Дискретная математика
Коллоквиум 2
Определения

12 марта 2017 г.

1. Основные определения элементарной теории вероятностей: исходы, события, вероятность события.

Вероятностным пространством – называется конечное множество U, его элементы называются **возможными исходами**.
На вероятностном пространстве задана функция $Pr : U \to [0, 1]$, такая что $\sum\limits_{x \in U} Pr[x] = 1$. Функция Pr называется вероятностным распределением, а число $Pr[x]$ называется **вероятностью исхода** $x \in U$.
Событием называется произвольное подмножество $A \subseteq U$. **Вероятностью события** A называется число $Pr[A] = \sum\limits_{x \in A} Pr[x]$.

2. Случайные графы.

Случайный граф на n вершинах – элемент вероятностного пространства U, состоящего из всех графов на n вершинах, каждому из которых приписана некоторая вероятность. В нашем курсе такой граф не содержит петель и кратных ребер, поэтому всего графов на n вершинах $2^{\binom{n}{2}} \Rightarrow |\Omega| = 2^{\binom{n}{2}}$. Понятно, что случайным будет множество ребер графа.

Пример конструкции — каждому графу Ω присвоена одинаковая вероятность, т.е. все графы равно вероятны (т.е. для любого графа $G = (V, E) \in \Omega$, для каждой пары вершин $u, v \in V$, $Pr[(u, v) \in E] = \frac{1}{2}$).

Случайные графы используются для изучения каких-то свойств графов. Например, нестрогая постановка вопроса при работе со случайными графами: велика ли вероятность того, что граф обладает данным свойством? Более конкретный пример использования: доказательство того, что при достаточно большом числе вершин, случайный граф (в равновозможной модели) будет почти всегда связен. Формально: Ω_n — вероятностное пространство состоящее из графов на n вершинах, все графы равновозможны, событие A_n — случайный граф на n вершинах связен; доказать $\lim_{n \to \infty} Pr[A_n] = 1$.

3. Условная вероятность.

Условной вероятностью события A при условии B называется число

$$Pr[A|B] = \frac{Pr[A \cap B]}{Pr[B]}$$

Заметим, что условная вероятность имеет смысл, только если $Pr[B] > 0$. Иначе знаменатель обращается в ноль.

Определение условной вероятности можно переписать следующим образом:

$$Pr[A \cap B] = Pr[B] \cdot Pr[A|B]$$

Другими словами, чтобы найти вероятность пересечения событий A и B достаточно найти вероятность события B и условную вероятность события A при условии события B.

Формула Байеса. Если вероятность событий A B положительна, то

$$Pr[A|B] = Pr[A] \cdot \frac{Pr[B|A]}{Pr[B]}.$$

Доказательство.
Следует из

$$Pr[A \cap B] = Pr[B] \cdot Pr[A|B] = Pr[A] \cdot Pr[B|A].$$

4. Независимые события. Основные свойства независимых событий.

Событие A не зависит от события B, если

$$Pr[A] = Pr[A|B]$$

Чтобы не возникало никаких тонкостей с нулевыми вероятностями полезно условиться, что вероятности событий A и B ненулевые. Из определения условной вероятности мы сразу получаем эквивалентное определение независимости событий. Событие A не зависит от события B, если

$$Pr[A \cap B] = Pr[A] \cdot Pr[B]$$

Из этой формы определения видно замечательное свойство независимости событий: она симметрична. То есть, событие A не зависит от события B тогда и только тогда, когда событие B не зависит от события A.

Отметим также, что если события A и B независимы, и вероятность события $\overline{B}$ положительна, то события A и $\overline{B}$ независимы. $\overline{B} = U \setminus B$ – событие, дополнительное к B.

События A и B, для которых $0 < Pr[B] < 1$, независимы тогда и только тогда, когда $Pr[A|B] = Pr[A|\overline{B}]$.

5. Случайная величина и математическое ожидание.

Случайная величина – это числовая функция на вероятностном пространстве, то есть функция вида $f : U \to R$. То есть, по сути, случайная величина – это обычная числовая функция, но теперь на её аргументах задано вероятностное распределение. Таким образом, например, мы можем говорить о вероятности того, что случайная величина f равна какому-то конкретному значению a: это есть просто вероятность события

$\{u \in U \mid f(u) = a\}$. Случайные величины представляют собой числовые характеристики вероятностных экспериментов, и на самом деле, мы с ними уже неоднократно сталкивались, просто не говорили об этом. Например, если мы бросаем кубик, то исходом эксперимента является выпадение той или иной грани, а случайной величиной – число написанное на грани (каждой грани соответствует своё число – это функция).

Пусть вероятностное событие состоит из k исходов, случайная величина $f : U \to R$ принимает на них значения $a_1, \ldots, a_k$ соответственно и вероятности исходов равны $p_1, \ldots, p_k$ соответственно. В частности, $\sum\limits_{i=1}^{k} p_i = 1$. Предположим, что мы повторяем эксперимент по выбору случайного элемента из U n раз. Если n достаточно большое, то случайная величина f примет значение a_1 примерно $p_1 n$ раз, значение a_2 – примерно $p_2 n$ раз, и так далее, значение a_k – примерно $p_k n$ раз. Подсчитаем теперь примерное среднее арифметическое значений случайной величины f в этих экспериментах:

$$\frac{a_1 p_1 n + a_2 p_2 n + \ldots + a_k p_k n}{n} = \sum_{i=1}^{k} a_i p_i$$

Математическим ожиданием случайной величины f, принимающей значения $a_1, \ldots, a_k$ с вероятностями $p_1, \ldots, p_k$ соответственно, называется величина

$$E[f] = \sum_{i=1}^{k} a_i p_i$$

Математическое ожидание линейно.

