

ТЕОРЕТИЧЕСКИЕ (“МАЛЫЕ”) ДОМАШНИЕ ЗАДАНИЯ

Теория типов, ИТМО, М3334-М3339, осень 2018 года

Домашнее задание №1: «знакомство с лямбда-исчислением»

1. Расставьте скобки:

$\lambda f.\lambda x.f\ x\ (\lambda c.g\ f)\ x\ a\ \lambda b.\lambda a.x$

2. Проведите бета-редукции и приведите выражения к нормальной форме:

(a) $(\lambda f.\lambda x.f\ (f\ x))\ (\lambda f.\lambda x.f\ (f\ x))$

(b) $(\lambda a.\lambda b.b)\ ((\lambda x.x\ x)\ (\lambda x.x\ x\ x))$

3. Выразите следующие функции в лямбда-исчислении:

(a) Or, Xor

(b) isZero (T, если аргумент равен 0, иначе F)

(c) isEven (T, если аргумент чётный)

(d) умножение на 2, умножение

(e) возведение в степень

(f) вычитание 1, вычитание

Домашнее задание №2: «пропущенные теоремы лямбда-исчисления»

Докажите следующие леммы, упомянутые, но недоказанные на лекции:

1. Если отношение R обладает ромбовидным свойством, то и отношение R^* (транзитивное и рефлексивное замыкание R) также им обладает.
2. Отношение альфа-эквивалентности является отношением эквивалентности.
3. Если $P_1 \Rightarrow_\beta P_2$ и $Q_1 \Rightarrow_\beta Q_2$, то $P_1[x := Q_1] \Rightarrow_\beta P_2[x := Q_2]$.
4. (\Rightarrow_β) обладает ромбовидным свойством.
5. $(\Rightarrow_\beta) \subseteq (\rightarrow_\beta)^*$
6. $(\rightarrow_\beta) \subseteq (\Rightarrow_\beta)$

Домашнее задание №3: «просто типизированное лямбда-исчисление»

1. Докажите лемму о промежуточных типах (Generation lemma, 3.1.6 из Morten Heine B. Sørensen, Pawel Urzyczyn: Lectures on the Curry-Howard Isomorphism). А именно, покажите, что:
 - (a) $\Gamma \vdash x : \tau$ влечёт $x : \tau \in \Gamma$.
 - (b) $\Gamma \vdash MN : \sigma$ влечёт существование типа τ , такого, что $\Gamma \vdash M : \tau \rightarrow \sigma$ и $\Gamma \vdash N : \tau$.
 - (c) $\Gamma \vdash \lambda x.M : \sigma$ влечёт существование типов τ и ρ , таких, что $\Gamma, x : \tau \vdash M : \rho$ и $\sigma = \tau \rightarrow \rho$
2. Докажите лемму о подстановке (Substitution lemma, 3.1.8):
 - (a) Обозначим за $\sigma[\alpha := \tau]$ (за $\Gamma[\alpha := \tau]$) замену всех элементарных типов α на тип τ в типе σ (во всех типах в Γ). Тогда, если $\Gamma \vdash M : \sigma$, то $\Gamma[\alpha := \tau] \vdash M : \sigma[\alpha := \tau]$.
 - (b) Если $\Gamma, x : \tau \vdash M : \sigma$ и $\Gamma \vdash N : \tau$, то $\Gamma \vdash M[x := N] : \sigma$.
3. Докажите лемму о редукции терма (Subject reduction proposition, 3.1.9): если $\Gamma \vdash M : \sigma$ и $M \rightarrow_\beta N$, то $\Gamma \vdash N : \sigma$.
4. Пользуясь предыдущими пунктами, покажите, что Y нетипизируем в просто типизированном лямбда-исчислении.

5. Найдите терм M и два различных типа σ и τ , что $\vdash M : \sigma$ и $\vdash M : \tau$. А существует ли терм M , имеющий в точности один тип?
6. Покажите, что лемма о редукции терма не работает «в обратную сторону». А именно, что:
 - (a) Найдутся M , N и τ , что $\vdash N : \tau$, $M \rightarrow_{\beta} N$, но M не имеет типа.
 - (b) Найдутся M , N , σ и τ , что $\vdash M : \sigma$, $\vdash N : \tau$ и $M \rightarrow_{\beta} N$, но $\nvdash M : \tau$.

