Проблемы населенности типов и их разрешимость: система $\lambda\cap$

Денис Николаевич Москвин

25.11.2017

План лекции

1. Система $\lambda\cap$
2. Обитаемость типов

План лекции

1. Система $\lambda\cap$
2. Обитаемость типов

Мотивация для системы $\lambda\cap$

- Рассмотрим следующий редукционный переход

$$(\lambda x.\, x\, x)(\lambda y.\, y) \to_\beta (\lambda y.\, y)(\lambda y.\, y)$$

- В системе $\lambda_\to$ правая часть типизируема, а левая — нет.
- Проблема в том, что $\mathbf{I}$ используется при подстановке в двух разных местах с разными, *неунифицируемыми* требованиями к типизации:

$$\begin{aligned}&\lambda y^{\alpha\to\alpha}.\, y : (\alpha\to\alpha)\to\alpha\to\alpha\\ &\lambda y^{\alpha}.\, y : \alpha\to\alpha\end{aligned}$$

- Идея: разрешить приписывать терму несколько типов одновременно.

$$(\lambda x^{(\alpha\to\alpha)\cap((\alpha\to\alpha)\to\alpha\to\alpha)}.\, x\, x)(\lambda y.\, y)^{(\alpha\to\alpha)\cap((\alpha\to\alpha)\to\alpha\to\alpha)}$$

- Протокол использования порождает естественное отношение подтипизации: $\sigma\cap\tau \leqslant \sigma$.

Мотивация для универсального типа

- Рассмотрим следующий редукционный переход

$$(\lambda x\, y.\, y)\, \mathbf{\Omega} \to_\beta \lambda y.\, y$$

- В системе $\lambda_\to$ правая часть типизируема, а левая — нет.
- Однако тут нет никакого пересечения, то есть оно пустое.
- Вводят универсальный тип ω, который можно приписать любым термам.

$$(\lambda x^\omega\, y^\alpha.\, y)\, \mathbf{\Omega}^\omega : \alpha \to \alpha$$

Типы системы $\lambda\cap$

Определение

Множество ***типов*** $\mathbb{T}$ системы $\lambda\cap$ строится из типовых переменных переменных из $\mathbb{V} = \{\alpha, \beta, \ldots\}$:

$$\alpha \in \mathbb{V} \Rightarrow \ \alpha \in \mathbb{T} \qquad \text{(переменные типа)}$$

$$\sigma, \tau \in \mathbb{T} \ \Rightarrow \ \sigma \to \tau \in \mathbb{T} \qquad \text{(типы пространства функций)}$$

$$\sigma, \tau \in \mathbb{T} \ \Rightarrow \ \sigma \cap \tau \in \mathbb{T} \qquad \text{(типы-пересечения)}$$

- В абстрактном синтаксисе:

$$\mathbb{T} \ ::= \ \mathbb{V} \mid \mathbb{T} \to \mathbb{T} \mid \mathbb{T} \cap \mathbb{T} \mid \omega$$

- Приоритет $\cap$ выше, чем $\to$.

Предпорядок на типах $\lambda\cap$

(refl)	$\sigma \leqslant \sigma$
(incl L)	$\sigma \cap \tau \leqslant \sigma$
(incl R)	$\sigma \cap \tau \leqslant \tau$
(glb)	$\dfrac{\rho \leqslant \sigma \quad \rho \leqslant \tau}{\rho \leqslant \sigma \cap \tau}$
(trans)	$\dfrac{\rho \leqslant \sigma \quad \sigma \leqslant \tau}{\rho \leqslant \tau}$
(ω top)	$\sigma \leqslant \omega$
($\omega \to$)	$\omega \leqslant \sigma \to \omega$
($\to \cap$)	$(\sigma \to \tau_1) \cap (\sigma \to \tau_2) \leqslant \sigma \to \tau_1 \cap \tau_2$
($\to$)	$\dfrac{\sigma' \leqslant \sigma \quad \tau \leqslant \tau'}{\sigma \to \tau \leqslant \sigma' \to \tau'}$

Если $\sigma \leqslant \tau$ и $\tau \leqslant \sigma$, то вводят отношение эквивалентности $\sigma = \tau$. При этом синтаксическое равенство обозначают $\sigma \equiv \tau$.

