Теоретические домашние задания

Теория типов, ИТМО, M3232-M3239, весна 2025 года

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

1. Бесконечное количество комбинаторов неподвижной точки. Дадим следующие определения

$$\begin{array}{l} L := \lambda abcdefghijklmnopqstuvwxyzr.r(thisisafixedpointcombinator) \\ R := LLLLLLLLLLLLLLLLLLLLLLLLLL \end{array}$$

   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 32 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. 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):

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

   Заметим, что данные равенства объясняют смысл названий комбинаторов:

S	verSchmelzung, «сплавление»
K	Konstanz
I	Identität

4. 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 $(\varphi \to \varphi \to \psi) \to (\varphi \to \psi)$.

5. 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.

6. Сформулируем определение бета-редукции на языке пред-лямбда-термов. $A \to_\beta B$, 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.

7. Две задачи на вычисление СЗНФ при помощи нормального порядка редукции.

   (a) Постройте функцию прибавления 1 к значениям из списка в лямбда-исчислении: `let plus1 l = map (fun x -> x+1) l`. Постройте бесконечный список из нечётных чисел `[1,3,..]`. Примените функцию `plus1` к списку и получите результат в СЗНФ.

   (b) Напишите функцию вычисления суммы первых двух элементов списка: `let compute (l1::l2::ls) = (l1+l2, ls)`, примените её к результату предыдущего пункта, получите результат в СЗНФ.

8. 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

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

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

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

   (a) Покажите, что в исчислении по Карри не выполняется свойство расширения типизации (subject expansion) даже в такой формулировке: если $\vdash_\text{к} M : \sigma$, $M \twoheadrightarrow_\beta N$ и $\vdash_\text{к} N : \tau$, то необязательно, что $\sigma = \tau$.

   (b) Покажите, что в исчислении по Чёрчу свойство расширения типизации в такой формулировке также не выполняется:

   найдутся такие M, N, σ, что $\vdash_\text{ч} N : \sigma$, $M \twoheadrightarrow_\beta N$, но $\not\vdash_\text{ч} M : \sigma$.

   Но при этом в исчислении по Чёрчу выполнено свойство расширения типизации в такой формулировке:

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

Домашнее задание №2: теорема о сильной нормализуемости просто типизированного лямбда-исчисления

1. Найдите $[\![\alpha \to \alpha]\!]$ и $[\![(\alpha \to \alpha) \to \alpha]\!]$.

2. Покажите, что $[\![\alpha]\!]$ насыщенное, и что если $\mathcal{A}$ и $\mathcal{B}$ насыщены, то $\mathcal{A} \to \mathcal{B}$ насыщенное.

3. Покажите, что SN — насыщенное (постройте полноценные рассуждения по индукции для определения).

Домашнее задание №3: 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.