

ТЕОРЕТИЧЕСКИЕ ДОМАШНИЕ ЗАДАНИЯ

Теория типов, ИТМО, совместно М3232-М3239 и М3332-М3339, весна 2024 года

Домашнее задание №1: лямбда исчисление — бестиповое и просто-типизированное

1. Бесконечное количество комбинаторов неподвижной точки. Дадим следующие определения

$$\begin{aligned} L &:= \lambda abcdefghijklmnopqrstuvwxyzr.(thisisafixedpointcombinator) \\ R &:= LLLLLLLLLLLLLLLLLLLLLLLLLLLLL \end{aligned}$$

В данном определении терм R является комбинатором неподвижной точки: каков бы ни был терм F , выполнено $R F =_{\beta} F (R F)$.

- (a) Докажите, что данный комбинатор — действительно комбинатор неподвижной точки.
- (b) Пусть в качестве имён переменных разрешены русские буквы. Постройте аналогичное выражение по-русски: с 33 параметрами и осмысленной русской фразой в терме L ; покажите, что оно является комбинатором неподвижной точки.
2. Найдите необитаемый тип в просто-типизированном лямбда-исчислении (напомним: тип τ называется необитаемым, если ни для какого P не выполнено $\vdash P : \tau$).

3. Напомним определение: комбинатор — лямбда-выражение без свободных переменных. Также напомним:

$$\begin{aligned} S &:= \lambda x.\lambda y.\lambda z.x z (y z) \\ K &:= \lambda x.\lambda y.x \\ I &:= \lambda x.x \end{aligned}$$

Известна теорема о том, что для любого комбинатора X можно найти выражение P (состоящее только из скобок, пробелов и комбинаторов S и K), что $X =_{\beta} P$. Будем говорить, что комбинатор P *выражает* комбинатор X в базисе SK .

Косвенным аргументом (пояснением, но не доказательством!) в пользу этой теоремы являются два следующих соображения:

- теорема о замкнутости ИФИИВ: если $\vdash \varphi$, то $\vdash_{\rightarrow} \varphi$, значит, если выражение имеет тип, то этот тип можно получить с помощью доказательства в стиле Гильберта;
- типы комбинаторов S и K — это, соответственно, вторая и первая схемы аксиом.

Докажите тип следующих выражений как логическое высказывание с помощью гильбертового вывода и, пользуясь этим доказательством как источником вдохновения, выразите комбинаторы в базисе SK :

- (a) $\lambda x.\lambda y.\lambda z.y$
(b) $\lambda x.\lambda y.\lambda z.yxz$
(c) \bar{I}
(d) Not
(e) Xor
(f) InR

4. Покажите на основании следующего преобразования полноту комбинаторного базиса SKI (проведите полное рассуждение по индукции, из которого будет следовать отсутствие в результате других термов, кроме SKI , бета-эквивалентность и определённость результата для всех комбинаторов σ):

$$[\sigma] = \begin{cases} x, & \sigma = x \\ [\varphi] [\psi], & \sigma = \varphi \psi \\ K [\varphi], & \sigma = \lambda x.\varphi, \quad x \notin FV(\varphi) \\ I, & \sigma = \lambda x.x \\ [\lambda x. [\lambda y.\varphi]], & \sigma = \lambda x.\lambda y.\varphi, \quad x \in FV(\varphi), x \neq y \\ S [\lambda x.\varphi] [\lambda x.\psi], & \sigma = \lambda x.\varphi \psi, \quad x \in FV(\varphi) \cup FV(\psi) \end{cases}$$

Заметим, что данные равенства объясняют смысл названий комбинаторов:

S verSchmelzung, «сплавнение»
 K Konstanz
 I Identität

5. Покажите, что следующая система комбинаторов образует полный базис в бестиповом лямбда-исчислении, но соответствующая им система аксиом в исчислении гильбертового типа не образует полного базиса для импликативного фрагмента:

$$\begin{aligned} I &:= \lambda x.x \\ K &:= \lambda x.\lambda y.x \\ S' &:= \lambda i.\lambda x.\lambda y.\lambda z.i \ (i \ ((x \ (i \ z)) \ (i \ (y \ (i \ z))))) \end{aligned}$$

Указание: покажите невыводимость $(\varphi \rightarrow \varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \psi)$.

