Лекция 1. Отношения на множествах

Определение 1. *Пусть A, B — множества. Декартово произведение $A \times B$ — это множество упорядоченных пар*

$$A \times B = \{(a, b) : a \in A, b \in B\}.$$

Определение 2. *Любое подмножество декартова квадрата $A \times A$ называется отношением или предикатом (точнее, бинарным отношением или двуместным предикатом) на множестве A.*

Обозначать отношения будем «красивыми» заглавными латинскими буквами:

$$(x, y) \in \mathcal{R} \quad \text{или} \quad x\mathcal{R}y.$$

Если $x\mathcal{R}y$, то говорят, что x и y связаны отношением $\mathcal{R}$.

Пример 1. *Нуль–отношение $\mathcal{R} = \varnothing$ не связывает никакие элементы. Универсальное отношение — это $\mathcal{R} = A^2$. Отношение равенства или тождественное бинарное отношение — это отношение $\mathcal{I} = \{(a, a) : a \in A\}$.*

Отношения можно обращать.

Определение 3. *Если $\mathcal{R}$ — отношение на множестве A, то обратным отношением называют*

$$\mathcal{R}^{-1} := \{(a, b) : (b, a) \in \mathcal{R}\}.$$

Некоторые отношения называют функциями или отображениями.

Определение 4. *Отношение $\mathcal{F}$ называется отображением из A в A, если*

$$\forall x \begin{cases} (x, y) \in \mathcal{F}, \\ (x, z) \in \mathcal{F} \end{cases} \implies y = z.$$

Если отношение $\mathcal{F}$ — отображение, то запись $(x, y) \in \mathcal{F}$ меняют на $y = \mathcal{F}(x)$. При этом говорят, что элемент x — прообраз элемента y при отображении $\mathcal{F}$, а y — образ элемента x при отображении $\mathcal{F}$. Образом отображения $\mathcal{F}$ называют

$$\operatorname{Rn} \mathcal{F} := \{y \in A : \exists x \in A,\, y = \mathcal{F}(x)\}$$

У некоторых $x \in A$ может не быть никакого образа (нет ни одной пары $(x, y) \in \mathcal{F}$). Тогда говорят, что на этих элементах отображение просто не определено. Обычно, когда говорят об отображении на множестве A, подразумевают, что отображение определено на всех элементах $x \in A$, т.е.

$$\forall x \in A \exists y \in A : \; y = \mathcal{F}(x).$$

У некоторых $y \in A$ может не быть прообраза. Тогда говорят, что образ отображения $\mathcal{F}$ не совпадает с множеством A.

Определение 5. *Если $\mathcal{F}$ — отображение из A в A и его образ совпадает со всем множеством A, т.е.*

$$\forall y \in A \exists x \in A : y = \mathcal{F}(x),$$

то отображение называют сюръекцией (или «отображением на»).

У некоторых $y \in A$ может быть несколько прообразов.

Определение 6. *Если $\mathcal{F}$ — отображение из A в A и*

$$\forall y \in A \text{ существует не более одного } x \in A : y = \mathcal{F}(x),$$

то отображение называют инъекцией (или вложением).

При инъекции разные элементы переходят в разные.

Определение 7. *Отображение, которое инъективно и сюръективно, называют биекцией (или взаимно однозначным отображением).*

Итак, прообразов может быть несколько, но образ — только один.

Пример 2. *Пусть $A = \{a, b, c\}$. Отношение $\mathcal{R} = \{(a, a), (a, b), (a, c)\}$ — не отображение. Отношение $\mathcal{F} = \{(a, a), (b, a), (c, a)\}$ — отображение, определенное на всем A, не инъекция и не сюръекция. Отношение $\mathcal{G} = \{(a, b), (b, c), (c, a)\}$ — биекция на A.*

Для каждого отображения есть обратное отношение. Но оно может не быть отображением.

Утверждение 1. *Только у инъекции обратное отношение является отображением. У биекции обратное отображение — тоже биекция.*

Доказательство. Если $\mathcal{F}$ — инъекция, то у некоторых $y \in A$ прообраза вообще нет (ведь $\mathcal{F}$ может не быть сюръекцией), а у других $y \in A$ ровно один прообраз $x \in A$. Для каждого такого y имеем $(y, x) \in \mathcal{F}^{-1}$, а других пар $(y, z) \in \mathcal{F}^{-1}$ нет. Получили определение отображения. Если теперь $\mathcal{F}$ не инъекция, то найдутся две пары вида $(x, y) \in \mathcal{F}$ и $(z, y) \in \mathcal{F}$, где $x \neq z$. Но тогда $(y, x) \in \mathcal{F}^{-1}$ и $(y, z) \in \mathcal{F}^{-1}$ — нарушается определение отображения.

Разберемся с биекциями. Пусть $\mathcal{F} : A \to A$ — биекция. Для $\mathcal{F}$ обратное отображение определено (ведь любая биекция — это инъекция) и определено на всем A. Поскольку само $\mathcal{F}$ тоже определено на всем A, то $\mathcal{F}^{-1}$ — сюръекция. А инъективностью обладают все обратные отображения: если бы $(y, x) \in \mathcal{F}^{-1}$ и $(z, x) \in \mathcal{F}^{-1}$ для $y \neq z$, то $(x, y) \in \mathcal{F}$ и $(x, z) \in \mathcal{F}$ — нарушено определение отображения. □

Пока что речь шла про произвольное множество A. У конечных множеств своя специфика.

Утверждение 2. *Пусть A — конечное множество. Тогда любая инъекция $\mathcal{F} : A \to A$ автоматически сюръекция. Любая сюръекция — автоматически инъекция.*

Доказательство. Пусть A содержит $n \geq 1$ элементов. Разберем первый случай. Пусть $\mathcal{F}$ — инъекция. Тогда раззные элементы переходят в разные, то есть в образе $\mathcal{F}(A)$ ровно n различных элементов. Но поскольку $\mathcal{F}(A) \subset A$, то значит $\mathcal{F}(A) = A$, т.е. $\mathcal{F}$ сюръекция.

Разберем второй случай. Пусть $\mathcal{F}$ — сюръекция, т.е. у каждого элемента y из A есть прообраз (точнее, полный прообраз — подмножество в A, обозначим его $\mathcal{F}^{-1}(y)$, а число элементов в нем q_y). Но у разных элементов прообразы не пересекаются. Действительно, если $y \neq z$, а $\mathcal{F}^{-1}(y) \cap \mathcal{F}^{-1}(z) \neq \varnothing$, то есть $x \in A$: $\mathcal{F}(x) = y$ и $\mathcal{F}(x) = z$?! Объединим все эти прообразы и посчитаем количество элементов в объединении:

$$\sum_{y \in A} q_y.$$

Эта сумма содержит n натуральных слагаемых и не превосходит n. Значит каждое слагаемое равно 1, а это и означает, что F — инъекция. □

Обратите внимание, что для бесконечных множеств наше утверждение неверно.

Пример 3. *Отображение $x \to x + 1$ на множестве натуральных чисел инъективно, но не сюръективно. Отображение $x \to x - 1$ на том же множестве (с поправкой $1 \to 1$) сюръективно, но не инъективно.*

Разберем другие типы бинарных отношений.

Определение 8. *Отношение $\mathcal{R}$ называется рефлексивным, если $\forall x \in A : (x, x) \in \mathcal{R}$.*

Например, отношение $\leqslant$ на множестве целых чисел рефлексивно, а отношение $<$ нерефлексивно.

Определение 9. *Отношение $\mathcal{R}$ называется симметричным, если из $(x, y) \in \mathcal{R}$ следует $(y, x) \in \mathcal{R}$.*

Например, отношение на множестве студентов в этой аудитории «сидим за одной партой» симметрично.

Утверждение 3. *Другой способ сказать, что отношение $\mathcal{R}$ симметрично: $\mathcal{R} = \mathcal{R}^{-1}$.*

Доказательство. Пусть отношение симметрично. Докажем, что $\mathcal{R} = \mathcal{R}^{-1}$. Пусть пара $(x, y) \in \mathcal{R}$. Тогда пара $(y, x) \in \mathcal{R}$, а тогда $(x, y) \in \mathcal{R}^{-1}$. Пусть пара $(x, y) \in \mathcal{R}^{-1}$. Тогда пара $(y, x) \in \mathcal{R}$, а тогда пара $(x, y) \in \mathcal{R}$. Значит множества $\mathcal{R}$ и $\mathcal{R}^{-1}$ совпадают.

Пусть теперь множества $\mathcal{R}$ и $\mathcal{R}^{-1}$ совпадают. Если $(x, y) \in \mathcal{R}$, то $(x, y) \in \mathcal{R}^{-1}$. Тогда $(y, x) \in \mathcal{R}$. □

Хороший способ представить себе отношение на множестве из n элементов — записать его матрицей размера $n \times n$ из нулей и единиц. Единица в матрице $R = (r_{ij})$ ставится в точности тогда, когда $a_i \mathcal{R} a_j$.

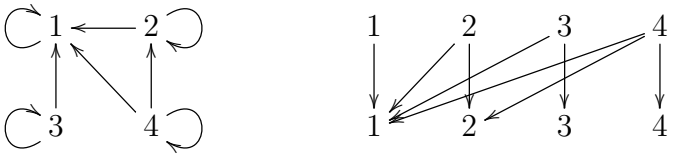
Другой способ изобразить отношение — нарисовать его *ориентированный граф (орграф)*. Изобразим на плоскости n точек — элементов множества $A = \{a_1, \ldots, a_n\}$ и соединим a_j с a_j стрелкой в точности тогда, когда $a_i \mathcal{R} a_j$.

Третий способ изобразить отношение — нарисовать его *функциональную схему*. Из элементов $a_1, \dots a_n$ формируют два одинаковых столбца (или две строки) и отношение $a_i \mathcal{R} a_j$ изображают стрелкой из элемента a_i левого столбца в элемент a_j правого

Пример 4. *Пусть* $A = \{1, 2, 3, 4\}$, *а отношение* $\mathcal{R}$ *описывается условием*

$$(x, y) \in \mathcal{R} \Longleftrightarrow x \text{ делится нацело на } y.$$

Тогда матрица отношения примет вид $\mathcal{R} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$. *Орграф и функциональная схема:*



Упражнение 1. *Отношение является отображением тогда и только тогда, когда в каждой строке этой матрицы не более одной единицы. Это отображение является сюръекцией, когда вдобавок в матрице нет нулевых столбцов. Отношение рефлексивно тогда и только тогда, когда на диагонали матрицы стоят единицы. Отношение симметрично тогда и только тогда, когда матрица симметрична (не меняется при транспозиции). Придумайте сами характеристическое свойство инъекций.*

Определение 10. *Отношение* $\mathcal{R}$ *называется транзитивным, если*

$$\begin{cases} (x, y) \in \mathcal{R}, \\ (y, z) \in \mathcal{R} \end{cases} \Longrightarrow (x, z) \in \mathcal{R}.$$

Например, отношения из примера 4 транзитивно.

Упражнение 2. *Что такое транзитивность с точки зрения орграфа отношения?*

Определение 11. *Отношение, которое одновременно рефлекцивно, симметрично и транзитивно, называется эквивалентность. Его часто обозначают* $x \sim y$.

Например, отношение «треугольники равны» на множестве треугольников на плоскости является эквивалентностью. Отношение на множестве студентов Московского университета «они учатся в одной академической группе» — тоже эквивалентность.

Определение 12. *Пусть* $\mathcal{R}$ *— отношение эквивалентности на множестве* A. *Говорят, что элементы* x *и* y *лежат в одном классе эквивалентности, если* $x \mathcal{R} y$. *В противном случае говорят, что они лежат в разных классах эквивалентности.*

Утверждение 4. *Любое отношение эквивалентности разбивает множество* A *на непересекающиеся подмножества — классы эквивалентности.*

Доказательство. Пусть A_1 и A_2 — два различных пересекающихся класса эквивалентности. Раз они пересекаются, то найдется элемент $x \in A_1$, $x \in A_2$. Раз они различны, то найдется другой элемент $y \in A_1$, $y \notin A_2$ (либо наоборот, но тогда мы просто поменяем номера классов местами). Получается, что, с одной стороны, $x \sim y$ (они оба лежат в A_1), а, с другой стороны, $x \nsim y$ (они лежат в разных классах)?! $\square$

Лекция 2. Порядок на множествах. Комбинаторика (начало).

Определение 13. *Отношение $\mathcal{R}$ на множестве A называется отношением порядка, если оно рефлексивно, транзитивно и антисимметрично*

$$
\begin{aligned}
&1) \quad (x,x) \in \mathcal{R} \quad \forall x;\\
&2) \quad (x,y) \in \mathcal{R},\ (y,z) \in \mathcal{R} \Longrightarrow (x,z) \in \mathcal{R};\\
&3) \quad (x,y) \in \mathcal{R},\ x \neq y \Longrightarrow (y,x) \notin \mathcal{R}.
\end{aligned}
$$

Отношение порядка часто обозначается символом $\leqslant$. Множество A, на котором введено отношение порядка называется частично упорядоченным множеством.

Пример 5. *Отношение классического порядка $x \leqslant y$ на множестве натуральных чисел является отношением порядка.*

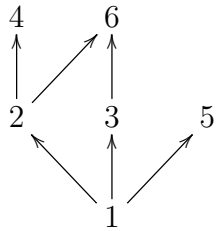
Пример 6. *Скажем, что вектор* $\mathbf{x}$ *с вещественными координатами (x_1, x_2) меньше либо равен вектора* $\mathbf{y}$ *с координатами (y_1, y_2), если $x_1 \leqslant y_1$ и $x_2 \leqslant y_2$. Это отношение — порядок.*

Пример 7. *Отношение $x \leqslant y \Longleftrightarrow y$ делится нацело на x на множестве натуральных чисел тоже является отношением порядка.*

Определение 14. *Порядок $\leqslant$ называется линейным, если для любой пары (x, y) либо $x \leqslant y$, либо $y \leqslant x$.*

Отношение порядка в примере 5 линейно. В примере 6 линейным не является: векторы $(0, 1)$ и $(1, 0)$ несравнимы. Что насчет примера 7?

Порядок на множестве удобно иллюстрировать ориентированным графом отношения. При этом часто точки графа располагают на горизонтальных уровнях в порядке возрастания (это называется *диаграмма Хассе*). Возьмем, например, порядок из примера 7 (для простоты введем его на множестве $A = \{1, 2, 3, 4, 5, 6\}$). Иллюстрация выглядит так



Упражнение 3. *Продолжите эту диаграмму на все множество натуральных чисел. Какие число лежат на «нулевом» уровне? Какие на первом? Какие на втором?*

Упражнение 4. *Как выглядит диаграмма Хассе линейно упорядоченного множества?*

Из любого порядка можно сделать *строгий порядок*: $x < y$ тогда и только тогда, когда $x \leqslant y$ и $x \neq y$. Когда про множество говорят, что *оно упорядочено*, то имеют в виду, что на нем введено отношение линейного порядка.

Поговорим о комбинаторике.

Будем работать с конечным множеством A и с *выборками* элементов этого множества. Выборки бывают *упорядоченные* и *неупорядоченные*, *с повторениями* и *без повторений*.

Определение 15. *Упорядоченная выборка без повторений объема n (размещение) — это произвольное n-элементное подмножество множества A на котором введен строгий линейный порядок. Удобно воспринимать такую выборку как вектор множества A^n все кординаты которого различны.*

Часто вместо вектора говорят о *слове* в алфавите A.

Определение 16. *Упорядоченная выборка с повторениями объема n (размещение с повторениями) — это произвольный вектор множества A^n.*

Геометрически при $k = 2$ это точки квадрата $A \times A$, а при $k = 3$ точки куба $A \times A \times A$.

Определение 17. *Неупорядоченная выборка без повторений объема n (сочетание) — это произвольное n-элементное подмножество элементов множества A.*

От любой упорядоченной выборки можно перейти к неупорядоченной следующим образом. Введем на множестве A^n векторов с n координатами отношение эквивалентности: два вектора эквивалентны, если один к другому сводится перестановкой координат. Классы эквивалентности — это и есть неупорядоченная выборка.

Определение 18. *Неупорядоченная выборка с повторениями объема n (сочетание с повторениями) — это класс эквивалентности по описанному выше отношению.*

Прежде всего, сформулируем два очевидных комбинаторных принципа.

Утверждение 5. *Если действие A можно совершить m различными способами, а действие B — n способами, причем результат первого действия никак не влияет на второе действие, то всего есть $n \cdot m$ способов осуществить оба действия: A и затем B.*

Если действие A можно совершить m различными способами, а действие B — n способами, причем осуществить и A, и B вместе невозможно, то действие «A или B» можно осуществить $m + n$ способами.

Первое утверждение называют «правило произведения», а второе — «правило суммы». Во многих задачах эти принципы применяются последовательно необходимое число раз.

Очень часто встречается *булев куб* размерности k — множество всех векторов длины k с координатами из множества $A = \{0, 1\}$. Если множество $A = \{a_1, \ldots, a_n\}$, то говорят о k–мерном кубе со стороной размера n.

Утверждение 6. *Число элементов k–мерного куба со стороной размера n равно 2^{n^k}.*

Доказательство. Первый элемент можно выбрать n способами, второй — тоже n способами и так k раз. □

Утверждение 7. *Пусть $A = \{a_1, \ldots, a_n\}$.*

1. *Число различных подмножеств множества A равно 2^n.*

2. *Число всех перестановок на A равно $n! = 1 \cdot 2 \dots n$.*

3. *Зафиксируем целое число $k \in [0, n]$. Число всех размещений из A по k равно*

$$(n)_k := n(n-1)\dots(n-k+1) = \frac{n!}{(n-k)!}.$$

4. *Число всех сочетаний из A по k равно*

$$\binom{n}{k} = C_n^k := \frac{n!}{k!(n-k)!}.$$

5. *Число всех размещений с повторениями из A по k равно*

$$\hat{(n)}_k := n^k.$$

6. *Число всех сочетаний с повторениями из A по k равно*

$$\hat{C}_n^k = C_{n+k-1}^{n-1}.$$

Доказательство. 1. Организуем алгоритм последовательного перебора элементов множества A. Для перовго элемента a_1 есть две возможности: включить его в подмножество или не включать. То же самое для элемента a_2, a_3 и т.д. Для того, чтобы сформировать подмножество необходимо провести выбор для каждого элемента. Согласно принципу произведения, получаем $2 \cdot 2 \dots 2 = 2^n$ вариантов.

2. и 3. Организуем перебор мест в предполагаемой выборке. На первое место можем поставить любой элемент множества A — n возможностей. На второе место — любой элемент из оставшихся — $n-1$ возможность. И т.д. Всего получим $n \cdot (n-1) \dots (n-k+1) = (n)_k$ вариантов.

4. Пусть дано некоторое сочетание E из множества A по k элементов. Мы можем упорядочить элементы множества E $k!$ способами (число перестановок). То есть каждому сочетанию соответствует $k!$ различных размещений. Тогда сочетаний в $k!$ раз меньше, чем размещений, т.е.

$$\binom{n}{k} = \frac{(n)_k}{k!} = \frac{n!}{k!(n-k)!}.$$

5. На первое место в выборке можем поставить любой элемент из A — n возможностей. На второе место — снова любой (повторения разрешены) — n возможностей. И так k раз.

6. Пусть элемент a_1 встречается в выбоке k_1 раз, элемент a_2 встречается k_2 раз и т.д. Закодируем каждую выборку последовательностью нулей и единиц следующим образом. Сначала запишем k_1 раз единицу, затем ноль, затем k_2 раз единицу, затем ноль, затем k_3 раз единицу и т.д. В конце вектора будет последний блок из k_n единиц. Мы построили отображение: каждой выборке сопоставили вектор длины $k_1+1+k_2+1+\dots+k_n = k+n-1$.

По построению наше кодирование инъекция. Легко описать образ отображения — все векторы длины $k+n-1$ из нулей и единиц, среди координат которых ровно $n-1$ нулей. Биекция сохраняет мощности множеств. Значит для ответа остается посчитать число таких векторов. Для того, чтобы составить такой вектор достаточно поместить $n-1$ ноль на $k+n-1$ место (оставшиеся места заполняем потом единицами). Иными словами, надо выбрать $n-1$ координату из $k+n-1$, причем порядок выбора не важен. Это можно сделать C_{n+k-1}^{n-1} способами. □

Лекция 3. Комбинаторика (продолжение)

Докажем несколько свойств чисел $\binom{n}{k}$ (эти числа называют биномиальными коэффициентами, а в русской математической литературе их еще часто обозначают C_n^k).

Утверждение 8. *Справедливы следующие соотношения*

1. $(a+b)^n = \sum_{k=0}^{n} C_n^k a^k b^{n-k}$,
2. $\sum_{k=0}^{n} C_n^k = 2^n, \quad \sum_{k=0}^{n} (-1)^k C_n^k = 0$,
3. $C_{n-1}^k + C_{n-1}^{k-1} = C_n^k$,
4. $C_n^{n-k} = C_n^k$,
5. *при фиксированном n последовательность C_n^k возрастает при $k \le n/2$, а затем убывает.*
6. *Пусть A — произвольное n-элементное подмножества $\mathbb{N}$, а $k \le n$. Тогда C_n^k — число всех монотонно возрастающих k–элеменных последовательностей чисел из A.*

Доказательство. Первое свойство (бином Ньютона) доказано в курсе математического анализа. Второе следует из первого при $a = b = 1$ и при $a = 1$, $b = -1$. Третье проверяется непосредственно. Четвертое очевидно. Проверим пятое свойство. Легко видеть, что

$$\frac{C_n^k}{C_n^{k-1}} = \frac{n!}{k!(n-k)!} \frac{(k-1)!(n-k+1)!}{n!} = \frac{n-k+1}{k}.$$

При $k \le n/2$ имеем $n-k+1 > k$, т.е. $C_n^k > C_n^{k-1}$. Убывание при $k \ge n/2$ следует из свойства 4.

Проверим шестое свойство. Количество выборок без повторения k элементов из A равно $(n)_k$. Для каждой выборки есть только одна монотонно возрастающая перестановка ее членов. Введем на выборках отношение эквивалентности: две выборки эквивалентны, если отличаются перестановкой своих элементов. Тогда число классов эквивалентности равно $\frac{(n)_k}{k!} = C_n^k$. Мы берем ровно одного представителя из каждого класса. Значит число этих представителей тоже C_n^k. □

Это далеко не полный список свойств биномиальных коэффициентов. Применим одно из них для доказательства простой *формулы включений–исключений*. Представим субе такую ситуацию. Пусть A_1, $A_2, \ldots$, A_n — какие-то конечные подмножества множества A. Предположим мы умеем считать количество элементов $|A_j|$ в каждом из них. Можем ли мы узнать, сколько элементов в их объединении? Очевидно, нет. Вот если все они попарно не пересекаются, то

$$|A_1 \cup A_2 \cup \cdots \cup A_n| = \sum_{k=1}^{n} |A_k|.$$

А если пересекаются? Попробуем два множества:

$$|A_1 \cup A_2| = |A_1| + |A_2| - |A_1 \cap A_2|.$$

Вообще, если никакая тройка множеств не имеет общих элементов, то

$$|A_1 \cup A_2 \cup \cdots \cup A_n| = \sum_{k=1}^{n} |A_k| - \sum_{1 \le k_1 < k_2 \le n} |A_{k_1} \cap A_{k_2}|.$$

Действительно, суммируя мощности множеств A_k, мы некоторые элементы посчитали дважды (те, которые лежат в каком-либо пересечении $A_{k_1} \cap A_{k_2}$). При этом ни один элемент мы не посчитали трижды (по условию, тройных пересечений нет). Вычтем количество элементов, которые мы посчитали дважды — получим правильный ответ.