Примеры

Покажите, что

- $\sigma \cap \tau = \tau \cap \sigma.$
- $\sigma = \sigma \cap \sigma.$
- $\sigma = \sigma \cap \omega.$

Лемма

Отношение $\leqslant$ разрешимо, то есть существует алгоритм, который для заданных σ и τ проверяет, выполняется ли $\sigma \leqslant \tau$ или нет.

Типизация для системы $\lambda\cap$

(ax)	$\Gamma \vdash x{:}\sigma,$	если $(x{:}\sigma) \in \Gamma$
$(\mathrm{I} \to)$	$\dfrac{\Gamma, x{:}\sigma \vdash M : \tau}{\Gamma \vdash \lambda x.\, M : \sigma \to \tau},$	если $(x{:}\sigma) \notin \Gamma$
$(\mathrm{E} \to)$	$\dfrac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash M\, N : \tau}$	
$(\mathrm{I} \cap)$	$\dfrac{\Gamma \vdash M : \sigma \quad \Gamma \vdash M : \tau}{\Gamma \vdash M : \sigma \cap \tau}$	
$(\leqslant)$	$\dfrac{\Gamma \vdash M : \sigma}{\Gamma \vdash M : \tau},$	если $\sigma \leqslant \tau$
(ω)	$\Gamma \vdash M : \omega$	

Понятие контекста определено так же как в $\lambda_{\to}$.

Типизация для системы $\lambda\cap$ пример

Вывод типа для $\lambda x.\, x\, x$. Для компактности введено $\Gamma = x : \alpha \cap (\alpha \to \beta)$.

$$\dfrac{\dfrac{\Gamma \vdash x : \alpha \cap (\alpha \to \beta)}{\Gamma \vdash x : \alpha \to \beta}(\leqslant) \qquad \dfrac{\Gamma \vdash x : \alpha \cap (\alpha \to \beta)}{\Gamma \vdash x : \alpha}(\leqslant)}{\dfrac{x : \alpha \cap (\alpha \to \beta) \vdash x\, x : \beta}{\vdash \lambda x.\, x\, x : \alpha \cap (\alpha \to \beta) \to \beta}(I \to)}(E \to)$$

- Если α и β рассматривать как метапеременные, то можно говорить о *схеме вывода*.
- Постройте дерево вывода для утверждения типизации $\vdash \mathbf{Y} : (\omega \to \sigma) \to \sigma$.

Типизация для системы $\lambda\cap$ пример (2)

Вывод типа для $\lambda f\,x.\,(\lambda y.\,x)\,(f\,x)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{f:\alpha, x:\beta, y:\omega \vdash x:\beta}{f:\alpha, x:\beta \vdash \lambda y.\,x:\omega \to \beta}\,(I\to) \qquad \dfrac{}{f:\alpha, x:\beta \vdash (f\,x):\omega}\,(\omega)}{f:\alpha, x:\beta \vdash (\lambda y.\,x)\,(f\,x):\beta}\,(E\to)}{f:\alpha \vdash \lambda x.\,(\lambda y.\,x)\,(f\,x):\beta \to \beta}\,(I\to)}{\vdash \lambda f\,x.\,(\lambda y.\,x)\,(f\,x):\alpha \to \beta \to \beta}\,(I\to)$$

- В $\lambda_{\to}$ выводимо лишь $\lambda f\,x.\,(\lambda y.\,x)\,(f\,x) : (\beta \to \gamma) \to \beta \to \beta$.
- В $\lambda_{\to}$ этому терму невозможно приписать тип $\alpha \to \beta \to \beta$, хотя можно его сокращению.
- В отличие от простой системы экспансия в $\lambda\cap$ сохраняет тип.

Связь типизируемости и нормализуемости для $\lambda\cap$

Теорема

Замкнутый терм M имеет головную нормальную форму (HNF) тогда и только тогда, когда существует отличный от ω тип σ, такой что $\vdash M : \sigma$.

Например, $\vdash \mathbf{Y} : (\omega \to \sigma) \to \sigma$, но $\vdash \mathbf{\Omega} : \omega$.

