Введение в лямбда-исчисление

Белкин Дмитрий, гр. 4362
Бертыш Вадим, гр.4374

2-е издание

Под редакцией Кравчука Ивана,

группа 5302

Оглавление

Предисловие редактора....3
Введение....4
Формализация....5
Свободные и связанные переменные....6
Комбинаторы....7
Функции нескольких переменных. Каррирование....8
Подстановки....9
Переименование связанных переменных....10
β-редукция....11
Редексы....12
Нормальная форма....13
Теорема о неподвижной точке....14
Нормальный и аппликативный порядок редукции....15
Редукционные графы....16
Теорема Черча-Россера....17
Следствия из теоремы Черча - Россера....18
Заключение....19

Предисловие редактора

В данном издании были добавлены несколько новых тем, в частности подробнее рассмотрены функции нескольких переменных, а также описаны редукционные графы. В этой версии я решил исключить из изучения тему “чистое лямбда-исчисление как язык программирования”, поскольку это более углубленное изучение темы, что не отвечает наименованию пособия

Кравчук И.В., группа 5302

Введение

В 1928 году Давид Гильберт сформулировал так называемую проблему разрешимости, суть которой примерно выражается в следующем. Было необходимо сформулировать такой алгоритм, который на основании поданных на вход формального языка и утверждения, записанного этим языком, за конечное число шагов приходил к ответу, является ли это утверждение ложным, или же истинным. В 1936 году два математика, Алонсо Черч и Алан Тьюринг независимо друг от друга опубликовали работы, в которых заявлялось, что для арифметических утверждений такой алгоритм составить невозможно, а потому и в более общих случаях эта проблема неразрешима. Впоследствии это утверждение стало известно, как теорема Черча- Тьюринга и породило как минимум два средства формализации алгоритмов и понятия вычислимости. В случае с Аланом Тьюрингом это была небезызвестная машина Тьюринга. В случае же с Алонсо Черчем это было лямбда исчисление.

Основными объектами лямбда исчисления являются функции. В современной математике под функцией f чаще всего понимают некоторое соответствие элементам области определения X элементов области значения Y, то есть для каждой $f : X \to Y$ верно, что $f \subseteq X \times Y$. Тем не менее, лямбда исчисление – это теория функций как формул. Это система, позволяющая работать с функциями как с выражениями. Для примера вспомним другой хорошо известный язык выражений – арифметику. Арифметические выражения составляются из переменных $(x, y, z, \dots)$, чисел $(1, 2, 3, \dots)$, и операторов $(+, -, \times, etc)$. Выражение вроде $x + y$ представляет из себя *результат* сложения. Главное преимущество этого языка в том, что выражения можно вкладывать одно в другое, не указывая отдельно промежуточные результаты. Так, например, мы пишем

$$A = (x + y) \times z^2$$

а не

“Пусть $w = x + y$, тогда пусть $u = z^2$, тогда пусть $A = w \times u$”

Использование второго варианта было бы утомительным и неудобным. Лямбда исчисление расширяет идею языка выражений на функции. Так вместо выражения вида

“Пусть f функция $x \longmapsto x^2$. Тогда будем считать, что $A = f(5)$”

в лямбда исчислении будет следующее выражение

$$A = (\lambda x.\, x^2)(5)$$

Запись $(\lambda x.\, x^2)$ выражает функцию, которая ставит в соответствие прообразу x образ x^2. Как и в арифметике, скобки используются, чтобы группировать термы.

Формализация

Пусть $V = \{x, y, z \dots\}$ – множество переменных. Тогда множество лямбда-термов Λ индуктивно строится из множества V следующим образом:

$x \in V \Rightarrow x \in \Lambda$, (переменная является лямбда-термом)

$M, N \in \Lambda \Rightarrow M\ N \in \Lambda$ (аппликация)

$M \in \Lambda, v \in V \Rightarrow \lambda v.M \in \Lambda$ (абстракция)

В дальнейшем условимся, что

- Прописными буквами обозначаются произвольные переменные
- Заглавными буквами обозначаются произвольные лямбда-термы

В лямбда исчислении существуют два способа построения выражений:

- Аппликация (применение)

Аппликация записывается следующим образом

$$M\ N$$

С точки зрения программиста эта запись означает “алгоритм M применяется к N ”, то есть N является входными данными для M.

Различия между алгоритмами и данными отсутствуют, в частности возможно само-применение.

- Абстракция

Пусть $M = M[x]$ – выражение, (возможно) содержащее x.

Тогда абстракция

$$\lambda x.M$$

обозначает функцию

$$x \mapsto M[x]$$

то есть каждому значению x сопоставляется значение $M[x]$.

Если x отсутствует в выражении $M[x]$, то $\lambda x.M$ – константная функция со значением M.