Теорема 1. *Пусть A_1, $A_2, \ldots$, A_n — какие-то конечные подмножества множества A. Тогда*

$$|A_1 \cup A_2 \cup \cdots \cup A_n| = \sum_{k=1}^{n} |A_k| - \sum_{1 \le k_1 < k_2 \le n} |A_{k_1} \cap A_{k_2}| + \sum_{1 \le k_1 < k_2 < k_3 \le n} |A_{k_1} \cap A_{k_2} \cap A_{k_3}| - \ldots$$
$$\cdots + (-1)^j \sum_{1 \le k_1 < k_2 < \cdots < k_j \le n} |A_{k_1} \cap A_{k_2} \cap \cdots \cap A_{k_j}| + \cdots + (-1)^{n-1} |A_1 \cap A_2 \cap \cdots \cap A_n|. \quad (1)$$

Доказательство. Введем в оборот полезную индикаторную функцию множества $B \subset A$

$$\chi_B(x) = \begin{cases} 1, & \text{если } x \in B; \\ 0, & \text{если } x \notin B. \end{cases}$$

Очевидно, что $\sum_{x \in A} \chi_B(x) = |B|$. Рассмотрим функцию на элементах $a \in A$

$$f(a) = \sum_{k=1}^{n} \chi_{A_k}(a) - \sum_{1 \le k_1 < k_2 \le n} \chi_{A_{k_1} \cap A_{k_2}}(a) + \cdots + (-1)^{n-1} \chi_{A_1 \cap A_2 \cap \cdots \cap A_n}(a). \quad (2)$$

Если $a \notin A_1 \cap \cdots \cap A_n$, то все слагаемые равны нулю и $f(a) = 0$. Зафиксируем произвольный элемент $a \in A_1 \cap \cdots \cap A_n$. Для него можно однозначно указать те множества, в которые он входит (назовем их «избранными», а их число обозначим m). Сколько раз мы посчитаем этот элемент в правой части 2? Вначале m раз в первой сумме. Затем некоторое количество раз во второй сумме — столько раз, сколько попарных пересечений $A_{k_1} \cap A_{k_2}$ с индексами

$k_1 < k_2$ содержат наш элемент. А сколько таких пересечений? Для того, чтобы элемент a попал в такое попарное пересечение необходимо и достаточно, чтобы он попал в оба множества A_{k_1} и A_{k_2}, т.е. чтобы оба эти множества были «избранными». Два различных индекса из m индексов можно выбрать C_m^2 способами (см. шестое свойство в утверждении 8). Итак, мы сосчитали элемент a во второй сумме C_m^2 раз. Аналогично, в следующей сумме мы сосчитаем его C_m^3 раз и т.д. Последний раз мы сосчитаем его в m-ой сумме. Сосчитаем мы там его один раз (в пересечении всех «избранных» множеств). Остальные суммы будут равны нулю. Получается, что

$$f(a) = m - C_m^2 + C_m^3 - \cdots + (-1)^{m+1} = C_m^1 - C_m^2 + C_m^3 - \cdots + (-1)^{m+1} C_m^m = C_m^0 = 1$$

(мы применили второе свойство из утверждения 8). Оказывается, что f — характеристическая функция множества $A_1 \cup \cdots \cup A_n$. Тогда

$$\sum_{a \in A} f(a) = |A_1 \cup A_2 \cup \cdots \cup A_n| = \sum_{k=1}^{n} |A_k| - \ldots .$$

□

Определение 19. *Разбиение множества A на k подмножеств — это произвольный набор A_1, $A_2, \ldots, A_k$ попарно не пересекающихся непустых подмножеств в A, объединение которых равно A (порядок, в котором указываются множества A_j не важен).*

Пример 8. *Пусть $A = \{a, b, c\}$. Перечислим все разбиения этого множества:*

$$\{a, b, c\}; \qquad \{a, b\} \text{ и } \{c\}; \qquad \{a, c\} \text{ и } \{b\};$$
$$\{b, c\} \text{ и } \{a\}; \qquad \{a\}, \ \{b\} \text{ и } \{c\}.$$

Количество различных разбиений n–элементного множества A на k подмножеств обозначают $\Phi(n, k)$. Очевидно, что $\Phi(n, k) = 0$ при $k > n$, $\Phi(n, n) = 1$. Удобно считать, что $\Phi(n, 0) = \Phi(0, k) = 0$. Количество всех разбиений n–элементного множества A обозначают $\Phi(n)$,

$$\Phi(n) = \sum_{k=1}^{n} \Phi(n, k).$$

Утверждение 9. *Для любых n, $k \geq 1$*

$$\Phi(n, k) = \Phi(n-1, k-1) + k\Phi(n-1, k).$$

Доказательство. Зафиксируем элемент a множества A и составим произвольное разбиение A на k подмножеств. Возможны два случая: в этом разбиении есть подмножество $\{a\}$ или такого подмножества нет. В первом случае для достроения разбиения необходимо разбить множество $A \setminus \{a\}$ на $k-1$ подмножество. Таких разбиений $\Phi(n-1, k-1)$. Во втором случае необходимо разбить множество $A \setminus \{a\}$ на k подмножество и затем добавить элемент a в одно из подмножеств разбиения. Первое действие можно осуществить $\Phi(n-1, k)$ способами, а второе k способами. Отсюда получаем наше утверждение. □

Вы, наверное, заметили, что мы не получили явных формул для чисел $\Phi(k,n)$ — только *рекуррентные соотношения*. Но если вдуматься, то $n!$ — это тоже всего лишь новое обозначение. Для малых n факториалы легко считать по определению, но при больших n это совершенно неудобно. Возникает вопрос: можно ли *асимптотически* выразить факториалы через известные нам операции?

Утверждение 10. *При всех $n \in \mathbb{N}$ справедливы оценки*

$$\left(\frac{n}{e}\right)^n < n! < 2\left(\frac{n}{2}\right)^n. \tag{3}$$

Доказательство. Левое неравенство докажем по индукции. Мы знаем, что $e = \lim\left(1+\frac{1}{n}\right)^n$, причем

$$2 \leq \left(1+\frac{1}{n}\right)^n < e < 3$$

при всех $n \geq 1$. Тогда база индукции $\frac{1}{e} < 1$ доказана. Индуктивный переход:

$$(n+1)! = (n+1)n! > (n+1)\left(\frac{n}{e}\right)^n \text{ для числа } n \text{ неравенство уже доказано} =$$
$$= \frac{n+1}{e^n}n^n > \frac{n+1}{e^n}\frac{(n+1)^n}{e} = \frac{(n+1)^n}{e^{n+1}}.$$

Правое неравенство докажем при помощи элементарного неравенства $(a+b)^2 \geqslant 4ab$. Если число n нечетно, то

$$n! = n\cdot(1\cdot(n-1))\cdot(2\cdot(n-2))\ldots\left(\frac{n-1}{2}\cdot\frac{n+1}{2}\right) \leqslant$$
$$\leqslant n\cdot\underbrace{\left(\frac{n}{2}\right)^2\ldots\left(\frac{n}{2}\right)^2}_{(n-1)/2 \text{ сомножителей}} = n\cdot\left(\frac{n}{2}\right)^{n-1} = 2\left(\frac{n}{2}\right)^n.$$

Если n четно, то

$$n! = n(n-1)! < 2n\left(\frac{n-1}{2}\right)^{n-1} \leqslant n\left(\frac{n}{2}\right)^{n-1} = 2\left(\frac{n}{2}\right)^n.$$

В последнем переходе было использовано неравенство $2(n-1)^{n-1} \leq n^{n-1}$. □

Позже мы узнаем, что левая оценка в (3) очень близка к точной асимптотической формуле для чисел $n!$. А именно,

$$\lim\frac{n!}{a_n} = 1, \qquad a_n = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

(это называется *формула Стирлинга*), но доказательство требует знания теории интегрирования.

Упражнение 5. *Из определение имеем* $C_n^1 = n$ *(линейный рост),* $C_n^2 = \frac{n(n-1)}{2}$ *(квадратичный рост) и т.д. Но из утверждения 8 нам известно, что из чисел* C_n^k *самое большое* $C_n^{n/2}$ *(считаем для простоты* n *четным). А как быстро растет последовательность* $C_n^{n/2}$*? Докажите, что*

$$\frac{4^{n-1}}{e^n} < C_n^{n/2} < 2e^n$$

при всех четных $n \geq 2$*.*

Упражнение 6. *С помощью индукции докажите, что* $\Phi(n, 2) = 2^{n-1} - 1$ *при всех* $n \geq 1$*.*

Лекция 4. Булевы функции (начало)

Определение 20. *Логической (двоичной, булевой) переменной называется переменная, принимающая значения* 0 *(ложь) и* 1 *(истина). Булевой функцией называется функция от конечного числа логических переменных, принимающая значения* 0 *и* 1.

Обозначим через E_2 множество $\{0, 1\}$. Таким образом, мы рассматриваем отображения из множества $(E_2)^n$ в E_2. Самый простой способ задать такое отображение — перечислить все его значения. Это называется *таблица истинности* функции f.

Пример 9. *Приведем таблицы истинности всех булевых функций одной переменной*

x	0	1	x	$\overline{x}$
0	0	1	0	1
1	0	1	1	0

Наверху написаны названия этих функций: 0 *— константа ноль (абсолютная ложь),* 1 *— константа один (абсолютная истина), x — тождественная функция, $\overline{x}$ — отрицание (используется также обозначение $\neg x$).*

Пример 10. *Приведем таблицы истинности некоторых (наиболее популярных) функций двух переменных*

x	y	$x \vee y$	xy	$x \oplus y$	$x \Rightarrow y$	$x \sim y$	$x \mid y$	$x \downarrow y$
0	0	0	0	0	1	1	1	1
0	1	1	0	1	1	0	1	0
1	0	1	0	1	0	0	1	0
1	1	1	1	0	1	1	0	0

При заполнении таблицы значения переменных заполняются в лексикографическом порядке (по возрастанию двоичных чисел). Функция $x \vee y$ называется *дизъюнкцией*. Функция $xy = x \cdot y = x \wedge y = x\&y$ называется *конъюнкцией*. Функция $x \oplus y$ называется *сложением по модулю 2*. Функция $x \Rightarrow y$ называется *импликацией или логическим следствием*. Функция $x \sim y$ равна отрицанию сложения мо модулю 2 и называется *эквивалентностью*. Функция $x \mid y$ равна отрицанию конъюнкции и называется *штрих Шеффера*. Функция $x \downarrow y$ равна отрицанию дизъюнкции и называется *стрелка Пирса*.

Посчитаем, сколько всего имеется булевых функций от n переменных.

Лемма 1. *Существует ровно 2^{2^n} различных отображений из $(E_2)^n$ в E_2.*

Доказательство. Каждая функция задается столбцом из нулей и единиц. Обозначим высоту столбца N. Тогда имеем 2^N вариантов расстановки нулей и единиц. Теперь посчитаем число N. Каждый столбец имеет столько строчек, сколько разных наборов из нулей и единиц могут принять n логических переменных. Составляя такой набор мы расставляем нули и единицы на n мест, т.е. имеем $N = 2^n$ наборов. □

Логично записывать новые функции через суперпозицию простейших, например $f(x,y) = \neg((xy) \oplus (\neg z))$. При такой записи договариваемся о приоритетах операций: наивысший приоритет имеет отрицание, затем конъюнкция, затем дизъюнкция и сложение по модулю два (их приоритет одинаков). Так, последнюю функцию можно записать $g(x,y) = \overline{xy \oplus \overline{z}}$ и функции f и g равны.

А что означает вообще фраза 'функции f и g равны'? Видимо, имеют одинаковые столбцы в таблице истинности. А как быть тогда с функциями $(x \oplus y) \oplus y$ и x?

Определение 21. *Переменная x_i для функции $f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)$ называется существенной, если найдется такой набор $(\alpha_1, \alpha_2, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n)$ нулей и единиц, что*

$$f(\alpha_1, \ldots, \alpha_{i-1}, 0, \alpha_{i+1}, \ldots, \alpha_n) \neq f(\alpha_1, \ldots, \alpha_{i-1}, 1, \alpha_{i+1}, \ldots, \alpha_n).$$

В противном случае переменная называется фиктивной.

Для краткости, наборы нулей и единиц, отличающиеся значением ровно одной переменной x_i, называют *соседними по x_i*. Тогда наше определение можно сформулировать так: переменная x_i называется *существенной*, если найдутся два соседних по x_i набора, с разным значением функции. Переменная x_i называется *фиктивной для функции f*, если f принимает одинаковые значения на любой паре соседних по x_i наборов.

Пример 11. *У функции $f(x,y) = (x \oplus y) \oplus y$ переменная y является фиктивной. Действительно, $f(0,0) = f(0,1)$ и $f(1,0) = f(1,1)$.*

Фиктивную переменную можно изъять из множества аргументов функции без изменения самой функции. Обратно, в аргументы любой функции можно добавить фиктивную переменную.

Определение 22. *Две функции называются равными, если одну из них можно получить из другой добавлением и/или изъятием некоторого количества фиктивных переменных.*

Пример 12. *Функции $f(x,y) = (x \oplus y) \oplus y$ и $g(x) = x$ равны.*

Упражнение 7. *Ни одна из функций $f_1, \ldots f_7$ в примере 10 не имеет фиктивных переменных. А вот в примере 9 есть функции с фиктивными переменными.*

Приведем список основных равенств для введенных выше функций. Все их надо знать наизусть!

Коммутативность:

$$x \vee y = y \vee x; \qquad xy = yx; \qquad x \oplus y = y \oplus x;$$
$$x \sim y = y \sim x; \qquad x \mid y = y \mid x; \qquad x \downarrow y = y \downarrow x.$$

Ассоциативность:

$$(x \vee y) \vee z = x \vee (y \vee z); \qquad (xy)z = x(yz); \qquad (x \oplus y) \oplus z = x \oplus (y \oplus z).$$

Обратите внимание, что стрелка Пирса и штрих Шеффера ассоциативностью не обладают.

Упражнение 8. *Проверьте, что функции* $(x \mid y) \mid z$ *и* $x \mid (y \mid z)$ *отличаются на наборе* $(1,1,0)$. *Докажите, что функции* $(x \downarrow y) \downarrow z$ *и* $x \downarrow (y \downarrow z)$ *не равны.*

Дистрибутивность:

$$(x \vee y)z = xz \vee yz; \qquad (x \oplus y)z = xz \oplus yz; \qquad (xy) \vee z = (x \vee z) \cdot (y \vee z).$$

Правила де Моргана

$$\overline{x \vee y} = \overline{x} \cdot \overline{y}; \qquad \overline{xy} = \overline{x} \vee \overline{y}.$$

Законы поглощения

$$x \vee x = x; \qquad x \cdot x = x; \qquad x \vee \overline{x} = 1; \qquad x \cdot \overline{x} = 0;$$
$$x \vee 1 = 1; \qquad x \cdot 1 = x; \qquad x \vee 0 = x; \qquad x \cdot 0 = 0.$$

Просто важные равенства:

$$\overline{\overline{x}} = x; \qquad x \mid y = \overline{xy}; \qquad x \downarrow y = \overline{x \vee y};$$
$$x \Rightarrow y = \overline{x} \vee y; \qquad x \oplus y = (x\overline{y}) \vee (\overline{x}y); \qquad x \sim y = (xy) \vee (\overline{xy}).$$

Величина 2^{2^n} очень быстро растет с ростом числа n. Например, если мы захотим выписать таблицу всех булевых функций от 5 переменных, то такая таблица будет содержать $2^5 = 32$ строки и $2^{2^5} = 4'294'967'296$ столбцов. Необходимо какое-то представление булевых функций более простое, чем таблица истинности.

Договоримся об обозначении $\begin{cases} x^0 = \overline{x}, \\ x^1 = x \end{cases}$. Сразу заметим, что $x^\sigma = 1 \Leftrightarrow x = \sigma$ для $\sigma \in \{0, 1\}$.

Теорема 2 ((о совершенной дизъюнктивной нормальной форме))**.** *Любая ненулевая булева функция от* n *логических переменных допускает представление*

$$f(x_1, \ldots, x_n) = \bigvee_{(\sigma_1,\ldots,\sigma_n):\, f(\sigma_1,\ldots,\sigma_n)=1} x_1^{\sigma_1} x_2^{\sigma_2} \ldots x_n^{\sigma_n}.$$

Пример 13. *Запишем СДНФ для штриха Шеффера, пользуясь таблицей истинности из примера 10*

$$x \mid y = x^0y^0 \vee x^0y^1 \vee x^1y^0 = \overline{x} \cdot \overline{y} \vee \overline{x}y \vee x\overline{y}.$$

Тот же результат можно получить преобразованиями

$$x \mid y = \overline{xy} = \overline{x} \vee \overline{y} = \overline{x} \cdot 1 \vee \overline{y} \cdot 1 = \overline{x}(y \vee \overline{y}) \vee \overline{y}(x \vee \overline{x}) = (\overline{x}y \vee \overline{x} \cdot \overline{y}) \vee (x\overline{y} \vee \overline{x} \cdot \overline{y}) =$$
$$= \overline{x}y \vee \overline{x} \cdot \overline{y} \vee x\overline{y}.$$

Доказательство. Возьмем произвольный набор $(\alpha_1, \ldots, \alpha_n)$ и вычислим значение правой части на этом наборе. В нашу большую дизъюнкцию войдут значения внутренних конъюнкций. Конъюнкция равна 1 в точности тогда, когда равны 1 все ее сомножители, т.е. в точности при $\alpha_1 = \sigma_1, \ldots, \ \alpha_n = \sigma_n$. Получается, что если $f(\alpha_1, \ldots, \alpha_n) = 1$, то ровно один операнд дизъюнкции равен 1 и правая часть равна 1. Если же $f(\alpha_1, \ldots, \alpha_n) = 0$, то все операнды дизъюнкции равны 0 и правая часть равна 0. $\square$

Иногда удобно раскладывать функцию n переменных по функциям меньшего числа, скажем k переменных.

Теорема 3. *Любая ненулевая булева функция от n логических переменных допускает представление*

$$f(x_1,\dots,x_n)=\bigvee_{(\sigma_1,\dots,\sigma_k)\in E_2^k} x_1^{\sigma_1}\cdot x_2^{\sigma_2}\cdot\dots\cdot x_k^{\sigma_k}\cdot f(\sigma_1,\dots,\sigma_k,x_{k+1},\dots,x_n).$$

Доказательство. Возьмем произвольный набор $(\alpha_1,\dots,\alpha_n)$ и вычислим значение правой части на этом наборе. В нашу большую дизъюнкцию войдут значения внутренних конъюнкций. Конъюнкция равна 1 в точности тогда, когда равны 1 все ее сомножители, т.е. в точности при $\alpha_1=\sigma_1,\dots,\ \alpha_k=\sigma_k$. Получим

$$f(\alpha_1,\dots,\alpha_n)=0\vee 0\vee\dots\vee 0\vee f(\alpha_1,\dots,\alpha_k,\alpha_{k+1},\dots,\alpha_n).$$

□

Вместо СДНФ можно использовать СКНФ — совершенную конъюнктивную нормальную форму.

Теорема 4 ((о совершенной дизъюнктивной нормальной форме)). *Любая не тождественно единичная булева функция от n логических переменных допускает представление*

$$f(x_1,\dots,x_n)=\bigwedge_{(\sigma_1,\dots,\sigma_n):\, f(\sigma_1,\dots,\sigma_n)=0}\left(x_1^{1-\sigma_1}\vee x_2^{1-\sigma_2}\vee\dots\vee x_n^{1-\sigma_n}\right).$$

Доказательство. Запишем СДНФ функции $\neg f$ в виде

$$\overline{f(x_1,\dots,x_n)}=\bigvee_{(\sigma_1,\dots,\sigma_n)\in E_2^n} x_1^{\sigma_1}\cdot x_2^{\sigma_2}\cdot\dots\cdot x_n^{\sigma_n}\cdot\overline{f(\sigma_1,\dots,\sigma_n)}.$$

Теперь возьмем отрицание обеих частей

$$f(x_1,\dots,x_n)=\bigwedge_{(\sigma_1,\dots,\sigma_k)\in E_2^k}\left(x_1^{1-\sigma_1}\vee x_2^{1-\sigma_2}\vee\dots\vee x_n^{1-\sigma_n}\vee f(\sigma_1,\dots,\sigma_n)\right)=$$

$$=\bigwedge_{(\sigma_1,\dots,\sigma_n):\, f(\sigma_1,\dots,\sigma_n)=0}\left(x_1^{1-\sigma_1}\vee x_2^{1-\sigma_2}\vee\dots\vee x_n^{1-\sigma_n}\right).$$

□

Упражнение 9. *Разложите в СКНФ штрих Шеффера.*

Лекция 5. Полные системы, замкнутые системы.

Определение 23. *Пусть A — конечное множество булевых функций. Формулой над A называется булева функция, которая представима в виде некоторой суперпозиции функций из множества A.*

Пример 14. *Пусть* $A = \{\vee,\ \wedge,\ \neg\}$. *Любая функция раскладывается в СДНФ — суперпозицию этих связок. Значит, любая булева функция — формула над* A.

Пример 15. *Пусть* $A = \{\oplus\}$. *Заведомо любая формула над* A *не меняется от произвольной перестановки своих переменных.*

Определение 24. *Система* A *называется полной, если любая булева функция является формулой над* A.

