Лекция 4.
Типовая система Хиндли-Милнера
Обобщённая типовая система, лямбда-куб

Ранг типа

Напомним, что $\exists\alpha.\varphi := \forall\beta.(\forall\alpha.\varphi \to \beta) \to \beta$.

Определение

Функция «ранг типа» rk $\subseteq T \times \mathbb{N}_0$. $rk(\sigma) = [mrk(\sigma), +\infty) \cap \mathbb{N}_0$, *где mrk:*

$$mrk(\tau) = \begin{cases} 0, & \tau \text{ без кванторов} \\ \max(mrk(\sigma), 1), & \tau = \forall x.\sigma \\ \max(mrk(\sigma_1) + 1, mrk(\sigma_2)), & \tau = \sigma_1 \to \sigma_2, \tau \text{ имеет кванторы} \end{cases}$$

Лемма

Если $rk(\sigma, 1)$, *то для формулы* σ *найдётся эквивалентная формула с поверхностными кванторами.*

Пример

$0 \notin rk(\forall\alpha.\gamma \to \beta)$; $1 \notin rk((\forall\alpha.\gamma \to \beta) \to f) = \{2, 3, \dots\}$
$1 \notin rk(\exists\alpha.\gamma) = rk(\forall\beta.(\forall\alpha.\gamma \to \beta) \to \beta) = \{2, 3, \dots\}$
$1 \in rk(\forall\alpha.\delta \to \forall\beta.\delta \to \forall\gamma.\delta)$

Типовая система Хиндли-Милнера: язык

Определение

Тип (τ) *и типовая схема:*

$$\tau ::= \alpha \mid (\tau \to \tau) \qquad \sigma ::= \forall x.\sigma \mid \tau$$

Пред-лямбда-терм (типизация по Карри)

$$H ::= x \mid (H\ H) \mid (\lambda x.H) \mid (\mathit{let}\ x = H\ \mathit{in}\ H)$$

Редукция для let:

$$\mathit{let}\ x = E_1\ \mathit{in}\ E_2 \to_\beta E_2[x := E_1]$$

Пример

$$\mathit{let}\ \mathit{Inc} = \lambda n.\lambda f.\lambda x.n\ f\ (f\ x)\ \mathit{in}\ \mathit{Inc}(\mathit{Inc}\ \overline{0}) \twoheadrightarrow_\beta \overline{2}$$

Типовая система Хиндли-Милнера: специализация

Определение

Пусть $\sigma_1 = \forall\alpha_1.\forall\alpha_2.\ldots.\forall\alpha_n.\tau_1$. *Тогда* σ_2 — *частный случай или специализация* σ_1 *(обознается как* $\sigma_1 \sqsubseteq \sigma_2$*), если*

$$\sigma_2 = \forall\beta_1.\forall\beta_2.\ldots.\forall\beta_m.\tau_1[\alpha_1 := S(\alpha_1), \ldots, \alpha_n := S(\alpha_n)] \text{ и } \beta_i \notin FV(\forall\alpha_1.\forall\alpha_2.\ldots.\forall\alpha_n.\tau_1)$$

Пример

$$\forall\alpha.\alpha \to \alpha \sqsubseteq \forall\beta_1.\forall\beta_2.(\beta_1 \to \beta_2) \to (\beta_1 \to \beta_2)$$

Типовая система Хиндли-Милнера: правила вывода

$$\frac{}{\Gamma, x : \sigma \vdash x : \sigma}\ x \notin FV(\Gamma) \qquad \frac{\Gamma \vdash E_0 : \tau \to \tau' \qquad \Gamma \vdash E_1 : \tau}{\Gamma \vdash E_0\ E_1 : \tau'} \qquad \frac{\Gamma, x : \tau \vdash E : \tau'}{\Gamma \vdash \lambda x.E : \tau \to \tau'}$$

$$\frac{\Gamma \vdash E_0 : \sigma \qquad \Gamma, x : \sigma \vdash E_1 : \tau}{\Gamma \vdash \mathit{let}\ x = E_0\ \mathit{in}\ E_1 : \tau} \qquad \frac{\Gamma \vdash E : \sigma'}{\Gamma \vdash E : \sigma}\ \sigma' \sqsubseteq \sigma \qquad \frac{\Gamma \vdash E : \sigma}{\Gamma \vdash E : \forall\alpha.\sigma}\ \alpha \notin FV(\Gamma)$$

