

# Функциональное программирование

## Лекция 12. Алгоритм вывода типов

Денис Николаевич Москвин

СПбАУ РАН, CSC

30.04.2015

- 1 Главный тип
- 2 Подстановка типа и унификация
- 3 Теорема Хиндли-Милнера
- 4 К практике

- 1 **Главный тип**
- 2 Подстановка типа и унификация
- 3 Теорема Хиндли-Милнера
- 4 К практике

# Система $\lambda \rightarrow$ а ля Карри

<p>Предтермы</p> $\Lambda ::= \quad V$ $\quad   \quad MN$ $\quad   \quad \lambda x. M$	<p>Редукция</p> $(\lambda x. M) N \rightarrow_{\beta} M[x := N]$
<p>Типы</p> $\mathbb{T} ::= \quad \mathbb{V}$ $\quad   \quad \sigma \rightarrow \tau$ <hr/> <p>Контексты</p> $\Gamma ::= \quad \emptyset$ $\quad   \quad \Gamma, x:\sigma$	<p>Типизация</p> $\frac{x:\sigma \in \Gamma}{\Gamma \vdash x:\sigma}$ $\frac{\Gamma \vdash M:\sigma \rightarrow \tau \quad \Gamma \vdash N:\sigma}{\Gamma \vdash (MN):\tau}$ $\frac{\Gamma, x:\sigma \vdash M:\tau}{\Gamma \vdash (\lambda x. M):\sigma \rightarrow \tau}$

Здесь  $V = \{a, b, \dots\}$ ,  $\mathbb{V} = \{\alpha, \beta, \dots\}$  и  $x \in V$ ;  $M, N \in \Lambda$ ;  $\sigma, \tau \in \mathbb{T}$ .

# Система $\lambda \rightarrow$ а ля Чёрч

<p>Предтермы</p> $\Lambda_{\mathbb{T}} ::= \begin{array}{l} V \\   MN \\   \lambda x^{\sigma}. M \end{array}$	<p>Редукция</p> $(\lambda x^{\sigma}. M) N \rightarrow_{\beta} M[x := N]$
<p>Типы</p> $\mathbb{T} ::= \begin{array}{l} V \\   \sigma \rightarrow \tau \end{array}$ <hr/> <p>Контексты</p> $\Gamma ::= \begin{array}{l} \emptyset \\   \Gamma, x:\sigma \end{array}$	<p>Типизация</p> $\frac{x:\sigma \in \Gamma}{\Gamma \vdash x:\sigma}$ $\frac{\Gamma \vdash M:\sigma \rightarrow \tau \quad \Gamma \vdash N:\sigma}{\Gamma \vdash (MN):\tau}$ $\frac{\Gamma, x:\sigma \vdash M:\tau}{\Gamma \vdash (\lambda x^{\sigma}. M):\sigma \rightarrow \tau}$

Здесь  $V = \{a, b, \dots\}$ ,  $\mathbb{V} = \{\alpha, \beta, \dots\}$  и  $x \in V$ ;  $M, N \in \Lambda_{\mathbb{T}}$ ;  $\sigma, \tau \in \mathbb{T}$ .

# Главный тип (principle type)

- В версии Чёрча  $\lambda \rightarrow$  термы атрибутированы типами, поэтому тип терма единственен. Для

$$S_{\sigma\tau\rho} \equiv \lambda f^{\sigma \rightarrow \tau \rightarrow \rho} g^{\sigma \rightarrow \tau} z^{\sigma}. f z (g z):$$

$$S_{\sigma\tau\rho} : (\sigma \rightarrow \tau \rightarrow \rho) \rightarrow (\sigma \rightarrow \tau) \rightarrow \sigma \rightarrow \rho$$

$$S_{\sigma\tau\sigma} : (\sigma \rightarrow \tau \rightarrow \sigma) \rightarrow (\sigma \rightarrow \tau) \rightarrow \sigma \rightarrow \sigma$$

$$S_{(\tau \rightarrow \rho)\tau\rho} : ((\tau \rightarrow \rho) \rightarrow \tau \rightarrow \rho) \rightarrow ((\tau \rightarrow \rho) \rightarrow \tau) \rightarrow (\tau \rightarrow \rho) \rightarrow \rho$$