Приняты следующие соглашения:

- Внешние скобки опускаются
- Аппликация ассоциативна влево

$$F\ X\ Y\ Z = \big((FX)\ Y\big)\ Z$$

- Абстракция ассоциативна вправо

$$\lambda xyz.M = \lambda x.(\lambda y.\big(\lambda z.(M)\big))$$

- Тело абстракции простирается вправо насколько это возможно

$$\lambda x.MNK = \lambda x.(MNK)$$

Свободные и связанные переменные

Связанные переменные – локальные переменные для некоторого выражения

Свободные переменные – параметры выражения.

Связывание переменных происходит при абстрагировании.

Множество $FV(T)$ свободных переменных в терме T:

$$FV(x) = \{x\}$$

$$FV(M\ N) = FV(M) \cup FV(N)$$

$$FV(\lambda x.M) = FV(M) \backslash \{x\}$$

Множество $BV(T)$ связанных переменных в терме T:

$$BV(x) = \emptyset$$

$$BV(M\ N) = BV(M) \cup BV(N)$$

$$BV(\lambda x.M) = BV(M) \cup \{x\}$$

Комбинаторы

Комбинатор – замкнутый терм. M называется замкнутым термом, если $FV(M) = \emptyset$. Множество замкнутых лямбда-термов обозначается Λ°

Примеры классических комбинаторов:

$$I = \lambda x.x$$

$$\omega = \lambda x.(x\ x)$$

$$\Omega = \omega\omega = \lambda x.(x\ x)\ (\lambda x.(x\ x))$$

$$K = \lambda xy.x$$

$$K^* = \lambda xy.y$$

$$S = \lambda fgx.fx(g\ x)$$

$$B = \lambda fgx.f(g\ x)$$

Функции нескольких переменных. Каррирование

Функция нескольких переменных может быть записана как конечная последовательность функций одной переменной.

Пусть $\varphi(x, y)$ – терм, содержащий свободные x и y. Путем последовательных абстракций, введем:

$$\Phi_x = \lambda y.\varphi(x, y)$$

$$\Phi = \lambda x.\Phi_x = \lambda x.(\lambda y.\varphi(x, y)) = \lambda xy.\varphi(x, y)$$

Тогда применение этого терма к произвольным X и Y может быть выполнено последовательно.

$$\Phi XY = (\Phi X)Y = \Phi_x Y = \lambda y.\varphi(X, y)\ Y = \varphi(X, Y)$$

Каррирование – переход от функции нескольких переменных к функции, принимающей аргументы “по одному”.

Подстановки

$M[x \coloneqq N]$ называется подстановкой N вместо свободных вхождений x в M.

Основные правила подстановки:

$$x[x \coloneqq N] = N$$

$$y[x \coloneqq N] = y$$

$$(PQ)[x \coloneqq N] = (P[x \coloneqq N])(Q[x \coloneqq N])$$

$$(\lambda y.P)[x \coloneqq N] = \lambda y.(P[x \coloneqq N]), y \notin FV(N)$$

$$(\lambda x.P)[x \coloneqq N] = \lambda x.P$$

Подразумевается, что $x \not\equiv y$

Соглашение Барендрегта:

Имена связанных переменных будем выбирать так, чтобы они отличались от имен свободных переменных

$$\lambda y.xy[x \coloneqq y] =_\alpha \lambda y'.xy'[x \coloneqq y] = \lambda y'.yy' \ [1]$$

[1] Альфа-преобразование описано на следующей странице

Переименование связанных переменных

Термы часто определяются с точностью до имен связанных переменных. Введем отношение эквивалентности на множестве лямбда-термов

$$\forall M \in \Lambda: M =_\alpha M$$

$$\lambda x.M[x] =_\alpha \lambda y.M[x \coloneqq y], y \notin FV(M)$$

Некоторые аксиомы α-эквивалентности:

$$M =_\alpha M$$

$$M =_\alpha N \Rightarrow N =_\alpha M$$

$$M =_\alpha N, N =_\alpha L \Rightarrow M =_\alpha L$$

$$M =_\alpha M' \Rightarrow MZ =_\alpha M'Z$$

$$M =_\alpha M' \Rightarrow ZM =_\alpha ZM'$$

$$M =_\alpha M' \Rightarrow \lambda x.M =_\alpha \lambda x.M'$$

Процесс замены имени переменной называется α-конверсией и определяется следующим образом:

$$\lambda x.M[x] \rightarrow_\alpha \lambda y.M[x \coloneqq y], y \notin FV(M)$$

β-редукция

Для любых $M, N \in \Lambda$ справедливо соотношение

$$(\lambda x.M)N = M[x \coloneqq N] \qquad (\beta)$$

Исходя из этого соотношения сформулируем несколько следствий:

$$(\lambda x_1 x_2 x_3 \dots x_n.M)X_1X_2X_3 \dots X_n = M[x_1 \coloneqq X_1] \dots [x_n \coloneqq X_n]$$

$$M[x \coloneqq N_1][y \coloneqq N_2] = M[y \coloneqq N_2][x \coloneqq N_1[y \coloneqq N_2]$$

