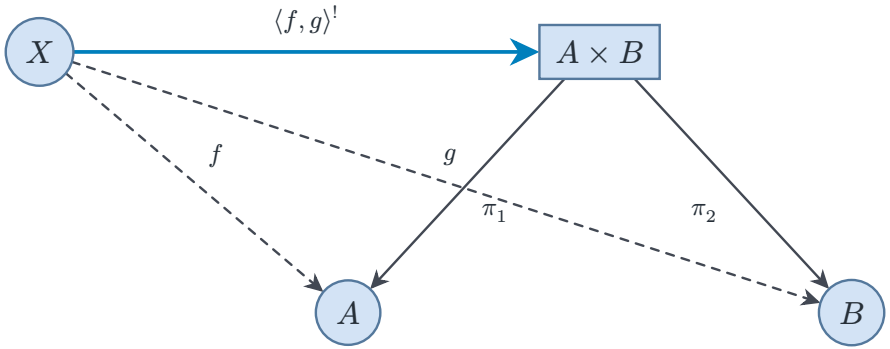
$$\mathrm{map}\, g\, (\mathrm{head}\, \mathrm{xs}) \quad = \quad \mathrm{head}\, (\mathrm{map}\, g\, \mathrm{xs})$$

для любой функции g. Достать первый и преобразовать — то же, что преобразовать все и достать первый. Левая часть — путь через Option, правая — через список: это два пути одного квадрата.

Параметричность — имя этого явления.

Полиморфная функция не может подсматривать в элементы, и квадраты естественности выдаются *бесплатно*.

Универсальные свойства



Определение 6 (Произведение)

Объект $A \times B$ со стрелками π_1, π_2 называется **произведением**, если любая пара $f : X \to A, g : X \to B$ пропускается через него ровно одной стрелкой $\langle f, g \rangle$.

Свойство относится не к внутренностям объекта, а к его положению среди стрелок.

Одна конструкция — пять категорий

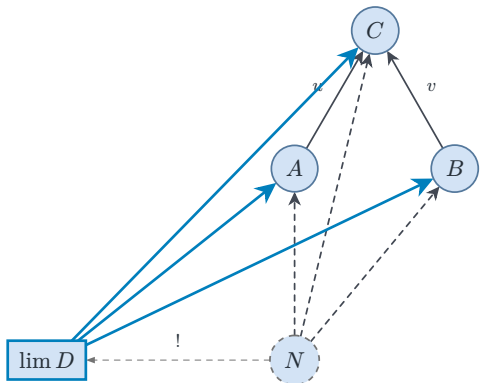
Категория	Произведение	Копроизведение
множества	пары $A \times B$	дизъюнктное объединение
логика	$p \wedge q$	$p \vee q$
порядок	инфимум $a \wedge b$	супремум $a \vee b$
делимость	НОД(a, b)	НОК(a, b)
автоматы	прямое произведение	дизъюнктное объединение

Универсальное свойство доказывается один раз — и работает во всех категориях сразу. Единственность с точностью до изоморфизма прилагается бесплатно.

Конус над диаграммой

Определение 7 (Конус)

Конус над диаграммой — объект с семейством стрелок-ножек в каждый объект диаграммы, согласованным с её стрелками.



Предел

Определение 8 (Предел)

Предел — это терминальный конус: через него пропускается любой другой конус, причём ровно одной стрелкой m с $n_A = l_A \circ m$ для каждой ножки.

Произведение — предел пары объектов без стрелок.

Пределы вокруг нас

Пределы и копределы — рабочие конструкции всей математики.

1. пересечение подмножеств — предел диаграммы вложений
2. дизъюнктная сумма и объединение — копределы
3. НОК целых чисел — копредел диаграммы делимости
4. пополнения и p-адические числа — пределы обратных систем

Лемма Йонеды

Теорема 1 (Лемма Йонеды)

Естественные преобразования $\mathcal{C}(A, -) \to F$ находятся во взаимно однозначном соответствии с элементами $F(A)$:

$$\mathrm{Nat}(\mathcal{C}(A, -),\ F) \quad \cong \quad F(A).$$

Соответствие строится явно: преобразование переходит в своё значение на тождественной стрелке id_A, а элемент $x \in F(A)$ — в семейство $\alpha_B(f) = F(f)(x)$.

Объект **полностью определяется стрелками** в него и из него. Всё, что можно узнать об объекте, — это его связи с другими объектами.

