

Федеральное государственное автономное образовательное учреждение  
высшего образования «Национальный исследовательский университет  
«Высшая школа экономики»

Факультет Санкт-Петербургская школа физико-математических и  
компьютерных наук

Департамент информатики

Основная образовательная программа

Прикладная математика и информатика

Кайсин Илья Сергеевич

## Задача обитаемости в системах типов низкого ранга

Выпускная квалификационная работа

Допущена к защите.

Зав. кафедрой:

д. ф.-м. н., профессор Омельченко А. В.

Научный руководитель:

к. ф.-м. н., профессор Москвин Д. Н.

Рецензент:

Пеленицын А. М

Санкт-Петербург  
2019

# Оглавление

1. Система типов $\lambda_{\wedge\eta}$	3
1.1. Подтипизация . . . . .	3
1.2. Существенность $\eta$ правила . . . . .	7
2. Населяющий алгоритм	8
2.1. Населяющий алгоритм для $\lambda_{\wedge}$ . . . . .	8
2.2. Населяющий алгоритм для $\lambda_{\wedge\eta}$ . . . . .	9
2.3. Отличия алгоритмов . . . . .	9
3. Свойства алгоритма	10
3.1. Корректность . . . . .	10
3.2. Полнота . . . . .	10
3.3. Завершаемость . . . . .	13
Список литературы	15

## 1. Система типов $\lambda_{\wedge\eta}$

Система типов  $\lambda_{\wedge\eta}$  отличается от просто типизированного лямбда исчисления новым типовым конструктором:  $\wedge$ , соответствующим пересечению типов. Эту операцию можно понимать в теоретико-множественном смысле: типом  $\sigma \wedge \tau$  типизируются такие и только такие термы, которые типизируются и  $\sigma$ , и  $\tau$ . Правила вывода, соответствующие этому поведению, обозначаются  $(I\wedge)$  и  $(E\wedge)$  (введение пересечения и элиминация пересечения соответственно).

Кроме того, вводится ещё одно дополнительное правило  $(\eta)$ , позволяющее проводить эта-экспансию.

Таким образом, система состоит из следующих правил вывода:  $Ax$ ,  $I \rightarrow$ ,  $E \rightarrow$ ,  $I\wedge$ ,  $E\wedge$ ,  $\eta$ .

$$\begin{array}{c} \frac{}{\Gamma, x: \tau \vdash x: \tau} (Ax) \\ \frac{\Gamma, x: \sigma \vdash M: \tau}{\Gamma \vdash (\lambda x.M): \sigma \rightarrow \tau} (I \rightarrow) \\ \frac{\Gamma \vdash M: \sigma \rightarrow \tau \quad \Gamma \vdash N: \sigma}{\Gamma \vdash (MN): \tau} (E \rightarrow) \\ \frac{\Gamma \vdash M: \sigma \quad \Gamma \vdash M: \tau}{\Gamma \vdash M: \sigma \wedge \tau} (I\wedge) \\ \frac{\Gamma \vdash M: \sigma \wedge \tau}{\Gamma \vdash M: \sigma} (E\wedge) \\ \frac{\Gamma \vdash M: \sigma}{\Gamma \vdash (\lambda x.Mx): \sigma} (\eta) \end{array} \quad (1)$$

### 1.1. Подтипизация

Определим на типах отношение подтипизации. С теоретико-множественной точки зрения подтипизация соответствует отношению «быть подмножеством»: если терм типизируется  $\sigma$ , то он типизируется всеми надтипами  $\sigma$ , то есть такими  $\tau$ , что  $\sigma \leq \tau$ . Отношение определим следующими аксиомами и правилами, аналогично определению из [3].

$$\begin{aligned}
& \overline{\sigma \leq \sigma} \quad (A1) \\
& \overline{\sigma \leq \sigma \wedge \sigma} \quad (A2) \\
& \overline{\sigma \wedge \tau \leq \sigma} \quad (A3) \\
& \overline{\sigma \wedge \tau \leq \tau} \quad (A4) \\
& \overline{(\sigma \rightarrow \tau_1) \wedge (\sigma \rightarrow \tau_2) \leq \sigma \rightarrow (\tau_1 \wedge \tau_2)} \quad (A5) \\
& \frac{\sigma \leq \sigma' \quad \tau \leq \tau'}{\sigma \wedge \tau \leq \sigma' \wedge \tau'} \quad (R1) \\
& \frac{\sigma \leq \sigma' \quad \tau \leq \tau'}{\sigma' \rightarrow \tau \leq \sigma \rightarrow \tau'} \quad (R2) \\
& \frac{\tau_1 \leq \tau_2 \quad \tau_2 \leq \tau_3}{\tau_1 \leq \tau_3} \quad (R3)
\end{aligned} \tag{2}$$

В [2] показано, что правило  $(\eta)$  может быть заменено следующим правилом:

$$\frac{\Gamma \vdash M : \sigma \quad \sigma \leq \tau}{\Gamma \vdash M : \tau} (\leq)$$

