Теория типов: Конспект

Шубин Владислав

28 февраля 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

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

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:

```
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 используем 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, обладающее свойством «числа».*

Замечание 1.0.2. *Заметим, что с философской точки зрения теория категорий является полностью независимой теорией оснований математики от теории множеств. Существует подход — интуиционисткая теория типов, — позволяющий вывести теорию категорий из теории типов. Иными словами, можно считать, что теория типов является более фундаментальным подходом к основаниям математики, чем другие теории.*

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)

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

Аксиоме подстановки подобны следующие аксиомы:

- 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

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

Замечание 1.1.1. *Разница между* Nat *и* $\mathbb{N}$ *фундаментальна. Первое — это синтаксический класс, класс правил, которым натуральные числа следуют. Второе же — это множество, поэтому мы используем его, подразумевая, что утверждения вроде* $y : \mathbb{N} \vdash y \cdot y : \mathbb{N}$ *нам доказывать не нужно.*

Мы будем пользоваться общей аксиомой подстановки, хотя, иногда имеет смысл ослабить систему теории типов.

Замечание 1.1.2. *Обычно, все доказательства в теории типов — это индукция по типу или терму, с разбором всех случаев. Чем сильнее аксиома, тем больше существует особых случаев. Поэтому в некоторых теориях типов доказательства становятся очень большими и сложными с технической точки зрения.*

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»:

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

- «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 $n = S(m)$. Для этого мы определяем пять переменных t_0, x, y, t_s и n. Значение t_0 и n понятно — это начало и конец рекурсии. Переменные x, y суть предыдущий номер вызова, предыдущие значение. Переменная t_s и есть «следующий элемент» в индукции.

Пример 3.

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

Аннотации

Давайте посмотрим внимательно на аксиому проекции

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

Из этой аксиомы нельзя точно понять, на каком типе определена функция π_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

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

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:

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

Замечание 1.1.3. *Аннотировать можно как и сверху* rec^C*, так и снизу* rec_C*. Для стройности изложения мы предпочтём аннотацию сверху.*

Упрощение выражений

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

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

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

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

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

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

Получается, что для каждого $\Gamma \vdash x, y : \mathsf{Nat}$ мы даем $\Gamma \vdash \mathrm{add}(x, y) : \mathsf{Nat}$.

Пример 4.

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

то есть $1 + 0 = 1$.

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

действительно $2 + 2 = 4$.

Для умножения мы будем использовать уже введенную функцию $\mathrm{add}(x, y)$. Как мы помним, $m \cdot 0 := 0$, $m \cdot S(n) := m \cdot n + m$. Тогда мы можем определить умножение как $\mathsf{mult}(x, y) := \mathrm{rec}^{\mathsf{Nat}}(0; (z, w, \mathsf{add}(w, x)); y)$, если $x, y : \mathsf{Nat}$.

Пример 5. Как мы знаем, $2 \times 2 = 4$, проверим

