Теория типов: Конспект

Шубин Владислав

31 января 2025 г.

Оглавление

Глава 1

Введение

Теория типов представляет собой новое направление в логике, изучающее системы типов. С математической точки зрения существует два пути происхождения теории типов. Первый из них есть анализ математического текста, например из книги Шафаревича [1] мы видим следующее «для $y \in Y$ и $x \in f^{-1}(y)$ мы получаем уравнение

$$t_i(x)^k + a_1(y)t_i(x)^{k-1} + \cdots + a_k(y) = 0.\text{»}$$

В этом контексте мы хотим анализировать подобные высказывания, которые понимаются, обычно, интуитивно. Например, здесь бессмысленно было бы сказать $x \in \Bbbk$, по причине того, что x — это точка аффинной схемы, но не элемент поля. В этом смысле они обладают разными «типами». Человек, знакомый с алгебраической геометрией, понимает, что $k \in \mathbb{N}$, $t_i : X \to \Bbbk$, $a_i : Y \to \Bbbk$, где $\Bbbk$ есть поле.

Подобные выражения недоступны в языке логики первого порядка и для того, чтобы оперировать с такой «математической грамматикой» нам понадобится теория типов

В теории типов мы, подобно символу $\in$, используем символ $:$, означающий «иметь тип». Например, $x : A$ означает «x имеет тип A». Пусть у нас есть несколько типов $x : A, y : B$, которые мы отделяем запятой. Если из них возможен вывод, мы его будем называть суждением, а посылку контекстом. То есть

$$\underbrace{x : A, y : B}_{\text{контекст}} \vdash \underbrace{t : C}_{\text{суждение}} .$$

Кроме чисто математической точки зрения мы имеем довольно важную практическую сторону этой теории — это языки программирования. С точки зрения компьютера (если у него, конечно, может быть какая-либо точка зрения) любые данные представляют из себя кусок бинарного кода. Картинки, программы, музыка — все это есть лишь последовательность нулей и единиц. Поэтому в ранних языках программирования существовала проблема типизации. Для примера возьмем Си:

```c
void rev(char *str, int len){
    int start = 0;
    int end = len - 1;
    while(start < end) {
        char tmp = str[start];
        str[start] = str[end];
        str[end] = tmp;
        end--;
        start++;
    }
}
```

Современный язык Си использует систему типов, в которой тип `char` представляет символьный тип. Однако, нет никакой существенной разницы между типами `char` и `short`, например (второй тип представляет однобайтовое число). Поэтому эту функцию можно применить и к массиву чисел, что создает существенные проблемы при разработке сложного ПО.

Из-за этой проблемы сформировалось, своего рода, ad hoc решение, называемое строгой типизацией.

Замечание 1.0.1. *В математике любой объект можно представить как некоторое множество, однако это не лишает теорию типов приложимости. Нам важно, что натуральное число — это не просто множество, а что-то новое, обладающие свойствами «числа».*

Пример 1. Можно представить следующий пример вычислений с типами

$$\frac{m, n : \mathbb{N} \vdash 2 \cdot m + n : \mathbb{N} \qquad \vdash 2 : \mathbb{N}}{\vdash 2 \cdot 5 + 3 : \mathbb{N}}$$

1.1 Простая теория типов

Введем нашу первую модель теории типов формально. Типы мы определим как

$$A := 1 \mid A_1 \times A_2 \mid \mathsf{Nat},$$

то есть свободно порожденное множество с элементами $1 \in \mathsf{Ty}$, $\mathsf{Nat} \in \mathsf{Ty}$, $\times : \mathsf{Ty}^2 \to \mathsf{Ty}$. Под 1 мы имеем ввиду унарный тип. И, как и во всякой теории, нам понадобятся термы

$$t := x \mid \langle\rangle \mid \langle t_1, t_2\rangle \mid p_1(t) \mid p_2(t) \mid 0 \mid S(t) \mid \mathrm{rec}(...),$$

где rec — рекурсия, чьи аргументы мы уточним позже.