Неравенство Маркова. Пусть f – случайная величина, принимающая только неотрицательные значения. Тогда для всякого $\alpha > 0$ верно

$$Pr[A|B] = Pr[A] \cdot \frac{Pr[B|A]}{Pr[B]}.$$

То есть вероятность того, что случайная величина f сильно больше своего математического ожидания невелика. Заметим, что лемма становится содержательной, только когда $\alpha > E[f]$.

*?. Определение бесконечного множества.

Определение: Множество A – конечно тогда и только тогда, когда $\exists n \in \mathbb{N}_0 : A \sim [n]$ ($[n] = \{1, 2, \ldots, n\}, [0] = \varnothing$).

Запомните, что: $n > k \Rightarrow [n] \nsim [k]$.

Определение: Множество бесконечно тогда и только тогда, когда оно не конечно.

6. Определение равномощных множеств. Основные свойства равномощности.

Определение: Равномощными множествами называются такие множества, между которыми установима биекция. Обозначение: $A \sim B$.

Очевидные свойства равномощных множеств: $\forall A$ – множеств.

- $A \sim A$.
- $A \sim B \Rightarrow B \sim A$.
- $A \sim B, B \sim C \Rightarrow A \sim C$.

7. Определение счетного множества. Примеры.

A – бесконечно. Значит A – не пусто. $\exists a_o \in A$. Пусто ли $A \setminus a_0$? Нет. Иначе A – содержит один элемент и конечно.

Тогда $\exists a_1 \in A \setminus a_0$. Множество и без этих двух элементов бесконечно. Ну и так далее.

Определение: Получившееся множество $A' = \{a_0, a_1, a_2, \ldots, a_n, \ldots\}$ назовем счётным (равномощным множеству натуральных чисел). Биекция в этом случае очевидна: $f : i \mapsto a_i$.

Утверждение: $\mathbb{N}$ – бесконечно.

Доказательство. Пусть это не так и $\exists f : [n] \to \mathbb{N}$ – инъекция. Тогда верно следующее: $\mathbb{N} \ni max\{f(0), f(1), \ldots, f(n)\} + 1 \notin f([n])$. А значит f – не биекция. А значит $\mathbb{N}$ не равномощно никакому $[n]$. **Q.E.D.**

Примеры счётных множеств:

- $\{0,1\}^*$ – множество двоичных слов.
- $\mathbb{N}$ – множество натуральных чисел (целые положительные и 0).
- $\mathbb{N} \times \mathbb{N}$ – множество пар натуральных чисел.
- $\mathbb{N}^*$ – множество конечных последовательностей натуральных чисел.

Утверждение: Множество бесконечно тогда и только тогда, когда оно равномощно какому-то своему подмножеству.

Доказательство. Докажем, что если множество бесконечно, то оно равномощно некоторому подмножеству.

Как мы уже выяснили, в любом бесконечном множестве есть счётное подмножество. Пусть $B = \{b_0, b_1, \ldots, b_n, \ldots\}$ – счётное подмножество бесконечного множества A.

Установим биекцию $f : A \setminus \{b_0\} \to A$.

$$f(x) = \begin{cases} b_{n-1}, x \in B \\ x, x \notin B \end{cases}$$

Получили то, что и требовалось.

В обратную сторону доказывается на семинарах, но примерно так: пусть $B \subset A, B \sim A$. Пусть A – конечно. Тогда $|B| < |A|$ **Q.E.D.**

8. Основные свойства счетных множеств.

1. A – счётное множество. Тогда $A' \subseteq A$ счётно или конечно.

   Доказательство. $A = \{a_0, a_1, \ldots, a_n, \ldots\}$. Вычеркнем все элементы, в A' не входящие. $A' = \{a_{j_0}, a_{j_1}, \ldots, a_{j_n}, \ldots\}$.

   Если последовательность $\{a_{j_n}\}$ конечна, то и A' конечно. Если она бесконечна, то A' очевидно счётно. **Q.E.D.**

2. Если A,B – счётные, то и $A \cup B$ счётно.

   Доказательство. $A = (a_0, a_1, \ldots, a_n, \ldots)$. $B = (b_1, b_2, \ldots, b_n, \ldots)$.

   $A \cup B = (a_0, b_0, a_1, b_1 \ldots, a_n, b_n, \ldots)$.

   Но может получиться так, что в новой последовательности некоторые элементы встречаются по два раза (они входят в оба множества). Вычеркнем каждый такой элемент по одному разу. И получим последовательность, задающую счётное множество. **Q.E.D.**

3. $\mathbb{Z}$ – счётно.

   Доказательство. $Z = \mathbb{N} \cup (-\mathbb{N})$ – объединение счётных множеств. Счётно по свойству 2 ($-A = \{-a \mid a \in A\}$). **Q.E.D.**

4. Если A – счётно, а B – конечно или счётно, то $A \cup B$ счётно.

   Доказательство. Доказывается аналогично свойству 2. **Q.E.D.**

5. Если A – счётно. И $B_1, B_2, \ldots, B_k$ – счётны или конечны, то $A \cup B_1 \cup \ldots \cup B_k$ – счётно.

   Доказательство. К доказательству свойства 4 нужно добавить доказательство по индукции. **Q.E.D.**

6. Счётное объединение конечных или счётных множеств конечно или счётно.

   $\{A_0, A_1, \ldots, A_n, \ldots\} = \mathfrak{F} \sim \mathbb{N}$. A_i – множество. $\mathfrak{F}$ называется семейством множеств. $A = \bigcup\limits_{i=0}^{\infty} A_i$.

   Утверждение: A – счётно.

   Доказательство.

   $$A_0 = (a_{00}, a_{01}, \ldots, a_{0n}, \ldots)$$
   $$A_1 = (a_{10}, a_{11}, \ldots, a_{1n}, \ldots)$$

   Некоторые из множеств могут быть конечны. Дополним их до счётных пустым символом $\lambda \notin A$.