Утверждение 11. *Если система* A *полна и любую функцию из* A *является формулой над системой* B, *то* B *тоже полна.*

Теорема 5. *Системы* $\{\vee,\ \neg\}$, $\{\wedge,\ \neg\}$, $\{|\}$, $\{\wedge,\ \oplus,\ 1\}$ *полны.*

Доказательство. Первая система полна потому, что $x \wedge y = \neg((\neg x) \vee (\neg y))$. Вторая система полна потому, что $x \vee y = \neg((\neg x) \wedge (\neg y))$. Третья система полна потому, что $\neg x = x \mid x$ и $x \wedge y = \neg(x \mid y)$. Четвертая система полна потому, что $\neg x = x \oplus 1$. □

Определение 25. *Монотонной конъюнкцией переменных* $x_1,\ x_2 \ldots, x_n$ *называются константа* 1 *и любые произведение вида* $x_{i_1} \cdot x_{i_2} \cdot x_{i_s}$ *составленное из этих переменных.*

Определение 26. *Полиномом Жегалкина переменных* $x_1,\ x_2 \ldots, x_n$ *называются константа* 0 *и произвольные суммы* $K_1 \oplus K_2 \oplus \cdots \oplus K_n$, *составленные из монотонных конъюнкций.*

Пример 16. *Функция* $x \oplus xy \oplus xyz$ *— полином Жегалкина переменных* x, y *и* z, *а функция* $x \oplus \overline{y}$ *полиномом Жегалкина не является.*

Теорема 6. *Любая булева функция представима полиномом Жегалкина. Это представление единственно (с точностью до перестановки конъюнкций и множителей в них).*

Доказательство. Запишем функцию f в системе $\{\wedge,\ \oplus,\ 1\}$. Пользуясь дистрибутивностью, раскроем все скобки. Получим сумму конъюнкций. Каждая конъюнкция составлена из переменных и единиц. Единицы убираем, дублирование переменных убираем. Получаем монотонные конъюнкции. Если теперь среди конъюнкций есть повторения, то преобразуем $A \oplus A = 0$, $A \oplus 0 = A$.

Докажем единственность представления. Подсчитаем, сколько монотонных конъюнкций из n переменных, можно образовать. Для каждой переменной кодируем 0, если она не входит в конъюнкцию и 1 иначе. Константе 1 при этом соответствует набор нулей. Всего получим 2^n наборов. Теперь каждой сумме конъюнкций сопоставим кодировку нулей и единиц — если данная конъюнкция присутствует в сумме, пишем 1, иначе пишем 0. При этом константе 0 соответствует набор нулей. Получаем двоичные набору длины 2^n. Итого, 2^{2^n} различных многочленов Жегалкина. Ровно столько различных функций от n переменных существует. Мы построили отображение многочленов Жегалкина в множество булевых функций, причем оно сюръективно. Тогда оно и биективно. □

Фактически, мы дали алгоритм построения многочлена Жегалкина:
пишем СДНФ $\longrightarrow$ заменяем каждую дизъюнкцию $\vee$ на $\oplus$ (поскольку в СДНФ никакие две конъюнкции не могут быть истинными одновременно, это равносильный переход) $\longrightarrow$ заменяем кажде отрицание $\overline{x}$ на $(1 \oplus x)$ $\longrightarrow$ раскрываем скобки $\longrightarrow$ вычеркиваем одинаковые слагаемые.

Пример 17.

$$f(x,y,z) = \overline{x}yz \vee x\overline{y}z \vee xyz = \overline{x}yz \oplus x\overline{y}z \oplus xyz = (1 \oplus x)yz \oplus x(1 \oplus y)z \oplus xyz = \\ = yz \oplus xyz \oplus xz \oplus xyz \oplus xyz = xz \oplus yz \oplus xyz.$$

Можно дать другой алгоритм построения; здесь исходной информацией является таблица истинности функции:
записываем многочлен в порядке, определяемом строками таблицы, с неизвестными коэффициентами $\longrightarrow$ подставляем последовательно наборы переменных, проходя таблицу сверху вниз; на каждом шаге определяем один из коэффициентов.

Пример 18. *Запишем таблицу истинности функции из предыдущего примера:*

x	y	z	$f(x,y,z)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Запишем многочлен с неизвестными кэффициентами:

$$f(x,y,z) = c_0 \oplus c_1 z \oplus c_2 y \oplus c_3 yz \oplus c_4 x \oplus c_5 xz \oplus c_6 xy \oplus c_7 xyz.$$

Пройдем таблицу сверху вниз:

$$f(0,0,0) = c_0 = 0, \qquad f(0,0,1) = c_0 \oplus c_1 = 0 \Longrightarrow c_1 = 0,$$
$$f(0,1,0) = c_0 \oplus c_2 = 0 \Longrightarrow c_2 = 0, \qquad f(0,1,1) = c_0 \oplus c_1 \oplus c_2 \oplus c_3 = 1 \Longrightarrow c_3 = 1,$$
$$f(1,0,0) = c_0 \oplus c_4 = 0 \Longrightarrow c_4 = 0 \qquad f(1,0,1) = c_0 \oplus c_1 \oplus c_4 \oplus c_5 = 1 \Longrightarrow c_5 = 1,$$
$$f(1,1,0) = c_0 \oplus c_2 \oplus c_4 \oplus c_6 = 0 \Longrightarrow c_6 = 0,$$
$$f(1,1,1) = c_0 \oplus c_1 \oplus c_2 \oplus c_3 \oplus c_4 \oplus c_5 \oplus c_6 \oplus c_7 = 1 \Longrightarrow 1 \oplus 1 \oplus c_7 = 1 \Longrightarrow c_7 = 1.$$

Получим $f(x,y,z) = yz \oplus xz \oplus xyz$.

Определение 27. *Класс булевых функций от* n *переменных будем обозначать* $P_2(n)$. *Класс всех булевых функций обозначим* P_2.

Если $A \subset P_2$, *то множество всех формул над* A *называем замыканием системы* A *и обозначаем* $[A]$.

Пример 19. *Если* $A = \{\vee,\ \wedge,\ \neg\}$, *то* $[A] = P_2$.

Сформулируем очевидные свойства замыкания

1. $A \subset [A]$;

2. $[[A]] = [A]$;

3. если $A \subset B$, то $[A] \subset [B]$ (при этом строгое вложение может стать равенством);

4. система A полна тогда и только тогда, когда $[A] = P_2$.

Определение 28. *Система A называется замкнутой, если $[A] = A$.*

Утверждение 12. *Пусть система A замкнута и $A \neq P_2$, а система $B \subset A$. Тогда $[B] \neq P_2$.*

Доказательство. Так как $B \subset A$, то $[B] \subset [A] \neq P_2$. □

Утверждение 13. *Подмножество неполной системы также неполно.*

Доказательство. Если $B \subset A$ и $[A] \neq P_2$, то $[B] \subset [A] \neq P_2$. □

Разберем несколько замкнутых классов

Определение 29. *Обозначим T_0 класс всех функций, равных 0 на наборе $(0, \ldots, 0)$.*

Пример 20. *Функции $\vee$ $\oplus$ и $\wedge$ принадлежат классу T_0, а функция $\neg$ — нет.*

Любая функция из $T_0(n)$ задается своими значениями на 2^{n-1} наборе. То есть, таких функций ровно $2^{2^n-1} = \frac{1}{2}2^{2^n}$.

Теорема 7. *Класс T_0 замкнут.*

Доказательство. Пусть функции $f_1,\ f_2, \ldots, f_n$ принадлежат классу T_0, а мы берем от них функцию $f \in T_0$. Тогда композиция $f(f_1, f_2, \ldots, f_n)$ на нулевом наборе равна $f(0, \ldots, 0) = 0$. □

По аналогии с T_0 вводится класс T_1 функций, равных 1 на наборе $(1, \ldots, 1)$.

Теорема 8. *Класс T_1 замкнут.*

Доказательство. Пусть функции $f_1,\ f_2, \ldots, f_n$ принадлежат классу T_1, а мы берем от них функцию $f \in T_1$. Тогда композиция $f(f_1, f_2, \ldots, f_n)$ на наборе $(1, \ldots, 1)$ равна $f(1, \ldots, 1) = 1$. □

Определение 30. *Назовем функцию $f(x_1, \ldots, x_n)$ линейной, если ее многочлен Жегалкина имеет вид*

$$f(x_1, \ldots, x_n) = c_0 \oplus c_1 x_1 \oplus \ldots \oplus c_n x_n, \quad c_j \in \{0, 1\}.$$

Класс линейных функций обозначим L.

Теорема 9. *Класс L замкнут.*

Доказательство. Пусть функции $f_1,\ f_2, \ldots, f_n$ принадлежат классу L, а мы берем от них функцию $f \in L$. Подставляем суммы для функций $f_1, \ldots, f_n$ в сумму для f, раскрываем скобки, убираем повторяющиеся слагаемые, получаем линейную функцию. □

Упражнение 10. *Найдите $|L|$ (количество различных линейных функций от n переменных).*

Определение 31. *Введем на двоичных наборах E_2^n отношение порядка*

$$(\alpha_1, \alpha_2, \ldots, \alpha_n) \le (\beta_1, \beta_2, \ldots, \beta_n) \iff \alpha_j \le \beta_j \ \forall j = 1, \ldots, n.$$

Функция $f(x_1, \ldots, x_n)$ называется монотонной, если

$$(\alpha_1, \ldots, \alpha_n) \le (\beta_1, \ldots, \beta_n) \implies f(\alpha_1, \ldots, \alpha_n) \le f(\beta_1, \ldots, \beta_n).$$

Класс всех монотонных функций обозначаем M.

Пример 21. *Функции $\vee$ и $\wedge$ монотонны, а функции $|$, $\downarrow$, $\oplus$ и $\sim$ нет.*

Теорема 10. *Класс M замкнут.*

Доказательство. Пусть $f,\ f_1, \ldots,\ f_n$ — монотонные булевы функции $n,\ k_1, \ldots,\ k_n$ переменных соответственно. Нам надо проверить монотонность функции

$$g(x_1, \ldots, x_m) = f(f_1(x_1, \ldots, x_{k_1}), \ldots, f_n(x_1, \ldots, x_{k_n})).$$

Возьмем два двоичных набора $\alpha = (\alpha_1, \ldots, \alpha_m) \le \beta = (\beta_1, \ldots, \beta_m)$ и подставим их в функцию g. Будем считать, что все функции $f_1, \ldots, f_n$ зависят от всех переменных $x_1, \ldots, x_m$, добавляя новые фиктивные переменные по необходимости. Вычислим вначале $f_k(\alpha_1, \ldots, \alpha_m)$ — обозначим эти числа γ_k. Аналогично, положим $f_k(\beta_1, \ldots, \beta_m) = \delta_k$. Из монотонности функций f_k имеем $\gamma_k \le \delta_k$. Тогда

$$g(\alpha_1, \ldots, \alpha_m) = f(\gamma_1, \ldots, \gamma_n) \le f(\delta_1, \ldots, \delta_n) = g(\beta_1, \ldots, \beta_m).$$

□

Определение 32. *Двойственной функцией к функции $f(x_1, \ldots, x_n)$ называется*

$$f^*(x_1, \ldots, x_n) := \overline{f(\overline{x_1}, \ldots, \overline{x_n})}.$$

Сразу из определения следует, что $(f^*)^* = f$.

Лекция 6. Теорема Поста.

Теорема 11 (принцип двойственности)**.** *Двойственная функция к суперпозиции есть суперпозиция двойственных функций. Более подробно: пусть f_1 зависит от переменных $x_{11}, \ldots, x_{1n_1}$, функция f_2 зависит от переменных $x_{21}, \ldots, x_{2n_2}$, и т.д., функция f_k зависит от переменных $x_{2k}, \ldots, x_{2n_k}$, а функция f зависит от k переменных. Занумеруем все эти переменные $x_{11}, \ldots, x_{kn_k}$ единым образом $x_1, \ldots, x_m$ и положим*

$$\Phi(x_1, \ldots, x_m) = f(f_1(x_{11}, \ldots, x_{1n_1}), f_2(x_{21}, \ldots, x_{2n_2}), \ldots, f_k(x_{k1}, \ldots, x_{kn_k})).$$

Тогда

$$\Phi^*(x_1, \ldots, x_m) = f^*(f_1^*(x_{11}, \ldots, x_{1n_1}), f_2^*(x_{21}, \ldots, x_{2n_2}), \ldots, f_k^*(x_{k1}, \ldots, x_{kn_k})).$$

Доказательство.

$$\Phi^*(x_1,\dots,x_m) = \overline{f}(f_1(\overline{x_{11}},\dots,\overline{x_{1n_1}}),\dots,f_k(\overline{x_{k1}},\dots,\overline{x_{kn_k}})) =$$
$$= \overline{f}(\overline{f_1^*}(x_{11},\dots,x_{1n_1}),\dots,\overline{f_k^*}(x_{k1},\dots,x_{kn_k})) = f^*(f_1^*(x_{11},\dots,x_{1n_1}),\dots,f_k^*(x_{k1},\dots,x_{kn_k}).$$

□

Следствие 1. *Если f — формула над системой $A = \{f_1,\dots,f_k\}$, то f^* — формула над системой $B = \{f_1^*,\dots,f_n^*\}$.*

Определение 33. *Функция f назвыается самодвойственной, если $f^* = f$.*

Класс самодвойственных функций обозначаем S.

Пример 22. *Проверим, что функция $f(x,y,z) = xy \vee xz \vee yz$ принадлежит классу S. Действительно,*

$$f^*(x,y,z) = \overline{f}(\overline{x},\overline{y},\overline{z}) = \overline{\overline{x}\cdot\overline{y} \vee \overline{x}\cdot\overline{z} \vee \overline{y}\cdot\overline{z}} =$$
$$= (x\vee y)(x\vee z)(y\vee z) = (x\vee xy\vee xz\vee yz)(y\vee z) = (x\vee yz)(y\vee z) = xy\vee yz\vee xz\vee yz = f(x,y,z).$$

Теорема 12. *Класс S замкнут.*

Доказательство. Двойственная функция к суперпозиции есть суперпозиция двойственных функций. Если все участники суперпозиции самодвойственны, то двойственная функция к суперпозиции есть суперпозиция исходных функций, т.е. равна самой себе. □

Мы рассмотрели пять важнейших замкнутых классов. Ясно, что составля из них произвольные пересечения, мы получим новые замкнутые классы.

Лемма 2 (о несамодвойственной функции)**.** *Пусть $f(x_1,\dots,\ x_n) \notin S$. Тогда, подставляя вместо некоторых переменных x, а вместо других $\overline{x}$, можно добиться того, что получится постоянная функция.*

Пример 23. *Пусть $f(x,y) = x\vee y$. Тогда $f(x,\overline{x}) \equiv 1$. Пусть $g(x,y) = xy$. Тогда $g(x,\overline{x}) \equiv 0$.*

Доказательство. $f \notin S$, т.е. $\overline{f}(\overline{x_1},\dots,\ \overline{x_n}) \neq f(x_1,\dots,\ x_n)$, т.е. эти функции отличаются на некотором наборе $\alpha = (\alpha_1,\dots,\ \alpha_n)$. Заметим, что

$$\overline{f}(\overline{\alpha_1},\dots,\ \overline{\alpha_n}) \neq f(\alpha_1,\dots,\ \alpha_n) \Longleftrightarrow f(\overline{\alpha_1},\dots,\ \overline{\alpha_n}) = f(\alpha_1,\dots,\alpha_n).$$

Положим теперь $g(x) := f(x\oplus\alpha_1,\dots,\ x\oplus\alpha_n)$. Тогда

$$g(0) = f(\alpha_1,\dots,\ \alpha_n) = f(\overline{\alpha_1},\dots,\ \overline{\alpha_n}) = g(1).$$

Теперь заметим, что $x\oplus\alpha_j = x$, если $\alpha_j = 0$, и $= \overline{x}$, если $\alpha_j = 1$. Это означает, что функция g получается из f при подстановке x вместо некоторых переменных и $\overline{x}$ вместо остальных. □

Лемма 3 (о немонотонной функции)**.** *Пусть $f(x_1,\dots,\ x_n) \notin M$. Тогда, подставляя вместо некоторых переменных x, вместо некоторых других 0, а вместо остальных 1, можно добиться того, что получится функция $g(x) = \overline{x}$.*

Пример 24. *Пусть* $f(x,y) = x \mid y$. *Тогда* $f(x,1) = \overline{x}$.

Доказательство. Раз $f \notin M$, то найдутся наборы $(\alpha_1, \ldots, \alpha_n)$ и $(\beta_1, \ldots, \beta_n)$ такие, что $\alpha_j \le \beta_j$ для всех j, однако $f(\alpha) > f(\beta)$ (т.е. $f(\alpha) = 1$, а $f(\beta) = 0$). Выделим те индексы $i_1 < i_2 < \cdots < i_s$, для которых $\alpha_{i_j} \ne \beta_{i_j}$ (очевидно, $\alpha_{i_j} = 0$, а $\beta_{i_j} = 1$). Будем по очереди менять переменные α_{i_j} с нулей на единицы (вначале поменяем α_{i_1}, потом α_{i_2} и т.д.). При этом будем следить за значением функции f и дождемся того момента, когда 1 сменится на ноль (таких моментов может быть несколько — мы, для определенности, возьмем первый из них). Итак, мы построили два набора $(\gamma_1, \ldots, \gamma_n)$ и $(\delta_1, \ldots, \delta_n)$, отличающихся одной координатой (с номером, скажем i), причем $\gamma_i = 0$, а $\delta_i = 1$, $f(\gamma) = 1$, $f(\delta) = 0$. Подставим теперь вместо этой координаты x, а остальные зафиксируем равными $\gamma_j = \delta_j$ (т.е. подставим вместо этих переменных нули и единицы). Получим функцию $g(x)$, для которой $g(0) = 1$, а $g(1) = 0$. □

Лемма 4 (о нелинейной функции)**.** *Пусть* $f(x_1, \ldots, x_n) \notin L$. *Тогда, подставляя вместо переменных выражения* x, $\overline{x}$, y, $\overline{y}$, 0 *и* 1 *можно получить либо функцию* $g(x,y) = xy$, *либо функцию* $g(x,y) = \neg(xy)$.

Доказательство. Функция f нелинейна, а значит ее полином Жегалкина содержит множитель $x_j x_k \cdot K$, где K — какая-то ненулевая монотонная конъюнкция остальных переменных. Составим функцию $h(x,y)$, подставив x вместо x_k, y вместо x_k и константы 0 и/или 1 вместо остальных переменных так, чтобы конъюнкция K обратилась в 1. Тогда многочлен Жегалкина примет вид $xy \oplus x \cdot b \oplus y \cdot c \oplus d$ с некоторыми b, c, $d \in \{0, 1\}$. А вот теперь рассмотрим функцию $g(x,y) = h(x \oplus c, y \oplus b)$. Раскроем скобки

$$g(x,y) = xy \oplus bx \oplus cy \oplus bc \oplus bx \oplus bc \oplus cy \oplus bc \oplus d = xy \oplus d.$$

В зависимости от d это либо функция xy, либо функция $xy \oplus 1 = \neg(xy)$. □

Мы готовы доказать теорему Поста о полноте.

Теорема 13. *Система функций* A *является полной в* P_2 *тогда и только тогда, когда* A *не вкладывается ни в один из классов* T_0, T_1, L, M *и* S.

Замечание 1. *Иными словами, среди функций системы* A *должна быть функция* $f \notin T_0$, *функция* $g \notin T_1$ *и т.д. При этом не требуется, чтобы эти функции были разными. Например, штрих Шеффера не лежит ни в одном из классов* T_0, T_1, L, M *и* S. *И действительно, система* $A = \{|\}$ *полна.*

Доказательство. Покажем, что с помощью суперпозиций функций системы A можно получить отрицание и конъюнкцию. Это будет означать полноту системы.
Шаг 1. Получим отрицание. Мы знаем, что в A есть функция $f_0 \notin T_0$. Положим $g(x) = f_0(x, \ldots, x)$. Поскольку $g(0) = 1$, то возможны два случая: либо $g(x) = \neg x$ (и мы завершили Шаг 1), либо $g(x) = 1$ (тогда мы получили константу $1 \in [A]$). Мы знаем, что в A есть функция $f_1 \notin T_1$. Положим $h(x) = f_1(x, \ldots, x)$. Поскольку $h(1) = 0$, то возможны два случая: либо $h(x) = \neg x$ (и мы завершили Шаг 1), либо $h(x) = 0$ (тогда мы получили константу $0 \in [A]$). Предположим, что нам «не повезло» и функцию $\neg x$ мы не

получили. Тогда в $[A]$ входят константы 0 и 1. Мы знаем, что в A есть функция $f_2 \notin M$. По лемме о немонотонной функции, подставляя в нее 0 и 1, мы получим отрицание.
Шаг 2. Получим константы 0 *и* 1. Итак, у нас есть функция $\neg \in [A]$. Возможно, что на первом шаге мы также получили функции 0 и/или 1. Предположим, нам «не повезло», и мы получили только функцию $\neg$. Поскольку композиция отрицания с самим собой дает тождественную функцию $I(x) = x$, то $I \in [A]$. Мы знаем, что в A есть функция $f_3 \notin S$. По лемме о несамодвойственной функции, подставляя в нее отрицание и тождественную функцию, можем получить одну из констант 0 или 1. Теперь берем композицию функции $\neg$ и этой константы — получаем вторую константу.
Шаг 3. Получим конъюнкцию. Мы знаем, что в A есть функция $f_4 \notin L$. По лемме о нелинейной функции, можно составить такую композицию этой функции с функциями $\neg$, I, 0 и 1 (все они уже получены), что получится либо конъюнкция, либо антиконъюнкция. В последнем случае возьмем еще отрицание от этой функции.
Завершение доказательства. Итак, система $B = \{\neg, \wedge\}$ содержится в $[A]$. Но $[B] = P_2$. Поскольку $[B] \subset [[A]] = [A]$ (см. свойства замыкания), то $[A] = P_2$. □

Пример 25. *Функция* $|$ *не лежит в* T_0, *т.к.* $0 \mid 0 = 1$. *Функция* $|$ *не лежит в* T_1, *т.к.* $1 \mid 1 = 0$. *Функция* $|$ *не лежит в* L, *т.к.* $x \mid y = 1 \oplus xy$. *Функция* $|$ *не лежит в* M, *т.к.* $0 \mid 0 = 1$, *а* $1 \mid 1 = 0$. *Функция* $|$ *не лежит в* S, *т.к.* $\overline{x} \mid \overline{y} = xy$. *Выполнены все условия теоремы Поста. И, действительно, система* $\{|\}$ *полна.*

Пример 26. *Система* $\{\vee, \wedge\}$ *не полна уже потому, что обе эти функции лежат в* T_0.

Лекция 7. Базисы. Предполные системы.

Определение 34. *Система функций* A *называется базисом в* P_2, *если*
1) $[A] = P_2$ *(свойство полноты);*
2) $\forall f \in A$ *система* $A \setminus \{f\}$ *не полна (свойство минимальности).*

Утверждение 14. *Условие минимальности можно переформулировать так: ни одна функция из системы не выражается в виде суперпозиции других функций системы.*

Доказательство. Если функция $f \in A$ выражается через другие функции системы, то система $A \setminus \{f\}$ остается полной, т.е. A не минимальна. Если наоборот, есть функция $f \in A$ такая, что система $A \setminus \{f\}$ полна, то функции этой системы выражают, в частности, функцию f. □

Пример 27. *Система* $\{|\}$, *естественно, базис. Система* $\{\neg, \vee, \wedge\}$ *не базис, потому, что* $x \vee y = \neg((\neg x) \wedge (\neg y))$. *Система* $\{\neg, \wedge\}$ *— базис. Действительно, если удалить отрицание, то полнота пропадет (даже система* $\{\vee, \wedge\}$ *не полна). Если удалить конъюнкцию, то полнота вновь пропадет (хотя бы потому, что функция* $\neg$ *самодвойственна).*

Мы знаем из курса аналитической геометрии, что на плоскости любой базис состоит из двух векторов; в пространстве — всегда из трех. Что можно сказать о количестве функций в базисах алгебры логики? Базис $\{|\}$ состоит из одной функции. Базис $\{\neg, \wedge\}$ — из двух функций.

