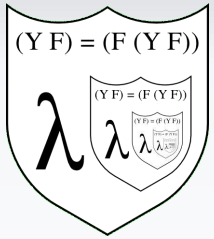
Введение в лямбда-исчисление

Дополнительная лекция
2025 г.

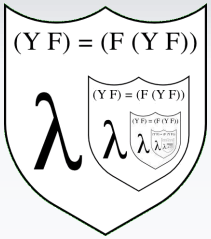


Перегруженность равенства

$$f(a) = g(a + 1)$$

Это:

- равенство заранее заданных f, g в точке a?

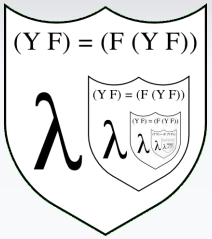


Перегруженность равенства

$$f(a) = g(a + 1)$$

Это:

- равенство заранее заданных f, g в точке a?
- равенство заранее заданных f, g для всех a?

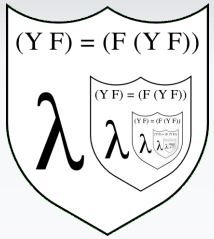


Перегруженность равенства

$$f(a) = g(a + 1)$$

Это:

- равенство заранее заданных f, g в точке a?
- равенство заранее заданных f, g для всех a?
- способ описания **новой** функции f с помощью заранее заданной функции g?

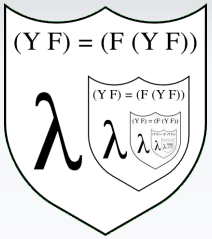


λ-исчисление: новая надежда

$$f(a) = g(a + 1)$$

Происхождение знака λ (согласно Россеру)

$$g(\hat{a} + 1) \rightarrow \hat{a}.g(a + 1) \rightarrow /\backslash a.g(a + 1)$$



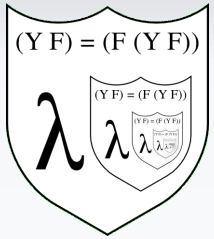
λ-исчисление: новая надежда

$$f(a) = g(a + 1)$$

Наше время

$$f = \lambda a.g(a + 1)$$

Алонзо Чёрч: порождение функций (абстракция) + применение + логические связки = надежда на формализацию математики.



λ-исчисление: новая надежда

$$f(a) = g(a + 1)$$

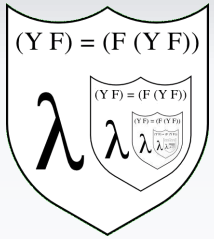
Наше время

$$f = \lambda a.g(a + 1)$$

Алонзо Чёрч: порождение функций (абстракция) + применение + логические связки = надежда на формализацию математики.

Парадокс Карри

Пусть $D = \lambda x.(x\ x) \Rightarrow A$, тогда $(D\ D) \Leftrightarrow ((D\ D) \Rightarrow A)$, что влечёт A.



λ-исчисление: новая надежда

$$f(a) = g(a + 1)$$

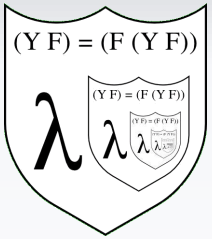
Наше время

$$f = \lambda a.g(a + 1)$$

Алонзо Чёрч: порождение функций (абстракция) + применение + логические связки = надежда на формализацию математики.

Высказывание Карри

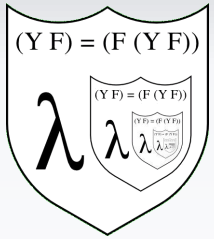
Высказывание C, являющееся собственной посылкой: $C = C \Rightarrow A$.



Неформально о λ-исчислении

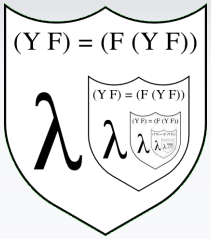
- Мощная модель вычислений, позволяет компактно описывать семантику языков программирования, без акцента на детали реализации.
- Бестиповая версия — А. Черч, 1930-е (и много типизированных).
- Базисные операции — применение (функция → данные) и абстракция (данные → функция).

Абстракция + применение + безопасная подстановка = достаточно мощный инструмент, чтобы выразить любую рекурсивную функцию.



Зачем это нужно знать?

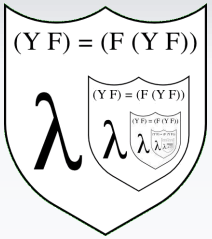
- Культурный минимум ПМИ. Статьи по статическому анализу чаще всего используют тот или иной вариант λ-исчисления.
- Параметрический полиморфизм. Система типов в λ-исчислении — теоретическая основа для функциональных дженериков.
- Композиционный (комбинаторный) стиль. Элементы ФП проникли в мейнстримовые языки (Python, Kotlin, etc). Программирование в комбинаторном стиле — замена паттерну DI в ООП.
- Проектирование DSL. Безопасная подстановка — важный аспект метапрограммирования, статического анализа и DSL. Согласно статье 2014 года на OOPSLA — больше 80% DSL реализованы небезопасно.



Неформально о λ-исчислении

Пусть F, X — термы. F X — операция применения терма F (функции) к терму X (данным). Различения по типам нет, возможно самоприменение: F F.

Пусть $M \equiv M[x]$ — терм, возможно содержащий x. Тогда абстракция $\lambda x.M$ обозначает анонимную (неименованную) функцию от x: $x \mapsto M[x]$.



Неформально о λ-исчислении

Пусть F, X — термы. F X — операция применения терма F (функции) к терму X (данным). Различения по типам нет, возможно самоприменение: F F.

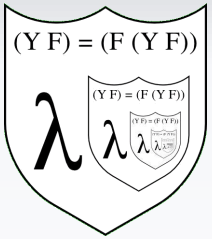
Scheme

```
; Первый элемент пары -- функция, применяемая
; ко второму элементу.
(Fun1 Fun2)
```

Пусть $M \equiv M[x]$ — терм, возможно содержащий x. Тогда абстракция $\lambda x.M$ обозначает анонимную (неименованную) функцию от x: $x \mapsto M[x]$.

Scheme

```
(lambda (x) M)
```



Неформально о λ-исчислении

Применение и абстракция согласованы:

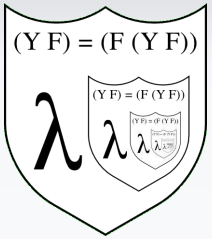
$$(\lambda x.x\ x)\ (\lambda y.y) = (\lambda y.y)\ (\lambda y.y) = \lambda y.y$$

β-эквивалентность:

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

Чистое λ-исчисление

- Конструктор применения
- Конструктор абстракции
- β-эквивалентность.

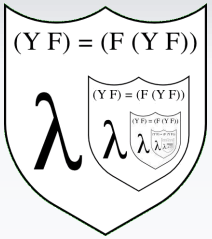


Термы λ-исчисления

- $x \in V \Rightarrow x \in \Lambda$;
- $M, N \in \Lambda \Rightarrow (M\ N) \in \Lambda$;
- $M \in \Lambda, x \in V \Rightarrow (\lambda x.M) \in \Lambda$.

Пример

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



Соглашения о скобках

- Внешние скобки опускаются.
- Применение ассоциативно влево:

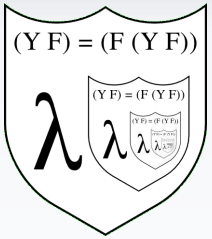
M N P Q — то же, что (((M N) P) Q).

- Можно группировать абстракции (не в Scheme!):

λx y.F — то же, что (λx.(λy.F)).

- Тело абстракции простирается максимально вправо.

λx.M N P — то же, что λx.(M N P).

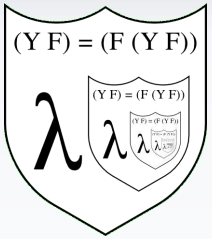


Упрощение скобочных термов (исчисление)

Пример

Внешние скобки опускаются.

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

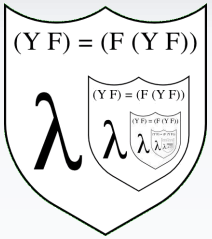


Упрощение скобочных термов (исчисление)

Пример

Применение ассоциативно влево.

