Applied Type Theory

Лекция 1: введение в λ-исчисление

Воронов Михаил Сергеевич

ВМК МГУ

Осень 2025

План лекции

План лекции

1. Оргчасть и устройство курса
2. Мотивация
3. Введение в λ-исчисление
4. Формальный синтаксис и нотация
5. Редукции и отношения эквивалетности
6. Комбинаторы и кодирование данных

Теоретический план курса

1. Простое нетипизированное $\lambda \to \lambda_{\to} \to$ алгебра типов.
2. Классические логики и исчисления (NJ/LK).
3. $\lambda 2$, $\lambda\omega$, зависимые типы (MLTT/HoTT).
4. Субструктурные типы: линейные/аффинные, связь с Rust и Haskell.

Практический план курса

1. Теоретические задачи на λ-исчисление
2. Практические задачи на Coq/Agda/TLA+/OCaml
3. Задачи вида "опишите условия и следствия теоремы на языке 7-летнего ребёнка"
4. Практические задачи CTF-типа на sandbox escape
5. Задачи на проектирование API на базе теории типов
6. Каждое домашнее задание будет содержать обязательные задачи и задачи со звёздочкой, которые можно сдать до конца семестра

Критерий оценивания

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

1. 0–79 — 2
2. 80–149 — 3
3. 150–199 — 4
4. 200+ — 5

Литература

- Pierce — *Types and Programming Languages*.
- Nederpelt, Geuvers — *Type Theory and Formal Proof*.
- Software Foundations (Vol. 1–2) — `https://softwarefoundations.cis.upenn.edu/`.
- Bertot, Castéran — *Interactive Theorem Proving and Program Development*.
- Mimram — *Program = Proof* (lecture notes).
- Курс Москвитина Д. Н. — *Функциональное программирование* (YouTube).

План лекции

Зачем это разработчику и специалисту ИБ

Инженерная максима

Свести классы ошибок к *невозможным по типам* и формально проверить и доказать критические свойства систем.

- **Типы как контракты:** сигнатуры фиксируют допустимые состояния и переходы, некорректные программы не компилируются.
- **Раннее обнаружение дефектов:** становятся невозможными целые *классы* уязвимостей (например, null deref, double free, data races, неправильные состояния протокола) на этапе компиляции.
- **Формальная верификация:** Coq/Agda/... доказывают инварианты безопасности и корректность алгоритмов, инвариант «ошибка не может произойти» выражается как теорема.
- **Композиционная надёжность:** свойства собираются из локальных «контрактов» модулей (ADT, parametricity, зависимые типы, сессионные типы).
- **Знание ограничений подхода:** при всех плюсах данный подход не является панацеей и имеет свои ограничения, которые нужно понимать перед использованием.

Примеры использования типовых техник

- **Null dereference** $\Rightarrow$ Option/Maybe, исчерпывающий pattern matching.
- **Use-after-free / double free** $\Rightarrow$ линейные/аффинные типы, владение/заимствование (ownership/borrowing), времена жизни (lifetimes).
- **Data race** $\Rightarrow$ неизменяемость по умолчанию, типы эффектов, правила заимствования, сессионные типы.
- **Out-of-bounds / Heartbleed-класс** $\Rightarrow$ индексированные по длине структуры (напр., $\mathrm{Vec}\ n\ \alpha$), безопасная индексация.
- **Нарушение протокола / неправильный порядок шагов** $\Rightarrow$ typestate, сессионные типы ($!\tau.S,\ ?\tau.S,\ \mathrm{end}$).
- **TOCTTOU** $\Rightarrow$ capability-токены как линейные ресурсы, атомарные переходы состояний.
- **Инъекции (SQL/XSS)** $\Rightarrow$ типизированные AST/DSL вместо строк, параметризованные запросы, GADT-кодирование выражений.
- **Неполное покрытие вариантов** $\Rightarrow$ ADT $+$ исчерпывающий pattern matching.
- **Смешение единиц/домена** $\Rightarrow$ новые типы/фантомные параметры ($\mathrm{Meters},\ \mathrm{Seconds}$); запрет неосмысленных операций.

