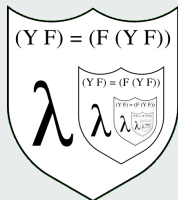


Бестиповое лямбда-исчисление



А.Н. Непейвода
2022 г.



Лямбда-исчисление



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

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

То, что мы знаем теперь

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

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

- $\forall x P(x) \equiv \forall (\lambda x. P(x));$
- $\sum_{i=1}^{\infty} \frac{1}{i^2} \equiv \text{sum}(1, \infty, \lambda x. \frac{1}{x^2}).$

Происходит зарождение синтаксиса функций высшего порядка.



λ -исчисление: новая надежда

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

То, что мы знаем теперь

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

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

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

То, что мы знаем теперь

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

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

- Формальная модель вычислений, позволяет компактно описывать семантику ЯП.
- Бестиповая версия — А. Черч, 1935 (и много типизированных).
- Базисные операции — применение (функция \rightarrow данные) и абстракция (данные \rightarrow функция).



Неформально о λ -исчислении

Пусть F, X — термы. $F X$ — операция применения терма F (функции) к терму X (данному). 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 : $x \rightarrow M[x]$.



Неформально о λ-исчислении

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

Haskell

```
-- 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 : $x \rightarrow M[x]$.

Haskell

```
\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

- Применение
- Абстракция
- 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

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



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

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

Haskell

```
ghci> ((\y -> y) (\y -> y)) 1
1
ghci> (\x y -> y x) (\z1 z2 -> z1 z2) (\y z -> z +1) 3
4
```

Функция $\lambda x.x x$ в базовом Haskell не допустима из-за строгой статической типизации (см. следующая лекция).



Соглашения о скобках

- Внешние скобки опускаются.
- Применение ассоциативно влево:

$M N P Q$ — то же, что $((M N) P) Q$.

- Абстракция ассоциативна вправо:

$\lambda x y.F$ — то же, что $(\lambda x.(\lambda y.F))$.

- Тело абстракции простирается максимально вправо.

$\lambda x.M N P$ — то же, что $\lambda x.(M N P)$.



Соглашения о скобках (Haskell)

Абстракции:

```
\x -> \y -> x (\z -> z)
-- То же, что и λx.(λy.x (λz.z))
-- Но можно короче: \x y -> x (\z -> z)
```

Применения:

```
\x -> \y -> x y y y
-- То же, что и λx.(λy.(((x y) y) y)))
-- Но не это: λxy.x (y y y)
-- И не это: λxy.(x y) (y y)
```



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

Абстракция ассоциативна вправо.

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



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

$$(\lambda x.\textcolor{red}{x} \textcolor{red}{x}) ((\lambda x \textcolor{red}{y}.\textcolor{red}{x}) ((\lambda y.\textcolor{blue}{y}) \textcolor{blue}{z}) \textcolor{blue}{w})$$

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 $BV(M)$:

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

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 $BV(M)$:

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

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 $FV(M) = \emptyset$.

Часто используемые комбинаторы

$I = \lambda x.x;$

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

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

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

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

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

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

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

Haskell

```
k = \x y -> x
```

```
-- id -- имя I, const - имя K, flip -- имя C.
```



α-ЭКВИВАЛЕНТНОСТЬ

Связанные переменные можно переименовывать.

α-ЭКВИВАЛЕНТНЫЕ λ-ТЕРМЫ

$$I = \lambda x.x = \lambda y.y = \lambda f.f$$

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, $(+ 3)$ 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):

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

Частичного применения ко второму аргументу (без первого) нет!



Комбинаторная логика

Комбинаторы можно определить через их поведение на аргументах:

- $I \ x = x$
- $K \ x \ y = x$
- $\omega \ x = x \ x$
- $S \ x \ y \ z = x \ z \ (y \ z)$

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



Упражнение

λ-исчисление

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

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



Упражнение

λ -исчисление

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

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



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

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

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



Аксиомы λ -исчисления

Основная аксиома β -конверсии

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

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 $\text{---} \lambda \vdash M = N$.



