Лекция 7

Ilya Yaroshevskiy

19 октября

Содержание

1 Типовая система Хиндли-Милнера

Определение. Ранг типа. Пусть σ — тип без кванторов. $R \subseteq \text{тип} \times \mathbb{N}$:

1. $R(\sigma, 0)$
2. Если $R(\tau, k)$, то $R(\forall\overline{\alpha}.\tau, \max(k, 1))$
3. Если $R(\tau_0, k)$ и $R(\tau_1, k+1)$, то $R(\tau_0 \to \tau_1, k+1)$

Пример.

$$R(\forall\alpha.\alpha, 5)$$
$$R(\alpha, 0) \implies R(\alpha, 5)$$
$$R(\forall\alpha.\alpha, 5)$$

Пример.

$$R(\alpha \to \alpha, 0)$$
$$R(\forall\alpha.\alpha \to \alpha, 1)$$
$$R(\forall\alpha.\alpha \to \alpha, 5)$$

Пример.

$$\neg R((\forall\alpha.\alpha) \to \beta, 1), \text{т.к.} \neg R(\forall\alpha.\alpha, 0)$$
$$R((\forall\alpha.\alpha) \to \beta, 2)$$

Определение. Типовая система Хиндли-Милнера:
Рассмотрим λ-исчисление второго порядка по Карри. Типы:

1. типы(без кванторов) $\tau = \alpha | (\tau \to \tau)$
2. типовые схемы $\sigma = \forall\alpha.\sigma | \tau$

Определение. Отношение **"Быть частным случаем"** (специализация)

$$\begin{tikzcd}
\tau \arrow[r] \arrow[dr] \arrow[ddr] & \texttt{string} & \\
 & \alpha \to \beta \arrow[r] & \texttt{string} \to \texttt{int} \\
 & (\alpha \to \beta) \to (\texttt{string} \to \texttt{int}) &
\end{tikzcd}$$

$\sigma_1 \sqsubseteq \sigma_2$ (σ_2 — частный случай σ_1), если:

- $\sigma_1 \equiv \forall \alpha_1 \dots \forall \alpha_n . \tau_1$
- $\sigma_2 \equiv \forall \beta_1 \dots \forall \beta_k . \tau_1 [\alpha_1 \coloneqq \Theta_1, \dots, \alpha_n \coloneqq \Theta_n]$

Новые переменные $\beta_1, \dots, \beta_n$ не входят свободно в σ_1.

Пример.

$$\forall \alpha . \alpha \to \alpha \sqsubseteq \forall \beta . (\beta \to \beta) \to (\beta \to \beta)$$

Определение.

$$\frac{}{\Gamma, x : \sigma \vdash x : \sigma} \quad x \notin FV(\Gamma)$$

$$\frac{\Gamma \vdash A : \tau \to \tau' \quad \Gamma \vdash B : \tau}{\Gamma \vdash A\ B : \tau'}$$

$$\frac{\Gamma, x : \tau \vdash A : \tau'}{\Gamma \vdash \lambda x . A : \tau \to \tau'}$$

$$\frac{\Gamma \vdash A : \sigma \quad \Gamma, x : \sigma \vdash B : \tau}{\Gamma \vdash \texttt{let}\ x = A\ \texttt{in}\ B : \tau}$$

$$\frac{\Gamma \vdash A : \sigma'}{\Gamma \vdash A : \sigma} \quad \sigma' \sqsubseteq \sigma$$

$$\frac{\Gamma \vdash A : \sigma}{\Gamma \vdash A : \forall \alpha . \sigma} \quad \alpha \notin FV(\Gamma)$$

Пример.

$$I \equiv \lambda x . x : \forall \alpha . \alpha \to \alpha$$

$(I\ 1, I\ \text{“}a\text{“}) : (\texttt{int}, \texttt{string})$ — В первом элементе $I : \texttt{int} \to \texttt{int}$, во втором $I : \texttt{string} \to \texttt{string}$

1. $\texttt{let}\ I = \lambda x . x\ \texttt{in}\ (I\ 1, I\ \text{“}a\text{“})$, где $I : \forall \alpha . \alpha \to \alpha$
2. $(\lambda i . (i\ 1, i\ \text{“}a\text{“}))\ (\lambda x . x)$

