

ТЕОРЕТИЧЕСКИЕ (“МАЛЫЕ”) ДОМАШНИЕ ЗАДАНИЯ

Теория типов, ИТМО, М3235-М3239, весна 2020 года

Домашнее задание №1: «вводная лекция для ТТ и ФП»

1. Напомним определения с лекций:

Обозначение	лямбда-терм	название
T	$\lambda a. \lambda b. a$	истина
F	$\lambda a. \lambda b. b$	ложь
Not	$\lambda x. x \ F \ T$	отрицание
And	$\lambda x. \lambda y. x \ y \ F$	конъюнкция

Проредуцируйте следующие выражения и найдите нормальную форму:

- (a) $T \ F$
 - (b) $(T \ Not \ (\lambda t. t)) \ F$
 - (c) $And \ F \ T$
 - (d) $And \ T \ T$
2. Постройте лямбда-выражения для следующих булевских выражений:
- (a) Дизъюнкция
 - (b) Штрих Шеффера («и-не»)
 - (c) Исключающее или
3. Напомним определения с лекций:

$$f^{(n)} \ X ::= \begin{cases} X, & n = 0 \\ f^{(n-1)} \ (f \ X), & n > 0 \end{cases}$$

Обозначение	лямбда-терм	название
\bar{n}	$\lambda f. \lambda x. f^{(n)} \ x$	чёрчевский нумерал
$(+1)$	$\lambda n. \lambda f. \lambda x. n \ f \ (f \ x)$	прибавление 1
$IsZero$	$\lambda n. n \ (\lambda x. F) \ T$	проверка на 0

Используя данные определения, постройте выражения для следующих операций над числами:

- (a) Сложение
 - (b) Умножение на 2 ($Mul2$)
 - (c) Умножение
 - (d) Возведение в степень
 - (e) Проверка на чётность
 - (f) Деление на 3 (могут потребоваться пары и/или вычитания)
 - (g) Сравнение двух чисел ($IsLess$) — истина, если первый аргумент меньше второго (могут потребоваться пары и/или вычитания)
4. Проредуцируйте выражение и найдите его нормальную форму:
- (a) $\bar{2} \ \bar{2}$
 - (b) $\bar{2} \ \bar{2} \ \bar{2}$
 - (c) $\bar{2} \ \bar{2} \ \bar{2} \ \bar{2} \ \bar{2} \ \bar{2} \ \bar{2}$
5. Напомним определения с лекций:

Обозначение	лямбда-терм	название
$MkPair$	$\lambda a. \lambda b. (\lambda x. x \ a \ b)$	создание пары
PrL	$\lambda p. p \ T$	левая проекция
PrR	$\lambda p. p \ F$	правая проекция
$Case$	$\lambda l. \lambda r. \lambda c. c \ l \ r$	case для алгебраического типа
InL	$\lambda l. (\lambda x. \lambda y. x \ l)$	левая инъекция
InR	$\lambda r. (\lambda x. \lambda y. y \ r)$	правая инъекция

