

Функциональное программирование

Лекция 12. Алгоритм вывода типов

Денис Николаевич Москвин

Кафедра математических и информационных технологий
Санкт-Петербургского академического университета

18.05.2012

- 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> $\Lambda_{\mathbb{T}} ::= \begin{array}{l} V \\ \quad M N \\ \quad \lambda x:\sigma. M \end{array}$	<p>Редукция</p> $(\lambda x:\sigma. M) N \rightarrow_{\beta} M[x := N]$
<p>Типы</p> $\mathbb{T} ::= \begin{array}{l} V \\ \quad \sigma \rightarrow \tau \end{array}$ <hr/> <p>Контексты</p> $\Gamma ::= \begin{array}{l} \emptyset \\ \quad \Gamma, x:\sigma \end{array}$	<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 присвоить 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 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 подстановок — подстановка с носителем, являющимся объединением носителей.

Для

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

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

имеем

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

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

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

Определение

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

Пример

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

Их унификатор

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

Действительно

$$S(\sigma) \equiv S(\tau) = (\gamma \rightarrow \gamma) \rightarrow \alpha \rightarrow \gamma \rightarrow \gamma$$

Определение

Унификатор S — это *главный унификатор* для σ и τ , если для любого другого унификатора S' существует подстановка T , такая что

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

Пример

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

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

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

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

Теорема унификации

Существует алгоритм унификации U , который для заданных типов σ и τ возвращает:

- главный унификатор S для σ и τ , если σ и τ могут быть унифицированы;
 - сообщение об ошибке в противном случае.
-
- Алгоритм $U(\sigma, \tau)$ 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, поскольку на каждом шаге сокращает либо количество \rightarrow , либо количество типовых переменных.
- $\mathcal{U}(\sigma, \tau)$ унифицирует. По индукции; используем, что если S унифицирует (σ, τ) , 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].

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 \mathcal{U} , при перестановке элементов в E : $E = \{\beta \sim \alpha \rightarrow \delta, \beta \sim (\gamma \rightarrow \delta) \rightarrow \varepsilon\}$
Что изменится в этом случае?

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

Теорема

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

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

Алгоритм E можно задать так

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

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

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

Здесь α и β — всякий раз «свежие»!

Главная пара (Principle Pair)

Определение

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

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

Лемма

Если $(\Gamma, \sigma) = PP(M)$, то $FV(M) = \text{dom}(\Gamma)$.

Пример

Для $M = \lambda x. x y$ имеем

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

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

Определение

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

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

Теорема Хиндли – Милнера

Теорема Хиндли – Милнера

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

- главную пару (Γ, σ) , 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 $\sigma_0 = \beta$.

Тогда алгоритм PP можно задать так

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

Следствие

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

- главный тип σ , 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 алгоритмы РР и РТ на Haskell.

Лямбда-термы можно закодировать так

```
type Sym = String

data Expr
  = Var Sym
  | App Expr Expr
  | Lam Sym Expr
  deriving (Eq, Read, Show)
```

То есть

`App (Lam "x" $ Lam "y" $ Var "x") (Lam "z" $ Var "z")`
кодирует $(\lambda x. \lambda y. x) (\lambda z. z)$.

Свободные переменные терма

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

Попробуйте написать реализацию.

Свободные переменные терма

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

Попробуйте написать реализацию.

Реализация

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

- Типы можно закодировать так

```
data Type
  = TVar Sym
  | Arrow Type Type
  deriving (Eq, Read, Show)
```

- То есть `Arrow (Arrow (TVar "a") (TVar "b")) (TVar "c")` кодирует $(a \rightarrow b) \rightarrow c$.
- Можно заменить конструктор данных `Arrow` на инфиксный `:->`. При этом кодирование $(a \rightarrow b) \rightarrow c$ будет более наглядным `((TVar "a") :-> (TVar "b")) :-> (TVar "c")`

Контексты можно закодировать так

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

Полезными могут оказаться пустой контекст и функция расширения контекста

```
emptyEnv :: Env  
emptyEnv = Env []
```

```
extend :: Sym -> Type -> Env -> Env  
extend s t (Env r) = Env $ (s, t) : r
```



Benjamin C. Pierce.

Types and Programming Languages.

MIT Press, 2002.