

Вывод типов от Хиндли – Милнера до GHC 8.8

Лекции 1–2

В. Н. Брагилевский

2 марта 2019 г.

Computer Science клуб (Санкт-Петербург)

Институт математики, механики и компьютерных наук
имени И. И. Воровича, Южный федеральный университет (Ростов-на-Дону)

Давайте договоримся относительно предпочтений в области ЯП



Bartosz Milewski

@BartoszMilewski

Читать



functional languages are better than procedural, strong typing is better than weak, static typing better than dynamic
macbeth.cs.ucdavis.edu/lang_study.pdf

Перевести твит

22:24 - 4 нояб. 2014 г. Siena, Toscana

153 ретвита 149 отметок «Нравится»



14



153



149



- функциональные языки
- сильная типизация
- статическая типизация

- функциональные языки
- сильная типизация
- статическая типизация
- **вывод типов (type inference)**

Сегодня и завтра мы будем работать по следующему плану:

- вывод типов в простом функциональном языке с поддержкой полиморфизма
- современный взгляд на алгоритмы вывода типов
- трудности вывода типов в Haskell
- вывод типов в системе OutsideIn(X)
- вывод типов в Haskell с обобщёнными алгебраическими типами данных, классами типов и семействами типов

В основе этого курса лекций лежат два источника:

1. Жиль Довек, Жан-Жак Леви. Введение в теорию языков программирования. ДМК Пресс, 2013. 134 с. (глава 6)
2. D. Vytiniotis, S. Peyton Jones, T. Schrijvers, M. Sulzmann. OutsideIn(x) modular type inference with local assumptions. Journal of Functional Programming, Volume 21, Issue 4-5, September 2011.

Язык РСФ

Синтаксис языка PCF (Programming Computable Functions)

$t = x$	(переменная)
n	(числовой литерал)
$t + t$ $t - t$ $t * t$ t / t	(операции)
fun $x \rightarrow t$	(функция)
$t t$	(применение)
ifz t then t else t	(условие)
fix $x t$	(рекурсия)
let $x = t$ in t	(let-объявление)

Давайте запрограммируем функцию вычисления факториала!

Давайте запрограммируем функцию вычисления факториала!

```
fix f fun n → ifz n then 1 else n*f (n - 1)
```

$T = X$	(переменная)
\mathbb{N}	(литерал типа)
$T \rightarrow T$	(тип функции)

И их можно проверять!

Введём отношение типизации $e \vdash t:A$
(«терм t имеет тип A в окружении e »):

И их можно проверять!

Введём отношение типизации $e \vdash t:A$
(«терм t имеет тип A в окружении e »):

$$\frac{}{e \vdash x:A} \text{ если } e \text{ содержит } x:A,$$

И их можно проверять!

Введём отношение типизации $e \vdash t:A$
(«терм t имеет тип A в окружении e »):

$$\frac{}{e \vdash x:A} \text{ если } e \text{ содержит } x:A,$$

$$\frac{}{e \vdash n:\mathbb{N}},$$

$$\frac{e \vdash t:\mathbb{N} \quad e \vdash u:\mathbb{N}}{e \vdash t \otimes u:\mathbb{N}},$$

$$\frac{e, x:A \vdash t:B}{e \vdash \mathbf{fun} x \rightarrow t:A \rightarrow B},$$

$$\frac{e, x:A \vdash t:B}{e \vdash \mathbf{fun} x \rightarrow t:A \rightarrow B},$$

$$\frac{e \vdash u:A \quad e \vdash t:A \rightarrow B}{e \vdash t u:B},$$

$$\frac{e \vdash t:\mathbb{N} \quad e \vdash u:A \quad e \vdash v:A}{e \vdash \mathbf{ifz} \ t \ \mathbf{then} \ u \ \mathbf{else} \ v:A},$$

$$\frac{e \vdash t:\mathbb{N} \quad e \vdash u:A \quad e \vdash v:A}{e \vdash \mathbf{ifz} \ t \ \mathbf{then} \ u \ \mathbf{else} \ v:A},$$

$$\frac{e, x:A \vdash t:A}{e \vdash \mathbf{fix} \ x \ t:A},$$

$$\frac{e \vdash t:\mathbb{N} \quad e \vdash u:A \quad e \vdash v:A}{e \vdash \mathbf{ifz} \ t \ \mathbf{then} \ u \ \mathbf{else} \ v:A},$$

$$\frac{e, x:A \vdash t:A}{e \vdash \mathbf{fix} \ x \ t:A},$$

$$\frac{e \vdash t:A \quad e, x:A \vdash u:B}{e \vdash \mathbf{let} \ x = t \ \mathbf{in} \ u:B}.$$

