

# Теория типов

Михайлов Максим

20 сентября 2021 г.

## Оглавление

Лекция 1	7 сентября	2
1	Лямбда-исчисление . . . . .	2
1.1	Определение . . . . .	2
1.2	Булево исчисление . . . . .	2
1.3	Числа . . . . .	3
1.4	Типизированное лямбда-исчисление . . . . .	4
1.5	Y-комбинатор и противоречивость нетипизированного $\lambda$ -исчисления .	4
Лекция 2	14 сентября	6

# Лекция 1

## 7 сентября

### 1 Лямбда-исчисление

То, чем мы будем заниматься, можно назвать прикладной матлогикой.

В рамках курса матлогики мы рукомахательно рассмотрели изоморфизм Карри-Ховарда, в этом курсе мы его формализуем. Мы затронем систему типов Хиндли-Милнера (*Haskell*) и язык *Arend*, основанный на гомотопической теории типов.

#### 1.1 Определение

В 20-30х годах XX века Алонзо Чёрчем была создана альтернатива теории множеств как основе математики — лямбда-исчисление. Основная идея — выбросить из языка все, кроме вызова функций.

В лямбда исчислении есть три конструкции:

- Функция (*абстракция*):  $(\lambda x. A)$
- Применение функции (*аппликация*):  $(A B)$
- Переменная (*атом*):  $x$

Большими буквами начала латинского алфавита мы будем обозначать термы, малыми буквам конца — переменные.  $\lambda$  жадная, как  $\forall$  и  $\exists$  в исчислении предикатов. Аппликация идёт слева направо, т.е.  $\lambda p. p F T = \lambda p. ((p F) T)$

Вычисление происходит с помощью  $\beta$ -редукции, его мы определим позже, общее понимание у нас есть из вводной лекции функционального программирования.

#### 1.2 Булево исчисление

Определим булево исчисление в  $\lambda$ -исчислении:

- $T := \lambda x. \lambda y. x$  — истина
- $F := \lambda x. \lambda y. y$  — ложь
- $\text{Not} := \lambda p. p \ F \ T$

$$\begin{aligned} \text{Not } F &\rightarrow_{\beta} \\ ((\lambda x. \lambda y. y) \ F) \ T &\rightarrow_{\beta} \\ (\lambda y. y) \ T &\rightarrow_{\beta} T \end{aligned}$$

- $\text{And} := \lambda a. \lambda b. a \ b \ F$

And берёт свой второй аргумент, если первый аргумент истина и ложь иначе.

And использует идею **карринга** — функция от 2 аргументов есть функция от первого аргумента, возвращающая другую функцию от второго аргумента.<sup>1</sup> Например, в выражении “((+) 2) 3” ((+) 2) это функция, которая прибавляет к своему аргументу 2.

### 1.3 Числа

Числа в лямбда-исчислении кодируются **нумералами Чёрча**. Это только один из способов кодировки, есть и другие. Общая идея — число  $n$  применяет данную функцию к данному аргументу  $n$  раз.

- $0 = \lambda f. \lambda x. x$
- $1 = \lambda f. \lambda x. f \ x$
- $3 = \lambda f. \lambda x. f \ (f \ (f \ x))$
- $\overline{n+1} = \lambda f. \lambda x. f \ (\overline{n} \ f \ x)$
- $(+1) = \lambda n. \lambda f. \lambda x. n \ f \ (f \ x)$  — функция инкремента.
- $(+) = \lambda a. \lambda b. b \ ((+) \ \overline{1}) \ a$ :  $b$  раз прибавляет единицу к  $a$ .
- $(\cdot) = \lambda a. \lambda b. a \ ((+) \ b) \ \overline{0}$ :  $a$  раз прибавляет  $b$  к 0.

Ходят легенды, что Клини изобрел декремент у зубного врача под действием наркоза. Существует много способов определить декремент различных степеней упоротости.

Рассмотрим декремент, основанный на следующей идее: пусть есть упорядоченная пара  $\langle a, b \rangle$  и функция  $(*) : \langle a, b \rangle \mapsto \langle b, b+1 \rangle$ . Тогда применив  $(*)$   $n$  раз к  $\langle 0, 0 \rangle$  и взяв первый элемент, возьмём первый элемент пары.

<sup>1</sup> Аналогично для  $n$  аргументов.

Упорядоченная пара определяется следующим способом:

$$\text{MkPair} = \lambda a. \lambda b. (\lambda p. p \ a \ b)$$

Можно потрогать эмулятор лямбда-исчисления lci, будет полезно для домашних заданий.

## 1.4 Типизированное лямбда-исчисление

Лямбда-исчисление для нас будет просто языком программирования. Для начала мы его типизируем, потому что нетипизированное лямбда-исчисление противоречиво.