Далее нам понадобится контекст, определяемый как $\Gamma := \langle\rangle \mid \Gamma, x : A$, то есть контексты суть конечные списки переменных с типами (например, $x_1 : A_1, \ldots, x_n : A_n$).

Теперь займемся суждением о типе — отношении $\Gamma \vdash t : A$ или «в контексте Γ переменная t имеет тип A» что аналогично «выводимости» в логике первого порядка.

1.1.1 Аксиомы

Первая часть правил будет относиться к так называемым структурным правилам. Под i мы будем иметь в виду любой тип в контексте.

Перечислим их:

- (Axiom, Identity, Assumption)

$$\frac{}{x_1 : A_1, \ldots, x_n : A_n \vdash x_i : A_i}$$

- Подстановка (Substitution)

$$\frac{\Delta \vdash s_i : A_i \quad \Gamma \vdash t : C}{\Delta \vdash t[s_i/x_i] : C}$$

Аксиому подстановки можно вывести из следующих аксиом:

- Ослабление (weakening):

$$\frac{\Gamma \vdash t : A}{\Gamma, x : B \vdash t : A}$$

- Замена (exchange):

$$\frac{\Delta \vdash s_i : A_i \quad \Gamma \vdash t : C}{\Gamma, x : A, y : B \quad \Delta \vdash t : C}$$

- Замена одной переменной:

$$\frac{\Gamma, x : A \vdash t : B \qquad \Gamma \vdash a : A}{\Gamma \vdash t[a/x] : B}$$

Пример 2. Пример работы аксиомы подстановки

$$\frac{y : \mathbb{N} \vdash y \cdot y : \mathbb{N} \quad x_1, x_2 : \mathbb{N} \vdash x_1 + x_2 : \mathbb{N}}{x_1, x_2 : \mathbb{N} \vdash x_1 \cdot x_1 + x_2 \cdot x_2 : \mathbb{N}}$$

Наша система все еще не является теорией типов, поскольку у нас нет правил вывода типов. Давайте их введем:

- Произведение

$$\frac{\Gamma \vdash t_1 : A_1, \Gamma \vdash t_2 : A_2}{\Gamma \vdash \langle t_1, t_2 \rangle : A_1 \times A_2}$$

- Проекция

$$\frac{\Gamma \vdash t : A_1 \times A_2}{\Gamma \vdash p_i(t) : A_i}$$

- 0

$$\frac{}{\Gamma \vdash 0 : \mathsf{Nat}}$$

- $S(n)$

$$\frac{\Gamma \vdash n : \mathsf{Nat}}{\Gamma \vdash S(n) : \mathsf{Nat}}$$

- 1

$$\frac{}{\Gamma \vdash \langle\rangle : 1}$$

- rec

$$\frac{\Gamma \vdash n : \mathsf{Nat} \qquad \Gamma \vdash t_0 : C \qquad \Gamma, x : \mathsf{Nat}, y : C \vdash t_s(x, y) : C}{\Gamma \vdash \mathrm{rec}(t_0; (x, y, t_s); n) : C}$$

Чтобы определить рекурсию на $n \in \mathbb{N}$ нам нужна база $n = 0$ и индуктивный переход $n = S(m)$.

Теперь мы ввели все, что нужно для теории типов. Однако, стоит учитывать один нюанс.

Аннотации

Давайте посмотрим внимательно на аксиому проекции

$$\frac{\Gamma \vdash t : A_1 \times A_2}{\Gamma \vdash p_i(t) : A_i}$$

Из этой аксиомы нельзя точно понять, на каком типе определена функция p_i, но мы видим этот тип из контекста. В теории типов существует аннотирование, которое приписывает на каком типе определена функция. С ней наша аксиома должна выглядеть как

$$\frac{\Gamma \vdash t : A_1 \times A_2}{\Gamma \vdash p_i^{A_1,\,A_2}(t) : A_i}$$

Аннотация присутствует всегда, но для читаемости ее не выписывают. Нам также необходимо аннотировать аксиомы пары, поскольку функция пары для каждого типа своя:

$$\frac{\Gamma \vdash t_1 : A_1, \Gamma \vdash t_2 : A_2}{\Gamma \vdash \langle t_1, t_2\rangle^{A_1,\,A_2} : A_1 \times A_2}$$