API как контракт: typestate

Основная идея

Состояние системы выражается в (фантомных) типах, а нелегальные переходы не типизируются.

```
(* Phantom session state *)
type logged_out
type logged_in

type ('state) session

val login : creds -> logged_out session -> logged_in session
val logout : logged_in session -> logged_out session
val transfer : logged_in session -> amount -> account -> logged_in session
(* Attempt transfer without login: ill-typed *)
```

- Контролируем «правильность сценариев» ⇒ отсутствуют *целые классы ошибок* без ad-hoc проверок в коде.

Ресурсы и память: линейные типы

Контракты ресурсов

Субструктурные типы: ресурс потребляется ровно один раз или не более одного раза (например, std::unique_ptr в C++).

$$
\begin{aligned}
\text{alloc} &: \ \text{Unit} \to \text{Cap Buf} \\
\text{write} &: \ \text{Cap Buf} \multimap \text{Byte} \to \text{Cap Buf} \\
\text{free} &: \ \text{Cap Buf} \multimap \text{Unit}
\end{aligned}
$$

- $\text{free} \circ \text{free}$ *не типизируется* $\Rightarrow$ исключаем double free на уровне типов.
- Заимствование $\Rightarrow$ одновременная запись из двух мест не проходит проверку типов $\Rightarrow$ нет data race.

Протоколы как типы

Сессионные типы
Client: !Req.?Ack.end Server: ?Req.!Ack.end
Композиция $\Rightarrow$ корректное взаимодействие, отсутствие перепутанных сообщений и «зависаний». Можно сказать, что в типе кодируется машина состояний протокола.

TLA+ инвариант
$\text{Inv} \triangleq \forall c \in \mathit{Clients}:$
$\mathit{SentReq}[c] \Rightarrow \neg \mathit{RecvAck}[c] \;\; \mathbf{U}$
$\mathit{RecvAck}[c]$
Проверка инварианта Inv model-чекером $\Rightarrow$ (с высокой вероятностью) отсутствие нежелательных межсостояний.

Формальная верификация: что будем доказывать

- **Инварианты API в Agda/Coq:** сохранение размера очереди, отсутствие чтения удалённого, корректность ролей/прав.
- **Свойства протокола в TLA+:** безопасность (safety) и живость (liveness) для простых handshake/lock-free сценариев.
- **Parametricity & free theorems:** автоматические «бесплатные» законы корректности для полиморфного кода (напр., $\mathrm{length}(\mathrm{map}\ f) = \mathrm{length}$).

Практический эффект

Свойства становятся частью артефакта сборки: *если собрано, значит, проходит доказательства и проверки*.

Ограничения и зона ответственности

- Типы и доказательства не заменяют криптографию, настройку ОС и операционные и административные регламенты.
- Канальные утечки и физические побочные каналы требуют дополнительных методик (профилирование времени, изоляция, TEE).
- Политика безопасности должна быть **корректно** *смоделирована*; неверная модель $\Rightarrow$ корректная, но неверная система.
- На практике модели и доказательства часто имеют ограниченно смоделированный контекст предметной области.

Итог мотивации

1. Типы — это исполнимые спецификации: что запрещено — *не скомпилируется*.
2. С помощью Coq/Agda/TLA+ критические свойства становятся теоремами, а не надеждой на тесты.
3. Фокус курса — перенести эти принципы в практику: от теории λ-исчисления до применения этого в API и описания протоколов с использованием типов.

План лекции

λ-исчисление

- λ-исчисление - формальная система, разработанная Алонзо Чёрчем в 1930-х для формализации и анализа понятия вычислимости
- Имеет нетипизированную (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": $f : \mathbb{R} \to \mathbb{R}$
- в некотором смысле эту функцию можно рассматривать, как отображение $x \to x^2 + 1$
- 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: $(\lambda x.x^2 + 1)(3)$

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: 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 $x \to M[x]$
3. Абстракция - способ задать неименованную функцию
4. Если x в $M[x]$ отсутствует, то $\lambda x.M$ - константная функция со значением M.