Пример

$$\frac{\dfrac{\dfrac{}{x : \alpha \vdash x : \alpha}}{\vdash \lambda x.x : \alpha \to \alpha}}{\vdash \lambda x.x : \forall\alpha.\alpha \to \alpha}$$

$$\frac{\dfrac{\dfrac{}{\mathtt{id} : \forall\alpha.\alpha \to \alpha \vdash \mathtt{id} : \forall\alpha.\alpha \to \alpha}}{\mathtt{id} : \forall\alpha.\alpha \to \alpha \vdash \mathtt{id} : \mathtt{int} \to \mathtt{int}}\ S(\alpha) = \mathtt{int} \qquad \dfrac{}{\mathtt{id} : \forall\alpha.\alpha \to \alpha \vdash 0 : \mathtt{int}}}{\mathtt{id} : \forall\alpha.\alpha \to \alpha \vdash \mathtt{id}\ 0 : \mathtt{int}}$$

Отсюда: $\mathtt{let\ id}{=}\lambda x.x\ \mathtt{in}\ \langle \mathtt{id}\ 0, \mathtt{id}\ \text{«a»} \rangle : \mathtt{int\&string}$

Алгоритм реконструкции типа W

На вход подаются Γ, M, на выходе наиболее общая пара: $\langle S, \tau \rangle = W(\Gamma, M)$

1. $M = x$, $x : \tau \in \Gamma$ (иначе ошибка)
   - τ' — τ без кванторов, все свободные переменные переименованы в свежие.

   возвращаем $\langle \varnothing, \tau' \rangle$; например, $W(\{x : \forall\alpha.\varphi, y : \beta\}, x) = \langle \varnothing, \varphi[\alpha := \gamma] \rangle$
2. $M = \lambda n.E$
   - $\Gamma' = \{x : \sigma \mid x : \sigma \in \Gamma, x \neq n\} \cup \{n : \alpha\}$, α — свежая типовая переменная
   - $\langle S,\ \tau \rangle = W(\Gamma', E)$

   возвращаем $\langle S, S(\alpha) \to \tau \rangle$
3. $M = P\ Q$
   - $\langle S_1, \tau_1 \rangle = W(\Gamma, P)$; $\langle S_2, \tau_2 \rangle = W(S_1(\Gamma), Q)$
   - $S_3 = \mathcal{U}\big[S_2(\tau_1), \tau_2 \to \alpha\big]$, α — свежая

   возвращаем $\langle S_3 \circ S_2 \circ S_1, S_3(\alpha) \rangle$
4. $M = (\mathit{let}\ n = P\ \mathit{in}\ Q)$
   - $\langle S_1, \tau_1 \rangle = W(\Gamma, P)$
   - $\Gamma' = \{x : \sigma \mid x : \sigma \in \Gamma, x \neq n\} \cup \{n : \forall\alpha_1 \ldots \alpha_k.\tau_1\}$, где $\alpha_1 \ldots \alpha_k$ — все свободные переменные τ_1
   - $\langle S_2, \tau_2 \rangle = W(S_1(\Gamma'), Q)$

   возвращаем $\langle S_2 \circ S_1, \tau_2 \rangle$

Рекурсия в НМ: делаем НМ тьюринг-полной

1. Рекурсия для термов. Y-комбинатор. Добавим специальное правило вывода:

$$\frac{}{Y : \forall\alpha.(\alpha \to \alpha) \to \alpha}$$

2. Рекурсия для типов. Рассмотрим список

$$\texttt{Nil} = In_L\ 0 \qquad \texttt{Cons}\ e\ l = In_R\langle e, l\rangle \qquad \texttt{List} : ?$$

Заметим, что при попытке выписать уравнение для типа мы получим рекурсию:

$$\tau = \texttt{Int} \vee \langle \texttt{Int}, \tau\rangle$$

Рекурсивный тип надо добавить явно:

$$\tau = \mu\alpha.\texttt{Int} \vee \langle \texttt{Int}, \alpha\rangle$$

Мю-оператор — это Y-комбинатор для типов. Как его добавить в типовую систему?

