Основы лямбда-исчисления

1. Лямбда-исчисление: ключевые понятия

Лямбда-исчисление, разработанное А. Черчем для преодоления некоторых проблем в математической логике, фактически служит теоретическим базисом функционального программирования. Это исчисление формализует понятие вычислимости на базе математического понятия функции, причем мощность этого исчисления такова, что все функциональные программы можно преобразовать в эквивалентные вычислимые лямбда-выражения. Кроме того, можно описать все базовые свойства функционального языка, опираясь на понятия лямбда-исчисления. В то же время лямбда-исчисление элементарно с точки зрения синтаксиса его объектов (лямбда-выражений) и производимых над ними вычислительных действий.

Над лямбда-выражениями могут быть выполнены всего две операции: функциональная абстракция и функциональная аппликация.

Функциональная абстракция служит для превращения некоторого выражения в функцию, устанавливая ее аргумент (что соответствует понятию формального параметра в языках программирования). К примеру, выражение x+y еще не является функцией, однако запись f(x)=x+y уже фиксирует аргумент функции (и ее имя), тем самым определяя функцию.

Функциональная аппликация - есть по сути применение функции к ее аргументу.

Лямбда-исчисление оперирует только функциями от одной переменной, причем функции не имеют имени. Тем самым это исчисление анонимных (безымянных) функций, которое включает:

- специальную нотацию (лямбда-нотацию), устанавливающую синтаксические правила записи функций и выражений;
- правила преобразования (вычисления/переписывания/вывода) функциональных выражений.

Как и в любом формальном исчислении (в частности, логическом), преобразования функциональных выражений в лямбда-исчислении носят чисто синтаксический характер, являясь по сути текстовыми преобразованиями (переписыванием строк текста). Таким образом, лямбда-исчисление служит для чисто синтаксического описания свойств функций и манипулирования с ними.

2 Лямбда-выражения

Особенностью синтаксиса функциональных выражений в лямбда-исчислении является использование особой префиксной записи. В отличие от традиционной в математике префиксной записи функций, при которой имя функции стоит перед аргументами, а все аргументы группируются скобками и разделяются запятыми, например: f(x,y) , в лямбда-исчислении применяется префиксная запись, в которой аргументы не разделяются запятыми и не группируются скобками: f x y. По сути разделителем аргументов выступает пробел, а скобки используются при необходимости для группировки выражения, выступающего в роли аргумента функции. Инфиксная же запись операций вида x + y не

применяется вовсе. Вот пример записи в лямбда-исчислении выражения с операциями-функциями сложения и умножения: *(+ x y)2

Приведем формальное определение понятия лямбда-выражения, или лямбда-терма, используя для этого БНФ, (в угловых скобках указываются определяемые понятия):

< лямбда-выражение>::= λ<идентификатор> . <лямбда-выражение> | (1)
<лямбда-выражение>< лямбда -выражение> | (2)
(<лямбда-выражение>) | (3)
<идентификатор> | (4)
<константа> (5)
<константа>::= 1|0|-1|2|-2|… |true|false|+| -| *| /| >| >=| …

В этом определении представлены пять случаев (видов) лямбда-выражения (терма). Первый из них (1) соответствует операции функциональной абстракции. Символ λ обозначает операцию абстракции, после него стоит идентификатор-переменная, которая рассматривается как аргумент функции. Абстракция по сути означает превращение переменной в аргумент функции. После точки записывается выражение для вычисления значения функции, им может быть любое допустимое лямбда-выражение, оно называется телом абстракции. Само же выражение называют лямбда-абстракцией, а переменную между λ и точкой – связанной переменной абстракции.
Лямбда-абстракция интерпретируется следующим образом: функция от указанной переменной, которая возвращает значение – тело абстракции, например:
λx.+ x 1 – функция от x , возвращающая x+1 (увеличение аргумента на 1);
λx.λy.*(+ x y)2 – функция, вычисляющая удвоенную сумму двух своих аргументов x и y (абстракция применена дважды).

Второй случай (вид) лямбда-выражения (2) соответствует функциональной аппликации, т.е. применению (вызову) функции и записывается в виде лямбда-выражения для этой функции (им может быть лямбда-абстракция), за которым следует выражение для её аргумента, например:
f(+ z 9)– применение функции f к аргументу(+ z 9);
(λx.+ x 1)7 – применение функции увеличения на 1 к аргументу-числу 7 ;

Последние примеры демонстрируют также третий случай (3) лямбда-выражения - группировку скобками выражения для аргумента функции или группировку лямбда-абстракции.