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 называемое β-преобразование:
$$(\lambda x.M)N \to_\beta M[x := N]$$
4. $M[x := N]$ - это M, в котором свободные вхождения x заменены на N

Введение в λ-исчисление: примеры β-редукции

$$(\lambda x.\, x^2 + 1)\, 3 \;\to_\beta\; 3^2 + 1$$

$$(\lambda y.\, 5)\, 1 \;\to_\beta\; 5$$

$$(\lambda x.\, x)\, (\lambda y.\, y) \;\to_\beta\; \lambda y.\, y$$

$$\lambda z.\big((\lambda x.\, x)\, (\lambda y.\, y)\big) \;\to_\beta\; \lambda z.\, \lambda y.\, y$$

План лекции

Формальное построение λ-исчисления

Definition

Множество λ-термов Λ строится индуктивно:

$$x \in V \Rightarrow x \in \Lambda \qquad M, N \in \Lambda \Rightarrow (MN) \in \Lambda \qquad M \in \Lambda,\, x \in V \Rightarrow (\lambda x.\, M) \in \Lambda$$

- **Абстракция** $\lambda x.\, M$: анонимная функция.
- **Применение** MN: вызов функции M к аргументу N.
- Принятые соглашения: применение левоассоциативно, абстракция правоассоциативна, тело абстракции тянется максимально вправо.

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:

$$FXYZ \text{ обозначает } (((FX)Y)Z)$$

- Абстракция правоассоциативна:

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

- Тело абстракции простирается вправо насколько это возможно

$$\lambda x.MNK \text{ обозначает } \lambda x.(MNK)$$

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

Definition

$\mathrm{FV}(\cdot)$ и $\mathrm{BV}(\cdot)$ задаются рекурсивно:

$$\mathrm{FV}(x) = \{x\}, \quad \mathrm{FV}(MN) = \mathrm{FV}(M) \cup \mathrm{FV}(N), \quad \mathrm{FV}(\lambda x.\, M) = \mathrm{FV}(M) \setminus \{x\};$$
$$\mathrm{BV}(x) = \varnothing, \quad \mathrm{BV}(MN) = \mathrm{BV}(M) \cup \mathrm{BV}(N), \quad \mathrm{BV}(\lambda x.\, M) = \mathrm{BV}(M) \cup \{x\}.$$

Example

В $\lambda y.\, (\lambda x.\, xz)\, y\, w$ связаны x, y, свободны z, w.

Упражнение: свободные и связанные переменные

Для каждого терма определите множества $\mathrm{FV}(\cdot)$ и $\mathrm{BV}(\cdot)$. Обратите внимание на случаи затенения (shadowing) переменных!

1. $\lambda x.\, x\,(\lambda y.\, x\, y\, z)\, u$
2. $\lambda f.\, \lambda x.\, f\,(g\, x)\,(\lambda y.\, h\, y)$
3. $\lambda x.\, (\lambda x.\, x\, y)\, x\, z$
4. $(\lambda a.\, \lambda b.\, a\, c\,(\lambda c.\, b\, c))\, d$
5. $\lambda p.\, \lambda q.\, p\,(\lambda p.\, q\, p\, r)\, s$

План лекции

Бинарные отношения и отношение эквивалентности

Definition (Бинарное отношение)

Пусть A — множество. **Бинарное отношение** R на A — это подмножество $A \times A$. Пишем $a\,R\,b$, если $(a, b) \in R$.

Definition (Свойства отношений)

Для отношения $R \subseteq A \times A$:

- **Рефлексивность**: $\forall a \in A \; a\,R\,a$
- **Симметричность**: $\forall a, b \in A \; a\,R\,b \Rightarrow b\,R\,a$
- **Транзитивность**: $\forall a, b, c \in A \; a\,R\,b \wedge b\,R\,c \Rightarrow a\,R\,c$

Definition (Отношение эквивалентности)

Эквивалентность — это отношение, которое одновременно рефлексивно, симметрично и транзитивно. Эквивалентности индуцируют разбиение множества на *классы эквивалентности*.

Definition (α-преобразование)

$$\lambda x.\, M \xrightarrow{\alpha} \lambda y.\, M[x \mapsto y] \quad \text{если } y \notin \mathrm{FV}(M).$$

Замечание: Здесь $M[x \mapsto z]$ — *переименование* связанного параметра x в M на z (не подстановка терма).