К примеру, давайте что-нибудь «проверим»

$$\vdash (\mathbf{fun} \ x \rightarrow 2 + x) \ 3 : \mathbb{N}$$

К примеру, давайте что-нибудь «проверим»

$$\vdash \mathbf{fun} \ x \rightarrow 2 + x : \mathbb{N} \rightarrow \mathbb{N} \qquad \vdash 3 : \mathbb{N}$$

$$\vdash (\mathbf{fun} \ x \rightarrow 2 + x) \ 3 : \mathbb{N}$$

К примеру, давайте что-нибудь «проверим»

$$\frac{\frac{x:\mathbb{N} \vdash 2 + x:\mathbb{N}}{\vdash \mathbf{fun} x \rightarrow 2 + x:\mathbb{N} \rightarrow \mathbb{N}} \quad \frac{}{\vdash 3:\mathbb{N}}}{\vdash (\mathbf{fun} x \rightarrow 2 + x) 3:\mathbb{N}}$$

К примеру, давайте что-нибудь «проверим»

$$\frac{\frac{x:\mathbb{N} \vdash 2:\mathbb{N} \quad x:\mathbb{N} \vdash x:\mathbb{N}}{x:\mathbb{N} \vdash 2 + x:\mathbb{N}}}{\vdash \mathbf{fun} \ x \rightarrow 2 + x:\mathbb{N} \rightarrow \mathbb{N}} \quad \frac{}{\vdash 3:\mathbb{N}}$$
$$\frac{}{\vdash (\mathbf{fun} \ x \rightarrow 2 + x) \ 3:\mathbb{N}}$$

К примеру, давайте что-нибудь «проверим»

$$\frac{\frac{\frac{}{x:\mathbb{N} \vdash 2:\mathbb{N}}}{x:\mathbb{N} \vdash 2 + x:\mathbb{N}} \quad \frac{}{x:\mathbb{N} \vdash x:\mathbb{N}}}{\vdash \mathbf{fun} \ x \rightarrow 2 + x:\mathbb{N} \rightarrow \mathbb{N}} \quad \frac{}{\vdash 3:\mathbb{N}}}{\vdash (\mathbf{fun} \ x \rightarrow 2 + x) \ 3:\mathbb{N}}$$

fun $x \rightarrow x$

fun $x \rightarrow x$

\vdash **fun** $x \rightarrow x : \mathbb{N} \rightarrow \mathbb{N}$

\vdash **fun** $x \rightarrow x : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow (\mathbb{N} \rightarrow \mathbb{N})$

fun $x \rightarrow x$

\vdash **fun** $x \rightarrow x : \mathbb{N} \rightarrow \mathbb{N}$

\vdash **fun** $x \rightarrow x : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow (\mathbb{N} \rightarrow \mathbb{N})$

\vdash **fun** $x \rightarrow x : X \rightarrow X$

- Замкнутый терм (нет свободных переменных)
- Свободные переменные в типе

Вообще говоря, языку PCF нужно много чего ещё:

- операционная семантика
- интерпретатор
- абстрактная машина
- компилятор
- денотационная семантика
- расширения для работы с данными