Два последних (4), (5) случая лямбда-выражения – идентификатор и константа. Константы включают не только числа, но и знаки (имена) арифметических операций и операций сравнения.

Приведем дополнительные примеры лямбда-выражений: 6 + 6 1, λz.(λy.z), λx.x (последнее – тождественная функция)

Заметим, что лямбда-исчисление может не включать константы – в этом случае *оно* называется чистым. Для чистого исчисления можно сформулировать более короткую БНФ для лямбда-выражения (терма), в которой символ x обозначает переменную (идентификатор), а t - терм,: t::= x | λx.t | t t| (t)

Таким образом, с помощью операции абстракции можно конструировать новые функции, а с помощью аппликации их применять (реализовывать) с конкретными аргументами.

В отличие от математики и программирования лямбда-исчисление оперирует только функциями от одной переменной. Этой минимальной возможности (в смысле количества переменных) достаточно, чтобы выразить функции от нескольких аргументов. Действительно, известны два способа свести функции с несколькими параметрами только лишь к функциям от одного аргумента. В частности, для функции f от двух аргументов: 1). - можно сгруппировать ее аргументы в кортеж (пару) вида(x,y), тогда применение функции имеет вид f(x,y) – что соответствует записи, принятой в математике. 2).- можно использовать идею частичного применения функции, когда у нее фиксируется один аргумент – в результате получается функция, у которой на один аргумент меньше, чем у исходной. Например, фиксируя у функции вычисления максимума один аргумент, мы получаем функцию от другого ее аргумента: max(3,y)=f(y) . Аналогично, для лямбда-абстракции, в которой фиксирован второй аргумент операции сложения: (λx.+ x 1). В обоих случаях функция двух переменных рассматривается как функция одной переменной, возвращающая функцию другой переменной.

Поскольку в лямбда-исчислении используется идея частичного применения, запись в лямбда-исчислении λx.λy.*(+ x y) 2 по сути означает функцию от x, возвращающую функцию от y, которая вычисляет 2*(x+y). Запись же (+ 6) означает одноместную функцию. Поскольку результатом частичного вычисления является другая функция, то функции от нескольких аргументов фактически являются функционалами. Напомним, что функционалом, или функцией высшего порядка называется функция, аргументом или результатом которой является другая функция.

Идею использования частичного вычисления для преобразования функции от нескольких переменных к одноместным (одноаргументным) функциям высказывали многие математики, но названо по имени одного из них – американского математика Хаскелла Карри (Haskel Curry).

Карринг (каррирование) – преобразование функции от нескольких аргументов в эквивалентный набор (суперпозицию) одноаргументных функций (или иными словами, в функцию, берущую аргументы по одному). Функция от двух аргументов f: A×B→C в результате каррирования преобразуется в функцию f':A→(B→C), т.е. отображение декартова произведения множеств A и B (областей определения аргументов) во множество C преобразуется в отображение из A во множество функций из B в C.

В λ–исчислении каррирование фактически делается при записи функций. В некоторые функциональные языки (в частности, в Хаскель и ML) оно встроено, т.е. осуществляется автоматически.

Строго говоря, в лямбда-исчислении применения одноместных функций, образующих функцию от нескольких аргументов, должны заключаться в скобки (B – тело функции), тем не менее для упрощения записи скобки могут быть опущены:

(…((λx1. λx2…. λxn.B) E1)E2)…En)≡ (λx1. λx2…. λxn.B) E1 E2 …En

Такая возможность опирается на установленное соглашение об операции аппликации (применения функции): аппликация левоассоциативна. Это означает, что в записи вида f E1 … En функция всегда применяется к самому левому аргументу, и поэтому скобки

возле каждого вложенного применения можно опустить без изменения смысла выражения. В частности, запись f E1 E2 E3 эквивалентна следующей записи:

f E1 E2 E3 ≡ (((f E1) E2) E3)

u v w = (u v) w

При записи математических выражений скобки полезны, поскольку они фиксируют аргументы операций и тем самым явно указывают структуру выражений. Тем не менее, **соглашения** о приоритете и ассоциативности операций позволяют сократить количество (избавиться от излишних) скобок.