$$\begin{aligned}\mathsf{mult}(S(S(0)), S(S(0))) \equiv \mathrm{rec}^{\mathsf{Nat}}(0; (z, w\,.\,\mathsf{add}(w, S(S(0)))); S(S(0))) &\rightsquigarrow \\ \rightsquigarrow \mathsf{add}(w, S(S(0)))[\mathrm{rec}^{\mathsf{Nat}}(0; (z, w\,.\,\mathsf{add}(w, S(S(0)))); S(0))/w] &\rightsquigarrow \\ \rightsquigarrow \mathsf{add}(\mathrm{rec}^{\mathsf{Nat}}(0; (z, w\,.\,\mathsf{add}(w, S(S(0)))); S(0)), S(S(0))) &\rightsquigarrow \\ \rightsquigarrow \mathsf{add}(\mathsf{add}(w, S(S(0)))[\mathrm{rec}^{\mathsf{Nat}}(0; (z, w\,.\,\mathsf{add}(w, S(S(0)))); 0)/w], S(S(0))) &\rightsquigarrow \\ \rightsquigarrow \mathsf{add}(\mathsf{add}(\mathrm{rec}^{\mathsf{Nat}}(0; (z, w\,.\,\mathsf{add}(w, S(S(0)))); 0), S(S(0))) &\rightsquigarrow \\ \rightsquigarrow \mathsf{add}(\mathsf{add}(0, S(S(0))), S(S(0))).\end{aligned}$$

И тогда

$$\begin{aligned}\mathsf{add}(\mathsf{add}(0, S(S(0))), S(S(0))) &\rightsquigarrow \\ \rightsquigarrow \mathsf{add}(\mathrm{rec}(0; z, w\,.\,S(w); SS0), S(S(0))) &\rightsquigarrow \\ \rightsquigarrow \mathsf{add}(Sw[\mathrm{rec}(0; z, w\,.\,S(w); S0)/w], S(S(0))) &\rightsquigarrow \\ \rightsquigarrow \mathsf{add}(S(\mathrm{rec}(0; z, w\,.\,S(w); S0)), S(S(0))) &\rightsquigarrow \\ \rightsquigarrow \mathsf{add}(S(Sw[\mathrm{rec}(0; z, w\,.\,S(w); 0)]), S(S(0))) &\rightsquigarrow \\ \rightsquigarrow \mathsf{add}(S(S(\mathrm{rec}(0; z, w\,.\,S(w); 0))), S(S(0))) &\rightsquigarrow \\ \rightsquigarrow \mathsf{add}(S(S(0)), S(S(0))) &\rightsquigarrow^* S(S(S(S(0)))).\end{aligned}$$

В последней стрелке $\rightsquigarrow^*$ имеется ввиду множественность необходимых шагов вывода, которые мы опускаем, потому что сделали это в предыдущем примере.

Пример 6. Более интересно посмотреть на пример с числами Фибоначчи. Для этого мы определим пары $p_n := \langle G_n, H_n\rangle$, где $p_0 = \langle G_0, H_0\rangle = \langle 0, 1\rangle$ и $p_{n+1} = \langle G_{n+1}, H_{n+1}\rangle = \langle H_n, G_n + H_n\rangle$. Тогда мы можем вычислить p_n следующим образом

$$n : \mathsf{Nat} \vdash \mathrm{rec}^{\mathsf{Nat}\times\mathsf{Nat}}(\langle 0, 1\rangle; (z, w\,.\,\langle \pi_2 w, \mathsf{add}(\pi_1 w, \pi_2 w)\rangle); n).$$

Тогда n-ое число Фибоначчи $\mathsf{Fib}(n)$ есть

$$n : \mathsf{Nat} \vdash \pi_1(p_n) : \mathsf{Nat}.$$

1.1.2 Формы суждений

Пусть A тип, тогда мы имеем следующие формы суждений (индуктивно определенные независимые отношения):

- $\Gamma \vdash a : A$
- $\Gamma \vdash a \rightsquigarrow a' : A$
- $\Gamma \vdash a \rightsquigarrow^* a' : A$

Обсуждаемые нами правила могут быть подразделены на следующие две категории:

- выводимые (derivable): если существует вывод правила J из некоторых заданных допущений $J_1, J_2, \ldots, J_n$.
- допустимые (admissible): если из выводимости $J_1, J_2, \ldots, J_n$ следует выводимость J.

Тогда в нашей системе относительно $\rightsquigarrow^*$ будет верно, что

- Следующее правило выводимо

$$\frac{\Gamma \vdash a \rightsquigarrow a' : A \qquad \Gamma \vdash a' \rightsquigarrow a'' : A}{\Gamma \vdash a \rightsquigarrow^* a''}$$

- Следующее правило допустимо, но не выводимо

$$\frac{\Gamma \vdash a \rightsquigarrow^* a' : A \qquad \Gamma \vdash a' \rightsquigarrow^* a'' : A}{\Gamma \vdash a \rightsquigarrow^* a''}$$

Однако, можно доказать, что выводимые правила являются допустимыми. Давайте приведем примеры допустимых правил, которые невыводимы.

Самым простым из них является

$$\frac{J_1}{J_2}$$

Действительно, из пустого набора правил нельзя произвести такое правило, однако, если J_1 и J_2 выводимы, то и все выражение выводимо.

Есть более интересный пример

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

Подобная ситуация называется inversion principle: если тип построен правильно, то и подтип построен правильно. Очевидно, что из наших аксиом мы не можем вывести это правило, однако из выводимости предпосылок будет следовать выводимость самого правила.