- (a) Убедитесь, что $PrL (MkPair a b) \rightarrow_{\beta} a$.
 - (b) Убедитесь, что $Case (\lambda x.T) (\lambda y.y) (InR p) \rightarrow_{\beta} p$.
 - (c) Постройте операцию вычитания 1 из числа
 - (d) Постройте операцию вычитания чисел
 - (e) Постройте операцию деления чисел
6. Напомним определение Y-комбинатора: $\lambda f.(\lambda x.f (x x)) (\lambda x.f (x x))$.
- (a) Покажите, что выражение $Y f$ не имеет нормальной формы;
 - (b) Покажите, что выражение $Y (\lambda f.\bar{0})$ имеет нормальную форму.
 - (c) Покажите, что выражение $Y (\lambda f.\lambda x.(IsZero x) \bar{0} (f Minus1 x)) 2$ имеет нормальную форму.
 - (d) Какова нормальная форма выражения $Y (\lambda f.\lambda x.(IsZero x) \bar{0} ((+1) (f Minus1 x))) \bar{n}$?
 - (e) Какова нормальная форма выражения $Y (\lambda f.\lambda x.(IsZero x) \bar{1} (Mul2 (f Minus1 x))) \bar{n}$?
 - (f) Определите с помощью Y-комбинатора функцию для вычисления n -го числа Фибоначчи.
7. Пусть $\eta = (\alpha \rightarrow \alpha) \rightarrow (\alpha \rightarrow \alpha)$. Покажите (т.е. постройте соответствующее доказательство в исчислении по Карри), что:
- (a) $\vdash \bar{2} : \eta$.
 - (b) $\vdash (+1) : \eta \rightarrow \eta$.
 - (c) $\vdash Plus : \eta \rightarrow \eta$.
 - (d) $\vdash Mul : \eta \rightarrow \eta$ (не каждая реализация умножения будет удовлетворять этому свойству; вам требуется найти нужную)
8. Определим на языке Хаскель следующую функцию: `show_church n = show (n (+1) 0)` Убедитесь, что `show_church (\f -> \x -> f (f x))` вернёт 2. Пользуясь данным определением и его идеей, реализуйте следующие функции:
- (a) `int_to_church` — возвращает чёрчевский нумерал (т.е. функцию от двух аргументов) по целому числу. Каков точный тип результата этой функции?
 - (b) сложение двух чёрчевских нумералов.
 - (c) умножение двух чёрчевских нумералов.
 - (d) можно ли определить вычитание 1 и вычитание? Что получается, а что — нет?
9. Типы для конъюнкции и дизъюнкции на Хаскеле. Списки.
- Заметим, что список (например, целых чисел) — это алгебраический тип:
- ```
List = Nil | Cons Integer List.
```
- Можно сконструировать значение данного типа: `Cons 3 (Cons 5 Nil)`. Можно, например, вычислить его длину:
- ```
length Nil = 0
length (Cons _ tail) = length tail + 1
```
- Определим $Nil = InL 0$, а $Cons a b = InR (MkPair a b)$. Заметим, что теперь списки могут быть напрямую перенесены в лямбда выражения. Тогда, используя данную идею, реализуйте в Хаскеле:
- (a) определите конструкции `mkipair`, `prl`, `prg` на Хаскеле — какой тип у данных конструкций? Сравните его с типом конъюнкции с лекции.
 - (b) определите конструкции `case`, `inl`, `inr` — какой тип у данных конструкций? Сравните его с типом дизъюнкции с лекции.
 - (c) постройте список целых чисел из данных конструкций.
 - (d) определите функцию вычисления длины списка целых чисел с помощью данных конструкций (к сожалению, скомпилировать это выражение на Хаскеле не получится — поэтому достаточно написать исходный код).

Домашнее задание №2: «формализация лямбда-исчисления»

1. На лекции было использовано понятие свободы для подстановки.

- (a) Найдите лямбда-выражение, которое при однократной редукции требует переименования связанных переменных (редукция невозможна без переименования).
- (b) Заметим, что даже если мы запретим использовать одни и те же переменные в разных лямбда-абстракциях, это не будет решением проблемы переименований. Предложите лямбда-выражение, в котором (a) все лямбда-абстракции указаны по разным переменным; но (б) через некоторое количество редукций потребуется переименование связанных переменных.

2. Дадим определение: комбинатор — лямбда-выражение без свободных переменных.

Также напомним определение:

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

$$K := \lambda x. \lambda y. x$$

$$I := \lambda x. x$$

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 .

Выразите в базисе SK :

- (a) $F = \lambda x. \lambda y. y$
- (b) $\bar{1}$
- (c) Not
- (d) Xor
- (e) InL
- (f) \bar{n}

3. Бесконечное количество комбинаторов неподвижной точки. Дадим следующие определения

$$L := \lambda abcdefghijklmnopqrstuvwxyzr. r(this is a fixed point combinator)$$

$$R := LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL$$

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.
4. Пусть задано $n \in \mathbb{N}$. Постройте лямбда-выражение, которое преобразуется в нормальную форму в n раз медленнее с помощью нормального порядка редукции, чем с помощью какого-то другого (самого быстрого) порядка редукции.
5. Чёрчевские нумералы соответствуют натуральным числам в аксиоматике Пеано.
- (a) Предложите «двоичные нумералы» — способ кодирования чисел, аналогичный двоичной системе (такой, при котором длина записи числа соответствует логарифму числового значения).
 - (b) Предложите реализацию функции $(+1)$ в данном представлении.
 - (c) Предложите реализацию лямбда-выражения преобразования числа из двоичного нумерала в чёрчевский.
 - (d) Предложите реализацию функции сложения в данном представлении.
 - (e) Предложите реализацию функции вычитания в данном представлении.
 - (f) Какова вычислительная сложность арифметопераций с двоичными нумералами?
6. Предложим альтернативные аксиомы для конъюнкции:

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

- (а) Предложите лямбда-выражения, соответствующие данным аксиомам; поясните, как данные выражения абстрагируют понятие «упорядоченной пары».
- (б) Выразите изложенные в лекции аксиомы конъюнкции через приведённые в условии.
- (с) Выразите приведённые в условии аксиомы конъюнкции через изложенные в лекции.
7. Как мы уже разбирали, $\not\vdash x : \tau$ в силу дополнительных ограничений аксиомы

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

Найдите лямбда-выражение N , что $\not\vdash N : \tau$ в силу ограничения аксиомы

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

Домашнее задание №3: «вывод типов; алгоритм унификации»