На самом деле, поскольку  $\sigma \wedge \tau \leq \sigma$  и  $\sigma \wedge \tau \leq \tau$ , правило  $(E\wedge)$  также избыточно. Таким образом, правила в этой системе следующие:

$$\begin{aligned}
& \overline{\Gamma, x : \tau \vdash x : \tau} \quad (Ax) \\
& \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x.M) : \sigma \rightarrow \tau} \quad (I \rightarrow) \\
& \frac{\Gamma \vdash M : \sigma \rightarrow \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash (MN) : \tau} \quad (E \rightarrow) \\
& \frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash M : \tau}{\Gamma \vdash M : \sigma \wedge \tau} \quad (I\wedge) \\
& \frac{\Gamma \vdash M : \sigma \quad \sigma \leq \tau}{\Gamma \vdash M : \tau} (\leq)
\end{aligned} \tag{3}$$

Полученную систему будем называть  $\lambda_{\wedge \leq}$ . Она эквивалентна  $\lambda_{\wedge \eta}$  в смысле типизации: утверждение типизации, верное в системе  $\lambda_{\wedge \leq}$ , верно в  $\lambda_{\wedge \eta}$  и наоборот. Поэтому в контексте нашей задачи эти две системы полностью взаимозаменяемы.

Определим отношение эквивалентности на типах следующим образом.

**Определение 1.**  $\sigma \sim \tau \iff \sigma \leq \tau$  и  $\tau \leq \sigma$

Благодаря правилу  $(\leq)$ , для эквивалентных типов верно следующее утверждение:

**Лемма 1.1.** Множества термов, типизируемых эквивалентными типами, в точности совпадают.

В частности, это означает, что переход к эквивалентному типу не влияет на его обитаемость.

Легко видеть, что отношение « $\sim$ » является отношением конгруэнтности (в смысле [1]) а именно, верна следующая Лемма:

**Лемма 1.2.** Пусть  $\alpha, \beta, \gamma, \delta$  – типы. Если  $\alpha \sim \beta$  и  $\gamma \sim \delta$ , то  $(\alpha \rightarrow \gamma) \sim (\beta \rightarrow \delta)$ , а также  $(\alpha \wedge \gamma) \sim (\beta \wedge \delta)$ .

Для удобства введём следующее обозначение:

**Определение 2.**

$$\hat{\tau} = \{\tau_i \mid \tau_1 \wedge \dots \wedge \tau_k = \tau$$

и  $\tau_i$  не является пересечением ни для какого  $i\}$

То есть  $\hat{\tau}$  – множество, полученное разделением всех пересечений на верхнем уровне  $\tau$ .

В [3] показана следующая эквивалентность:

**Лемма 1.3.**  $\alpha \rightarrow \beta \wedge \gamma \sim (\alpha \rightarrow \beta) \wedge (\alpha \rightarrow \gamma)$

Применяя её многократно, мы можем привести тип к нормальной форме. А именно, введём операцию « $*$ » следующим образом:

**Определение 3.**

$$\alpha^* = \alpha$$

если  $\alpha$  – типовая переменная

$$(\sigma \wedge \tau)^* = \sigma^* \wedge \tau^*$$

$$(\sigma \rightarrow \tau)^* = \bigwedge_{\varphi \in \hat{\tau}^*} \sigma \rightarrow \varphi$$

Например,  $(\alpha \rightarrow \gamma \wedge (\beta \rightarrow \gamma \wedge \delta))^* = (\alpha \rightarrow \beta \rightarrow \gamma) \wedge (\alpha \rightarrow \beta \rightarrow \delta) \wedge (\alpha \rightarrow \gamma)$ .

Операция « $*$ » переводит тип в эквивалентный.

**Лемма 1.4.** Для любого типа  $\tau$ :  $\tau \sim \tau^*$

**Определение 4.** Тип  $\tau$  находится в нормальной форме, если  $\tau^* = \tau$ .

Общий вид типа в нормальной форме после применения « $*$ » описывается следующей леммой:

**Лемма 1.5.** Для любого типа  $\tau$ :  $\tau^* = \bigwedge_i (\varphi_{i1} \rightarrow \dots \rightarrow \varphi_{ik_i} \rightarrow \alpha_i)$ , где  $\alpha_i$  – типовая переменная.

**Следствие 1.5.1.** *Типы в нормальной форме без пересечений на верхнем уровне — в точности типы вида  $\dots \rightarrow \alpha$ , где  $\alpha$  — некоторая типовая переменная.*

В [3] показано, что отношение  $(\leq)$  рекурсивно и существует алгоритм, позволяющий определить для двух типов  $\alpha$  и  $\beta$  верно ли, что  $\alpha \leq \beta$ :

**Алгоритм 1.**

$$\begin{aligned}
&\alpha \leq \beta = (\alpha == \beta) \\
&\text{если } \alpha \text{ и } \beta \text{ — типовые переменные} \\
&\sigma \leq \tau = \text{False} \\
&\text{если один из типов — переменная, а другой тип стрелочный} \\
&(\sigma_1 \rightarrow \tau_1) \leq (\sigma_2 \rightarrow \tau_2) = (\tau_1 \leq \tau_2) \text{ AND } (\sigma_2 \leq \sigma_1) \\
&\sigma \leq \tau = \forall \tau_i \in \hat{\tau}^*: \exists \sigma_j \in \hat{\sigma}^*: \sigma_j \leq \tau_i
\end{aligned}$$

Введём ещё несколько обозначений.