Пусть у каждого выражения  $A$  есть тип  $\tau$ , что обозначается  $A : \tau$ . Также используется некоторый контекст с переменными и их типами, обозначаемый  $M$ . Все вместе это записывается как  $M \vdash A : \tau$ , что напоминает исчисление предикатов.

## 1.5 $Y$ -комбинатор и противоречивость нетипизированного $\lambda$ -исчисления

Мы хотим, чтобы  $\rightarrow_\beta$  сохраняло значения, т.к. иначе мы вообще не можем говорить о равенстве термов.

**Определение.**  $Y := \lambda f. (\lambda x. f \ (x \ x)) (\lambda x. f \ (x \ x))$  —  $Y$ -комбинатор, для него верно  $Y f \approx f(Y f)$ . Такое свойство называется “быть комбинатором неподвижной точки”, т.е. он находит неподвижную точку функции:  $A$  такое, что  $f(A) = A$ .

Пусть мы добавили бинарную операцию  $(\supset)$  — импликацию с некоторыми аксиомами. Оказывается, что доказуемо любое  $A$ . Мы это докажем на последующих лекциях.

$Y$ -комбинатор полезен тем, что позволяет реализовывать рекурсию.

*Пример.* Запишем факториал в неформальном виде:

$$\text{Fact} = \lambda n. \text{If} \ (\text{IsZero } n) \ \bar{1} \ (\text{Fact } (n - 1) \cdot n)$$

На самом деле  $\text{Fact}$  есть неподвижная точка функции

$$\lambda f. \lambda n. \text{If} \ (\text{IsZero } n) \ \bar{1} \ (f \ (n - 1) \cdot n)$$

по определению неподвижной точки функции. Тогда  $\text{Fact}$  это

$$Y(\lambda f. \lambda n. \text{If} \ (\text{IsZero } n) \ \bar{1} \ (f \ (n - 1) \cdot n))$$

У нас появляется проблема: есть выражения, которым мы не можем приписать значение, например

$$Y(\lambda f. \lambda x. f \ (\text{Not } x))$$

Эта проблема происходит из-за того, что наш язык слишком мощный — мы написали решатель любых уравнений, даже тех, у которых нет решения. Логичный выход из этой ситуации — запретить то, из-за чего у нас возникают проблемы. Как запретить  $Y$ ? Оказывается, это позволяют сделать типы — они будут делить выражения на добропорядочные и недобропорядочные.

# Лекция 2

## 14 сентября

**Определение.** Пред- $\lambda$ -терм определяется индуктивно как одно из:

1.  $x$  — переменная
2.  $(L\ L)$  — применение
3.  $(\lambda x.L)$  — абстракция

Почему пред- $\lambda$ -терм? Мы не хотим различать  $\lambda x.x$  и  $\lambda y.y$ .

**Определение.**  $\alpha$ -эквивалентность — обозначается  $A =_\alpha B$  и выполняется, если<sup>1</sup>:

1.  $A \equiv x, B \equiv x$  — одна и та же переменная
2.  $A \equiv P\ Q, B \equiv R\ S, P =_\alpha R, Q =_\alpha S$
3.  $A \equiv \lambda x.P, B \equiv \lambda y.Q$  и существует  $t$  — новая переменная, такая что  $P[x := t] =_\alpha Q[y := t]$

**Определение.** Свобода для подстановки:  $A[x := B]$ , никакое свободное вхождение переменной в  $B$  не станет связанным.

**Определение** ( $\lambda$ -терм). Множество всех  $\lambda$ -термов это  $\Lambda / =_\alpha$

**Определение** ( $\beta$ -редекс). Выражение вида  $(\lambda x.A)\ B$

**Определение** ( $\beta$ -редукция). Обозначается  $A \rightarrow_\beta B$  и выполняется, если выполняется одно из:

1.  $A \equiv P\ Q, B \equiv R\ S$  и либо  $P \rightarrow_\beta R$  и  $Q =_\alpha S$ , либо  $P =_\alpha R$  и  $Q \rightarrow_\beta S$ .
2.  $A \equiv \lambda x.P, B \equiv \lambda x.Q$  и  $P \rightarrow_\beta Q$
3.  $A \equiv (\lambda x.P)\ Q, B \equiv P[x := Q]$  и  $Q$  свободно для подстановки.

---

<sup>1</sup> И только если.