Упражнение 11. *Докажите, что система* $\{1,\ \wedge,\ \oplus\}$ *— базис.*

Упражнение 12. *Докажите, что система* $\{0,\ 1,\ \wedge,\ f\}$*, где* $f(x,y,z) = x \oplus y \oplus z$*, полна.*

Замечание 2. *Система* $\{0,\ 1,\ \wedge,\ f\}$ *минимальна и, таким образом, является базисом.*

Доказательство. Удалим из системы функцию f. Получим $\{0,\ 1,\ \wedge\} \subset M$ (базисом не является). Удалим из системы функцию $\{\wedge\}$. Получим $\{0,\ 1,\ f\} \subset L$ (базисом не является). Удалим из системы функцию $\{1\}$. Получим $\{0,\ \wedge,\ f\} \subset T_0$ (базисом не является). Удалим из системы функцию $\{0\}$. Получим $\{1,\ \wedge,\ f\} \subset T_1$ (базисом не является). □

Теорема 14. *Любой базис содержит не более четырех функций.*

Доказательство. Пусть A — полная система. Тогда (по теореме Поста) найдутся функции $f_0 \in A$, $f_0 \notin T_0$; $f_1 \in A$, $f_1 \notin T_1$; $f_M \in A$, $f_M \notin M$; $f_S \in A$, $f_S \notin S$ и $f_L \in A$, $f_L \notin L$. По теореме Поста, система $\{f_0,\ f_1,\ f_M,\ f_S,\ f_L\}$ полна. Возможны два случая. Если $f_0(1,\ldots,1) = 1$, то $f_0 \notin S$ и (опять по теореме Поста) система $\{f_0,\ f_1,\ f_M,\ f_L\}$ полна. Если $f_0(1,\ldots,1) = 0$, то $f_0 \notin M$ и полной оказывается система $\{f_0,\ f_1,\ f_S,\ f_L\}$. В любом случае, мы доказали, что из любой полной системы можно выделить полную подсистему из не более, чем четырех функций. Значит, никакой базис не может содержать пяти функций. □

Мы уже проводили аналогию с базисами в трехмерном пространстве. Продолжим ее. Заметим, что если A — плоскость в $\mathbb{R}^3$, то добавляя к A любой вектор, не лежащий в A, т.е. переходя к системе $A \cup \{x\}$, получим полную линейную систему (она содержит базис — два любых неколлинеарных вектора из A и вектор x).

Определение 35. *Система* $A \subset P_2$ *называется предполной, если* A *не полна в* P_2 *(т.е.* $[A] \neq P_2$*), но* $\forall f \notin A$ *система* $A \cup \{f\}$ *уже полна в* P_2*.*

Теорема 15. *В* P_2 *есть ровно пять предполных систем:* T_0*,* T_1*,* M*,* L *и* S*.*

Лемма 5. *Ни один из классов* T_0*,* T_1*,* L*,* M*,* S*, не содержится в другом.*

Доказательство. Надо предъявить функции, показывающие различие классов. Например, $0 \in T_0 \setminus S$, а $\neg \in S \setminus T_0$. И так для всех остальных пар. Составим итоговый список

$$
\begin{array}{ll}
0 \in T_0 \setminus T_1, \quad 1 \in T_1 \setminus T_0; & \wedge \in T_0 \setminus L, \quad 1 \in L \setminus T_0; \\
\oplus \in T_0 \setminus M, \quad 1 \in M \setminus T_0; & 0 \in T_0 \setminus S, \quad \neg \in S \setminus T_0; \\
\wedge \in T_1 \setminus L, \quad 0 \in L \setminus T_1; & \sim \in T_1 \setminus M, \quad 0 \in M \setminus T_1; \\
1 \in T_1 \setminus S, \quad \neg \in S \setminus T_1; & \neg \in L \setminus M, \quad \wedge \in M \setminus L; \\
\neg \in L \setminus S, \quad f \in S \setminus L; & 0 \in M \setminus S, \quad \neg \in S \setminus M.
\end{array}
$$

□

Упражнение 13. *Подберите функцию* f *для предыдущего утверждения.*

Доказательство теоремы **??**. Докажем, что каждый из классов T_0, T_1, L, M, S является предполным. Пусть N один из этих пяти классов. Рассмотрим систему $N \cup \{f\}$, где $f \notin N$. Эта система не лежит в N, т.к. $f \notin N$. Эта система не лежит ни в одном из четырех других классов (скажем N'), т.к., согласно лемме, найдется функция $g \in N \setminus N'$. Тогда по теореме Поста система $N \cup \{f\}$ полна.

Докажем обратное утверждение. Если A предполная система, то, прежде всего, $[A] \neq P_2$. Тогда по теореме Поста, найдется такой класс N из наших пяти классов T_0, T_1, M, L, S, что $A \subset N$. Если $A = N$, то все доказано. Если $A \subsetneq N$, то возьмем функцию $f \in N \setminus A$. Тогда $[A \cup \{f\}] \subset [N] = N \subsetneq P_2$ и система $A \cup \{f\}$ не полна, а значит система A не предполная (противоречие). □

На последок сформулируем без доказателсьтва два результата Поста.

Теорема 16. *В P_2 существует ровно счетное число различных замкнутых классов.*

Определение 36. *Система функций A называется базисом в класее $B \subset P_2$, если*
1) $[A] = B$ (свойство полноты);
2) $\forall f \in A$ система $A \setminus \{f\}$ свойством 1) уже не обладает, т.е. $[A \setminus f] \subsetneq B$ (свойство минимальности).

Теорема 17. *В любом замкнутом классе существует конечный базис.*

Пример 28. *Найдем конечный базис в классе T_0. Заметим, что многочлен Жегалкина любой функции из T_0 не содержит слагаемого* 1. *Это означает, что любую функцию из T_0 можно записать суперпозицией связок $\oplus$ и $\wedge$. Мы доказали, что система $A = \{\oplus, \wedge\}$ полна в T_0. Докажем, что она базис в T_0. Удалим из системы первую связку. Но функция $\wedge$ монотонна. Значит ее суперпозиции тоже всегда монотонны и функцию $\oplus$ мы при таких суперпозициях не получим. Удалим из A вторую связку. Но функция $\oplus$ линейна. Значит ее суперпозиции тоже всегда линейны и функцию $\wedge$ мы при таких суперпозициях не получим.*

Упражнение 14. *В каждом из четырех классов T_1, M, L и S найдите конечный базис.*

Упражнение 15. *Докажите, что в P_2 существует ровно счетное число различных базисов.*

Лекция 8. Графы. Изоморфзим. Пути в графах.

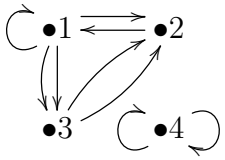
Определение 37. *Графом называют пару $G = (V, E)$, где V — произвольное множество ("vertices" — вершины графа), а E — некоторое семейство пар из V ("edges" — ребра графа).*

Мы будем рассматривать только конечные графы, т.е. считать, что множества V и E конечны.

Пример 29. *Пусть*

$$V = \{1,\, 2,\, 3\ 4\}, \quad E = \{(1,1),\, (1,2),\, (1,3),\, (1,3),\, (2,1),\, (3,2),\, (3,2),\, (4,4),\, (4,4)\}.$$

Изобразим это так



Итак, каждое ребро e — это пара $e = (v_i, v_j)$, $v_i,\, v_j \in V$.

Определение 38. *Если пары, составляющие множество E упорядочены (как в нашем примере), то ребра обладают ориентацией и граф называют ориентированным (или орграфом). Для краткости его (ориентированные) ребра называют дугами. Если же пары неупорядочены, то граф называют неориентированным.*

Естественно, имея орграф, можно перейти к неориентированному графу. Надо просто «забыть» про порядок в парах, т.е. считать, что пары (v_i, v_j) и (v_j, v_i) одинаковы.

Определение 39. *Пара (v, v) называется петлей. Если пара (v_i, v_j) встречается в семействе E несколько раз, то такое ребро (или дугу) называют кратным.*

Договоримся далее назвыать неориентированный граф без кратных ребер и без петель *простым графом*. Ориентированный граф без кратных ребер и без петель — *простым орграфом*. Неориентрованный граф без петель (но, возможно, с кратными ребрам) — *мультиграфом*. Ориентированный граф без петель — *ориентированным мультиграфом*. Если в мультиграфе разрешены петли, то такой объект будем называть *псевдографом* или *оиентированным псевдографом*. Таким образом, граф из примера 29 — это ориентированный псевдограф.

В определении графа мы назвали E «семейством» потому, что в зависимости от типа графа, E — различный объект. Если V — простой граф, то E — множество неупорядоченных пар элементов V, не содержащее пар вида (v, v). Если V — простой орграф, то E — множество упорядоченных пар элементов V, не содержащее пар вида (v, v). Если V — ориентированный псевдограф, то E — неупорядоченная выборка с повторениями из множества V^2. И т.д.

Определение 40. *Две вершины неориентированного псевдографа, которые соединены ребром, называют смежными. Если вершина такого графа лежит на каком-либо ребре, то эти вершину и ребро называют инцидентными.*

Определение 41. *Степенью вершины v неориенированного псевдографа G называют число ребер, инцидентных v и обозначают $\deg v$. При этом петли учитывают дважды.*

Утверждение 15. *Для любого неориентрованного псевдографа G с p вершинами и q ребрами верно соотношение $\sum_{i=1}^{p} \deg v_i = 2q$*

Доказательство. Складывая степени вершин, мы посчитаем все ребра графа, причем каждое ребро будет посчитано дважды (в своем «левом» и в «правом» конце). Петли также учитываются дважды (таково правило подсчета степени вершины). □

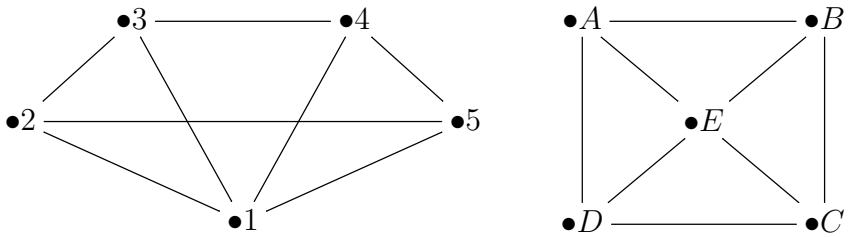
Определение 42. *Пусть все вершины графа занумерованы $V = \{v_1, \ldots, v_p\}$. Каждому ориентированному псевдографу с p вершинами сопоставим матрицу смежности $A = (a_{ij})_1^p$ по правилу a_{ij} есть количество ребер вида (v_i, v_j).*

В частности, если из v_i в v_j не ведет ни одного ребра, то $a_{ij} = 0$. Если $i = j$, то a_{ij} — число петель в вершине v_i. Например, матрица смежности для графа G из примера 29 равна

$$A = \begin{pmatrix} 1 & 1 & 2 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}.$$

Один и тот же граф можно изобразить на рисунке сильно по-разному.

Пример 30. *Два рисунка*



изображают один и тот же граф. Надо просто на первом рисунке передвинуть вершину 1 внутрь трапеции "2345", а потом перенумеровать вершины $1 \mapsto E$, $2 \mapsto D$, $3 \mapsto A$, $4 \mapsto B$ и $5 \mapsto C$.

Определение 43. *Два простых орграфа $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$ называются изоморфными, если существует такая биекция $\varphi : V_1 \to V_2$ (перенумерация вершин), что для любых $u,\ v \in V_1$: $(u, v) \in E_1 \iff (\varphi(u), \varphi(v)) \in E_2$.*

В общем случае определение такое.

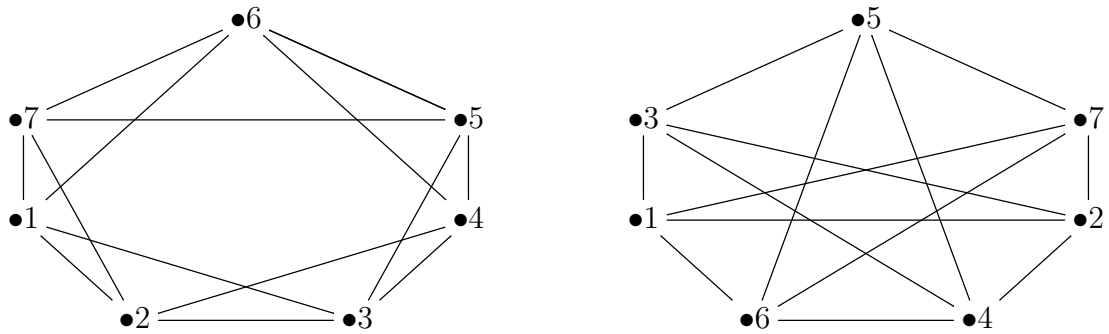
Определение 44. *Два графа $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$ называются изоморфными, если существует такая биекция $\varphi_1 : V_1 \to V_2$ (перенумерация вершин) и биекция $\varphi_2 : E_1 \to E_2$ (перенумерация ребер), что для любых $u,\ v \in V_1$: $\varphi_2((u, v)) = (\varphi_1(u), \varphi_1(v))$.*

Вот несколько простых утверждений.

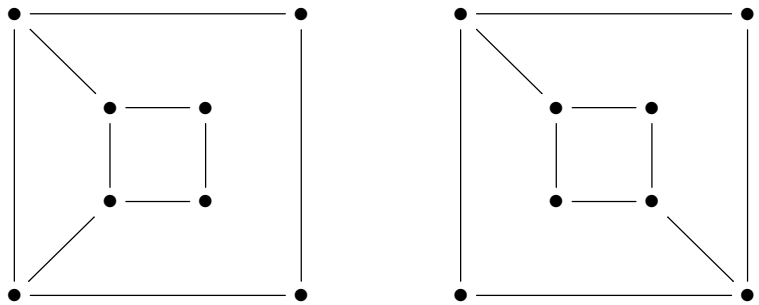
Утверждение 16. *Пусть* $G_1 = (V_1, E_1)\tilde{G}_2 = (V_2, E_2)$.
1). Если пара отображений φ_1, φ_2 *осуществляет изоморфзим графа* G_1 *на граф* G_2, *то пара обратных отображений* φ_1^{-1}, φ_2^{-1} *осуществляет изоморфзим графа* G_2 *на граф* G_1.
2). Изоморфизм графов — это отношение эквивалентности на множестве графов.
3). Изоморфные графы имеют одинаковое число вершин и одинаковое число ребер.
4). Вершины $u, v \in V_1$ *смежны тогла и только тогда, когда смежны их образы* $\varphi_1(u)$, $\varphi_1(v)$.
5). Вершина $u \in V_1$ *и ребро* $e \in E_1$ *инцидентны тогда и только тогда, когда инцидентны их образы* $\varphi_1(u)$ *и* $\varphi_2(e)$.
6). Степени вершин $v \in V_1$ *и* $\varphi_1(v) \in V_2$ *равны.*
7). Последовательность $\{\deg(v_1), \ldots, \deg(v_n)\}$ *отличается от последовательности* $\{\deg(\varphi_1(v_1), \ldots, \varphi_1(v_n)\}$ *лишь перестановкой.*

Совпадение числа вершин, числа ребер, последовательностей степеней вершин — всего этого может быть недостаточно для изоморфзима графов.

Пример 31. *Графы* G_1 *и* G_2 *на рисунке ниже имеют по* 7 *вершин, по* 14 *ребер и степени всех вершин равны* 4. *При этом они изоморфны.*



А графы G_3 *и* G_4 *на рисунке ниже имеют по* 8 *вершин, по* 10 *ребер и степени вершин равны* $\{3, 3, 3, 3, 2, 2, 2, 2\}$. *При этом они не изоморфны (у* G_3 *три цикла длины* 4, *а у* G_4 *— только два).*



Определение 45. *Граф* G_1 *называется подграфом* G_2, *если* $V_1 \subset V_2$ *и* $E_1 \subset E_2$.

Определение 46. *Путь в графе — это произвольная последовательность*

$$v_0, (v_0, v_1), v_1, (v_1, v_2), \ldots, v_{n-1}, (v_{n-1}, v_n), v_n.$$

Число n *здесь — это длина пути.*

Определение 47. *Путь называется цепью, если все его ребра различны. Цепь называется простой, если вершины тоже не повторяются*

Определение 48. *Путь называется замкнутым, если $v_0 = v_n$. Замкнутый путь называется циклом, если все его ребра различны. Цикл называется простым, если его вершины тоже различны (за исключением $v_0 = v_n$).*

Пример 32. *Путь* $1, (1,2), 2, (2,3), 3, (3,5), 5, (5,7), 7$ *— простая цепь. Путь* $1, (1,2), 2, (2,7), 7, (7,1), (1,6), 6$ *— цепь, но не простая. Путь* $1, (1,2), 2, (2,7), 7, (7,1), 1$ *— простой цикл.*

Конечно, если граф простой (нет кратных ребер и петель), то можно при записи пути не писать ребра, указывая только вершины.

Утверждение 17. *Пусть C — произвольный путь из вершины v_0 в вершину $v_1 \neq v_0$. Тогда из него можно выделить подпуть из v_0 в v_1, который является простой цепью.*

Доказательство. Пусть вершина v повторяется на пути C, т.е. $v_0 C v_1 = v_0 C_1 v C_2 v C_3 v_1$. Удалим часть пути — рассмотрим путь $v_0 C_1 v C_3 v_1$. Вершина v теперь не повторяется (точнее, если и повторяется, то меньшее число раз). Продолжим этот процесс и выделим подпуть, вершины которого не повторяются вовсе. □

Утверждение 18. *Пусть C — произвольный замкнутый путь из вершины v_0 в вершину v_0. Тогда из него можно выделить подпуть из v_0 в v_0, который является простым циклом.*

Определение 49. *Граф называется связным, если для любых двух его вершин v_0 и v_1 найдется путь с началом в v_0 и концом в v_1.*

Определение 50. *Граф называется эйлеровым, если в нем найдется эйлеров цикл — цикл, содержащий все ребра графа.*

Пример 33. *Граф G_1 эйлеров, так как* $1, 2, 3, 4, 5, 6, 7, 1, 3, 5, 7, 2, 4, 6, 1$ *— эйлеров цикл.*

Теорема 18 (Л.Эйлер, 1736 г.)**.** *Граф является эйлеровым тогда и только тогда, когда он связен и степени всех его вершин четны.*

Доказательство. Докажем необходимость. Пусть $v_0 C v_0$ — эйлеров путь. Пройдем по нему. Каждую вершину мы пройдем хотя бы раз. Значит, граф связен. Пусть вершину $v \neq v_0$ мы прошли k раз. Тогда мы прошли $2k$ ребер, инцидентных v. Значит, степень вершины четна. Если $v = v_0$, то ее степень равна $2k + 2$ (один раз вышли, k раз прошли, один раз вернулись).

Докажем достаточность.
Шаг 1. Зафиксируем произвольную вершину v_0. Она обязательно соединена с какой-то вершиной v_1 (граф связен). Пройдем ребро (v_0, v_1) и сотрем его (далее все время будем стирать ребро, пройдя по нему — чтобы нельзя было пройти его повторно). Теперь

степень вершин v_0 и v_1 нечетна. Пройдем все петли вершины v_1 (степень v_1 останется нечетной). Сделаем еще один шаг и сотрем еще одно ребро. Теперь степень вершины v_1 четна. Продолжим этот процесс, пока не зайдем в тупик — не придем в вершину X, из которой выйти не сможем (все ребра, ведущие из этой вершины стерты). Заметим, что в момент выхода из произвольной вершины на нашем пути, степени всех вершин оставались четными, кроме степени вершины v_0. Значит, $X = v_0$. Обозначим полученный цикл C_1.

Шаг 2. Посмотрим, прошли ли мы все ребра. Если да, то процесс закончен. Если нет, то есть непройденные ребра. Возьмем одно из них. Зафиксируем один его конец — вершину v_2. Если v_2 лежит на пути C_1, то назовем эту точку новой точкой старта. Если v_2 не лежит на C_1, то свяжем v_2 с v_0 (граф связен). Проходя путь от v_2 до v_0 мы обязательно в какой-то момент впервые пройдем вершину, лежащую на пути C_1. И это не v_0 (все ребра, выходящие из v_0 уже включены в путь C_1). Назовем эту вершину v_3 точкой нового старта. Удалим из графа все ребра пути C_1 и все изолированные вершины (они могут появиться). При этом степени всех вершин останутся четными. Начнем новый путь из v_3 (один шаг мы точно можем сделать). Повторим рассуждения — получим цикл C с началом и концом в v_3. Добавим его к циклу C_1. Получим новый цикл $v_0C_2v_0$, но число пройденных ребер по сравнению с циклом C_1 увеличилось.

Шаг 3. Посмотрим, прошли ли мы все ребра. Если нет, то повторим процесс. При каждом повторении число пройденных ребер растет. Граф конечен — значит в какой-то момент цикл C_m включит в себя все ребра. □

Определение 51. *Граф называется гамильтоновым, если он содержит гамильтонов цикл — простой цикл, проходящий через все вершины графа.*

Конечно, любое ребро в гамильтоновом цикле проходится только один раз.

Пример 34. *Гамильтонов цикл не обязан проходить все ребра. Например, граф G_1 гамильтонов:* 1, 2, 3, 4, 5, 6, 7, 1.

Замечание 3. *Если $G_1\tilde{G}_2$, причем G_1 гамильтонов, то G_2 — тоже гамильтонов граф.*

Упражнение 16. *Граф G_3 гамильтонов, а граф G_4 — нет.*

Лекция 9. Деревья.

В этой лекции все графы пердполагаем простыми.

Утверждение 19. *Назовем две вершины v_0 и v_1 графа G эквивалентными, если найдется путь v_0Cv_1. Это отношение удовлетворяет аксиомам эквивалентности.*

Доказательство. Если $v_1 = v_0$, то возьмем пустой путь. Если есть путь из v_0 в v_1, то запишем его в обратном порядке — получим путь из v_1 в v_0. Если есть путь C_1 из v_0 в v_1 и есть путь C_2 из v_1 в v_2, то объединим их — получим путь $v_0C_1v_1C_2v_2$ из v_0 в v_2. □

Разобъем вершины графа на классы эквивалентности. Это будут связные графы. Их называют *связными компонентами* графа G. Каждая связная компонента — это связный подграф графа G. При этом, в отличие от общей ситуации, этот подграф можно задать списком $\{v_1, \ldots, v_k\}$ его вершин — список ребер подграфа есть просто все ребра графа G, инцидентные вершинам $\{v_1, \ldots, v_k\}$.