**Определение 5.** *Обозначим множество всех подтипов  $\tau$  через  $\underline{\tau}$ , а множество всех его надтипов через  $\bar{\tau}$ . То есть  $\underline{\tau} = \{\sigma \mid \sigma \leq \tau\}$ ,  $\bar{\tau} = \{\sigma \mid \tau \leq \sigma\}$ .*

Далее будем считать, что надтипы и подтипы могут использоваться везде, где могут использоваться типы. При этом операции над типами «поднимаются» на уровень множеств. Так, например, выражение  $\alpha \rightarrow \bar{\beta} \rightarrow \underline{\gamma}$  следует понимать как множество  $\{\alpha \rightarrow \beta' \rightarrow \gamma' \mid \beta' \in \bar{\beta}, \gamma' \in \underline{\gamma}\}$ .

Утверждение о типизации, в котором фигурируют надтипы или подтипы, будем считать выводимым, если оно выводимо для некоторых элементов соответствующих множеств надтипов и подтипов.

Запись о вхождении типизации в контекст  $(x : T) \in \Gamma$  в случае, если в  $T$  фигурируют надтипы или подтипы, означает, что  $(x : \tau) \in \Gamma$  для некоторого  $\tau \in T$ .

Запись вида  $T_1 \leq T_2$ , где в  $T_i$  могут фигурировать подтипы и надтипы, означает, что  $\forall \tau_1 \in T_1, \tau_2 \in T_2 : \tau_1 \leq \tau_2$ .

**Лемма 1.6.** *Пусть  $\sigma \rightarrow \tau$  — тип в нормальной форме. Тогда  $\underline{\sigma \rightarrow \tau} = \varphi \wedge (\bar{\sigma} \rightarrow \underline{\tau})$ , для некоторого (возможно, пустого)  $\varphi$ .*

*Доказательство.* Включение  $\varphi \wedge (\bar{\sigma} \rightarrow \underline{\tau}) \subseteq \underline{\sigma \rightarrow \tau}$  почти очевидно:

$$\frac{\frac{\sigma \leq \bar{\sigma} \quad \underline{\tau} \leq \tau}{\bar{\sigma} \rightarrow \underline{\tau}} (R2)}{\varphi \wedge (\bar{\sigma} \rightarrow \underline{\tau}) \leq \sigma \rightarrow \tau} (A4)$$

Для включения  $\underline{\sigma \rightarrow \tau} \subseteq \varphi \wedge (\bar{\sigma} \rightarrow \underline{\tau})$  достаточно посмотреть на Алгоритм 1. Подтип  $\sigma \rightarrow \tau$  — это либо тип из  $(\bar{\sigma} \rightarrow \underline{\tau})$  (третий случай алгоритма), либо тип-пересечение, один из элементов которого — подтип  $\sigma \rightarrow \tau$ ; остальные элементы можно «переместить» в  $\varphi$  перестановкой.  $\square$

## 1.2. Существенность $\eta$ правила

Насколько добавление  $\eta$  - правило меняет систему? Система  $\lambda_{\wedge}$  существенно слабее  $\lambda_{\wedge\eta}$ , а именно, есть тип, обитаемый в системе без эта-правила и обитаемый в системе с ним.

**Лемма 1.7.** *Утверждение о типизации  $x : \alpha \rightarrow \beta \wedge \gamma \vdash x : \alpha \rightarrow \beta$  верно в  $\lambda_{\wedge\eta}$  но неверно в  $\lambda_{\wedge}$ .*

*Доказательство.* Докажем выводимость в системе  $\lambda_{\wedge\leq}$ :

$$\frac{x : \alpha \rightarrow \beta \wedge \gamma \vdash x : \alpha \rightarrow \beta \wedge \gamma \quad \alpha \rightarrow \beta \wedge \gamma \leq \alpha \rightarrow \beta}{x : \alpha \rightarrow \beta \wedge \gamma \vdash x : \alpha \rightarrow \beta}$$

Чтобы доказать, что утверждение неверно в системе  $\lambda_{\wedge}$ , предположим обратное. Рассмотрим последнее правило вывода, которое могло быть применено, чтобы получить данное утверждение.  $(I \rightarrow)$  и  $(E \rightarrow)$  не могли быть применены, поскольку они типизируют абстракцию и аппликацию соответственно, а  $x$  — типовая переменная. Правило  $(Ax)$  требует наличия соответствующей типизации в контексте, правило  $(I\wedge)$  приписывает терму тип-пересечение, а не стрелочный тип.