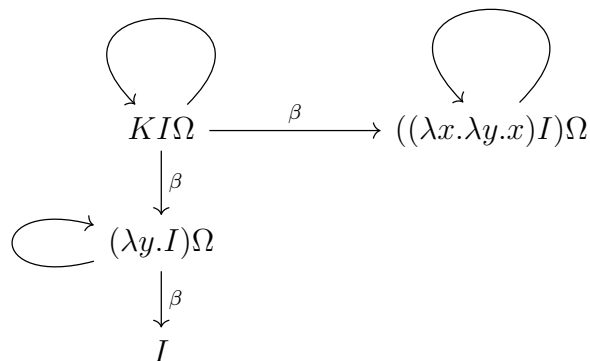
**Определение.** Придуман Моисеем Шейнфинкелем.

$I := \lambda x.x$  — Identität<sup>2</sup>

**Определение.**

- $K = \lambda x.\lambda y.x$
- $\Omega = \omega \omega$
- $\omega = \lambda x.x x$

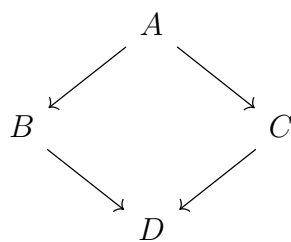
**Пример.**



**Определение.**  $R$  обладает ромбовидным свойством (*diamond*), если для любых  $a, b, c$ , таких что:

1.  $a R b, a R c$
2.  $b \neq c$

существует  $d$ :  $b R d$  и  $c R d$ .



**Пример.**  $>$  на  $\mathbb{Z}$  не ромбовидно: для  $a = 3, b = 2, c = 1$  выполнено условие, но  $\nexists d$ .

$>$  на  $\mathbb{R}$  ромбовидно.

**Определение** ( $\beta$ -редуцируемость). Рефлексивное, транзитивное замыкание отношения  $\rightarrow_\beta$ , обозначается  $\twoheadrightarrow_\beta$ .

**Теорема 1** (Чёрча-Россера).  $\beta$ -редуцируемость обладает ромбовидным свойством.

---

<sup>2</sup> Тожество (с немецкого)

**Определение.**  $\Rightarrow_\beta$  — параллельная  $\beta$ -редукция, выполняется если:

0.  $A =_\alpha B$
1.  $A \equiv P Q, B \equiv R S$  и  $P \Rightarrow_\beta R$  и  $Q \Rightarrow_\beta S$ .
2. Аналогично  $\beta$ -редукции.
3. Аналогично  $\beta$ -редукции.

**Лемма 1.**  $(\Rightarrow_\beta)$  обладает ромбовидным свойством.

**Лемма 2.** Если  $R$  обладает ромбовидным свойством, то  $R^*$  обладает ромбовидным свойством.

*Доказательство.* Две индукции. □

**Лемма 3.**  $(\Rightarrow_\beta) \subseteq (\rightarrow_\beta)$

*Доказательство теоремы Чёрча-Россера.* Заметим, что:

1.  $(\Rightarrow_\beta)^* \subseteq (\rightarrow_\beta)$  — из леммы
2.  $(\rightarrow_\beta) \subseteq (\Rightarrow_\beta)^*$  — из определения
3. Т.к.  $(\Rightarrow_\beta)^*$  обладает р.с., то и  $(\rightarrow_\beta)$  обладает р.с.

□

**Следствие 1.1.** У  $\lambda$ -выражения существует не более одной нормальной формы.

*Доказательство.* Пусть  $A$  имеет две нормальные формы:  $A \rightarrow_\beta B, A \rightarrow_\beta C$  и  $B \neq_\alpha C$ . Тогда есть  $D$ :  $B \rightarrow_\beta D$  и  $C \rightarrow_\beta D$ . Противоречие. □

**Определение.** Нормальный порядок редукции — редуцируем самый левый редекс.

**Теорема 2.** Если нормальная форма существует, она может быть получена нормальным порядком редукции.

*Примечание.* Нижеследующее объяснение — с практики.

Рассмотрим  $Y f =_\beta f (Y f) =_\beta f (f (Y f)) =_\beta \dots$ . Можно считать, что у  $f$  сколько угодно аргументов, первый аргумент можно считать указателем на свой рекурсивный вызов.

*Пример.* Числа Фибоначчи:

```
fib a b n =
  if n = 0 then a
  else fib b (a + b) (n - 1)
```



Здесь решение уравнения заматано под ковер, в  $\lambda$ -исчислении оно видно:

$$\text{Fib} = \lambda f. \lambda a. \lambda b. \lambda n. (\text{IsZero } n) \ a \ (f \ f \ a \ (a + b) \ (n - 1))$$

Здесь  $f$  передается само себе, чтобы иметь ссылку на себя для рекурсивного вызова.

Для работы Fib нужно дать его самому себе: `Fib Fib 1 1 10`.