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

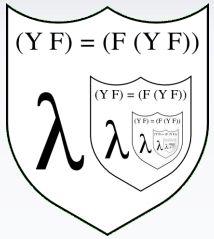


Упрощение скобочных термов (исчисление)

Пример

Абстракции можно сгруппировать.

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

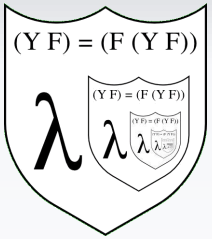


Упрощение скобочных термов (исчисление)

Пример

Итоговый терм (остальные скобки снять нельзя):

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



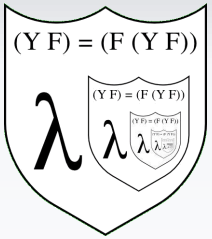
Свободные и связанные переменные

Абстракция $\lambda x.M$ **связывает** переменную x в терме M.

Пример

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

Связанные **вхождения** переменных выделены красным; свободные — синим.



Свободные переменные: формально

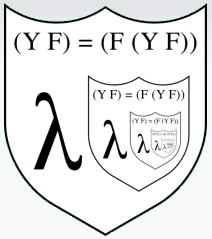
Множество свободных переменных $FV(M)$ в терме M определяется индуктивно:

- $FV(x) = \{x\}$;
- $FV(M\ N) = FV(M) \cup FV(N)$;
- $FV(\lambda x.M) = FV(M) \setminus \{x\}$.

Множество связанных переменных $BV(M)$:

- $BV(x) = \varnothing$;
- $BV(M\ N) = BV(M) \cup BV(N)$;
- $BV(\lambda x.M) = BV(M) \cup \{x\}$.

Верно ли, что $BV(M)$ — множество всех переменных, входящих в M, минус $FV(M)$?



Свободные переменные: формально

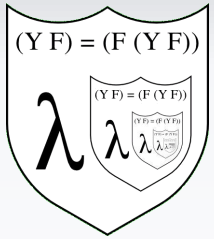
Множество свободных переменных $FV(M)$ в терме M определяется индуктивно:

- $FV(x) = \{x\}$;
- $FV(M\ N) = FV(M) \cup FV(N)$;
- $FV(\lambda x.M) = FV(M) \setminus \{x\}$.

Множество связанных переменных $BV(M)$:

- $BV(x) = \varnothing$;
- $BV(M\ N) = BV(M) \cup BV(N)$;
- $BV(\lambda x.M) = BV(M) \cup \{x\}$.

Верно ли, что $BV(M)$ — множество всех переменных, входящих в M, минус $FV(M)$? Нет! Пример: $(\lambda x.x)\ x$.



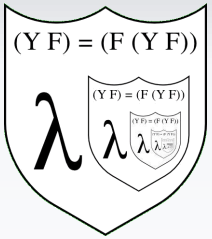
Свободные переменные: формально

Множество свободных переменных $FV(M)$ в терме M определяется индуктивно:

- $FV(x) = \{x\}$;
- $FV(M\ N) = FV(M) \cup FV(N)$;
- $FV(\lambda x.M) = FV(M) \setminus \{x\}$.

- Разные вхождения переменной x могут иметь разный статус!
- Каждая связанная переменная x в $\lambda x.M$ относится к самой внутренней абстракции, связывающей ее!

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



Комбинаторы

Определение

Терм M называется комбинатором, если $FV(M) = \varnothing$.

Часто используемые комбинаторы

$I = \lambda x.x;$

$K = \lambda x\, y.x;$

$K_* = \lambda x\, y.y;$

$\omega = \lambda x.x\ x;$

$\Omega = (\lambda x.x\ x)\ (\lambda x.x\ x);$

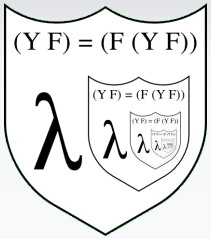
$C = \lambda f\, x\, y.f\ y\ x;$

$B = \lambda f\, g\, x.f\ (g\ x);$

$S = \lambda f\, g\, x.f\ x\ (g\ x).$

Scheme

```
(define K (lambda (x y) x))
```



α-эквивалентность

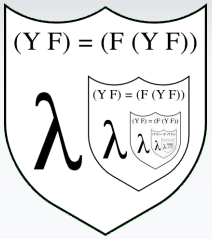
Связанные переменные можно переименовывать.

α-эквивалентные λ-термы

$$\mathbf{I} = \lambda x.x = \lambda y.y = \lambda f.f$$

α-эквивалентные термы дают один и тот же результат при β-преобразовании.

- $(\lambda x.x)\ M = M$;
- $(\lambda y.y)\ M = M$;
- $(\lambda f.f)\ M = M$.



Частичные функции

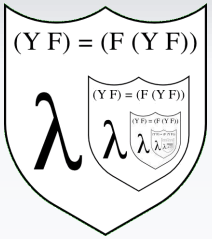
Пусть x, y свободны в $\varphi[x, y]$. Построим абстракции:

- $\Phi_x = \lambda y.\varphi[x, y]$;
- $\Phi = \lambda x.\Phi_x = \lambda x.(\lambda y.\varphi[x, y]) = \lambda x\ y.\varphi[x, y]$.

Применим эти абстракции к двум аргументам (каррирование, карринг):

$$\Phi\ X\ Y = (\Phi\ X)\ Y = \Phi_X\ Y = (\lambda y.\varphi[X, y])\ Y = \varphi[X, Y].$$

Например, $(\lambda x\, y.x + y)\ 3 =_\beta \lambda y.3 + y$ в λ-исчислении можно понимать как операцию прибавления 3.



Частичные функции

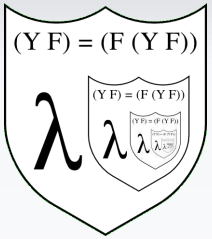
Пусть x, y свободны в $\varphi[x, y]$. Построим абстракции:

- $\Phi_x = \lambda y.\varphi[x, y]$;
- $\Phi = \lambda x.\Phi_x = \lambda x.(\lambda y.\varphi[x, y]) = \lambda x\ y.\varphi[x, y]$.

Применим эти абстракции к двум аргументам (каррирование, карринг):

$$\Phi\ X\ Y = (\Phi\ X)\ Y = \Phi_X\ Y = (\lambda y.\varphi[X, y])\ Y = \varphi[X, Y].$$

Ассоциативность применения влево $\Rightarrow$ можно специализировать только слева направо

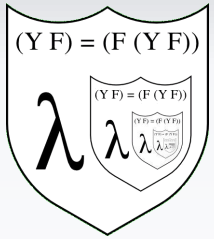


Комбинаторная логика

Комбинаторы можно определить через их поведение на аргументах:

- $\mathbf{I}\ x = x$
- $\mathbf{K}\ x\ y = x$
- $\boldsymbol{\omega}\ x = x\ x$
- $\mathbf{S}\ x\ y\ z = x\ z\ (y\ z)$

Базис $\{\mathbf{K}, \mathbf{S}\}$ + применение — система, эквивалентная λ-исчислению.

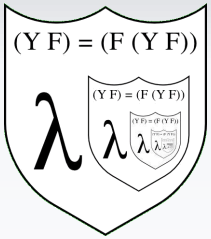


Упражнение

λ-исчисление

$\mathbf{S\ K\ I\ K} =_\beta\ ?$

$\mathbf{S\ K\ K\ K} =_\beta\ ?$

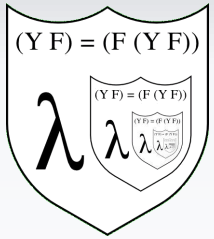


Упражнение

λ-исчисление

$$\mathbf{S}\ \mathbf{K}\ \mathbf{I}\ \mathbf{K} =_\beta \mathbf{K}\ \mathbf{K}\ (\mathbf{I}\ \mathbf{K}) =_\beta \mathbf{K}$$

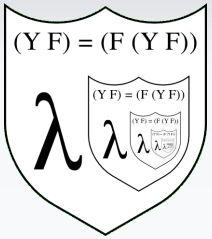
$$\mathbf{S}\ \mathbf{K}\ \mathbf{K}\ \mathbf{K} =_\beta \mathbf{K}\ \mathbf{K}\ (\mathbf{K}\ \mathbf{K}) =_\beta \mathbf{K}$$



Подстановка

Определение

$$
\begin{aligned}
x[x := N] &= N \\
y[x := N] &= y \\
(P\ Q)[x := N] &= (P[x := N])\ (Q[x := N]) \\
(\lambda y.P)[x := N] &= \lambda y.(P[x := N]),\ y \notin FV(N) \\
(\lambda x.P)[x := N] &= \lambda x.P
\end{aligned}
$$



Подстановка

Определение

$$
\begin{array}{rcl}
x[x := N] & = & N \\
y[x := N] & = & y \\
(P\ Q)[x := N] & = & (P[x := N])\ (Q[x := N]) \\
(\lambda y.P)[x := N] & = & \lambda y.(P[x := N]),\ y \notin FV(N) \\
(\lambda x.P)[x := N] & = & \lambda x.P
\end{array}
$$

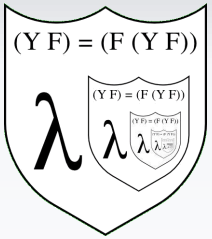
Что делать, если $y \in FV(N)$ (4 правило)?