Заметим еще одну интересную деталь. Выводимость сохраняется при расширении контекста, тогда как допустимость нет. Пусть J допустимо, тогда при добавлении правила, что из ничего выводится J нарушит допустимость J.

Лемма 1.1.1. *Если $\Gamma \vdash a \rightsquigarrow a' : A$ выводимо, тогда $\Gamma \vdash a : A$ и $\Gamma \vdash a' : A$ выводимы.*

Лемма 1.1.2. *Если $\Gamma \vdash a \rightsquigarrow^* a' : A$ выводимо, тогда $\Gamma \vdash a : A$ и $\Gamma \vdash a' : A$ выводимы.*

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 $[\![\Gamma]\!] \underset{[\![\Gamma \vdash t:A]\!]}{\to} [\![A]\!]$. Она будет правильно определена тогда и только тогда, когда соблюдены правила типизации. Иначе мы получим undefined. Итак, будем действовать рекурсивно по терму t:

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

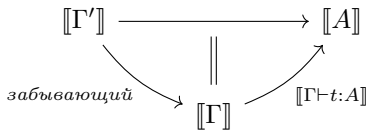
Замечание 1.2.1. *Заметим, что* Nat *— это синтаксический класс натуральных чисел, но не множество натуральных чисел* $\mathbb{N}$.

В этой модели имеют место следующие результаты

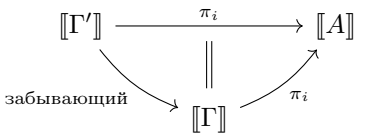
Лемма 1.2.1. *Если* $\Gamma \vdash t : A$*, тогда* $[\![\Gamma \vdash t : A]\!]$ *определен.*

Доказательство. Эта лемма доказывается по индукции по $\Gamma \vdash t : A$. □

Лемма 1.2.2. *Если* Γ' *есть расширение (супермножество) контекста* Γ *и* $[\![\Gamma \vdash t : A]\!]$ *определен, то* $[\![\Gamma' \vdash t : A]\!]$ *определен. В виде диаграммы это выглядит так*



Доказательство. Мы доказываем это по индукции на t. Разберем каждый случай. Начнем со случая когда t есть x_1 (переменная), тогда если $[\![\Gamma \vdash x_i : A]\!]$ определено, то легко видеть, что $x_i[a_i/x_i] \equiv a_i$ и имеет место следующая диаграмма



□

Лемма 1.2.3. *Для данных* Γ, t, A *и замена* $f : \Delta \to \Gamma$ *(например,* $\Gamma = (x_i : A_i)_{i\in I}$*,* $f = (a_i)_{i\in I}$*,* $\Delta \vdash a_i : A_i$*), термы* $(a_i)_{i\in I}$ *и* Δ*, если* $[\![\Gamma \vdash t : C]\!]$ *определено и каждый* $[\![\Delta \vdash a_i : A_i]\!]$ *определен, тогда* $[\![\Delta \vdash t[a_i/x_i] : C]\!]$ *определен и равен*

$$[\![\Delta]\!] \xrightarrow{[\![\Delta\vdash a_i:A_i]\!]} \prod_i [\![A_i]\!] \vdash [\![\Gamma]\!] \xrightarrow{[\![\Gamma\vdash t:C]\!]} [\![C]\!].$$

Доказательство. По индукции на t. □

Пример 7. Пусть $x_1, x_2, x_3 : \mathbb{N} \to x_1 \cdot x_2 \cdot x_3 : \mathbb{N}$ и $\Delta = \omega : \mathbb{N}$. Построим такую замену для $a_1 = \omega, a_2 = \omega + 1, a_3 = \omega + 2$ через

$$(x_1 \cdot x_2 \cdot x_3)[a_i/x_i] = \omega(\omega + 1)(\omega + 2).$$

$$\mathbb{N} \xrightarrow{a_i} \mathbb{N}^3 \xrightarrow{x_1,x_2,x_3} \mathbb{N}$$

$$\omega \longmapsto (\omega, \omega + 1, \omega + 2)$$

$$(x_1, x_2, x_3) \longmapsto x_1 \cdot x_2 \cdot x_3$$

$$\omega \longmapsto \omega(\omega + 1)(\omega + 2)$$

1.2.1 Нормализация

Под нормализацией в теории типов понимают приведение выражения к нормальной или упрощенной форме. Мы уже ввели правила редукции, если последовательное применение их приведет к выражению, которое уже не редуцируется, то мы его будем называть нормальным. Давайте введем эти определения

Определение 1.2.1. Тип $\Gamma \vdash t : A$ называется нормальным, если он максимален относительно редукции $(\rightsquigarrow^*)$

Определение 1.2.2. Терм $\Gamma \vdash t : A$ называется нормализуемым, если существует нормальный тип $\Gamma \vdash t' : A$, такой что $\Gamma \vdash t \rightsquigarrow^* t' : A$.

