

## 1 Немного истории.

## 2 Введение в лямбда-исчисление: определения и базовые результаты.

Рассмотрим мотивирующий пример. Когда мы пишем, что «функция отображает аргумент  $x$  в  $M$ », где  $M$  — это метапеременная, в которой лежит тело функции, то мы используем следующую нотацию  $x \mapsto M$ , тогда как запись  $\lambda x.M$  следует читать точно также в содержательном смысле. Расширим наш мотивирующий и не совсем формальный пример, заменив метапеременную  $M$  на более понятное арифметическое выражение :  $x \mapsto x^2 + 6x + 9$  и  $\lambda x.x^2 + 6x + 9$ .

Теперь же перейдем к формальным определениям. Базовое понятие в  $\lambda$ -исчислении — это *предтерм*. Предположим, у нас есть бесконечный алфавит:

$$\Lambda = v_0, v_1, v_2, v_3, \dots \quad (1)$$

*Предтермами* мы будем называть конечные строки над алфавитом  $\Lambda$ , порожденные следующей грамматикой:

$$\Lambda_{term} ::= \Lambda \mid (\Lambda_{term} \Lambda_{term}) \mid \lambda \Lambda. \Lambda_{term} \quad (2)$$

Примеры конечных строк, порожденных заданной грамматикой:

- 1)  $((v_3 v_5) v_8)$ ;
- 2)  $\lambda v_6.v_5 v_6$
- 3)  $\lambda v_0.v_0$
- 4)  $\lambda v_{05091995}.\lambda v_{38}.v_4$

Как мы видим из определения грамматики, предтермы бывают трех видов.

Зададим классификацию предтермов в соответствии с грамматикой:

1) Предтерм первого вида (это просто элементы  $\Lambda$ ) называется *переменной*, которые мы будем обозначать тремя предпоследними буквами латинского алфавита  $x, y, z, \dots$  (возможно с индексами);

2) Предтерм второго вида (записанные два подряд предтерма) называется *аппликацией* (или *применением*), которую мы будем обозначать  $(MN)$ , где  $M$  и  $N$  — это произвольные предтермы, которые впредь будут обозначаться метапеременными  $M, N, O, \dots$  (возможно с индексами);

3) Предтерм третьего вида (знак  $\lambda$  с переменной, точка и предтерм) называется  *$\lambda$ -абстракцией*, которая будет обозначаться как  $\lambda x.M$ , где  $x$  является *связанной переменной*. Если в предтерме встречается переменная  $x$ , которая связана  $\lambda$ -оператором, то такая переменная будет называться *свободной переменной*.

Поясним, что  $\lambda$  — это оператор связывания. Пусть у нас есть некоторый предтерм  $M$ , содержащий свободные вхождения  $x$ . Теперь мы  $\lambda$ -абстрагируемся по  $x$  и получим предтерм третьего вида  $\lambda x.M$ , представляя таким образом выражение, зависящее от значения параметра  $x$ .

**Важное терминологическое соглашение:** любой предтерм, удовлетворяющий тому или иному виду, мы будем называть  $\lambda$ -термами.

$\lambda$ -исчисление же начинается тогда, когда мы вводим систему правил преобразования термов:

1)  $\alpha$ -конверсия — правило переименования связанных переменных:  $\lambda x.M \rightarrow_\beta \lambda y.M[x := y]$ . Важно следить, чтобы переименование не вызывало коллизий, например:  $\lambda x.xy \rightarrow_\beta \lambda y.yy[x := y]$ . Видно, что до переименования в нашем терме была одна связанная переменная, которая в теле функции применяется к свободному параметру  $y$ . Далее, мы заменили связанную переменную на  $y$  и получили на выходе терм, который внешне отличен от исходного, поскольку после переименования наша связанная переменная в терме в теле функции уже применяется сама к себе.

Над  $\lambda$ -термами мы можем ввести отношение  $\alpha$ -эквивалентности (что пишется как  $M \equiv_\alpha N$ ), которое, как и любое другое отношение эквивалентности, рефлексивно, симметрично и транзитивно:

- i)  $M \equiv_\alpha M$
- ii)  $M \equiv_\alpha N \Rightarrow N \equiv_\alpha M$
- iii)  $M \equiv_\alpha N, N \equiv_\alpha P \Rightarrow M \equiv_\alpha P$

Действительно, во-первых, любой лямбда-терм тривиально эквивалентен сам себе при тождественном переименовании связанных переменных, никак не меняющем исходные имена. Во-вторых, если мы переименовали связанные переменные, то мы вполне имеем право сделать обратное переименование, восстанавливающее исходный терм, что и дает нам симметричное свойство отношения  $\alpha$ -эквивалентности. И, в-третьих, если мы переименовали связанные переменные, а затем переименовали связанные переменные в результате первого переименования, то мы вправе рассматривать такую цепочку переименований как переименование связанных переменных в исходном терме на имена, что связанные переменные имеют в конце данной цепочки. Таким образом, транзитивность у нас также проходит.

Поскольку,  $\alpha$ -эквивалентность — это отношение эквивалентности, то мы имеем право разбить  $\lambda$ -термы на классы эквивалентностей следующего вида:

$$[M]_\alpha = \{ N \in \Lambda_{term} \mid M \equiv_\alpha N \} \quad (3)$$

Таким образом, фактор-множество по данному отношению, будет состоять из указанных выше классов эквивалентностей, и данное фактор-множество как раз и будет совокупностью  $\lambda$ -термов, то есть мы рассматриваем здесь и далее мы рассматриваем  $\lambda$ -термы с точностью до  $\alpha$ -эквивалентности.

2)  $\beta$ -редукция представляет собой некоторое обобщение правила вычисления. Формально правило выглядит так:

$$(\lambda x.M)N \rightarrow_\beta M[x := N] \quad (4)$$

Мы берем лямбда-терм и применяем к нему терм  $N$ , а результатом данного применения является терм  $M$ , в котором все вхождения  $x$  заменяются на  $N$ .

- 3 Комбинаторная логика и ее связь с лямбда-исчислением.
- 4 Простое типизированное лямбда-исчисление: типизация по Карри и по Черчу.
- 5 Типизированные комбинаторы.
- 6 Практическая реализация лямбда-исчисления, комбинаторной логики и теории типов.