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 $y \in FV(N)$?

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



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:

$$\begin{aligned} &\lambda z.((\lambda z.(y \ z \ x))[y := z])[x := y]) \\ &\lambda z.((\lambda w.((y \ z \ x)[z := w])[y := z]))[x := y]) \\ &\lambda z.((\lambda w.(y \ w \ x)[y := z]))[x := y]) \\ &\lambda z.((\lambda w.(z \ w \ x))[x := y]) \\ &\lambda z.(\lambda w.(z \ w \ y)) \end{aligned}$$



Применение α-конверсии

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

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



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 (экстенциональность 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 (экстенциональность 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 (экстенциональность 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 (экстенциональность равенства).

Примеры

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

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



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

$$\begin{array}{c} (\lambda x. x \ x) (\lambda y \ z. y \ z) \\ \downarrow \\ x \mapsto \lambda y \ z. y \ z \\ \downarrow \\ (\lambda y \ z. y \ z) (\lambda y \ z. y \ z) \end{array}$$



Пример конверсии

$$\begin{array}{c} (\lambda x. x \ x) \ (\lambda y \ z. y \ z) \\ \downarrow \\ x \mapsto \lambda y \ z. y \ z \\ \downarrow \\ (\lambda y. (\lambda z. y \ z)) \ (\lambda y \ z. y \ z) \\ \downarrow \\ y \mapsto \lambda y \ z. y \ z \\ \downarrow \\ \lambda z. (\lambda y \ z. y \ z) \ z \end{array}$$



Пример конверсии

$$\begin{aligned} & (\lambda x. x \ x) (\lambda y \ z. y \ z) \\ & \quad \downarrow x \mapsto \lambda y \ z. y \ z \\ & (\lambda y. (\lambda z. y \ z)) (\lambda y \ z. y \ z) \\ & \quad \downarrow y \mapsto \lambda y \ z. y \ z \\ & \lambda z. (\lambda y \ z. y \ z) \ z \\ & \quad \downarrow \alpha\text{-преобразование} \\ & \lambda z. (\lambda y \ z'. y \ z') \ z \end{aligned}$$

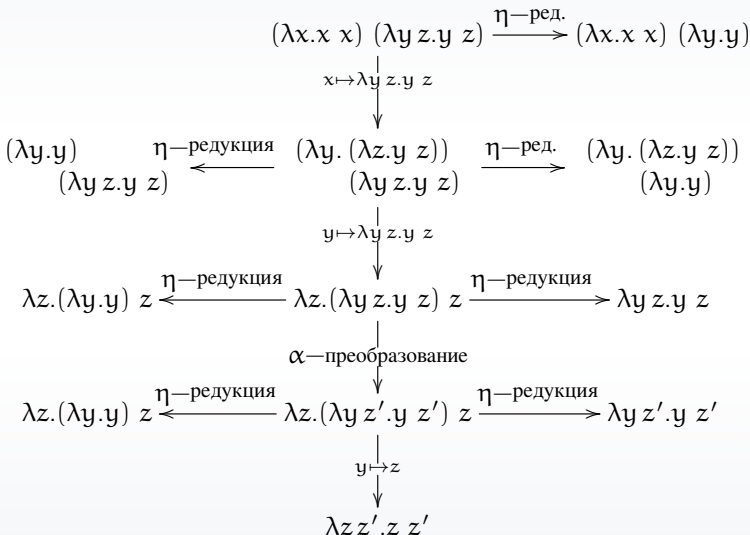


Пример конверсии

$$\begin{aligned} & (\lambda x. x \ x) (\lambda y \ z. y \ z) \\ & \quad \downarrow x \mapsto \lambda y \ z. y \ z \\ & (\lambda y. (\lambda z. y \ z)) (\lambda y \ z. y \ z) \\ & \quad \downarrow y \mapsto \lambda y \ z. y \ z \\ & \lambda z. (\lambda y \ z. y \ z) \ z \\ & \quad \downarrow \alpha\text{-преобразование} \\ & \lambda z. (\lambda y. (\lambda z'. y \ z')) \ z \\ & \quad \downarrow y \mapsto z \\ & \lambda z \ z'. z \ z' \end{aligned}$$



Пример конверсии





Применение η -конверсии

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

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



Применение η -конверсии

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

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

Задачи

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



Скобочная абстракция

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

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

Таким образом удаётся перейти к бесточечному представлению λ -функции. В частности, это то, чего мы добиваемся, когда переносим зависимости!