2 Алгоритм W

Утверждение. *Система типов XM **разрешима***

Определение. Хотим решить $? \vdash A :?$, при чем найти наиболее общий тип. $\mathcal{V}$ — вызов алгоритма унификации

$$W(\Gamma, E) \Rightarrow (\tau, S)$$

1. $E \equiv x$, $x \in \Gamma$, $x : \sigma_x$
   Новые переменные $\beta_1, \dots, \beta_n$, $\sigma_x = \forall \alpha_1 \dots \alpha_n . \tau$
   $$(\forall \beta_1 \dots \forall \beta_n . \tau, \varnothing)$$
2. $E \equiv \lambda x . P$
   $$W(\Gamma, E) = (S_1(\gamma \to \tau_P), S_1)$$
   $$W(\Gamma \cup \{x : \gamma\}, P) = (\tau_P, S_1)$$
3. $E \equiv P\ Q$
   $$W(\Gamma, E) = (S_3 \gamma, S_3 \circ S_2 \circ S_1)$$
   $$W(\Gamma, P) = (\tau_P, S_1)$$
   $$W(S_1 \Gamma, Q) = (\tau_Q, S_2)$$
   $$\mathcal{V}(S_2 \tau_P, \tau_Q \to \gamma) = S_3$$

4. $E \equiv \texttt{let } x = P \texttt{ in } Q$
$$W(\Gamma, E) = (\tau_Q, S_2 \circ S_1)$$
$$W(\Gamma, P) = (\tau_P, S_1)$$
$$W(S_1\Gamma \cup \{x : \forall.\tau_f\}, Q) = (\tau_Q, S_2)$$
, где $\forall.\tau_f$ — кванторы по всем свободным перменным из τ_f

Пример.
$$\texttt{let } I = \lambda x.x \texttt{ in } (I\ 1, I\ \text{“}a\text{“})$$
Применяя 4 пункт алгоритма:
$$I : \forall\alpha.\alpha \to \alpha \vdash (I\ 1, I\ \text{“}a\text{“})$$

3 Апгрейд

Добавим правило для Y:

Определение. $Y : \forall\alpha.(\alpha \to \alpha) \to \alpha$

Примечание.

```haskell
data IntList = Nil | Cons Int IntList
```

Есть два способа разрешения рекурсивных типов:

1. **Эквирекурсивный**:
Введем оператор аналогичный Y-комбинатору, только на типах: $\mu\alpha.f(\alpha) = f(\mu\alpha.f(\alpha))$
Пример: выведем тип $\lambda x.x\ x$:
Пусть $\tau = \mu\alpha.\alpha \to \beta$. Если мы раскроем τ один раз, то получим $\tau = \tau \to \beta$. Если раскроем еще раз, то получим $\tau = (\tau \to \beta) \to \beta$.
$$\dfrac{\dfrac{\dfrac{}{x : \tau \vdash x : \tau}}{x : \tau \vdash x : \tau \to \beta} \quad \dfrac{}{x : \tau \vdash x : \tau}}{\dfrac{x : \tau \vdash x\ x : \beta}{\vdash \lambda x.x\ x : \tau \to \beta}}$$
Также можем доказать $Y \equiv \lambda f.(\lambda x.f\ (x\ x))\ (\lambda x.f\ (x\ x))$.

2. **Изорекурсивный**:
Будем считать, что $\mu\alpha.f(\alpha)$ изоморфно $(\mu\alpha.f(\alpha))$. Такой подход используется в `C`:

```c
struct list {
  list* x;
  int a;
};
```

Можем использовать так: `x->x->x->a`. Заметим, что мы неявно использовали разыменование: `*` : `list*` $\to$ `list`. В изорекурсивных типах введены специальные операции для работы с этими типами:

- roll : `list*` $\to$ `list`
- unroll : `list` $\to$ `list*`

В более общем виде (введение в типовую систему):

- roll : $f(\alpha) \to \alpha$
- unroll : $\alpha \to f(\alpha)$