- Любой из этих типов можно приписать терму  $S \equiv \lambda f g z. f z (g z)$  в версии Карри.
- Однако, первый «лучше» в том смысле, что остальные получаются из него подстановкой типа вместо типовой переменной.

# Вывод главного типа (пример)

$$\lambda x y. y (\lambda z. y x) \qquad \lambda x^\alpha y^\beta. \underbrace{y^\beta (\lambda z^\gamma. \overbrace{y^\beta x^\alpha}^\delta)}_\varepsilon$$

- 1 Присвоим типовую (мета-)переменную всем термовым переменным:  $x^\alpha, y^\beta, z^\gamma$ .
- 2 Присвоим типовую переменную всем *аппликативным* подтермам:  $(y x) : \delta, (y (\lambda z. y x)) : \varepsilon$ .
- 3 Выпишем уравнения (ограничения) на типы, необходимые для типизируемости терма:  $\beta \sim \alpha \rightarrow \delta, \beta \sim (\gamma \rightarrow \delta) \rightarrow \varepsilon$ .
- 4 Найдём *главный унификатор* для типовых переменных (подстановку), дающий решения уравнений:  
 $\alpha := \gamma \rightarrow \delta, \beta := (\gamma \rightarrow \delta) \rightarrow \varepsilon, \delta := \varepsilon$ .
- 5 Главный тип  $(\lambda x y. y (\lambda z. y x)) : (\gamma \rightarrow \varepsilon) \rightarrow ((\gamma \rightarrow \varepsilon) \rightarrow \varepsilon) \rightarrow \varepsilon$ .

- 1 Главный тип
- 2 Подстановка типа и унификация
- 3 Теорема Хиндли-Милнера
- 4 К практике



## Определение

**Подстановка типа** — это операция  $S: \mathbb{T} \rightarrow \mathbb{T}$ , такая что

$$S(\sigma \rightarrow \tau) \equiv S(\sigma) \rightarrow S(\tau)$$

- Обычно подстановка тождественна на всех типовых переменных, кроме конечного носителя  $\text{sup}(S) = \{\alpha \mid S(\alpha) \neq \alpha\}$ .
- Пример подстановки  $S = [\alpha := \gamma \rightarrow \beta, \beta := \alpha \rightarrow \gamma]$ .
- Тождественную подстановку (с пустым носителем) обозначают  $[]$ .
- Подстановка выполняется *параллельно*; для  $\tau = \alpha \rightarrow \beta \rightarrow \gamma$

$$\begin{aligned} S(\tau) &= [\alpha := \gamma \rightarrow \beta, \beta := \alpha \rightarrow \gamma](\alpha \rightarrow \beta \rightarrow \gamma) \\ &= (\gamma \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma) \rightarrow \gamma \end{aligned}$$

## Определение

**Композиция двух подстановок** — подстановка с носителем, являющимся объединением носителей, над которым последовательно выполнены обе подстановки.

Для

$$S = [\alpha := \gamma \rightarrow \beta, \beta := \alpha \rightarrow \gamma];$$

$$T = [\alpha := \beta \rightarrow \gamma, \gamma := \beta]$$

$$T \circ S = [\alpha := T(S(\alpha)), \beta := T(S(\beta)), \gamma := T(S(\gamma))], \text{ то есть}$$

$$T \circ S = [\alpha := \beta \rightarrow \beta, \beta := (\beta \rightarrow \gamma) \rightarrow \beta, \gamma := \beta]$$

Подстановки образуют моноид относительно  $\circ$  с единицей  $[\ ]$ .  
(проверьте этот факт самостоятельно)

## Определение

**Унификатор** для типов  $\sigma$  и  $\tau$  — это подстановка  $S$ , такая что  $S(\sigma) \equiv S(\tau)$ .

## Пример

Пусть  $\sigma = \alpha \rightarrow \beta \rightarrow \gamma$  и  $\tau = (\delta \rightarrow \epsilon) \rightarrow \zeta$ .