Теорема

Замкнутый терм M имеет нормальную форму (NF) тогда и только тогда, когда существует не содержащий ω тип σ, такой что $\vdash M : \sigma$.

Рассмотрим систему $\lambda\cap^-$ без типа ω.

Теорема (van Bakel, Krivine)

Замкнутый терм M сильно нормализуем тогда и только тогда, когда он типизируем в системе $\lambda\cap^-$.

Проблемы разрешимости для $\lambda\cap$

- Есть ли алгоритм, который позволяют решить задачу?

$\Gamma \vdash M : \sigma ?$	Задача проверки типа Type Checking Problem	ЗПТ TCP
$? \vdash M : ?$	Задача синтеза типа Type Synthesis (or Assgnment) Problem	ЗСТ TSP, TAP
$\Gamma \vdash ? : \sigma$	Задача обитаемости типа Type Inhabitation Problem	ЗОТ TIP

- Для $\lambda\cap$ ЗПТ неразрешима.
- Для $\lambda\cap$ ЗСТ разрешима тривиально - все термы типизируемы с помощью ω.
- Для $\lambda\cap^{-}$ (система без типа ω) ЗСТ неразрешима.

План лекции

1 Система $\lambda\cap$

2 Обитаемость типов

ЗОТ для $\lambda\cap$ неразрешима

- Павел Уржицин (Pawel Urzyczyn, 1999) доказал неразрешимость задачи обитаемости для $\lambda\cap$.
- Скелет доказательства:
  - процесс конструирования обитателя для типа выражает поведение некоторой машины (tree-maker);
  - проблема останова для этой машины сводима к проблеме останова для queue automata.
- Доказательство работает для 4х вариантов системы с пересечениями (при наличии или отсутствии универсального типа и подтипизации).

ЗОТ для $\lambda\cap$ неразрешима

- Павел Уржицин (Pawel Urzyczyn, 1999) доказал неразрешимость задачи обитаемости для $\lambda\cap$.
- Скелет доказательства:
  - процесс конструирования обитателя для типа выражает поведение некоторой машины (tree-maker);
  - проблема останова для этой машины сводима к проблеме останова для queue automata.
- Доказательство работает для 4х вариантов системы с пересечениями (при наличии или отсутствии универсального типа и подтипизации).
- Однако есть ограниченные версии, для которых задача обитаемости разрешима.

Определение (порядок типа в $\lambda_{\rightarrow}$)

$$\begin{aligned}\mathrm{ord}(\alpha) &= 0\\ \mathrm{ord}(\sigma \rightarrow \tau) &= \max(\mathrm{ord}(\sigma)+1, \mathrm{ord}(\tau))\end{aligned}$$

Типы второго порядка — это типы, среди аргументов которых есть функции первого порядка, но не выше.

Определение (ранг типа в $\lambda\cap$)

$$\begin{aligned}\mathrm{rank}(\alpha) &= 0\\ \mathrm{rank}(\sigma \rightarrow \tau) &= \max(\mathrm{rank}(\sigma)+1, \mathrm{rank}(\tau))\\ \mathrm{rank}(\sigma \cap \tau) &= \max(1, \mathrm{rank}(\sigma), \mathrm{rank}(\tau))\end{aligned}$$

Во втором уравнении в σ или τ должны быть пересечения.

- $\mathrm{rank}(\alpha \cap (\alpha \rightarrow \beta \rightarrow \gamma)) = ?$
- $\mathrm{rank}(\alpha \cap \beta \rightarrow \alpha) = ?$
- $\mathrm{rank}(\alpha \cap (\beta \rightarrow \gamma) \rightarrow \alpha) = ?$
- $\mathrm{rank}(\alpha \cap \beta \rightarrow \alpha \cap (\beta \rightarrow \gamma) \rightarrow \alpha) = ?$

Системы с рангом 3 и выше

- Уже в (Urzyczyn, 1999) было отмечено, что ЗОТ для систем $\lambda\cap$ с рангом выше 3 неразрешима.
- Это следует из доказательства общей неразрешимости, при построении tree-maker'ов используются так называемые «хорошие» типы, рангом не выше 3.
- Для системы ранга 3 неразрешимость была показана в (Urzyczyn, 2009), сведением к Expanding Tape Machine.