- Вполне упорядоченным множеством назовём такое линейно-упорядоченное отношение ($<$) множество S (и такой порядок назовём *полным*), что какое бы ни было множество $U \subseteq S$, в U найдётся наименьший элемент.
 - Покажите, что неотрицательные вещественные числа $[0, +\infty)$ — не вполне упорядоченное множество. Существуют ли конечные и счётные не вполне упорядоченные множества?
 - Определим лексикографический порядок на \mathbb{N}^n : положим, что $\langle a_1, a_2, \dots, a_n \rangle < \langle b_1, b_2, \dots, b_n \rangle$, если найдётся такой k , что $a_1 = b_1, \dots, a_{k-1} = b_{k-1}$, но $a_k < b_k$. Покажите, что такой порядок — полный.
 - Пусть S вполне упорядочено отношением ($<$), определим $a > b := b < a$. Пусть $a_1 > a_2 > a_3 > \dots$ — строго монотонно убывающая последовательность значений из S . Покажите, что данная последовательность всегда имеет конечную длину.
- Рассмотрим полное интуиционистское исчисление высказываний. Дополните алгоритм вывода типов дополнительными функциональными символами для связок $\&$, \vee и \perp (а также сделайте дополнительные необходимые исправления в нём) и продемонстрируйте вывод типов для выражения, использующего хотя бы две из данных трёх конструкций.
- Поясним название «алгебраические типы» — это семейство составных типов, позволяющих строить «алгебраические» выражения на типах:

название	обозначение	алгебраический смысл
тип-сумма, «алгебраический»	$\alpha \vee \beta$	$\alpha + \beta$
тип-произведение, пара	$\alpha \& \beta$	$\alpha \times \beta$
тип-степень, функция	$\alpha \rightarrow \beta$	β^α

Название «алгебраический» закрепилось в первую очередь за типом-суммой (видимо потому, что остальные типы имеют устоявшиеся названия), однако, может быть отнесено и к другим типам.

Поясните «типовый» (программистский) смысл следующих алгебраических тождеств — и постройте программы, их доказывающие:

- $\gamma \times (\alpha + \beta) = \gamma \times \alpha + \gamma \times \beta$.
 - $\gamma^{\alpha \times \beta} = (\gamma^\alpha)^\beta$. Как называется данное тождество?
 - $\gamma^{\alpha + \beta} = \gamma^\alpha \times \gamma^\beta$.
4. Найдите лямбда-выражения, доказывающие:
- Формулу де-Моргана $\neg(\alpha \vee \beta) \rightarrow \neg\alpha \& \neg\beta$.
 - Контрапозицию $(\alpha \rightarrow \beta) \rightarrow (\neg\beta \rightarrow \neg\alpha)$.
 - Закон исключённого третьего после применения теоремы Гливленко $\neg\neg(\alpha \vee \neg\alpha)$.

Домашнее задание №4: «логика второго порядка; система F»

1. Докажите в исчислении предикатов второго порядка теоремы-аналоги правил вывода для следующих связок:

- (a) конъюнкция;
- (b) дизъюнкция;
- (c) отрицание и ложь;
- (d) квантор существования.

Напомним: в задании требуется по заданным аксиоме и аргументам связки найти такое дерево применений правил И.П. 2 порядка, которым можно было бы заменить применение исходной аксиомы.

2. Напомним, что в системе F естественно предложить выражение $\Lambda\alpha.\lambda f^{\alpha \rightarrow \alpha}.\lambda x^\alpha.f^n x$ как аналог для чёртевого нумерала. Постройте в F выражения для следующих функций и выведите их тип:

- (a) сложение;
- (b) умножение;
- (c) возведение в степень
- (d) вычитание 1

3. Перенесите в систему F выражения для булевских значений и операции And и выведите их тип.

4. Выразите в языке Хаскель конструкции системы F через импликацию и квантор всеобщности.

Точнее: запишите выражение, напишите программы, доказывающие соответствующие аксиомы, укажите, как преобразовать данное выражение в нативный хаскелевский тип и наоборот. При реализации вам потребуется использовать квантор всеобщности в типах (`type T = forall x. ...`) и включить опцию RankNTypes.

Список конструкций:

- (a) конъюнкция;
- (b) дизъюнкция;
- (c) отрицание и ложь (соответствует `undefined`).

5. *Экзистенциальные типы.* Укажем правила вывода для квантора существования в системе F:

$$\frac{\Gamma \vdash M : \sigma[\alpha := \tau]}{\Gamma \vdash \mathbf{pack} \ \tau, M \ \mathbf{to} \ \exists\alpha.\sigma : \exists\alpha.\sigma} \text{ Введ. } \exists$$

$$\frac{\Gamma \vdash M : \exists\alpha.\sigma \quad \Gamma, x : \sigma \vdash N : \rho}{\Gamma \vdash \mathbf{abstype} \ \alpha \ \mathbf{with} \ x : \sigma \ \mathbf{is} \ M \ \mathbf{in} \ N : \rho} \text{ Удал. } \exists; \alpha \notin FV(\Gamma, \rho)$$

Постараемся прояснить смысл данных конструкций. В объектно-ориентированных языках программирования обычно хранение данных объединено с библиотечной структурой: структура данных всегда неявно присутствует как параметр методов класса. Аналогия с квантором существования подсказывает, что эта связь необязательна: библиотека характеризуется сигнатурой её методов (σ) и типом, хранящим значения (α).