Их унификатор

$$S = [\alpha := \delta \rightarrow \epsilon, \zeta := \beta \rightarrow \gamma]$$

Действительно, в результате этой подстановки получаем и из  $\tau$  и из  $\sigma$  один и тот же тип

$$S(\sigma) \equiv S(\tau) \equiv (\delta \rightarrow \epsilon) \rightarrow \beta \rightarrow \gamma$$

Попробуем унифицировать типы

$$\sigma = \alpha \rightarrow \beta \rightarrow \alpha$$

$$\tau = \gamma \rightarrow \delta$$

Для построения унификатора нужно соединить подстановки  $[\alpha := \gamma]$  и  $[\delta := \beta \rightarrow \alpha]$ .

$$S' = [\alpha := \gamma] \circ [\delta := \beta \rightarrow \alpha] = [\alpha := \gamma, \delta := \beta \rightarrow \gamma]$$

$$S'' = [\delta := \beta \rightarrow \alpha] \circ [\alpha := \gamma] = [\delta := \beta \rightarrow \alpha, \alpha := \gamma]$$

Одна из подстановок — унификатор, другая — нет:

$$S'(\sigma) \equiv \gamma \rightarrow \beta \rightarrow \gamma \qquad S'(\tau) \equiv \gamma \rightarrow \beta \rightarrow \gamma$$

$$S''(\sigma) \equiv \gamma \rightarrow \beta \rightarrow \gamma \qquad S''(\tau) \equiv \gamma \rightarrow \beta \rightarrow \alpha$$

Мораль: выделив одну из элементарных подстановок, следует тут же выполнить ее повсюду.

## Определение

Унификатор  $S$  — это *главный унификатор* для  $\sigma$  и  $\tau$ , если для любого другого унификатора  $S'$  существует подстановка  $T$ , такая что

$$S' \equiv T \circ S$$

## Пример

Для  $\sigma = \alpha \rightarrow \beta \rightarrow \alpha$  и  $\tau = \gamma \rightarrow \delta$  главный унификатор

$$S = [\alpha := \gamma, \delta := \beta \rightarrow \gamma]$$

$$S' = [\alpha := \gamma, \beta := \varepsilon \rightarrow \varepsilon, \delta := (\varepsilon \rightarrow \varepsilon) \rightarrow \gamma]$$

$$S' = [\beta := \varepsilon \rightarrow \varepsilon] \circ S$$

## Теорема унификации (Робинсон, 1965)

Существует алгоритм унификации  $\mathcal{U}$ , который для заданных типов  $\sigma$  и  $\tau$  возвращает:

- главный унификатор  $S$  для  $\sigma$  и  $\tau$ , если  $\sigma$  и  $\tau$  могут быть унифицированы;
  - сообщение об ошибке в противном случае.
- 
- Алгоритм  $\mathcal{U}(\sigma, \tau)$  позволяет искать «минимальное» решение уравнения на типы  $\sigma \sim \tau$ .
  - Ключевой момент всех рассуждений про унификацию:

$$\sigma_1 \rightarrow \sigma_2 \equiv \tau_1 \rightarrow \tau_2 \Leftrightarrow \sigma_1 \equiv \tau_1 \wedge \sigma_2 \equiv \tau_2$$