Домашнее задание №4: «просто типизированное лямбда-исчисление; алгебраические типы»

1. Списки и алгебраические типы. В данном задании потребуются строить и преобразовывать довольно сложные лямбда-выражения. Для проверки рекомендуем пользоваться интерпретатором, например, можно взять LCI: <https://chatziko.github.io/lci/>. Возможно, для демонстрации домашнего задания вам потребуется использовать свой ноутбук и проектор.
 - (a) Определите алгебраический тип для списка целых чисел в вашем любимом языке программирования. На Окамле это будет `type int_list = Nil | Cons of (int * int_list)`. Вы можете использовать и не функциональный язык (C++, Kotlin и т.п.), но вы должны применять именно алгебраический тип или его аналог (то есть, `union` в C++, `sealed class` в Kotlin и т.п.).
 - (b) Напишите функции вычисления длины списка, подсчёта суммы списка, произведения списка.
 - (c) Определите функцию высшего порядка `map` (применяющую переданную параметром функцию к каждому элементу списка), и примените её для построения списка нулей (превратить список чисел в список нулей той же длины), удвоенных значений (превратить список $[1, 3, 5]$ в $[2, 6, 10]$), списка остатков от деления на 2 ($[2, 3, 5]$ в $[0, 1, 1]$).
 - (d) Перепишите весь код из предыдущих пунктов в чистых лямбда-выражениях, используя рассмотренные на лекции представления в лямбда-исчислении для упорядоченных пар и алгебраических типов.
2. Ещё немного алгебраических типов. Аналогично предыдущему пункту, определите на языке высокого уровня алгебраический тип для корней квадратного уравнения. Варианты значений: «нет решений» без параметров, «одно решение» с одним параметром, «два решения» с двумя параметрами. Определите функции вычисления корней по коэффициентам квадратного уравнения и печати корней.
3. Деревья с помощью алгебраических типов. Определите на языке высокого уровня тип для дерева двоичного поиска, варианты для узла: «лист» без параметров и «ветвь» с двумя сыновьями и целочисленным значением. Определите:
 - (a) функцию печати дерева;
 - (b) функцию поиска значения в дереве;
 - (c) функцию добавления значения в дерево двоичного поиска;
 - (d) функцию удаления значения из дерева.
4. Доопределите бета-редукцию для просто типизированного лямбда-исчисления по Чёрчу.
5. Докажите теорему Чёрча-Россера для просто типизированного лямбда-исчисления по Чёрчу.
6. Докажите лемму о поднятии с лекции, а именно, что для всех $M, N \in \Lambda_{\mathbf{q}}$:
 - (a) если $M \rightarrow_{\beta} N$, то для любого $M' \in \Lambda_{\mathbf{q}}$, такого, что $|M'| = M$, найдётся такой $N' \in \Lambda_{\mathbf{q}}$, что $|N'| = N$ и $M' \rightarrow_{\beta} N'$;
 - (b) если $\Gamma \vdash M : \sigma$, то найдётся такой $M' \in \Lambda_{\mathbf{q}}$, что $\Gamma \vdash_{\mathbf{q}} M' : \sigma$.

Домашнее задание №5: «выразительная сила просто типизированного лямбда-исчисления, алгоритм унификации»

1. Покажите, что чёрчевский нумерал в общем случае имеет тип $(\alpha \rightarrow \alpha) \rightarrow (\alpha \rightarrow \alpha)$. Имеют ли нумералы для 0, 1 и 2 какие-то более общие типы?
2. Обозначим тип для целых чисел $\eta = (\alpha \rightarrow \alpha) \rightarrow (\alpha \rightarrow \alpha)$. В данных обозначениях покажите, что операция сложения имеет тип $\eta \rightarrow \eta \rightarrow \eta$.
3. Напомним, что $\overline{m} = \lambda f.\lambda x.f^{(m)} x$ (чёрчевский нумерал для m). Рассмотрим выражение $Power = \lambda m.\lambda n.n\ m$.
 - (a) найдите тип $Power$;
 - (b) покажите, что $Power\ \overline{m}\ \overline{n} = \overline{m^n}$, найдите тип $Power\ \overline{m}\ \overline{n}$;
 - (c) покажите, что $\lambda x.Power\ x\ x$ не имеет типа;
 - (d) объясните кажущееся противоречие между предыдущими пунктами: почему $Power\ \overline{2}\ \overline{2}$ имеет тип, а $(\lambda x.Power\ x\ x)\ \overline{2}$ не имеет типа.
4. Докажите, что изложенный на лекции алгоритм унификации всегда завершается. Указание: постройте оценку сложности уравнения в алгебраических термах и покажите, что эта оценка уменьшается при каждом шаге алгоритма.