Таким образом, единственным возможным правилом могло быть  $(E\wedge)$ :

$$\frac{\begin{array}{c} \vdots \\ x : \alpha \rightarrow \beta \wedge \gamma \vdash x : (\alpha \rightarrow \beta) \wedge \sigma \end{array}}{x : \alpha \rightarrow \beta \wedge \gamma \vdash x : \alpha \rightarrow \beta}$$

Какие правила вывода могли привести к данному утверждению о типизации? Только  $(I \rightarrow)$  и  $(E \rightarrow)$ . Однако легко видеть, что их «обратное» применение порождает утверждение вида  $x : \alpha \rightarrow \beta \wedge \gamma \vdash x : (\alpha \rightarrow \beta) \wedge \sigma$  для некоторого (возможно, пустого)  $\sigma$ , но утверждение такого вида уже встречалось ранее, и могло быть получено лишь с помощью  $(I \rightarrow)$  и  $(E \rightarrow)$ . Значит, дерева вывода не существует. □

**Лемма 1.8.** *Тип  $\delta \wedge (\alpha \rightarrow \beta \wedge \gamma) \rightarrow \delta \wedge (\alpha \rightarrow \beta)$  пуст в  $\lambda_{\wedge}$ , но содержит терм  $\text{id}$  в  $\lambda_{\wedge\eta}$ .*

*Доказательство.* Анализом возможных деревьев вывода, аналогичным рассуждениям в предыдущей лемме, получаем, что для существования типизации  $\vdash M : \delta \wedge (\alpha \rightarrow \beta \wedge \gamma) \rightarrow \delta \wedge (\alpha \rightarrow \beta)$  необходима и достаточна типизация  $x : \alpha \rightarrow \beta \wedge \gamma \vdash x : \alpha \rightarrow \beta$ . □

**Замечание 1.8.1.** *Тип  $(\alpha \rightarrow \beta \wedge \gamma) \rightarrow \alpha \rightarrow \beta$  обитает в обеих системах. В  $\lambda_{\wedge}$  он содержит  $\lambda a.b.a$ , но не  $\lambda a.a$ .*

## 2. Населяющий алгоритм

В [4] был представлен населяющий алгоритм для системы  $\lambda_{\wedge}$ . Однако этот алгоритм описан недостаточно полно, что выяснилось при его реализации. Кроме того, алгоритм и доказательство его корректности содержат ошибку. Данная работа устраняет эти неточности и ошибки. Алгоритм для  $\lambda_{\wedge \leq}$  является модификацией алгоритма из [4], поэтому сначала рассмотрим его (в исправленном варианте).

### 2.1. Населяющий алгоритм для $\lambda_{\wedge}$

**Алгоритм 2.** В процессе алгоритма решается система из нескольких задач:  $[\Gamma_1 \vdash X : \tau_1, \dots, \Gamma_n \vdash X : \tau_n]$ . Решением системы является терм  $X$ , удовлетворяющий всем утверждениям типизации одновременно. При этом поддерживается инвариант: все задачи системы имеют общий набор переменных в контексте (при этом одной и той же типовой переменной могут соответствовать разные типы в разных контекстах). Алгоритм заключается в применении следующих преобразований до тех пор, пока это возможно:

- 1. Один из типов  $\tau_i$  – пересечение ( $\tau_i = \sigma \wedge \rho$ ). Тогда  $i$ -я задача  $\Gamma_i \vdash X : \tau_i$  заменяется двумя:  $\Gamma_i \vdash X : \sigma$  и  $\Gamma_i \vdash X : \rho$ . Таким образом, размер системы увеличивается
- 2. Все типы  $\tau_i$  стрелочные ( $\tau_i = \sigma_i \rightarrow \rho_i$ ,  $i = 1 \dots n$ ). Тогда решение системы –  $X = \lambda x. X'$ , где  $X'$  – решение новой системы  $[(\Gamma_1, x : \sigma_1 \vdash X' : \rho_1), \dots, (\Gamma_n, x : \sigma_n \vdash X' : \rho_n)]$ . То есть в этом случае во всех типах редуцируется первый аргумент.
- 3 $_{\wedge}$ . Один из типов  $\tau_i$  – это типовая переменная. Тогда  $X$  не может быть абстракцией и должен быть (возможно пустой) аппликацией некоторой головной переменной, взятой из контекста, к термам. В этом случае необходимо недетерминированно выбрать из контекста головную переменную  $x$  и число  $k \geq 0$  таким образом, чтобы  $\Gamma_i \vdash \lambda z_1 \dots z_k. x z_1 \dots z_k : \rho_i^1 \rightarrow \dots \rightarrow \rho_i^k \rightarrow \tau_i$  для всех  $i = 1 \dots n$ . Тогда решение –  $X = x Z^1 \dots Z^k$ , где  $Z^i$  – решение системы  $[(\Gamma_1 \vdash Z^i : \rho_i^1), \dots, (\Gamma_n \vdash Z^i : \rho_i^k)]$ . Здесь  $k$  систем независимы и могут быть решены параллельно.

Если ни одно из преобразований применить невозможно, то система не имеет решения. «Точка выхода» из алгоритма – тот случай в преобразовании 3, при котором  $k = 0$ , при этом решением является переменная.

В данном алгоритме не совсем ясным является шаг 3 $_{\wedge}$ : как именно выбираются  $\rho_i^j$ . Типизация  $\Gamma_i \vdash \lambda z_1 \dots z_k. x z_1 \dots z_k : \rho_i^1 \rightarrow \dots \rightarrow \rho_i^k \rightarrow \tau_i$  не гарантирует того, что тип  $x$  в контексте  $\Gamma_i$  обязан быть  $\rho_i^1 \rightarrow \dots \rightarrow \rho_i^k \rightarrow \tau_i$ .