## Алгоритм унификации $\mathcal{U}$

$$\begin{aligned}\mathcal{U}(\alpha, \alpha) &= [] \\ \mathcal{U}(\alpha, \tau) \mid \alpha \in FV(\tau) &= \text{ошибка} \\ \mathcal{U}(\alpha, \tau) \mid \alpha \notin FV(\tau) &= [\alpha := \tau] \\ \mathcal{U}(\sigma_1 \rightarrow \sigma_2, \alpha) &= \mathcal{U}(\alpha, \sigma_1 \rightarrow \sigma_2) \\ \mathcal{U}(\sigma_1 \rightarrow \sigma_2, \tau_1 \rightarrow \tau_2) &= \mathcal{U}(\mathcal{U}(\sigma_2, \tau_2)\sigma_1, \mathcal{U}(\sigma_2, \tau_2)\tau_1) \circ \mathcal{U}(\sigma_2, \tau_2)\end{aligned}$$

- $\mathcal{U}(\sigma, \tau)$  завершается. Деревья типа конечны и количество типовых переменных сокращается на 1 через конечное число шагов.
- $\mathcal{U}(\sigma, \tau)$  унифицирует. По индукции; используем, что если  $S$  унифицирует  $(\sigma, \tau)$ , то  $S \circ [\alpha := \rho]$  унифицирует  $(\sigma \rightarrow \alpha, \tau \rightarrow \rho)$ .
- $\mathcal{U}(\sigma, \tau)$  даёт главный унификатор. По индукции; см. TAPL (глава 22.4) [Pie02] или LCwT (глава 4.4) [Bar92].

# Алгоритм унификации $\mathcal{U}$ : пример

$$\begin{aligned}\mathcal{U}(\alpha, \alpha) &= [] \\ \mathcal{U}(\alpha, \tau) \mid \alpha \in FV(\tau) &= \text{ошибка} \\ \mathcal{U}(\alpha, \tau) \mid \alpha \notin FV(\tau) &= [\alpha := \tau] \\ \mathcal{U}(\sigma_1 \rightarrow \sigma_2, \alpha) &= \mathcal{U}(\alpha, \sigma_1 \rightarrow \sigma_2) \\ \mathcal{U}(\sigma_1 \rightarrow \sigma_2, \tau_1 \rightarrow \tau_2) &= \mathcal{U}(\mathcal{U}(\sigma_2, \tau_2)\sigma_1, \mathcal{U}(\sigma_2, \tau_2)\tau_1) \circ \mathcal{U}(\sigma_2, \tau_2)\end{aligned}$$

Для  $\lambda x y. y (\lambda z. y x)$  система уравнений на типы имела вид  $E = \{\beta \sim (\gamma \rightarrow \delta) \rightarrow \varepsilon, \beta \sim \alpha \rightarrow \delta\}$ . Алгоритм  $\mathcal{U}$  даёт:

$$\begin{aligned}\mathcal{U}(E) &= \mathcal{U}(\beta \rightarrow \beta, ((\gamma \rightarrow \delta) \rightarrow \varepsilon) \rightarrow (\alpha \rightarrow \delta)) \\ &= \mathcal{U}(\mathcal{U}(\beta, \alpha \rightarrow \delta)\beta, \mathcal{U}(\beta, \alpha \rightarrow \delta)(\gamma \rightarrow \delta) \rightarrow \varepsilon) \circ \mathcal{U}(\beta, \alpha \rightarrow \delta) \\ &= \mathcal{U}(\alpha \rightarrow \delta, (\gamma \rightarrow \delta) \rightarrow \varepsilon) \circ [\beta := \alpha \rightarrow \delta] \\ &= [\alpha := \gamma \rightarrow \varepsilon] \circ [\delta := \varepsilon] \circ [\beta := \alpha \rightarrow \delta] \\ &= [\alpha := \gamma \rightarrow \varepsilon, \delta := \varepsilon, \beta := (\gamma \rightarrow \varepsilon) \rightarrow \varepsilon]\end{aligned}$$



- Проследите за изменениями в работе алгоритма Ц, при перестановке элементов в E:  
 $E = \{\beta \sim \alpha \rightarrow \delta, \beta \sim (\gamma \rightarrow \delta) \rightarrow \varepsilon\}$
- Изменится ли что-то в этом случае, и, если изменится, то что?