Домашнее задание №6: «унификация и типы, комбинаторы»

1. Выразите $\lambda f.\lambda x.f\ x$ через S и K .
2. Докажите, что алгоритм устранения абстракций T с лекции, преобразующий замкнутое лямбда-выражение в выражение в комбинаторах S и K , корректен. То есть, для любого лямбда-выражения A :
 - (a) $T(A)$ определено и вычисляется за конечное время;
 - (b) $T(A) =_{\beta} A$;
 - (c) если A замкнуто, то $T(A)$ не содержит абстракций и свободных переменных и состоит только из применений (аппликаций) и комбинаторов S и K .
3. Покажите, что базис B, C, K, W позволяет выразить любое замкнутое лямбда-выражение.
4. Постройте систему аксиом для импликационного фрагмента просто типизированного лямбда-исчисления на основе базиса B, C, K, W .
5. Будем говорить, что тип σ есть частный случай типа θ (и записывать это как $\sigma \subseteq \theta$), если существует такая подстановка S , что $\sigma = S(\theta)$. Рассмотрим лямбда-выражение M , такое, что $\vdash M : \sigma$ и $\vdash M : \theta$.
 - (a) Покажите, что найдётся тип τ , что $\vdash M : \tau$, $\sigma \subseteq \tau$ и $\theta \subseteq \tau$.
 - (b) Всегда ли найдётся τ , что $\tau \subseteq \sigma$ и $\tau \subseteq \theta$?
 - (c) Всегда ли выполнено либо $\theta \subseteq \sigma$, либо $\sigma \subseteq \theta$?
 - (d) Можно ли определить решётку на типах для данного M с определённым выше отношением предпорядка (\subseteq)? Естественно, вам потребуется рассмотреть классы эквивалентности типов, чтобы «склеить» случаи типов, отличающихся только переименованием переменных. Какими свойствами эта решётка будет обладать (дистрибутивность, имплицативность, существование 0 и т.п.)?

Домашнее задание №7: «исчисление второго порядка, система F»

1. Докажите, что введённые на лекции представления для связок соответствуют правилам для связок:
 - (a) Конъюнкция. Если $\varphi \& \psi \equiv \forall \alpha.(\varphi \rightarrow \psi \rightarrow \alpha) \rightarrow \alpha$, то всегда можно доказать заключения следующих правил при доказанных посылках:

$$\frac{\Gamma \vdash \varphi \quad \Gamma \vdash \psi}{\Gamma \vdash \varphi \& \psi} \qquad \frac{\Gamma \vdash \varphi \& \psi}{\Gamma \vdash \varphi} \qquad \frac{\Gamma \vdash \varphi \& \psi}{\Gamma \vdash \psi}$$

(b) Дизъюнкция. Если $\varphi \vee \psi \equiv \forall \alpha. (\varphi \rightarrow \alpha) \rightarrow (\psi \rightarrow \alpha) \rightarrow \alpha$, то можно показать такие правила:

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \vee \psi} \quad \frac{\Gamma \vdash \psi}{\Gamma \vdash \varphi \vee \psi} \quad \frac{\Gamma, \varphi \vdash \pi \quad \Gamma, \psi \vdash \pi}{\Gamma \vdash \pi} \quad \frac{\Gamma \vdash \varphi \vee \psi}{\Gamma \vdash \pi}$$

(c) Квантор существования. Если $\exists \alpha. \varphi \equiv \forall \theta. (\forall \alpha. \varphi \rightarrow \theta) \rightarrow \theta$, то можно показать и следующие правила:

$$\frac{\Gamma \vdash \varphi[\alpha := \theta]}{\Gamma \vdash \exists \alpha. \varphi} \quad \frac{\Gamma \vdash \exists \alpha. \varphi \quad \Gamma, \varphi \vdash \psi}{\Gamma \vdash \psi} \quad (\alpha \notin FV(\Gamma, \psi))$$