Бывает так, что связная компонента состоит из одной вершины. Такие вершины называют *изолированными*. Ясно, что если граф G на p вершинах разбивается на p связных компонент, то каждая его компонента содержит ровно одну вершину, т.е. граф состоит из p изолированных вершин (не имеет ребер совсем). Такой граф называют *нуль–графом*.

Определение 52. *Дерево — это связный граф без циклов.*

Напомним, что $p(G)$ — это число вершин графа G, а $q(G)$ — число его ребер.

Лемма 6. *Если к дереву T, $p(T) \geq 2$, добавить одно ребро, то возникнет цикл. Вообще, если к связному графу G добавить новое ребро, то возникнет цикл. А к несвязному графу всегда можно добавить ребро так, что цикл не появится.*

Лемма 7. *Если в дереве T, $p(T) \geq 2$, удалить одно ребро, то получится два несвязных дерева.*

Доказательство. Пусть мы удалили ребро $e = (u, v)$. Если граф остался связным, то существует путь uCv. Тогда путь $uCveu$ — цикл. Противоречие. Если теперь w — какая–то третья вершина дерева T, то существовал путь uC_1w. Одно из двух, либо этот путь не проходил через ребро e, либо проходил (возможно, несколько раз). В первом случае — этот путь остался в графе $T \setminus \{e\}$, т.е. w и u лежат в одной связной компоненте этого графа (обозначаем $w\tilde{u}$). Во втором случае найдем в пути C_1 последнее вхождение ребра e. Получим либо запись uC_2uevC_3w, либо запись uC_2veuC_3w. Тогда путь C_3 лежит в $T \setminus \{e\}$, т.е. либо $v\tilde{w}$, либо $u\tilde{w}$ в графе $T \setminus \{e\}$. Итак, граф $T \setminus \{e\}$ разбился на два несвязных графа. Новых циклов точно появиться не могло, значит это два дерева. □

Теорема 19. *Для дерева $p = q + 1$.*

Доказательство. Берем дерево T. Удаляем одно ребро — получаем два несвязных дерева. Удаляем второе ребро — получаем три несвязных дерева и т.д. Процесс закончится тогда, когда кончатся ребра. Это произойдет в точности тогда, когда каждая связная компонента состоит из одной вершины, т.е. таких компонент p. Произойдет это после $p-1$ шага. Значит у нас было $p - 1$ ребро. □

Поставим следующую задачу. У дерева циклов нет. А что если нам дан какой–то связный граф, у которого циклы есть. Сколько ребер надо удалить, чтобы получить дерево?

Лемма 8. *Если некоторое ребро e связного графа G входит в цикл на графе, то при удалении этого ребра граф останется связным.*

Доказательство. Берем две произвольные вершины u и v. Они были связаны путем. Если в этот путь входило ребро e, то заменим его цепочкой $C \setminus e$. □

Лемма 9. *Если связный граф имеет циклы, то из него можно удалить несколько ребер и получить дерево.*

Доказательство. Будем удалять ребра, пока в графе есть циклы. Получим дерево. □

Определение 53. *Полученное дерево называют остовным деревом графа.*

Теорема 20. *Следующие утверждения эквивалентны*
(i) Граф G связен и не имеет циклов;
(ii) Граф G связен и $p = q + 1$;
(iii) Граф G не имеет циклов и $p = q + 1$;
(iv) Граф G связен, но при удалении любого ребра перестает быть связным;
(v) Граф G не имеет циклов, но при добавлении любого ребра, цикл появляется.

Доказательство. $(i) \Rightarrow (ii)$ доказали в теореме 19.
$(ii) \Rightarrow (iii)$ следует из леммы 9. Если связный граф G имеет циклы, то мы удалим несколько ребер и получим дерево с $q < p - 1$. Противоречие.
$(iii) \Rightarrow (iv)$ Если бы граф был несвязен, то мы могли бы добавить к нему несколько ребер и получить дерево (лемма 6) с $q > p - 1$. Противоречие. Значит, граф G связен и не имеет циклов. Тогда он дерево, а значит при удалении любого ребра теряет связность (лемма 6).
$(iv) \Rightarrow (v)$ Если бы в графе был цикл, мы бы удалили одно из его ребер и получили бы связный граф. Мы доказали, что наш граф — дерево. Тогда второе утверждение — это лемма 6.
$(v) \Rightarrow (i)$ Если граф не связен, то найдутся вершины u и v, лежащие в разных компонентах связности. Т.е. путь из u в v отсутствует. Добавим ребро (u, v) — цикла не возникнет (иначе был бы путь из u в v). Противоречие. □

Конечно, как любые графы, одно и то же дерево можно по-разному изобразить на плоскости. Как и для графов, два разных дерева могут быть изоморфными. Однако для деревьев появляются и новые понятия.

Определение 54. *Дерево называется корневым, если в нем выделена одна вершина (она называется корнем). Два корневых дерева называем изоморфными, если между ними существует изоморфизм в смысле графов, переводящий корень в корень.*

Очень часто вершины деревья приходится нумеровать. Часто используется нумерация (точнее, индексация) "предок–потомок". Применяется в структурах данных. Корень индексируем (). Занумеруем все вершины, смежные с корнем, в некотором порядке. Сопоставим каждой из них индекс (i), где $i \leq 1$. Такие вершины называют потомками первого уровня. Переберем все незанумерованные вершины, смежные с вершиной (1). Занумеруем их в произвольном порядке и каждой выдадим индекс $(1, j)$. Потом так же поступим с вершиной (2) и так далее. Получим индексацию вида (i, j) некоторого множества вершин. Эти вершины будем называть потомками второго уровня. И так далее. Каждая вершина получит индекс вида $(i_1, i_2, \ldots, i_s)$. Теперь можно еще записать эти индексы в лексикографическом порядке и, таким образом, занумеровать вершины. При графическом изображении такого индексированного дерева всегда изображают потомков одного уровня на одной строке, причем в порядке их нумерации.

Определение 55. *Индексированное таким способом дерево называют упорядоченным деревом.*

Часто в дополнение к индексации ребра дерева, соединяющие вершины с индексами $(i_1, i_2, \ldots, i_s)$ и $(i_1, i_2, \ldots, i_s, j)$, ориентируют (именно в таком порядке). При этом у каждой вершины получается ровно один предок (с индексом, который получается отбрасыванием последней координаты). Потомков может быть любое число.

Замечание 4. *Очевидно, полный набор индексов полностью определяет дерево.*

Определение 56. *Поддерево дерева T — это подграф T, являющийся деревом.*

Сопоставим каждому упорядоченному дереву двоичный код. Кодирование определим по индукции. Дерево, состоящее из одного корня, вообще кодировать не будем. Дерево, состоящее из одного ребра, закодируем (01). Пусть мы научились кодировать все деревья с числом ребер $< q$. Пусть нам дано упорядоченное дерево с q ребрами. Обозначим T_1 — поддерево, состоящее из вершин с индексами $(1, *)$. Вообще, обозначим T_i — поддерево, состоящее из вершин с индексами $(i, *)$. Каждое из них содержит $< q$ ребер. Значит, каждое из них мы уже умеем кодировать. Пусть $\vec{\alpha}_i$ — их двоичные коды. Тогда дерево T закодируем кодом

$$(0\vec{\alpha}_1 10\vec{\alpha}_2 10\vec{\alpha}_3 1 \ldots 0\vec{\alpha}_s 1). \tag{4}$$

По построению, каждому упорядоченному дереву корректно сопоставлен двоичный код.

Замечание 5. *Дерево с двумя ребрами имеет код либо* (0011)*, либо* (0101)*. Любой код начинается с нуля и заканчивается единицей. Любой код имеет одинаковое число нулей и единиц. Длина кода равна $2q$, где q — число ребер дерева.*

Оказывается, любой такой двоичный код задает алгоритм обхода дерева. Этот алгоритм называют *поиском в глубину*. Опишем этот алгоритм:
1) Обход всегда начинаем из корня. Все ребра дерева считаем ориентированными (см. выше) и активными.
2) Проходя про какому-либо ребру против его ориентации, договоримся помечать это ребро неактивным.
3) Читаем код слева направо. Символ 0 дает команду перейти к потомку с наименьшим индексом (неактивные ребра не рассматриваются).
4) Символ 1 дает команду перейти к предку.

Утверждение 20. *Закодируем произвольное дерево T. Тогда его код полностью определяет набор индексов вершин, а значит, и все дерево.*

Доказательство. Опишем алгоритм восстановления индексов. Начальный индекс по определению равен (). Каждая команда кода 0 заставляет нас приписать к текущему индексу одну координату. При этом выбираем для приписывания такое минимальное число, которое создаст новый индекс (отличный от всех предыдущих). Каждая команда 1 заставляет нас удалить одну (последнюю) координату у текущего индекса.
Теперь докажем, что такой алгоритм действительно восстановит исходное дерево. По индукции (индукцию ведем по числу ребер). База, код 01 очевидна. Каждый код имеет

структуру (4). В начале чтения кода $\vec{\alpha}_1$ текущий индекс равен (1). Тогда, по предположению индукции, код $\vec{\alpha}_1$ восстановит все индексы поддерева T_1. Мы же получим все индексы вида $(1, *)$. По окончании чтения вектора $\vec{\alpha}_1$ текущий индекс будет равен (1). Тогда в начале чтения кода $\vec{\alpha}_2$ текущий индекс будет равен (2). И так далее. □

Следствие 2. *Число упорядоченных деревьев с q ребрами не превосходит 4^q.*

Лекция 10. Планарные графы.

Вначале немного топологии.

Определение 57. *Непрерывная кривая в $\mathbb{R}^3$ — это тройка непрерывных функций $\gamma(t) = (x(t), y(t), z(t))$ параметра $t \in [0, 1]$.*

Определение 58. *Пусть $G = (V, E)$ — неориентированный граф. Пусть задана инъекция $\mathcal{F} : V \to \mathbb{R}^3$. Пусть каждому ребру $e = (u, v)$ сопоставлена непрерывная кривая $\gamma_e(t)$ без самопересечений, причем $\gamma(0) = \mathcal{F}(u)$, $\gamma(1) = \mathcal{F}(v)$. Если эти кривые не пересекаются, то говорят, что задана геометрическая реализация графа в $\mathbb{R}^3$.*

Утверждение 21. *Любой неориентированный граф можно реализовать в $\mathbb{R}^3$.*

Доказательство. Расположим все точки $\mathcal{F}(u)$ на прямой l в $\mathbb{R}^3$. Теперь зададим систему плоскостей, проходящих через l, в количестве $q = |E|$ штук. Занумеруем ребра и ребру с номером k сопоставимм подходящую кривую в плоскости с номером k. □

Определение 59. *Граф называется планарным, если существует его реализация $\mathcal{F}$ в $\mathbb{R}^2$.*

Определение 60. *Множество U на плоскости называется открытым, если каждую свою точку (x_0, y_0) оно содержит вместе с некоторой ϵ–окрестностью. Открытое множество называется связным, если любые его две точки можно соединить непрерывной кривой, целиком находящейся в U.*

Назовем две точки множества U эквивалентными, если их можно соединить непрерывной кривой в U. Все множество тогда разобъется на связные компоненты.

Теорема 21 (Лемма Жордана)**.** *Любая непрерывная замкнутая кривая (т.е. $\gamma(0) = \gamma(1)$) без самопересечений разбивает открытое связное множество в $\mathbb{R}^2$ на две компоненты связности (два открытых связных множества; их обозначают $\operatorname{int}\gamma$ и $\operatorname{ext}\gamma$). Любая незамкнутая кривая без самопересечений оставляет плоскость связной (т.е. $\mathbb{R}^2 \setminus \gamma$ — открытое связное множество).*

Определение 61. *Пусть $\mathcal{F}(G)$ — геометрическая реализация графа G на плоскости $\mathbb{R}^2$. Применяя теорему Жордана несколько раз видим, что набор из q кривых $\mathcal{F}((u, v))$ разбивает $\mathbb{R}^2$ на $\leq q + 1$ компоненту связности. Эти компоненты называют гранями планарной реализации $\mathcal{F}(G)$. Их число обозначаем r.*

Пример 35. *Дерево — это связный граф без циклов. Тогда любая его планарная реализация имеет ровно одну грань.*

Упражнение 17. *У любого дерева существует планарная реализация, т.е. дерево — планарный граф.*

Теорема 22 (Формула Эйлера)**.** *Если $\mathcal{F}(G)$ — какая либо планарная реализация графа с p вершинами, q ребрами и r гранями, то $p - q + r = 2$.*

Доказательство. Проведем индукцию по числу q при фиксированном p. Наименьшее число ребер у связного грфа с p вершинами равно $p - 1$ (если предположить, что $q < p - 1$, то ликвидируя при необходимости несколько циклов удалением ребер, придем к дереву с $q' < p - 1$). Итак, база индукции: $q = p - 1$. В этом случае граф — дерево, $r = 1$ и формула верна.

Пусть утверждение доказано для всех $q < q_0$, где $q_0 > p - 1$. Рассмотрим связный граф с p вершинами и q_0 ребрами. Он не дерево — значит у него есть цикл. Если цикл не простой, то выделим простой подцикл. При геометрической реализации этот цикл отобразится в замкнутую непрерывную кривую. По лемме Жордана эта кривая разбивает $\mathbb{R}^2$ на две компоненты связности. Если же мы удалим из графа ребро, входящее в данный цикл, то компонента будет одна. Восстановим все остальные элементы геометрической реализации графа. Видим, что при реализации графа G число граней на 1 больше, чем при реализации $G \setminus \{e\}$. Переход индукции доказан. □

Утверждение 22. *Граф планарен тогда и только тогда, когда существует его геометрическая реализация на сфере.*

Доказательство. Рассмотрим стереографическую проекцию. □

Утверждение 23. *Для любого выпуклого многогранника в $\mathbb{R}^3$ верна формула Эйлера.*

Идея доказательства. Поместим многогранник в шар с центром O, не лежащим на многограннике. Устроим центральную проекцию из точки многогранника на сферу. Потом стереографической проекцией перенесем картинку на плоскость. □

Какие графы планарны, а какие нет?

Пример 36. *Полный пятиугольник не планерен.*

Доказательство. У него $p = 5$, $q = 10$. Если бы он был планерен, то $r = 7$. Но граница каждой грани содержит ≥ 3 кривых — образов ребер. Посчитаем количество ребер, лежащих на границе для каждой фиксированной грани, а потом сложим эти числа. Получим ≥ 21 ребер. С другой стороны, каждое ребро лежит на границе либо одной, либо двух граней. Каждое ребро мы считали либо один раз, либо дважды. Но тогда эта сумма не может быть больше $2q = 20$. Противоречие. □

Пример 37. *Двудольный шестиугольник не планерен.*

Доказательство. У него $p = 6$, $q = 9$. Если бы он был планерен, то $r = 5$. Но граница каждой грани содержит ≥ 4 кривых — образов ребер (у него нет циклов длины 3). Посчитаем количество ребер, лежащих на границе для каждой фиксированной грани, а потом сложим эти числа. Получим ≥ 20 ребер. С другой стороны, каждое ребро лежит на границе либо одной, либо двух граней. Значит, каждое ребро мы считали либо один раз, либо дважды. Но тогда эта сумма не может быть больше $2q = 18$. Противоречие. □

Определение 62. *Операцией сжатия ребра $e = (u, v)$ (иногда эту операцию называют операцией слияния вершин u и v) в графе G называют переход к графу G' (см. рисунок).*

Теорема 23 (Куратовский)**.** *Граф планарен тогда и только тогда, когда путем удаления ребер, удаления вершин и сжатия ребер из него нельзя получить K_5 или $K_{3,3}$.*

Пример 38. *Граф на рисунке не планарен.*

Поговорим немного о раскраске графов.

Определение 63. *Множество $V_1 \subset V$ называется независимым, если никакие две его вершины не смежны. Множество V_1 называется кликой, если наоборот, любые две его вершины смежны. Раскраской вершин графа называется отображение $\mathfrak{P} : V \to \mathbb{N}$. Раскраска называется правильной, если полные прообразы $\mathcal{P}(1)$, $\mathcal{P}(2)$ и т.д. являются независимыми множествами. Наименьшее число цветов, необходимое для правильной раскраски вершин графа называют его хроматическим числом.*

Иными словами, каждой вершине v мы сопоставляем цвет с номером $\mathfrak{P}(v)$. А потом требуем, чтобы никакие вершины одного цвета не были смежны.

Упражнение 18. *Полный граф K_n можно раскрасить правильным способом только в n цветов. Вообще, если граф содержит клику размера n, то потребуется как минимум n цветов. Любой двудольный граф можно правильно раскрасить в 2 цвета.*

Пример 39. *Дерево всегда можно раскрасить правильным образом в два цвета.*

Доказательство. Покрасим корень в черный цвет. Все вершины первого уровня — в красный. Все вершины второго уровня — в черный. И т.д. □

Теорема 24. *Любой планарный граф можно правильным образом покрасить в 5 цветов.*

Определение 64. *Раскраской ребер называют отображение $\mathcal{P} : E \to \mathbb{N}$. Раскраска называется правильной, если ребра одного цвета не смежны. Наименьшее число цветов, необходимое для правильной раскраски ребер, называют реберным хроматическим числом (или хроматическим индексом).*

Определение 65. *Тотальной раскраской графа называют отображение $\mathcal{P} : V \cup E \to \mathbb{N}$. Раскраска называется правильной, если ни смежные вершины, ни смежные ребра, ни инцидентные друг другу вершины и ребра, не имеют одного цвета.*

Определение 66. *Раскраской граней планарного графа называют отображение $\mathcal{P}$, сопоставляющее цвет из $\mathbb{N}$ для каждой грани. Раскраска называется правильной, если грани, обладающие общей границей (т.е. кривой γ, лежащей на границе обеих граней), раскрашены в разный цвет.*

Теорема 25. *Грани любого планарного графа можно правильно раскрасить в 4 цвета.*

Лекция 11. Алфавитное кодирование.

Определение 67. *Алфавит — это конечное множество $\mathcal{A} = \{a_1, \ldots, a_p\}$, символы a_j — буквы. Слово длины n в алфавите $\mathcal{A}$ — это элемент множества $\mathcal{A}^n$, т.е. упорядоченный набор из n букв.*

Слово длины 0 — это пустое слово. Его обозначаем символом Λ. Другие слова обозначаем заглавными латинскими буквами. Длину слова A обозначаем $|A|$. При записи слова часто обрамляют скобками $(\ldots)$ или апострофами $`\ldots`$ (при этом предполагают, что символы (и) или, соответсвенно символ ` не входят в алфавит).

Определение 68. *Операция конкатенации слов A и B — это слово AB (пишем слово A и без пробела пишем слово B).*

Если слово A можно представить в виде BC, то B называется *префиксом* в A. Если $A = CB$, то B — *суффикс* в слове A. Если $A = CBD$, то B — *подслово* в A, или еще говорят, что B *входит в* A.

Пример 40. *Пусть $\mathcal{A}$ — алфавит русского языка. Тогда 'космос' — это слово длины 6. Слово `кос` — его префикс, слово `ко` — тоже префикс. И даже `космос` — префикс (и суффикс тоже). Слово `ос` входит в `космос` дважды.*

Определение 69. *Множество всех слов в алфавите $\mathcal{A}$ обозначаем через $\mathcal{A}^*$.*

Ясно, что

$$\mathcal{A}^* = \emptyset \cup \mathcal{A} \cup \mathcal{A}^2 \cup \ldots.$$

Определение 70. *Пусть есть два алфавита $\mathcal{A} = \{a_1, \ldots, a_p\}$ и $\mathcal{B} = \{b_1, \ldots, b_q\}$. Кодированием называется отображение $\varphi : \mathcal{A} \to \mathcal{B}^*$.*

Иными словами, каждой букве a_j алфавита $\mathcal{A}$ мы сопоставляем слово $\varphi(a_j) = B_j$ в алфавите $\mathcal{B}$. Это слово может состоять из одной буквы, а может и из нескольких. Эти слова называем *кодовыми словами.*

Определение 71. *Кодирование распространяется на все слова в алфавите $\mathcal{A}$ с помощью конкатенации*

$$\varphi : \ A = (a_{i_1} a_{i_2} \ldots a_{i_s}) \to B_{i_1} B_{i_2} \ldots B_{i_s}.$$

Пустое слово всегда кодируем пустым словом.

Пример 41. *Пусть $\mathcal{A}$ — алфавит русского языка, а $\mathcal{B} = \{0, 1, \ldots, 9\}$. Занумеруем русские буквы по порядку, т.е. устроим кодирование*

$$\varphi(а) =' 1', \ \varphi(б) =' 2', \ \ldots, \ \varphi(я) =' 33'$$

Тогда $\varphi(космос) = 121619141619$.

Определение 72. *Кодирование называется однозначным (или декодируемым), если $\varphi : \mathcal{A}^* \to \mathcal{B}^*$ инъекция (разные слова переходят в разные).*

Конечно, для однозначности кодирования необходимо, чтобы все кодовые слова были разными (далее по умолчанию считаем, что это условие выполнено). Но этого недостаточно для однозначности.

Пример 42. *Кодирование из примера 41 не однозначно. Например, слово ‘аа‘ и слово ‘й‘ оба кодируются словом ‘11‘. Т.е. φ не инъективно даже на $\mathcal{A} \cup \mathcal{A}^2$.*

Определение 73. *Кодирование называют равномерным, если $|B_1| = |B_2| = \cdots = |B_p|$.*

Утверждение 24. *Любое равномерное кодирование (при условии, что все B_j различны) однозначно.*

Доказательство. Обозначим $|B_j| = \beta$. Пусть $\varphi(A_1) = \varphi(A_2) = B$. Тогда $|A_1| = |A_2| = |B|/\beta$. Посмотрим на префикс слова B длины β. Это одно из кодовых слов. Значит, у слов A_1 и A_2 совпадают первые буквы. Посмотрим на следующее подслово в B длины β. И т.д. □

Определение 74. *Кодирование называется префиксным, если никакое из слов B_j не является префиксом никакого слова B_k.*

Иными словами, равенство $B_k = B_j C$ невозможно.

Утверждение 25. *Любое префиксное кодирование однозначно.*

Доказательство. Пусть $F_1 := \varphi(A_1) = F_2 := \varphi(A_2)$. Пусть несколько (возможно, ноль) первых букв в словах A_1 и A_2 совпадают, т.е. $A_1 = AC_1$, $A_2 = AC_2$. Тогда $F_1 = \varphi(A)\varphi(C_1)$, $F_2 = \varphi(A)\varphi(C_2)$, т.е. начало у слов F_1 и F_2 тоже совпадает. Предположим, что затем у слов A_1 и A_2 идут различные буквы (скажем, $a_i \neq a_j$). Тогда у слов F_1 и F_2 после слова $\varphi(A)$ идут слова B_i и B_j. Но тогда получается, что либо B_i префикс B_j, либо наоборот. Противоречие. □

Определение 75. *Кодирование называется постфиксным (или суффиксным), если никакое из слов B_j не является суффиксом никакого слова B_k.*

Иными словами, равенство $B_k = CB_j$ невозможно.