Сопряжённые функторы

Определение 9 (Сопряжённые функторы)

Функторы $F : \mathcal{C} \to \mathcal{D}$ и $G : \mathcal{D} \to \mathcal{C}$ образуют **сопряжение** $F \dashv G$, если стрелки $F(X) \to Y$ взаимно однозначно соответствуют стрелкам $X \to G(Y)$ — одним правилом, естественным образом.

Сопряжение — *центральное* понятие теории: пределы, свободные структуры и универсальные конструкции получаются из подходящих сопряжений.

Свободное и забывающее

Пример (Списки)

Забывающий функтор U выбрасывает операцию моноида. Свободный функтор L превращает множество в моноид списков. Сопряжение $L \dashv U$: гомоморфизмы моноидов $A^* \to M$ — это функции $A \to U(M)$.

Гомоморфизм собирается по значениям на буквах, причём единственным образом.

Для $f(0) = a$, $f(1) = b$ продолжение $L(f)$ переводит список «011» в «abb» — конкатенация сохраняется.

Единица сопряжения $\eta_A(a) = [a]$, коединица ε_M — свёртка: $[1, 2, 3] \mapsto 6$ в $(\mathbb{Z}, +, 0)$.

Связка Галуа

Пример (Абстрактная интерпретация)

Пара функций α, γ между решётками состояний с условием

$$\alpha(X) \preceq d \quad \Leftrightarrow \quad X \subseteq \gamma(d)$$

— это сопряжение между тонкими категориями.

Статический анализ программ построен на этом сопряжении: приближение α и конкретизация γ действуют как сопряжённые функторы, и корректность переноса свойств — прямое следствие сопряжённости.

Каррирование

Пример (Экспоненциал)

В категории Set функции двух аргументов — то же, что функции в функции:

$$\mathrm{Set}(A \times B,\ C) \quad \cong \quad \mathrm{Set}(A,\ C^B).$$

Конкретика: $f(x, y) = x \oplus y$ переходит в $x \mapsto (y \mapsto x \oplus y)$ — функция двух битов становится функцией, возвращающей функцию.

Взятие произведения $- \times B$ сопряжено к экспоненцированию $(-)^B$.

Декартово замкнутая категория — модель просто типизированного лямбда-исчисления.

Монада

Определение 10 (Монада)

Монада — эндофунктор T с единицей $\eta : A \to T(A)$ и умножением $\mu : T(T(A)) \to T(A)$, сплющивающими обёртки согласованным образом.

Пример (Option и список)

`Option` — единица `Some`, умножение `join`, внешний `None` отказывает всё. Список — единица одноэлементный список, умножение — конкатенация списка списков.

Сопряжение $L \dashv U$ сворачивается в монаду $T = U \circ L$ с $\mu = U\varepsilon L$ — так рождается монада списка. Механизм общий: любое $F \dashv G$ даёт монаду $G \circ F$.

Категория Клейсли

Стрелки из объекта A в объект B — функции $A \to \mathrm{Option}(B)$. **Композиция Клейсли** склеивает цепочки, отказ любого звена отказывает всё.

```
// Композиция Клейсли: отказ любого звена отказывает цепочку.
fn and_then<T, U, V>(
    parse: impl Fn(T) -> Option<U>,
    validate: impl Fn(U) -> Option<V>,
) -> impl Fn(T) -> Option<V> {
    move |x| parse(x).and_then(|y| validate(y))
}
```

Оператор `?` в Rust, `and_then`, `flat_map` — синтаксис поверх одной диаграммы.

Итоги

Три этажа языка:

1. категории описывают миры
2. функторы переносят между мирами
3. естественные преобразования сравнивают переносы

На этой тройке держатся универсальные свойства, лемма Йонеды, сопряжения и монады.

> Словарь для программиста: типы и функции — категория, полиморфные конструкции — функторы и естественность, эффекты — монады, тип с эффектом — стрелка Клейсли.

Что дальше?

1. **моноидальные категории** и стринг-диаграммы — язык потоковых графов и квантовых схем
2. **высшие категории** — топология и гомотопическая теория типов
3. **топосы** — категории с внутренней логикой, от Гейтинга до зависимых типов
4. **прикладная волна** — базы данных, вероятности, сети, машинное обучение

Словарь при этом один, а глубина владения определяется запасом примеров.