Домашнее задание №8: «экзистенциальные типы, ранг типа»

1. Докажите, что если $\sigma \in R(1)$, то существует ψ с поверхностными кванторами, что $\psi \vdash \sigma$ и $\sigma \vdash \psi$.
2. Докажите, что если $M : \sigma$ и $\sigma \in R(1)$, то существует ψ с поверхностными кванторами и лямбда-выражения N, F и B в системе F, такие, что $N : \psi, F : \sigma \rightarrow \psi$ и $B : \psi \rightarrow \sigma$.
3. Пусть задан экзистенциальный тип $\exists \alpha. \sigma$. Если рассматривать экзистенциальные типы в терминах АТД, то α здесь — переменная для типа структуры данных (массив, список и т.п.), а σ — интерфейс (например, рассматривавшийся на лекции интерфейс стэка: $\alpha \& (\alpha \rightarrow \nu \rightarrow \alpha) \& (\alpha \rightarrow (\nu \& \alpha))$).

Также пусть задана реализация этого типа (актуальный тип для структуры данных — β , и актуальная реализация интерфейса — T).

(a) Операция *упаковки*: обозначим за

$$\text{pack } \{\beta, T\} \text{ to } \exists \alpha. \sigma$$

выражение

$$\Lambda \gamma. \lambda x : (\forall \alpha. \sigma \rightarrow \gamma). x \beta T$$

Покажите выполнение правила

$$\frac{\Gamma \vdash T : \sigma[\alpha := \beta]}{\Gamma \vdash \text{pack } \{\beta, T\} \text{ to } \exists \alpha. \sigma : \exists \alpha. \sigma}$$

(b) Операция *распаковки*: обозначим за

$$\text{unpack } \{\alpha, x : \sigma\} = T \text{ in } E^\delta$$

выражение

$$T \delta (\Lambda \alpha. \lambda x : \sigma. E)$$

Покажите выполнение правила

$$\frac{\Gamma \vdash T : \exists \alpha. \sigma \quad \Gamma, x : \sigma \vdash E : \delta}{\Gamma \vdash \text{unpack } \{\alpha, x : \sigma\} = T \text{ in } E^\delta : \delta}$$

(c) Пункт убран.

4. Рассмотрим абстрактный тип данных **NatNum** («натуральное число»), с операциями: (а) константа 0; (б) сравнение двух значений; (в) прибавление 1; (г) печать в строку. Задайте его на языке Хаскель с помощью экзистенциальных типов, предусмотрите три реализации (через **Integer**, аксиоматику Пеано и в виде строчек из цифр), напишите функции: (—) печати списка чисел в строку (**[NatNum] -> String**); (—) удвоения каждого числа в списке (**[NatNum] -> [NatNum]**); (—) суммы списка (**[NatNum] -> NatNum**).

Указание: напомним, что значение экзистенциального типа — это функция, берущая на вход функцию, проводящую вычисление с абстрактным типом данных. Этот абстрактный тип доступен только внутри вызова, наружу его передать нельзя. Поэтому, чтобы увидеть результат исполнения функций (—), (—) и (—), нужно также обернуть их в какое-то вычисление, выдающее «не-абстрактный» результат (например, текстовую строку), и уже это вычисление передавать внутрь значения экзистенциального типа.

Смотрите пример реализации <https://github.com/shd/tt2018/blob/master/existential.hs>

Домашнее задание №9: «Idris, введение»