Пример: $\lambda y.x\ y[x := y]$

Вопрос

Подойдет ли следующее правило, если $y \in FV(N)$?

$$(\lambda y.P)[x := N] = \lambda z.((P[y := z])[x := N]),\ z \notin FV(N) \cup FV(P)$$



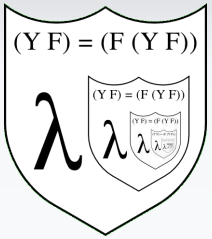
Коллизия (захвата) имен

Соглашение Барендрегта

Имена связанных переменных всегда выбираются так, чтобы они отличались от имен свободных переменных.

Пример

Редукция $((\lambda x\ y.x\ y)\ y)$ приводит к коллизии: $(\lambda y.x\ y)[x := y]$. В редукции $((\lambda x\ z.x\ z)\ y)$ коллизии нет.



Аксиомы λ-исчисления

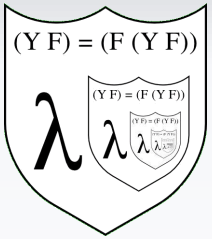
Основная аксиома β-конверсии

Для любых $M, N \in \Lambda$ $\big((\lambda x.M)\ N = M[x := N]\big)$.

Аксиомы равенства

$$M = M$$
$$M = N \Rightarrow N = M$$
$$M = N\ \&\ N = L \Rightarrow M = L$$
$$M = M' \Rightarrow M\ Z = M'\ Z$$
$$M = M' \Rightarrow Z\ M = Z\ M'$$
$$M = M' \Rightarrow \lambda x.M = \lambda x.M'$$

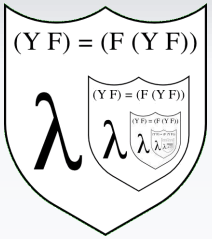
$M = N$ доказуемо — $\lambda \vdash M = N$.



α-преобразование

Основная аксиома α-конверсии

Для любых M, y таких, что $y \notin FV(M)$,
$\lambda x.M =_\alpha \lambda y.M[x := y]$.



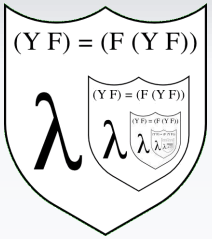
α-преобразование

Основная аксиома α-конверсии

Для любых M, y таких, что $y \notin FV(M)$,
$\lambda x.M =_\alpha \lambda y.M[x := y]$.

...но есть нюанс

Рассмотрим $\lambda x.(\lambda y.x\ y)$. Формально y не свободна в $\lambda y.x\ y$. После подстановки $[x := y]$ получаем: $\lambda y.(\lambda y.y\ y)$. В чем ошибка?



α-преобразование

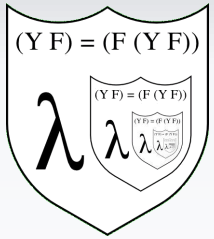
Основная аксиома α-конверсии

Для любых M, y таких, что $y \notin FV(M)$,
$\lambda x.M =_\alpha \lambda y.M[x := y]$.

Вопрос

Подойдет ли следующее правило, если $y \in FV(N)$?

$$(\lambda y.P)[x := N] = \lambda z.((P[y := z])[x := N]),\ z \notin FV(N) \cup FV(P)$$

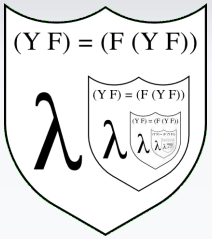


α-преобразование

Основная аксиома α-конверсии

Для любых M, y таких, что $y \notin FV(M)$,
$\lambda x.M =_\alpha \lambda y.M[x := y]$.

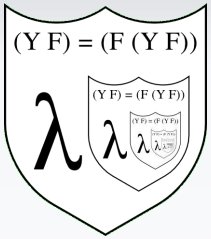
Подстановка не определена полностью, пока не будет пройден весь терм. Имея $(\lambda y.(\lambda z.(y\ z\ x)))[x := y]$, формально строим последовательность:
$\lambda z.((\lambda z.(y\ z\ x))[y := z])[x := y]$
$\lambda z.((\lambda w.((y\ z\ x)[z := w])[y := z]))[x := y]$
$\lambda z.((\lambda w.(y\ w\ x)[y := z]))[x := y]$
$\lambda z.((\lambda w.(z\ w\ x))[x := y])$
$\lambda z.(\lambda w.(z\ w\ y))$



Применение α-конверсии

Пусть $\omega = \lambda x.x\ x, \mathbf{1} =_\alpha \lambda y\, z.y\ z.$

$$
\begin{array}{rcl}
\omega\ \mathbf{1} & = & (\lambda x.x\ x)\ (\lambda y\, z.y\ z) \\
& =_\beta & (\lambda y\, z.y\ z)\ (\lambda y\, z.y\ z) \\
& =_\beta & \lambda z.(\lambda y\, z.y\ z)\ z \\
& =_\alpha & \lambda z.(\lambda y\, z'.y\ z')\ z \\
& =_\beta & \lambda z.(\lambda z'.z\ z') \\
& = & \lambda z\, z'.z\ z'
\end{array}
$$

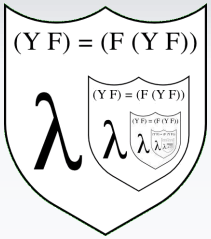


η-конверсия

Схема аксиом η-конверсии

Пусть $x \notin FV(M)$. Тогда $\lambda x.M\ x =_\eta M$.

Поскольку $\forall N((\lambda x.M\ x)\ N) =_\beta (M\ N)$, термы $\lambda x.M\ x$ и M неразличимы по свойствам (экстенсиональность равенства).



η-конверсия

Схема аксиом η-конверсии

Пусть $x \notin FV(M)$. Тогда $\lambda x.M\ x =_\eta M$.

Поскольку $\forall N((\lambda x.M\ x)\ N) =_\beta (M\ N)$, термы $\lambda x.M\ x$ и M неразличимы по свойствам (экстенсиональность равенства).

Примеры

$\lambda x\, y.x\ y =_\eta\ ?$
$\lambda x\, y.y\ x =_\eta\ ?$

η-конверсия

Схема аксиом η-конверсии

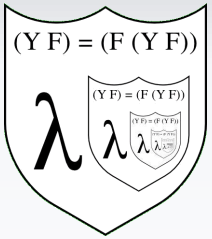
Пусть $x \notin FV(M)$. Тогда $\lambda x.M\ x =_\eta M$.

Поскольку $\forall N((\lambda x.M\ x)\ N) =_\beta (M\ N)$, термы $\lambda x.M\ x$ и M неразличимы по свойствам (экстенсиональность равенства).

Примеры

$$\lambda x\, y.x\ y = \lambda x.\overbrace{(\lambda y.x\ y)}^{\text{конверсия}} =_\eta \lambda x.x$$

$$\lambda x\, y.y\ x =_\eta\ ?$$



η-конверсия

Схема аксиом η-конверсии

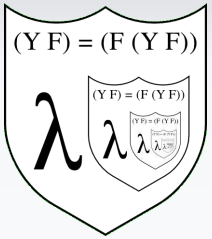
Пусть $x \notin FV(M)$. Тогда $\lambda x.M\ x =_\eta M$.

