Теоретические домашние задания

Теория типов, ИТМО, совместно M3232-M3239 и M3332-M3339, весна 2024 года

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

1. Бесконечное количество комбинаторов неподвижной точки. Дадим следующие определения

$$L := \lambda abcdefghijklmnopqstuvwxyzr.r(thisisafixedpointcombinator)$$
$$R := LLLLLLLLLLLLLLLLLLLLLLLLLL$$

   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:

$$S := \lambda x.\lambda y.\lambda z.x\ z\ (y\ z)$$
$$K := \lambda x.\lambda y.x$$
$$I := \lambda x.x$$

   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 $\vdash_\to \varphi$, 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) $\overline{1}$

   (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, «сплавление»
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:

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

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

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. $A \to_\beta B$, 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 $P \to_\beta P'$ и $Q = Q'$, либо $P = P'$ и $Q \to_\beta Q'$;
- $A \equiv (\lambda x.P)$, $B \equiv (\lambda x.P')$, и $P \to_\beta P'$.

(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

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

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

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

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: $A \to_\beta B$, если

   $A = (\lambda x^\sigma.P)\ Q$, $\quad B = P[x := Q]$
   $A = P\ Q$, $\quad B = P\ Q'$ или $B = P'\ Q$ при $P \to_\beta P'$ и $Q \to_\beta Q'$
   $A = \lambda x^\sigma.P$, $\quad B = \lambda x^\sigma.P'$ при $P \to_\beta P'$

   (a) Покажите, что в исчислении по Карри не выполняется даже «ограниченное» свойство распространения типизации (subject expansion): если $\vdash_к M : \sigma$, $M \twoheadrightarrow_\beta N$ и $\vdash_к N : \tau$, то необязательно, что $\sigma = \tau$.

   (b) Покажите, что в исчислении по Чёрчу «полное» свойство распространения типизации также не выполняется:

   $$\text{найдутся такие } M, N, \sigma, \text{ что } \vdash_ч N : \sigma,\ M \twoheadrightarrow_\beta N, \text{ но } \nvdash_ч M : \sigma.$$

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

   $$\text{если } \vdash_к M : \sigma,\ M \twoheadrightarrow_\beta N \text{ и } \vdash_к N : \tau, \text{ то тогда } \sigma = \tau.$$

2. Покажите, что никакие связки в ИИВ не выражаются друг через друга: то есть, нет такой формулы $\varphi(A, B)$ из языка интуиционистской логики, не использующей связку $\star$, что $\vdash A \star B \to \varphi(A, B)$ и $\vdash \varphi(A, B) \to A \star B$. 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^L_{xx} и β^R_{xx}? Примечание: при одной переменной для обоих аппликаций система в данном случае, очевидно, несовместна: $\beta_{xx} \neq \beta_{xx} \to \sigma$. 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 $\lambda_\to$, система F

1. Найдите $[\![\alpha \to \alpha]\!]$.

2. Найдите $[\![(\alpha \to \alpha) \to \alpha]\!]$.

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 $\mathcal{A} \to \mathcal{B}$ насыщенное.

5. Покажите, что построенная на лекции простая модель для ИИП второго порядка неполна.

6. Напомним, что мы можем задать $\exists p.\varphi$ как $\forall q.(\forall p.\varphi \to q) \to q$ (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?

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

   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 $I = \Lambda\pi.\lambda p^{\pi}.p \to p$.

    (a) S, K.
    (b) инъекции и *case* (операции с алгебраическим типом);
    (c) истина, ложь, исключающее или;
    (d) черчёвский нумерал (он должен иметь тип $\forall\alpha.(\alpha \to \alpha) \to (\alpha \to \alpha)$) 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
$[] \to [1]$	$[]$	*push* 1
$[1] \to [3;1]$	$[]$	*push* 3
$[3;1] \to []$	$[] \to [1;3] \to [3]$	*pull*
$[] \to [5]$	$[3]$	*push* 5