α-преобразование: примеры

- $\lambda x.\,\lambda y.\,x\,(\lambda x.\,y\,x) \;=_\alpha\; \lambda u.\,\lambda v.\,u\,(\lambda x.\,v\,x)$
- $\lambda x.\,(\lambda x.\,x\,x) \;=_\alpha\; \lambda z.\,(\lambda u.\,u\,u)$
- $\lambda y.\,x\,y \;\neq_\alpha\; \lambda x.\,x\,x$ — захват свободного x при $y \to x$ недопустим
- $\lambda x.\,\lambda y.\,x\,y \;\neq_\alpha\; \lambda y.\,\lambda x.\,y\,x$ — перестановка связок не α-эквивалентность
- $\lambda x.\,\lambda y.\,x\,(\lambda y.\,y) \;=_\alpha\; \lambda p.\,\lambda q.\,p\,(\lambda r.\,r)$

Definition

Определим отношение $=_\alpha$ на Λ
индукцией по структуре термов.

- **Переменная** $x =_\alpha x$.
- **Применение** Если $M_1 =_\alpha N_1$ и $M_2 =_\alpha N_2$, то $M_1\, M_2 =_\alpha N_1\, N_2$.
- **Абстракция** $\lambda x.\, M =_\alpha \lambda y.\, N$
  тогда и только тогда, когда существует переменная z такая, что
  $z \notin \mathrm{FV}(M) \cup \mathrm{FV}(N)$ и
  $$M[x \mapsto z] \;=_\alpha\; N[y \mapsto z].$$

Замечание 1: Здесь $M[x \mapsto z]$ — *переименование* связанного параметра x в M на z (не подстановка терма).

Замечание 2: Далее проверяется (нетрудно), что $=_\alpha$ — отношение эквивалентности.

Конвенция Барендрехта (формулировка)

Соглашение (BVC)

В рассуждениях об λ-термах мы по умолчанию работаем с α-эквивалентным представителем M' терма M таким, что:

1. все *связанные* переменные используют попарно разные имена (нет затенения);
2. ни одна связанная переменная не совпадает ни с одной *свободной* переменной: $\mathrm{BV}(M') \cap \mathrm{FV}(M') = \varnothing$.

Интуиция. Имена связанных переменных — «временные ярлыки». Их можно переименовать (α-переименованием) так, чтобы они не мешали подстановкам и доказательствам.

Почему конвенция применима всегда

- α-эквивалентность разрешает свободное переименование связанных переменных на *свежие* имена (не попадающие в FV).
- Алфавит переменных бесконечен $\Rightarrow$ свежие имена всегда существуют.
- Значит, для любого M существует $M' =_\alpha M$, удовлетворяющий BVC, и рассуждать «с точностью до α» корректно.

Мини-следствие

Если $y \notin \mathrm{FV}(M)$, то $\lambda x.\, M \; =_\alpha \; \lambda y.\, M[x \mapsto y]$.

Подстановка без захвата

Definition (Подстановка M[x:=N])

Обозначим через $M[x := N]$ подстановку терма N вместо свободных вхождений x в M, задаваемую правилами

$$x[x := N] = N, \qquad y[x := N] = y \quad (y \neq x),$$

$$(P\,Q)[x := N] = (P[x := N])\,(Q[x := N]),$$

$$(\lambda y.\,P)[x := N] = \begin{cases} \lambda y.\,P, & y = x; \\ \lambda y.\,(P[x := N]), & y \neq x,\ y \notin \mathrm{FV}(N); \\ \lambda z.\,(P[y := z])[x := N], & y \neq x,\ y \in \mathrm{FV}(N),\ z \notin \mathrm{FV}(P) \cup \mathrm{FV}(N). \end{cases}$$

Definition (β-редукция)

$(\lambda x.\, M)\, N \;\to_\beta\; M[x := N]$.

Definition (Редекс)

Редекс (reducible expression) - это выражение вида $(\lambda x.\, M)\, N$.