Если мы хотим задать некоторый абстрактный тип данных, мы используем `pack`. Например, давайте построим АТД «список» в некотором обобщении лямбда-исчисления:

```
let intstack = pack list, ⟨Nil, ⟨(λllist.λxint.Cons x l), (λllist.⟨Head l, Tail l⟩)⟩⟩
to ∃α.α & ((α → int → α) & (α → int & α))
```

Если же мы желаем использовать такой АТД, мы используем `abstype`; вот такой код вернёт число 12:

```
abstype α with x : α & ((α → int → α) & (α → int & α)) is intstack in
let stα = PrL x in
let stα = (PrL (PrR x)) st 12 in
PrL ((PrR (PrR x)) st)
```

В завершение определим правило бета-редукции для данных конструкций.

$$\mathbf{abstype} \ \alpha \ \mathbf{with} \ x : \sigma \ \mathbf{is} \ \mathbf{pack} \ M, \tau \ \mathbf{to} \ \exists\alpha.\sigma \ \mathbf{in} \ N \rightarrow_\beta N[\alpha := \tau][x := M]$$

- (a) Предъявите лямбда-выражение, соответствующее `pack` в системе F (без использования сокращений записи — в частности, без (\exists)). Покажите, что оно действительно имеет приписываемый ему тип.
- (b) Предъявите лямбда-выражение, соответствующее `abstype` в системе F (без (\exists)). Покажите, что оно действительно имеет приписываемый ему тип.
- (c) Рассмотрим вариант системы F, типизированный по Карри (отсутствуют типовые абстракции и применения, а также указания типов в аргументах). Предложите реализацию `pack` и `abstype` в этом варианте исчисления.
- (d) Предложите реализацию на Хаскеле и пример использования для массива фиксированного размера (размер указывается при инициализации), соответствующего интерфейсу, заданному следующим экзистенциальным типом (для компиляции требуется включить опцию `RankNTypes`):


```
data AbstractArray x = AA (forall b . (forall a .
      (Integer -> a, a -> (Integer, x) -> a, a -> Integer -> (a, x)) -> b) -> b)
```
- (e) Предположим, что в Джаве мы не используем наследования, а только разрешаем реализовывать интерфейсы. Предложите формализацию для классов и интерфейсов с использованием экзистенциальных типов. Как реализовать публичные и приватные поля?

Домашнее задание №5: «Типовая система Хиндли-Милнера»

1. Покажите, что если ϕ — тип в системе Хиндли-Милнера, то существует $n \in \mathbb{N}_0$, что $\phi \in R(n)$.
2. Покажите, что если $\phi \in R(n)$ и $n < m$, то $\phi \in R(m)$.
3. Пусть ϕ — формула И.П. ранга 1. Покажите, что существует такое выражение σ с поверхностными кванторами, что $\vdash \phi \rightarrow \sigma$ и $\vdash \sigma \rightarrow \phi$. Для этого:
 - (a) Покажите, что если ϕ — формула ранга 1, то она имеет вид либо ξ , либо $\forall x.\psi$, либо $\xi \rightarrow \psi$, где ψ — формула ранга 1, а ξ — формула ранга 0.
 - (b) Покажите, что если $\phi = \chi \rightarrow \forall x.\psi$, где $x \notin FV(\chi)$, то $\vdash \phi \rightarrow (\forall x.\chi \rightarrow \psi)$ и $\vdash (\forall x.\chi \rightarrow \psi) \rightarrow \phi$.
 - (c) Покажите, что $\vdash (\forall x.\psi) \rightarrow \forall y.\psi[x := y]$ и $\vdash (\forall y.\psi[x := y]) \rightarrow \forall x.\psi$, если y не входит свободно в ψ .
 - (d) Опираясь на утверждения выше, покажите искомое утверждение.
4. О выразительной силе НМ. 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);;
```

Если бы такое можно было выразить в типовой системе Хиндли-Милнера, то операция добавления элемента к списку записалась бы 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)
```

- (a) Какой тип имеет `add` (обратите внимание на ключевое слово `rec`: для точного указания соответствующего лямбда-выражения и вывода типа необходимо использовать Y-комбинатор)? Считайте, что семейство типов `bin_list 'a` предопределено, и обозначается как τ_a . Также считайте, что определены функции `roll` и `unroll` с надлежащими типами.
- (b) Какой ранг имеет тип этой функции? Почему этот тип не выразим в типовой системе Хиндли-Милнера?
- (c) Предложите функцию для удаления элемента списка (головы).

