1 Функции

1.1 Понятие "функция"

Как было показано в предыдущих лекциях, в теории множеств понятие "функция"не является фундаментальным, а скорее происходит из фундаментального понятия "множество". Лямбда-исчисление предоставляет другой подход к определению "функции при котором она является первичным, фундаментальным понятием. Также стоит отметить две важные вещи:

- λ-исчисление - это пример *формального* исчисления, т.е. один из способов работы с функциями механически (например, на компьютере)

- λ-исчисление можно считать обобщённой *моделью вычислений*, т.е. формальной моделью того, чем в действительности являются вычисления.

1.2 Функции высшего порядка, каррирование

Некоторые функции могут принимать другие функции в качестве аргументов и возвращать функции в качестве результата. Эти функции называются **функциями высшего порядка**. В *бестиповом* λ-исчислении можно сказать, что (почти) все функции являются функциями высшего порядка, потому что можно применять любую функцию к любому аргументу и, в частности, к функциям. Единственное возможное исключение - функции от нуля аргументов (константы), которые могут представлять скалярные значения..

Каррирование

Использование функций высшего порядка позволяет упростить общую теорию, рассматривая только одноместные (функции от одного аргумента) или функции без аргументов. А именно любая двухместная функция $f(x, y)$ может быть представлена как одноместная функция высшего порядка f', принимающая аргумент x и возвращающая функцию которая, в свою очередь, принимает аргумент y:

$$f'(x)(y) = f(x, y)$$

Эта техника называется **каррирование** и может применяться к функциям с произвольным количеством аргументов.

1.3 Аппликация

Рассмотрим "функции"с неформальной точки зрения, на примерах классических функций, таких как $sin(x)$ или x^2. Далее будем рассматривать только *одноместные* функции (т.е. функции от одного аргумента). Способ представления функций с большим количеством аргументов будет рассмотрен позже..

Аппликация

Дана *функция* (что бы это ни значило), её главное свойство заключается в том, что, будучи применённой к аргументу, она возвращает определённое значение. Следовательно, оператор **аппликации** принимает функцию f, её аргумент a и выполняет *применение* f к a. В стандартной математической нотации это записывается как $f(a)$.

В λ-исчислении применение функции f к Аргументу a обозначается следующим образом:

$$(f\ a)$$

Отметим, что мы не используем скобки, чтобы отделять символ функции от символа аргумента, потому что мы интерпретируем первый символ как функцию, а все остальные как аргументы.

1.4 Композиция аппликаций

Из классической математики известно, что если есть две одноместные функции: x^2 и $x + 1$, то из них можно составить две сложные функции: $x^2 + 1$ и $(x + 1)^2$. Как правило, это делается при помощи оператора *композиции* функций.

Композиция

Даны две функции f и g, а также аргумент, можно составить последовательную аппликацию f к результату аппликации g к a:

$$(f\ (g\ a))$$

Отметим, что можно было выполнять аппликации в другом порядке:

$$((f\ g)\ a)$$

и это даст другой результат: $(f\ g)$ возвращает *функцию*, которая затем применяется к a.

1.5 Функции высшего порядка

Функции высшего порядка

Рассмотрим сложную аппликацию $((f\ g)\ a)$: здесь результат $(f\ g)$ применяется к a, так что это должна быть функция. Пример композиции вроде $((f\ g)\ a)$ делает обязательным рассмотрение так называемых **функций высшего порядка**, т.е. функций, результатом выполнения которых является функция или какой-либо аргумент является функцией.

Соглашение

Далее будет использоваться **левоассоциативность** для итеративного оператора аппликации:

$$(f_1\ f_2\ \ldots\ f_n) \rightleftharpoons (\ldots((f_1\ f_2)\ f_3)\ldots\ f_n)$$

Отметим, что здесь все функции $f_1, f_2, \ldots, f_{n-1}$ являются функциями высшего порядка.

1.6 Абстракция

При помощи оператора аппликации мы можем перейти от функции и ее аргумента к возвращаемому значению. Возможно ли создать противоположный оператор: который принимает выражение, определяющее значение, полученное из аргумента, и преобразует это выражение в соответствующую функцию? Ответ "да и такой оператор называется абстракцией.

Абстракция

Для любой переменной x и выражения e, определяющего, как его значение рассчитывается из x, следующая запись

$$\lambda x.e$$

называется **абстракцией** образованной из x и e.

пример

Рассмотрим запись $x^2 + 1$. Строго говоря, под этой записью мы понимаем *значение* соответствующей функции для данного x, но не самой *функции*. Но если написать $\lambda x.(x^2 + 1)$, оно станет функцией, а точнее λ абстракцией для функции "квадрат аргумента плюс 1".

1.7 Последовательная абстракция

Как и оператор аппликации, абстракция может быть последовательно выполнена любое количество раз.

Итеративная абстракция

Даны переменные $x_1, \ldots, x_n$ и выражение e, можно составить **итеративную абстракцию** следующим образом

$$\lambda x_1 \cdots x_n.e \rightleftharpoons \lambda x_1.(\lambda x_2.(\ldots(\lambda x_n.e)\ldots))$$

Заметим, что в отличие от итеративной аппликации, здесь **абстракция правоассоциативна**.