Алгоритм вывода типов Хиндли

Начнём с примера и выведем тип терма $\text{fun } f \rightarrow 2 + f \ 1$

$\vdash \text{fun } f \rightarrow 2 + f \ 1 : ?$

Начнём с примера и выведем тип терма $\text{fun } f \rightarrow 2 + f 1$

$\vdash \text{fun } f \rightarrow 2 + f 1 : ?$

$f : X \vdash 2 + f 1 : ?$

Начнём с примера и выведем тип терма $\text{fun } f \rightarrow 2 + f 1$

$\vdash \text{fun } f \rightarrow 2 + f 1 : ?$

$f : X \vdash 2 + f 1 : ?$

$f : X \vdash 2 : ?$

Начнём с примера и выведем тип терма $\text{fun } f \rightarrow 2 + f 1$

$\vdash \text{fun } f \rightarrow 2 + f 1 : ?$

$f : X \vdash 2 + f 1 : ?$

$f : X \vdash 2 : \mathbb{N}$

Начнём с примера и выведем тип терма $\text{fun } f \rightarrow 2 + f 1$

$\vdash \text{fun } f \rightarrow 2 + f 1 : ?$

$f : X \vdash 2 + f 1 : ?$

$f : X \vdash 2 : \mathbb{N}$

$f : X \vdash f 1 : ?$

Начнём с примера и выведем тип терма $\text{fun } f \rightarrow 2 + f 1$

$\vdash \text{fun } f \rightarrow 2 + f 1 : ?$

$f : X \vdash 2 + f 1 : ?$

$f : X \vdash 2 : \mathbb{N}$

$f : X \vdash f 1 : ?$

$f : X \vdash 1 : ?$

Начнём с примера и выведем тип терма $\text{fun } f \rightarrow 2 + f 1$

$\vdash \text{fun } f \rightarrow 2 + f 1 : ?$

$f : X \vdash 2 + f 1 : ?$

$f : X \vdash 2 : \mathbb{N}$

$f : X \vdash f 1 : ?$

$f : X \vdash 1 : \mathbb{N}$

Начнём с примера и выведем тип терма $\text{fun } f \rightarrow 2 + f 1$

$\vdash \text{fun } f \rightarrow 2 + f 1 : ?$

$f : X \vdash 2 + f 1 : ?$

$f : X \vdash 2 : \mathbb{N}$

$f : X \vdash f 1 : Y$

Уравнения на типах

$X = \mathbb{N} \rightarrow Y$

Начнём с примера и выведем тип терма $\text{fun } f \rightarrow 2 + f 1$

$\vdash \text{fun } f \rightarrow 2 + f 1 : ?$

$f : X \vdash 2 + f 1 : \mathbb{N}$

Уравнения на типах

$X = \mathbb{N} \rightarrow Y$

$\mathbb{N} = \mathbb{N}$

$Y = \mathbb{N}$

Начнём с примера и выведем тип терма $\text{fun } f \rightarrow 2 + f 1$

$\vdash \text{fun } f \rightarrow 2 + f 1 : X \rightarrow \mathbb{N}$

Уравнения на типах

$$X = \mathbb{N} \rightarrow Y$$

$$\mathbb{N} = \mathbb{N}$$

$$Y = \mathbb{N}$$

Начнём с примера и выведем тип терма $\text{fun } f \rightarrow 2 + f 1$

$\vdash \text{fun } f \rightarrow 2 + f 1 : X \rightarrow \mathbb{N}$

Уравнения на типах

$$X = \mathbb{N} \rightarrow Y$$

$$\mathbb{N} = \mathbb{N}$$

$$Y = \mathbb{N}$$

Решение

$$X = \mathbb{N} \rightarrow \mathbb{N}, Y = \mathbb{N}$$

Начнём с примера и выведем тип терма $\text{fun } f \rightarrow 2 + f 1$

$\vdash \text{fun } f \rightarrow 2 + f 1 : X \rightarrow \mathbb{N}$

Уравнения на типах

$$X = \mathbb{N} \rightarrow Y$$

$$\mathbb{N} = \mathbb{N}$$

$$Y = \mathbb{N}$$

Решение

$$X = \mathbb{N} \rightarrow \mathbb{N}, Y = \mathbb{N}$$

$\vdash \text{fun } f \rightarrow 2 + f 1 : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$

Алгоритм Хиндли в такой форме состоит из двух этапов

1. Рекурсивно обходим терм, получая тип и систему уравнений: $e \vdash t \rightsquigarrow A, E$.
2. Решаем систему уравнений E , подставляем решение в A и получаем результат: $e \vdash t:T$.

Первый этап алгоритма Хиндли можно описать набором правил

$$\frac{}{e \vdash x \rightsquigarrow A, \emptyset} \text{ если } e \text{ содержит } x:A,$$

$$\frac{}{e \vdash n \rightsquigarrow \mathbb{N}, \emptyset},$$

Первый этап алгоритма Хиндли можно описать набором правил

$$\frac{}{e \vdash x \rightsquigarrow A, \emptyset} \text{ если } e \text{ содержит } x:A,$$

$$\frac{}{e \vdash n \rightsquigarrow \mathbb{N}, \emptyset},$$

$$\frac{e \vdash u \rightsquigarrow A, E \quad e \vdash t \rightsquigarrow B, F}{e \vdash t \otimes u \rightsquigarrow \mathbb{N}, E \cup F \cup \{A = \mathbb{N}, B = \mathbb{N}\}},$$

$$\frac{e, x:X \vdash t \rightsquigarrow A, E}{e \vdash \mathbf{fun} \ x \rightarrow t \rightsquigarrow X \rightarrow A, E'}$$

$$\frac{e, x:X \vdash t \rightsquigarrow A, E}{e \vdash \mathbf{fun} x \rightarrow t \rightsquigarrow X \rightarrow A, E'}$$

$$\frac{e \vdash u \rightsquigarrow A, E \quad e \vdash t \rightsquigarrow B, F}{e \vdash tu \rightsquigarrow X, E \cup F \cup \{B = A \rightarrow X\}}$$