Эквирекурсивные и изорекурсивные типы: $\mu\alpha.\sigma(\alpha)$

- Мю-оператор (неподвижная точка для типов): $\mu\alpha.\sigma(\alpha)$ — такой τ, что $\tau \approx \sigma(\tau)$.
- Эквирекурсивные типы. Считаем, что $\alpha = \sigma(\alpha)$. Например, в Java:

```
public abstract class Enum<E extends Enum<E>>
        implements Constable, Comparable<E>, Serializable
{ ... }
```

  Уравнение (частный случай): $E = \mathit{Enum}(E)$, или $E = \mu\varepsilon.\mathit{Enum}(\varepsilon)$.
- Изорекурсивные типы. $\alpha \neq \sigma(\alpha)$, но есть изоморфизм:

$$\texttt{roll} : \sigma(\alpha) \to \alpha \qquad \texttt{unroll} : \alpha \to \sigma(\alpha)$$

Например, для `struct List { List* next; int value; }`:

Комп.	В C++	Пример
`roll`	взятие ссылки	`List a; a.next = NULL; return len(&a)`
`unroll`	разыменование	`len (List* a) { return (*a).next ? ... : 0 }`

Разрешимость задачи реконструкции типа в разных вариантах F

Ранг типов	Собственное название	Разрешимость
0	$\lambda_{\to}$	разрешимо (лекция 2)
1	*HM*	разрешимо (алгоритм W)
2		разрешимо
$\geqslant 3$		неразрешимо

Генерики, зависимые типы

```
template <class X>
class Z {
    X field;
}
```

Что такое Z? Это функция, возвращающая тип по другому типу (генерик).

```
int main() {
    unsigned sz;
    std::cin >> sz;
    int temp_array [sz];
    std::cout << sizeof(temp_array);
    return 0;
}
```

Что такое конструкция `int[sz]`? Это функция, возвращающая тип по значению (зависимый тип).

Терминология: типы, рода, сорта

Мы будем рассматривать конструкции следующих сортов:

Название сорта	Примеры сортов	Примеры конструкций, имеющих сорт
Тип	α, $\alpha \to \beta$, $\star \to \alpha$	$3 : \texttt{int}, \texttt{id} : \forall\alpha.\alpha \to \alpha$
Род (kind)	$\star$, $\star \to \star$, $\alpha \to \star$	$\texttt{list} : \star \to \star$
Сорт	$\square$	$\star \to \star : \square$

Язык обобщённой типовой системы

Откажемся от различных пространств имён для значений, типов и прочего, а также от синтаксического их разделения. Все переменные для значений любого сорта, все лямбда-выражения для любых функций — всё записывается единообразно.

Определение (синтаксис выражений)

Константы сортов: $c ::= \{\star, \Box\}$
Выражение:

$$T ::= x \mid c \mid T\ T \mid \lambda x^T.\ T \mid \Pi x^T.\ T$$

Сокращения:

$$A \to B ::= \Pi x^A.B, \quad x \notin FV(B)$$

$$\forall x.P ::= \Pi x^\star.P \quad \Lambda x.\sigma ::= \lambda x^\star.\sigma$$

Метапеременные термов: $A, B, C, \ldots, \quad \rho, \sigma, \tau, \ldots.$
Метапеременные переменных: x, y, z

Что такое Π

Неформально: Π — аналог лямбда-выражения для типизации конструкции:

$$\lambda x^{\tau}.P : \Pi x^{\tau}.\pi$$

Вспомним сокращения:

$$A \to B ::= \Pi x^{A}.B,\ x \notin FV(B) \qquad \forall x.P ::= \Pi x^{\star}.P$$