Построим последовательность: $a_{00}, a_{01}, a_{10}, a_{02}, a_{11}, a_{20}, \ldots$. (то есть проходим последовательно все значения сумм индексов от 0 до ∞).

Теперь исключим из последовательности повторения и символы λ. Получим требуемую последовательность $(a'_0, a'_1, \ldots, a'_n, \ldots)$.

Теперь получим функцию $f : [n] \to A$ или $f : \mathbb{N} \to A$. f – биекция. В первом случае множество конечно, во втором счётно.

Можно было бы и не вводить λ, а исключать эти элементы сразу, но так проще (нет никаких условий). **Q.E.D.**

Примеры:

- Пусть $A_i = \{i\}$. Тогда $A = \mathbb{N}$ (счётно).
- Пусть $A_i = \{1\}$. Тогда $A = \{1\}$ (конечно).

7. Декартово произведение счётных множеств счётно. Напомним, что

$$A \times B = \{(a; b) \mid a \in A, b \in B\}$$

Доказательство. По определению декартово произведение есть множество всех упорядоченных пар вида $\langle a, b\rangle$, в которых $a \in A$ и $b \in B$. Разделим пары на группы, объединив пары с одинаковой первой компонентой (каждая группа имеетвид $\{a\} \times B$ для какого-то $a \in A$). Тогда каждая группа счётна, поскольку находится во взаимно однозначном соответствии с B (пара определяется своим вторым элементом), и групп столько же, сколько элементов в A, то есть счётное число. **Q.E.D.**

8. Если A – счётно, то A^k – счётно.

Доказательство. Очевидно по индукции из свойства 7. **Q.E.D.**

9. $\mathbb{Q}$ – счётно.

Доказательство. Рассмотрим множество $\mathbb{Q}_p$ несократимых дробей. Пусть функция $f : \mathbb{Q}_p \to \mathbb{Z} \times \mathbb{N}_+$ – инъекция (она переводит дробь в пару чисел числитель-знаменатель). Тогда она является биекцией на $f(\mathbb{Q}_p) \subset \mathbb{Z} \times \mathbb{N}_+$. Причём $f(\mathbb{Q}_p)$ тогда счётно по свойству 1 так как не является конечным, а $\mathbb{Z} \times \mathbb{N}_+$ счётно по свойству 7. **Q.E.D.**

10. Пусть A^* – конечные последовательности конечного (непустого) или счётного алфафита A.

Утверждение: A^* – счётно

Доказательство. $A^* = \bigcup\limits_{n=0}^{\infty} A^n$. При этом A^n – слова длины n. A^n – счётно по свойству 8. И тогда само A^* счётно по свойству 6. **Q.E.D.**

11. Определение: $\alpha \in \mathbb{R}$ – алгебраическое число тогда и только тогда, когда α – корень некоторого многочлена с целыми коэфициентами.

Утверждение: Множество алгебраических чисел счётно.

Доказательство. Приведём только план доказательства:

(a) Докажем, что многочленов степени n ($n \in \mathbb{N}$) с целыми коэфициентами счётно.

(b) Для каждого из этих многочленов есть не более n корней – алгебраических чисел.

(c) Удаляем повторяющиеся корни.

(d) Получим все алгебраические числа, которых, очевидно, счётно.

Q.E.D.

9. Определение множества мощности континуум. Примеры.

Определим действительные числа следующим образом: сопоставим каждому $x \in \mathbb{R}$ двоичное число: $\pm\underbrace{10110\ldots1011}_{\text{целая часть}}.\overbrace{110001\ldots00110}^{\text{дробная часть}}$. Считаем известным, что ряд из каких-то степеней двоек сходится, причём запрещаем в числах данного вида "хвосты из единиц".

Определение: Будем говорить, что множество X имеет мощность континуум, если $X \sim \mathbb{R}$.

Примеры:

- $\Phi(\mathbb{N})$ – множество всех подмножеств $\mathbb{N}$.
- $2^{\mathbb{N}}$ – множество последовательностей натуральных чисел.
- $\mathbb{R}$ – само множество действительных чисел.

10. Основные свойства континуальных множеств.

1. Любое континуальное множество имеет счётное подмножество.
2. Мощность объединения не более чем континуального количества множеств, каждое из которых не более чем континуально, не превосходит континуума.
3. При разбиении континуального множества на конечное или счётное число частей хотя бы одна из частей будет иметь мощность континуум.

11. Булевы функции. Задание булевых функций таблицами истинности.

Булева функция от n аргументов - отображение из B^n в B, где B - {0,1}. Количество всех n-арных булевых функций равно 2^{2^n}. Булеву функцию можно задать таблицей истинности, поскольку она задается конечным набором значений.

12. Определение полного базиса. Примеры полных и неполных базисов.

Полный базис: Базис B – *полный*, если любую булеву функцию можно вычислить схемой в базисе B.

Примеры полных базисов:

1. Стандартный Базис
2. Дизъюнкция, инверсия
3. Конъюнкция, инверсия
4. Импликация, инверсия
5. Базис Жегалкина $\{1, \oplus, \vee\}$
6. Стрелка Пирса
   (a) $0 \downarrow 0 = 1, 0 \downarrow 1 = 0, 1 \downarrow 0 = 0, 1 \downarrow 1 = 0$
   (b) $X \downarrow X \equiv \neg X$ — отрицание
   (c) $(X \downarrow X) \downarrow (Y \downarrow Y) \equiv X \wedge Y$ — конъюнкция
   (d) $(X \downarrow Y) \downarrow (X \downarrow Y) \equiv X \vee Y$ — дизъюнкция
   (e) $((X \downarrow X) \downarrow Y) \downarrow ((X \downarrow X) \downarrow Y) = X \to Y$ — импликация
7. Штрих Шеффера
   (a) $0 \| 0 = 1, 0 | 1 = 1, 1 | 0 = 1, 1 | 1 = 0$
   (b) $X \,|\, X = \neg X$ — отрицание
   (c) $(X \,|\, X) \,|\, (Y \,|\, Y) = X \vee Y$ — дизъюнкция
   (d) $(X \,|\, Y) \,|\, (X \,|\, Y) = (X \wedge Y)$ — конъюнкция
   (e) $X \,|\, \neg X$ — константа 1

Примеры неполных базисов:

1. Монотонный базис
2. $\{\wedge, \oplus\}$
3. $\{1, \wedge\}$
4. $\{1, \oplus\}$
5. Конъюнкция, дизъюнкция и разность
6. Большинство одноэлементных базисов

13. Разложение Рида.

