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 $\lambda_{\wedge\eta}$

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 σ, и τ. Правила вывода, соответствующие этому поведению, обозначаются $(I\wedge)$ и $(E\wedge)$ (введение пересечения и элиминация пересечения соответственно).

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

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

$$
\begin{gathered}
\frac{}{\Gamma, x\colon \tau \vdash x\colon \tau}\ (Ax) \\[1ex]
\frac{\Gamma, x\colon \sigma \vdash M\colon \tau}{\Gamma \vdash (\lambda x.M)\colon \sigma \to \tau}\ (I \to) \\[1ex]
\frac{\Gamma \vdash M\colon \sigma \to \tau \quad \Gamma \vdash N\colon \sigma}{\Gamma \vdash (MN)\colon \tau}\ (E \to) \\[1ex]
\frac{\Gamma \vdash M\colon \sigma \quad \Gamma \vdash M\colon \tau}{\Gamma \vdash M\colon \sigma \wedge \tau}\ (I\wedge) \\[1ex]
\frac{\Gamma \vdash M\colon \sigma \wedge \tau}{\Gamma \vdash M\colon \sigma}\ (E\wedge) \\[1ex]
\frac{\Gamma \vdash M\colon \sigma}{\Gamma \vdash (\lambda x.Mx)\colon \sigma}\ (\eta)
\end{gathered}
\tag{1}
$$

1.1. Подтипизация

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

$$\frac{}{\sigma \leqslant \sigma}\ (A1)$$

$$\frac{}{\sigma \leqslant \sigma \wedge \sigma}\ (A2)$$

$$\frac{}{\sigma \wedge \tau \leqslant \sigma}\ (A3)$$

$$\frac{}{\sigma \wedge \tau \leqslant \tau}\ (A4)$$

$$\frac{}{(\sigma \to \tau_1) \wedge (\sigma \to \tau_2) \leqslant \sigma \to (\tau_1 \wedge \tau_2)}\ (A5) \qquad (2)$$

$$\frac{\sigma \leqslant \sigma' \quad \tau \leqslant \tau'}{\sigma \wedge \tau \leqslant \sigma' \wedge \tau'}\ (R1)$$

$$\frac{\sigma \leqslant \sigma' \quad \tau \leqslant \tau'}{\sigma' \to \tau \leqslant \sigma \to \tau'}\ (R2)$$

$$\frac{\tau_1 \leqslant \tau_2 \quad \tau_2 \leqslant \tau_3}{\tau_1 \leqslant \tau_3}\ (R3)$$

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

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

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

$$\frac{}{\Gamma, x\colon \tau \vdash x\colon \tau}\ (Ax)$$

$$\frac{\Gamma, x\colon \sigma \vdash M\colon \tau}{\Gamma \vdash (\lambda x.M)\colon \sigma \to \tau}\ (I \to)$$

$$\frac{\Gamma \vdash M\colon \sigma \to \tau \quad \Gamma \vdash N\colon \sigma}{\Gamma \vdash (MN)\colon \tau}\ (E \to) \qquad (3)$$

$$\frac{\Gamma \vdash M\colon \sigma \quad \Gamma \vdash M\colon \tau}{\Gamma \vdash M\colon \sigma \wedge \tau}\ (I\wedge)$$

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

Полученную систему будем называть $\lambda_{\wedge\leqslant}$. Она эквивалентна $\lambda_{\wedge\eta}$ в смысле типизации: утверждение типизации, верное в системе $\lambda_{\wedge\leqslant}$, 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. $\sigma \sim \tau \iff \sigma \leqslant \tau$ *и* $\tau \leqslant \sigma$

Благодаря правилу $(\leqslant)$, 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 $(\alpha \to \gamma) \sim (\beta \to \delta)$, а также $(\alpha \wedge \gamma) \sim (\beta \wedge \delta)$.*

Для удобства введём следующее обозначение:

***Определение* 2.**

$$\hat{\tau} = \{\tau_i \,|\tau_1 \wedge \cdots \wedge \tau_k = \tau \text{ и } \tau_i \text{ не является пересечением ни для какого } i\}$$

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

В [3] показана следующая экивалентность:

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

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

***Определение* 3.**

$$\begin{aligned} \alpha^* &= \alpha \\ &\quad \text{если } \alpha \text{ — типовая переменная} \\ (\sigma \wedge \tau)^* &= \sigma^* \wedge \tau^* \\ (\sigma \to \tau)^* &= \bigwedge_{\varphi \in \hat{\tau^*}} \sigma \to \varphi \end{aligned}$$

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

Операция «*» переводит тип в эквивалентный.