6. Напомним определение аппликативного порядка редукции: редуцируется самый левый из самых вложенных редексов. Например, в случае выражения $(\lambda x.I \ I) \ (\lambda y.I \ I)$ самые вложенные редексы — применения $I \ I$:

$$(\lambda x.\underline{I \ I}) \ (\lambda y.\underline{I \ I})$$

и надо выбрать самый левый из них:

$$(\lambda x.\underline{I \ I}) \ (\lambda y.I \ I)$$

- (a) Проведите аппликативную редукцию выражения 2 2.
- (b) Докажите или опровергните, что параллельная бета-редукция из теоремы Чёрча-Россера не медленнее (в смысле количества операций для приведения выражения к нормальной форме), чем нормальный порядок редукции с мемоизацией.
- (c) Найдите лямбда-выражение, которое редуцируется медленнее при нормальном порядке редукции, чем при аппликативном, даже при наличии мемоизации.
7. Напомним определение бета-редукции. $A \rightarrow_\beta B$, если:
- $A \equiv (\lambda x.P) \ Q$, $B \equiv P \ [x := Q]$, при условии свободы для подстановки;
 - $A \equiv (P \ Q)$, $B \equiv (P' \ Q')$, при этом $P \rightarrow_\beta P'$ и $Q = Q'$, либо $P = P'$ и $Q \rightarrow_\beta Q'$;
 - $A \equiv (\lambda x.P)$, $B \equiv (\lambda x.P')$, и $P \rightarrow_\beta P'$.
- (a) Найдите лямбда-выражение, бета-редукция которого не может быть произведена из-за нарушения правила свободы для подстановки (для продолжения редукции потребуется производить переименование связанных переменных). Поясните, какое ожидаемое ценное свойство будет нарушено, если ограничение правила проигнорировать.
- (b) Покажите, что недостаточно наложить требования на исходное выражение, и свобода для подстановки может быть нарушена уже в процессе редукции исходно полностью корректного лямбда-выражения.
8. Будем говорить, что выражение A находится в *слабой заголовочной нормальной форме* (WHNF), если оно не имеет вид $A \equiv (\lambda x.P) \ Q$ (то есть, самый верхний терм его не является редексом). Выражение находится в *заголовочной нормальной форме* (HNF), когда его верхний терм — не редекс и не лямбда-абстракция с бета-редексами в теле.
- (a) Приведите нормальным порядком редукции выражение 2 2 в СЗНФ.
- (b) Приведите нормальным порядком редукции выражение $Y \ (\lambda f.\lambda x.(IsZero \ x) \ 1 \ (x \cdot f(x - 1))) \ 3$ в СЗНФ.
- (c) Верно ли, что «нормальность» формы выражения может в процессе редукции только усиливаться (никакая — слабая заголовочная Н.Ф. — заголовочная Н.Ф. — нормальная форма)?
9. Как мы уже разбирали, $\not\vdash x \ x : \tau$ в силу дополнительных ограничений правила

$$\frac{}{\Gamma, x : \tau \vdash x : \tau} \quad x \notin FV(\Gamma)$$

Найдите лямбда-выражение N , что $\not\vdash N : \tau$ в силу ограничений правила

$$\frac{\Gamma, x : \sigma \vdash N : \tau}{\Gamma \vdash \lambda x.N : \sigma \rightarrow \tau} \quad x \notin FV(\Gamma)$$

Домашнее задание №2: задачи типизации лямбда исчисления

1. Рассмотрим подробнее отличия исчисления по Чёрчу и по Карри. Определим точно бета-редукцию в исчислении по Чёрчу: $A \rightarrow_\beta B$, если

$$\begin{aligned} A &= (\lambda x^\sigma. P) Q, & B &= P[x := Q] \\ A &= P Q, & B &= P Q' \text{ или } B = P' Q \text{ при } P \rightarrow_\beta P' \text{ и } Q \rightarrow_\beta Q' \\ A &= \lambda x^\sigma. P, & B &= \lambda x^\sigma. P' \text{ при } P \rightarrow_\beta P' \end{aligned}$$

- (a) Покажите, что в исчислении по Карри не выполняется даже «ограниченное» свойство распространения типизации (subject expansion): если $\vdash_K M : \sigma$, $M \rightarrow_\beta N$ и $\vdash_K N : \tau$, то необязательно, что $\sigma = \tau$.
- (b) Покажите, что в исчислении по Чёрчу «полное» свойство распространения типизации также не выполняется:

найдутся такие M, N, σ , что $\vdash_C N : \sigma$, $M \rightarrow_\beta N$, но $\nvdash_C M : \sigma$.

Но при этом в исчислении по Чёрчу выполнено «ограниченное» свойство распространения типизации:

если $\vdash_K M : \sigma$, $M \rightarrow_\beta N$ и $\vdash_K N : \tau$, то тогда $\sigma = \tau$.