## 2.2. Населяющий алгоритм для $\lambda_{\wedge\eta}$

Алгоритм для системы с  $\eta$ -правилом во многом повторяет алгоритм для  $\lambda_{\wedge}$ . Принципиальное отличие заключается в преобразовании 3.

**Алгоритм 3.** *Поддерживается система из нескольких задач:  $\Gamma_1 \vdash X : \tau_1, \dots, \Gamma_n \vdash X : \tau_n$ . До тех пор, пока это возможно, выполняется один из четырёх шагов:*

- 0. *Один из типов  $\tau_i$  не находится в нормальной форме. Тогда заменим  $\tau_i$  на  $\tau_i^*$ .*
- *Шаги 1 и 2 аналогичны Алгоритму 2.*
- $3_{\wedge\eta}$ . *Один из типов  $\tau_i$  – это типовая переменная. Тогда  $X$  не может быть абстракцией и должен быть (возможно, пустой) аппликацией некоторой головной переменной, взятой из контекста, к термам. В этом случае необходимо недетерминированно выбрать переменную  $x$  и число  $k \geq 0$  таким образом, чтобы  $\Gamma_i \vdash x : \rho_i^1 \rightarrow \dots \rightarrow \rho_i^k \rightarrow \tau_i$  для всех  $i = 1 \dots n$ . Тогда решение –  $X = xZ^1 \dots Z^k$ , где  $Z^i$  – решение системы  $(\Gamma_1 \vdash Z^i : \rho_1^i), \dots, (\Gamma_n \vdash Z^i : \rho_n^i)$ . Здесь  $k$  систем независимы и могут быть решены параллельно.*

*Следствие 3.1.3 описывает, как именно устроен недетерминированный выбор  $x$ ,  $k$  и  $\rho_i^j$ .*

*Аналогично алгоритму 2, если ни одно из преобразований применить невозможно, система не имеет решения.*

## 2.3. Отличия алгоритмов

В [4] в качестве населяющего алгоритма для системы  $\lambda_{\wedge}$  приведён алгоритм, почти совпадающий с Алгоритмом 3, описанным выше. Однако на следующем примере можно убедиться, что данный алгоритм не является полным в системе  $\lambda_{\wedge}$ .

Пусть задача –  $p : \alpha \rightarrow \beta \wedge \gamma, q : \alpha \vdash X : \beta$ . Алгоритм должен произвести преобразование  $3_{\wedge\eta}$ , но в контексте нет такого  $x$ , что  $p : \alpha \rightarrow \beta \wedge \gamma, q : \alpha \vdash x : \dots \rightarrow \beta$ , поскольку, согласно Лемме 1.7, утверждение о типизации  $p : \alpha \rightarrow \beta \wedge \gamma \vdash p : \alpha \rightarrow \beta$  неверно в  $\lambda_{\wedge}$ . Поэтому алгоритм завершится, не найдя решение  $X = pq$ . Но преобразование  $3_{\wedge}$  применить можно:  $p : \alpha \rightarrow \beta \wedge \gamma, q : \alpha \vdash \lambda z.pz : \alpha \rightarrow \beta$ . Здесь  $\lambda$  позволяет применить  $(I \rightarrow)$  и удалить  $\rightarrow$ .

### 3. Свойства алгоритма

В этом разделе будут доказаны свойства Алгоритма 3 для системы  $\lambda_{\wedge\eta}$ . А именно, его корректность (soundness), полнота (completeness), завершаемость (termination) на типах ранга  $\leq 2$ .

#### 3.1. Корректность

Корректность алгоритма означает, что ответ, данный им, удовлетворяет входным условиям. То есть если некий терм был найден, то он действительно типизируется заданным типом.

**Теорема 1** (Soundness). *Если алгоритм находит терм  $M$  для входного типа  $\tau$ , то  $\vdash M : \tau$*

*Доказательство.* Усилим утверждение: докажем, что алгоритм корректно решает систему задач. Доказательство – индукция по количеству шагов алгоритма. В базе индукции используется аксиома  $(Ax)$ . В переходах индукции в шагах 1, 2 и  $3_{\wedge\eta}$  используются правила вывода  $(I\wedge)$ ,  $(I\rightarrow)$  и  $(E\rightarrow)$  соответственно.  $\square$

#### 3.2. Полнота

Полнота – гораздо более содержательное свойство. Оно означает, что если существует терм заданного типа, то алгоритм обязательно найдёт или его, или другой терм этого типа. В нашем случае это будет терм в так называемой длинной форме. Далее будет доказан ряд лемм, имеющих отношение к преобразованию  $3_{\wedge\eta}$  и проясняющих его смысл.