- β-редукция — основной механизм вычисления в λ-исчислении
- Моделирует применение функции к аргументу
- Подстановка должна избегать захвата переменных

Примеры β-редукции

Example

$$
\begin{aligned}
(\lambda x.\, x)(\lambda y.\, y) &\to_\beta \lambda y.\, y \\
(\lambda f.\, \lambda x.\, f\, x)(\lambda y.\, y) &\to_\beta \lambda x.\, (\lambda y.\, y)\, x \to_\beta \lambda x.\, x \\
(\lambda f.\, \lambda g.\, \lambda x.\, f\,(g\, x))(\lambda y.\, y\, y)(\lambda z.\, z) & \\
&\to_\beta (\lambda g.\, \lambda x.\, (\lambda y.\, y\, y)\,(g\, x))(\lambda z.\, z) \\
&\to_\beta \lambda x.\, (\lambda y.\, y\, y)((\lambda z.\, z)\, x) \\
&\to_\beta \lambda x.\, (\lambda y.\, y\, y)\, x \\
&\to_\beta \lambda x.\, x\, x
\end{aligned}
$$

Замкнутость относительно контекстов

Понятие: «замкнутое относительно контекстов»

Под *контекстом* понимаем терм с одной «дыркой» $C[\,\cdot\,]$, строящийся по грамматике:

$$C ::= [\,\cdot\,] \mid C\,M \mid M\,C \mid \lambda x.\,C.$$

Замкнутость относительно контекстов означает: если $M \sim N$, то для любого C имеем $C[M] \sim C[N]$. Не путать с *замкнутым термом* ($\mathrm{FV}(M) = \varnothing$).

Example

Пусть $C = \lambda z.\,[\,\cdot\,]\,z$. Тогда

$$C[\lambda x.\,x] \;=_\alpha\; C[\lambda y.\,y] \quad \text{т.е.} \quad \lambda z.\,(\lambda x.\,x)\,z \;=_\alpha\; \lambda z.\,(\lambda y.\,y)\,z.$$

β-эквивалентность

Definition (β-эквивалентность)

β-эквивалентность $=^*_\beta$ — наименьшее отношение эквивалентности, замкнутое относительно контекстов и содержащее β-преобразование:

1. (Основа) $(\lambda x.\, M)\, N \to_\beta M[x := N] \Rightarrow (\lambda x.\, M)\, N =^*_\beta M[x := N]$
2. (Рефлексивность) $M =^*_\beta M$
3. (Симметричность) $M =^*_\beta N \Rightarrow N =^*_\beta M$
4. (Транзитивность) $M =^*_\beta N,\ N =^*_\beta L \Rightarrow M =^*_\beta L$
5. (Совместимость) $M =^*_\beta N \Rightarrow \lambda x.\, M =^*_\beta \lambda x.\, N,\ ML =^*_\beta NL,\ LM =^*_\beta LN$

β-эквивалентность: цепочки редукций

Definition («Конструктивное» определение)

Обозначим $M \leftrightarrow_\beta N$, если $M \to_\beta N$ или $N \to_\beta M$ (один шаг в любую сторону). Тогда $M =_\beta N$ тогда и только тогда, когда существует конечная цепочка

$$M_0 \leftrightarrow_\beta M_1 \leftrightarrow_\beta \cdots \leftrightarrow_\beta M_k, \quad M_0 = M, \; M_k = N.$$

Замечание

Эквивалентно: $=_\beta$ — рефлексивно–симметрично–транзитивное замыкание $\to_\beta$ и наименьшая конгруэнция, содержащая $\to_\beta$.

β-эквивалентность: пример

Пусть $M \equiv (\lambda f.\, \lambda x.\ f\,(f\,x))\,g\,x$, $N \equiv g\,(g\,x)$. Тогда

$$M \ \to_\beta\ (\lambda x.\ g\,(g\,x))\,x \ \to_\beta\ g\,(g\,x) \ =\ N,$$

откуда $M =_\beta^* N$.

Ещё цепочка (шаги в обе стороны $\leftrightarrow_\beta$):

$$(\lambda x.\, x)\,((\lambda y.\, y)\,t) \ \leftrightarrow_\beta\ (\lambda y.\, y)\,t \ \leftrightarrow_\beta\ t.$$

Эти примеры иллюстрируют конструктивное определение через конечные цепочки $\leftrightarrow_\beta$ и существование общего редукта.