2. Покажите, что никакие связи в ИИВ не выражаются друг через друга: то есть, нет такой формулы $\varphi(A, B)$ из языка интуиционистской логики, не использующей связку \star , что $\vdash A \star B \rightarrow \varphi(A, B)$ и $\vdash \varphi(A, B) \rightarrow A \star B$. Покажите это для каждой связки в отдельности:

- (a) конъюнкция;
- (b) дизъюнкция;
- (c) импликация;
- (d) отрицание.

3. Рассмотрим алгоритм построения системы уравнений, а именно случай, когда рассматривается два разных вхождения одинакового по тексту применения. Например, $(x \ x) (x \ x)$ имеет два разных вхождения одной и той же аппликации $x \ x$. Всегда ли для корректной работы алгоритма достаточно одной типовой переменной β_{xx} для этих двух вхождений, или нужны две разные β_{xx}^L и β_{xx}^R ? Примечание: при одной переменной для обеих аппликаций система в данном случае, очевидно, несовместна: $\beta_{xx} \neq \beta_{xx} \rightarrow \sigma$. Но контрпримером это не является, поскольку типа у данного выражения всё равно нет.

4. Предложим альтернативные аксиомы для конъюнкции:

$$\frac{\Gamma \vdash \alpha \quad \Gamma \vdash \beta}{\Gamma \vdash \alpha \ \& \ \beta} \text{ Введ. } \& \qquad \frac{\Gamma \vdash \alpha \ \& \ \beta \quad \Gamma, \alpha, \beta \vdash \gamma}{\Gamma \vdash \gamma} \text{ Удал. } \&$$

- (a) Предложите лямбда-выражения, соответствующие данным аксиомам; поясните, как данные выражения абстрагируют понятие «упорядоченной пары».
 - (b) Выразите изложенные в лекции аксиомы конъюнкции через приведённые в условии.
 - (c) Выразите приведённые в условии аксиомы конъюнкции через изложенные в лекции.
5. Постройте систему уравнений для Y -комбинатора и примените к ней алгоритм унификации (ожидается, что система окажется несовместной).

Домашнее задание №3: сильная нормализуемость λ_{\rightarrow} , система F

1. Найдите $\llbracket \alpha \rightarrow \alpha \rrbracket$.
2. Найдите $\llbracket (\alpha \rightarrow \alpha) \rightarrow \alpha \rrbracket$.
3. Покажите, что SN — насыщенное (постройте полноценное рассуждение по индукции для п.2 определения).
4. Покажите, что если \mathcal{A} и \mathcal{B} насыщены, то $\mathcal{A} \rightarrow \mathcal{B}$ насыщенное.
5. Покажите, что построенная на лекции простая модель для ИИП второго порядка неполна.

6. Напомним, что мы можем задать $\exists p.\varphi$ как $\forall q.(\forall p.\varphi \rightarrow q) \rightarrow q$ (где q — некоторая свежая переменная). Покажите, что правила для квантора существования могут быть выведены из такого определения.
7. Требуется ли свобода для подстановки в правилах с квантором?

$$\frac{\Gamma \vdash \varphi[p := \theta]}{\Gamma \vdash \exists p.\varphi} \quad \frac{\Gamma \vdash \forall p.\phi}{\Gamma \vdash \phi[p := \theta]}$$

Если да — приведите пример доказуемой при отсутствии свободы для подстановки, но некорректной формулы. Если нет — предложите доказательство корректности правил при любых подстановках.

8. Пусть $\Gamma \vdash \varphi$. Всегда ли можно перестроить доказательство φ , добавив ещё одну гипотезу: $\Gamma, \psi \vdash \varphi$? Если нет, каковы могли бы быть ограничения на ψ ?
9. Пусть $\Gamma \vdash \forall x.\varphi$. Верно ли тогда, что $\Gamma \vdash \forall y.\varphi[x := y]$? Если это неверно в общем случае, возможно, это верно при каких-то ограничениях? В случае наличия ограничений приведите надлежащие контрпримеры.
10. Перенесите в систему F из бестипового лямбда-исчисления следующие функции — иными словами, постройте их обобщение для системы F (приведите обобщённое выражение, укажите его тип и докажите его). Например, можно рассмотреть $I = \Lambda\pi.\lambda p^\pi.p \rightarrow p$.
 - (a) S, K.
 - (b) инъекции и *case* (операции с алгебраическим типом);
 - (c) истина, ложь, исключающее или;
 - (d) черчёвский нумерал (он должен иметь тип $\forall\alpha.(\alpha \rightarrow \alpha) \rightarrow (\alpha \rightarrow \alpha)$) и инкремент;
 - (e) возведение в степень: $\lambda m.\lambda n.n$ m ;
 - (f) вычитание единицы (трюк зуба мудрости) и вычитание.