**Лемма 3.1.** *Пусть  $x$  – переменная,  $M_i$  – термы,  $\tau$  – тип в нормальной форме без пересечений на верхнем уровне (см. следствие 1.5.1). Тогда равносильно:*

$$\begin{aligned} \Gamma \vdash xM_1 \dots M_k : \tau \\ \iff \\ \Gamma \vdash x : \alpha_1 \rightarrow \dots \rightarrow \alpha_k \rightarrow \tau \text{ и } \Gamma \vdash M_i : \alpha_i \text{ для } i = 1 \dots k \end{aligned}$$

*Доказательство.* Докажем равносильность в обе стороны.

- $\Leftarrow$ ) применяя правило  $(E\rightarrow)$   $k$  раз, получаем в точности необходимое утверждение о типизации.
- $\Rightarrow$ ) Если  $k = 0$ , то утверждение леммы очевидно. Иначе докажем индукцией по размеру дерева вывода (анализируя дерево с конца, аналогично доказательству

Леммы 1.7). Рассуждения будет удобнее провести в системе  $\lambda_{\wedge \leq} (3)$ , поскольку в ней меньше правил вывода.

Какое правило вывода могло быть применено последним, чтобы получить утверждение  $\Gamma \vdash xM_1 \dots M_k : \tau$ ? Это не могло быть  $(Ax)$ , так как оно типизирует переменную; это не могло быть  $(I \rightarrow)$ , так как оно типизирует лямбда абстракцию; это не могло быть  $(I \wedge)$ , поскольку в  $\tau$  нет пересечений на верхнем уровне. Таким образом, могли быть применены только два правила:  $(\leq)$  или  $(E \rightarrow)$ .

Согласно Лемме 1.5.1,  $\tau$  имеет вид  $\rho_1 \rightarrow \dots \rightarrow \rho_m \rightarrow \beta$ , где  $\beta$  — типовая переменная. По Лемме 1.6, подтип  $\tau$  тогда обязан иметь вид  $\varphi_1 \wedge (\overline{\rho_1} \rightarrow \varphi_2 \wedge (\overline{\rho_2} \rightarrow \varphi_3 \wedge \dots (\overline{\rho_k} \rightarrow \beta) \dots))$ . К такой типизации можно или снова применить  $(\leq)$ , получив нечто такой же формы (поскольку  $\underline{\tau} = \underline{\tau}$ ) или применить  $(I \wedge)$ , получив в одной из веток нечто той же формы. Так или иначе, в определённый момент будет применено  $(E \rightarrow)$  к  $\underline{\tau}$ .

Таким образом, в общем случае дерево вывода имеет следующий вид:

$$\frac{\begin{array}{c} \vdots \\ \Gamma \vdash xM_1 \dots M_{k-1} : \alpha_k \rightarrow \underline{\tau} \end{array} \quad \begin{array}{c} \vdots \\ \Gamma \vdash M_k : \alpha_k \end{array}}{\Gamma \vdash xM_1 \dots M_k : \underline{\tau}} (E \rightarrow)$$

$$\begin{array}{c} \vdots \\ \vdots (\leq), (I \wedge) \\ \Gamma \vdash xM_1 \dots M_k : \tau \end{array}$$

Поскольку  $\alpha_k \rightarrow \underline{\tau} \leq \alpha_k \rightarrow \tau$ , верно утверждение о типизации  $\Gamma \vdash xM_1 \dots M_{k-1} : \alpha_k \rightarrow \tau$ , которое подходит под условия леммы. Поэтому, применив те же рассуждения ещё  $k - 1$  раз, получим  $\Gamma \vdash x : \alpha_1 \rightarrow \underline{\alpha_2 \rightarrow \dots \rightarrow \alpha_k \rightarrow \tau}$ , а также  $\Gamma \vdash M_i : \alpha_i$  для  $i = 1 \dots k$ .

Легко видеть, что  $\alpha_1 \rightarrow \underline{\alpha_2 \rightarrow \dots \rightarrow \alpha_k \rightarrow \tau} \leq \alpha_1 \rightarrow \dots \rightarrow \alpha_k \rightarrow \tau$ , поэтому мы можем применить  $(\leq)$  и получить необходимую типизацию.

□

**Замечание 3.1.1.** Условие о том, что  $\tau$  — тип без пересечений на верхнем уровне, существенно. Так, например, для  $\Gamma = \{x : (\alpha \rightarrow \beta) \wedge (\gamma \rightarrow \delta), M : (\alpha \wedge \gamma)\}$ ,  $\Gamma \vdash xM : \beta \wedge \gamma$ , но при этом  $\Gamma \not\vdash x : \dots \rightarrow \beta \wedge \gamma$ .

**Замечание 3.1.2.** Условие о том, что  $\tau$  — тип в нормальной форме, существенно. Так, например, для  $\Gamma = \{x : (\alpha \rightarrow \varphi \rightarrow \beta) \wedge (\gamma \rightarrow \varphi \rightarrow \delta), M : (\alpha \wedge \gamma)\}$ ,  $\Gamma \vdash xM : \varphi \rightarrow (\beta \wedge \gamma)$ , но при этом  $\Gamma \not\vdash x : \dots \rightarrow \varphi \rightarrow (\beta \wedge \gamma)$ .