Для рекурсии нужно аннотировать саму функцию rec:

$$\frac{\Gamma \vdash n : \mathsf{Nat} \qquad \Gamma \vdash t_0 : C \qquad \Gamma, x : \mathsf{Nat}, y : C \vdash t_s(x,y) : C}{\Gamma \vdash \mathrm{rec}^C(t_0; (x,y,t_s); n) : C}$$

Упрощение выражений

Мы уже ввели теорию типов. Для нее не обязательны аксиомы для упрощения выражений, но давайте их введем для удобства.

$$\frac{\Gamma \vdash t_1 : A_1, \Gamma \vdash t_2 : A_2}{\Gamma \vdash p_i\langle t_1,\, t_2\rangle \rightsquigarrow t_i : A_i}$$

$$\frac{\Gamma \vdash t_0 : C \qquad \Gamma, x : \mathsf{Nat}, y : C \vdash t_s(x,y) : C}{\Gamma \vdash \mathrm{rec}^C(t_0; (x,y,t_s); 0) : C \rightsquigarrow t_0 : C}$$

$$\frac{\Gamma \vdash m : \mathsf{Nat} \qquad \Gamma \vdash t_0 : C \qquad \Gamma, x : \mathsf{Nat}, y : C \vdash t_s(x,y) : C}{\Gamma \vdash \mathrm{rec}^C(t_0; (x,y,t_s); S(m)) \rightsquigarrow t_s[m/x, \mathrm{rec}^C(t_0; (x,y,t_s); m)/y] : C}$$

Проведем теперь испытание наших определений и “запрограммируем” что-нибудь. Начнем с функции суммы. Как известно $x + 0 = x$ и $x + Sy = S(x + y)$. Тогда мы можем в нашей системе типов определить функцию $\mathrm{add}(x, y)$ как

$$\cfrac{\cfrac{x, y : \mathsf{Nat} \vdash y : \mathsf{Nat} \qquad x, y : \mathsf{Nat} \vdash x : \mathsf{Nat} \qquad x, y, z, w : \mathsf{Nat} \vdash S(w) : \mathsf{Nat}}{x, y : \mathsf{Nat} \vdash \mathrm{rec}(x; z, w, S(w); y) : \mathsf{Nat}}}{\mathrm{add}(x, y)}$$

Пример 3.

$$\mathrm{add}(S0, 0) \equiv \mathrm{rec}(S0; z, w, S(w); 0) \rightsquigarrow S0,$$

то есть $1 + 0 = 1$.

$$\begin{aligned} \mathrm{add}(SS0, SS0) \equiv \mathrm{rec}(SS0; z, w, S(w); SS0) \rightsquigarrow Sw[\mathrm{rec}(SS0; z, w, S(w); S0)/w] \equiv \\ \equiv S(\mathrm{rec}(SS0; z, w, S(w); S0)) \rightsquigarrow S(Sw[\mathrm{rec}(SS0; z, w, S(w); 0)]/w) \equiv \\ \equiv SS(\mathrm{rec}(SS0; z, w, S(w); 0)) \rightsquigarrow SSSS(0), \end{aligned}$$

действительно $2 + 2 = 4$.

1.2 Модель теории множеств для теории типов

Мы будем использовать стандартную модель, где для каждого типа A существует множество $[\![A]\!]^{\mathcal{M}}$, где $\mathcal{M}$ — наша модель. Мы будем её опускать, поскольку она стандартная. Далее мы определяем множества рекурсивно по типу A:

- $[\![1]\!] := 1$ (или любой другой синглтон)
- $[\![A \times B]\!] := [\![A]\!] \times [\![B]\!]$
- $[\![\mathbb{N}]\!] := \mathbb{N}$

Для контекста $\Gamma = (x_1 : A_1, \ldots, x_n : A_n)$ мы определяем множество $[\![\Gamma]\!] := \prod_i [\![A_i]\!]$. Далее нам требуется рекурсивно определить правила типизации. Это можно сделать двумя способами.