Поскольку $\forall N((\lambda x.M\ x)\ N) =_\beta (M\ N)$, термы $\lambda x.M\ x$ и M неразличимы по свойствам (экстенсиональность равенства).

Примеры

$$\lambda x\,y.x\ y = \lambda x.\overbrace{(\lambda y.x\ y)}^{\text{конверсия}} =_\eta \lambda x.x$$

$$\lambda x\,y.y\ x = \lambda x.\overbrace{(\lambda y.(y\ x))}^{\text{x во внутренней абстракции}} \text{ редукция невозможна.}$$

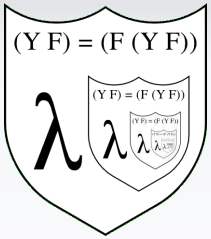


Пример конверсии

Пусть x не свободна в M. Тогда $\lambda x.M\ x =_\eta M$.

Пример редукции:

$$(\lambda x.x\ x)\ (\lambda y\, z.y\ z)$$

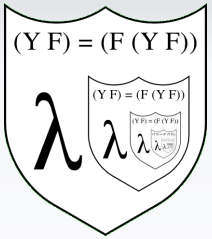


Пример конверсии

$$(\lambda x.x\ x)\ (\lambda y\, z.y\ z)$$

$$\Big\downarrow{\scriptstyle x \mapsto \lambda y\, z.y\ z}$$

$$(\lambda y\, z.y\ z)\ (\lambda y\, z.y\ z)$$



Пример конверсии

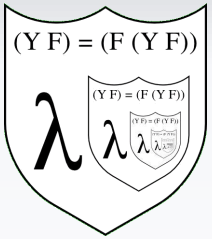
$$(\lambda x.x\ x)\ (\lambda y\, z.y\ z)$$

$$\downarrow_{x \mapsto \lambda y\, z.y\ z}$$

$$(\lambda y.\ (\lambda z.y\ z))\ (\lambda y\, z.y\ z)$$

$$\downarrow_{y \mapsto \lambda y\, z.y\ z}$$

$$\lambda z.(\lambda y\, z.y\ z)\ z$$



Пример конверсии

$$(\lambda x.x\ x)\ (\lambda y\, z.y\ z)$$

$$\downarrow \quad x \mapsto \lambda y\, z.y\ z$$

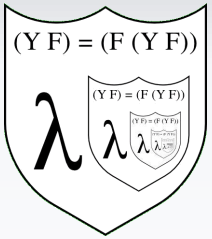
$$(\lambda y.\ (\lambda z.y\ z))\ (\lambda y\, z.y\ z)$$

$$\downarrow \quad y \mapsto \lambda y\, z.y\ z$$

$$\lambda z.(\lambda y\, z.y\ z)\ z$$

$$\downarrow \quad \alpha\text{—преобразование}$$

$$\lambda z.(\lambda y\, z'.y\ z')\ z$$



Пример конверсии

$$(\lambda x.x\ x)\ (\lambda y\, z.y\ z)$$

$$\downarrow\ x \mapsto \lambda y\, z.y\ z$$

$$(\lambda y.\ (\lambda z.y\ z))\ (\lambda y\, z.y\ z)$$

$$\downarrow\ y \mapsto \lambda y\, z.y\ z$$

$$\lambda z.(\lambda y\, z.y\ z)\ z$$

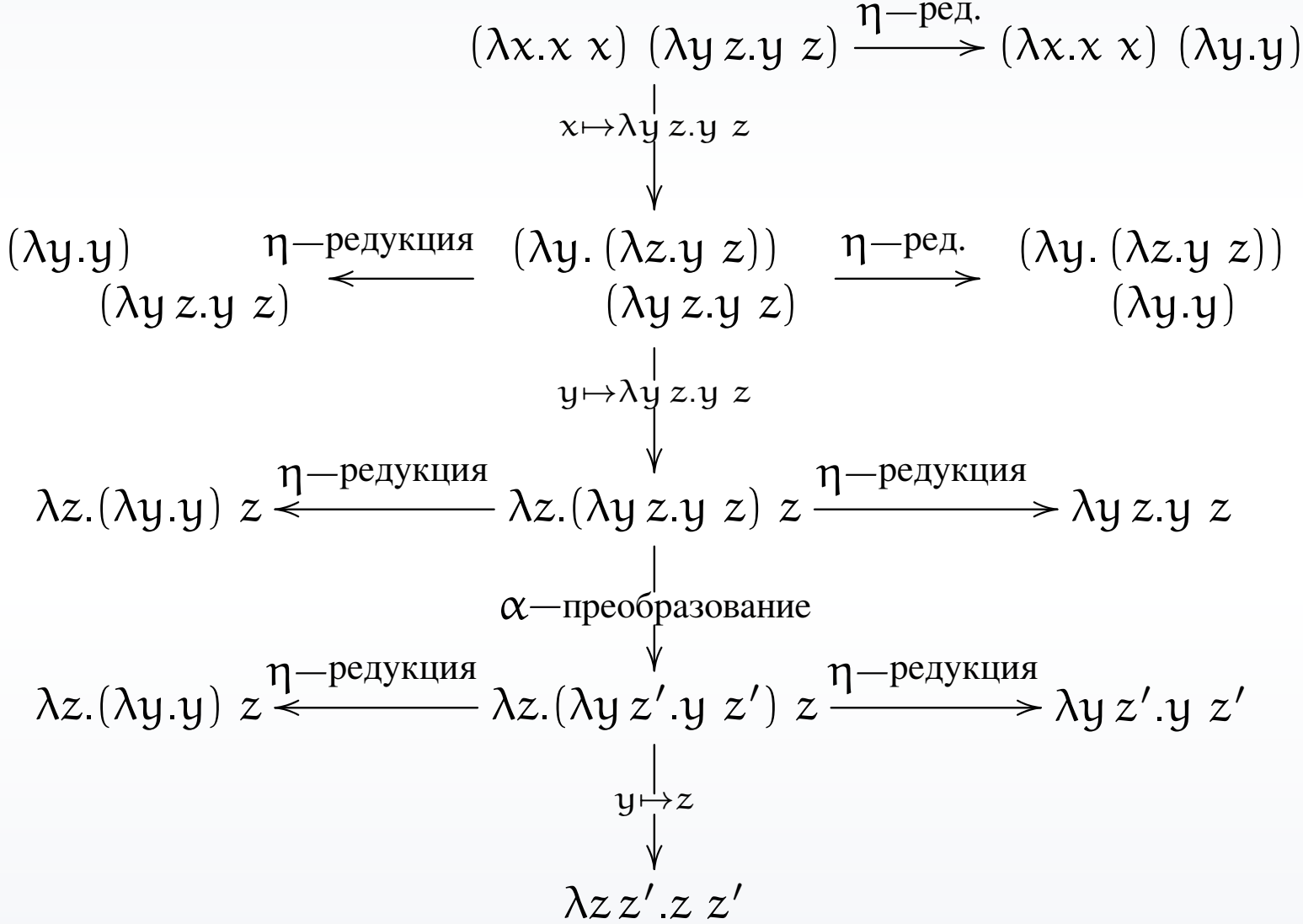
$$\downarrow\ \alpha\text{—преобразование}$$

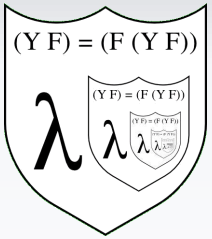
$$\lambda z.(\lambda y.\ (\lambda z'.y\ z'))\ z$$

$$\downarrow\ y \mapsto z$$

$$\lambda z\, z'.z\ z'$$

Пример конверсии

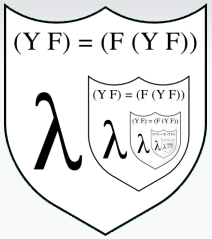




Применение η-конверсии

- $\mathbf{I}\ x = x$
- $\mathbf{K}\ x\ y = x$
- $\mathbf{S}\ x\ y\ z = x\ z\ (y\ z)$

Базис $\{\mathbf{K}, \mathbf{S}\}$ + применение + η-конверсия — система, эквивалентная λ-исчислению.



