Дискретная математика

Теория типов

Константин Чухарев

Осень 2026

Простые типы

> *Есть два способа построить программу: настолько простую, что недостатков очевидно нет, или настолько сложную, что очевидных недостатков нет.*
>
> **— Тони Хоар**

Типы

Компилятор не даёт сложить число и строку — ловит ошибку до запуска. Типы формализуют, какие термы можно составлять.

Определение 1 (Простые типы)

- Базовые: Nat, Bool.
- Функциональный: $\sigma \to \tau$.

Никаких других типов нет.

Тип — классификация терма по поведению.

Тип — не значение, а инвариант: он фиксирует, что с термом можно делать.

Примеры типов

Пример

- $\mathrm{Bool} \to \mathrm{Bool}$ — отрицание, тождественная.
- $\mathrm{Nat} \to \mathrm{Nat} \to \mathrm{Nat}$ — сложение (каррировано).
- $(\mathrm{Nat} \to \mathrm{Nat}) \to \mathrm{Nat}$ — принимает функцию, возвращает число.

Параллель с логикой

Тип $\sigma \to \tau$ структурно совпадает с импликацией. «Дайте мне σ — верну τ» — то же самое, что «если σ, то τ».

Функция — конструктивное доказательство импликации: из входных данных она строит результат.

Совпадение стрелки типа и импликации не случайно: изоморфизм Карри–Ховарда будет разобран ниже.

Правила типизации

Три правила

Γ — контекст: набор допущений вида $x : \sigma$.

$\Gamma \vdash M : \sigma$ читается как «из Γ выводится, что M имеет тип σ».

$\Gamma, x : \sigma$ — контекст Γ, расширенный новым допущением $x : \sigma$.

Определение 2 (Правила $\lambda \to$)

- **var**: если $x : \sigma \in \Gamma$, то $\Gamma \vdash x : \sigma$.
- **app**: из $M : \sigma \to \tau$ и $N : \sigma$ получаем $MN : \tau$.
- **abs**: из $\Gamma, x : \sigma \vdash M : \tau$ выводим $\Gamma \vdash \lambda x : \sigma . M : \sigma \to \tau$.

Каждое правило — шаг логического вывода.

Терм корректен, если есть дерево вывода.

Примеры типизации

Пример

- $\lambda x : \mathrm{Nat}\ .x$ имеет тип $\mathrm{Nat} \to \mathrm{Nat}$.
- $\lambda x : \mathrm{Nat}\ .\lambda y : \mathrm{Bool}\ .x$ имеет тип $\mathrm{Nat} \to \mathrm{Bool} \to \mathrm{Nat}$.

Второй аргумент игнорируется — тип это фиксирует.

Проверка типов — алгоритмический поиск дерева.

Для $\lambda \to$ проверка типа занимает линейное время.

Типизируемость

Что типизируется

- $K = \lambda x.\lambda y.x$: тип $\sigma \to \tau \to \sigma$.
- $S = \lambda x.\lambda y.\lambda z.xz(yz)$: типизируется.

Типы K и S — аксиомы минимальной логики.

Что не типизируется

$Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ не типизируется.

Для xx переменная x должна иметь сразу тип $\alpha \to \beta$ и α, что даёт $\alpha = \alpha \to \beta$. Конечный тип не может быть собственным подвыражением себя.

Рекурсивных термов в $\lambda \to$ нет.

Поэтому просто типизированное исчисление не Тьюринг-полно.

Это цена за гарантию остановки.

Свойства

Теорема 1 (Сильная нормализация)

Каждый *типизируемый* терм $\lambda \to$ редуцируется к нормальной форме.

Порядок редукции не важен.

Доказательство

Сильная нормализация доказывается индукцией по типам методом приводимости Тейта.

Каждому типу σ сопоставляется множество приводимых термов: для базового типа это (после подстановки) термы, редуцирующиеся к значению, а терм в $\sigma \to \tau$ приводим, если его результат при любом приводимом аргументе приводим в τ; нейтральные термы приводятся, как только приводятся их свободные переменные.

Свойства: индукция по выводу

Доказательство

Индукцией по типам доказываются два свойства: каждый терм типа σ редуцируется к нормальной форме, и каждый терм типа σ с нормализующимися свободными переменными приводим в σ.

Затем индукцией по выводу типизации показывается, что всякий типизируемый терм приводим в своём типе, и потому нормализуется.

Свойства: вывод

Типизируемые термы всегда завершаются.

Проверка типов — гарантия остановки.

Сохранение типа при редукции: каждый шаг $\overset{\beta}{\twoheadrightarrow}$ сохраняет тип терма.

Сильная нормализация: любая цепочка β-редукций корректно типизированного терма обрывается на нормальной форме.

Тип — инвариант вычисления

Пример

$(\lambda x : \mathrm{Nat}\ .x)3 \overset{\beta}{\twoheadrightarrow} 3$, и тип Nat не изменился.

Терм изменился, тип Nat остался.

Тип приписан синтаксису: выводится до запуска, от значения не зависит.

Значение может меняться, тип — нет.

Тип — статическая характеристика программы, верная на каждом шаге.

Карри–Ховард

Соответствие

Теория типов	Логика
тип σ	высказывание
терм $M : \sigma$	доказательство
$\sigma \to \tau$	импликация
абстракция	$\to$-введение
аппликация	modus ponens

Карри–Ховард — изоморфизм между просто типизированным λ-исчислением и интуиционистской логикой высказываний.