И рассмотрим map:

$$\mathtt{map} : \forall a.\forall b.(a \to b) \to [a] \to [b]$$

Перепишем:

$$\mathtt{map} : \Pi a^{\star}.\Pi b^{\star}.(\Pi f^{\Pi x^{a}.b}.\Pi l^{[a]}.[b])$$

Заметим, что операция $[\sigma]$ строит из σ другой тип, то есть $[\sigma] = (\lambda x^{\star}.\langle\text{тип реализации списка из } x\rangle)\sigma$, можем раскрыть дальше:

$$\mathtt{map} : \Pi a^{\star}.\Pi b^{\star}.(\Pi f^{\Pi x^{a}.b}.\Pi l^{(\lambda x^{\star}\ldots)a}.(\lambda x^{\star}\ldots)b)$$

Заметим, что $\lambda\sigma^{\star}.[\sigma] : \star \to \star$

Обобщённая типовая система: семейство систем

Семейство параметризовано множеством пар $\mathcal{S} \subseteq \{\star, \Box\} \times \{\star, \Box\}$

Аксиома:

$$\frac{}{\vdash \star : \Box}$$

Общие правила вывода: $\sigma \in \{\star, \Box\}$

$$\frac{\Gamma \vdash A : \sigma}{\Gamma, x : A \vdash x : A}\ x \notin \Gamma \qquad \frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : \sigma}{\Gamma, x : C \vdash A : B}$$

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : \sigma \qquad B =_\beta B'}{\Gamma \vdash A : B'} \qquad \frac{\Gamma \vdash F : (\Pi x^A.B) \qquad \Gamma \vdash H : A}{\Gamma \vdash (F\ H) : B[x := H]}$$

Частные правила: $\langle \sigma_1, \sigma_2 \rangle \in \mathcal{S}$

$$\frac{\Gamma \vdash A : \sigma_1 \qquad \Gamma, x : A \vdash B : \sigma_2}{\Gamma \vdash (\Pi x^A.B) : \sigma_2}\ \Pi\text{-правило}$$

$$\frac{\Gamma \vdash A : \sigma_1 \qquad \Gamma, x : A \vdash P : B \qquad \Gamma, x : A \vdash B : \sigma_2}{\Gamma \vdash (\lambda x^A.P) : (\Pi x^A.B)}\ \lambda\text{-правило}$$

Типизация $\Lambda\alpha.\lambda x^{\alpha}.x$

Выражение $\Lambda\alpha.\lambda x^{\alpha}.x$ перепишется как $\lambda\alpha^{\star}.\lambda x^{\alpha}.x$, ожидаем тип $\Pi\alpha^{\star}.\Pi x^{\alpha}.\alpha$. Потребуются частные правила для $\langle\star,\star\rangle$ и $\langle\Box,\star\rangle$.

$$\frac{\Gamma \vdash A : \sigma_1 \qquad \Gamma, x : A \vdash B : \sigma_2}{\Gamma \vdash (\Pi x^{A}.B) : \sigma_2} \qquad \frac{\Gamma \vdash A : \sigma_1 \qquad \Gamma, x : A \vdash P : B \qquad \Gamma, x : A \vdash B : \sigma_2}{\Gamma \vdash (\lambda x^{A}.P) : (\Pi x^{A}.B)}$$

$$\dfrac{\dfrac{}{\vdash \star : \Box} \qquad \dfrac{\dfrac{}{a : \star \vdash a : \star} \quad \dfrac{}{a : \star, x : a \vdash x : a} \quad \dfrac{}{a : \star, x : a \vdash a : \star}\,\langle\star,\star\rangle}{a : \star \vdash \lambda x^{a}.x : \Pi x^{a}.a} \qquad \dfrac{\dfrac{}{a : \star, x : a \vdash a : \star}}{a : \star \vdash \Pi x^{a}.a : \star}\,\langle\Box,\star\rangle}{\vdash \lambda a^{\star}.\lambda x^{a}.x : \Pi a^{\star}.\Pi x^{a}.a}$$