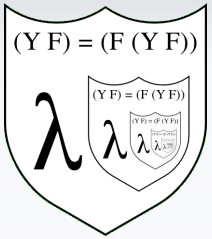
Применение η-конверсии

- $\mathbf{I}\ x = x$
- $\mathbf{K}\ x\ y = x$
- $\mathbf{S}\ x\ y\ z = x\ z\ (y\ z)$

Базис $\{\mathbf{K}, \mathbf{S}\}$ + применение + η-конверсия — система, эквивалентная λ-исчислению.

Задачи

1. Выразить $\mathbf{I}$ в базисе $\{\mathbf{K}, \mathbf{S}\}$.
2. Верно ли, что если $\forall x(\mathbf{X}\ x = \mathbf{Y}\ x)$, то $\mathbf{X}$ и $\mathbf{Y}$ можно свести к одному терму без η-конверсии, используя только правила применения $\mathbf{I}$, $\mathbf{K}$, $\mathbf{S}$, данные выше?
3. Упростить $\mathbf{S}\ (\mathbf{S}(\mathbf{K}\ \mathbf{S})(\mathbf{S}\ (\mathbf{K}\ \mathbf{K})\ \mathbf{K}))(\mathbf{K}\ (\mathbf{S}\ \mathbf{K}\ \mathbf{K}))$.

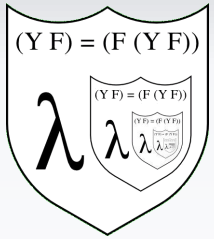


Скобочная абстракция

Интерпретатор $\mathbf{S} + \mathbf{K}$ = минимальный интерпретатор Тьюринг-полного ЯП. Чтобы перевести λ-терм в комбинаторный «байт-код», используется функция скобочной абстракции $\mu(\bullet)$.

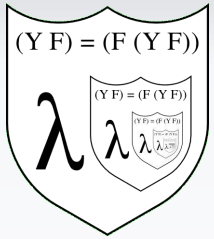
1. $\mu(\lambda x.x) \longrightarrow \mathbf{SKK}$ (для краткости обозначается $\mathbf{I}$);
2. $\mu(\lambda x.M) \longrightarrow \mathbf{K}\mu(M)$, если x не свободна в M;
3. $\mu(\lambda x.(M\ N)) \longrightarrow \mathbf{S}\ (\mu(\lambda x.M))\ (\mu(\lambda x.N))$.

Таким образом удаётся перейти к бесточечному представлению λ-функции.



Скобочная абстракция

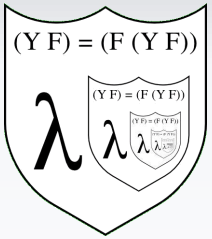
Перейдём к комбинаторной версии `flip id`: $\lambda x\, y.y\ x$.



Скобочная абстракция

Перейдём к комбинаторной версии `flip id`: $\lambda x\,y.y\ x$.

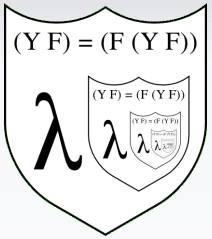
- По алгоритму: $\lambda x.\mathbf{S}\ (\lambda y.y)\ (\lambda y.x) \to \lambda x.\mathbf{S}\ \mathbf{I}\ (\mathbf{K}x) \to \mathbf{S}\ (\lambda x.\mathbf{SI})\ (\lambda x.\mathbf{K}x) \to \mathbf{S}\ (\mathbf{K}(\mathbf{SI}))(\mathbf{S}\ (\lambda x.\mathbf{K})\ (\lambda x.x)) \to \mathbf{S}\ (\mathbf{K}(\mathbf{SI}))(\mathbf{S}\ (\mathbf{KK})\ \mathbf{I})$.



Скобочная абстракция

Перейдём к комбинаторной версии `flip id`: $\lambda x\, y.y\ x$.

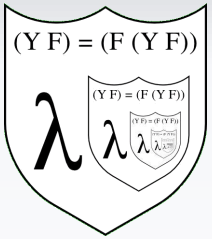
- По алгоритму: $\lambda x.\mathbf{S}\ (\lambda y.y)\ (\lambda y.x) \to \lambda x.\mathbf{S}\ \mathbf{I}\ (\mathbf{K}x) \to \mathbf{S}\ (\lambda x.\mathbf{SI})\ (\lambda x.\mathbf{K}x) \to \mathbf{S}\ (\mathbf{K}(\mathbf{SI}))(\mathbf{S}\ (\lambda x.\mathbf{K})\ (\lambda x.x)) \to \mathbf{S}\ (\mathbf{K}(\mathbf{SI}))(\mathbf{S}\ (\mathbf{KK})\ \mathbf{I})$.
- А если подумать?



Скобочная абстракция

Перейдём к комбинаторной версии `flip id`: $\lambda x\, y.y\ x$.

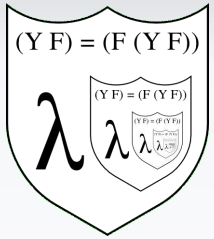
- $\lambda x\, y.y\ x$ меняет местами переменные, а $\mathbf{K}$ линейна $\Rightarrow$ внешняя функция точно $\mathbf{S}$. А ей нужен ещё хотя бы один аргумент-комбинатор, кроме x и y. Пишем заглушку: $\mathbf{S}\ M_1$.



Скобочная абстракция

Перейдём к комбинаторной версии `flip id`: $\lambda x\, y.y\ x$.

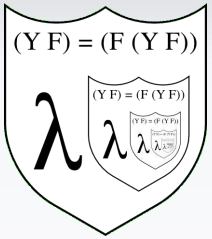
- $\lambda x\, y.y\ x$ меняет местами переменные, а $\mathbf{K}$ линейна $\Rightarrow$ внешняя функция точно $\mathbf{S}$. А ей нужен ещё хотя бы один аргумент-комбинатор, кроме x и y. Пишем заглушку: $\mathbf{S}\ M_1$.
- Тестируем: $\mathbf{S}\ M_1\ x\ y = M_1\ y\ (x\ y)$. Это плохо: из $(x\ y)$ нельзя извлечь x. Нужен ещё один аргумент-комбинатор для $\mathbf{S}$.



Скобочная абстракция

Перейдём к комбинаторной версии `flip id`: $\lambda x\, y.y\ x$.

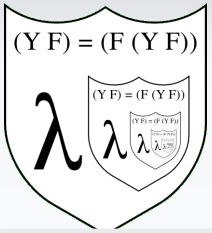
- $\lambda x\, y.y\ x$ меняет местами переменные, а $\mathbf{K}$ линейна $\Rightarrow$ внешняя функция точно $\mathbf{S}$. А ей нужен ещё хотя бы один аргумент-комбинатор, кроме x и y. Пишем заглушку: $\mathbf{S}\ M_1$.
- Тестируем: $\mathbf{S}\ M_1\ x\ y = M_1\ y\ (x\ y)$. Это плохо: из $(x\ y)$ нельзя извлечь x. Нужен ещё один аргумент-комбинатор для $\mathbf{S}$.
- $\mathbf{S}\ M_1\ M_2\ x = M_1\ x\ (M_2\ x)$. Переменная x раздвоилась, причём её первое вхождение явно лишнее. Избавимся от него, положив $M_1 = \mathbf{K}\ M_3$.



Скобочная абстракция

Перейдём к комбинаторной версии `flip id`: $\lambda x\, y.y\ x$.

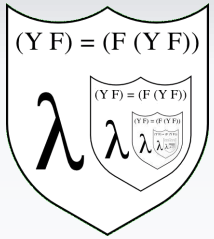
- $\lambda x\, y.y\ x$ меняет местами переменные, а $\mathbf{K}$ линейна $\Rightarrow$ внешняя функция точно $\mathbf{S}$. А ей нужен ещё хотя бы один аргумент-комбинатор, кроме x и y. Пишем заглушку: $\mathbf{S}\ M_1$.
- Тестируем: $\mathbf{S}\ M_1\ x\ y = M_1\ y\ (x\ y)$. Это плохо: из $(x\ y)$ нельзя извлечь x. Нужен ещё один аргумент-комбинатор для $\mathbf{S}$.
- $\mathbf{S}\ M_1\ M_2\ x = M_1\ x\ (M_2\ x)$. Переменная x раздвоилась, причём её первое вхождение явно лишнее. Избавимся от него, положив $M_1 = \mathbf{K}\ M_3$.
- Получаем $M_3\ (M_2\ x)$. Из переменных остался только y, значит, M_3 должен иметь вид $M_4\ M_5$, причём $M_4 = \mathbf{S}$, иначе до y добраться не удастся.



