Теория типов

Человек, который поспорил на 2 торта ♡

1 λ-исчисление

Определение (λ-выражение)**.** λ-выражение — выражение, удовлетворяющее грамматике:

$$\begin{aligned} \Lambda ::= &\lambda x.\Lambda && \text{(абстракция)} \\ &|\, \Lambda\Lambda && \text{(аппликация)} \\ &|\, x \\ &|\, (\Lambda) \end{aligned}$$

(a) аппликация левоассоциативна

(b) абстракция распространяется как можно дальше вправо

(c) смысла в этом нет

Пример. $((\lambda z.(z(yz)))(zx)z) = (\lambda z.z(yz))(zx)z$

Есть понятия связанного и свободного вхождения переменной (аналогично ИП). $\lambda x.A$ связывает все свободные вхождения x в A. Договоримся, что:

(a) Переменные — x, a, b, c.

(b) Термы (части λ-выражения) — X, A, B, C.

(c) Фиксированные переменные обозначаются буквами из начала алфавита, метапеременные — из конца.

Определение (α-эквивалентность)**.** A и B называются α-эквивалентными ($A =_\alpha B$), если выполнено одно из следующих условий:

1. $A \equiv x$ и $B \equiv x$.
2. $A \equiv \lambda x.P$ и $B \equiv \lambda x.Q$. Пусть t — новая переменная, тогда $P_{[x:=t]} =_\alpha Q_{[y:=t]}$.
3. $A \equiv PQ$, $B \equiv RS$, $P =_\alpha R$, $Q =_\alpha S$.

Пример. $\lambda x.\lambda y.xy =_\alpha \lambda y.\lambda x.yx$.

Доказательство.

$$\begin{aligned} \lambda y.ty =_\alpha \lambda x.tx &\implies \lambda x.\lambda y.xy =_\alpha \lambda y.\lambda x.yx \\ tz =_\alpha tz &\implies \lambda y.ty =_\alpha \lambda x.tx \end{aligned}$$

$tz =_\alpha tz$ верно по третьему условию. □

Определение (β-редекс)**.** Терм вида $(\lambda a.A)\,B$ называется β-редексом.

Пример. В выражении $\underbrace{(\lambda f.(\lambda x.\overbrace{f(xx)}^{A_1})\overbrace{(\lambda x.f(xx))}^{B_1})}_{A_2}\underbrace{g}_{B_2}$ два β-редекса.

Определение. Множество λ-термов $\mathbf{\Lambda}$ назовём множеством классов эквивалентности Λ по $(=_\alpha)$.

Определение (β-редукция)**.** $A \to_\beta B$ (состоят в отношении β-редукции), если выполняется одно из условий:

1. $A \equiv PQ$, $B \equiv RS$ и

$$\begin{aligned} &\text{либо } P \to_\beta R \text{ и } Q =_\alpha S \\ &\text{либо } P =_\alpha R \text{ и } Q \to_\beta S \end{aligned}$$

2. $A \equiv \lambda x.P$, $B \equiv \lambda x.Q$, $P \to_\beta Q$ (x из какого-то класса из $\mathbf{\Lambda}$).

3. $A \equiv (\lambda x.P)Q$, $B \equiv P_{[x:=Q]}$, Q свободно для подстановки в P вместо x.

Итак, лулзы. Хотите знать, что такое истина?

$$\begin{aligned} \text{T} &= \lambda x \lambda y.x \\ \text{F} &= \lambda x \lambda y.y \\ \text{Not} &= \lambda a.a\text{FT} \end{aligned}$$

Похоже на тип boolean, не правда ли?