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 $\sigma \to \sigma'$, 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 HM 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
```

   (a) 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?

   (b) Предложите функции для печати списка и для удаления элемента списка (головы).

   (c) Предложите функцию для эффективного соединения двух списков (источник для вдохновения — сложение двух чисел в столбик).

   (d) Предложите функцию для эффективного выделения n-го элемента из списка.

7. Рассмотрим следующий код на Окамле, содержащий определения чёрчевских 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:

   (a) определение $e2$ компилируется и работает (предъявите доказательство типа в системе HM);

   (b) определение e не компилируется (например, примените алгоритм W и покажите шаг, где он выведет ошибку).

Домашнее задание №5: Обобщённая система типов, гомотопическая теория типов, язык Аренд

1. Задача на доказательство сильной нормализуемости $\lambda_{\to}$: найдите примеры лямбда-термов, не принадлежащие (1) множеству $[\![\alpha \to \alpha]\!]$ и (2) множеству $[\![(\alpha \to \alpha) \to \alpha]\!]$ (для выполнения задания надо выполнить оба пункта).

2. Укажите тип (род) в исчислении конструкций для следующих выражений (при необходимости определите типы используемых базовых операций и конструкций самостоятельно):

   (a) В алгебраическом типе `'a option = None | Some 'a` предложите тип (род) для: `Some`, `None` и `option`.

   (b) Пусть задан род **nonzero** : $\star \to \star$, выбрасывающий нулевой элемент из типа. Например, **nonzero unsigned** — тип положительных целых чисел. Тогда, для кода

   ```
   template<typename T, T x>
   struct NonZero { const static std::enable_if_t<x != T(0), T> value = x; };
   ```

   предложите тип (род) поля value.

3. Предложите выражение на языке C++ (возможно, использующее шаблоны), имеющее следующий род (тип):

   (a) $\star \to \star \to \star$; $\star \to$ **unsigned**

   (b) **int** $\to (\star \to \star)$

   (c) $(\star \to$ **int**$) \to \star$

   (d) $\Pi x^{\star}.n^{\mathbf{int}}.F(n,x)$, где

   $$F(n,x) = \begin{cases} \mathbf{int}, & n = 0 \\ x \to F(n,x), & n > 0 \end{cases}$$

4. Определите функции из следующих частей λ-куба (в обобщённой типовой системе) и докажите их тип:

   (a) $(\Box, \star)$

   (b) $(\star, \Box)$

   (c) $(\Box, \Box)$

5. Рассмотрим правый дальний нижний угол λ-куба ($\{(\star,\star);(\star,\Box);(\Box,\star)\}$). Можно предположить, что тогда в такой системе возможны и функции рода $f : \star \to \star$ (как композиция функций $p : \star \to \alpha$ и $q : \alpha \to \star$ — например, можно кодировать тип его именем, затем по имени типа восстанавливать сам тип обратно). Почему всё-таки такие функции в обобщённых типовых системах невозможны без четвёртого элемента $(\Box, \Box)$?

6. Какова должна быть топология на множестве пар натуральных чисел (интуитивно мы будем понимать эти пары как рациональные числа, пары «числитель-знаменатель»), чтобы непрерывными были бы те и только те функции, для которых выполнено $f(p,q) = f(p',q')$ для всех таких p, p', q и q', что $p \cdot q' = p' \cdot q$. Напомним, что равенство мы понимаем как наличие непрерывного пути между точками.

7. Докажите, приведя компилирующуюся программу на языке Аренд (возможно, вам потребуются функции и приёмы, изложенные в документации по языку: `https://arend-lang.github.io/documentation/tutorial/PartI/`):

   (a) ассоциативность сложения;

   (b) коммутативность сложения;

   (c) коммутативность умножения;

   (d) дистрибутивность: $(a + b) \cdot c = a \cdot c + b \cdot c$;

   (e) куб суммы: $(a + 1)^3 = a^3 + 3 \cdot a^2 + 3 \cdot a + 1$.

8. Определим, что x делится на p, если обитаем тип `\Sigma (q : Nat) (p * q = x)`.

   (a) Покажите, что если x делится на 6, то x делится и на 3;

(b) Покажите, что $x!$ делится на x;

(c) Покажите, что если x делится на y и y делится на z, то x делится на z;

9. Определите предикат (т.е. функцию с надлежащим типом) для формализации понятия простого числа `isPrime`. Покажите, что:

   (a) 3 и 11 — простые числа;

   (b) Произведение простых чисел непросто;

   (c) 2 — единственное чётное простое число.

10. Определим отношение «меньше» на натуральных числах так (с помощью индуктивного типа, обобщения алгебраического типа данных — зависимого типа, в котором при разных значениях аргументов типа допустимы разные конструкторы):