- 1 Главный тип
- 2 Подстановка типа и унификация
- 3 Теорема Хиндли-Милнера**
- 4 К практике

# Теорема о существовании системы ограничений

- Наша первая цель — построить систему ограничений на типы для терма  $M$  (возможно незамкнутого).
- Для типизации таких термов необходим контекст  $\Gamma$ , в котором объявляются типы всех свободных переменных.
- Для подстановки  $S$ , унифицирующей систему уравнений на типы  $E = \{\sigma_1 \sim \tau_1, \dots, \sigma_n \sim \tau_n\}$ , введём обозначение  $S \vDash E$ .

## Теорема о существовании системы ограничений

Для любых терма  $M \in \mathcal{L}$ , контекста  $\Gamma$  ( $FV(M) \subseteq \text{dom}(\Gamma)$ ) и типа  $\sigma \in \mathbb{T}$  существует конечное множество уравнений на типы  $E = E(\Gamma, M, \sigma)$ , такое что для некоторой подстановки  $S$ :

- $S \vDash E(\Gamma, M, \sigma) \Rightarrow S(\Gamma) \vdash M : S(\sigma)$ ;
- $S(\Gamma) \vdash M : S(\sigma) \Rightarrow S' \vDash E(\Gamma, M, \sigma)$ , для некоторой  $S'$ , имеющего тот же эффект, что и  $S$ , на типовых переменных в  $\Gamma$  и  $\sigma$ .

## Алгоритм построения системы ограничений E

$$E(\Gamma, x, \sigma) = \{\sigma \sim \Gamma(x)\}$$

$$E(\Gamma, M N, \sigma) = E(\Gamma, M, \alpha \rightarrow \sigma) \cup E(\Gamma, N, \alpha)$$

$$E(\Gamma, \lambda x. M, \sigma) = E(\Gamma \cup \{x:\alpha\}, M, \beta) \cup \{\alpha \rightarrow \beta \sim \sigma\}$$

- В первом равенстве контекст  $\Gamma$  рассматривается как функция из множества переменных в множество типов.
- Переменные  $\alpha$  во втором и третьем равенствах и  $\beta$  в третьем всякий раз должны быть «свежими»!
- Самостоятельно постройте системы ограничений для следующих троек  $(\Gamma, M, \sigma)$

$$E(x:\gamma \rightarrow \delta, x, \alpha \rightarrow \beta \rightarrow \alpha) = ???$$

$$E(x:\gamma \rightarrow \delta, x x, \alpha \rightarrow \beta \rightarrow \alpha) = ???$$

$$E(x:\gamma \rightarrow \delta, \lambda x. x, \alpha \rightarrow \beta \rightarrow \alpha) = ???$$

# Главная пара (Principle Pair)

## Определение

Для  $M \in \Lambda$  *главной парой* называют пару  $(\Gamma, \sigma)$ , такую что

- $\Gamma \vdash M:\sigma$
- $\Gamma' \vdash M:\sigma' \Rightarrow \exists S [S(\Gamma) \subseteq \Gamma' \wedge S(\sigma) \equiv \sigma']$

## Пример

Для  $M = \lambda x. x y$  имеем

$$PP(M) = (y:\alpha, (\alpha \rightarrow \beta) \rightarrow \beta)$$

$$y:\alpha \vdash (\lambda x. x y):(\alpha \rightarrow \beta) \rightarrow \beta$$

## Теорема Хиндли – Милнера

Существует алгоритм  $PP$ , возвращающий для  $M \in \Lambda$

- главную пару  $(\Gamma, \sigma)$ , если  $M$  имеет тип;
- сообщение об ошибке в противном случае.