Напомним, что если $\vdash n : \mathsf{Nat}$, то $\underline{n}$ называется нумералом и обозначает $S^n 0$.

Пример 8. Как мы уже видели $\vdash \underline{2} + \underline{2} : \mathbb{N}$, тогда этот терм нормализуем, поскольку $\vdash \underline{2} + \underline{2} \rightsquigarrow \underline{4} : \mathbb{N}$.

Терм же $\vdash \underline{2} : \mathbb{N}$ нормален, поскольку не существует редукции для него.

Как можно заметить по приведенному примеру, редукция совершает вычисление.

Определение 1.2.3. Мы говорим, что теория типов (строго) нормализуема, если каждый терм (строго) нормализуем.

Теорема 1.2.4. *Теория типов с* $1, \times, \mathsf{Nat}$ *строго нормализуема.*

Определение 1.2.4. Терм t сильно нормализуем, если любая редукция, в конечном счете, приводит к одной и той же нормальной форме t'.

Напомним некоторые правила редукции

- $\pi_1\langle t_1, t_2\rangle \rightsquigarrow t_1$
- $\pi_2\langle t_1, t_2\rangle \rightsquigarrow t_2$
- $\mathrm{rec}(d_0; x, y\,.\,d_s; 0) \rightsquigarrow d_0$
- $\mathrm{rec}(d_0; x, y\,.\,d_s; Sn) \rightsquigarrow d_s[n/x, \mathrm{rec}(d_0; x, y\,.\,d_s; n)/y]$

Формы π_1, π_2 и rec являются элиминативными формами (elimination forms), поскольку они сводят термы к менее сложным.

Противоположны им конструкторы или интро формы (intro forms), такие как $\langle\rangle, \langle\cdot,\cdot\rangle, 0, S$.

Существуют еще правила конгруэнции на $\rightsquigarrow$ такие как

$$\frac{\Gamma \vdash t \rightsquigarrow t' : \mathsf{Nat}}{\Gamma \vdash St \rightsquigarrow St' : \mathsf{Nat}}$$

Чтобы доказать теорему о нормализируемости теории типов нам пригодится следующая лемма.

Лемма 1.2.5. *Любой подтерм нормального терма нормален.*

Доказательство. Очевидно из возможности редукции внутри терма. □

Лемма 1.2.6. *Любой нормальный замкнутый терм $\vdash t : A$ есть одна из следующих форм:*

1. $\Gamma \vdash \langle\rangle : 1$,
2. $\Gamma \vdash \langle t_1, t_2\rangle : A \times B$, *если* t_1, t_2 *нормальны,*
3. $\Gamma \vdash 0 : \mathsf{Nat}$,
4. $\Gamma \vdash Sn : \mathsf{Nat}$, *если* n *нормален,*

то есть конструкторы с нормальными аргументами. Эта лемма называется канонической леммой (canonicity).

Доказательство. Будем доказывать индукцией по $\Gamma \vdash t : A$. Если t является замкнутым нормальным термом, то он канонической формы. В самом деле, если t — переменная, то нечего доказывать, поскольку t замкнуто. Если же t конструктор, то в силу того, что его аргументы нормальны и замкнуты (по предыдущей лемме) сам конструктор каноничен. И последний случай, когда t элиминативная форма, тогда будем доказывать от противного. Если аргументы t нормальны, то мы приходим к канонической форме из предыдущих случаев. То есть t не нормален; противоречие. □

Таким образом, мы описали все нормальные (замкнутые) термы в нашей теории типов. Из этого можно получить следующее следствие.

Следствие 1. *Любой замкнутый нормальный терм вида $\Gamma \vdash a : \mathsf{Nat}$ представляется в виде $\underline{n}$ для некоторого натурального n.*

1.3 Функциональный тип

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