```
\data NatLessEq (a b : Nat) \with
  | 0, m => natlesseq-zero
  | suc m, suc n => natlesseq-next (NatLessEq m n)
```

Например, конструктор `natlesseq-zero` можно использовать только если первый аргумент типа — число 0. А конструктор `natlesseq-next` применим только если первый аргумент больше 0; при этом данный конструктор требует значение типа с определёнными аргументами в качестве своего аргумента.

Будем говорить, что $a \preceq b$ тогда и только тогда, когда `NatLessEq a b` обитаем. Например, утверждение $1 \preceq 3$ доказывается так:

```
\func one-le-three : NatLessEq 1 3 => natlesseq-next (natlesseq-zero)
```

В самом деле, `natlesseq-zero` может являться конструктором типа `NatLessEq 0 b`, а тогда

```
natlesseq-next (natlesseq-zero) : NatLessEq 1 (b+1)
```

Унифицировать $b+1$ и 3 компилятор (в данном случае) может самостоятельно, и потому код выше проходит проверку на корректность.

Докажите (везде предполагается, что $a, b, c : \mathtt{Nat}$, если не указано иного):

   (a) $a \preceq b$ тогда и только тогда, когда a меньше или равно b в смысле натуральных чисел (здесь требуется рассуждение на мета-языке).

   (b) $a \preceq a + b + 1$; то есть, определите функцию
   `\func n-less-sum (a b : Nat) : NatLessEq a (a Nat.+ suc b)`

   (c) Если $a \preceq b$, то $a + c \preceq b + c$

   (d) Если $a \preceq b$ и $c \preceq d$, то $a \cdot c \preceq b \cdot d$

   (e) $a \preceq 2^a$

   (f) Транзитивность: если $a \preceq b$ и $b \preceq c$, то $a \preceq c$

   (g) $a \preceq b \vee b \preceq a$

   (h) Найдите стандартное определение отношения «меньше» в библиотеке Аренда (`Nat.<`) и докажите, что $a \preceq b$ тогда и только тогда, когда $a < b$ или $a = b$ (реализуйте функции
   `there (p : NatLessEq a b) : Data.Or (a Nat.< b) (a = b)` и обратную к ней).

   (i) Покажите, что $a \preceq b$ тогда и только тогда, когда $\exists k^{\mathbb{N}_0}.a + k = b$.

11. С точки зрения изоморфизма Карри-Ховарда индуктивные типы можно воспринимать как аналоги предикатов. В этом задании надо построить индуктивные типы для различных предикатов:

   (a) Факториал (`IsFact n`), который будет обитаем только для таких n, что $n = k!$. Докажите на языке Аренд, что `IsFact` $(1 \cdot 2 \cdot 3 \cdot \dots \cdot n)$ всегда обитаем, а тип `IsFact 3` — необитаем.

   (b) Наибольший общий делитель двух чисел `GCD x a b`; *указание/пожелание:* воспользуйтесь алгоритмом Эвклида.

   (c) Ограниченное натуральное число `IndFin n`, обитателями типа являются только те числа, которые меньше n. В стандартной библиотеке `Fin` определён через натуральные числа; сделайте это исключительно через индуктивные типы. Покажите, что если `x : IndFin m` и `y : IndFin n`, то `x + y : IndFin (m + n)`.