$$\frac{e \vdash t \rightsquigarrow A, E \quad e \vdash u \rightsquigarrow B, F \quad e \vdash v \rightsquigarrow C, G}{e \vdash \mathbf{ifz} \ t \ \mathbf{then} \ u \ \mathbf{else} \ v \rightsquigarrow B, E \cup F \cup G \cup \{A = \mathbb{N}, B = C\}},$$

$$\frac{e, x:X \vdash t \rightsquigarrow A, E}{e \vdash \mathbf{fix} \ x \ t \rightsquigarrow A, E \cup \{X = A\}},$$

$$\frac{e, x:X \vdash t \rightsquigarrow A, E}{e \vdash \mathbf{fix} \ x \ t \rightsquigarrow A, E \cup \{X = A\}},$$

$$\frac{e \vdash t \rightsquigarrow A, E \quad e, x:A \vdash u \rightsquigarrow B, F}{e \vdash \mathbf{let} \ x = t \ \mathbf{in} \ u \rightsquigarrow B, E \cup F}.$$

Систему уравнений можно решать алгоритмом Робинсона:

Форма уравнения	Действие
$A \rightarrow B = C \rightarrow D$	замена на $A = C$ и $B = D$
$\mathbb{N} = \mathbb{N}$	удаление
$\mathbb{N} = A \rightarrow B$	ОШИБКА
$A \rightarrow B = \mathbb{N}$	ОШИБКА
$X = X$	удаление
$X = A$ или $A = X$	если X входит в A и не совпадает с ним, то ОШИБКА если X не входит в A , то X заменяется на A во всех уравнениях

Если алгоритм завершается без ошибки,
то система приобретает вид:

$$X_1 = A_1, \dots, X_n = A_n,$$

где X_i – это различные переменные,
не входящие в A_i .

$$\sigma = A_1/X_1, \dots, A_n/X_n \text{ (} A_1 \text{ вместо } X_1, \dots \text{)}$$

Эта подстановка является *главным* (*principal*) решением системы: для любого решения θ исходной системы имеется некоторая подстановка η , такая что $\theta = \eta \circ \sigma$.

$$\sigma = A_1/X_1, \dots, A_n/X_n \text{ (} A_1 \text{ вместо } X_1, \dots \text{)}$$

Эта подстановка является *главным (principal)* решением системы: для любого решения θ исходной системы имеется некоторая подстановка η , такая что $\theta = \eta \circ \sigma$.

Пишут: $\sigma = \text{mgu}(E)$ – *наиболее общий унификатор (most general unifier)*.

Если $\vdash t \rightsquigarrow A$, E и σ – главное решение E , то $\vdash t : \sigma A$.

Если $\vdash t \rightsquigarrow A$, E и σ – главное решение E , то $\vdash t : \sigma A$.

Более того, σA – *главный* тип терма t : для любого другого типа B терма t существует такая подстановка η , что $B = \eta \sigma A$.

Что на самом деле делает алгоритм Робинсона?

Эрбрановские термы (конечные деревья):

- переменная – это терм (например, X)
- константа – это терм (например, \mathbb{N})
- n -арный функциональный символ, применённый к n термам – это терм:
 $f(t_1, t_2, \dots, t_n)$ (например, $X \rightarrow \mathbb{N}$)

Что на самом деле делает алгоритм Робинсона?

Эрбрановские термы (конечные деревья):

- переменная – это терм (например, X)
- константа – это терм (например, \mathbb{N})
- n -арный функциональный символ, применённый к n термам – это терм:
 $f(t_1, t_2, \dots, t_n)$ (например, $X \rightarrow \mathbb{N}$)

Алгоритм Робинсона унифицирует систему эрбрановских термов с ограничениями в виде синтаксических равенств.

Алгоритм Хиндли с немедленным разрешением

Два этапа не нужны!

Отношение: $e \vdash t \rightsquigarrow A, \rho$

(A — это главный тип t в окружении ρe).

Два этапа не нужны!

Отношение: $e \vdash t \rightsquigarrow A, \rho$

(A – это главный тип t в окружении ρe).

$$\frac{}{e \vdash x \rightsquigarrow A, \emptyset} \text{ если } e \text{ содержит } x:A,$$

$$\overline{e \vdash n \rightsquigarrow \mathbb{N}, \emptyset},$$

$$\overline{e \vdash n \rightsquigarrow \mathbb{N}, \emptyset},$$

$$\frac{e \vdash u \rightsquigarrow A, \rho \quad \sigma \rho e \vdash t \rightsquigarrow B, \rho'}{e \vdash t \otimes u \rightsquigarrow \mathbb{N}, \sigma' \circ \rho' \circ \sigma \circ \rho} \text{ если } \begin{cases} \sigma = mgu(A = \mathbb{N}), \\ \sigma' = mgu(B = \mathbb{N}), \end{cases}$$