- (d) Предложите функцию для эффективного соединения двух списков (источник для вдохновения — сложение двух чисел в столбик).
- (e) Предложите функцию для эффективного выделения n -го элемента из списка.
5. Используя расширения системы Хиндли-Милнера (изо-рекурсивные типы и Y -комбинатор), определите тип для списка и реализуйте функцию, вычисляющую длину списка.
6. Выразите Y -комбинатор в Хаскеле и докажите с его помощью, что $\phi \rightarrow \neg\phi$, $\alpha \rightarrow \alpha \ \& \ \beta$ и $\neg\neg\phi \rightarrow \phi$.
7. Постройте в НМ вывод (дерево из аксиом и правил вывода) для **let** $t = \lambda f.\lambda x.f\ x$ **in** $t\ t$.
8. 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 *)
```

Поясните, почему:

- (a) определение $e2$ компилируется и работает;
- (b) определение e не компилируется.

Пояснение должно содержать необходимые фрагменты вывода типа в системе Хиндли-Милнера, или должно показывать, что нужного вывода типа не существует.

Домашнее задание №6: «Обобщённые типовые системы, исчисления конструкций»

1. 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 : * → *`, 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`.
2. Предложите выражение на языке C++ (возможно, использующее шаблоны), имеющее следующий род (тип):
- (a)  $\star \rightarrow \star \rightarrow \star$ ;  $\star \rightarrow \text{unsigned}$
- (b)  $\text{int} \rightarrow (\star \rightarrow \star)$
- (c)  $(\star \rightarrow \text{int}) \rightarrow \star$
- (d)  $\Pi x^{\star}. n^{\text{int}}. F(n, x)$ , где

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

3. Аналогично типу  $\Pi$ , мы можем ввести тип  $\Sigma$ , соответствующий квантору существования в смысле изоморфизма Карри-Ховарда.
- (a) Определите правила вывода для  $\Sigma$  в обобщённой типовой системе (воспользуйтесь правилами для экзистенциальных типов в системе  $F$ ).
- (b) Укажите способ выразить  $\Sigma$  через  $\Pi$  (также воспользуйтесь идеями для системы  $F$ ).

## Домашнее задание №7: «Теория множеств»

1. Пусть  $a$  и  $b$  — пустые множества. Покажите, что  $a = b$ .
2. Построение множеств. Покажите, что если  $a, b$  — множества, то следующие «наивные» конструкции тоже являются множествами:
  - (a)  $\bigcap a$  (пересечение всех подмножеств множества  $a$ );
  - (b)  $a \setminus b$  (разность множеств);
  - (c)  $a \uplus b$  (дизъюнктное объединение множеств);
  - (d)  $a \times b$  (декартово произведение множеств:  $\{\langle p, q \rangle \mid p \in a, q \in b\}$ ).
3. Покажите, что если  $\langle a, b \rangle = \langle c, d \rangle$ , то  $a = b$  и  $c = d$ .
4. Последовательностью элементов из  $S$  длиной  $k$  назовём множество  $A$  пар  $\langle o, s \rangle$ , что:
  - (a)  $o \in k, s \in S$ ;
  - (b) нет таких двух пар  $\langle o_1, s_1 \rangle$  и  $\langle o_2, s_2 \rangle$ , что  $o_1 = o_2$  и  $s_1 \neq s_2$ ;
  - (c)  $\langle 0, s \rangle \in A$  при некотором  $s$ ;
  - (d) Если  $\langle o, s_1 \rangle \in A$  и  $o' \in k$ , то  $\langle o', s_2 \rangle \in A$  при некотором  $s_2$ .

Будем записывать элементы последовательности как  $s_o$  и говорить что они занумерованы ординалами до  $k$ .

Пусть  $a_0, a_1, a_2, \dots$  — некоторая убывающая последовательность ординалов, занумерованных до некоторого ординала  $k$ : то есть,  $a_i$  — ординал, и, если  $i \in j$ , то  $a_j \in a_i$ . Покажите, что тогда ординал  $k$  — конечный.

5. Давайте покажем, что ординалы линейно упорядочены.
  - (a) Пусть  $a$  — ординал. Покажите, что  $\emptyset \in a$  или  $a = \emptyset$ .
  - (b) Пусть  $a$  — ординал. Покажите, что если  $x \in a$ , то  $x$  — ординал.
  - (c) Пусть  $a$  и  $b$  — ординалы, и  $a \in b$ . Покажите, что  $a' \in b$  или  $a = b$ .
  - (d) Пусть  $a$  и  $b$  — два ординала. Покажите, что  $a \in b$ , или  $b \in a$ , или  $a = b$ .
6. Покажите, что ординалы вполне упорядочены.
7. Пусть  $S$  — множество ординалов: если  $x \in S$ , то  $x$  — ординал. Определите операцию  $\sup_{ord} S$  — строящую минимальный ординал  $k$ , что  $x \in k$ , если  $x \in S$ .
8. Упростите по необходимости левую и правую часть равенств на ординалах и проверьте равенства:
  - (a)  $(\omega + 1)^2 = \omega^2 + \omega \cdot 2 + 1$ ;
  - (b)  $(\omega + \omega)^2 = \omega^2 \cdot 4$ ;
  - (c)  $(\omega^2)^\omega = \omega^\omega$ ;
  - (d)  $1^\omega = \omega$ ;
  - (e)  $2^\omega = \omega$ ;
  - (f)  $(\omega + 1) \cdot \omega = \omega^2 + \omega$ ;
  - (g)  $(\omega + 1)^\omega = \omega^\omega + 1$ .

## Домашнее задание №8: «Язык Аренд»

1. Равенство.
  - (a) Докажите, что `left = right`
  - (b) Докажите, что если  $a, b : \text{Nat}$  и  $a = b$ , то  $\neg(a \neq b)$
  - (c) Докажите, что если  $a, b : \text{Nat}$ , то  $a = b \vee a \neq b$
2. Определим отношение «меньше» на натуральных числах так:



```
\data NatLess (a b : Nat) \with
 | 0, suc m => natless_less
 | suc m, suc n => natless_next (NatLess m n)
```

Данный тип изоморфен утверждению  $a < b$ . Например, утверждение  $1 < 3$  доказывается так:

```
\func zerolesszero : NatLess 1 3 => natless_next (natless_less)
```

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

- (a)  $a < a + b + 1$ ; то есть, определите функцию
 

```
\func n_less_sum (a b : Nat) : NatLess a (a Nat.+ suc b)
```
  - (b) Если  $a < b$ , то  $a + c < b + c$
  - (c) Если  $a < b$  и  $c < d$ , то  $a \cdot c < b \cdot d$
  - (d)  $a < 2^a$
  - (e) Транзитивность: если  $a < b$  и  $b < c$ , то  $a < c$
  - (f) Определите аналогичное отношение «меньше или равно» и докажите его антисимметричность
  - (g) Если  $a \neq b$ , то  $a < b \vee b < a$
  - (h) Докажите, что  $a < b$  тогда и только тогда, когда  $a < b$  в смысле стандартных определений Аренда.
3. Аналогично предыдущему упражнению, определите тип данных **Even** («чётное натуральное число») и докажите следующие утверждения:
- (a) Если  $n$  — чётное, то  $\exists x. x + x = n$
  - (b) Если  $n$  таково, что  $\exists x. x + x = n$ , то  $n$  — чётное
  - (c) Если  $\exists x. x + x + 1 = n$ , то неверно, что  $n$  — чётное
4. Определите отношение (тип) «делится нацело» и докажите:
- (a) Рефлексивность, транзитивность и антисимметричность отношения
  - (b)  $a \cdot b$  делится нацело на  $a$  и на  $b$
  - (c) Если  $a$  делится нацело на  $b$ , то  $\exists t. a = b \cdot t$
  - (d) Если  $\exists t. a = b \cdot t$ , то  $a$  делится нацело на  $b$
  - (e) Если  $a < b$ , то невозможно, чтобы  $a$  делилось нацело на  $b$

## Домашнее задание №9: «Ещё доказательства»

1. Докажите недостающее свойство транзитивности `prove-transitive`:

```
\record Integer-carrier {
 | pos : Nat
 | neg : Nat
}

\func integer-eq-relation (x y : Integer-carrier) : \Type =>
 x.neg Nat.+ y.pos = x.pos Nat.+ y.neg

\func prove-transitive (x y z : Integer-carrier)
 (r1 : integer-eq-relation x y) (r2 : integer-eq-relation y z) :
 integer-eq-relation x z => {?} }
```

2. Как уже упоминалось,  $\Pi$ -типы соответствуют кванторам всеобщности, а  $\Sigma$ -типы — кванторам существования.

Как известно, доказательством квантора всеобщности является функция; докажем  $\forall x^{\text{Int}}. x = x$ :

```
\func proof1 : \Pi (x : Int) -> x = x => \lam x => idp
```

А доказательством квантора существования является пара из примера и доказательства его соответствия условию; докажем, что  $\forall x. \exists y. y = x^2$ :

```
\func proof2 : \Pi (x : Int) -> \Sigma (y : Int) (y = x * x) => \lam x => (x * x, idp)
```

Если мы доказываем утверждение, используя доказательство квантора существования, мы можем сослаться на его составные части (на пример и доказательство соответствия примера утверждению). Ниже  $p$  — это доказательство  $\exists x^{\text{Nat}}.x^2 = 77$ , тогда  $p.1$  — это пример значения, а  $p.2$  — доказательство утверждения  $(p.1)^2 = 77$ :