Домашнее задание №4: экзистенциальные типы, типовая система Хиндли-Милнера

1. Постройте экзистенциальный тип для очереди, и реализуйте его с помощью двух стеков. Реализацию напишите на Хаскеле, используя `AbstractStack` с лекции как АТД стека (возможно, этот пример надо будет расширить нужными вам методами), и реализуйте какой-нибудь простой классический алгоритм с её помощью. Как, интересно, осуществить инстанциацию вложенного абстрактного типа данных? Придумайте.

Как с помощью двух стеков можно реализовать очередь со средним временем доступа $\Theta(1)$: входные значения кладём во входной стек, выходные достаём из выходного, при исчерпании — переносим всё из входного в выходной:

Входной стек	Выходной стек	Действие
$\square \rightarrow [1]$	\square	<i>push 1</i>
$[1] \rightarrow [3; 1]$	\square	<i>push 3</i>
$[3; 1] \rightarrow \square$	$\square \rightarrow [1; 3] \rightarrow [3]$	<i>pull</i>
$\square \rightarrow [5]$	$[3]$	<i>push 5</i>

2. Выразите дизъюнкцию через квантор существования в ИИП 2 порядка, а также алгебраический тип через экзистенциальный.
3. Покажите, что если $rk(\sigma, 1)$, то для выражения σ найдётся эквивалентное σ' с поверхностными кванторами.
4. Покажите, что в предыдущем задании также имеется изоморфизм типов: существует биективная функция $\sigma \rightarrow \sigma'$, которую можно выразить лямбда-выражениями.
5. Рассмотрим QuickSort:

```
let rec quick l = match l with
  [] -> []
  | l1 :: ls -> List.filter (fun x -> x < l1) ls @ [l1] @ List.filter (fun x -> x >= l1)
```

Укажите полные типовые схемы в системе НМ для всех функций, участвующих в данном примере (тип списка раскрывать не надо).

6. Заметим, что список — это «параметризованные» числа в аксиоматике Пеано. Число — это длина списка, а к каждому штриху мы присоединяем какое-то значение. Операции добавления и удаления элемента из списка — это операции прибавления и вычитания единицы к числу.

Рассмотрим тип «бинарного списка»:

```
type 'a bin_list = Nil | Zero of (('a*'a) bin_list) | One of 'a * (('a*'a) bin_list);;
```

Заметим, что здесь мы рассматриваем двоичную запись числа (чередующиеся `Zero` и `One`) — двоичную запись длины бинарного списка, и элемент двоичной записи номер n хранит 2^n или $2^n + 1$ значение (в зависимости от типа элемента). Например, 5-элементный список:

```
One ("a", Zero (One (((("b","c"),("d","e")), Nil)))
```

По идее, операция добавления элемента к списку записывается на языке Окамль вот так (сравните с прибавлением 1 к числу в двоичной системе счисления):

```
let rec add elem lst = match lst with
  Nil -> One (elem,Nil)
  | Zero tl -> One (elem,tl)
  | One (hd,tl) -> Zero (add (elem,hd) tl)
```

Однако, тип этой функции Окамль вывести автоматически не может, его надо указывать явно, чтобы код скомпилировался:

```
let rec add : 'a . 'a -> 'a bin_list -> 'a bin_list = fun elem lst -> match lst with
```

- Какой тип имеет `add` в (расширенной) системе F (напомним, поскольку функция рекурсивна, она должна использовать Y -комбинатор в своём определении)? Считайте, что семейство типов `bin_list 'a` предопределено и обозначается как τ_α . Также считайте, что определены функции `roll` и `unroll` с надлежащими типами. Какой ранг имеет тип этой функции? Почему этот тип не выразим в типовой системе Хиндли-Милнера?
 - Предложите функции для печати списка и для удаления элемента списка (головы).
 - Предложите функцию для эффективного соединения двух списков (источник для вдохновения — сложение двух чисел в столбик).
 - Предложите функцию для эффективного выделения n -го элемента из списка.
7. Рассмотрим следующий код на Окамле, содержащий определения чётрчевских нумералов и некоторых простых операций с ними:

```
let zero = fun f x -> x;;
let plus1 a = fun f -> fun x -> a f (f x);;
let power m n = n m;;

let two = plus1 (plus1 zero);;
let two2 = fun f x -> f (f x);;

let e = power two two;;          (* не компилируется *)
let e2 = power two2 two2;;       (* компилируется и работает *)
```

Разберите вывод типов в этом фрагменте (относительно типовой системы Хиндли-Милнера) и поясните, почему:

- определение `e2` компилируется и работает (предъявите доказательство типа в системе НМ);
- определение `e` не компилируется (например, примените алгоритм W и покажите шаг, где он выведет ошибку).