Разложением Шеннона функции $f : \{0,1\}^n \to \{0,1\}$ по переменной x_i называется представление фукнции f в виде:

$$f(x_n, \dots, x_i, \dots, x_1) = \overline{x_i} \cdot f(x_n, \dots, 0, \dots x_1) \vee x_i \cdot f(x_n, \dots, 1, \dots, x_1)$$

Разложением Рида называется следующее представление функции:

$$f(x_n, \dots, x_i, \dots, x_1) = g_0 \oplus (g_0 \oplus g_1) \cdot x_i),$$

$$g_0 = f(x_n, \dots, 0, \dots x_1)$$
$$g_1 = f(x_n, \dots, 1, \dots, x_1)$$

14. ДНФ, СДНФ и СКНФ

Необходимое определение. Простой конъюнкцией называется конъюнкция одной или нескольких переменных или их отрицаний, причём каждая переменная встречается не более одного раза.

Простая конъюнкция

- **полная**, если в неё каждая переменная (или её отрицание) входит ровно 1 раз;
- **монотонная**, если она не содержит отрицаний переменных.

Дизъюнктивная нормальная форма, она же ДНФ – нормальная форма, в которой булева функция имеет вид дизъюнкции нескольких *простых* конъюнктов.

Пример ДНФ: $f(x,y,z) = (x \wedge y) \vee (y \wedge \neg z)$.

Совершенная дизъюнктивная нормальная форма, СДНФ – ДНФ, удовлетворяющая условиям:

- в ней нет одинаковых простых конъюнкций,
- каждая простая конъюнкция полная.

Пример СДНФ: $f(x,y,z) = (x \wedge \neg y \wedge z) \vee (x \wedge y \wedge \neg z)$.

Конъюктивная нормальная форма и Совершенная конъюктивная нормальная форма определяются аналогично:

- Конъюктивная нормальная форма – конъюнкция простых дизъюнктов
- СКНФ – КНФ, в которой каждый дизъюнкт — полный.

15. Полином Жегалкина

Полином Жегалкина — полином с коэффициентами вида 0 и 1, где в качестве произведения берётся конъюнкция, а в качестве сложения исключающее или. Каждая булева функция единственным образом представляется в виде полинома Жегалкина.

16. Определение схемы в некотором функциональном базисе. Представление схем графами.

Схема — это функция, заданная последовательностью присваиваний.

Иными словами, булевой схемой от переменных $x_1, \ldots, x_n$ в некотором функциональном базисе мы будем называть последовательность булевых функций $g_1, \ldots, g_s$, в которой всякая g_i или равна одной из переменных, или получается из предыдущих применением одной из функций из базиса.

Также в профессиональной среде схемы называют SLP (*straight line programmes*).

Рассмотрим такую функцию f, определенную для булевых значений (*булеву функцию*): $f : \{0,1\}^n \to \{0,1\}$.

Базисом B булевой функции будем называть некий набор $B : \{f_1, f_2, \ldots, f_n\}$, где $f_1 \ldots f_n$ - булевы функции.

Булева схема в базисе B — последовательность функций $x_1, x_2, x_3 ... x_n, x_{n+1} := S_1, x_L := S_{L-n}$, которая вычисляет $x_L(x_1, \ldots, x_n)$.

$$S_j = g(S_{i_1}, ..., s_{i_r}), g \in B, i_< j$$

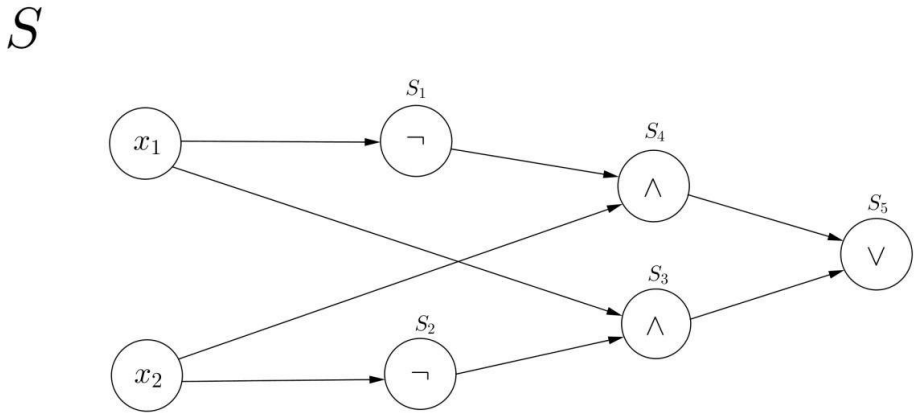
Стандартный базис есть базис, состоящий из операций отрицания, конъюнкции и дизъюнкции: $\{\neg, \vee, \wedge\}$

Схемы можно представлять в виде графов:

Пример 1

Зададим булеву схему с помощью стандартного базиса.

$x_1, \ldots, x_n, s_1 := \neg x_1, s_2 = \neg x_2, s_3 := x_1 \wedge s_2, s_4 := x_2 \wedge s_1; s_5 := s_3 \vee s_4$



Если $x_2 = 0$, то $s_5 = x_1$

Если $x_2 = 1$, то $s_5 = \neg x_1$

Результатом выполнения булевой схемы является сложение по модулю 2 (1, если значения x_1 и x_2 разные) - $\oplus$.

