Редукция и неподвижная точка

А.Н. Непейвода

2023 г.

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?

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

- Терм разрешим $\Leftrightarrow$ у него существует головная нормальная форма (пример — Y).

Неразрешимые термы (вроде $(\lambda x.x\ x)\ (\lambda x.x\ x)$) понимаются как всегда зацикливающиеся 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

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

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

Понятие неподвижной точки

Рассмотрим понятие из математики $a : f(a) = a$. Можно понимать как стабилизацию ($f(f(...(f(a)))) = a$), а можно как доступ к развёртке ($a \to f(a) \to f(f(a)) \to \ldots$).

Понятие неподвижной точки

Рассмотрим понятие из математики $a : f(a) = a$. Можно понимать как стабилизацию ($f(f(...(f(a)))) = a$), а можно как доступ к развёртке ($a \to f(a) \to f(f(a)) \to \ldots$).

Неформальное утверждение

Если понимать функции как возможно зацикливающиеся (на расширенной области определения, содержащей $\perp$), тогда у каждой функции есть НТ.

Понятие неподвижной точки

Рассмотрим понятие из математики $a : f(a) = a$. Можно понимать как стабилизацию $(f(f(...(f(a)))) = a)$, а можно как доступ к развёртке $(a \to f(a) \to f(f(a)) \to \dots)$.

Неформальное утверждение

Если понимать функции как возможно зацикливающиеся (на расширенной области определения, содержащей $\perp$), тогда у каждой функции есть НТ.

Доказательство: Пусть значение функции f зависит от её аргумента x. Возьмём в качестве x зацикливание ($\perp$). Тогда $f(\perp)$ тоже зациклится, значит, $\perp$ неподвижная точка.

Теперь пусть значение f от x не зависит. Тогда это константа k. Имеем $f(k) = k$.

Неподвижная точка

Дан λ-терм F. Как найти такой λ-терм A, что $F\ A = A$?

Неподвижная точка

Дан λ-терм F. Как найти такой λ-терм A, что $F\ A = A$?

Рассмотрим $W = \lambda x.F\ (x\ x)$, $X = W\ W$.

Неподвижная точка

Дан λ-терм F. Как найти такой λ-терм A, что $F\ A = A$?

Рассмотрим $W = \lambda x.F\ (x\ x)$, $X = W\ W$.

Тогда $X = W\ W = (\lambda x.F\ (x\ x))\ W = F\ (W\ W) = F\ X$.

Неподвижная точка

Дан λ-терм F. Как найти такой λ-терм A, что $F\ A = A$?

Теорема

У каждого λ-терма существует неподвижная точка.

Д-во: Рассмотрим $W = \lambda x.F\ (x\ x)$, $X = W\ W$.

Тогда $X = W\ W = (\lambda x.F\ (x\ x))\ W = F\ (W\ W) = F\ X$.

Чему равны неподвижные точки следующих λ-термов?

- $\lambda x\,y.y$
- $\lambda x.x$
- $\lambda x\,y.x$

Комбинатор неподвижной точки

Как найти неподвижную точку произвольного терма F?

$F = \lambda x.M[x]$

Комбинатор неподвижной точки

Как найти неподвижную точку произвольного терма F?

$F = \lambda x.M[x]$

Пробная попытка: $(F\ F) = M[\lambda x.M[x]]$

Комбинатор неподвижной точки

Как найти неподвижную точку произвольного терма F?

$F = \lambda x.M[x]$

Пробная попытка: $(F\ F) = M[\lambda x.M[x]]$

Вместо x подставлена λ-абстракция $\to$ нужно добавить для нее дополнительный аргумент.
Положим $M' = \lambda F.(\lambda x.M[x])\ (F\ F)$. И добавим абстракцию по M.

$Y = \lambda m.(\lambda f.m\ (f\ f))\ (\lambda f.m\ (f\ f))$

Работает?

Комбинатор неподвижной точки

Как найти неподвижную точку произвольного терма F?

$F = \lambda x.M[x]$

Пробная попытка: $(F\ F) = M[\lambda x.M[x]]$

Вместо x подставлена λ-абстракция $\rightarrow$ нужно добавить для нее дополнительный аргумент.
Положим $M' = \lambda F.(\lambda x.M[x])\ (F\ F)$. И добавим абстракцию по M.