**Следствие 3.1.1.** В условиях предыдущей леммы, при выполнении любой из равносильных частей,  $\Gamma \ni x : \underline{\alpha_1 \rightarrow \dots \rightarrow \alpha_k \rightarrow \tau}$ ,

*Доказательство.* Для доказательства достаточно провести ещё один шаг в рассуждениях леммы, за тем лишь исключением, что вместо  $(E \rightarrow)$  должно быть применено  $(Ax)$ , что и гарантирует наличие нужной типизации в контексте.  $\square$

**Следствие 3.1.2.** *В условиях предыдущей леммы, при выполнении любой из равносильных частей, в контексте  $\Gamma$  есть типизация  $x: \sigma$  такая, что  $\hat{\sigma}^* \ni \bar{\alpha}_1 \rightarrow \dots \rightarrow \bar{\alpha}_k \rightarrow \tau$ .*

*В частности, если  $\tau$  — типовая переменная, то  $\hat{\sigma}^* \ni \bar{\alpha}_1 \rightarrow \dots \rightarrow \bar{\alpha}_k \rightarrow \tau$ .*

*Доказательство.* Воспользуемся предыдущим следствием и тем наблюдением, что  $\alpha_1 \rightarrow \dots \rightarrow \alpha_k \rightarrow \tau = \varphi_1 \wedge (\bar{\alpha}_1 \rightarrow \varphi_2 \wedge (\bar{\alpha}_2 \rightarrow \varphi_3 \wedge \dots (\bar{\alpha}_k \rightarrow \tau) \dots))$   $\square$

Следующее следствие проясняет, как с алгоритмической точки зрения устроено преобразование  $\mathcal{Z}_{\wedge\eta}$  в Алгоритме 3. А именно, как выбрать  $x$ ,  $k$  и  $\rho_i^j$ .

**Следствие 3.1.3.** *Пусть в шаге  $\mathcal{Z}_{\wedge\eta}$  Алгоритма 3,  $\tau_i$  — типовая переменная. Тогда для того, чтобы выбрать  $x$ ,  $k$  и  $\rho_i^j$  ( $j = 1 \dots k$ ), достаточно перебрать все элементы  $(x: \sigma)$  контекста такие, что  $\hat{\sigma}^*$  содержит тип, заканчивающийся на  $\tau_i$ , то есть имеющий вид  $\rho_i^1 \rightarrow \dots \rightarrow \rho_i^k \rightarrow \tau_i$ .*

*Для того, чтобы проверить, что выбранный  $x$  удовлетворяет всем остальным задачам, а также чтобы подобрать  $\rho_t^j$  (для всех  $t \neq i$  и  $j = 1 \dots k$ ), достаточно найти среди элементов  $\hat{\sigma}^*$  (где  $\Gamma_t \ni (x: \sigma)$ ) типы, у которых после «отщепления»  $k$  аргументов остаётся подтип  $\tau_t$ , то есть имеющие вид  $\rho_t^1 \rightarrow \dots \rightarrow \rho_t^k \rightarrow \tau_t'$ , где  $\tau_t' \leq \tau_t$ .*

**Теорема 2 (Completeness).** *Если у системы задач существует решение, то Алгоритм 3 найдёт его или другое решение.*

*Доказательство.* Доказательство — индукция по размеру наибольшего типа в задачах системы. Пусть дана система  $T = [\Gamma_1 \vdash X: \tau_1, \dots, \Gamma_n \vdash X: \tau_n]$ , а  $M$  — её решение.

Если один из  $\tau_i$  — тип-пересечение, алгоритм разобьёт его на два. При этом решение  $M$  останется решением системы, но уменьшится размер задач системы, поэтому можно применить индукционное предположение. Аналогичные рассуждения можно провести в случае, если один из  $\tau_i$  не находится в нормальной форме (при этом, чтобы размеры типов уменьшились, нужно снять пересечения, сделав несколько раз преобразование 1).

Пусть ни один из типов  $\tau_i$  не является пересечением. Рассмотрим, как может быть устроено  $M$ .

- $M = \lambda x. M'$ .

Тогда все типы в задачах должны быть стрелочные:  $\tau_i = \alpha_i \rightarrow \beta_i$ . Алгоритм перейдёт к системе  $T = [\Gamma_1, x: \alpha_1 \vdash X': \beta_1, \dots, \Gamma_n, x: \alpha_n \vdash X': \beta_n]$ , у которой существует решение  $M'$ .

- $M = xM_1M_2 \dots M_k$ .

- Пусть все типы в задачах стрелочные:  $\tau_i = \alpha_i \rightarrow \beta_i$ . Тогда алгоритм, аналогично предыдущему пункту, перейдёт к системе  $T = [\Gamma_1, y: \alpha_1 \vdash X': \beta_1, \dots, \Gamma_n, y: \alpha_n \vdash X': \beta_n]$ . решение которой —  $xM_1M_2 \dots M_ky$ .
- Пусть один из типов в задачах — переменная:  $\tau_i = \beta$ . По Лемме 3.1, верно следующее:  $\Gamma_i \vdash x: \alpha_i^1 \rightarrow \dots \rightarrow \alpha_i^k \rightarrow \tau_i$  и  $\Gamma_i \vdash M_j: \alpha_i^j$  для  $i = 1 \dots n, j = 1 \dots k$ . Поэтому, выбрав  $k$  и  $x$ , алгоритм перейдёт к системам  $T_j = [(\Gamma_1 \vdash X_j: \overline{\alpha_1^j}), \dots, (\Gamma_n \vdash X_j: \overline{\alpha_n^j})]$ .  $M_j$  является решением  $j$ -й системы.