Однократное применение соотношения (β) обозначается $\rightarrow_\beta$. Применение (β) ноль или более раз называется β-редукцией и обозначается как $\twoheadrightarrow_\beta$

$$M \rightarrow_\beta N \Rightarrow ZM \rightarrow_\beta ZN$$

$$M \rightarrow_\beta N \Rightarrow MZ \rightarrow_\beta NZ$$

$$M \rightarrow_\beta N \Rightarrow \lambda x.M \rightarrow_\beta \lambda x.N$$

Введем отношение β-эквивалентности, которое обозначим $=_\beta$.Оно определяется как

$$\forall M, N \in \Lambda: M \twoheadrightarrow_\beta N \Rightarrow M =_\beta N$$

$$M =_\beta N \Rightarrow N =_\beta M$$

$$M =_\beta N, N =_\beta L \Rightarrow M =_\beta L$$

Пример

$$KI = (\lambda xy.x)(\lambda z.z) \rightarrow_\beta \lambda yz.z$$

$$IIK^* = (\lambda x.x)(\lambda x.x)(\lambda yz.z) \twoheadrightarrow_\beta \lambda yz.z$$

KI редуцируется в K^*

IIK^*редуцируется в K^*

Следовательно, $KI =_\beta IIK^*$

Редексы

Редекс – терм вида

$$(\lambda x.M)N$$

Иными словами, редекс можно описать как “применение абстракции к некоторому терму”.

Процесс раскрытия редекса и называется редукцией.

Любое вычисление представляет собой некоторое количество шагов β-редукции. Вычисления заканчиваются, когда в терме не остается *редексов.* Однако, количество редексов может не изменяться при раскрытии или даже увеличиваться.

Пример(абстрактный)

$$(\lambda x.\_x\_x\_x)N$$

где N – некоторое количество редексов

Нормальная форма

- Терм находится в β-нормальной форме, если он не имеет подтермов, являющихся редексами.
- Терм M имеет нормальную форму, если для некоторого N выполняется $M =_\beta N$ и N находится в нормальной форме
- Не все термы имеют нормальную форму

Пример терма, не имеющего нормальной формы

$$\Omega = \omega\omega = \lambda x.(x\ x)\left(\lambda x.(x\ x)\right) \to_\beta \lambda x.(x\ x)\left(\lambda x.(x\ x)\right) \to_\beta \ldots$$

Но, возможно, существует $N \in \Lambda$, такой что $\Omega =_\beta N$

$$M \twoheadrightarrow_\beta \Omega$$

$$M \twoheadrightarrow_\beta N$$

$$\Omega =_\beta M, M =_\beta N \Rightarrow \Omega =_\beta N$$

Теорема о неподвижной точке

Теорема:

Для любого λ-терма F существует неподвижная точка:

$$\forall F \in \Lambda\ \exists X \in \Lambda\colon FX = X$$

Доказательство

Введем $W = \lambda x.F(xx)$ и $X = WW$.

Тогда $X = WW = \lambda x.F(xx)W \rightarrow_\beta F(WW) = F(X)$

Теорема доказана

Теорема:

Существует комбинатор неподвижной точки Y, такой что

$$\forall F \in \Lambda\colon YF = F(YF)$$

Доказательство

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

Имеем $YF \rightarrow_\beta \lambda x.F(xx)\lambda x.F(xx) \rightarrow_\beta F\big(\lambda x.F(xx)\lambda x.F(xx)\big) = F(YF)$

Теорема доказана

Нормальный и аппликативный порядок редукции

Как уже указывалось ранее, не все термы имеют нормальную форму. Рассмотрим следующий терм:

$$N = (\lambda xy.y)((\lambda x.xx)(\lambda x.xx))M$$

В этом терме два редекса, соответственно редукцию можно произвести двумя способами

$$N \rightarrow_\beta M$$

$$N \rightarrow_\beta (\lambda xy.y)((\lambda x.xx)(\lambda x.xx))M$$

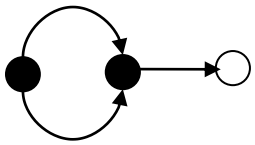
В первом случае сначала редуцировали внешний редекс, во втором – внутренний (который оказался уже знакомым нам комбинатором Ω, редуцирующимся в самого себя). Таким образом можно заметить, что привести N к нормальной форме, редуцируя внутренний редекс, невозможно. Это значит, что, если существует нормальная форма, к ней не обязательно ведет любой порядок редукции. То, в каком порядке производится редукция, определяет стратегию редукции. Выделяют несколько различных стратегий, мы же затронем две из них: нормальную (она соответствует ленивым вычислениям в языках программирования) и аппликативную (она соответствует энергичным вычислениям).