Мы для любого контекста Γ и типа A мы определим функцию $[\![\Gamma]\!] \underset{[\![\Gamma \vdash t:A]\!]}{\rightarrow} [\![A]\!]$. Она будет правильно определена тогда и только тогда, когда соблюдены правила типизации. Иначе мы получим undefined. Итак, будем действовать рекурсивно по терму t:

- $[\![\Gamma]\!] \overset{[\![\Gamma \vdash x:A]\!]}{\longrightarrow} [\![A]\!] := \begin{cases} [\![\Gamma]\!] \overset{\pi_x}{\longrightarrow} [\![A]\!] & \text{если } (x : A) \in \Gamma, \\ \text{undefined} & \text{иначе} \end{cases}.$
- $[\![\Gamma]\!] \overset{[\![\Gamma \vdash \langle\rangle:A]\!]}{\longrightarrow} [\![A]\!] := \begin{cases} [\![\Gamma]\!] \longrightarrow [\![A]\!] = 1 & \text{если } A = 1, \\ \text{undefined} & \text{иначе} \end{cases}.$
- $[\![\Gamma]\!] \overset{[\![\Gamma \vdash \langle a,b\rangle^{B_1 \times B_2}:A]\!]}{\longrightarrow} [\![A]\!] := \begin{cases} [\![\Gamma]\!] \overset{\langle [\![\Gamma \vdash b_1:B_1]\!], [\![\Gamma \vdash b_2:B_2]\!]\rangle}{\longrightarrow} [\![A]\!] = [\![B_1]\!] \times [\![B_2]\!] & \text{если } A = B_1 \times B_2, \\ \text{undefined} & \text{иначе} \end{cases}.$
- $[\![\Gamma]\!] \overset{[\![\Gamma \vdash \pi_i^{B_1,B_2} p:A]\!]}{\longrightarrow} [\![A]\!] := \begin{cases} [\![\Gamma]\!] \overset{\pi_i [\![\Gamma \vdash p:B_1 \times B_2]\!]}{\longrightarrow} [\![A]\!] & \text{если } A = B_i \in \Gamma, \\ \text{undefined} & \text{иначе} \end{cases}.$
- $[\![\Gamma]\!] \overset{[\![\Gamma \vdash 0:A]\!]}{\longrightarrow} [\![A]\!] := \begin{cases} [\![\Gamma]\!] \overset{0}{\longrightarrow} [\![\mathbb{N}]\!] & \text{если } A = \mathbb{N} \in \Gamma, \\ \text{undefined} & \text{иначе} \end{cases}.$
- $[\![\Gamma]\!] \overset{[\![\Gamma \vdash S_n:A]\!]}{\longrightarrow} [\![A]\!] := \begin{cases} [\![\Gamma]\!] \overset{[\![\Gamma \vdash n:\mathbb{N}]\!]}{\longrightarrow} (\mathbb{N} \to \mathbb{N}) & \text{если } A = \mathbb{N}, \\ \text{undefined} & \text{иначе} \end{cases}.$

Литература

[1] Шафаревич И.Р. *Основы алгебраической геометрии.* УМН, 24:6(150) (1969), 3–184; Russian Math. Surveys, 24:6 (1969), 1–178.

[2] Robert Harper. *Type Systems for Programming Languages.* School of Computer Science, Carnegie Mellon University, Spring, 2000, url: https://people.mpi-sws.org/~dreyer/ats/papers/harper-tspl.pdf

[3] Per Martin-Löf. *Intuitionistic Type Theory.* Bibliopolis. url: https://archive-pml.github.io/martin-lof/pdfs/Bibliopolis-Book-retypeset-1984.pdf

[4] Bengt Nordström, Kent Petersson, Jan M. Smith. *Programming in Martin-Löf's Type Theory.* Department of Computing Sciences, University of Göteborg, Sweden.

[5] Аксель П. и др. *Гомотопическая теория типов.* Программа Унивалентных Оснований, Иститут Перспективных Исследований, пер.: Геннадий Чернышев, url: https://henrychern.wordpress.com/wp-content/uploads/2022/10/hott2.pdf