$$\frac{e, x:X \vdash t \rightsquigarrow A, \rho}{e \vdash \mathbf{fun} \ x \rightarrow t \rightsquigarrow (\rho X \rightarrow A), \rho},$$

$$\frac{e, x:X \vdash t \rightsquigarrow A, \rho}{e \vdash \mathbf{fun} x \rightarrow t \rightsquigarrow (\rho X \rightarrow A), \rho},$$

$$\frac{e \vdash u \rightsquigarrow A, \rho \quad \rho e \vdash t \rightsquigarrow B, \rho'}{e \vdash tu \rightsquigarrow \sigma X, \sigma \circ \rho' \circ \rho} \text{ если } \sigma = \text{mgu}(B = \rho' A \rightarrow X),$$

$$\frac{
\begin{array}{c}
e \vdash t \rightsquigarrow A, \rho \\
\sigma \rho e \vdash u \rightsquigarrow B, \rho' \quad \rho' \sigma \rho e \vdash v \rightsquigarrow C, \rho''
\end{array}
}{
e \vdash \mathbf{ifz} \ t \ \mathbf{then} \ u \ \mathbf{else} \ v \rightsquigarrow \sigma' C, \sigma' \circ \rho'' \circ \rho' \circ \sigma \circ \rho
}$$

если $\sigma = mgu(A = \mathbb{N})$ и $\sigma' = mgu(\rho'' B = C)$,

$$\frac{e, x:X \vdash t \rightsquigarrow A, \rho}{e \vdash \mathbf{fix} \ x \ t \rightsquigarrow \sigma A, \sigma \circ \rho} \text{ если } \sigma = \text{mgu}(A = \rho X),$$

$$\frac{e, x:X \vdash t \rightsquigarrow A, \rho}{e \vdash \mathbf{fix} \ x \ t \rightsquigarrow \sigma A, \sigma \circ \rho} \text{ если } \sigma = \text{mgu}(A = \rho X),$$

$$\frac{e \vdash t \rightsquigarrow A, \rho \quad \rho e, x:A \vdash u \rightsquigarrow B, \rho'}{e \vdash \mathbf{let} \ x = t \ \mathbf{in} \ u \rightsquigarrow B, \rho' \circ \rho}.$$

Разберём случай терма $\mathbf{fun} \ x \rightarrow x$

$$\frac{\frac{}{e \vdash x \rightsquigarrow A, \emptyset} \text{ если } e \text{ содержит } x:A}{e, x:X \vdash t \rightsquigarrow A, \rho}}{e \vdash \mathbf{fun} \ x \rightarrow t \rightsquigarrow (\rho X \rightarrow A), \rho}$$

Разберём случай терма $\text{fun } x \rightarrow x$

$$\frac{\frac{\frac{}{e \vdash x \rightsquigarrow A, \emptyset} \text{ если } e \text{ содержит } x:A}{e, x:X \vdash t \rightsquigarrow A, \rho}}{e \vdash \mathbf{fun } x \rightarrow t \rightsquigarrow (\rho X \rightarrow A), \rho}}$$
$$\frac{\frac{}{x:X \vdash x \rightsquigarrow X, \emptyset}}{\vdash \mathbf{fun } x \rightarrow x \rightsquigarrow (X \rightarrow X), \emptyset}}$$

Можем даже замахнуться на $(\text{fun } x \rightarrow x)(\text{fun } x \rightarrow x)$

$$\frac{e \vdash u \rightsquigarrow A, \rho \quad \rho e \vdash t \rightsquigarrow B, \rho'}{e \vdash tu \rightsquigarrow \sigma X, \sigma \circ \rho' \circ \rho}$$

если $\sigma = \text{mgu}(B = \rho' A \rightarrow X)$

Можем даже замахнуться на $(\text{fun } x \rightarrow x)(\text{fun } x \rightarrow x)$

$$\frac{e \vdash u \rightsquigarrow A, \rho \quad \rho e \vdash t \rightsquigarrow B, \rho'}{e \vdash tu \rightsquigarrow \sigma X, \sigma \circ \rho' \circ \rho}$$

если $\sigma = \text{mgu}(B = \rho' A \rightarrow X)$

⋮

⋮

$$\vdash \text{fun } x \rightarrow x \rightsquigarrow Y \rightarrow Y, \emptyset$$

$$\vdash \text{fun } x \rightarrow x \rightsquigarrow Z \rightarrow Z, \emptyset$$

$$\vdash (\text{fun } x \rightarrow x)(\text{fun } x \rightarrow x) \rightsquigarrow \sigma X, \sigma$$