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

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

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

***Лемма* 1.5.** *Для любого типа τ: $\tau^* = \bigwedge_i (\varphi_{i1} \to \cdots \to \varphi_{ik_i} \to \alpha_i)$, где α_i - типовая переменная.*

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

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

Алгоритм 1**.**

$$\alpha \leqslant \beta = (\alpha \texttt{ == } \beta)$$

если α *и* β *— типовые переменные*

$$\sigma \leqslant \tau = False$$

если один из типов — переменная, а другой тип стрелочный

$$(\sigma_1 \to \tau_1) \leqslant (\sigma_2 \to \tau_2) = (\tau_1 \leqslant \tau_2) \texttt{ AND } (\sigma_2 \leqslant \sigma_1)$$

$$\sigma \leqslant \tau = \forall \tau_i \in \hat{\tau^*} \colon \exists \sigma_j \in \hat{\sigma^*} \colon \sigma_j \leqslant \tau_i$$

Введём ещё несколько обозначений.

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

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 $\alpha \to \bar{\beta} \to \underline{\gamma}$ следует понимать как множество $\{\alpha \to \beta' \to \gamma' \mid \beta' \in \bar{\beta}, \gamma' \in \underline{\gamma}\}$.

Утверждения о типизации, в которых фигурируют надтипы и подтипы, следует считать истинным, если оно истинно *для некоторых* элементов соответствующих можеств надтипов и подтипов.

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

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

$$\dfrac{\dfrac{\sigma \leqslant \bar{\sigma} \quad \underline{\tau} \leqslant \tau}{\bar{\sigma} \to \underline{\tau}}\ (R2)}{\varphi \wedge (\bar{\sigma} \to \underline{\tau}) \leqslant \sigma \to \tau}\ (A4)$$

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

1.2. Существенность η правила

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

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

Доказательство. Докажем истинность в системе $\lambda_{\wedge\leqslant}$:

$$\dfrac{\overline{x\colon \alpha \to \beta \wedge \gamma \vdash x\colon \alpha \to \beta \wedge \gamma} \quad \alpha \to \beta \wedge \gamma \leqslant \alpha \to \beta}{x\colon \alpha \to \beta \wedge \gamma \vdash x\colon \alpha \to \beta}$$

Чтобы доказать, что утверждение неверно в системе $\lambda_\wedge$, предположим обратное. Рассмотрим последнее правило вывода, которое могло быть применено, чтобы получить данное утверждение. $(I \to)$ и $(E \to)$ 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 $(E\wedge)$:

$$\dfrac{\begin{array}{c}\vdots\\ x\colon \alpha \to \beta \wedge \gamma \vdash x\colon (\alpha \to \beta) \wedge \sigma\end{array}}{x\colon \alpha \to \beta \wedge \gamma \vdash x\colon \alpha \to \beta}$$

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

□

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

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

□

***Замечание* 1.8.1.** *Тип $(\alpha \to \beta \wedge \gamma) \to \alpha \to \beta$ обитаем в обеих системах. В $\lambda_\wedge$ он содержит $\lambda ab.ab$, 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 $\lambda_{\wedge\leqslant}$ 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:* $[\Gamma_1 \vdash X : \tau_1, \ldots, \Gamma_n \vdash X : \tau_n]$*. 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* τ_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* τ_i *стрелочные* ($\tau_i = \sigma_i \to \rho_i$, $i = 1 \ldots n$)*. Тогда решение системы –* $X = \lambda x.X'$*, где* X' *– решение новой системы* $[(\Gamma_1, x : \sigma_1 \vdash X' : \rho_1), \ldots, (\Gamma_n, x : \sigma_n \vdash X' : \rho_n)]$*. То есть в этом случае во всех типах редуцируется первый аргумент.*