Скобочная абстракция

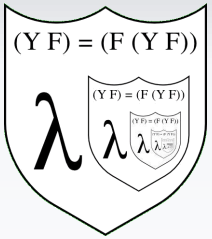
Перейдём к комбинаторной версии `flip id`: $\lambda x\, y.y\ x$.

- $\lambda x\, y.y\ x$ меняет местами переменные, а $\mathbf{K}$ линейна $\Rightarrow$ внешняя функция точно $\mathbf{S}$. А ей нужен ещё хотя бы один аргумент-комбинатор, кроме x и y. Пишем заглушку: $\mathbf{S}\ M_1$.
- Тестируем: $\mathbf{S}\ M_1\ x\ y = M_1\ y\ (x\ y)$. Это плохо: из $(x\ y)$ нельзя извлечь x. Нужен ещё один аргумент-комбинатор для $\mathbf{S}$.
- $\mathbf{S}\ M_1\ M_2\ x = M_1\ x\ (M_2\ x)$. Переменная x раздвоилась, причём её первое вхождение явно лишнее. Избавимся от него, положив $M_1 = \mathbf{K}\ M_3$.
- Получаем $M_3\ (M_2\ x)$. Из переменных остался только y, значит, M_3 должен иметь вид $M_4\ M_5$, причём $M_4 = \mathbf{S}$, иначе до y добраться не удастся.
- $\mathbf{S}\ M_5\ (M_2\ x)\ y = M_5\ y(M_2\ x\ y)$. Теперь очевидно, что $M_5 = \lambda x.x = \mathbf{I}$, $M_2 = \mathbf{K}$. Значит, `flip id` $= \mathbf{S}(\mathbf{K}(\mathbf{SI}))\mathbf{K}$.



Подытожим

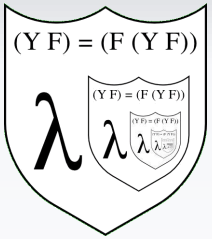
- α-преобразование — переименование связанных переменных;
- β-редукция — применение функции к терму;
- η-преобразование — переход к бесточечным версиям функций и обратно.



Редексы

Определение

Терм $(\lambda x.M[x])\ N$ — **редекс**.
Замена редекса на $M[x := N]$ — **сокращение редекса**.

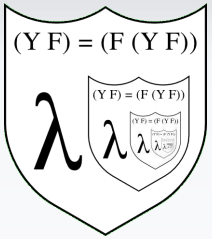


Редексы

Определение

Терм $(\lambda x.M[x])\ N$ — **редекс**.
Замена редекса на $M[x := N]$ — **сокращение редекса**.

- Сколько редексов может быть в терме (один или...)?
- Всегда ли сокращение редекса приводит к сокращению терма?



Редексы

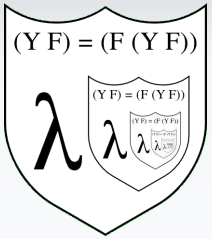
Определение

Терм $(\lambda x.M[x])\ N$ — **редекс**.
Замена редекса на $M[x := N]$ — **сокращение редекса**.

Одношаговая β-редукция

$M \to_\beta N$ определяется следующим образом:

- $(\lambda x.M)\ N \to_\beta M[x := N]$
- $M \to_\beta N \Rightarrow M\ Z \to_\beta N\ Z$
- $M \to_\beta N \Rightarrow Z\ M \to_\beta Z\ N$
- $M \to_\beta N \Rightarrow \lambda x.M \to_\beta \lambda x.N$

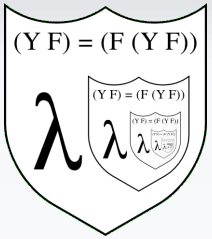


β-редукция

Определение

- β-редукция — транзитивное рефлексивное замыкание $\to_\beta$.
- β-эквивалентность $=_\beta$ — симметричное транзитивное замыкание β-редукции.
- Терм находится в β-нормальной форме (NF), если он не содержит редексов.
- Терм M имеет β-нормальную форму, если существует N: $M =_\beta N$ и N находится в β-NF.

Все ли λ-термы имеют нормальную форму?



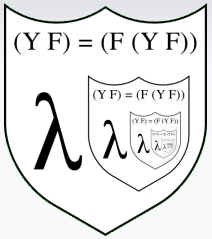
Теорема Чёрча-Россера

Теорема (конфлюэнтность)

Если терм M β-редуцируется к термам N и N′, то существует терм L такой, что N и N′ оба β-редуцируются к L.

Единственность β-NF

λ-терм имеет не больше одной β-NF.

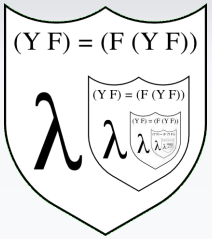


Стратегии редукции

- **Нормальная** — сокращается самый левый внешний редекс.
- **Аппликативная** — сокращается самый левый внутренний редекс.

Теорема о нормализации

Если терм имеет β-NF, то к ней гарантированно приводит нормальная стратегия редукции.



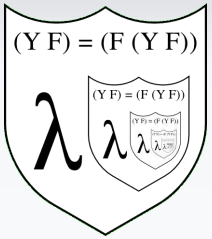
Термы без нормальной формы

Термы вида $\lambda x_1, \ldots x_n.x_i\ Q$, где Q произвольно (в том числе может содержать редексы), называются термами в головной нормальной форме.

- Если для терма T выполняется условие: $\exists N_1 \ldots N_k(T\ N_1 \ldots N_k = \mathbf{I})$, он называется разрешимым.

- Терм разрешим $\Leftrightarrow$ у него существует головная нормальная форма .

Неразрешимые термы (вроде $\mathbf{\Omega}$) понимаются как всегда зацикливающиеся и условно отождествляются друг с другом. Разрешимые термы без нормальной формы — частично определенные функции.



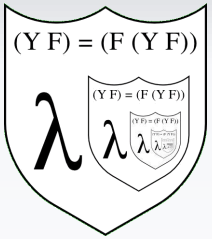
Противоречивость λ-исчисления

Парадокс Рассела

Рассмотрим $\mathbf{R} = \lambda x.\neg(x\ x)$.

$$\begin{array}{rcl}(\mathbf{R}\ \mathbf{R}) & = & (\lambda x.\neg(x\ x))\ (\lambda x.\neg(x\ x)) \\ & =_\beta & \neg((\lambda x.\neg(x\ x))\ (\lambda x.\neg(x\ x))) \\ & = & \neg(\mathbf{R}\ \mathbf{R})\end{array}$$

Чистое (без логических операторов) λ-исчисление непротиворечиво.

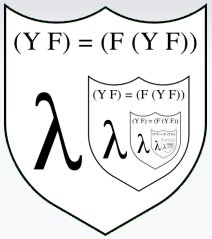


Противоречивость λ-исчисления

Чистое (без логических операторов) λ-исчисление непротиворечиво.

Пример

$\mathbf{K} = \lambda x\, y.x$, $\mathbf{K}_* = \lambda x\, y.y$. Если $\mathbf{K} = \mathbf{K}_*$, то $\forall x, y(x = y)$, поэтому $\mathbf{K} \neq \mathbf{K}_*$ в чистом λ-исчислении.



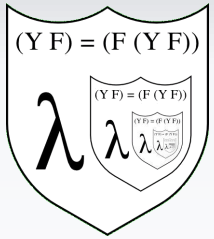
Противоречивость λ-исчисления

Чистое (без логических операторов) λ-исчисление непротиворечиво.

Пример

$\mathbf{K} = \lambda x\, y.x$, $\mathbf{K}_* = \lambda x\, y.y$. Если $\mathbf{K} = \mathbf{K}_*$, то $\forall x, y(x = y)$, поэтому $\mathbf{K} \neq \mathbf{K}_*$ в чистом λ-исчислении.