Не метафора: каждая строка — точное соответствие.

Термы как доказательства

Пример

- $\lambda x : A.x$ — доказательство $A \to A$.
- $\lambda x : A.\lambda y : B.x$ — аксиома $A \to B \to A$.

Композиция — транзитивность импликации:

$\lambda f : A \to B.\lambda g : B \to C.\lambda x : A.g(fx)$.

Терм и доказательство выражают одно и то же разными словами, но с одной структурой.

Два направления

- Доказываете теорему — терм и есть программа.
- Пишете типизируемую программу — тип и есть теорема.
- Компилятор — верификатор доказательства.

Интуиционистская логика требует конструктивных доказательств.

Три линии сошлись в одном изоморфизме:

- Вычислительная (Чёрч).
- Логическая (Гейтинг).
- Инженерная (де Брёйн).

За пределы $\lambda \rightarrow$

Расширения соответствия

Карри–Ховард не ограничен импликацией.

Каждая новая конструкция типов — новая логическая связка.

Теория типов	**Логика**
тип-произведение $\sigma \times \tau$	конъюнкция $\sigma \wedge \tau$
тип-сумма $\sigma + \tau$	дизъюнкция $\sigma \vee \tau$
пустой тип $\bot$	ложь, ex falso quodlibet

Пара населяет $\sigma \times \tau$ — значит, есть и σ, и τ.

Из $\bot$ выводится любое утверждение — принцип ex falso.

λ-куб

Три вида зависимости между термами и типами — три оси куба Барендрегта.

Определение 3 (Три оси)

- $\lambda\to$: термы зависят от термов.
- $\lambda 2$: термы зависят от типов — полиморфизм.
- λP: типы зависят от термов — зависимые типы.
- $\lambda\omega$: типы зависят от типов — операторы над типами.

Пример

Оператор над типами: $\mathrm{List} : \star \to \star$.

Оператор List отображает тип элементов в тип списка.

Вершина куба

Вершина куба — λC (Calculus of Constructions): все три зависимости сразу.

λC с индуктивными типами — основа Coq.

Полиморфизм: $\lambda 2$

В $\lambda \to$ терм монотипен (*monomorphic*): один фиксированный тип.

System F даёт один терм для всех типов сразу.

Определение 4 (System F ($\lambda 2$))

Термы зависят от типов.

Квантор $\forall \alpha$ связывает переменную типа.

Пример

Полиморфная тождественная функция:

$\Lambda \alpha . \lambda x : \alpha . x : \forall \alpha . \alpha \to \alpha$.

Зависимые типы: λP

Определение 5 (Зависимые типы)

Типы зависят от термов.

Тип $\mathrm{Vec}\ n$ — вектор длины n — меняется вместе со значением n.

Пример

Конкатенация векторов фиксирует длины в типе:

$\mathrm{concat} : \Pi n m : \mathrm{Nat}\ .\ \mathrm{Vec}\ n \to \mathrm{Vec}\ m \to \mathrm{Vec}\ (n + m)$.

Тип $\mathrm{Even}\ n$ населён ровно тогда, когда n чётно.

Зависимый тип — не булева функция, а утверждение о значении.

Пруф-ассистенты

Ассистенты доказательств превращают изоморфизм Карри–Ховарда в инструмент верификации.

Доказательство — это терм, а проверка доказательства — это проверка типа.

Пример

- Coq: корректность компилятора CompCert, теорема о четырёх красках.
- Lean 4: формализация современной математики.
- Agda: язык программирования и пруф-ассистент одновременно.

Принцип де Брёйна: ядро проверки типов — маленькая программа.

Если ядро приняло терм, доказательство корректно, как бы оно ни было получено.

Строгость и выразительность

Чем строже система типов, тем больше ошибок отлавливается на этапе компиляции.

Тем труднее выразить некоторые корректные программы.

Современные языки (Haskell, Rust, OCaml) ищут баланс: достаточно богатые типы для рекурсии и полиморфизма, достаточно строгие — чтобы исключить целые классы ошибок.

Логическая цена строгости: рекурсия, Y-комбинатор, неподвижные точки требуют выхода за пределы $\lambda \to$.

Полиморфизма System F недостаточно — нужны рекурсивные типы или оператор неподвижной точки.

Что мы поняли?

- Тип — инвариант о поведении терма. Функциональный тип $\sigma \to \tau$ структурно совпадает с импликацией.
- Типизируемость — существование дерева вывода. Сильная нормализация даёт гарантию остановки.
- Карри–Ховард: терм — доказательство, тип — высказывание, проверка типов — проверка доказательства.
- Зависимые типы типизируют утверждения о значениях. Пруф-ассистенты превращают изоморфизм в инструмент верификации.

Вопросы для самопроверки

- Определите простые типы.
- Сформулируйте три правила типизации.
- Почему $\lambda x : \mathrm{Nat}\ .\lambda y : \mathrm{Bool}\ .x$ типизируется?
- Почему Y не типизируется?
- Что гарантирует сильная нормализация?
- Сформулируйте соответствие Карри–Ховарда.
- Какие логические связки дают тип-произведение, тип-сумма и пустой тип?
- Назовите три оси λ-куба Барендрегта.
- Какой тип у полиморфной тождественной функции?
- Почему тип $\mathrm{Vec}\ n$ невыразим в $\lambda{\to}$?