- $3_\wedge$*. Один из типов* τ_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* $k \geqslant 0$ *таким образом, чтобы* $\Gamma_i \vdash \lambda z_1 \ldots z_k.xz_1 \ldots z_k : \rho_i^1 \to \cdots \to \rho_i^k \to \tau_i$ *для всех* $i = 1 \ldots n$*. Тогда решение –* $X = xZ^1 \ldots Z^k$*, где* Z^i *– решение системы* $[(\Gamma_1 \vdash Z^i : \rho_1^i), \ldots, (\Gamma_n \vdash Z^i : \rho_n^i)]$*. 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 ρ_i^j. Типизация $\Gamma_i \vdash \lambda z_1 \ldots z_k.xz_1 \ldots z_k : \rho_i^1 \to \cdots \to \rho_i^k \to \tau_i$ не гарантирует того, что тип x в контексте Γ_i обязан быть $\rho_i^1 \to \cdots \to \rho_i^k \to \tau_i$.

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

Алгоритм для системы с η-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* τ_i *не находится в нормальной форме. Тогда заменим* τ_i *на* τ_i^**.*
- *Шаги* 1 *и* 2 *аналогичны Алгоритму 2.*
- $3_{\wedge\eta}$*. Один из типов* τ_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* $k \geqslant 0$ *таким образом, чтобы* $\Gamma_i \vdash x : \rho_i^1 \to \dots \to \rho_i^k \to \tau_i$ *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* ρ_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 – $p\colon \alpha \to \beta \wedge \gamma, q\colon \alpha \vdash X\colon \beta$. Алгоритм должен произвести преобразование $3_{\wedge\eta}$, но в контексте нет такого x, что $p\colon \alpha \to \beta \wedge \gamma, q\colon \alpha \vdash x\colon \dots \to \beta$, поскольку, согласно Лемме 1.7, утверждение о типизации $p\colon \alpha \to \beta \wedge \gamma \vdash p\colon \alpha \to \beta$ неверно в $\lambda_{\wedge}$. Поэтому алгоритм завершится, не найдя решение $X = pq$. Но преобразование $3_{\wedge}$ применить можно: $p\colon \alpha \to \beta \wedge \gamma, q\colon \alpha \vdash \lambda z.pz\colon \alpha \to \beta$. Здесь λ позволяет применить $(I \to)$ и удалить $\to$.

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 $\leqslant 2$.

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 τ, 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 $(I\wedge)$, $(I \to)$ и $(E \to)$ соответственно. □

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, τ – тип в нормальной форме без пересечений на верхнем уровне (см. следствие 1.5.1). Тогда равносильно:*

$$\Gamma \vdash xM_1 \dots M_k : \tau$$

$$\iff$$

$$\Gamma \vdash x : \alpha_1 \to \dots \to \alpha_k \to \tau \text{ и } \Gamma \vdash M_i : \alpha_i \text{ для } i = 1 \dots k$$

Доказательство. Докажем равносильность в обе стороны.

$\Leftarrow$) применяя правило $(E \to)$ 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 $\lambda_{\wedge\leqslant}$ (3), поскольку в ней меньше правил вывода.

Какое правило вывода могло быть применено последним, чтобы получить утверждение $\Gamma \vdash xM_1 \dots M_k \colon \tau$? 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 τ нет пересечений на верхнем уровне. Таким образом, могли быть применены только два правила: $(\leqslant)$ или $(E \rightarrow)$.

Согласно Лемме 1.5.1, τ имеет вид $\rho_1 \rightarrow \dots \rightarrow \rho_m \rightarrow \beta$, где β — типовая переменная. По Лемме 1.6, подтип τ 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 $(\leqslant)$, получив нечто такой же формы (поскольку $\underline{\underline{\tau}} = \underline{\tau}$) 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:

$$\dfrac{\begin{array}{cc} \vdots & \vdots \\ \Gamma \vdash xM_1 \dots M_{k-1} \colon \alpha_k \rightarrow \underline{\tau} & \Gamma \vdash M_k \colon \alpha_k \end{array}}{\Gamma \vdash xM_1 \dots M_k \colon \underline{\tau}}\ (E \rightarrow)$$
$$\vdots\ (\leqslant), (I\wedge)$$
$$\Gamma \vdash xM_1 \dots M_k \colon \tau$$