Пусть  $FV(M) = \{x_1, \dots, x_n\}$ ,  $\Gamma_0 = \{x_1:\alpha_1, \dots, x_n:\alpha_n\}$  и  $\sigma_0 = \beta$ .

## Алгоритм $PP$

$$\begin{aligned} PP(M) \mid \mathcal{U}(E(\Gamma_0, M, \sigma_0)) \equiv \text{ошибка} &= \text{ошибка} \\ PP(M) \mid \mathcal{U}(E(\Gamma_0, M, \sigma_0)) \equiv S &= (S(\Gamma_0), S(\sigma_0)) \end{aligned}$$

Стартуем с произвольных переменных типа, приписанных свободным переменным типизируемого термина  $M$  и всему терму.

## Определение

Для  $M \in \Lambda^0$  **главным типом** называют тип  $\sigma$ , такой что

- $\vdash M:\sigma$
- $\vdash M:\sigma' \Rightarrow \exists S [S(\sigma) \equiv \sigma']$

## Следствие теоремы Хиндли – Милнера

Существует алгоритм РТ, возвращающий для  $M \in \Lambda^0$

- главный тип  $\sigma$ , если  $M$  имеет тип;
- сообщение об ошибке в противном случае.

- 1 Главный тип
- 2 Подстановка типа и унификация
- 3 Теорема Хиндли-Милнера
- 4 К практике**



# Задание для практики (и на дом)

- 1 Реализуйте алгоритм Ц на Haskell.
- 2 Реализуйте алгоритм Е на Haskell.
- 3 Реализуйте алгоритм РР на Haskell.

- Лямбда-термы можно закодировать так

```
type Symb = String

infixl 2 :@

data Expr
  = Var Symb
  | Expr :@ Expr
  | Lam Symb Expr
  deriving (Eq, Show)
```

- Например, выражение  
`(Lam "x" $ Lam "y" $ Var "x") :@ (Lam "z" $ Var "z")`  
кодирует терм  $(\lambda x. \lambda y. x) (\lambda z. z)$ .

Свободные переменные терма

```
freeVars :: Expr -> [Symb]
```

Попробуйте написать реализацию.

Свободные переменные терма

```
freeVars :: Expr -> [Symb]
```

Попробуйте написать реализацию.

Реализация

```
freeVars :: Expr -> [Symb]
freeVars (Var v)      = [v]
freeVars (t1 :@ t2) = freeVars t1 'union' freeVars t2
freeVars (Lam v t)   = freeVars t \\ [v]
```

- Типы можно закодировать так

```
infixr 3 :->

data Type
  = TVar Symb
  | Type :-> Type
  deriving (Eq, Show)
```

- Например, выражение  
 $(\text{TVar "a"} \text{ :-> TVar "b"}) \text{ :-> TVar "c"}$  кодирует тип  
 $(a \rightarrow b) \rightarrow c$ , а выражение  
 $\text{TVar "a"} \text{ :-> TVar "b"} \text{ :-> TVar "c"}$  кодирует тип  
 $a \rightarrow b \rightarrow c$ .

- Контексты можно закодировать так

```
newtype Env = Env [(Symb, Type)]  
deriving (Eq, Show)
```

- Полезными могут оказаться пустой контекст и функция расширения контекста

```
emptyEnv :: Env  
emptyEnv = Env []  
  
extendEnv :: Env -> Symb -> Type -> Env  
extendEnv (Env env) s t = Env $ (s,t) : env
```



Б. Пирс.

*Типы в языках программирования.*

Лямбда пресс, Добросвет, Москва, 2012.



H.P. Barendregt.

*Lambda calculi with types.*

*In Handbook of Logic in Computer Science*, pages 117–309.

Oxford University Press, 1992.



Benjamin C. Pierce.

*Types and Programming Languages.*

MIT Press, 2002.