если $\sigma = \text{mgu}(Z \rightarrow Z = (Y \rightarrow Y) \rightarrow X)$

- $Z \rightarrow Z = (Y \rightarrow Y) \rightarrow X$

- $Z \rightarrow Z = (Y \rightarrow Y) \rightarrow X$
- $Z = Y \rightarrow Y, Z = X$

- $Z \rightarrow Z = (Y \rightarrow Y) \rightarrow X$
- $Z = Y \rightarrow Y, Z = X$
- $X = Y \rightarrow Y$

- $Z \rightarrow Z = (Y \rightarrow Y) \rightarrow X$
- $Z = Y \rightarrow Y, Z = X$
- $X = Y \rightarrow Y$
- $\sigma = X / (Y \rightarrow Y) - \text{mgu}$

- $Z \rightarrow Z = (Y \rightarrow Y) \rightarrow X$
- $Z = Y \rightarrow Y, Z = X$
- $X = Y \rightarrow Y$
- $\sigma = X / (Y \rightarrow Y) - \text{mgu}$
- $\vdash (\mathbf{fun} \ x \rightarrow x)(\mathbf{fun} \ x \rightarrow x): Y \rightarrow Y$

К сожалению, бывают и проблемные термы

let *id* = **fun** $x \rightarrow x$ **in** *id id*

let $id = \mathbf{fun} \ x \rightarrow x \ \mathbf{in} \ id \ id$

$$\frac{e \vdash t \rightsquigarrow A, \rho \quad \rho e, x:A \vdash u \rightsquigarrow B, \rho'}{e \vdash \mathbf{let} \ x = t \ \mathbf{in} \ u \rightsquigarrow B, \rho' \circ \rho}$$

let $id = \mathbf{fun} \ x \rightarrow x \mathbf{in} \ id \ id$

$$\frac{e \vdash t \rightsquigarrow A, \rho \quad \rho e, x:A \vdash u \rightsquigarrow B, \rho'}{e \vdash \mathbf{let} \ x = t \mathbf{in} \ u \rightsquigarrow B, \rho' \circ \rho}$$

Нужно предварительно типизировать термы $\mathbf{fun} \ x \rightarrow x$ и $id \ id$, но последний терм нетипизируем.

Полиморфные типы в RCF

Расширим язык типов схемами типов с кванторами

$$A = X$$

$$| \mathbb{N}$$

$$| A \rightarrow A$$

$$S = Y$$

$$| [A]$$

$$| \forall X S$$

Расширим язык типов схемами типов с кванторами

$A = X$

| \mathbb{N}

| $A \rightarrow A$

$S = Y$

| $[A]$

| $\forall X S$

Пример схемы типа

$\forall X_1 \dots \forall X_n A,$

где A — это тип

Теперь определим отношение типизации со схемами $e \vdash t:S$

$\frac{}{e \vdash x:S}$ если e содержит $x:S$,

Теперь определим отношение типизации со схемами $e \vdash t:S$

$$\frac{}{e \vdash x:S} \text{ если } e \text{ содержит } x:S,$$

$$\frac{}{e \vdash n:[\mathbb{N}]},$$

$$\frac{e \vdash u:[\mathbb{N}] \quad e \vdash t:[\mathbb{N}]}{e \vdash t \otimes u:[\mathbb{N}]},$$

$$\frac{e, x:[A] \vdash t:[B]}{e \vdash \mathbf{fun} \ x \rightarrow t:[A \rightarrow B]},$$

$$\frac{e, x:[A] \vdash t:[B]}{e \vdash \mathbf{fun} \ x \rightarrow t:[A \rightarrow B]},$$

$$\frac{e \vdash u:[A] \quad e \vdash t:[A \rightarrow B]}{e \vdash t \ u:[B]},$$

$$\frac{e \vdash t : [\mathbb{N}] \quad e \vdash u : [A] \quad e \vdash v : [A]}{e \vdash \mathbf{ifz} \ t \ \mathbf{then} \ u \ \mathbf{else} \ v : [A]},$$

$$\frac{e \vdash t: [\mathbb{N}] \quad e \vdash u: [A] \quad e \vdash v: [A]}{e \vdash \mathbf{ifz} \ t \ \mathbf{then} \ u \ \mathbf{else} \ v: [A]},$$

$$\frac{e, x: [A] \vdash t: [A]}{e \vdash \mathbf{fix} \ x \ t: [A]},$$

$$\frac{e \vdash t:[\mathbb{N}] \quad e \vdash u:[A] \quad e \vdash v:[A]}{e \vdash \mathbf{ifz} \ t \ \mathbf{then} \ u \ \mathbf{else} \ v:[A]},$$

$$\frac{e, x:[A] \vdash t:[A]}{e \vdash \mathbf{fix} \ x \ t:[A]},$$

$$\frac{e \vdash t:S \quad e, x:S \vdash u:[B]}{e \vdash \mathbf{let} \ x = t \ \mathbf{in} \ u:[B]},$$

