Лекция 5.
Обобщённая типовая система, лямбда-куб

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: $A, B, C, \dots, \quad \rho, \sigma, \tau, \dots$.
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:

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

Перепишем:

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

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:

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

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 $\langle\square,\star\rangle$.

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

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

Общие свойства обобщённой системы типов

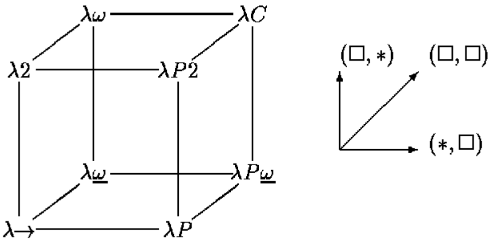
Теорема

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:

$\lambda_{\to}$	$\{\langle\star,\star\rangle\}$	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))
```