При вычислении алгебраических выражений обычно учитывается и применяется левая ассоциативность операций сложения и умножения: a*b*c=(a*b)*c и правая ассоциативность операции возведения в степень: a↑b↑c=a↑(b↑c).

В лямбда-исчислении используется еще одно соглашение о записи выражений, которое позволяет опускать часть скобок при записи тел лямбда-абстракций. Считается, что тело абстракции простирается как можно дальше – либо до конца всего выражения, либо до первой непарной закрывающей скобки (иными словами: абстракция забирает себе все, до чего дотянется). К примеру, в записи λx.λy.x y z скобки должны быть расставлены следующим образом:

λx.(λy.x y z) ≡ λx.(λy.((x y)z)))

Кроме этих соглашений, скобки играют привычную роль группировки операций в выражениях. Так, выражение λx.λy.(λz.z)x(+ y z)есть лямбда-абстракция, а с дополненными скобками (λx.λy.(λz.z)x)(+ y z) уже получаем лямбда-аппликацию.

Как без лишних скобок, так и с ними, лямбда-выражения могут быть сложны для восприятия: λy.((λy.y)(λx.(λy.y)x))(λz.y z)

Для выявления структуры выражений может быть использован так называемый абстрактный синтаксис. В отличие от конкретного (поверхностного) синтаксиса, относящегося к строчной, текстовой записи выражения, абстрактный синтаксис задает внутреннее представление в виде размеченного дерева, делающего структуру выражений более наглядной и понятной. Эти деревья называются абстрактными синтаксическими деревьями (АСД), или деревьями абстрактного синтаксиса.

Абстрактный синтаксис выявляет структуру выражений различного рода. В вершинах расположены знаки операций, листы – это константы или переменные.

Выражения в абстрактном синтаксисе упрощаются: исключаются лишние символы - разделители, скобки и т.п. В узлах дерева находятся операции и их операнды (последние - в листьях дерева).

Рассмотрим АСД для лямбда-выражений, в этих деревьях в качестве операций

будут фигурировать только абстракция (обозначается символом λ) и аппликация

(обозначается как apply). Например:

λx.λy.x y z = λx.(λy.((x y)z)))

и

λx.λy.(λz.z) x (+ y z) = λx.(λy.(((λz.z)x)(+ y z)))

Можно заметить, что не все переменные во втором выражении связаны некоторой абстракцией. Переменная является свободной в некотором выражении E, если она не является связанной никакой объемлющей операцией абстракции. Таковой является Переменная z, участвующая в операции сложения рассматриваемого выражения, является свободной. Поскольку в это выражение переменная z входит дважды, точнее говорить о связанности/свободности каждого вхождения переменной в некоторое выражение.

Вхождение переменной в выражение E свободно, если в АСД выражение оно не связано никакой выше лежащей абстракцией. Причем если по одной и той же переменной абстракция проводилась несколько раз, то эта переменная связана самой поздней (нижней в дереве) абстракцией. Так, в выражении λx.λy.λx.z переменная x связана с последней абстакцией по x.

Рассмотрим примеры:

В выражении (λy.x)(λx.x):
x свободная↑ x↑связанная
т.е. в своем первом вхождении переменная x свободна, а во втором - связана.

λx.λy.(λz.z)x(+ y z)первое вхождение z связано, а второе - свободно, вхождения же переменных x и y (они единственные) связаны.

Еще пример:

Заметим, что переменная может быть свободной в выражении E, но связанной в объемлющем выражении E', например, переменная z связана в E'=λz.λy.λx.z, и свободна в E=λx.z.

Лямбда-выражение без свободных переменных называется замкнутыми. Замкнутые выражения называются также комбинаторами. Простейший пример комбинатора - тождественная функция: id≡λx.x .

В чистом λ-исчислении всё является функциями (аргументы и результаты функций – функции.

3 Редукция в лямбда-исчислении

Редукция моделирует вычисление лямбда-выражений, она производится по нескольким правилам, представляющим по сути текстовые преобразования. Эти правила включают: α-преобразования, β- и δ-редукцию, а также правило вычисления констант и идентификаторов. Свое название редукция получила потому, что преобразования (вычисление) призваны редуцировать, т.е. упрощать исходное лямбда- выражение.