$$\frac{e \vdash t : S}{e \vdash t : \forall X S}$$

если X не входит свободно в e ,

$$\frac{e \vdash t : S}{e \vdash t : \forall X S}$$

если X не входит свободно в e ,

$$\vdash \mathbf{fun} \ x \rightarrow x : [X \rightarrow X]$$

$$\frac{e \vdash t : S}{e \vdash t : \forall X S}$$

если X не входит свободно в e ,

$$\frac{\vdash \mathbf{fun} \ x \rightarrow x : [X \rightarrow X]}{\vdash \mathbf{fun} \ x \rightarrow x : \forall X [X \rightarrow X]}$$

$$\frac{e \vdash t : \forall X S}{e \vdash t : (A/X)S}.$$

$$\frac{e \vdash t : \forall X S}{e \vdash t : (A/X)S}$$

$\vdash \mathbf{fun} \ x \rightarrow x : \forall X [X \rightarrow X]$

$$\frac{e \vdash t : \forall X S}{e \vdash t : (A/X)S}$$

$$\frac{\vdash \mathbf{fun} \ x \rightarrow x : \forall X [X \rightarrow X]}{\vdash \mathbf{fun} \ x \rightarrow x : [\mathbb{N} \rightarrow \mathbb{N}]}$$

Этот набор правил решает проблему типизации `let` с *id id*

$$\frac{e \vdash t:S \quad e, x:S \vdash u:[B]}{e \vdash \mathbf{let} \ x = t \ \mathbf{in} \ u:[B]}$$

Этот набор правил решает проблему типизации `let` с `id id`

$$\frac{e \vdash t:S \quad e, x:S \vdash u:[B]}{e \vdash \mathbf{let} \ x = t \ \mathbf{in} \ u:[B]}$$

⋮

$$\vdash \mathbf{fun} \ x \rightarrow x : \forall X [X \rightarrow X]$$

⋮

$$id : \forall X [X \rightarrow X] \vdash id \ id : [Y \rightarrow Y]$$

$$\vdash \mathbf{let} \ x = \mathbf{fun} \ x \rightarrow x \ \mathbf{in} \ id \ id : [Y \rightarrow Y]$$

Алгоритм Дамаса – Милнера

Для вывода полиморфного типа достаточно всего лишь...

Для вывода полиморфного типа достаточно всего лишь...

$$\frac{}{e \vdash x \rightsquigarrow (Y_1/X_1 \dots Y_n/X_n)A, \emptyset}$$

если e содержит $x: \forall X_1 \dots \forall X_n [A]$
и Y_1, \dots, Y_n – новые переменные

А также...

$$\frac{e \vdash t \rightsquigarrow A, \rho \quad \rho e, x: \text{Gen}(A, \rho e) \vdash u \rightsquigarrow B, \rho'}{e \vdash \mathbf{let } x = t \mathbf{ in } u \rightsquigarrow B, \rho' \circ \rho},$$

где $\text{Gen}(A, e)$ – это схема, полученная из $[A]$ навешиванием кванторов на все типовые переменные, входящие свободно в $[A]$, но не в e .

- Если разрешить использовать кванторы для **fun** и применения, то получится System F (Жирап — Рейнольдс), вывод типов в которой неразрешим (Wells, 1993).

- Если разрешить использовать кванторы для **fun** и применения, то получится System F (Жирап — Рейнольдс), вывод типов в которой неразрешим (Wells, 1993).
- Если разрешить кванторы для переменной в **fix**, то система также становится неразрешимой (Kfoury, Tiuryn и Urzyczyn, 1993).

- Если разрешить использовать кванторы для **fun** и применения, то получится System F (Жирар — Рейнольдс), вывод типов в которой неразрешим (Wells, 1993).
- Если разрешить кванторы для переменной в **fix**, то система также становится неразрешимой (Kfoury, Tiuryn и Urzyczyn, 1993).
- Есть много работ по расширению полиморфности системы с сохранением вывода типов (Kfoury и Tiuryn, 1992 и др.)

Кто все эти люди?



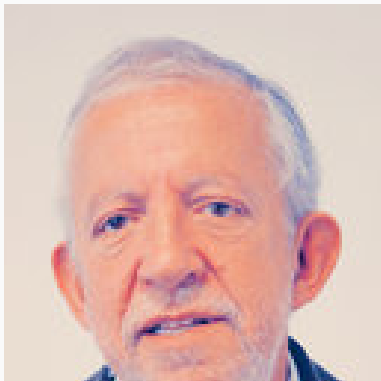
Роджер Хиндли
(1939)

J. Roger Hindley
(1969), The principal
type-scheme of an
object in combinatory
logic



Робин Милнер
(1934–2010)

Robin Milner (1978),
A theory of type
polymorphism
in programming



Луис Дамас

Luís Damas and Robin
Milner (1982),
Principal type-schemes
for functional
programs





Современный взгляд на вывод типов

- Одна система – один алгоритм?