2 λ-термы

2.1 Понятие λ-терма

Зафиксируем три множества: множество X переменных, C - множество констант (константных функций) и явно определенный алфавит λ исчисления $\mathcal{A}_\lambda = \{(,), \lambda, .\}$

Определение

λ-терм, составленный из переменных X и констант C - это слово в алфавите $\mathcal{A}_\lambda \cup X \cup C$, определяемое по индукции:

- любая переменная $x \in X$ и любая константа $c \in C$ являются λ-термом.

- для любых λ-термов p и q запись

$$(p\ q)$$

является λ-термом и называется **аппликацией** p к q.

- для любой переменной $x \in X$ и λ-терма f, запись

$$(\lambda x.f)$$

является λ-термом и называется **абстракцией** f от x.

Соглашение

- Обычно будем предполагать, что множество переменных X фиксировано.

- Множество всех λ термов над множеством констант C обозначается как $\Lambda(C)$.

- Если $C = \emptyset$, будем писать просто Λ для обозначения всех λ-термов без констант.

- Также обычно будем опускать крайние скобки в λ-термах и использовать итеративную абстракцию и аппликацию, чтобы сделать термы короче и понятнее:

$$\lambda xy.(f\ x\ y) = (\lambda x.(\lambda y((f\ x)\ y)))$$

2.2 Связанные и свободные переменные в λ-термах

Определение

Для любого λ-терма t можно составить два множества:

- $V(t)$ - множество **всех переменных**:

- $FV(t)$ - множество **свободных переменных**:

Эти множества определяются следующим образом:

- если $t = x \in X$ - переменная, то $FV(t) = V(t) = \{x\}$

- если $t = c \in C$ - константа, то $FV(t) = V(t) = \emptyset$
- если $t = (p\ q)$ - аппликация, то

$$FV(t) = FV(p) \cup FV(q), V(t) = V(p) \cup V(q)$$

- если $t = \lambda x.p$ - абстракция, то

$$FV(t) = FV(p) \setminus \{x\}, V(t) = V(p)$$

Переменные из $V(t) \setminus FV(t)$ называются **связанными**.

2.3 Замкнутые λ-термы и комбинаторы

Определение

λ-терм t называется **замкнутым**, тогда и только тогда, когда $FV(t) = \emptyset$. Замкнутый λ-терм называется **комбинатором**, тогда и только тогда, когда он не содержит свободных переменных и констант (только связанные переменные).

Примеры комбинаторов

- $I = \lambda x.x$
- $K = \lambda x\ y.x$
- $S = \lambda x\ y\ z.x\ z\ (y\ z)$
- $\lambda f\ x\ y.f\ y\ x$

Примеры незамкнутых термов

- $\lambda x\ y.f\ y\ x$
- $g \lambda x\ y.y\ x$

2.4 Подстановки (неформально)

Идея того, что некоторая переменная в выражении может быть заменена другим выражением, является интуитивной. Действительно, возьмем выражение

$$e = x^2 + y^2$$

оно содержит две переменные: x и y. Следовательно, их можно заменить другими выражениями, скажем $\sin(u)$ вместо x и $\cos(w)$ вместо y. Тогда мы получим новое выражение:

$$e' = \sin^2(u) + \cos^2(w)$$

Если, в свою очередь, заменить обе переменные u и w на z, мы получим:

$$e'' = \sin^2(z) + \cos^2(z)$$

ПРИМЕЧАНИЕ: несмотря на то, что в общепринятом смысле $\sin^2(z) + \cos^2(z) = 1$, мы не можем заменить e'' на 1, потому что подстановка является **чисто синтаксической** операцией, и с её помощью невозможно семантически сокращать выражения.

2.5 Подстановки (формально)

Определение

Подстановка - это отображение из некоторого множества переменных $\{x_1, \ldots, x_n\}$ в множество всех λ-термов $\Lambda(C)$. Обычно это обозначается следующим образом:

$$[x_1 = t_1, \ldots, x_n = t_n]$$

или в виде отображения $\theta = \{(x_i, t_i) | i \leq n\}$.

Примеры

Из предыдущего примера можно рассмотреть две подстановки:

- $\theta_1 = [x = \sin(u), y = \cos(w)]$
- $\theta_2 = [u = z, v = z]$

2.6 Применение подстановок

Определение

Для любой подстановки θ и λ-терма t можно определить **применение** $t\theta$ подстановки θ к терму t индуктивно построением t:

- если $t = c \in C$ - константа, то $t\theta = t$
- если $t = x \in V$ - переменная и $x \in dom(\theta)$ то $t\theta = \theta(x)$
- если $t = x \in V$ - переменная и $x \notin dom(\theta)$ то $t\theta = x$
- если $t = (f\ g)$ - аппликация, то $t\theta = (f\theta\ g\theta)$
- если $t = \lambda x.f$ - абстракция и $x \notin dom(\theta)$ то $t\theta = \lambda x.(f\theta)$
- если $t = \lambda x.f$ - абстракция и $x \in dom(\theta)$ то $t\theta = t$