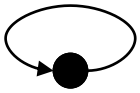
При аппликативной стратегии редукция термов производится справа налево, изнутри наружу. Это соответствует вычислению значений аргументов перед вызовом функции.

Нормальная стратегия предполагает редукцию слева направо, снаружи внутрь. Это соответствует отказу от вычисления значений аргументов перед вызовом функции.

Редукционные графы

Редукционный граф терма $M \in \Lambda(G_\beta(M))$ – это ориентированный мультиграф с вершинами в $\{N | M \twoheadrightarrow_\beta N\}$ и дугами $\rightarrow_\beta$

$G_\beta\big(I(Ix)\big) =$ 

$G_\beta(\Omega) =$ 

● – промежуточное состояние

○ – нормальная форма

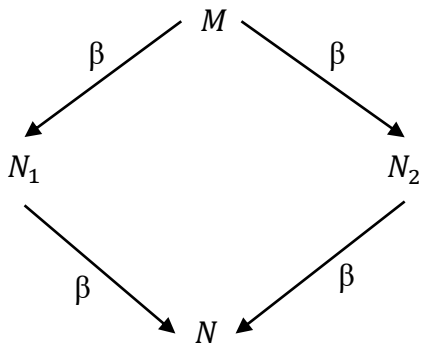
Теорема Черча-Россера

Мы подошли к центральному моменту в лямбда-исчислении. Часть вещей, описанных ранее, которые принимались как должное можно обосновать, используя приведенную ниже теорему. В какой-то мере эта теорема обосновывает все допущения о корректности исчисления.

Теорема:

$$\forall M, N_1, N_2 : M \twoheadrightarrow_\beta N_1, M \twoheadrightarrow_\beta N_2 \Longrightarrow \exists N : N_1 \twoheadrightarrow_\beta N, N_2 \twoheadrightarrow_\beta N$$

Иначе говоря, для редукции выполняется свойство ромба:



Следствия из теоремы Черча - Россера

1. Единственность нормальной формы

Пусть N_1 и N_2 – нормальные формы терма M. Тогда $N_1 = N_2$.

Доказательство

По теореме Черча-Россера

$$\exists M: N_1 \twoheadrightarrow_\beta M, N_2 \twoheadrightarrow_\beta M$$

N_1 и N_2. не имеют редексов по определению нормальной формы, следовательно, $N_1 = M$ и $N_2 = M$, следовательно, $N_1 = N_2$.

Теорема доказана

2. Если $M =_\beta N$ и один из этих термов имеет нормальную форму Z, второй также нормализуем, причем его нормальная форма равна Z

Доказательство

Не умаляя общности, пусть Z – нормальная форма M.

$$M \twoheadrightarrow_\beta Z \Rightarrow M =_\beta Z \Rightarrow N =_\beta Z \Rightarrow N \twoheadrightarrow_\beta Z$$

Последнее следствие допустимо ввиду того что Z не имеет редексов, поскольку является нормальной формой.

Теорема доказана

Заключение

В чистом лямбда-исчислении есть возможность построить такие термы, которые могут обеспечить функционал, аналогичный некоторым базовым возможностям языков программирования. В данном учебном пособии эта тема не рассматривалась, поскольку его целью является знакомство с основами лямбда-исчисления, его основными правилами и теоремами. Для более глубокого ознакомления с данной темой рекомендуется прочесть соответствующую специализированную литературу.[2]

В силу тех же причин здесь не были рассмотрены типизированные версии лямбда-исчисления. Теории типов и типизации лямбда исчисления можно посвятить полноценную отдельную книгу, объем информации по этой теме слишком велик, чтобы помещать его в учебное пособие такого формата как это.

Для практического закрепления изученного материала предлагается сборник практических задач, специально составленный по темам этого курса.

[2] В частности, рекомендуется к прочтению книга Хэнка Барендрегта “Лямбда-исчисление, его синтаксис и семантика”

Список литературы

[1] Peter Selinger, Lecture Notes on the Lambda Calculus, Department of Mathematics and Statistics, Dalhousie University, Halifax, Canada.

[2] Henk Barendregt, Erik Barendsen, Introduction to Lambda Calculus, Revised edition, December 1998, March 2000.

[3] Ra´ul Rojas, A Tutorial Introduction to the Lambda Calculus FU Berlin, WS-97/98

[4] Barendregt, H.P. (1984) The Lambda Calculus: Its Syntax and SemanticsStudies in Logic 103, second, revised edition, North-Holland, Amsterdam.