$Y = \lambda m.(\lambda f.m\ (f\ f))\ (\lambda f.m\ (f\ f))$

Работает?

$Y\ F = (\lambda z.F\ (z\ z))\ \lambda z.F\ (z\ z) = F(\underline{(\lambda z.F(z\ z))\ \lambda z.F(z\ z)}) = F\ (Y\ F)$

Пример применения

Рекурсивное определение факториала:

$(F\ x) = (\mathbf{IF}\ (\mathbf{IS0}\ x)\ 1\ (\mathbf{MUL}\ x\ (F\ (\mathbf{PRED}\ x))))$.

Вернемся к числам и стандартным операциям.

```
f x = if x == 0 then 1 else x * (f (x - 1))
```

```
fact x =
    if x == 0 then 1 else x * (fact (x - 1))
```

Пример применения

Рекурсивное определение факториала:

$(F\ x) = (\mathbf{IF}\ (\mathbf{IS0}\ x)\ 1\ (\mathbf{MUL}\ x\ (F\ (\mathbf{PRED}\ x))))$.

Вернемся к числам и стандартным операциям.

`f x = if x == 0 then 1 else x * (f (x - 1))`

```
fact x =
    if x == 0 then 1 else x * (fact (x - 1))
```

$F = \lambda$ `f.`λ `x.(if x == 0 then 1 else (x * (f (x - 1))).`

$Y\ F\ 2 = F\ (Y\ F)\ 2$ `= (if false then 1 else (2 * (`$Y\ F$ `1)))`
`= 2 * (if false then 1 else (1 * (`$Y\ F$ `0)))`
`= 2 * (1 * (if true then 1 else (`$Y\ F$ `...))) = 2 * (1 * 1)`

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.

Контекст вычислений

Контекст — окружение λ-терма (терм с дырой).
$C ::= [\,] \mid t\ C \mid C\ t \mid \lambda x.C$
Контексты вычислений — подмножество контекстов, зависящее от порядка редукции.

Контекст вычислений

Контекст — окружение λ-терма (терм с дырой).
$C \ ::= \ [\,] \mid t\ C \mid C\ t \mid \lambda x.C$
Контексты вычислений — подмножество контекстов, зависящее от порядка редукции.

Call-by-value

v — переменная или абстракция.
Контекст $E \ ::= \ [\,] \mid E\ e \mid v\ E$
Правила редукции:

$$\frac{e_1 \twoheadrightarrow_\beta e_2}{E[e_1] \twoheadrightarrow_\beta E[e_2]} \qquad \frac{}{(\lambda x.e)\ v \twoheadrightarrow_\beta e\{x := v\}}$$

Контекст вычислений

Контекст — окружение λ-терма (терм с дырой).
$C \ ::= \ [\,] \mid t\ C \mid C\ t \mid \lambda x.C$
Контексты вычислений — подмножество контекстов, зависящее от порядка редукции.

Call-by-name

Контекст $E \ ::= \ [\,] \mid E\ e$
Правила редукции:

$$\frac{e_1 \twoheadrightarrow_\beta e_2}{E[e_1] \twoheadrightarrow_\beta E[e_2]} \qquad \frac{}{(\lambda x.e_1)\ e_2 \twoheadrightarrow_\beta e_1\{x := e_2\}}$$

Лемма о генеричности

Лемма

Пусть M, N — термы λ-исчисления, причем M неразрешим, а N имеет н.ф. Тогда для любого контекста $C[\ ]$

$$C[M] =_\beta N \Rightarrow \forall L(C[L] =_\beta N)$$

Лемма о генеричности

Лемма

Пусть M, N — термы λ-исчисления, причем M неразрешим, а N имеет н.ф. Тогда для любого контекста $C[\ ]$

$$C[M] =_\beta N \Rightarrow \forall L (C[L] =_\beta N)$$

Неформально: чтобы отличить терм от других, его нужно (частично) вычислить.

Следствие леммы о генеричности

Возможно ли сравнивать *текстовые представления термов в чистом λ-исчислении?*

Следствие леммы о генеричности

Возможно ли сравнивать *текстовые представления термов в чистом λ-исчислении?*

Пусть существует комбинатор $\lambda x\, y.\mathsf{E}$, который возвращает T, если буквальные представления x и y совпадают (в каком-либо смысле), и F иначе.

