Лямбда исчисление

Косарев Дмитрий a.k.a. Kakadu

матмех СПбГУ

17 октября 2019 г.

В этих слайдах

1. Исчисление

2. Стратегии вычислений

ACHTUNG!

Любых слайдов абсолютно недостаточно, чтобы разобраться в теме.

Даже если Вы напишите с нуля интерпретатор λ-исчисления, гарантировать полного понимания невозможно.

Читайте умные книжки, например [2].

Правила вывода в исчислении

Пусть дан некоторый язык L, с помощью которого записываются P_i и C.

$(n+1)$-местные правила вывода имеют форму

$$\frac{P_1 \qquad ... \qquad P_n}{C}$$

- P_i – посылки (premises)
- C_i – заключение (conclusion)

По смыслу означает «если и P_1, и P_2, ..., и P_n, то C»

Исчисление

Состоит из

- непустого множества аксиом
- множества правил вывода

Определение

Аксиома – это правило вывода без посылок

Формальное определение можно прочитать в книжке Герамисова [3].

Пример исчисления. Дифференциальное исчисление

Языком L будет язык задания функций (который вообще-то надо формально определять, но не будем)

$$\frac{}{sin'(x) = cos(x)}\ \text{sin} \qquad \frac{}{cos'(x) = -sin(x)}\ \text{cos}$$

$$\frac{f'(x) = u \qquad g'(x) = v}{(u \cdot v)'(x) = u'(x) \cdot v(x) + v'(x) \cdot u(x)}\ \text{mul}$$

$$\frac{f' = u \qquad g' = v}{(f(g(x)))' = u'(g(x)) \cdot v'(x)}\ \text{cmps}$$

Дифференциальное исчисление. Пример вывода

$$\dfrac{\dfrac{}{sin'(x) = cos(x)}\ \text{sin} \qquad \dfrac{\dfrac{}{cos'(x) = -sin(x)}\ \text{cos} \qquad \dfrac{f' = u \qquad g' = v}{(f(g(x)))' = u'(g(x)) \cdot v'(x)}\ \text{cmps}}{cos'(2x) = -sin(2x) \cdot (2 \cdot x')}}{(sin(x) \cdot cos(2x))' = cos(x) \cdot cos(2x) + sin(x) \cdot (-sin(2x) \cdot (2 \cdot x'))}$$

На вывод можно смотреть как на **доказательство** того, что производная действительно посчитана правильно.

Результат вывода можно было бы упростить и дальше, но у нас недостаточно правил вывода для этого.

Язык Λ-выражений

В начале нужно выбрать язык $\mathcal{V}$ имен переменных. Традиционно используются два варианта:

- ✔ Непустая последовательность букв: `[a-z]([a-z])*`
- ☐ Натуральные числа

Из алфавита строятся слова(предложения) языка. Алфавитом для Λ будет $\{(,), \to, \lambda\} \cup \mathcal{V}$

Слова в языке Λ (*лямбда выражения* или *лямбда термы*) строятся по следующим грамматическим правилам

$$\begin{aligned} \langle \text{expr} \rangle &::= \langle \text{varname} \rangle \mid (\lambda \to \langle \text{expr} \rangle) \mid (\langle \text{expr} \rangle \langle \text{expr} \rangle) \\ \langle \text{varname} \rangle &::= \mathcal{V} \end{aligned}$$

Т.е. λ-выражение это или вхождение переменной, либо λ-*абстракция*, либо *применение* (аппликация).

В язык входят скобки, но они часто опускаются.

$$E_1 E_2 \dots E_n \sim (\dots (E_1 E_2) \dots E_n)$$

Например $(\lambda x \to x)$ это функция `id` из Haskell. Первый x – *аргумент* λ-абстракции, второй x – *тело*.

Имена аргументов и переменных не несут существенного смысла, т.е.

$$(\lambda x \to x) \equiv (\lambda y \to y) \equiv (\lambda z \to z) \equiv (\lambda t \to t)$$

Синтаксис $A \equiv B$ означает, что A – синоним B.

Подстановка

Функции можно применять к выражениям, например $(\lambda x \to x)y$ вычисляется путём замены в теле λ-абстракции аргумента абстракции на y, т.е.

$$(\lambda x \to x)\, y = [y/x]x = y$$

Запись $[A/x]B$ означает, что надо подставить A вместо всех вхождений x в B.

Также есть другая нотация

$$[A/x]B = B[x \mapsto A]$$

Свободные и связанные вхождения переменных

Формально:

- Имя n свободно в λ-выражении n.
- Имя n свободно в $(\lambda x \to E)$, если имя $n \neq x$ и n свободно в E.
- Имя n свободно в λ-выражении (MN) если либо оно свободно в M, либо оно свободно в N.

Пример

$$(\lambda x \to xy)(\lambda y \to y)$$

N.B. В одном λ-выражении одно и то же имя может входить и свободно, и связанно.

N.B. В некотором смысле λ – это квантор.

Подстановки

Корректный пример:

$$\mathfrak{I}\mathfrak{I} = (\lambda x \to x)(\lambda x \to x) = (\lambda x \to x)(\lambda z \to z) =$$

$$= [(\lambda z \to z)/x]x = (\lambda z \to z) = \mathfrak{I}$$

Подстановки

Корректный пример:

$$\mathfrak{I}\mathfrak{I} = (\lambda x \to x)(\lambda x \to x) = (\lambda x \to x)(\lambda z \to z) =$$

$$= [(\lambda z \to z)/x]x = (\lambda z \to z) = \mathfrak{I}$$

Некорректный пример:

$$(\lambda x \to (\lambda y \to xy))y \neq (\lambda y \to yy)$$

А нужно делать так

$$(\lambda x \to (\lambda y \to xy))y = (\lambda x \to (\lambda t \to xt))y = (\lambda t \to yt)$$

Основная идея:

- Если у нас есть $(\lambda x \to e)E$, то мы заменяем все *свободные* вхождения x на E.
- Если при замене свободная переменная в E вдруг становится связанной, то мы переименовываем связанную переменную в e перед выполнением подстановки.