1. Определите в языке Идрис конъюнкцию и дизъюнкцию. Определите все стандартные операции для них (инъекции, проекции и т.п.): эти операции, очевидно, будут доказательством некоторых утверждений в интуиционистской логике. Какие это утверждения, приведите их.
2. Определите функцию `swap: Vect n a -> (Fin n) -> (Fin n) -> Vect n a`, строящую новый вектор, в котором два элемента вектора поменяны местами.
3. Определите функции арифметики для `Fin`:
(a) `plus_fin: Fin a -> Fin b -> Fin (a+b)`
(б) `mul_fin: Fin a -> Fin b -> Fin (a*b)`
(в) `dec_fin: Fin (S a) -> Fin a`
4. Определите функции минимума для натуральных (Пeano) и конечных чисел:
`nat_min: Nat -> Nat -> Nat`
`min_fin: {a:Nat} -> {b:Nat} -> Fin (S a) -> Fin (S b) -> Fin (S (nat_min a b))`
Также определите функции:

```
map2: {X:Type} -> {Y:Type} -> {Z:Type} -> (a:Nat) -> (b:Nat) ->  
      (X->Y->Z) -> Vect a X -> Vect b Y -> Vect (nat_min a b) Z  
  
index2: {X:Type} -> {Y:Type} -> (a:Nat) -> (b:Nat) -> Fin (nat_min a b) ->  
       Vect a X -> Vect b Y -> (X,Y)
```

Домашнее задание №10: «Идрис, простые доказательства»

Ещё раз напомним основную идею доказательства: доказать утверждение σ — это построить такое выражение M , что $\vdash M : \sigma$.

В языках с типовой системой Хиндли-Милнера существует алгоритм, разрешающий задачу типизации: там для вывода типов достаточно самого выражения M и типа σ . В Идрисе, с учётом сложности языка, задача типизации неразрешима, поэтому компилятор может догадаться до типа сам только в очень простых случаях. В остальных ситуациях компилятору нужны подсказки.

Например, заметить, что `Refl: 0+z=z+0`, компилятор сам не может (сразу напомним, что если тип значений компилятору известен, то мы можем писать `+` вместо `plus`, `0` вместо `Z`, `1` вместо `S Z`). Ему можно это пояснить, написав функцию, проводящую доказательство по индукции, на лекции мы рассматривали такой её вариант:

```
plus_zero_commutative : (a : Nat) -> (0+a) = (a+0)  
plus_zero_commutative Z = Refl  
plus_zero_commutative (S k) = rewrite (plus_zero_commutative k) in Refl
```

Однако, магия конструкции `rewrite` оставила много вопросов, поэтому давайте копнём глубже, и разберём этот пример с точки зрения функции `replace`, которую использует `rewrite`.

```
replace: (x=y) -> P x -> P y.
```

Функция `replace` берёт два явных параметра, и один неявный (P). P — это некоторый тип, зависящий от x . Функция имеет естественный смысл: если два значения равны, и свойство выполнено для одного из них, то оно выполнено и для другого.

Неявность P предполагает, что компилятор может догадаться до того, что это за значение, но на практике он не справляется с этой задачей. Поэтому обычно P нужно указывать.

Давайте поймём, что это должен быть за P . В случае `plus_zero_commutative` мы, имея предположением индукции $0+a=a+0$, должны доказать $0 + (S a) = (S a) + 0$. Логично взять предположение за равенство $x = y$, при этом x будет соответствовать $0 + a$, а y будет соответствовать $a + 0$. Теперь подберём такое P , чтобы $P (0 + a)$ унифицировалось с $0 + (S a)$, а $P (a + 0)$ унифицировалось с $(S a) + 0$.

Давайте возьмём $P = \lambda w \Rightarrow S (0+a)=S w$. Тогда $P (0 + a)$ — это $S (0+a) = S (0+a)$ (что очевидно доказывается при помощи `Refl`), а $P (a+0)$ — это $S (0+a) = S (a+0)$ (что является требуемым утверждением, так как $S (0+a) = S(a) = 0+S(a)$; компилятор, как мы обсуждали, способен короткие цепочки подобных преобразований производить самостоятельно).

Итак, мы получили следующий код:

```

plus_zero_commutative: (a:Nat) -> 0 + a = a + 0
plus_zero_commutative Z = Refl
plus_zero_commutative (S a) =
  replace {P = \w => S (0+a)=S w} (plus_zero_commutative a) Refl

```

В отличие от `replace`, конструкция `rewrite` имеет дополнительный эвристический алгоритм для подбора соответствующего P , поэтому в части случаев мы можем довериться её интеллекту.