Утверждение 26. *Любое постфиксное кодирование однозначно.*

Что можно сказать о кодировании, которое не является ни равномерным, ни префиксным, ни постфиксным — однозначно ли оно? Для проверки однозначности кодирования можно использовать следующий алгоритм (Маркова).

Определение 76. *Допустимым разложением кодового слова B_i назовем его представление в виде*

$$B_i = \beta' B_{i_1} \dots B_{i_s} \beta'',$$

где B_{i_j} — другие кодовые слова. Будем называть B_{i_j} корнями, а весь конгломерат $B_{i_1} \dots B_{i_s}$ назовем мультикорнем разложения. Далее, β' и β'' — это просто какие–то слова в алфавите $\mathcal{B}$ (будем называть их началом и окончанием разложения). Корней в разложении может и не быть, т.е. допускается $s = 0$ (мультикорень разложения пустой). Начала или окончания тоже может не быть, т.е. допускается $\beta' = \Lambda$ или $\beta'' = \Lambda$. Мы, однако, требуем, чтобы из трех частей разложения: начало, мультикорень и окончание — было хотя бы две непустых части.

Опишем сам алгоритм.
1. Составить все допустимые разложения.
2. Составить множество всех начал M_1, множество всех окончаний M_2 и их пересечение $M = M_1 \cap M_2$.
3. Составить ориентированный мультиграф с петлями. Вершинами графа являются элементы множества M, а ребрами — мультикорни допустимых разложений.
4. Воспользоваться следующей теоремой.

Теорема 26 (А. А. Марков)**.** *Кодирование однозначно тогда и только тогда, когда граф Маркова не содержит пути с началом и концом в вершине* Λ.

Доказательство. Пусть в графе есть цикл v_0, e_1, v_1, e_2, $v_2, \ldots$, e_n, v_0, где $v_0 = \Lambda$. Прочтем его двумя способами. Первый способ

$$(e_1)(v_1e_2v_2)(e_3)(v_3e_4v_4)\ldots.$$

Напомним, что каждое ребро графа — это мультикорень, а значит записывается последовательностью кодовых слов. С другой стороны, каждая тройка $u(u,v)v$ из вершины, ребра и вершины — это допустимое разложение какого–то кодового слова. Значит первый способ дает последовательность кодовых слов, равную $\varphi(A_1)$. Предъявим второй способ

$$(v_0e_1v_1)(e_2)(v_2e_3v_3)(e_4)\ldots.$$

По тем же причинам — это есть слово $\varphi(A_2)$. Очевидно, $A_1 \neq A_2$ (хотя бы потому, что у них разнные первые буквы), но $\varphi(A_1) = \varphi(A_2)$. Кодирование неоднозначно.

Обратно, пусть кодирование неоднозначно. Запишем слово $B = b_{i_1}b_{i_2}\ldots$, которое допускает два раскодирования. Будем изображать слова первого раскодирования дугами над словом, а второго — дугами под словом. Основания дуг назовем верхними и нижними опорными точками. Первая опорная точка является общей — это начало слова. Будем считать, что вторая общая опорная точка — это конец слова. Иначе мы можем перейти к более короткому слову, допускающему неоднозначное раскодирование. Для определенности, предположим, что первая верхняя дуга длиннее первой нижней. Составим последовательность подслов по следующему алгоритму. Найдем вторую верхнюю опорную точку (окончание первой верхней дуги). Эта дуга станет набором e_1v_1 графа Маркова. Найдем слева от нее ближайшую нижнюю опорную точку. Часть слова между этими точками назовем β_1. Она является одновременно окончанием первого верхнего допустимого разложения e_1 и началом следующего разложения (первого нижнего). Она станет вершиной v_1. Возьмем следующую по счету нижнюю опорную точку. Найдем слева от нее ближайшую верхнюю опорную точку. Часть слова между этими точками назовем β_2. Получим допустимое разложение слова, отвечающего первой нижней дуге. Это разложение в графе станет $v_1e_2v_2$. Возьмем следующую по счету верхнюю опорную точку. И так далее. Получим искомый путь в графе Маркова. □

Пример 43. *В префиксном кодировании не бывает разложений вида* $B_i = \Lambda B_{i_1}\ldots B_{i_s}\beta''$, *т.е. граф Маркова вообще не содержит вершины* Λ.

Пример 44. *Пусть* $\mathcal{A} = \{a, b, c\}$, $\mathcal{B} = \{0, 1\}$, *отображение* φ *переводит*

$$a \to B_1 = (0), \quad b \to B_2 = (001), \quad c \to B_3 = (010).$$

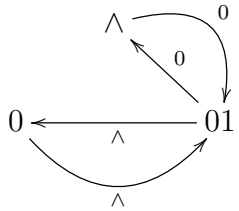
Составим допустимые разложения каждого из слов B_j

$$B_2 = (0) \wedge (01) = (00) \wedge (1) = \wedge(0)(01) = (0)(0)(1) = \wedge(00)(1),$$
$$B_3 = (0) \wedge (10) = (01) \wedge (0) = \wedge(0)(10) = (01)(0) \wedge .$$

Составим множества

$$M_1 = \{0, 00, \wedge, 01\}, \quad M_2 = \{01, 1, 10, 0, \wedge\}, \quad M = \{0, 01, \wedge\}.$$

Составим граф



Видим циклы с началом в точке $\wedge$. *Самый простой из них отвечает кодировке* $0010 = \varphi(ac) = \varphi(ba)$.

Искать циклы в графе — задача довольно сложная (в смысле сложности вычислений). Хотелось бы получить какую-то оценку сложности. Под сложностью будем понимать длину слова B в алфавите $\mathcal{A}$, допускающего неоднозначную раскодировку. В нашем последнем примере сложность равна 2. Ясно, что сложность — это число кодовых слов, которые входят в кодировку слова B.

Введем параметры для оценки. Пусть T — максимальное число кодовых слов, составляющих мультикорень какого–либо кодового слова. Пусть $l_i = |B_i|$ и $L = \sum_{i=1}^{p} |B_i|$.

Теорема 27 (А. А. Марков)**.** *Сложность не превосходит целой части числа* $\frac{(T+1)(L-p+2)}{2}$.

Доказательство. Ясно, что цикл в графе всегда можно сделать простым (если цикл проходит какую–то вершину дважды, отбросим часть цикла). Тогда длина цикла заведомо не больше, чем число вершин в графе Маркова. Таких вершин не больше, чем всевозможных начал. Для каждого кодового слова B_i можно записать не больше чем $l_i - 1$ его непустых начал. Тогда всего имеем не больше, чем

$$l_1 - 1 + l_2 - 1 + \cdots = L - p$$

вершин. Разобъем цикл на куски вида ‘(ребро)(вершина-ребро-вершина)‘. Таких кусков не больше, чем половина длины цикла $\frac{1}{2}L - p + 1$ (последний кусок, возможно, сформировать не удастся). В каждый кусок помещается не больше, чем $T + 1$ кодовое слово: T в первое ребро и 1 в ‘вершина-ребро-вершина‘. Итого, не более $\frac{T+1}{L-p+2}2$ кодовых слов. □

В примере 44 имеем $T = 2$, $L = 7$, $p = 3$. Тогда оценка сложности равна 9 (на самом деле, сложность равна 2).

Лекция 12. Оптимальные (по длине) коды.

Теорема 28 (Неравенство Макмиллана). *Пусть дано однозначное кодирование $\varphi(a_j) = B_j$, $1 \le j \le p$. Пусть в алфавите $\mathcal{B}$ ровно q букв, а длины кодовых слов равны $|B_j| = l_j$. Тогда*

$$\sum_{j=1}^{p} q^{-l_j} \le 1.$$

Доказательство. Обозначим левую часть неравенства через S. Пусть $l_{min} = \min\{l_1, \dots, l_p\}$, а $l_{max} = \max\{l_1, \dots, l_p\}$. Тогда

$$S^n = \underbrace{(q^{-l_1} + \dots + q^{-l_p}) \cdot (q^{-l_1} + \dots + q^{-l_p}) \dots (q^{-l_1} + \dots + q^{-l_p})}_{n \text{ раз}} = \sum_{k=n \cdot l_{min}}^{n \cdot l_{max}} c_k q^{-k}.$$

Как получается степень q^{-k} — при перемножении $q^{-l_{j_1}} \dots q^{-l_{j_n}}$, $l_{j_1} + \dots + l_{j_n} = k$. Сколько раз в итоговой сумме возникнет такое слагаемое (т.е. чему равно c_k)? Каждому такому слагаемому сопоставим слово $B = B_{j_1} \dots B_{j_n}$ (оно имеет длину k). Таких слов точно не больше, чем q^k. Наше сопоставление инъективно — слово B можно записать последовательностью кодовых слов единственным образом (иначе кодирование было бы неоднозначно). Тогда $c_k \le q^k$, а $S^n \le n(l_{max} - l_{min})$. Устремим в этом неравенстве $n \to \infty$. Получим

$$S \le \sqrt[n]{n(l_{max} - l_{min})} \to 1.$$

□

Замечательно то, что верно обратное.

Теорема 29. *Пусть в алфавите $\mathcal{A}$ имеется p букв, в алфавите $\mathcal{B}$ — q букв. Пусть нам даны p натуральных чисел $l_1, \dots, l_p$ таких, что $\sum_{j=1}^{p} q^{-l_j} \le 1$. Тогда можно построить такое однозначное (и даже префиксное) кодирование φ, что длина кодового слова $\varphi(a_j)$ будет равна l_j.*

Доказательство. Сгруппируем последовательность $l_1, \dots, l_p$ по возрастанию. Пусть в ней $\varkappa_1$ единиц, $\varkappa_2$ двоек и т.д. Числа $\varkappa_j$ могут быть нулями, они заведомо равны нулю при $j > l_{max}$. Конечно, $\varkappa_1 + \varkappa_2 + \dots = p$. Обозначим $S = \sum_{j=1}^{p} q^{-l_j}$ и сгруппируем в ней одинаковые слагаемые

$$S = \frac{\varkappa_1}{q} + \frac{\varkappa_2}{q^2} + \dots.$$

Раз $S \le 1$, то $\varkappa_1 \le q$ (мы отбросили в неравенстве все слагаемые, кроме первого). Значит, для кодировки однобуквенными словами нам вариантов хватает. Положим $\varphi(a_i) = b_i$ для $1 \le i \le \varkappa_1$ и эти буквы на первое место в кодовых словах больше ставить не будем. Тогда разрешенных двухбуквенных слов останется $q^2 - \varkappa_1 q$.

Отбросим в неравенстве все слагаемые, кроме первых двух. Получим $\varkappa_2 \le q^2 - \varkappa_1 q$. У нас в ассортименте есть как раз $q^2 - \varkappa_1 q$ двухбуквенных слов. Опять хватает. Закодируем ими следующие $\varkappa_2$ букв алфавита $\mathcal{A}$. Не будем ставить эти слова в начало трехбуквенных слов. Тогда разрешенных трехбуквенных слов останется $q^3 - \varkappa_1 q^2 - \varkappa_2 q$.

И так далее. На m–ом шаге у нас есть $q^m - \varkappa_1 q^2 - \cdots - \varkappa_{m-1} q$ разрешенных слов длины m. Но и $\varkappa_m \le q^m - \varkappa_1 q^2 - \cdots - \varkappa_{m-1} q$ (мы оставили в неравенстве m слагаемых). Значит, мы всегда можем продолжать процесс кодирования до тех пор, пока не кончится алфавит $\mathcal{A}$. Полученное кодирование является префиксным по построению. □

Пример 45. *Пусть* $\mathcal{A} = \{a, b, c, d\}$, *а* $\mathcal{B} = 0, 1$, *т.е.* $p = 4$, $q = 2$. *Пусть* $l_1 = 1$, $l_2 = 2$, $l_3 = l_4 = 3$. *Как раз* $2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = 1$. *Согласно нашему алгоритму,*

$$\varphi(a) = 0, \quad \varphi(b) = 10, \quad \varphi(c) = 110, \quad \varphi(d) = 111.$$

Следствие 3. *Любое однозначное кодирование* φ *можно заменить на префиксное кодирование* ψ *без изменения длины кодовых слов.*

Доказательство. Раз φ однозначно, то выполнено неравенство Макмиллана. Раз выполнено неравенство Макмиллана, то можно построить префиксное кодирование с заданными длинами слов. □

Пусть нам дан текст в алфавите $\mathcal{A}$. Нам надо закодировать его в алфавите $\{0, 1\}$ так, чтобы длина кодировки была поменьше. Как нам действовать? Логично, что надо посчитать, какие буквы $a_1, \ldots, a_p$ встречаются часто, а какие редко. Те, которые встречаются часто, надо закодировать короткими кодовыми словами, а тем, которые встречаются редко, можно дать и длинные кодовые слова.

Определение 77. *Пусть всего в тексте* N *букв. Частотой буквы* a_j *назовем число* $\omega_j = n_j/N$, *где* n_j *— количество вхождений буквы* a_j *в текст.*

Конечно, $\omega_1 + \cdots + \omega_p = 1$. Предположим, что $\varphi(a_j) = B_j$, $|B_j| = l_j$.

Определение 78. *Ценой кодирования* φ *назовем число* $c(\varphi) = \sum_{j=1}^{p} \omega_j l_j$.

Определение 79. *Кодирование называется оптимальным, если оно однозначно, а числа* l_j *подобраны так, что цена кодирования наименьшая (среди цен всех однозначных кодирований).*

Ясно, что если мы создадим оптимальное кодирование $\varphi(a_j) = B_j$, то кодирование $\psi(a_j) = \overline{B_j}$ тоже оптимально. Из этих двух кодирований мы предпочтем то, для которого единиц меньше.

Заметим три простых наблюдения.

Утверждение 27. *Если кодирование оптимально и* $\omega_i > \omega_j$, *то* $l_i \le l_j$.

Доказательство. Иначе найдется пара букв в алфавите $\mathcal{A}$ таких, что $\omega_i > \omega_j$, но $l_i > l_j$. Тогда кодирование точно не оптимально. Действительно, запишем кодирование ψ, которое во всем совпадает с нашим, за двумя исключениями: $\psi(a_i) = B_j$, $\psi(a_j) = B_i$. Тогда

$$c(\psi) = c(\varphi) - \omega_i l_i - \omega_j l_j + \omega_i l_j + \omega_j l_i = c(\varphi) + (\omega_i - \omega_j)(l_j - l_i) < c(\varphi).$$

□

Конечно, если среди частот есть повторяющиеся $\omega_i = \omega_j$, то мы всегда можем поменять местами слова B_i и B_j без увеличения стоимости.

Утверждение 28. *Пусть $\varphi : a_j \to B_j$ — оптимальное префиксное кодирование, а $l = \max_{1\le j\le p} l_j$. Тогда среди кодовых слов обязательно найдутся два таких, что $|B_i| = |B_j| = l$, причем $B_i = B'\alpha$, а $B_j = B'\overline{\alpha}$.*

Доказательство. Возьмем кодовое слово B_i длины l и запишем его в виде $B_i = B'\alpha$. Если среди кодовых слов нет слова $B'\overline{\alpha}$, то мы создадим кодирование ψ, которое во всем совпадает с нашим за одним исключением $\psi(a_i) = B'$. Кодирование φ префиксное (по условию), значит ни одно из кодовых слов не может быть началом слова B' (или самим словом B'). Значит, ψ — тоже префиксное. При этом $c(\psi) < c(\phi)$, т.е. φ было не оптимальным. Противоречие. □

Утверждение 29. *Пусть $\varphi : a_j \to B_j$ — оптимальное префиксное кодирование и $\omega_1 \ge \cdots \ge \omega_p$. Тогда перестановкой кодовых слов всегда можно перейти к префиксному оптимальному кодированию, для которого $B_p = B'\alpha$, а $B_{p-1} = B'\overline{\alpha}$.*

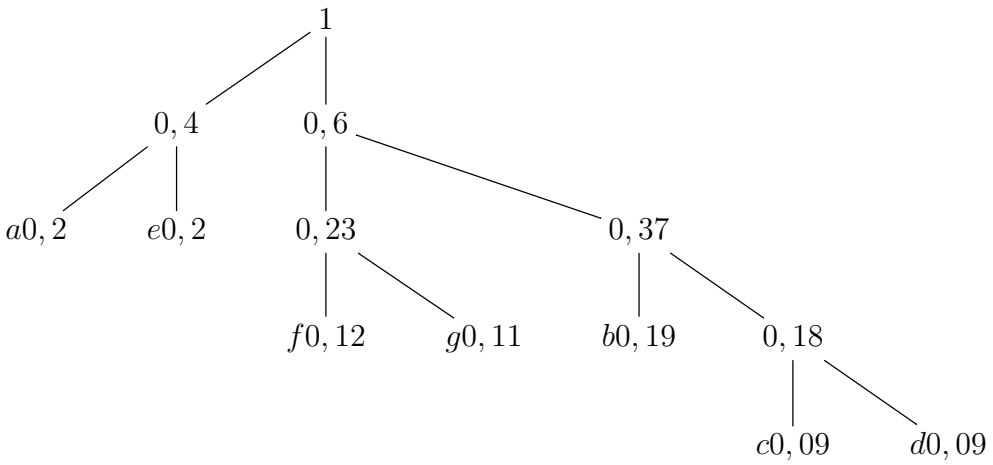
Доказательство. Пусть B_j — самое длинное кодовое слово (длины l). Переставим местами B_j и B_p. Получим кодирование с ценой $c' = c + (\omega_j - \omega_p)(l_p - l_j) \le c$. Значит, либо кодирование было не оптимально (противоречие), либо $c' = c$ и мы получили новое префиксное оптимальное кодирование, у которого на последнем месте стоит самое длинное слово. Мы знаем, что есть кодовое слово B_i, которое отличается от B_p только последней буквой. Переставим местами B_i и B_{p-1}. Получим кодирование с ценой $c'' = c+(\omega_i-\omega_{p-1})(l_{p-1}-l_p) \le c$. Значит, либо кодирование было не оптимально (противоречие), либо мы перешли к новосу префиксному кодированию, у которого два последних слова отличаются только последней буквой. □

Есть алгоритм (алгоритм Хаффмена), котоый создает оптимальное префиксное кодирование.
1. Отсортировать частоты ω_j по убыванию.
2. Сложить две наименьшие частоты. Упорядочить по убыванию.
3. Повторять пункт 2. до тех пор, пока чисел не станет ровно два.
4. Построить дерево.
5. Записать индексацию «предок–потомок» всех вершин дерева в алфавите $\{0, 1\}$.

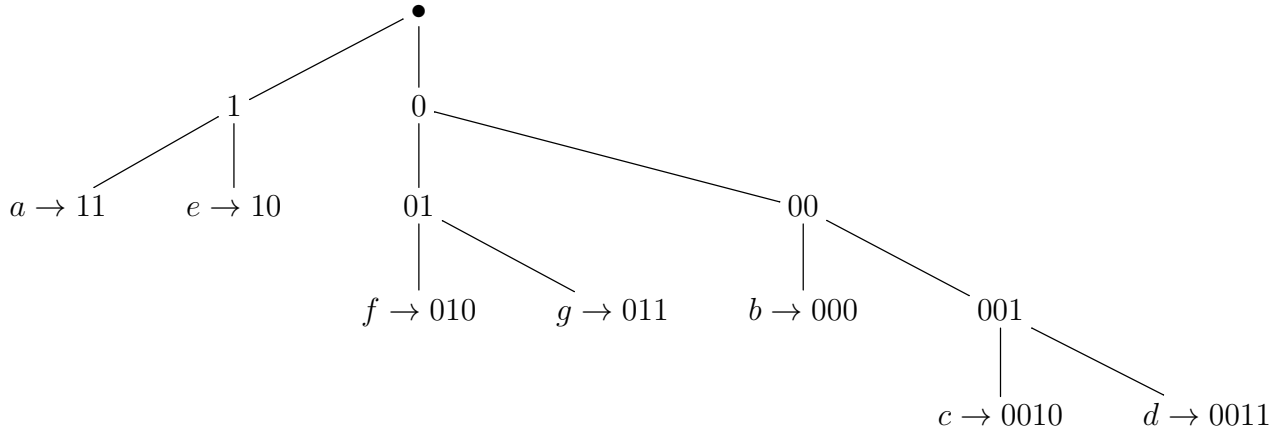
Пример 46. *Пусть $\mathcal{A} = \{a, b, c, d, e, f, g\}$, а частоты равны $0,2$, $0,19$, $0,09$, $0,09$, $0,2$, $0,12$, $0,11$. Результат работы п.2. алгоритма:*

$$\begin{vmatrix} 0,2 \\ 0,2 \\ 0,19 \\ 0,12 \\ 0,11 \\ 0,09 \\ 0,09 \end{vmatrix} \to \begin{vmatrix} 0,2 \\ 0,2 \\ 0,19 \\ 0,18 \\ 0,12 \\ 0,11 \end{vmatrix} \to \begin{vmatrix} 0,23 \\ 0,2 \\ 0,2 \\ 0,19 \\ 0,18 \end{vmatrix} \to \begin{vmatrix} 0,37 \\ 0,23 \\ 0,2 \\ 0,2 \end{vmatrix} \to \begin{vmatrix} 0,4 \\ 0,37 \\ 0,23 \end{vmatrix} \to \begin{vmatrix} 0,6 \\ 0,4 \end{vmatrix}.$$

Дерево будет выглядеть так



После индексации «предок–потомок» получим



Теорема 30. *Алгоритм Хаффмена создает оптимальное префиксное кодирование*

Доказательство. По построению, код является префиксным (индексация «предок–потомок» префиксная). Докажем оптимальность по индукции. В алфавите из двух символов с частотами $\omega_1 \geq \omega_2$ кодировка нулем и единицей оптимальна (очевидно). Докажем переход индукции (он называется «теорема о редукции»).

Пусть $\mathcal{B} = \{0, 1\}$, $\mathcal{A} = \{a_1, \ldots, a_p\}$, $\varphi : a_j \to B_j$ — *оптимальное префиксное кодирование и* $\omega_1 \geq \cdots \geq \omega_p$. *Пусть теперь* $\mathcal{C} = \{c_1, \ldots, c_{p+1}\}$ *с частотами* $\omega_1 \geq \ldots \omega_{s-1} \geq \omega_{s+1} \geq \omega_p \geq \omega' \geq \omega''$ *(естественно,* $\omega_s = \omega' + \omega''$*). Тогда кодирование*

$$\varphi_1(c_j) = B_j,\ 1 \leq j \leq s-1, \quad \varphi_1(c_j) = B_{j+1},\ s \leq j \leq p-1,$$
$$\varphi_1(c_p) = B_s 0, \quad \varphi_1(c_{p+1}) = B_s 1$$

тоже является префиксным оптимальным.