Пример:

$$(\lambda x \to (\lambda y \to (x(\lambda x \to xy))))y$$

В $(\lambda y \to (x(\lambda x \to xy)))$ только первый x может быть заменен. Но перед подстановкой необходимо переименовать y в теле на новое имя t.

$$[y/x](\lambda t \to (x(\lambda x \to xt))) = (\lambda t \to (y(\lambda x \to xt)))$$

Редексы (*REDucible EXpressions*)

Редекс – это подвыражение (подтерм) вида $((\lambda x \to e)e_2)$

Стратегия вычислений – способ, по которому мы выбираем какие редексы и в каком порядке будет упрощать (вычислять, β-редуцировать).

Будем обозначать как $e \longrightarrow_\beta e'$ β-редукцию терма e в e'.

Редекс находится *левее*, если его λ в записи левее.

Редекс считается *самый левый внешний(leftmost outermost)*, если он самый левый и не содержится ни в каком другом редексе.

Редекс считается *самый левый внутренний(leftmost innermost)*, если он самый левый и не содержит ни какого другого редекса.

Нормальные формы

У нас четыре возможности

- Редуцируем ли под абстракциями? (да/нет)
- Редуцируем ли аргументы перед подстановкой? (да/нет)

Редуцируем аргументы?	Редуцируем под абстракциями?	
	Да	Нет
Да	Normal form $E ::= \lambda x \to E \mid xE_1 \dots E_n$	Weak Normal form $E ::= \lambda x \to e \mid xE_1 \dots E_n$
Нет	Head normal form $E ::= \lambda x \to E \mid xe_1 \dots e_n$	Weak head normal form $E ::= \lambda x \to e \mid xe_1 \dots e_n$

В таблице E_j – это выражение в соответствующей нормальной форме, а e_i – произвольный λ-терм.

N.B. Нормальной формы может не быть!

Call-By-Name $\rightarrow$ Weak Head Normal Form

Редуцирует **самый левый внешний** терм, который **не под абстракцией**.

$$\frac{}{x \xrightarrow{cbn} x} \qquad \frac{}{(\lambda x \to e) \xrightarrow{cbn} (\lambda x \to e)}$$

$$\frac{e_1 \xrightarrow{cbn} (\lambda x \to e) \qquad [e_2/x]e \xrightarrow{cbn} e'}{(e_1 e_2) \xrightarrow{cbn} e'}$$

$$\frac{e_1 \xrightarrow{cbn} e' \neq (\lambda x \to e)}{(e_1 e_2) \xrightarrow{cbn} (e'_1 e_2)}$$

В lazy языках (например, Haskell) используется вариация CBN под названием Call-By-Need.

Call-by-Value $\to$ Weak Normal Form

Редуцирует **самый левый внутренний** терм, который **не под абстракцией**.

$$\frac{}{x \xrightarrow{cbv} x} \qquad \frac{}{(\lambda x \to e) \xrightarrow{cbv} (\lambda x \to e)}$$

$$\frac{e_1 \xrightarrow{cbv} (\lambda x \to e) \qquad e_2 \xrightarrow{cbv} e_2' \qquad [e_2'/x]e \xrightarrow{cbv} e'}{(e_1 e_2) \xrightarrow{cbv} e'}$$

$$\frac{e_1 \xrightarrow{cbv} e' \neq (\lambda x \to e) \qquad e_2 \xrightarrow{cbv} e_2'}{(e_1 e_2) \xrightarrow{cbv} (e_1' e_2')}$$

Applicative Order → Normal Form

Редуцирует **самый левый внутренний** терм, и **под абстракцией тоже**.

$$\frac{}{x \xrightarrow{ao} x} \qquad \frac{e \xrightarrow{ao} e'}{(\lambda x \to e) \xrightarrow{ao} (\lambda x \to e')}$$

$$\frac{e_1 \xrightarrow{ao} (\lambda x \to e) \qquad e_2 \xrightarrow{ao} e_2' \qquad [e_2'/x]e \xrightarrow{ao} e'}{(e_1 e_2) \xrightarrow{ao} e'}$$

$$\frac{e_1 \xrightarrow{ao} e' \neq (\lambda x \to e) \qquad e_2 \xrightarrow{ao} e_2'}{(e_1 e_2) \xrightarrow{ao} (e_1' e_2')}$$

N.B. Аппликативный порядок совершает больше редукций и выдает более простой ответ по сравнению с CBV, но не гарантирует, что редукция завершится.

Ссылки I


Демки на Haskell
Gitlab repo


Lambda-Calculus and Combinators, an Introduction
J. ROGER HINDLEY & JONATHAN P. SELDIN
PDF


Курс математической логики и теории вычислимости
Герасимов А.С.
PDF


Demonstrating Lambda Calculus Reduction
Peter Setsoft
PDF

Ссылки II


A Tutorial Introduction to the Lambda Calculus
Raúl Rojas
PDF


Lecture Notes on Natural Deduction
Frank Pfenning
PDF