

Applied type theory

lecture 1: introduction into λ – calculus

Воронов Михаил

ВМК МГУ, Fall 2024

План лекции

- 1 Устройство курса
- 2 Введение в λ -исчисление
- 3 Подстановка и преобразования

План лекции

1 Устройство курса

2 Введение в λ -исчисление

3 Подстановка и преобразования

Теоретический план курса

1 Simple untyped λ -calculus

2 $\lambda \rightarrow$

3 Алгебра типов

4 Классические и неклассические логики

5 λ^2

6 $\lambda\omega$

7 λP

8 λC

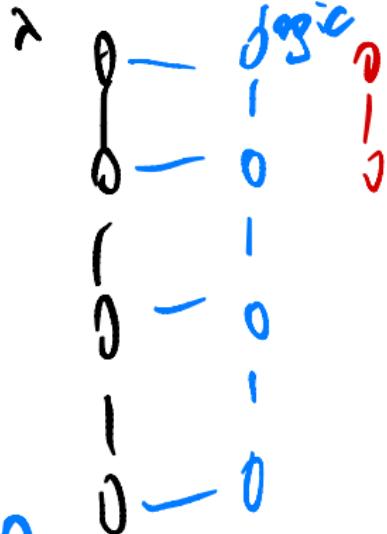
9 Основы MLTT

10 Основы HOTT

11 Основы CubicTT

process calculus

TR-calculus



Практический план курса

- ① Задачи на всевозможный λ -calculus
- ② Coq
- ③ Немного Agda, TLA+, Ocaml

Критерий оценивания

На данный момент планируется 6 домашних заданий, за каждый можно получить 40-50 баллов. Также за экзамен можно получить до 150 баллов, критерии оценивания (могут немного поменяться к концу семестра) за баллы по обязательным задачам:

- ① $< 55\%$ - 2
- ② $55\% - 75\%$ - 3
- ③ $75\% - 95\%$ - 4
- ④ $\geq 95\%$ - 5

Литература

- ① Rob Nederpelt, Herman Geuvers "Type theory and formal proof"
- ② Samuel Mimram "Program = proof" TAPL: ...
- ③ Benjamin Pierce "Types and Programming Languages"
- ④ Yves Bertot "Interactive program proving and program development"
- ⑤ Курс Москвитина Дениса Николаевича "Функциональное программирование" youtube

План лекции

1 Устройство курса

2 Введение в λ -исчисление

3 Подстановка и преобразования

λ -исчисление

$S\lambda LC$
 $STLC$

- λ -исчисление - формальная система, разработанная Алонзо Чёрчем в 1930-х для формализации и анализа понятия вычислимости
- Имеет бестиповую (simple untyped) и множество типовых версий
- Позволяет описывать семантику вычислительных процессов
- Является теоретической основой для многих пруверов

Поведение функций



Рассмотрим функцию $f(x) : x^2 + 1$

- эта функция имеет один "вход" (другими словами, зависит от одной переменной) и один "выход": $f : \mathbb{R} \rightarrow \mathbb{R}$
- в некотором смысле эту функцию можно рассматривать, как отображение $x \rightarrow x^2 + 1$
- чтобы подчеркнуть "абстрактную" роль x , используют специальный символ λ : $\lambda x. x^2 + 1$
- данная нотация выражает то, что x - это не конкретное число, а некоторая абстракция
- для конкретного значения можно "применить" данную функцию: $(\lambda x. x^2 + 1)(3)$

def foo (*):
 return x*x + 1

Введение в λ -исчисление

Из предыдущего слайда следует, что для работы с функциями достаточно двух способов построения выражений:

- ① Абстракция: из выражения M и переменной x можно составить новое выражение $\lambda x.M$ (абстракция x по M)
- ② Применение (application): из двух выражений M и N можно составить новое выражение MN

Введение в λ -исчисление: абстракция

- ① Пусть $M = M[x]$ - выражение, возможно содержащее x
- ② Тогда абстракция $\lambda x.M$ обозначает функцию $x \rightarrow M[x]$
- ③ Абстракция - способ задать неименованную функцию
- ④ Если x в $M[x]$ отсутствует, то $\lambda x.M$ - константная функция со значением M .