Префиксность очевидна. Допустим φ_1 не оптимально. Тогда найдется оптимальное префиксное кодирование $\psi_1 : a_j \to C_j$, $1 \leq j \leq p+1$. По утверждению 29, можно считать, что

$C_p = C'\alpha$ и $C_{p+1} = C'\overline{\alpha}$. Составим кодирование $\psi : a_j \to C_j$ для $1 \le j < s$, $\psi : a_j \to C_{j-1}$ для $s < j \le p$, $\psi(a_s) = C'$. Получим

$$c(\psi) = c(\psi_1) - \omega' - \omega'' < c(\varphi_1) - \omega' - \omega'' = c(\varphi),$$

т.е. φ было не оптимально. Противоречие. □

Лекция 13. Самокорректирующиеся коды.

Будем рассматривать коды в алфавите $\mathcal{B} = \{0, 1\}$.

Определение 80. *Пусть после кодирования код равен* $x = (\beta_1, \ldots, \beta_n)$. *Код* $\tilde{x} = (\beta_1, \ldots, \beta_{i-1}, \overline{\beta_i}, \beta_{i+1}, \ldots, \beta_n)$ *назовем кодом* x *с одной ошибкой в разряде* i.

Определение 81. *Пусть* x *и* y *— два слова в алфавите* $\mathcal{B}$ *одинаковой длины* n. *Расстоянием Хэмминга* $\rho(x, y)$ *назовем число* $\sum_{i=1}^{n} |\beta_i - \gamma_i|$.

Очевидно, что расстояние Хэмминга — это число различающихся букв в словах B и C.

Пример 47. *Пусть* $x = (1000\,0101)$, $y = (0101\,1100)$. *Тогда* $\rho(x, y) = 5$.

Определение 82. *Замкнутым шаром с центром* x *и радиусом* r *называют множество* $Bl(x, r) = \{y : \rho(x, y) \le r\}$. *Сферой с центром* x *и радиусом* r *называют множество* $Sp(x, r) = \{y : \rho(x, y) = r\}$.

Пример 48. *Пусть* $x = (1000\,0101)$. *Тогда шар* $B(x, 1)$ *состоит из центра шара — вектора* x, *и векторов* $(0000\,0101)$, $(1100\,0101)$, $(1010\,0101)$ *и т.д. Сфера* $S(x, 1)$ *состоит из тех же векторов (кроме вектора* x*).*

Ясно, что если у нас есть алгоритм кодирования, который может создать два слова x и y, отличающиеся только одним символом (т.е. $\rho(x, y) = 1$), то наше кодирование неустойчиво к ошибкам. На самом деле, даже если алгоритм может создать два слова x и y, отличающиеся ровно двумя символами (т.е. $\rho(x, y) = 2$), то кодирование все равно не устойчиво.

Пример 49. *Пусть* $\varphi(a) = 0$, $\varphi(b) = 10$, $\varphi(c) = 111$. *Закодируем слово* (ab), *получим слово* (010). *Закодируем слово* (c), *получим* (111). *Пусть при передаче первого слова произошла всего одна ошибка и получилось слово* (110). *Пусть при передаче второго слова тоже произошла всего одна ошибка и получилось слово* (110). *Теперь получатель догадывается, что сообщение содержит ошибку, но раскодировать его не может!*

Далее будем предполагать, что наше кодирование равномерно. Пусть у нас p кодовых слов B_i и каждое имеет длину l.

Определение 83. *Говорят, что код исправляет* r *ошибок, если при наличии в произвольном переданном слове не более чем* r *ошибок можно однозначно восстановить передачу.*

Утверждение 30. *Если $\rho(B_i, B_j) \geq 2r+1$ (при $i \neq j$), то код исправляет r ошибок. Если найдутся два кодовых слова с расстоянием $2r$ или меньше, то код может не исправить r ошибок.*

Доказательство. Пусть $\rho(B_i, B_j) \geq 2r+1$. Тогда шар с центром B_i радиуса r не пересекается с шаром с центром B_j радиусом r. Действительно, если бы какой–то вектор x лежал в обоих шарах, то его можно было бы получить из вектора B_i изменением $\leq r$ координат. Аналогично, изменив еще $\leq r$ координат, можно было бы получить вектор B_j. Но тогда вектора B_i и B_j отличались бы в $\leq 2r$ координатах. Противоречие. Теперь алгоритм раскодирования становится ясен. Пусть получено некоторое слово длины nl. Разбиваем его на куски длины l. Для каждого куска смотрим, в каком из шаров $Bl(B_j, r)$ этот кусок лежит. Центр шара и есть искомое кодовое слово для каждого куска.

Обратно. Пусть есть два кодовых слова такие, что $\rho(B_i, B_j) = \rho \leq 2r$. Тогда они отличаются в каких–то ρ координатах. Назовем эти координаты *выделенными*. Возьмем слово B_i. Передадим его с ошибками в r первых по номеру выделенных координатах (назовем переданный вектор x). Тогда $\rho(B_i, x) = r$. Вектор x отличается от B_j в оставшихся выделенных координатах. Их осталось $\rho - r \leq r$, т.е. $\rho(x, B_j) \leq r$. Тогда получаетель не сможет выяснить — передано ли слово B_i с r ошибками или B_j с $\rho - r$ ошибками. □

Определение 84. *Говорят, что код замечает r ошибок, если при наличии в произвольном переданном слове не более чем r ошибок можно уверенно утверждать, что ошибки при передаче были (или их не было).*

Пример 50. *Пусть $\varphi(a) = 000$, $\varphi(b) = 110$, $\varphi(c) = 101$. Этот код обнаруживает 1 ошибку, но не может ее исправить. Действительно, в правильно составленном сообщении четное число единиц. Если произошла одна ошибка, то мы получим сообщение с нечетным числом единиц и заметим наличие ошибки. А исправить не сможем. Например, получив сообщение (010) мы не знаем — это слово B_1 с ошибкой во втором разряде или это B_2 с ошибкой в первом.*

Утверждение 31. *Если $\rho(B_i, B_j) \geq r+1$ (при $i \neq j$), то код замечает r ошибок. Если найдутся два кодовых слова с расстоянием r или меньше, то код может не заметить r ошибок.*

Доказательство. Пусть $\rho(B_i, B_j) \geq r+1$. Тогда каждый из шаров $B(B_j, r)$ содержит ровно одно кодовое слово — свой центр. Теперь алгоритм распознавания ошибок становится ясен. Пусть получено некоторое слово длины nl. Разбиваем его на куски длины l. Для каждого куска смотрим, совпадает ли он с каким–то кодовым словом. Если да, то ошибки нет и мы знаем раскодировку. Если не совпадает, то ошибка есть.

Обратно. Пусть найдутся два кодовых слова B_i и B_j, которые отличаются в $\rho \leq r$ координатах. Возьмем слово B_i и передадим его с ошибками именно в этих координатах (во всех этих координатах). Получим слово B_j. Теперь получатель можно уже и не заметить, что ошибка была — он получит слово B_j и решит, что передавали именно его, а ошибок не было. □

Итак, если мы хотим получить код, устойчивый к помехам, надо выбирать кодовые слова с достаточно большими попарными расстояниями. Поговорим подробнее про коды, исправляющие одну ошибку.

Определение 85. *Пусть $M_1(n)$ — максимальное количество кодовых слов длины n, подобранных так, чтобы код мог исправить одну ошибку.*

Утверждение 32. $M_1(n) \leq \frac{2^n}{n+1}$.

Доказательство. Как ни подбирай кодовые слова B_i, придется их выбрать так, чтобы шары с центрами в B_i радиуса 1 не пересекались. Тогда число векторов в объединении шариков $\leq 2^n$ (общее число двоичных векторов длины n). В каждом таком шарике ровно $n+1$ вектор. Тогда число шаров, умноженное на $n+1$ не больше 2^n. □

Алгоритм Хэмминга. Вместо того, чтобы придумывать устойчивую кодировку, будем добавлять к сообщению контрольные биты. Пусть дано сообщение в алфавите $\{0, 1\}$ длины n. Как его закодировать:
1. Найдем такое минимальное число k, чтобы $2^k - k > n$.
2. Составим вектор длины $n+k$. Занумеруем его биты, начиная с единицы. Те биты, номера которых есть степени двойки, назовем контрольными. Так как $k + n < 2^k$, то получим ровно k контрольных бит. Остальные n бит назовем информационными.
3. В информационные биты запишем наше сообщение. В контрольные биты запишем нули.
4. Каждый номер бита запишем в двоичной системе. Получим таблицу номеров.
5. Составим контрольные суммы $S_1, \ldots, S_k$ по правилу: в сумму с номером j включаем те биты, у номера которых j–ая цифра равна 1 (цифры нумеруем справа налево).
6. Запишем сумму S_i в i–й контрольный бит (сумму считаем по модулю два).
Как раскодировать сообщение:
1. Вычислим контрольные суммы $S'_1, \ldots, S'_k$ по правилу: в сумму с номером j включаем те биты, у номера которых j–ая цифра равна 1 (цифры нумеруем справа налево) .
2. Если все контрольные суммы равны нулю (сумму считаем по модулю два), то ошибок при передаче не было, ничего исправлять не надо.
3. Если есть ненулевая контрольная сумма, то составим двоичное число, j–ая цифра которого (цифры нумеруем справа налево) равна S'_j.
4. Изменим бит с получившимся номером (в нем при передаче была ошибка).
5. Отбросим контрольные биты и прочтем информационные — получим исходное сообщение.

Пример 51. *Пусть дано сообщение* (101101). *Видим, что* $m = 6$, *тогда* $k = 4$. *Контрольные биты имеют номера* 1, 2, 4 *и* 8. *Информационные биты имеют номера* 3, 5, 6, 7, 9 *и* 10. *Составим таблицу. Разложим номера в двоичную запись. Подпишем разложение под каждым номером*

0	***0***	*1*	***0***	*0*	*1*	*1*	***0***	*0*	*1*
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
0	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*1*	*1*
0	*0*	*0*	*1*	*1*	*1*	*1*	*0*	*0*	*0*
0	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*1*
1	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*

Найдем контрольные суммы $S_1 = x_1+x_3+x_5+x_7+x_9 = 0$, $S_2 = x_2+x_3+x_6+x_7+x_{10} = 0$, $S_3 = x_4+x_5+x_6+x_7 = 0$, $S_4 = x_8 + x_9 + x_{10} = 1$. *Запишем эти суммы в контрольные биты. Получим сообщение* (0010011101).

Теорема 31. *Алгоритм Хэмминга исправляет* 1 *ошибку.*

Доказательство. Заметим, что после кодирования контрольные суммы равны $S'_j = S_j + S_j = 0(mod 2)$. Значит, если при передаче ошибок не было, то все $S'_j = 0$. Пусть произошла одна ошибка при передаче бита с номером s. Запишем число s в двоичной системе. Если его j–ая цифра равна ноль, то этот бит не входит в сумму S'_j. Тогда эта сумма от ошибки не изменится и будет равна нулю. Если j–ая цифра равна единице, то этот бит входит в сумму S'_j. Она была равна нулю, но после ошибки изменилась на 1 и теперь равна 1. Как бы то ни было, S'_j совпадает с j–ой цифрой номера s. Составив двоичное число, j–ая цифра которого (цифры нумеруем справа налево) равна S'_j, мы получим s и найдем ошибочный бит. □

Пример 52. *Пусть сообщение из предыдущего примера получено с одной ошибкой:* (0010111101). *Вычислим контрольные суммы* $S'_1 = x_1 + x_3 + x_5 + x_7 + x_9 = 1$, $S'_2 = x_2 + x_3 + x_6 + x_7 + x_{10} = 0$, $S'_3 = x_4 + x_5 + x_6 + x_7 = 1$, $S'_4 = x_8 + x_9 + x_{10} = 0$. *Видим, что ошибка была и была она в бите с номером* $\overline{0101}_2 = 5$. *Исправляем бит* 5, *получаем слово* (0010011101). *Отбрасываем контрольные биты, получаем* (101101).

Утверждение 33. $M_1(n) \geq \frac{2^n}{2n}$.

Доказательство. Алгоритм Хэмминга позволяет нам кодировать произвольное информационное сообщение длины n словами длины $n + k$ (добавляем k контрольных бит). Получаем 2^n информационных сообщений для $M_1(n + k)$, т.е. $M_1(n + k) \geq 2^n$. Обозначим $m = n + k$, заметим, что $k \leq \log_2(n + k) + 1$, откуда

$$M_1(m) \geq 2^n = 2^{m-k} \geq \frac{2^m}{2m}.$$

□

Лекция 14. Конечные автоматы.

Сначала дадим неформальное определение. *Конечный автомат*

$$\longrightarrow \boxed{\text{A}} \longrightarrow$$

— это устройство (черный ящик), которое имеет *входной канал* и *выходной канал*. Автомат работает дискретно — обработка информации производится в дискретные моменты времени (*такты*). В каждый момент времени автомат находится в одном из *состояний*. За каждый такт времени автомат
– принимает пакет информации по входному каналу;
– переходит в новое состояние;
– выдает пакет информации по выходному каналу.

Для того, чтобы задать автомат, необходимо:
– задать все возможные входящие пакеты информации;
– задать все возможные состояния автомата;
– задать все возможные выходящие пакеты информации;
– задать функцию, которая по входящей информации и текущему состоянию автомата

формирует новое состояние;
– задать функцию, которая по входящей информации и текущему состоянию автомата формирует выхлдящий пакет информации.

Для того, чтобы автомат начал работать, надо еще задать его *начальное состояние*. Мы будем считать наши автоматы *конечными*, т.е. будем предполагать, что есть лишь конечное число входящих пакетов, состояний и выходящих пакетов.

Определение 86. *Автомат — это набор из пяти объектов* $\mathbf{A} = \{\mathcal{A}, \mathcal{B}, \mathcal{Q}, \varphi, \psi\}$. *Здесь* $\mathcal{A} = \{a_1, \ldots, a_p\}$ *— входной алфавит, где каждая буква — это один из возможных входящих информационных пакетов. Аналогично,* $\mathcal{B} = \{b_1, \ldots, b_q\}$ *— выходной алфавит,* $\mathcal{Q} = \{Q_1, \ldots, Q_r\}$ *— алфавит состояний автомата. Функция* $\varphi(x, q)$ *каждой паре* $x \in \mathcal{A}$ *и* $q \in \mathcal{Q}$ *сопоставляет некоторое состояние из* $\mathcal{Q}$. *Функция* $\psi(x, q)$ *каждой паре* $x \in \mathcal{A}$ *и* $q \in \mathcal{Q}$ *сопоставляет некоторую букву выходного алфавита* $\mathcal{B}$.

Пример 53. *Пусть* $\mathcal{A} = \mathcal{B} = \mathcal{Q} = \{0, 1\}$, *а булевы функции* φ *и* ψ *заданы таблицей*

x	q	φ	ψ
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

или формулами $\varphi(x, q) = x$, $\psi(x, q) = q$. *Такой автомат называют задержкой.*

Автомат задержки есть ячейка памяти, в которой может лежать 0 или 1. Работает задержка так: если в ячейке лежит какое–то число q, а на вход пришло другое число x, то автомат кладет число x в ячейку памяти, а число q вынимает из ячейки и отправляет на выход.

При работе автомата на вход ему последовательно (в каждый такт времени) подаются буквы алфавита $\mathcal{A}$. Автомат в каждый такт времени меняет (или не меняет) свое состояние и выдает на выход символ алфавита $\mathcal{B}$. Соответствующие последовательности записывают в виде $a(1)a(2)a(3)\ldots a(t)$, $Q_0Q(1)Q(2)\ldots Q(t)$, $b(1)b(2)\ldots b(t)$.

Вот, например, листинг работы автомата задержки с начальным состоянием $Q_0 = 0$: входная строка $x = (0, 1, 0, 1)$, строка состояний $z = (0, 0, 1, 0, 1)$, выходная строка $y = (0, 0, 1, 0)$.

Пример 54 (двоичный сумматор последовательного действия)**.** *Пусть* $\mathcal{A} = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$, $\mathcal{Q} = \{0, 1\}$, $\mathcal{B} = \{0, 1\}$, $\varphi(x, q) = x_1x_2 \vee x_1q \vee x_2q$, $\psi(x, q) = x_1 \oplus x_2 \oplus q$.

Начальное состояние сумматора всегда 0. На вход сумматору подают пары из нулей и единиц. При этом двоичное число $x_1(t)x_1(t-1)\ldots x_1(1)$ трактуется как первое слагаемое, а $x_2(t)x_2(t-1)\ldots x_2(1)$ — второе. Сумматор принимает каждый раз две очередные цифры складываемых чисел, смотрит, был ли перенос из предыдущего разряда (он хранит его в своем состоянии) и складывает эти три числа. Понятно, что при сложении трех чисел из множества $\{0, 1\}$ мы можем получить двоичные числа 00, 01, 10 и 11. Сумматор выдает на выход последнюю цифру суммы, а первую цифру запоминает.

Вот пример работы сумматора. Пусть надо сложить числа $7 = 111_2$ и $11 = 1011_2$. Составим входной вектор $x = (11, 11, 10, 01, 00)$ — объединяем цифры слагаемых попарно, записываем справа налево и предусматриваем дополнительный разряд, поскольку при суммировании он может понадобиться. Тогда $z = (0, 1, 1, 1, 1, 0)$, $y = (0, 1, 0, 0, 1)$, т.е. результат сложения есть число $10010_2 = 18$.

Упражнение 19. *Постройте диаграмму Мура сумматора.*

Легко видеть, что работа автомата подчинена *канонической системе уравнений*

$$\begin{cases} z(0) = q_0, \\ z(t) = \varphi(x(t), z(t-1)), \qquad t = 1, 2, \ldots. \\ y(t) = \psi(x(t), z(t-1)), \end{cases}$$

Заметим, что

$$y(1) = \psi(x(1), q_0), \qquad y(2) = \psi(x(2), z(1)) = \psi(x(2), \varphi(x(1), q_0)),$$
$$y(3) = \psi(x(3), z(2)) = \psi(x(2), \varphi(x(2), z(1))) = \psi(x(2), \varphi(x(2), \varphi(x(1), q_0))),$$

и так далее: $y(t)$ выражается некоторой функцией от переменных $x(t)$, $x(t-1), \ldots, x(1)$, q_0. Если считать q_0 заданным, то приходим к определению:

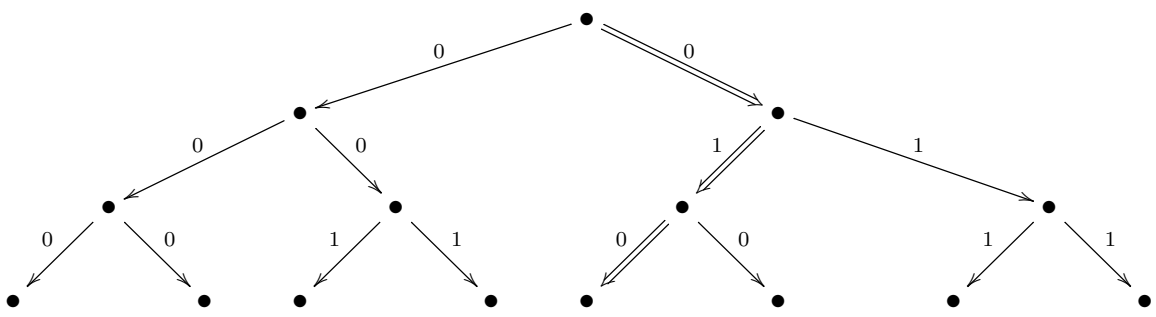
Определение 87. *Пусть A^∞ и B^∞ — множество всех последовательностей с элементами из $\mathcal{A}$ или $\mathcal{B}$, соответственно. Отображение $f : A^\infty \to B^\infty$ называется детерминированной функцией, если координата вектора $\vec{b} = f(\vec{a})$ с номером t определяется только величинами $a(1), \ldots, a(t)$. Более строго: функция f детерминирована, если*

$$\forall t \in \mathbb{N}\, \vec{x} \in A^t,\, \vec{y} \in A^\infty : \; f(\vec{x}\vec{y}) = f(\vec{x})$$

(никакое изменение координат с номерами $\geq t$ не поменяет значения функции).

Мы показали, что каждый автомат порождает детерминированную функцию (точнее, несколько функций — в зависимости от начального состояния). Детерминированную функцию удобно представлять себе функцией вершин *информационного дерева.*

Пример 55. *Пусть $y(1) = 0$, $y(t) = x(t-1)$ для всех $t \geq 2$. Очевидно, что это детерминированная функция, которую порождает автомат задержки с начальным состоянием $q_0 = 0$. Построим ее информационное дерево. Поскольку алфавит $\mathcal{A}$ состоит из двух букв, то дерево будет бинарным, т.е. у каждой вершины — два потомка. Будем нумеровать потомки индексами из нулей и единиц. Теперь сопоставим каждому ребру дерева, ведущему в вершину с индексом $(x_1 x_2 \ldots x_t)$ число $y(t)$ в той строке, которую порождает строка индекса, т.е. в строке $y = f(x)$. В нашем случае $y(t) = x_{t-1}$ и дерево выглядит так:*



На дереве выделен путь, отвечающий входной строке $x = (100\dots)$. При этом строка $y = f(x)$ читается по значениям в вершинах пути $y = (010\dots)$.

Важно то, что *не любое информационное дерево отвечает какой–либо функции, полученной при работе автомата.* Или так: *любой автомат (с заданным начальным состоянием) порождает детерминированную функцию, но не всякая детерминированная функция порождается автоматом.*

Все дело в том, что автомат имеет конечное число состояний $\mathcal{Q} = \{Q_1, \dots, Q_r\}$. Обозначим через f_j — функцию, порождаемую нашем автоматом, который в начальный момент находится в состоянии Q_j.

Определение 88. *Пусть f — детерминированная функция. Пусть A — какое-то слово в алфавите $\mathcal{A}$. Функция g, определенная равенством $g(x) = f(Ax)$ (запись Ax означает конкатенацию слов), называется остаточной функцией для функции f.*

Для произвольной детерминированной функции остаточных функций может быть бесконечно много (они определяются словом A, а таких слов — бесконечное число).