1. Свойства равенства. Докажите, что:

- (a) $x = y \rightarrow y = x$
- (b) $x = y \rightarrow y = z \rightarrow x = z$
- (c) (конгруэнтность) $(P: A \rightarrow B) \rightarrow x = y \rightarrow P\ x = P\ y$

2. Простая арифметика — сложение. Докажите, что:

- (a) $x = x + 0$
- (b) $S\ x = 1 + x$
- (c) $S\ x = x + 1$
- (d) $S\ x + S\ x = S\ (S\ (x + x))$
- (e) $S\ x + S\ y = S\ (S\ (x + y))$
- (f) $S\ (x + y) = x + (S\ y)$
- (g) $x + y = y + x$
- (h) $x + (y + z) = (x + y) + z$

3. Простая арифметика — умножение. Докажите, что:

- (a) $0 = x * 0$
- (b) $0 = 0 * x$
- (c) $x = 1 * x$
- (d) $x = x * 1$
- (e) $x * y = y * x$
- (f) $x * (y * z) = (x * y) * z$
- (g) $x * (y + z) = x * y + x * z$

4. Отношение «меньше или равно» определено в библиотеке Идрис так:

```

data LTE : (n : Nat) -> (m : Nat) -> Type where
  LTEZero : LTE 0 right  -- Zero is the smallest Nat
  LTSucc   : LTE left right -> LTE (S left) (S right)
  -- If n <= m, then n + 1 <= m + 1

```

- (a) Докажите, что $LTE\ 3\ 5$
- (b) Докажите, что $LTE\ x\ y \rightarrow LTE\ x\ (S\ y)$
- (c) Докажите, что $LTE\ x\ y \rightarrow LTE\ (x+n)\ (y+n)$

5. Определите отношение «строго больше», GT . Докажите, что

```
(x:Nat) -> (y:Nat) -> Either (LTE x y) (GT x y)
```

6. Определите ограниченное вычитание `sub` (`sub x y` равно 0, если $x < y$), докажите:

- (a) $LTE\ x\ y \rightarrow sub\ x\ y = 0$
- (b) $sub\ x\ y = 0 \rightarrow LTE\ x\ y$
- (c) $LTE\ y\ x \rightarrow y + (sub\ x\ y) = x$

Домашнее задание №11: «Идрис, Σ и Π -типы»

Данное домашнее задание предлагается для того, чтобы желающие могли самостоятельно разобраться в заключительных темах курса, не разобранных на практике.

1. Рассмотрим обобщённую типовую систему $\lambda\omega$ (сильная ω).

- (a) Рассмотрим тип в Хаскеле: `data Either a b = Left a | Right b`. Предложите его реализацию в обобщённой типовой системе. Выразите конструкторы `Left` и `Right` в ней, укажите (и докажете) их тип.
- (b) Рассмотрим функции `curry f = \x->\y->f (x,y)` и `uncurry f = \(x,y) -> f x y`. Выразите их и их типы в обобщённой типовой системе, докажете их типы.
- (c) Выразите тип Y -комбинатора через Π -тип (будем считать, что лямбда-исчисление расширено данным комбинатором).
- (d) Выразите функцию `repeat n f x = if n > 0 then f (repeat (n-1) f x) else x` и укажите её тип. Требуется ли Y -комбинатор для её задания?

2. Теперь рассмотрим λC и язык Идрис.

- (a) Рассмотрим тип равенства. Выразите его в λC , укажите и докажете его тип. Напомню, что равенство — это обобщённый алгебраический тип.
- (b) Выразите в λC функцию `(+)` для целых чисел, укажите и докажете её тип.
- (c) Выразите в λC доказательства `a = 0 + a` и `a = a + 0` (аналогично, выведите тип этих доказательств).
- (d) Рассмотрим тип `Vect a x` из языка Идрис. Перенесите его определение в «чистое» лямбда-исчисление в выбранной обобщённой типовой системе, укажите (и докажете) его тип.
- (e) Рассмотрим функцию `head: (a: Num) -> (x: Type) -> Vect (S a) x -> x`, возвращающую первый элемент вектора. Выразите её сигнатуру в λC . Перенесите её определение в λC , докажете тип.
- (f) Укажите сигнатуру в λC для `filter: (a: Num) -> (x: Type) -> (x -> Bool) -> Vect a x -> (b: Num ** Vect b x)`. Выразите её в данной типовой системе. Докажете её тип.
- (g) Докажите в Идрисе, что если $\exists y.x = S(S(y))$, то $x = y + 2$. Переведите это доказательство в λC , выведите тип доказательства.