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: $\lambda x. \mathbf{S} (\lambda y. y) (\lambda y. x) \rightarrow \lambda x. \mathbf{S} \mathbf{I} (\mathbf{K} x) \rightarrow \mathbf{S} (\lambda x. \mathbf{S} \mathbf{I}) (\lambda x. \mathbf{K} x) \rightarrow \mathbf{S} (\mathbf{K}(\mathbf{S} \mathbf{I})) (\mathbf{S} (\lambda x. \mathbf{K}) (\lambda x. x)) \rightarrow \mathbf{S} (\mathbf{K}(\mathbf{S} \mathbf{I})) (\mathbf{S} (\mathbf{K} \mathbf{K}) \mathbf{I})$.



Скобочная абстракция

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

- По алгоритму: $\lambda x. S (\lambda y. y) (\lambda y. x) \rightarrow \lambda x. S I (Kx) \rightarrow S (\lambda x. SI) (\lambda x. Kx) \rightarrow S (K(SI))(S (\lambda x. K) (\lambda x. x)) \rightarrow S (K(SI))(S (KK) I)$.
- 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 **K** линейна \Rightarrow внешняя функция точно **S**. А ей нужен ещё хотя бы один аргумент-комбинатор, кроме x и y . Пишем заглушку: $S M_1$.



Скобочная абстракция

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

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



Скобочная абстракция

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

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



Скобочная абстракция

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

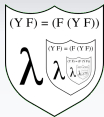
- $\lambda x y. y \ x$ меняет местами переменные, а **K** линейна \Rightarrow внешняя функция точно **S**. А ей нужен ещё хотя бы один аргумент-комбинатор, кроме x и y . Пишем заглушку: $S \ M_1$.
- Тестируем: $S \ M_1 \ x \ y = M_1 \ y \ (x \ y)$. Это плохо: из $(x \ y)$ нельзя извлечь x . Нужен ещё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



Скобочная абстракция

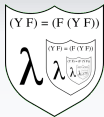
Перейдём к комбинаторной версии `flip id`: $\lambda x y. y x$.

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



Подытожим

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



Редукция

Редексы

Определение

Терм $(\lambda x. M[x]) N$ — **редекс**.

Замена редекса на $M[x := N]$ — **сокращение редекса**.



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

$M \rightarrow_{\beta} N$ определяется следующим образом:

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



β -редукция

Определение

- β -редукция — транзитивное рефлексивное замыкание \rightarrow_{β} .
- 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 $\lambda x_1, \dots, x_n. x_i Q$, где Q произвольно (в том числе может содержать редексы), называются термами в головной нормальной форме.

- Если для термина T выполняется условие:
 $\exists N_1 \dots N_k (T N_1 \dots N_k = I)$, он называется разрешимым.
- Терм разрешим \Leftrightarrow у него существует головная нормальная форма (пример — Y). (доказательство в сторону \leftarrow : 1 балл)

Неразрешимые термы (вроде Ω) понимаются как всегда закливающиеся и условно отождествляются друг с другом. Разрешимые термы без нормальной формы — частично определенные функции.



Противоречивость λ -исчисления

Чистое (без логических операторов) λ -исчисление непротиворечиво.

Пример

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

А как насчет $K = I$?



Альтернативные нумералы

Положим $\underline{0} = \lambda x. x$, $\underline{1} = \lambda x. \lambda x. x$, $\underline{N} = \lambda x. (\underline{N} - \underline{1})$.

Как они выглядят в комбинаторной логике? Как выразить $\text{Pred}(\underline{N})$?