Тогда $\mathsf{E}\ \mathbf{\Omega}\ \mathbf{I} = \mathsf{F}$, $\mathsf{E}\ \mathbf{I}\ \mathbf{I} = \mathsf{T}$, но лемма о генеричности влечет, что $\forall \mathsf{N}(\mathsf{E}\ \mathsf{N}\ \mathbf{I}) = \mathsf{F}$.

Следовательно, E не определим в чистом λ-исчислении.

Другие применения

С помощью леммы о генеричности можно доказывать невозможность реализации комбинатора, определённого логической формулой.

Предположим, существует такой M, что $\forall x, y(M\ (x\ y) = y)$. По определению, $M\ (\mathbf{\Omega}\ \mathbf{I}) = \mathbf{I}$, значит, по лемме о генеричности $\forall x(M\ (x\ \mathbf{I})) = \mathbf{I}$. В частности, $M\ (\mathbf{K}\ \mathbf{I}) = \mathbf{I}$. С другой стороны, $\mathbf{K}\ \mathbf{I} = \mathbf{I}\ (\mathbf{K}\ \mathbf{I})$, и по определению M получаем $M\ (\mathbf{I}\ (\mathbf{K}\ \mathbf{I})) = \mathbf{K}\ \mathbf{I}$, что невозможно.

Данную задачу можно было решить и без применения леммы о генеричности, используя сходные идеи (это подсказка к ДЗ от 18.09).

Вариант Y-комбинатора

$$Y' = \lambda f.(\lambda t.(t\ t))\ \lambda x.f\ (\lambda y.(x\ x)\ y)$$

```
Y1 =
  \f -> (\t -> (t t))
      (\x -> f (\y -> x x y))
```

Редукция может быть как конечной, так и не завершающейся. Как верифицировать корректное применение операторов, подобных Y?

Возникновение понятия типа

Изначально возник в трудах Б.Рассела, который заметил, что в наивной теории множеств существует парадокс:

Парадокс Рассела

$$\Omega = \{A \mid A \notin A\} \Rightarrow (\Omega \in \Omega \Leftrightarrow \Omega \notin \Omega)$$

Понятие типа ограничивает возможные операции над его сущностями $\Rightarrow$ исключает парадоксы (неверное поведение программ).

Определение типа

Система типов — гибко управляемый синтаксический метод доказательства отсутствия в программе определенных видов поведения при помощи классификации выражений языка по разновидностям вычисляемых ими значений.

Б.Пирс

В λ-исчислении типы — синтаксические конструкции, приписываемые термам по определенным правилам:

$$M : \sigma$$

Свойства типизации

- Статические vs динамические
- Явные vs неявные
- Сильные vs слабые

Сильные статические типы в функциональном языке ограничивают его синтаксические конструкции, но позволяют строить программы, которые уже частично верифицированы с помощью проверки типов.

Простые типы λ-исчисления

Определение

Множество типов $\mathbb{T}$ в $\lambda_{\rightarrow}$ определяется индуктивно.

- Переменные типа (α, β, γ etc) принадлежат $\mathbb{T}$.
- Если $\sigma \in \mathbb{T}$, $\phi \in \mathbb{T}$, то $(\sigma \to \phi) \in \mathbb{T}$.

Стрелка $\to$ правоассоциативна: $\sigma_1 \to \sigma_2 \to \ldots \sigma_n$ — сокращение для $\sigma_1 \to (\sigma_2 \to (\ldots \sigma_n) \ldots))$.
В силу наличия частичных вычислений тип $\sigma_1 \to \sigma_2 \ldots \to \sigma_n$ можно понимать как тип функции с $n-1$ аргументами типов $\sigma_1, \ldots, \sigma_{n-1}$ и типом результата σ_n; а можно как тип функции с одним аргументом σ_1 и типом результата — функцией с типом $\sigma_2 \to \ldots \to \sigma_n$ (а также все промежуточные варианты).

Полуформальный алгоритм 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 $\sigma \to \tau$.

Полуформальный алгоритм 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 $\perp$.

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 $\perp$.

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*)