Пример 2

Составим схему, которая является сложением по модулю 2 n переменных. Приведём индуктивное доказательство её существования:

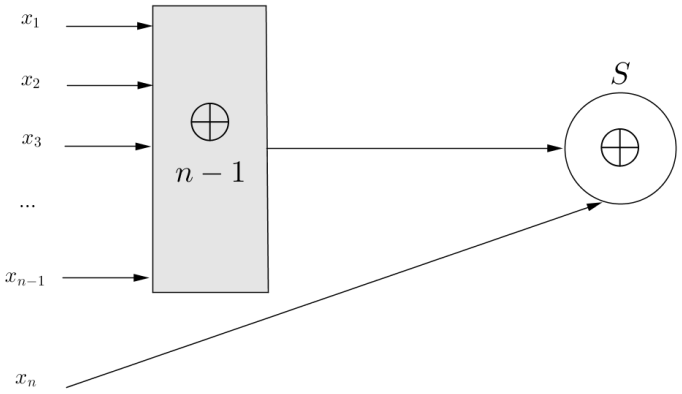
1. База индукции — $n = 2$. Сложение 2 переменных по модулю 2 возможно по схеме, описанной выше

2. Предположим существование такой схемы для $n - 1$ переменных

$$x_1 \oplus x_2 \oplus \ldots \oplus x_{n-1}$$

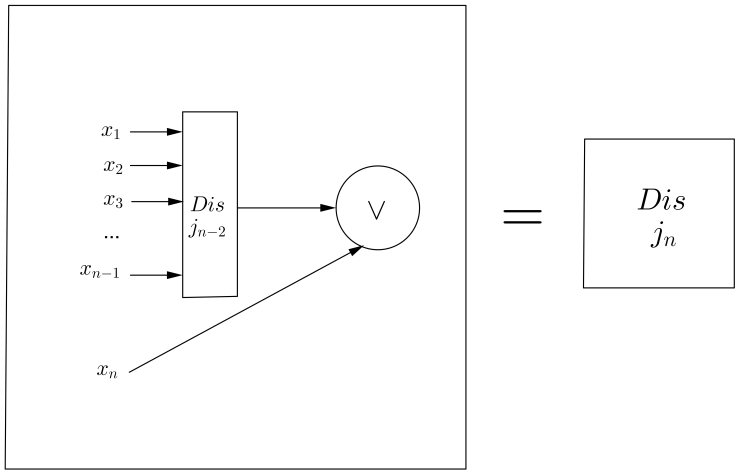
3. Рассмотрим сложение по модулю 2 n переменных. Представим его как сложение по модулю 2 x_n с результатом предыдущего шага. Существование второго слагаемого объясняется предположением индукции. Сложение с x_n можно выполнить по схеме выше.

$$x_1 \oplus x_2 \oplus \ldots \oplus x_{n-1} \oplus x_n = (x_1 \oplus x_2 \oplus \ldots \oplus x_{n-1}) \oplus x_n$$



Пример 3

Дизъюнкция n переменных — аналогично, по индукции. Такие рассуждения можно построить и для конъюнкции.



17. Определение схемной сложности функции.

Размер булевой схемы — это количество присваиваний в схеме $g_1, \ldots, g_L$ для вычисления функции $f : \{0,1\}^n \to \{0,1\}$.

Сложность функции f в базисе B — это минимальный размер булевой схемы, вычисляющей функцию f в базисе B. Если базис не указывают – имеют в виду стандартный базис $\{\neg, \vee, \wedge\}$. *Обозначение:* $C(f)$.

Определения различных оценок сложностей аналогичны тем, что мы используем на алгоритмах и в математическом анализе:

$$o(f(n)) = \{g(n) \mid \forall\ c > 0\ \exists\ n_0 > 0\ : \forall n \geq n_0 \Rightarrow 0 \leq g(n) < c \cdot f(n)\}$$

$$O(f(n)) = \{g(n) \mid \exists\ c > 0,\ \exists\ n_0 > 0\ : \forall n \geq n_0 \Rightarrow 0 \leq g(n) \leq c \cdot f(n)\}$$

$$\omega(f(n)) = \{g(n) \mid \forall\ c > 0\ \exists\ n_0 > 0\ : \forall n \geq n_0 \Rightarrow 0 \leq c \cdot f(n) < g(n)\}$$

$$\Omega(f(n)) = \{g(n) \mid \exists\ c > 0,\ \exists\ n_0 > 0\ : \forall n \geq n_0 \Rightarrow 0 \leq c \cdot f(n) \leq g(n)\}$$

$$\Theta(f(n)) = \{g(n) \mid \exists\ c_1 > 0, \exists\ c_2 > 0, \exists\ n_0 \in \mathbb{N} :\ \forall\ n \geq n_0 \Rightarrow c_1 \cdot f(n) \leq g(n) \leq c_2 \cdot f(n)\}$$

18. Основные свойства вычислимых функций.

Определение: Функция называется вычислимой, если для неё существует некоторый алгоритм вычисления.

Первое свойство алгоритмов. *Композиция вычислимых функций вычислима.*

Доказательство. Пусть существуют вычислимые функции f, переводящая множество входов X в множество выходов Y и g, переводящая Y в Z.

$$\begin{cases} f : X \to Y \\ g : Y \to Z \end{cases}$$

Построить $g \circ f : X \to Z$ достаточно просто. На первом шаге нужно применить функцию f. Далее берём множество выходов f и подаём на вход в g. На выходе получим некоторое множество выходов Z. Вычислимая композиция построена.