Первый шаг: абстрагирование алгоритма вывода типов

- Одна система – один алгоритм?
- Системы типов всегда расширяются.

- Одна система – один алгоритм?
- Системы типов всегда расширяются.
- Насколько жёстко связаны язык программирования (его синтаксис и семантика) и его система типов?

- Одна система – один алгоритм?
- Системы типов всегда расширяются.
- Насколько жёстко связаны язык программирования (его синтаксис и семантика) и его система типов?
- Более абстрактный алгоритм может оказаться более устойчивым к изменениям.

- Одна система – один алгоритм?
- Системы типов всегда расширяются.
- Насколько жёстко связаны язык программирования (его синтаксис и семантика) и его система типов?
- Более абстрактный алгоритм может оказаться более устойчивым к изменениям.
- Как извлечь из терма и представить только необходимую информацию?

- Система типов
- Конкретные ограничения (constraints)
- Язык для записи ограничений (логика!)

- Первый этап алгоритма: генерация ограничений (constraints generator).

- Первый этап алгоритма: генерация ограничений (constraints generator).
- Второй этап алгоритма: разрешение ограничений (constraints solver).




- Первый этап алгоритма: генерация ограничений (constraints generator).
- Второй этап алгоритма: разрешение ограничений (constraints solver).





Декомпозиция алгоритма (его модульность) облегчает обоснование корректности.

Этот подход широко распространён в современной теории типов

- **CLP(X)** – J. Jaffar, J.-L. Lassez (1987). Constraint logic programming.
- **HM(X)**
 - M. Odersky, M. Sulzmann, M. Wehr (1999). Type Inference with Constrained Types.
 - Advanced Topics in Types and Programming Languages (ed. by Benjamin C. Pierce). The MIT Press, 2004. 588 pp. (*chapter 10*, The essence of ML type inference, Pottier&Rémy, <http://crystal.inria.fr/attapl/>)
- **OutsiderIn(X)** – D. Vytiniotis, S. Peyton Jones, T. Schrijvers, M. Sulzmann (2011). OutsiderIn(x) modular type inference with local assumptions.

Список литературы

-  Hindley, R. (1969). “The Principal Type-Scheme of an Object in Combinatory Logic”. в: *Transactions of the American Mathematical Society* 146, с. 29–60.
-  Milner, Robin (1978). “A theory of type polymorphism in programming”. в: *Journal of Computer and System Sciences* 17.3, с. 348 – 375.
-  Damas, Luis и Robin Milner (1982). “Principal Type-schemes for Functional Programs”. в: *Proceedings of the 9th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '82. Albuquerque, New Mexico: ACM, с. 207–212.

-  Jaffar, J. и J.-L. Lassez (1987). “Constraint Logic Programming”. В: *Proceedings of the 14th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*. POPL '87. Munich, West Germany: ACM, с. 111–119.
-  Kfoury, A. J. и J. Tiuryn (июнь 1992). “Type Reconstruction in Finite Rank Fragments of the Second-order lambda-calculus”. В: *Inf. Comput.* 98.2, с. 228–257.
-  Kfoury, A. J., J. Tiuryn и P. Urzyczyn (апр. 1993). “Type Reconstruction in the Presence of Polymorphic Recursion”. В: *ACM Trans. Program. Lang. Syst.* 15.2, с. 290–311.
-  Wells, J. B. (1993). *Typability and Type Checking in the Second-Order Lambda-Calculus Are Equivalent and Undecidable*. тех. отч. Boston, MA, USA.



Odersky, Martin, Martin Sulzmann и Martin Wehr (янв. 1999).

“Type Inference with Constrained Types”. в: *Theor. Pract. Object Syst.* 5.1, с. 35–55.



Pierce, Benjamin C. (2004). *Advanced Topics in Types and Programming Languages*. The MIT Press.



Dowek, Gilles и Jean-Jacques Lévy (2011). *Introduction to the Theory of Programming Languages*. Имеется русский перевод: Введение в теорию языков программирования. ДМК Пресс, 2013. 134 с.



Vytiniotis, Dimitrios и др. (сент. 2011). “Outsidein(x) Modular Type Inference with Local Assumptions”. в: *J. Funct. Program.* 21.4-5.

Доступно по адресу

<https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/jfp-outsidein.pdf>, с. 333–412.