Общие свойства обобщённой системы типов

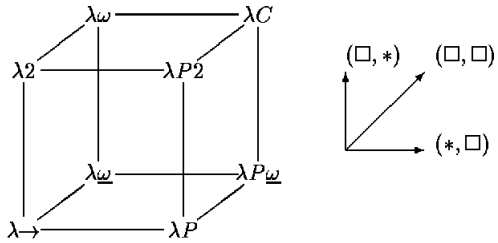
Теорема

Для обобщённой системы типов выполнена теорема Чёрча-Россера

Теорема

Обобщённая система типов сильно нормализуема

Лямбда-куб Барендрегта



Типовые системы и языки программирования:

Классические и функциональные языки:

$\lambda_{\rightarrow}$	$\{\langle\star,\star\rangle\}$	Классический Паскаль
$\lambda_{\underline{\omega}}$	$\{\langle\star,\star\rangle, \langle\Box,\star\rangle\}$	Система F
λ_{ω}	$\{\langle\star,\star\rangle, \langle\Box,\star\rangle, \langle\Box,\Box\rangle\}$	Haskell, Ocaml

Языки с зависимыми типами данных (обычно около λC):
Idris, Coq, Agda, Arend, C++ :).

Изоморфизм Карри-Ховарда

Рассмотрим формулу с квантором: $\forall x.\pi$. Ей соответствует $\Pi x.\pi$, а доказательство было бы $\lambda x.P : \Pi x.\pi$. Подробнее:

$\lambda x^\star.P : \Pi x^\star.\pi : \star$	$x \in V$, для логики 2 порядка
$\lambda x^\upsilon.P : \Pi x^\upsilon.\pi : \star$, если $\upsilon : \star$	$x \in U \subseteq D$, для (многосортной) логики 1 порядка

В самом деле: $\forall x.\pi$ требует $\pi[x := \theta]$ при всех θ (соответствующих υ).
Доказательство: функция $\lambda x.P$, отображающая θ в терм, обитающий в $\Pi x.\pi$.

Логика	λ-исчисление	Комментарий
π	$x : \pi$	Утверждение
$\pi(x)$	$P : \pi(x)$	Предикат
$\forall x \in U.\pi$	$\lambda x^\upsilon.P : \Pi x^\upsilon.\pi$	Тотальная функция
$\exists x \in U.\varepsilon$	$(X, U[x := X]) : \Sigma x^\upsilon.\varepsilon$	Зависимая пара

Idris: пример языка с зависимыми типами

```
data Nat : Type where
    Z : Nat
    S : Nat -> Nat

data Vect : Nat -> Type -> Type where
   Nil  : Vect Z a
   (::) : a -> Vect k a -> Vect (S k) a

(++) : Vect n a -> Vect m a -> Vect (n + m) a
(++) Nil       ys = ys
(++) (x :: xs) ys = x :: xs ++ ys
```

Зависимые типы: printf на Идрис

```
-- Mukesh Tiwari, https://github.com/mukeshtiwari/Idris/blob/master/Printf.idr
data Format = FInt Format
            | FString Format
            | FOther Char Format
            | FEnd

format : List Char -> Format
format ('%' :: 'd' :: cs ) = FInt ( format cs )
format ('%' :: 's' :: cs ) = FString ( format cs )
format ( c :: cs )         = FOther c ( format cs )
format []                  = FEnd

interpFormat : Format -> Type
interpFormat ( FInt f )     = Int -> interpFormat f
interpFormat ( FString f )  = String -> interpFormat f
interpFormat ( FOther _ f ) = interpFormat f
interpFormat FEnd           = String
```

Printf на Идрис

```
formatString : String -> Format
formatString s = format ( unpack s )

toFunction : ( fmt : Format ) -> String -> interpFormat fmt
toFunction ( FInt f ) a     = \i => toFunction f ( a ++ show i )
toFunction ( FString f ) a  = \s => toFunction f ( a ++ s )
toFunction ( FOther c f ) a = toFunction f ( a ++ singleton c )
toFunction FEnd a           = a

sprintf : ( s : String ) -> interpFormat ( formatString s )
sprintf s = toFunction ( formatString s ) ""

main : IO ()
main = putStrLn (sprintf "String: %s, integer: %d" "alpha" (10+23))
```