$$y = f(x)$$
$$y = 5$$

Введение в λ -исчисление: применение

- ① С точки зрения разработки ПО, применение F к X - это применение алгоритма (F) к данным (X)
- ② Однако явного различия между алгоритмами и данными нет, в частности, возможно самоприменение: FF
- ③ В общем случае применение - это так называемая β -эквивалентность:

$$(\lambda x.M)N =_{\beta} M[x := N]$$

- ④ $M[x := N]$ - это M , в котором вместо N подставлено вместо x

Введение в λ -исчисление: β -редукция

- ① $(\lambda x.x^2 + 1)(3) = (x^2 + 1)[x := 3] = 3^2 + 1$
- ② $(\lambda y.5)(1) = 5[x := 1] = 5$
- ③ $(\lambda x.x)(\lambda y.y) = x[x := (\lambda y.y)] = (\lambda y.y)$
- ④ $\lambda z.((\lambda x.x)(\lambda y.y)) = \lambda z.(x[x := (\lambda y.y)]) = \lambda z.(\lambda y.y)$

Definition

Множество λ -термов Λ определяется индуктивно из переменных $V = x, y, z, \dots$:

- $x \in V \rightarrow x \in \Lambda$ *применение*
- $M, N \in \Lambda \rightarrow (MN) \in \Lambda$ *составление*
- $M \in \Lambda, x \in V \rightarrow (\lambda x. M) \in \Lambda$ *абстракция*

- В абстрактном синтаксисе:

$$\Lambda ::= V | (\Lambda\Lambda) | (\lambda V. \Lambda)$$

- Произвольные термы будем обозначать заглавными буквами, а переменные - строчными

Примеры термов

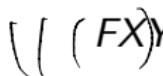
- x
- (xz)
- $(\lambda x.(xz))$
- $((\lambda x.(xz))y)$
- $(\lambda y.((\lambda x.(xz))y))$
- $((\lambda y.((\lambda x.(xz))y))w)$
- $(\lambda z.(\lambda w.((\lambda y.((\lambda x.(xz))y))w)))$

Термы (соглашения)

Общеприняты следующие соглашения:

- Внешние скобки опускаются

- Применение левоассоциативно:

 обозначает $((((FX)Y)Z))$

- Абстракция правоассоциативна:

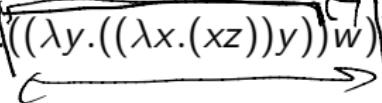
$\lambda xyz.M$ обозначает $(\lambda x.(\lambda y.(\lambda z.(M))))$

- Тело абстракции простирается вправо насколько это возможно

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



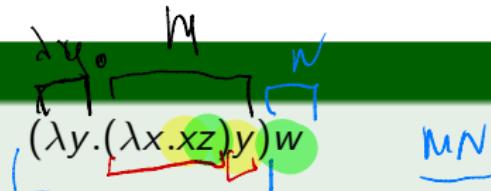
Примеры термов

- $x = x$
- $(xz) = xz$
- $(\lambda x.(xz)) = \lambda x.xz$
- $((\lambda x.(xz))y) = (\lambda x.xz)y$
- $(\lambda y.((\lambda x.(xz))y)) = \lambda y.(\lambda x.xz)y$
- $((\lambda y.((\lambda x.(xz))y))w) = (\lambda y.(\lambda x.xz)y)w$
- $(\lambda z.(\lambda w.((\lambda y.((\lambda x.(xz))y))w))) = \lambda zw.(\lambda y.(\lambda x.xz)y)w$ 

Свободные и связанные переменные

Абстракция $\lambda x.M[x]$ связывает свободную до этого переменную x в терме M .

Example



Переменные x и y - связанные, а z и w - свободные

Example

?

The diagram shows a lambda term $(\lambda x.(\lambda x.xz)x)x$. Handwritten annotations include 'where' above the first x , arrows pointing from the inner x in the red bracket to the outer x in the main term, and 'bound' written below the inner x .

Свободные переменные

```
int glob = read_input();
int foo(int x) {
    return x + glob;
}
```

Definition

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

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

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

$$FV(\lambda x. M) = FV(M) \setminus x;$$

Связанные переменные

Definition

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

$$BV(x) = \{\emptyset\};$$

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

$$BV(\lambda x. M) = BV(M) \cup x;$$

$$\lambda x. (\lambda x. x)$$

Комбинаторы

Definition

Комбинатор (замкнутый λ -терм) M - это такой λ -терм, что $FV(M) = \emptyset$. Множество всех замкнутых термов обозначается Λ^0 .