□

### 3.3. Завершаемость

В предыдущих разделах было доказано, что Алгоритм 3 выдаёт корректное решение, если оно существует. Однако от алгоритма требуется, чтобы он завершался за конечное число шагов, даже если тип пустой. В текущей системе типов такие гарантии получить невозможно: задача обитаемости неразрешима [6]. Для обеспечения завершаемости можно ограничить множество входных типов, а именно ввести ограничение на их ранг.

Понятие ранга вводится в [5] и определяется как максимальная глубина вложенности « $\wedge$ » в качестве левого аргумента « $\rightarrow$ ». Более формальное определение следующее:

**Определение 6.**

$$rank(\tau) = 0$$

если  $\tau$  — тип без пересечений

$$rank(\sigma \wedge \tau) = \max(1, rank(\sigma), rank(\tau))$$

$$rank(\sigma \rightarrow \tau) = \max(1 + rank(\sigma), rank(\tau))$$

если  $rank(\sigma) > 0$  или  $rank(\tau) > 0$

Так, например,  $rank(\varphi \wedge (\tau \rightarrow (\alpha \wedge \beta) \rightarrow \gamma)) = 1$

**Лемма 3.2.** Если  $rank(\rho_1 \rightarrow \dots \rightarrow \rho_k \rightarrow \tau) = r > 0$ , то  $rank(\rho_i) < r$  для всех  $i = 1 \dots k$ .

Следующее доказательство во многом повторяет доказательство завершаемости из [4].

**Теорема 3 (Termination).** Пусть  $\tau$  — тип с рангом не больше двух ( $rank(\tau) \leq 2$ ). Тогда Алгоритм 3 завершается на входе  $\tau$ .

*Доказательство.* Пусть  $\text{rank}(\tau) = 0$ , что равносильно тому, что в  $\tau$  нет пересечений. Тогда алгоритм завершится после линейного числа преобразований 2 (и затем одного  $3_{\wedge\eta}$ ).

Проследим за рангами целевых типов  $\tau_i$  и типов в контекстах  $\Gamma_i$ .

В преобразованиях 0 и 1 контексты не изменяются, а ранги типов в задачах не увеличиваются.

В преобразовании 2 ранг типов в задачах не увеличивается. А в контексты попадают типы  $\sigma_i$ , находящиеся слева от стрелки в типе  $\tau_i = \sigma_i \rightarrow \rho_i$ , а значит, имеющие или нулевой ранг, или хотя бы на 1 меньший ранг, чем  $\tau_i$  (см. Лемму 3.2).

В преобразовании  $3_{\wedge\eta}$  контексты не изменяются, а в целевыми становятся типы  $\alpha_i^j$ , являющиеся аргументами в типах из контекстов, а значит, имеющие на 1 меньший ранг (или нулевой).

Из вышеизложенных соображений следует, что типы в контекстах всегда имеют ранг  $\leq 1$ , поэтому преобразование  $3_{\wedge\eta}$  порождает задачи, в которых целевые типы имеют нулевой ранг. Эти задачи решаются за линейное число шагов. Преобразования 0 – 2 структурно уменьшают целевые типы, поэтому применяются лишь конечное число раз.

□

## Список литературы

- [1] Barendregt H. P. Handbook of Logic in Computer Science (Vol. 2) / Ed. by S. Abramsky, Dov M. Gabbay, S. E. Maibaum. — New York, NY, USA : Oxford University Press, Inc., 1992. — P. 117–309. — Access mode: <http://dl.acm.org/citation.cfm?id=162552.162561>.
- [2] Hindley J. Roger. Types with intersection: An introduction. — Vol. 4, no. 5. — P. 470–486. — Access mode: <https://doi.org/10.1007/BF01211394>.
- [3] Hindley J. R. The simple semantics for Coppo-Dezani-Sallé types // International Symposium on Programming / Ed. by Mariangiola Dezani-Ciancaglini, Ugo Montanari. — Berlin, Heidelberg : Springer Berlin Heidelberg, 1982. — P. 212–226.
- [4] Kuśmierek Dariusz. The Inhabitation Problem for Rank Two Intersection Types // Typed Lambda Calculi and Applications / Ed. by Simona Ronchi Della Rocca. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2007. — P. 240–254.
- [5] Leivant Daniel. Polymorphic Type Inference // Proceedings of the 10th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages. — POPL '83. — New York, NY, USA : ACM. — P. 88–98. — event-place: Austin, Texas. Access mode: <http://doi.acm.org/10.1145/567067.567077>.
- [6] Urzyczyn Pawel. The Emptiness Problem for Intersection Types. — Vol. 64, no. 3. — P. 1195–1215. — Access mode: <http://www.jstor.org/stable/2586625>.