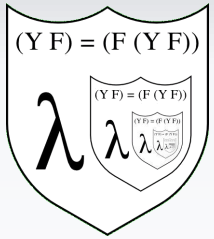
А как насчет $\mathbf{K} = \mathbf{I}$?



Практика

Booleans

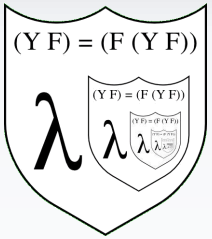
Положим $\mathbf{T} = \lambda x\, y.x$, $\mathbf{F} = \lambda x\, y.y$, $\mathbf{IF} = \lambda b\, x\, y.b\ x\ y$.
Построить **AND**, **OR**, **NOT**.



Практика

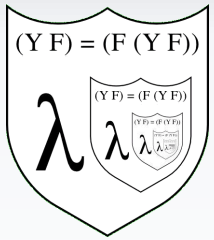
Нумералы Черча

Положим $\mathbf{0} = \lambda x\, y.y$, $\mathbf{1} = \lambda x\, y.x\ y$, $\mathbf{2} = \lambda x\, y.x\ (x\ y)$ и т.д.
Как выразить IsZero? Succ?



Полуформально о типизации λ-функций

1. Если $M[x]$ имеет тип σ в контексте $x : \tau$, тогда естественно, что $\lambda x.M$ имеет тип $\tau \to \sigma$;
2. Если $(M\ N)$ имеет тип σ, а N имеет тип τ, тогда естественно, что M имеет тип $\sigma \to \tau$.



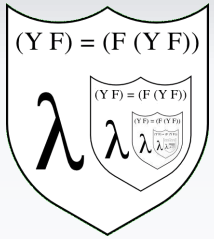
Полуформально о типизации λ-функций

1. Если $M[x]$ имеет тип σ в контексте $x : \tau$, тогда естественно, что $\lambda x.M$ имеет тип $\tau \to \sigma$;
2. Если $(M\ N)$ имеет тип σ, а N имеет тип τ, тогда естественно, что M имеет тип $\sigma \to \tau$.

Логическая спецификация

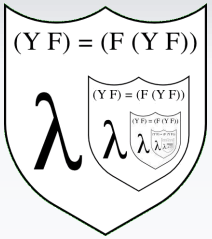
$$\frac{\Gamma, x : \tau \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \to \sigma}$$

$$\frac{\Gamma \vdash M : \tau \to \sigma, \Gamma \vdash N : \tau}{\Gamma \vdash (M\ N) : \sigma}$$



Проблема типизации λ-функций

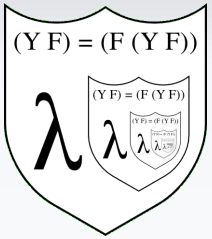
Рассмотрим терм $\lambda x.(x\ x)$. Какой у него тип?



Проблема типизации λ-функций

Рассмотрим терм $\lambda x.(x\ x)$. Какой у него тип?

- Пусть тип аргумента применения (то есть x) — это τ, а тип результата применения (то есть $(x\ x)$) — это σ. Тогда $\tau = \tau \to \sigma$. Ничего не напоминает?

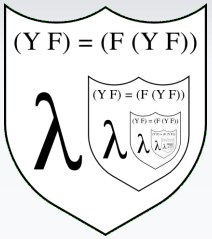


Проблема типизации λ-функций

Рассмотрим терм $\lambda x.(x\ x)$. Какой у него тип?

- Пусть тип аргумента применения (то есть x) — это τ, а тип результата применения (то есть $(x\ x)$) — это σ. Тогда $\tau = \tau \to \sigma$. Ничего не напоминает?

Уравнение $\tau = \tau \to \sigma$ — это предложение Карри! Оно не имеет неподвижной точки, отличной от $\bot$.



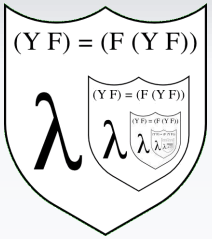
Проблема типизации λ-функций

Рассмотрим терм $\lambda x.(x\ x)$. Какой у него тип?

- Пусть тип аргумента применения (то есть x) — это τ, а тип результата применения (то есть $(x\ x)$) — это σ. Тогда $\tau = \tau \to \sigma$. Ничего не напоминает?

Уравнение $\tau = \tau \to \sigma$ — это предложение Карри! Оно не имеет неподвижной точки, отличной от $\bot$.

Зацикливается не только унификация: см. $(\lambda x.(x\ x))\ (\lambda x.(x\ x))$. Иногда успешно вычисляется: напр. $(\lambda x.(x\ x))\ (\lambda x.(\lambda y.(y\ x)))$.



Проблема типизации λ-функций

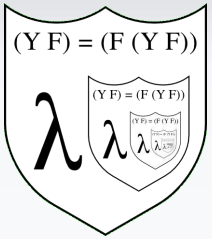
Рассмотрим терм $\lambda x.(x\ x)$. Какой у него тип?

- Пусть тип аргумента применения (то есть x) — это τ, а тип результата применения (то есть $(x\ x)$) — это σ. Тогда $\tau = \tau \to \sigma$. Ничего не напоминает?

Уравнение $\tau = \tau \to \sigma$ — это предложение Карри! Оно не имеет неподвижной точки, отличной от $\bot$.

Зацикливается не только унификация: см. $(\lambda x.(x\ x))\ (\lambda x.(x\ x))$. Иногда успешно вычисляется: напр. $(\lambda x.(x\ x))\ (\lambda x.(\lambda y.(y\ x)))$.

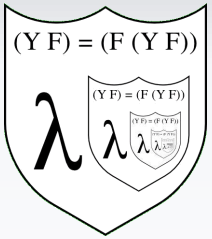
$\lambda x.(x\ x)$ — частичная функция и не может быть конечным образом определена на всех полиморфных типах.



Просто типизированное λ-исчисление

Ограничим множество λ-термов только такими, типы которых всегда выводимы по описанным выше правилам.

$$\frac{\Gamma, x : \tau \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \to \sigma} \qquad \frac{\Gamma \vdash M : \tau \to \sigma, \Gamma \vdash N : \tau}{\Gamma \vdash (M\ N) : \sigma}$$



Просто типизированное λ-исчисление

Ограничим множество λ-термов только такими, типы которых всегда выводимы по описанным выше правилам.

$$\frac{\Gamma, x : \tau \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \to \sigma} \qquad \frac{\Gamma \vdash M : \tau \to \sigma, \Gamma \vdash N : \tau}{\Gamma \vdash (M\ N) : \sigma}$$

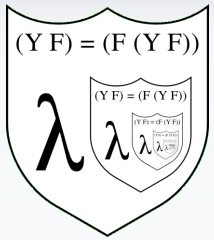
...а теперь забудем про термы и посмотрим только на типы. Что получилось?

$$\frac{\Gamma, \tau \vdash \sigma}{\Gamma \vdash \tau \to \sigma}$$

(правило введения импликации)

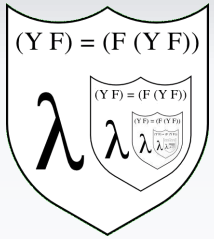
$$\frac{\Gamma \vdash \tau \to \sigma, \Gamma \vdash \tau}{\Gamma \vdash \sigma}$$

(правило удаления импликации aka *modus ponens*)



Связь логики и ФВП: соответствие Карри–Ховарда

- Существует взаимно-однозначное соответствие между типами замкнутых термов в просто типизированном λ-исчислении и тавтологиями в минимальной импликативной логике.
- (теорема о нормализации) Все термы просто типизированного λ-исчисления имеют нормальную форму.
- Доказательствам в минимальной логике соответствуют всюду определенные полиморфные функции высшего порядка.



Вывод = конструкция

Комбинаторная логика Карри

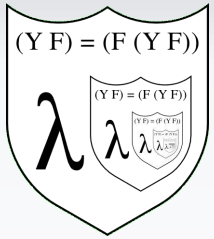
- комбинатор $\mathbf{K} :: A \Rightarrow (B \Rightarrow A)$
- комбинатор $\mathbf{S} ::$
  $(A \Rightarrow (B \Rightarrow C)) \Rightarrow (A \Rightarrow B) \Rightarrow (A \Rightarrow C)$

А теперь перенесёмся на 100 лет назад, во времена Д. Гильберта...