- $I = \lambda x.x$
- $\omega = \lambda x.xx$
- $\Omega = \omega \omega = (\lambda x.xx)(\lambda x.xx)$
- $K = \lambda xy.x$
- $K_* = \lambda xy.y$
- $S = \lambda fgx.fx(gx)$
- $B = \lambda fgx.f(gx)$

\forall
 $f g$

$f(x, g(x))$
 $f(g(x))$

План лекции

1 Устройство курса

2 Введение в λ -исчисление

3 Подстановка и преобразования

α -преобразование и α -эквивалентность

$$x \times y = dy \cdot y$$

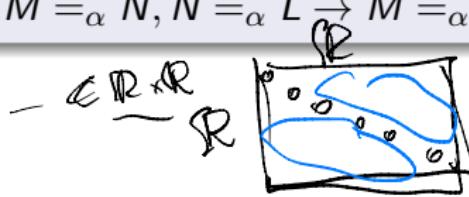
$$x = \alpha y$$

$$x \cdot w \cdot x =_2 dw \cdot y$$

Definition

α -преобразование и α -эквивалентность $=_{\alpha}$ - это отношение, удовлетворяющее следующим свойствам: $\boxed{\text{т. 3}}$

- ① (Переименование) $\lambda x.M =_{\alpha} \lambda y.M^{x \rightarrow y}$
 - ② (Сопоставимость) $M =_{\alpha} N \rightarrow ML =_{\alpha} NL, LM =_{\alpha} LN, \forall x \lambda x.M =_{\alpha} \text{V} \lambda x.N$
 $x \notin V$
 - ③ (Рефлексивность) $M =_{\alpha} M$
 - ④ (Симметричность) $M =_{\alpha} N \rightarrow N =_{\alpha} M$
 - ⑤ (Транзитивность) $M =_{\alpha} N, N =_{\alpha} L \rightarrow M =_{\alpha} L$



Подстановка терма

$$\lambda y. y \cdot (\lambda y. y) \{x = \lambda y. y\} \\ \lambda x. P \{x = \lambda y. y\}$$
$$(\lambda x. y \cdot (\lambda y. y)) = \lambda x. P$$

Definition

Обозначим через $M[x := N]$ подстановку N вместо свободных вхождений x в M , которая подчиняется следующим правилам

- ① $x[x := N] = N$
- ② $y[x := N] = y$
- ③ $(PQ)[x := N] = (P[x := N])(Q[x := N])$
- ④ $(\lambda y. P)[x := N] = \lambda y. (P[x := N]), y \notin FV(N)$
- ⑤ $(\lambda x. P)[x := N] = (\lambda x. P)$

$$\frac{\lambda y. y \cdot (\lambda y. y) \{x = \lambda y. y\}}{\lambda y. y \cdot y}$$

Definition

Обозначим через η преобразование следующего вида:

$$(\lambda x.M)x \underset{\eta}{=} M, x \notin FV(M)$$

$$\lambda x. y. (xyx) [x := \lambda y. y] = \lambda x y. (xyx)$$

$$\begin{aligned}(\lambda y. yx) [x := xy] &=_{\lambda} (\lambda z. zx) [x := xy] \\&= (\lambda z. (zx) [x := xy]) \\&= (\lambda z. (zx[x := xy])) (x[x := xy]) \\&= (\lambda z. \underbrace{z(xy)}_P)\end{aligned}$$

$$P = zx y = ((zy)y)$$

Redex

$$\frac{(\lambda x. P) +}{\text{redex}} \xrightarrow{\beta} P[x := +] \stackrel{=_\lambda}{=} P$$

$$((xy)(\lambda zy. zx(wx)y)) [x := w(\lambda x. wx)] \equiv$$

$$((P_1 P_2) P_3)$$

$$\equiv ((xy)[x := w(\lambda x. wx)]) ((\lambda zy. zx(wx)y)[x := w(\lambda x. wx)])$$

$$= w(\lambda x. wx) y (\lambda zy. (zx(wx)y)) [x := w(\lambda x. wx)] =$$

$$= w(\lambda x. wx) y (\lambda zy. (zx)[x := w(\lambda x. wx)] (wx)[x := w(\lambda x. wx)] y)$$