Algorithm 1 Алгоритм получения композиции функций

```
1: function Composition(x)
2:     t ← f(x)
3:     return g(t)
4: end function
```

Q.E.D.

Итак, есть вычислимая биекция $\pi : \mathbb{N}^* \to \mathbb{N}$.

Построим композицию $\mathbb{N}^* \to \mathbb{N} \to \{0, 1\}$.

$$\begin{cases} \pi^{-1} : B \to A \\ f : A \to A \\ \pi : A \to B \end{cases}$$

Построить это можно применением композиции $\pi \circ f \circ \pi^{-1}$. Получим алгоритм из B в B. Поэтому нам, по сути, **без разницы какие множества на входе и выходе** так как можно получить легко из одного другое.

Заметим, что если некоторая биекция π вычислима, то обратная ей π^{-1} также будет вычислима. Способ вычисления схож с алгоритмом для диофантовых уравнений.

Рассматриваем все возможные значения входов и вычисляем $\pi^{-1}(x)$.

Algorithm 2 Алгоритм построения обратной функции для биекции

```
1: function RevBiection(x)
2:     for n := 0 ... ∞ do
3:         if π(n) = x then
4:             return n
5:         end if
6:     end for
7: end function
```

Стоит отметить, что данный алгоритм всегда завершается так как (по определению) биекция сюръективна и найдется номер n для которого $\pi(n) = x$.

19. Определение разрешимого множества.

Вспомним что такое характеристическая функция множества S:

$$\chi_S(x) = \begin{cases} 1, \, x \in S \\ 0, \, x \notin S \end{cases}$$

Определение: Множество называется разрешимым, если его характеристическая функция вычислима.

20. Определение перечислимого множества.

Определение: Алгоритм перечисления – такой алгоритм, у которого нет входа, он работает и может выводить некоторые числа, причём все напечатанные числа составляют счетное множество.

Определение 1: Множество S называется перечислимым, если есть алгоритм перечисления всех его элементов.

Определение 2: Множество S – перечислимо, если существует такая вычислимая функция:

$$f : \mathbb{N} \to \mathbb{N} \begin{cases} f(\mathbb{N}) = S \\ \text{Область определения } f \text{ равна либо } \mathbb{N}, \text{либо } [n]. \end{cases}$$

Теорема. *Определения* 1 *и* 2 *эквивалентны.*

Доказательство.

$\Longrightarrow$ Пусть A – алгоритм перечисления множества S. Тогда возьмём следующий алгоритм B: принимает на вход число n, запускает алгоритм A и считает, сколько чисел напечатано. Как только вывели $n+1$ слово – алгоритм печатает результат.

Покажем, что соблюдаются свойства вычислимой функции:

1. $B(n) = S$, так как $\forall\ n\ \exists\ B(n) \Rightarrow \begin{cases} \forall\ x \in S\ \exists\ B(x) \\ \forall\ x \notin S \text{ никогда не выведет } B(x) \end{cases}$
2. Пусть S – бесконечное множество, тогда функция, задаваемая B, определена везде, значит $dom\ B = \mathbb{N}$. Если B работает на n числах, то алгоритм переберёт их и остановится.

$\Longleftarrow$ Возьмём следующий алгоритм перечисления B для множества S:

Algorithm 3 Алгоритм перечисления разрешимого множества

```
1: function PrintSet(S)
2:     for i := 0 ... ∞ do
3:         if f(i) = 1 then
4:             print i
5:         end if
6:     end for
7: end function
```

Если функция определена для некоторых n чисел, то ровно их он и напечатает. Если $dom\ f = \mathbb{N}$, то алгоритм никогда не остановится, то есть напечатает всю область определения f. Значит, существует алгоритм, перечисляющий S.

Q.E.D.

21. Свойства перечислимых множеств.

Утверждение: Если множество S разрешимо, то оно перечислимо.

Доказательство. Алгоритм перечисления множества S использует алгоритм раз- решения множества S. Он перебирает все числа, начиная с 0; для каждого числа n вычисляет индикаторную функцию $\chi S(n)$ и печатает число n, если полученное значение равно 1.

Algorithm 4 Алгоритм перечисления множества S

```
1: function Print(S(n))
2:     for n := 0 ... ∞ do
3:         if χ_S(n) = 1 then
4:             print n
5:         end if
6:     end for
7: end function
```

Корректность такого алгоритма ясна из определений. **Q.E.D.**

Пусть S – перечислимое непустое множество. Тогда для S выполнены следующие свойства:

1. S – область определения вычислимой функции.
2. S – область значений вычислимой функции.
3. S – область значений всюду определённой вычислимой функции.

Пусть A и B – перечислимые множества. Тогда:

1. $A \cup B$ перечислимо.
2. $A \cap B$ перечислимо.

Утверждение: Существует перечислимое неразрешимое множество.

Доказательство. Рассмотрим вычислимую функцию $f(x)$, не имеющую всюду определённого вычислимого продолжения. Её область определения F будет искомым множеством. В самом деле, F перечислимо (по одному из определений перечислимости). Если бы F было разрешимо, то функция

$$g(x) = \begin{cases} f(x), \text{если } x \in F \\ 0, \text{если } x \notin F. \end{cases}$$

была бы вычислимым всюду определённым продолжением функции f (при вычислении $g(x)$ мы сначала проверяем, лежит ли x в F, если лежит, то вычисляем $f(x)$). **Q.E.D.**

22. Определение универсальной вычислимой функции.

Определение: Универсальная вычислимая функция:

$$U : \mathbb{N} \times \mathbb{N} \to \mathbb{N} \mid \forall f\text{-вычислимая } \exists\, p : f(x) = U(p,\, x).$$

Также полезно знать про отладочную функцию:

Определение: Отладочная функция:

$$F : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \{0,\, 1\} \text{ – всюду определённая, причём:}$$

$$\begin{cases} \text{При фиксированных } p \text{ и } x \text{ монотонна по } t \\ F(p,\, x,\, t) = 0 \Leftrightarrow U(p,\, x) \text{ не определена} \\ F(p,\, x,\, t) = 1 \Leftrightarrow \text{ программа } p \text{ на входе } x \text{ заканчивает работу за } t \text{ шагов.} \end{cases}$$

23. Определение отладочной функции.