Примеры

- $(x^2 + y^2)[x = \sin(u), y = \cos(w)] = \sin^2(u) + \cos^2(w)$
- $(\sin^2(u) + \cos^2(w))[u = z, w = z] = \sin^2(z) + \cos^2(z)$

3 Редукции

3.1 Правила переписывания

Выше мы определили, что именно является основными объектами λ-исчисления, а точнее λ-термов, теперь можно определить основные операции над этими объектами. Обычно эти операции называют **редукцией** и преобразованием (переписыванием) λ-термов в другие λ-термы по строго определённым **правилам переписывания**. Эти правила переписывания (редукции) не изменяют значение λ-терма, но преобразуют его синтаксическое представление в некоторую другую форму.

Процесс последовательного переписывания исходного λ-терма имеет важное значение в λ-исчислении и может рассматриваться как общая модель вычислений. Идея в том, что редукции должны приводить терм к некоторому представлению, дальнейшее преобразование которого невозможно, и этот последний терм (называемый нормальной формой) представляет результат вычисления. Удивительно, но эта простая модель может представлять вычисления произвольной сложности.

3.2 α-редукция

Определение

α-редукция правило переписывания:

$$\lambda x.t \Rightarrow_\alpha \lambda y.(t[x = y])$$

может применяться, когда $y \notin FV(t)$ и подстановка $t[x = y]$ не нарушает смысла t, т.е. некоторое свободное вхождение x не должно становиться связанным после подстановки y вместо x. Далее формализуем это условие и скажем, что y свободно относительно x в t.

В действительности α-редукция - это не что иное, как переименование связанных переменных. Действительно, когда мы связываем переменную, она исчезает из внешней области видимости, и поэтому ее можно безопасно переименовать в любую другую переменную, которая не конфликтует с какой-либо переменной в текущей области видимости.

3.3 α-эквивалентность

Предложение

Если для двух λ-термов p и q верно, что $p \Rightarrow_\alpha q$ то $q \Rightarrow_\alpha p$

Доказательство

Для доказательства предложения достаточно заметить, что если при $p \Rightarrow_\alpha q$ переименовать некоторую переменную x в y, то обратная редукция может быть осуществлена переименованием y в x.

Следствие

Отношение $\Rightarrow_\alpha$ является эквивалентностью на $\Lambda(C)$.

Определение

Два λ-терма p и q называются α-**эквивалентными**, тогда и только тогда, когда

$$p \Rightarrow_\alpha q$$

3.4 β-редукция

Определение

β-**редукция** правило переписывания:

$$(\lambda x.t)s \Rightarrow_\beta t[x = s]$$

может применяться когда подстановка $t[x = s]$ не создаёт конфликта имен переменных в t, т.е. когда s свободно относительно x в t.

В действительности β-редукция - это элементарный шаг вычисления, при котором все вхождения переменной x просто заменяются на s внутри t, как только выражение $(\lambda x.t)s$ встречается в переписываемом терме. Терм вида $(\lambda x.t)s$ называется β-**редексом**, а результат редукции $t[x = s]$ называется β-**сокращением**.

3.5 η-редукция

Определение

η-**редукция** правило переписывания:

$$\lambda x.(f\ x) \Rightarrow_\eta f$$

может применяться, когда $x \notin FV(f)$.

В действительности η-редукция - это еще один элементарный шаг вычисления, при котором абстракция сокращается, когда в ней нет необходимости. Действительно, функция $\lambda x. \sin(x)$ имеет то же значение, что и функция sin. Терм вида $\lambda x.(f\ x)$ называется η-**редексом**, а результат редукции f называется η-**сокращением**.

3.6 Отношение переписывания

Определение

Прежде всего, введем отношение переписывания $\Rightarrow_{all}$ как объединение:

$$(\Rightarrow_{\alpha}) \cup (\Rightarrow_{\beta}) \cup (\Rightarrow_{\eta})$$

Отношение $\Rightarrow_{all}$ означает *"переписать по какой-либо редукции"*.

Определим отношение $\Rightarrow_1$ на множестве всех λ-термов $\Lambda(C)$. Во-первых, $\Rightarrow_{all} \subseteq \Rightarrow_1$, т.е. так как $t_1 \Rightarrow_{all} t_2$, то $t_1 \Rightarrow_1 t_2$. Чтобы распространить определение на другие термы, используем индукцию по λ-терму:

- если $f_1 \Rightarrow_1 f_2$ и $g_1 \Rightarrow_1 g_2$, то $(f_1\ g_1) \Rightarrow_1 (f_2\ g_2)$
- если $f \Rightarrow_1 g$ то $\lambda x.f \Rightarrow_1 \lambda x.g$

Отношение $\Rightarrow_1$ означает *"переписать некоторый подтерм по какой-либо редукции"*

Наконец, определим отношение $\Rightarrow$ как транзитивное замыкание $\Rightarrow_1$. Значение $\Rightarrow$: *"Существует последовательность редукций подтермов по какой-либо редукции"*.