Пример.

$$\text{Not T} = (\lambda a.a\text{FT})\text{T} \to_\beta \text{TFT} = (\lambda x.\lambda y.x)\text{FT} \to_\beta (\lambda y.\text{F})\text{T} \to_\beta \text{F}$$

Можно продолжить:

$$\begin{aligned} \text{And} &= \lambda a.\lambda b.ab\text{F} \\ \text{Or} &= \lambda a.\lambda b.a\text{T}b \end{aligned}$$

Попробуем определить числа:

Определение (Чёрчевский нумерал)**.**

$$\overline{n} = \lambda f.\lambda x.f^n x, \quad \text{где} \quad f^n x = \begin{cases} f\left(f^{n-1}x\right) & ,n > 0 \\ x & ,n = 0 \end{cases}$$

Пример.

$$\overline{3} = \lambda f.\lambda x.f(f(fx))$$

Несложно определить прибавление единицы к такому нумералу:

$$(+1) = \lambda n.\lambda f.\lambda x.f(nfx)$$

Пример.

$$(+1)\overline{1} = (\lambda n.\lambda f.\lambda x.f(nfx))(\lambda f.\lambda x.fx) \to_\beta \lambda f.\lambda x.f((\lambda f.\lambda x.fx)fx) \twoheadrightarrow_\beta \lambda f.\lambda x.f(fx) = \overline{2}$$

Определение (η-эквивалентность)**.**

$$\lambda x.fx =_\eta f$$

Аналог из C++: если **int** f(**int** x), то результат её вычисления равен результату вычисления [ ] (**int** x) { **return** f(x); } .

Арифметические операции:

$$\begin{aligned}
\text{IsZero} &= \lambda n.n(\lambda x.\text{T})\text{F} \\
\text{IsEven} &= \lambda n.n \text{ Not T} \\
\text{Add} &= \lambda a.\lambda b.\lambda f.\lambda x.af(bfx) \\
\text{Mul} &= \lambda a.\lambda b.a(\text{Add } b)\overline{0} \\
\text{Pow} &= \lambda a.\lambda b.b(\text{Mul } a)\overline{1} \\
\text{Pow}^* &= \lambda a.\lambda b.ba
\end{aligned}$$

Для того, чтобы определить (-1), сначала определим "пару":

$$\begin{aligned}
\langle a, b\rangle &= \lambda f.fab \\
\text{First} &= \lambda p.\text{T}p \\
\text{Second} &= \lambda p.\text{F}p
\end{aligned}$$

n раз применим функцию $f(\langle a, b\rangle) = \langle b, b+1\rangle$ и возьмём первый элемент пары:

$$(-1) = \lambda n.\text{First}\left(n\ (\lambda p.\langle(\text{Second } p), (+1)\ (\text{Second } p)\rangle)\ \left\langle\overline{0}, \overline{0}\right\rangle\right)$$

Сокращение записи:

$$\lambda xy.A = \lambda x.\lambda y.A$$

Определение (Нормальная форма)**.**
Терм A — нормальная форма (н.ф.), если в нём нет β-редексов.
Нормальной формой A называется такой B, что $A \twoheadrightarrow_\beta B$, B — н.ф.
$\twoheadrightarrow_\beta$ — транзитивно-рефлексивное замыкание $\to_\beta$.

Утверждение 1. *Существует λ-выражение, не имеющее н.ф.*

$$\begin{aligned}
\Omega &= \omega\omega \\
\omega &= \lambda x.xx
\end{aligned}$$

Определение (Комбинатор)**.** Комбинатор — λ-выражение без свободных переменных.

Комбинатор неподвижной точки:

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

Определение (β-эквивалентность)**.** $A =_\beta B$, если $\exists C : C \twoheadrightarrow_\beta A, C \twoheadrightarrow_\beta B$

Утверждение 2.

$$Yf =_\beta f(Yf)$$

Доказательство. (на лекции не давалось)

$$\begin{aligned} Yf &=_\beta (\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))f \\ &=_\beta (\lambda x.f(xx))(\lambda x.f(xx)) \\ &=_\beta f((\lambda x.f(xx))(\lambda x.f(xx))) \\ &=_\beta f(Yf) \end{aligned}$$

□

Таким образом, с помощью Y-комбинатора можно определять рекурсивные функции.

Пример.

$$\text{Fact} = Y(\lambda fn.\text{IsZero}\ n\ \overline{1}\ (\text{Mul}\ n\ (f\ (-1)\ n)))$$