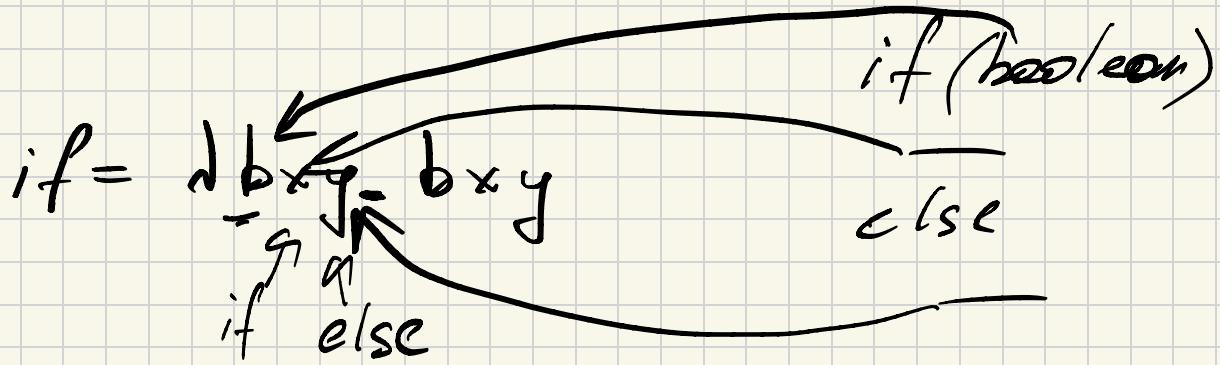
$$= w(\lambda x. wx) y (\lambda zy. z(w(\lambda x. wx))) (w(w(w(\lambda x. wx)))) y$$

$$((zx)(wx))$$

Rozvojovka řešení

$$\text{true} = \lambda f. t = K$$

$$\text{false} = \lambda f. f = K_*$$



$$\text{if true } A \underset{\beta}{=} A$$

$$\text{if false } B \underset{\beta}{=} B$$

$$\text{if true} = (\lambda b \times y. b \times y) (\lambda f. t) \underset{\text{reduces}}{\Rightarrow} \lambda xy. ((\lambda f. t) \times y) \underset{\beta}{\rightarrow}$$

$$\underset{\beta}{\rightarrow} \lambda xy. x$$

$$\text{if false} = \lambda xy. y$$

and = $\lambda x y. x y$ false

and true true = $\lambda ((\lambda x y. x y \text{ false}) \text{ true}) \text{ true} \rightarrow_B$

$\rightarrow_B (\lambda y. \text{ true } y \text{ false}) \text{ true} \rightarrow_B (\text{true } \text{ true}) \text{ false} \rightarrow_B \text{ true}$

and false true $\rightarrow_B ((\text{false } \text{ true}) \text{ false}) \rightarrow_B \text{ false}$

or = ?

or = $\lambda x y. x \text{ true } y$

or true false = $\frac{(\text{true } \text{ true}) \text{ false}}{\lambda f. \text{true}}$

$f(x, y)$ std::bind

$f(f, y)$

true = K = $\lambda f. f$
false = $K_f = \lambda f. f$

def or(x, y):
if x == true:
return true
return y

if $f_1(x) \parallel f_2(y)$ {

==

3

$$0 = \lambda f x. x$$

$$1 = \lambda f x. f x$$

$$\vdots$$

$$n = \lambda f x. \underbrace{f \dots f}_{n} x$$

$$\text{plus} = \lambda \overbrace{m n}^g f x . m f (\lambda f x) \quad (())$$

$$\text{plus } \underline{m} \underline{n} \xrightarrow[B]{\beta} \lambda f x. (\lambda x. \underbrace{f f \dots f}_{m} x) (\underbrace{f \dots f}_{n} x)$$

$$\xrightarrow[B]{\beta} \lambda f x. (\underbrace{f f \dots f}_{m} \underbrace{f \dots f}_{n} x) =$$

$$= m + n$$

Definition

Обозначим через η преобразование следующего вида:
 $(\lambda x.M)x = M, x \notin FV(M)$