```
\func proof3 : (\Sigma (x : Nat) (x Nat.* x = 77)) -> (\Sigma (x : Nat) (77 = x Nat.* x)) =>
 \lam p => (p.1, Paths.inv p.2)
```

Обратим внимание, что пара, доказывающая существование — «зависимая»: второй элемент пары доказывает утверждение с уже подставленным первым аргументом:

```
\func five_eq_five : 5 = 5 => idp {Nat} {5}
\func proof4 : \Sigma (x : Nat) (x = 5) => (5, five_eq_five)
```

Проверьте, какие из следующих утверждений доказуемы (и тогда постройте соответствующее доказательство), а какие нет (и тогда опровергните их, тоже на Аренде):

- (a)  $(\exists x.\exists y.\phi) \rightarrow (\exists y.\exists x.\phi)$  и  $(\forall x.\forall y.\phi) \rightarrow (\forall y.\forall x.\phi)$
- (b)  $(\forall x.\exists y.\phi) \rightarrow (\exists y.\forall x.\phi)$  и наоборот,  $(\exists y.\forall x.\phi) \rightarrow (\forall x.\exists y.\phi)$
- (c)  $(\neg\forall x.\phi) \rightarrow (\exists x.\neg\phi)$  и в обратную сторону.

3. Вспомним одну из формулировок аксиомы выбора: если задано семейство непустых подмножеств  $\{B_a\}_{a \in A}$ , то существует функция  $f : A \rightarrow B$ , что  $\forall a.a \in A \rightarrow f(a) \in B_a$ .

Давайте представим семейство  $B_a$  отношением  $Q \subseteq A \times B$ , сопоставляющим элементу  $a \in A$  множество  $\{b \in B \mid \langle a, b \rangle \in Q\}$ . Тогда налагаемое на функцию  $f$  условие становится таким:

$$\forall a.a \in A \rightarrow \langle a, f(a) \rangle \in Q$$

что при трактовке множеств как типов приводит к следующей «наивной аксиоме выбора»:

```
\func naiveChoiceAxiom (A B : \Type) (Q : A -> B -> \Type)
 (allInhabited : \Pi (a : A) -> \Sigma (b : B) (Q a b))
 : \Sigma (f : A -> B) (\Pi (a : A) -> Q a (f a))
```

Докажите эту «аксиому».

4. Двоичное дерево

```
\data Tree
 | node Tree Tree
 | leaf

\func leafs (t : Tree) : Nat \elim t
 | leaf => 1
 | node l r => leafs l + leafs r
```

- (a) Докажите, что в полном двоичном дереве глубины  $n$  всего  $2^n$  листьев.
- (b) Докажите, что всего два дерева имеют в точности 3 листа; при этом давайте понимать множества как списки (из стандартной библиотеки). То есть, существует список из двух деревьев, что:
  - (a) любое дерево из списка имеет 3 листа; (б) любое дерево с 3 листьями принадлежит списку;
  - (в) все элементы списка различны.
- (c) Найдите, сколько деревьев имеет глубину  $\leq 3$  — и докажите соответствующее утверждение.
- (d) Найдите, сколько деревьев имеет глубину  $\leq n$  — и докажите соответствующее утверждение.

## Домашнее задание №10: «Умеренно-простые доказательства»

### 1. Равенства, неравенства, логика.

- (a) Докажите, что  $a \leq b$  (в смысле определения из стандартной библиотеки) тогда и только тогда, когда  $a = b$  или  $a < b$ .
- (b) Докажите, что если  $a, b \in \mathbb{N}$  и  $a \leq b \leq a + 3$ , то  $b = a \vee b = a + 1 \vee b = a + 2 \vee b = a + 3$ .
- (c) Докажите, что если  $a, b, k \in \mathbb{N}$ ,  $a \cdot b = k$  и  $a = k$ , то  $b = 1$ .
- (d) Докажите, что если  $a, b, k \in \mathbb{N}$  и  $a + b < k$ , то  $a < k$  и  $b < k$ .
- (e) Докажите, что если  $a, b, k \in \mathbb{N}_0$  и  $a \cdot b < k$ , то  $a \leq k$  и  $b \leq k$ .

### 2. Простые числа.

- (a) Определите понятие простого числа в Аренде.
- (b) Докажите, что любое число из  $\text{Nat}$  — либо 0, либо 1, либо простое, либо составное.
- (c) Докажите, что 2 — единственное простое чётное число.
- (d) Докажите, что для любого списка различных простых чисел найдётся простое число, не принадлежащее ему.
- (e) Докажите, что если сумма двух простых чисел — простая, то одно из чисел равно 2.
- (f) Докажите, что никакие два различных простых числа друг на друга не делятся.
- (g) Если  $p$  — простое, и  $p$  делит  $ab$ , то  $p$  делит  $a$  или  $b$ .

### 3. Докажите, что 6 и 28 — единственные совершенные числа от 1 до 100.

### 4. Рассмотрим следующее доказательство уникальности элементов списка:

```
\func not-member (A : \Type) (elem : A) (l : List A) : \Type \elim l
| nil => \Sigma
| :: hd tl => \Sigma (Not (hd = elem)) (not-member A elem tl)

\func unique-list (A : \Type) (l : List A) : \Type \elim l
| nil => \Sigma
| :: hd tl => \Sigma (not-member A hd tl) (unique-list A tl)
```

Докажем, что список  $[0, 1, 2]$  состоит из уникальных элементов:

```
\func r-unique => unique-list Nat (0 :: 1 :: 2 :: nil)
\func x : r-unique => ((contradiction, (contradiction, ())), ((contradiction, ()), (((), ())))
```

- (a) Напишите функцию, строящую список  $b$  натуральных чисел от 0 до  $n$  и доказательство `unique-list Nat b`.
- (b) Покажите, что если  $a_0 < a_1 < \dots < a_n$ , то список  $[a_0, a_1, \dots, a_n]$  уникальный.
- (c) Покажите, что если  $n \geq 2$  и  $a_k \neq a_{k+1}$  при  $0 \leq k < n$ , то необязательно, что список  $[a_0, a_1, \dots, a_n]$  — уникальный.

## Домашнее задание №11: «Линейная логика и уникальные типы»

### 1. Как известно, интуиционистские связки могут быть выражены в линейном исчислении. Покажите соответствующие интуиционистские аксиомы для следующих выражений интуиционистских связок через линейные:

- (a)  $A \rightarrow B := !A \multimap B$
- (b) Интуиционистская дизъюнкция:  $A + B := !A \oplus B$
- (c) Интуиционистская конъюнкция:  $A \times B := A \& B$
- (d) Альтернативная конъюнкция:  $A \times B := !A \otimes B$
- (e) Покажите, что альтернативная конъюнкция влечёт обычную, но не наоборот:  $\langle !A \otimes B \rangle \vdash A \& B$ , но  $\langle A \& B \rangle \not\vdash !A \otimes B$ .

2. Покажите следующие линейные тождества:

- (a)  $\vdash \phi \multimap \phi$
- (b)  $\vdash !\phi \multimap !!\phi$

3. Дистрибутивность

- (a) Выполняется ли дистрибутивность для  $(\&)$  и  $(\oplus)$ ?
- (b) Выполняется ли дистрибутивность для  $(\&)$  и  $(\otimes)$ ?

4. Уникальные типы. Рассмотрим интерфейс массива в линейных типах:

$$\begin{aligned} \text{new} &: !Val \multimap (Arr \multimap Arr \otimes X) \multimap X \\ \text{lookup} &: !Ix \multimap Arr \multimap Arr \otimes !Val \\ \text{update} &: !Ix \multimap !Val \multimap Arr \multimap Arr \end{aligned}$$

Определите соответствующий интерфейс в уникальных типах.

5. Типизируйте следующие конструкции в системе с уникальными типами:

- (a)  $\lambda a. \lambda b. b \ a$
- (b)  $\lambda a. a \ a$
- (c)  $\lambda a. \lambda b. a$
- (d)  $\lambda a. \lambda b. \lambda c. a \ c \ (b \ c)$

## Задания на баллы

Каждое задание будет оценено в 10 баллов, сдавать можно любые, в любой момент. Решением является программа на языке Аренд.

1. Рассмотрим алгоритм Эвклида, использующий деление: алгоритм, по натуральным числам  $a$  и  $b$  возвращающий их наибольший общий делитель. Для точности укажем шаг этого алгоритма:

$$(a_{n+1}, b_{n+1}) := \begin{cases} (b_n, a_n \bmod b_n) & \text{если } a_n \geq b_n > 0 \\ (b_n, a_n) & \text{если } a_n < b_n \end{cases}$$

Формализуйте этот алгоритм и покажите, что он:

- (a) всегда заканчивается;
- (b) итоговая пара чисел выглядит как  $(\text{НОД}(a, b), 0)$

2. Рассмотрим кольцо вычетов по модулю  $p$ :

TBD

Определите в Аренде утверждение «существует делитель нуля»  $(\exists a. \exists b. a \neq 0 \ \& \ b \neq 0 \ \& \ a \cdot b = 0)$  и покажите, что если  $p$  — простое, то делителей нуля в `QuotientRing p` нет.

3. Рассмотрим список `Data.List` и определим на нём операцию обмена:

$$\text{swap}(v, a, b)[n] = \begin{cases} a, & \text{если } n = b \\ b, & \text{если } n = a \\ v[n], & \text{иначе} \end{cases}$$

Покажите, что  $\forall v. \forall a. \forall b. \text{swap}(v, a, b) = \text{swap}(v, b, a)$