Пусть $U(p, x)$ – универсальная вычислимая функция. Для данной у.в.ф. существует функция $F : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \{0, 1\}$, называемая отладочной, для которой выполняются следущие свойства:

- $F(p, x, t)$ не убывает по t (Т.е. $\forall x, p \in N : t_0 < t_1 \Leftrightarrow F(p, x, t_0) \leq F(p, x, t_1)$)
- $U(p, x)$ не определена $\Leftrightarrow \forall t \in N : F(p, x) = 0$

Неформально говоря, значение функции $F(p, x, t)$ равно 0 тогда и только тогда, когда программа p на входе x не закончила работу за количество шагов t. В противном случае значение функции $F(p, x, t)$ равно 1.

24. Определение главной универсальной вычислимой функции.

Определение: Главная универсальная функция (гёделева) – такая универсальная функция, что для любой вычислимой функции $V(n,\ x)$ существует всюду определённая вычислимая функция $s(n)$, что:

$$\forall\ n,\ x \Rightarrow V(n,\ x) = U(s(n),\ x).$$

Неформально это значит, что главная универсальная функция позволяет транслировать в себя любую другую унивесальная функция. Ну, вот например, есть язык C++, его можно назвать главной универсальной функцией так как любую программу на другом универсальном языке можно переписать на C++ автоматически (при помощи *транслятора*).

25. Формулировка теоремы Успенского–Райса.

Пусть есть некоторое свойство, которое мы хотим проверить для некоторой функции.

Формально: Пусть $\{f : \mathbb{N} \to \mathbb{N}\}$ – множество вычислимых функций. Разделим его на два непересеающихся подмножества A и $\overline{A}$.

$$\{f \mid f : \mathbb{N} \to \mathbb{N}\} = A \cup \overline{A}$$

A – множество тех функций, для которых выполняется некое свойство, $\overline{A}$ – множество тех функций, для которых это свойство не выполняется.

Возьмём некоторую универсальню функцию $U(p,\ x)$.

Обозначим за P_A множество всех p таких, что $U(p,\ x) \in A$.

$$P_A = \{p \mid U(p, x) \in A\}$$

Тогда вопрос можно поставить так: разрешимо ли множество программ, удовлетворяющих нашему свойству? На этот вопрос и отвечает теорема Успенского-Райса:

Теорема Успенского-Райса. *Если A – нетривиально ($A \neq \oslash$, $\overline{A} \neq \oslash$), а $U(q,\, x)$ – главная универсальная функция, то множество P_A неразрешимо.*

Введём для удобства ещё функции $\varepsilon \in \overline{A}$ (нигде не определённая) и $\xi \in A$ (какая-то функция, удовлетворяющая условию). Сделать это можно по аксиоме выбора.

Если A – это множество нигде не определённых функций, то поменяем их местами так как P_A разрешимо тогда и только тогда, когда его дополнение разрешимо.

26. Формулировка теоремы о неподвижной точке.

Теорема о неподвижной точке. *Пусть U – главная универсальная функция, $h(n)$ – любая всюду определённая вычислимая функция. Тогда:*

$$\exists\ q\ :\ U(q,\, x) = U(h(q),\, x).$$

Честно сказать, не все учёные понимают эту теорему, однако её можно объяснить неформально так: для любой программы на любом универсальном языке существует ещё одна программа, которая делает то же самое (то есть программы совпадают).

27.1. Определение машины Тьюринга (с одной лентой).

Мы рассмотрим классическую модель вычислений, на которой будут основано точное определение вычислимых функций, – машины Тьюринга (МТ).

МТ состоит из

- бесконечной в две стороны ленты, в ячейках которой могут быть записаны символы алфавита A (некоторого конечного множества)
- головки, которая может двигаться вдоль ленты, обозревая в каждый данный момент времени одну из ячеек
- оперативной памяти, которая имеет конечный размер (другими словами, состояние оперативной памяти – это элемент некоторого конечного множества, которое называется множеством состояний МТ Q)
- таблицы переходов (или программы), которая задаёт функцию $\delta : A \times Q \to A \times Q \times \{-1,\, 0,\, +1\}$

Поскольку таблица переходов – это функция на конечном множестве, её возможно задать таблицей. Каждая строка таблицы – это пять значений a, q, $a^{'}$, $q^{'}$, d, (другие способы записи: $\delta(a,\, q) = (a^{'},\, q^{'},\, d)$ или $\delta : (a,\, q) \mapsto (a^{'},\, q^{'},\, d)$), которые описывают следующий порядок действий МТ: если головка МТ находится над ячейкой, содержащей символ a, а состояние МТ равно q, то на очередном такте работы МТ записывает в текущую ячейку символ $a^{'}$, изменяет состояние на $q^{'}$ и сдвигает головку на d ячеек (отрицательное значение отвечает сдвигу влево, положительное – сдвигу вправо, 0 – не сдвигается).

Работа МТ состоит из последовательного выполнения тактов в соответствии с таблицей переходов. Может так случиться, что для текущей пары значений $(a,\, q)$ функция

переходов не определена. В этом случае работа машины заканчивается (машина останавливается). Обычно среди состояний МТ выделяют множество Q_f финальных состояний – таких состояний q_f, что таблица переходов не определена для всех пар (a, q_f). Попав в финальное состояние, машина обязательно остановится, отсюда и название.