η-преобразование

Definition (η-преобразование)

$(\lambda x.\, M)\, x \;=_\eta\; M$, если $x \notin FV(M)$.

Definition (η-эквивалентность)

η-эквивалентность $=_\eta$ — это наименьшее отношение эквивалентности, замкнутое относительно контекстов и содержащее η-преобразование:

1. (Основа) $(\lambda x.\, M)\, x =_\eta M$, если $x \notin FV(M)$
2. (Рефлексивность) $M =_\eta M$
3. (Симметричность) $M =_\eta N \Rightarrow N =_\eta M$
4. (Транзитивность) $M =_\eta N, N =_\eta L \Rightarrow M =_\eta L$
5. (Совместимость) $M =_\eta N \Rightarrow \lambda x.\, M =_\eta \lambda x.\, N, ML =_\eta NL, LM =_\eta LN$

Примеры η-эквивалентности

Экстенсиональность функций

η-эквивалентность выражает принцип **экстенсиональности**: две функции равны, если они дают одинаковые результаты для всех возможных аргументов.

Example

$$
\begin{aligned}
(\lambda x.\, f)\, x &=_\eta f \quad (\text{если } x \notin \mathrm{FV}(f)) \\
(\lambda y.\, (\lambda z.\, z))\, y &=_\eta \lambda z.\, z \\
(\lambda a.\, (\lambda b.\, \lambda c.\, b))\, a &=_\eta \lambda b.\, \lambda c.\, b \\
(\lambda x.\, x\, x)\, x &\neq_\eta x \quad (x \in \mathrm{FV}(x\, x))
\end{aligned}
$$

Отношения эквивалентности

Definition (Отношение эквивалентности)

Эквивалентность — это отношение, которое одновременно рефлексивно, симметрично и транзитивно. Эквивалентности индуцируют разбиение множества на *классы эквивалентности*.

Контекст лекции

$=_\alpha$, $=_\beta$ и $=_\eta$ — *отношения эквивалентности* на термах. Мы работаем с термами *с точностью до* этих эквивалентностей:

- по α: переименование связанных переменных несущественно;
- по β: вычисления/подстановки не меняют «смысл» терма;
- по η: функции, совпадающие по действию на всех аргументах, считаются одинаковыми (экстенсиональность).

Отсюда удобно рассуждать о классах эквивалентности $[M] = \{N \mid N \equiv M\}$ и формулировать свойства вроде «НФ единственна (с точностью до α)».

Отношения эквивалентности: пример

Example

$$\lambda x.\, x \;=_\alpha\; \lambda y.\, y, \quad (\lambda x.\, x)\, t \;=_\beta^*\; t, \quad (\lambda x.\, f)\, x \;=_\eta\; f \quad (x \notin \mathrm{FV}(f)).$$

План лекции

Комбинаторы

Definition

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

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

Кодирование Чёрча: булеаны, пары, числа

Булеаны

$$\mathrm{true} \equiv \lambda t.\, \lambda f.\, t, \quad \mathrm{false} \equiv \lambda t.\, \lambda f.\, f$$

$$\mathrm{if} \equiv \lambda b.\, \lambda x.\, \lambda y.\, b\, x\, y$$

Пары

$$\mathrm{pair} \equiv \lambda x.\, \lambda y.\, \lambda p.\, p\, x\, y$$

$$\pi_1 \equiv \lambda p.\, p\, (\lambda x.\, \lambda y.\, x), \quad \pi_2 \equiv \lambda p.\, p\, (\lambda x.\, \lambda y.\, y)$$

Натуральные (Чёрч)

$$0 \equiv \lambda f.\, \lambda x.\, x, \quad 1 \equiv \lambda f.\, \lambda x.\, f\, x, \ \ldots$$

$$\mathrm{succ} \equiv \lambda n.\, \lambda f.\, \lambda x.\, f\, (n\, f\, x)$$

$$\mathrm{plus} \equiv \lambda m.\, \lambda n.\, \lambda f.\, \lambda x.\, m\, f\, (n\, f\, x)$$

Что почитать

- Pierce, гл. 5–7 (untyped λ);
- Software Foundations: *LF*, главы о β-редукции.
- Более продвинутое: Barendregt — *The Lambda Calculus*.