Схемы аксиом для минимальной импликативной логики:

- $\Phi \Rightarrow (\Psi \Rightarrow \Phi)$, где Φ, Ψ — любые формулы;
- $(\Phi \Rightarrow (\Psi \Rightarrow \Xi)) \Rightarrow (\Phi \Rightarrow \Psi) \Rightarrow (\Phi \Rightarrow \Xi)$, где Φ, Ψ, Ξ — любые формулы.

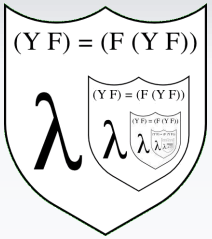
Правила вывода — подстановка + дедукция: $\mathcal{T} \vdash \Phi \Rightarrow \Psi$ влечёт $\mathcal{T}; \Phi \vdash \Psi$. Перенос в контекст отсутствует.



Гильбертовский вывод

Выведем $A \Rightarrow A$ в стиле логики 100-летней давности...

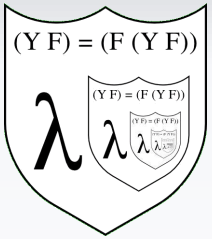
1. Возьмём $(\Phi \Rightarrow (\Psi \Rightarrow \Xi)) \Rightarrow (\Phi \Rightarrow \Psi) \Rightarrow (\Phi \Rightarrow \Xi)$ и положим $\Xi := A$ и $\Phi := A$, $\Psi := A \Rightarrow (B \Rightarrow A)$. Заметим, что Ψ — теорема (частный случай схемы $\Phi \Rightarrow (\Psi \Rightarrow \Phi)$).



Гильбертовский вывод

Выведем $A \Rightarrow A$ в стиле логики 100-летней давности...

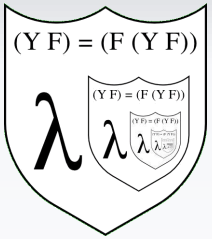
1. Возьмём $(\Phi \Rightarrow (\Psi \Rightarrow \Xi)) \Rightarrow (\Phi \Rightarrow \Psi) \Rightarrow (\Phi \Rightarrow \Xi)$ и положим $\Xi := A$ и $\Phi := A$, $\Psi := A \Rightarrow (B \Rightarrow A)$. Заметим, что Ψ — теорема (частный случай схемы $\Phi \Rightarrow (\Psi \Rightarrow \Phi)$).
2. Получается теорема $(A \Rightarrow ((A \Rightarrow (B \Rightarrow A)) \Rightarrow A)) \Rightarrow (A \Rightarrow (B \Rightarrow A)) \Rightarrow (A \Rightarrow A)$.



Гильбертовский вывод

Выведем $A \Rightarrow A$ в стиле логики 100-летней давности...

1. Возьмём $(\Phi \Rightarrow (\Psi \Rightarrow \Xi)) \Rightarrow (\Phi \Rightarrow \Psi) \Rightarrow (\Phi \Rightarrow \Xi)$ и положим $\Xi := A$ и $\Phi := A$, $\Psi := A \Rightarrow (B \Rightarrow A)$. Заметим, что Ψ — теорема (частный случай схемы $\Phi \Rightarrow (\Psi \Rightarrow \Phi)$).
2. Получается теорема $(A \Rightarrow ((A \Rightarrow (B \Rightarrow A)) \Rightarrow A)) \Rightarrow (A \Rightarrow (B \Rightarrow A)) \Rightarrow (A \Rightarrow A)$.
3. Теперь возьмём $\Phi \Rightarrow (\Psi \Rightarrow \Phi)$ и положим $\Phi := A$, $\Psi := A \Rightarrow (B \Rightarrow A)$. Получим $A \Rightarrow ((A \Rightarrow (B \Rightarrow A)) \Rightarrow A)$.



Гильбертовский вывод

Выведем $A \Rightarrow A$ в стиле логики 100-летней давности...

1. Возьмём $(\Phi \Rightarrow (\Psi \Rightarrow \Xi)) \Rightarrow (\Phi \Rightarrow \Psi) \Rightarrow (\Phi \Rightarrow \Xi)$ и положим $\Xi := A$ и $\Phi := A$, $\Psi := A \Rightarrow (B \Rightarrow A)$. Заметим, что Ψ — теорема (частный случай схемы $\Phi \Rightarrow (\Psi \Rightarrow \Phi)$).
2. Получается теорема $(A \Rightarrow ((A \Rightarrow (B \Rightarrow A)) \Rightarrow A)) \Rightarrow (A \Rightarrow (B \Rightarrow A)) \Rightarrow (A \Rightarrow A)$.
3. Теперь возьмём $\Phi \Rightarrow (\Psi \Rightarrow \Phi)$ и положим $\Phi := A$, $\Psi := A \Rightarrow (B \Rightarrow A)$. Получим $A \Rightarrow ((A \Rightarrow (B \Rightarrow A)) \Rightarrow A)$.
4. Применим дедукцию дважды. Теорема $A \Rightarrow A$ доказана.

Выведем $A \Rightarrow A$ в стиле логики 100-летней давности...

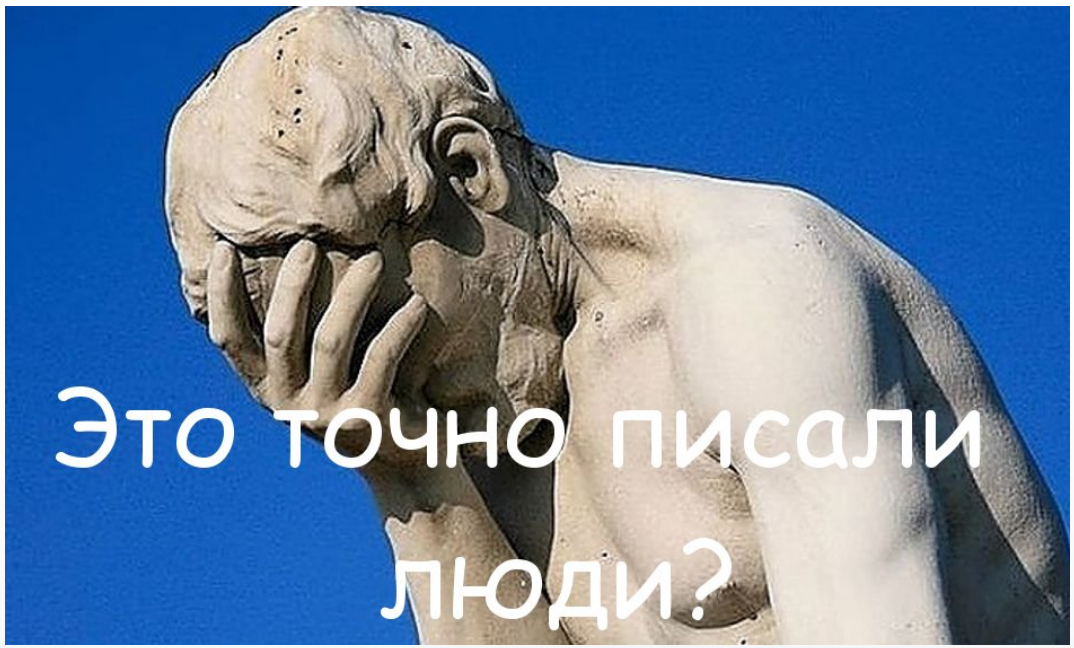
$\mathbf{S}$: $(A \Rightarrow ((A \Rightarrow (B \Rightarrow A)) \Rightarrow A)) \Rightarrow (A \Rightarrow (B \Rightarrow A)) \Rightarrow (A \Rightarrow A)$

$\mathbf{K}$: $A \Rightarrow ((A \Rightarrow (B \Rightarrow A)) \Rightarrow A)$

дедукция: $(A \Rightarrow (B \Rightarrow A)) \Rightarrow (A \Rightarrow A)$

$\mathbf{K}$: $A \Rightarrow (B \Rightarrow A)$

дедукция: $A \Rightarrow A$



"I'm not logician! I'm human!"
(c) R. Glück