Система с рангом 2

- В (Kusmierek, 2007) дан алгоритм и доказана его завершаемость (и сложность EXPTIME-hard).
- Алгоритм представляет собой обобщение алгоритма Ben-Yelles для простых типов на $\cap$-типы.
- Этот алгоритм универсален в том смысле, что может применяться для систем любого ранга.

Определение

Переменная x называется k-арной в контексте Γ, если этот контекст содержит $x : \sigma$, причем выполнено одно из следующих условий:

$$\sigma = \tau \cap (\rho_1 \to \ldots \to \rho_k \to \alpha)$$
$$\sigma = \rho_1 \to \ldots \to \rho_k \to \alpha$$

Иными словами, x можно применить к k аргументам.

Процедура удаления пересечений

Введем процедуру превращения утверждения типизации в множество утверждений без пересечений:

Операция Rem

$$\mathrm{Rem}(\Gamma \vdash M : \sigma) = \{\Gamma \vdash M : \sigma\}, \quad \text{если } \sigma \text{ не пересечение;}$$

$$\mathrm{Rem}(\Gamma \vdash M : \sigma_1 \cap \sigma_2) = \mathrm{Rem}(\Gamma \vdash M : \sigma_1) \cup \mathrm{Rem}(\Gamma \vdash M : \sigma_2).$$

С ее помощью мы будем решать задачу поиска обитателя без целевых $\cap$-типов, но не одну, а сразу множество:

$$\{\Gamma_1 \vdash M : \sigma_1, \ldots, \Gamma_n \vdash M : \sigma_n\}$$

Контексты разные (но носители их одинаковые), типы разные, а терм один.

Алгоритм

- Для заданного целевого типа σ нулевой шаг

$$Z_0 = \mathrm{Rem}(\vdash M : \sigma)$$

- Пусть текущая задача

$$Z = \{\Gamma_1 \vdash M : \sigma_1, \dots, \Gamma_n \vdash M : \sigma_n\}$$

- **Случай 1:** Все σ_i стрелки (скажем, $\tau_i \to \rho_i$).
  Вводим свежий x и удаляем пересечения:

  $$Z' = \mathrm{Rem}(\Gamma_1, x{:}\tau_1 \vdash M' : \rho_1) \cup \ldots \cup \mathrm{Rem}(\Gamma_n, x{:}\tau_n \vdash M' : \rho_n)$$

  Если рекурсивный вызов алгоритма вернет M', то $M = \lambda x.\, M'$, если вернет ошибку — то ошибка.

Алгоритм (продолжение)

- Напомним, текущая задача

$$Z = \{\Gamma_1 \vdash M : \sigma_1, \ldots, \Gamma_n \vdash M : \sigma_n\}$$

- **Случай 2:** Хотя бы одна σ_i — переменная (скажем, α). Ищем в каждом Γ_i k-арную переменную

$$\Gamma_i \vdash x : \sigma_{i1} \to \ldots \to \sigma_{ik} \to \alpha$$

Если не нашли — возвращаем ошибку; если нашли несколько — выбираем недетерминировано.

  - Если $k = 0$, то $M = x$.
  - Если $k > 0$, то $M = x\, M_1 \ldots M_k$, где M_i — это решение k независимых задач

$$\begin{aligned} Z_1 &= \{\Gamma_1 \vdash M_1 : \sigma_{11}, \ldots, \Gamma_n \vdash M : \sigma_{1n}\} \\ \ldots \\ Z_k &= \{\Gamma_1 \vdash M_1 : \sigma_{k1}, \ldots, \Gamma_n \vdash M : \sigma_{kn}\} \end{aligned}$$

Если хоть одна дает ошибку, общий ответ — ошибка.

Алгоритм (примеры)

Используя приведенный выше алгоритм, населите

- $\alpha \cap (\alpha \to \beta) \to \beta$
- $\alpha \cap \beta \cap (\alpha \to \beta \to \gamma) \to \gamma$
- $(\alpha \to \beta) \cap (\alpha \to \gamma) \to \alpha \to \beta \cap \gamma$
- $\alpha \to \beta \to \alpha \cap \beta$