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

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

$\square$

Замечание 3.1.1. *Условие о том, что τ – тип без пересечений на верхнем уровне, существенно. Так, например, для $\Gamma = \{x \colon (\alpha \rightarrow \beta) \wedge (\gamma \rightarrow \delta), M \colon (\alpha \wedge \gamma)\}$, $\Gamma \vdash xM \colon \beta \wedge \gamma$, но при этом $\Gamma \nvdash x \colon \dots \rightarrow \beta \wedge \gamma$.*

Замечание 3.1.2. *Условие о том, что τ – тип в нормальной форме, существенно. Так, например, для $\Gamma = \{x \colon (\alpha \rightarrow \varphi \rightarrow \beta) \wedge (\gamma \rightarrow \varphi \rightarrow \delta), M \colon (\alpha \wedge \gamma)\}$, $\Gamma \vdash xM \colon \varphi \rightarrow (\beta \wedge \gamma)$, но при этом $\Gamma \nvdash x \colon \dots \rightarrow \varphi \rightarrow (\beta \wedge \gamma)$.*

Следствие 3.1.1. *В условиях предыдущей леммы, при выполнении любой из равносильных частей, $\Gamma \ni x \colon \underline{\alpha_1 \rightarrow \dots \rightarrow \alpha_k \rightarrow \tau}$,* *Доказательство.* Для доказательства достаточно провести ещё один шаг в рассуждениях леммы, за тем лишь исключением, что вместо $(E \to)$ 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* Γ *есть типизация* $x\colon \sigma$ *такая, что* $\hat{\sigma^*} \ni \overline{\alpha_1} \to \cdots \to \overline{\alpha_k} \to \underline{\tau}$.

В частности, если τ *— типовая переменная, то* $\hat{\sigma^*} \ni\ \overline{\alpha_1} \to \cdots \to \overline{\alpha_k} \to \tau$.

Доказательство. Воспользуемся предыдущим следствием и тем наблюдением, что $\underline{\alpha_1 \to \cdots \to \alpha_k \to \tau} = \varphi_1 \wedge (\overline{\alpha_1} \to \varphi_2 \wedge (\overline{\alpha_2} \to \varphi_3 \wedge \ldots (\overline{\alpha_k} \to \underline{\tau}) \ldots))$ $\square$

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

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

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

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 $T = [\Gamma_1 \vdash X : \tau_1, \ldots, \Gamma_n \vdash X : \tau_n]$, а M — её решение.

Если один из τ_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 τ_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 τ_i не является пересечением. Рассмотрим, как может быть устроено M.

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

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

- $M = xM_1M_2 \dots M_k$.
  - Пусть все типы в задачах стрелочные: $\tau_i = \alpha_i \to \beta_i$. Тогда алгоритм, аналогично предыдущему пункту, перейдёт к системе $T = [\Gamma_1, y\colon \alpha_1 \vdash X'\colon \beta_1, \dots, \Gamma_n, y\colon \alpha_n \vdash X'\colon \beta_n]$. решение которой — $xM_1M_2 \dots M_k y$.
  - Пусть один из типов в задачах – переменная: $\tau_i = \beta$. По Лемме 3.1, верно следующее: $\Gamma_i \vdash x\colon \alpha_i^1 \to \dots \to \alpha_i^k \to \tau_i$ и $\Gamma_i \vdash M_j\colon \alpha_i^j$ 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 «$\to$». Более формальное определение следующее:

***Определение* 6.**

$$\begin{aligned}
rank(\tau) &= 0 \\
&\quad \textit{если } \tau \textit{ – тип без пересечений} \\
rank(\sigma \wedge \tau) &= max(1, rank(\sigma), rank(\tau)) \\
rank(\sigma \to \tau) &= max(1 + rank(\sigma), rank(\tau)) \\
&\quad \textit{если } rank(\sigma) > 0 \textit{ или } rank(\tau) > 0
\end{aligned}$$

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

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

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

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

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

Проследим за рангами целевых типов τ_i и типов в контекстах Γ_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 σ_i, находящиеся слева от стрелки в типе $\tau_i = \sigma_i \to \rho_i$, а значит, имеющие или нулевой ранг, или хотя бы на 1 меньший ранг, чем τ_i (см. Лемму 3.2).

В преобразовании $3_{\wedge\eta}$ контексты не изменяются, а в целевыми становятся типы α_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 $\leqslant 1$, 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.

$\square$

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.