Лента МТ бесконечна и это не соответствует нашей интуиции об алгоритмах: алгоритм на каждом шаге работы оперирует лишь данными конечного размера. Чтобы учесть это обстоятельство, мы предполагаем, что в алфавите машины есть специальный символ Λ (пробел или пустой символ) и все ячейки ленты за исключением конечного числа содержат пустые символы. Это свойство ленты сохраняется при работе МТ, поскольку за такт работы меняется содержимое не более одной ячейки ленты.

Состояние такой машины уже описывается конечными данными. Мы будем использовать конфигурации. Конфигурация – это слово в алфавите $A \cup Q$, в котором первый и последний символы непустые, и ровно один символ принадлежит множеству состояний. Договоримся считать, что символ состояния записывается слева от символа в той ячейке, над которой находится головка МТ. На ленте слева и справа от символов конфигурации стоят только пустые символы.

У МТ есть 2 основных конфигурации: начальная q_0, из которой МТ начинает работу, и финальная q_f.

Конфигурации МТ преобразуются такт за тактом, порождая последовательность конфигураций $c_0 = q_0 u$, c_1, $c_2, \ldots, c_t, \ \ldots$

Эта последовательность бесконечна, если машина не останавливается, и конечна в противном случае. Результатом работы является та часть финальной конфигурации, которая расположена между символом состояния и ближайшим к нему пустым символом справа.

27.2. Определение машины Тьюринга (с несколькими лентами).

Определение: Машина, у которой не одна лента, а несколько (фиксированное число для конкретной машины), называется *многоленточной*.

На каждой ленте есть своя головка. За такт работы головки могут перемещаться по всем лентам. Действие на такте работы зависит как от состояния машины, так и от всего набора символов, которые видят головки машины на всех лентах.

Чтобы задать машину с h лентами, нужно указать:

- алфавит A, в котором выделен пустой символ Λ
- множество состояний Q, в котором выделено начальное состояние q_0
- таблицу переходов, которая теперь является функцией вида $\delta : A^h \times Q \to A^h \times Q \times \{-1, 0, +1\}^h$ (первый аргумент – символы, которые машина видит на ленте, последний – команды движения для головок на каждой ленте).
- выделить среди лент ленту входа и ленту результата (возможно, что это одна и та же лента)

Таблица переходов по прежнему является функцией на конечном множестве, поэтому её возможно задать таблицей. Работа МТ состоит из последовательного выполнения тактов в соответствии с таблицей переходов. Может так случиться, что для текущего набора

значений $(a_1, a_2, \ldots, a_h, q)$ функция переходов не определена. В этом случае работа машины заканчивается (машина останавливается). Как и раньше, можно ввести множество финальных состояний Q_f, т.е. тех состояний q_f, для которых таблица переходов не определена для всех значений $(a_1, a_2, \ldots, a_h, q_f)$. В финальном состоянии машина обязательно останавливается.

Мы предполагаем, что h-МТ начинает работу в состоянии q_0, а все ленты кроме ленты входа содержат только пустые символы. На ленте входа записано входное слово, и головка находится над первой слева ячейкой, содержащей символы этого слова. Поскольку за такт работы меняется содержимое не более одной ячейки ленты, в процессе работы машины на каждой ленте будет записано лишь конечное количество непустых символов.

Конфигурация многоленточной машины может быть задана набором конфигураций на каждой ленте $(u_1 q_{v_1}, u_2 q_{v_2}, \ldots, u_h q_{v_h})$.

Символ состояния один и тот же, так как по нашим определениям состояние есть у машины, а не у головки.

Далее нам будет удобен другой способ представления конфигурации машины. Выровняем ленты и будем рассматривать *окно*, в которое заведомо помещаются все непустые символы на каждой ленте. В таком случае конфигурация однозначно определяется матрицей размера $h \times N$, в которой записаны символы на лентах. Нужно ещё указать положения головок на лентах (они-то не обязательно выровнены – машина способна перемещать головки независимо). По этой причине будем помещать в матрицу не символы алфавита A, а пары $(a, \hat{q})$, где $\hat{q}$ указывает, расположена ли на данной ленте головка над данной ячейкой. Если да, то $\hat{q} \in Q$ – текущее состояние машины. Если нет, то $\hat{q}$ – какой-то символ не из Q, который указывает, что над над данной ячейкой на данной ленте нет головки. Будем для единообразия использовать в качестве такого символа Λ. Такую матрицу в дальнейшем называем матрицей конфигурации. Вот пример матрицы начальной конфигурации для двухленточной машины:

(Λ, q_0)	(Λ, Λ)	(Λ, Λ)	(Λ, Λ)	(Λ, Λ)
(a, q_0)	(b, Λ)	(a, Λ)	(a, Λ)	(b, Λ)

Как и для одноленточной машины, работа h-МТ порождает последовательность конфигураций $c_0 = q_0 u$, c_1, $c_2, \ldots, c_t, \ldots$

Эта последовательность бесконечна, если машина не останавливается, и конечна в противном случае. Результатом работы является та часть финальной конфигурации на ленте результата, которая расположена между положением головки и ближайшим к нему пустым символом справа. Например, если у двухленточной МТ лента результата – нижняя, то результатом работы МТ, остановившейся в конфигурации, заданной окном

(Λ, Λ)	(Λ, Λ)	(a, q_f)	(a, Λ)	(Λ, Λ)
(a, Λ)	(b, q_f)	(a, Λ)	(Λ, Λ)	(b, Λ)

будет ba.

28. Определение функции, вычислимой на машине Тьюринга.

Определение: МТ M (h-МТ, если многоленточная) вычисляет функцию $f : B^* \to B^*$ (где B – подмножество алфавита машины, не содержащее пустого символа), если для каждого w из области определения функции f результат работы M равен $f(w)$, а для каждого w не из области определения f машина M не останавливается на входе w.

Функция f называется *вычислимой машинами Тьюринга*, если есть такая МТ, которая вычисляет f.