Утверждение 34. *Если детерминированная функция f порождена автоматом, то у нее конечное число остаточных функций.*

Доказательство. Пусть f порождена некоторым автоматом (с каким–то начальным состоянием). Подадим на вход некоторую строку $x = (x(1)x(2)\dots x(n)x(n+1)\dots)$. В момент времени n автомат придет к некоторому внутреннему состоянию Q_j. Тогда его работа в такты времени $n+1, n+2, \dots$ полностью определяется этим его состоянием и строкой $(x(n)x(n+1)\dots)$ (см. каноническую систему). Сделаем сдвиг — положим $\tilde{x} = (\tilde{x}(1)\tilde{x}(2)\dots) = (x(n)x(n+1)\dots)$. Иными словами, если $A = (x(1)x(2)\dots x(n))$, то $x = A\tilde{x}$. Тогда

$$f(x)(t) = f(A\tilde{x})(t) = g(\tilde{x})(t) = f_j(\tilde{x})(t-n+1) \quad \forall t \geq n.$$

Значит функция f имеет не более r различных остаточных функций. □

Этот эффект можно заметить и на информационном дереве. Пусть мы попали в вершину дерева с индексом A. Рассмотрим поддерево с вершинами с индексами Ax. Получим новое дерево (скорее всего, не изоморфное исхрдному). Но если функция порождается автоматом, то число различных попарно неизоморфных поддеревьев конечно.

Определение 89. *Детерминированная функция f называется ограниченно–детерминированной, если у нее конечное число различных остаточных функций. Это число называется весом функции f.*

Утверждение 35. *Любая ограниченно–детерминированная функция может быть порождена некоторым автоматом.*

Доказательство. Создадим автомат собственными руками. Алфавиты $\mathcal{A}$ и $\mathcal{B}$ уже пределены самой функцией f. Пусть f имеет r различных остаточных функций. Каждая такая

функция $g_1, \ldots, g_r$ порождается некоторым словом A по правилу $g_j(x) = f(Ax)$. Для определенности договоримся, что пустое слово порождает функцию g_1, т.е. $g_1(x) = f(x)$. Создадим отображение μ, которое по каждому конечному слову A вычисляет номер j остаточной функции, которую это слово порождает. Зададим алфавит состояний $\mathcal{Q} = \{1, \ldots, r\}$. Начальным состоянием автомата будет 1.

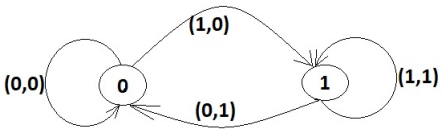
Пусть теперь $\mu(A_j) = j$. Прообразы у каждого состояния наверняка есть. Прообраз может быть один, а может их много (даже бесконечно много). Далее считаем, что A_j произвольный один из таких прообразов. Будем вычислять функцию φ по алгоритму $\varphi(j, a_i) = \mu(A_j a_i)$, т.е. допишем букву a_i к строке A_j и посмотрим, какую остаточную функцию породит такая новая строка. Получается, что при подаче на вход символа a_i автомат переходит из состояния j в $\varphi(j)$. Вторую функцию будем вычислять так: $\psi(j, a_i) = f(A_j a_i)(|A_j| + 1)$, т.е. допишем букву a_i к строке A_j, измерим длину этого слова и вычислим функцию f на этом слове.

Промоделируем работу автомата. По условию, его начальное состояние 1. Пусть задана строка $x = (x(1)x(2)\ldots) \in A^\infty$. Обозначим $y = f(x)$. Получив символ $x(1)$, наш автомат перейдет в состояние $\mu(x(1))$ и выдаст $y(1)$ (все по построению). Получив символ $x(2)$, наш автомат перейдет в состояние $\mu(x(1)x(2))$ и выдаст $y(2)$. И так далее, т.е. наш автомат действительно порождает функцию f. □

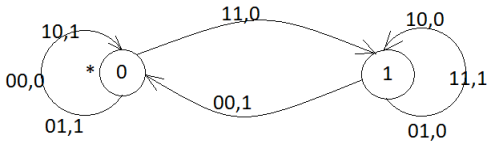
Упражнение 20. *Пусть $y(t) = x(1) \oplus x(2) \oplus \cdots \oplus x(t)$. Составьте информационное дерево этой функции, найдите ее вес, постройте автомат, который эту функцию порождает.*

Работу автомата (любого) удобно изображать *диаграмммой Мура* (нечто сходное с блок–схемой). Это есть граф (ориентированный мультиграф с петлями) с вершинами $Q_1, \ldots, Q_r$. Из каждой вершины выходит p ребер, отвечающих входной информации x. Каждое ребро направлено в вершину $\varphi(x, Q)$. Кроме символа x каждому ребру приписана еще выходная информация $\psi(x, Q)$.

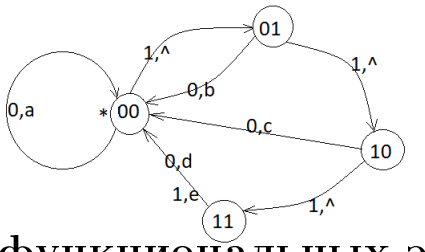
Пример 56. *Диаграма Мура задержки имеет вид*



Пример 57. *Для сумматора тоже достаточное двух вершин (значение бита переноса из предыдущего разряда). Диаграмма Мура сумматора имеет вид*



Пример 58. *Дано алфавитное кодирование $a \mapsto 0$, $b \mapsto 10$, $c \mapsto 110$, $d \mapsto 1110$, $e \mapsto 1111$. Построим автомат, занимающийся раскодированием. Состояниями автомата будет число прочитанных символов для текущей буквы. При чтении очередной буквы, мы можем находиться в самой начале ее кода, можем прочитать один символ, два три символа. Прочитав четыре символа, мы в любом случае раскодируем букву и перейдем к следующей, так что состояние автомата можно записать двоичным числом с двумя разрядами. Диаграма Мура этого автомата имеет вид*



Лекция 15. Схемы функциональных элементов.

Посмотрим, как могут быть устроены автоматы внутри. Зафиксируем алфавит $\mathcal{A} = E_2^n$, т.е. будем кодировать входящую информацию векторами из нулей и единиц с n координатами. Аналогично, положим $\mathcal{B} = E_2^m$ и $\mathcal{Q} = E_2^k$.

Пример 59. *Для автомата задержки* $n = k = m = 1$. *Для сумматора* $n = 2$, $k = m = 1$.

Будем собирать автоматы как конструкторы — из простейших деталей.

Определение 90. *Зафиксируем систему* $\mathrm{B} = \{g_1, g_2, \ldots, g_N\}$ *булевых функций и назовем ее базисом.*

В качестве базисной системы логично выбирать базис в том классе булевых функций, которые придется обрабатывать автомату. Мы далее будем считать, что базис B фиксирован и состоит из трех функций $\{\vee, \wedge, \neg\}$.

Рассмотрим ориентированный граф (мультиграф). Каждую его вершину пометим следующим образом. Если в эту вершину не входит ни одно ребро, то вершину назовем входом и припишем ей некоторую булеву переменную x_i(каждому входу — одну и разным входам — разные). Переменную x_i будем воспринимать как i–ую координата входного вектора x, так что входов у нас будет n.

Если из вершины не выходит ни одно ребро, то такую вершину назовем выходом и припишем ей выходную переменную y_j, $1 \leq j \leq m$. Граф может иметь изолированные вершины. Такие вершины одновременно являются и входом и выходом, т.е. $y_j = x_i$.

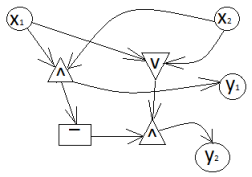
Всем остальным вершинам припишем либо функцию g из базиса B, либо задержку. Для того, чтобы функция g работала корректно, потребуем, чтобы в эту вершину входило столько ребер, сколько аргументов у этой функции (при этом ребра мы нумеруем, чтобы точно знать, какое ребро какому аргументу соответствует). Выходить из каждой вершины может несколько ребер, каждое содержит результат вычисления функции g или выход автомата задержки.

Состояние каждого автомата задержки обозначим z_i, $1 \leq i \leq k$. Все вместе они формируют вектор z состояний автомата.

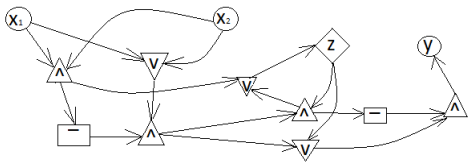
Наш граф должен удовлетворять некоторым естественным требованиям. Например, для каждого входа должен существовать путь с началом в этом входе и концом в каком–то выходе. Для каждого выхода должен существовать путь с началом в каком–то входе и концом в этом выходе. В графе могут быть ориентированные циклы, но каждый такой цикл должен проходить минимум через один автомат задержки.

Определение 91. *Описанный выше граф называют схемой из функциональных элементов и элемента задержки.*

Пример 60. *Построим функциональную схему сумматора. Для этого вначале создадим схему полусумматора*



Видно, что $y_1 = x_1x_2$, *а* $y_2 = \overline{(x_1 \wedge x_2)} \wedge (x_1 \vee x_2) = x_1 \oplus x_2$. *Теперь отправим переменную* y_2 *в блок конъюнкции с состоянием* z *автомата, которое возъмем в блоке задержки. Получим* $y_2 z = (x_1 \oplus x_2)z$. *Этот результат отправим вместе с переменной* y_1 *в блок дизъюнкции и получим* $y_1 \vee (x_1 \oplus x_2)z = x_1x_2 \vee x_1z \vee x_2z$. *Это и есть значение переменной* φ *автомата сумматора. Поместим это значение в элемент задержки. Остается вывести значение функции* $\psi = x_1 \oplus x_2 \oplus z$. *Для этого применим полусумматор к переменным* y_2 *и* z.



Определение 92. *Если в схеме отсутствуют автоматы задержки, то такая схема называется просто схемой функциональных элементов.*

Очевидно, что любая СФЭ вычисляет набор из m булевых функций, каждая из которых зависит от n переменных (некоторые из них могут оказаться фиктивными).

Теорема 32. *Любая СФЭ с задержками определяет некоторый автомат.*

Доказательство. Удалим из схемы элементы задержки и заменим их новыми точками входа $z_1, \ldots, z_k$ и новыми точками выхода $z_1, \ldots, z_k$. Как мы уже знаем, каждый выход есть значение некоторой булевой функции. Т.е. $y_j = \psi_j(x_1, \ldots, x_n, z_1, \ldots, z_k)$ и $z_j = \varphi_j(x_1, \ldots, x_n, z_1, \ldots, z_k)$. Из этих координатных функций составим векторные функции $y = \psi(x, z)$ и $z = \varphi(x, z)$. Теперь вернем элементы задержки на место. Мы доказали, что работа нашей системы подчинена системе уравнений

$$\begin{cases} y(t) = \psi(x(t), z(t-1)), \\ z(t) = \varphi(x(t), z(t-1)). \end{cases}$$

Сдвиг на один такт времени в правой части происходит потому, что элемент задержки выводит значение, полученное им в предыдущем цикле времени. Мы получили каноническую систему уравнений для работы некоторого автомата. □

Утверждение 36. *Для любой булевой функции можно составить схему функциональных элементов в базисе* $\{\vee, \wedge, \neg\}$.

Доказательство. Вспомним, что любая булева функция раскладывается в ДНФ. □

Теорема 33. *Любой автомат можно задать некоторой СФЭ с задержками.*

Доказательство. Перекодируем алфавиты автомата так, чтобы они приняли вид $\mathcal{A} = E_2^n$, $\mathcal{B} = E_2^m$, $\mathcal{Q} = E_2^k$. Представим функцию $\varphi(x, q)$ в координатном виде набором k штук булевых функций $\varphi_i(x_1, \ldots, x_n, q_1, \ldots, q_k)$. Реализуем каждую из них какой–то СФЭ. Теперь поместим на выход каждой схему элемент задержки. Выход i–го элемента соединим с входом нашей схемы с номером $n+i$. Реализуем каждую из булевых функций $\psi_j(x_1, \ldots, q_1, \ldots, q_k)$ какой–то СФЭ, причем выход этой схемы назовем y_j, а на вход с номером $n+i$ вновь подадим выход i–ого элемента задержки. Работа построенной схемы подчинена системе

$$\begin{cases} y(t) = \psi(x(t), z(t-1)), \\ z(t) = \varphi(x(t), z(t-1)), \end{cases}$$

т.е. схема реализует работу исходного автомата. □

Лекция 17. Основные понятия математической логики.

Как определить, что такое «логика»? Наверное, так: «логика — это наука о правильных методах рассуждения». Ясно, что это не есть определение в математическом смысле. Это лишь описание, которому трудно придать строгий смысл. Впрочем, придавать этому определению строгий смысл и не нужно. Каждый, изучающий математическую логику, через некоторое время сам поймет, что это такое.

Гораздо важнее ответить на вопрос, зачем вообще нужна такая наука. Нет, конечно, логика в понимании ее греческой и римской культурой, конечно нужна, тут согласны все. Этот предмет называют просто «логикой» (без приставки «математическая»). От этого часто возникает путаница. Я постараюсь сейчас навести здесь ясность. Начну с классической логики.

Определение 93. *Все наши предложения состоят из слов. Слова делятся на термы, переменные и предикаты.*

Грубо говоря, термы и переменные — это имена существительные, предикаты — это глаголы. Например, «река Волга» — это терм. Действительно, это объект, причем строго определенный. Мы отлично понимаем, о каком конкретном объекте идет речь. «Зонт» — это либо терм, либо переменная. Все зависит от контекста. Если мы уже описали, о каком конкретно зонте идет речь, (в английском языке ставим артикль the), значит терм. Если говорим о зонтах в целом, значит переменная.

Предикат — это функция на термах и переменных. С помощью термов, переменных и предикатов мы составляем высказывания.

Определение 94. *Высказывание — это повествовательное предложение, о котором в данном контексте можно сказать, что оно истинно или ложно, но не то и другое одновременно.*

Пример 61. *«Волга впадает в Каспийское море» — истинное высказывание. При этом контекстом является общепринятые обстоятельства. Например, мы понимаем, что речь идет о реке, а не об автомашине.*

Пример 62. *«Два больше трех» — ложное высказывание. «Идет снег» — высказывание, но его истинность зависит от обстоятельств — надо знать про какое место и какое время мы говорим. «Все крокодилы в Москве реке красного цвета» — истинное высказывание. «С Новым годом!» — не высказывание.*

Пример 63. *«Люди являются братьями» — не высказывание, а предикат! Здесь «люди» — переменная и высказвыание мы получим тогда, когда вместо абстрактных «людей» подставим в наше предложение конкретных людей. При этом мы иногда получим верное, а иногда ложное высказывание.*

Высказывания можно снабжать логическими связками. При этом достаточное трех связок: «И», «ИЛИ», «НЕ». Их таблица истинности вам хорошо известна. Кроме того, можно пользоваться двумя кванторами: «ЛЮБОЙ» и «СУЩЕСТВУЕТ». Смысл их тоже понятен.

Пример 64. *«Существует люди, которые являются братьями» — верное высказывание. «Все люди — братья» — неверное высказывание.*

То, что я сформулировал, составляет начальную часть науки «логики». Дальнейшее более–менее ясно (хотя деталей, конечно, много). Из одних термов составляют другие, более сложные. Например, терм $2+\sqrt{3}$ составлен из термов 2, 3, $+$ и $\sqrt{}$. Говорят о «логике высказываний» или о «языке первого порядка» — здесь вам, я думаю, все ясно.

Говорят о «логике предикатов» или о «языке второго порядка». Это сложнее, но в целом — это наука о булевых функциях. Здесь ключевым является правило логического вывода Modus Ponens: если высказывание A истинно и из A следует B, то B — тоже истинно.

Для нужд математики кроме всего вышесказанного необходимо конкретизировать еще две вещи. Любое математическое утверждение — это какое–то (верное) высказывание. Возьмем теорему «Любой угол, вписанный в окружность и опирающийся на диаметр — прямой». Мы уверены, что это верное высказывание. Но как в этом убедиться? Почему некоторые утверждения, мы считаем истинными, а некоторые нет? Ответ такой — некоторые утверждения кажутся нам настолько очевидными, что мы не требуем объяснений и доказательств. Такие утверждение мы называем аксиомами. Остальные утверждения называем теоремами и доказательство требуем. При этом само доказательство (в правиле Modus Ponens — доказательство истинности импликации) мы понимаем как «ясное убедительное рассуждение, прослушав которое, каждый в него поверит и сможет пересказать его другому».

Я сейчас обрисовал понимание математиков о логике на середину XIX века. Как видите, никакой специальной «математической» логики здесь нет. Дальнейшие события привели математиков к мысли, что математическая логика должна быть точной наукой. Все началось с пятого постулата Евклида — «через каждую точку, не лежащую на прямой, можно провести прямую, параллельную данной и притом только одну». Невозможность проследить за двумя прямыми «до бесконечности» делала эту аксиому в глазах многих

сомнительной. Возник вопрос, нельзя ли вывести ее из других четырех аксиом. Вначале усилия были направлены на поиски этого доказательства. Усилия эти успеха не достигли и к началу XIX века начало возникать подозрение о недоказуемости пятого постулата. Это подозрение перешло в почти полную уверенность после знаменитых работ Лобачевского и Бояи. Эти ученые построили далеко развитые системы геометрии на основе измененного постулата: «через каждую точку, не лежащую на прямой, можно провести несколько прямых, параллельных данной».

В геометрии Лобачевского и Бояи никаких противоречий не появлялось. Это однако не означало, что они не появятся никогда. Для доказательства недоказуемости пятого постулата потребовалась дополнительная работа, завершенная уже в XX веке. Для этого потребовалось существенно уточнить понятие доказательства. Действительно, пока мы, в сущности, не знаем, что такое «доказательство», мы никогда не сможем доказать, что нечто невозможно доказать! Вот в этот момент и появилась математическая логика — точная наука о рассуждениях.

Появившись, логика сразу же расколола математиков на несколько «конфессий» — в какие аксиомы мы верим и какие правила вывода считаем допустимыми. Пессимизма добавили теоремы Геделя о неполноте. Первая из них гласит, что в аксиоматике натуральных чисел обязательно найдется утверждение, которое мы никогда не сможем ни доказать, ни опровергнуть. Если же мы примем это утверждение на веру, т.е. запишем его (а можно — его отрицание) в список аксиом, то найдется следующее утверждение такого же рода. Причем мы не сможем ни доказать, ни опровергнуть его даже учитывая новую аксиому. И так без конца.

На настоящее время, математики (по своей «вере») делятся на классических математиков, конструктивистов и интуиционистов. Классические математики в основу всей математики положили теорию множеств. Вы конечно, слышали о парадоксах «наивной» теории множеств. Парадокс брадобрея: брадобрей бреет тех и только тех, кто не бреется сам; вопрос — бреет ли брадобрей себя? Парадокс Ришара (почитайте литературу, кому интересно) и другие. Значит, нужно строго определить, что такое множество.

Для современных классических математиков множество — это не неопределяемый объект, а объект, удовлетворяющий системе аксиом Цермело–Френкеля (их восемь). При этом большинство классических математиков добавляют к этим восьми еще одну, аксиому выбора. Доказано (в 1963 году Коэном), что эта аксиома не выводима из первых восьми. С другой стороны, некоторые классические математики аксиому выбора отвергают. Связано это с тем, что она приводит к парадоксу Банаха–Тарского. Вы конечно знаете, что можно построить биекцию отрезка на квадрат. Это странно, но все–таки допустимо. А вот то, что можно построить биекцию, которая *сохраняет расстояния между точками* и при этом отображает шар на два таких шара — это уже через чур! С другой стороны, отказавшись от этой аксиомы, мы, например, потеряем эквивалентность определений предела функции по Коши и по Гейне! (желающие могут поискать в доказательстве, где же там используется аксиома выбора).

Еще один краеугольный камень классических математиков — закон исключенного третьего. В список правил вывода добавлено безусловное разрешение на использование доказательства от противного. Основная же проблема классической математики в том, что доказать непротиворечивость аксиоматики Цермело–Френкеля нельзя! (это доказал Гедель).

Точнее, нельзя в рамках самой аксиоматики. А это означает, что никто не застрахован от утверждения, которое удастся доказать вместе со своим отрицанием! Конечно, никто из математиков не верит в возможность такого апокалипсиса, но очень расстраивает то, что нет никаких надежд обрести уверенность!

Гораздо более устойчиво положение конструктивистов. Их система аксиом гораздо проще и понятнее (вспомним, что изначально аксиома — это утверждение, которое не надо доказывать в силу его очевидности). При этом конструктивисты ограничивают правила вывода только алгоритмическими действиями. В качестве таких действия классически берут либо машину Тьюринга, либо нормальные алгоритмы Маркова (доказано, что любой нормальный алгоритм можно реализовать на машине Тбюринга и наоборот, машину Тьюринга можно задать нормальным алгоритмом Маркова). Это приходится добавить, как «принцип конструктивизма». Множество для конструктивиста — это результат работы алгоритма, т.е. не базовый объект.

В общем все хорошо, и очевидность аксиом, и алгоритмичность выводов теорем. Плохо то, что целая куча привычных нам объектов перестает существовать, а еще многие утверждения становятся принципиально недоказуемыми. Например, с точки зрения конструктивистов вещественных чисел не существует. Не то, чтобы совсем. Очень многие существуют (например, все рациональные числа, все корни рациональных степеней из рациональных чисел), но не все! Даже если сформулировать теорему Больцано–Вейерштрасса для последовательностей рациональных чисел, то с точки зрения конструктивиста, она не верна! Все дело в том, что этот предел не всегда можно найти алгоритмически, пусть даже члены последовательности алгоритмичны.

Мы с вами заканчиваем курс дискретной математики. Этот раздел математики имеет дело, в основном, с конечными множествами. Бесконечные множества возникают исключительно в виде потенциальной, но не актуальной бесконечности. Не удивительно, что здесь аксиоматика конструктивной математики очень естественна.

И классичсекая, и конструктивная логика начинаются с построения языка.

Определение 95. *Конечный набор элементарных знаков называется алфавитом.*

При записи букв мы пользуемся *абстракцией отождествления*, говоря, например, что буква a и буква a — это одна и та же буква, записанная дважды. Таким образом, две записанные буквы алфавита могут быть либо одинаковыми, либо различными.

Конечные упорядоченные строки из букв — это *слова* языка. К словам мы тоже применяем абстракцию отождествления.

При записи слов мы можем встретить препятствия: кончится место в строке, не хватит мела, мы устанем, а можем и умереть в процессе записи слова. Эти препятствия можно обходить: кончится место в строке — перейдем на другую строку, мысля новое слово продолжением стараго, мела не хватит — попросим принести кусочек из другой аудитории, жизни не хватит — завещаем потомкам дописать слово. Все это, однако никак не доказывает того, что можно всегда обойти любое препятствие. Тем не менее, мы считаем, что составить можно любое слово. Эта уверенность называется *абстракцией потенциальной осуществимости.*

Условимся обозначать буквы языка малыми латинскими буквами, а слова — заглавными латинскими буквами. Некоторые слова языка мы считаем составленными правильно,

а некоторые — неправильно.

Определение 96. *Конечный набор правил, которые определяют верно составленные слова, называют синтаксисом данного языка. Слова, удовлетворяющие правилам синтаксиса называют формулами языка или высказываниями языка.*

Все правила синтаксиса должны быть заданы явно и недвусмысленно — алгоритмично. Для конкретизации этого понятия алгоритмичности мы вводим *нормальные алгоритмы.*

Определение 97. *Правилом подстановки называют запись вида* $a_1 \dots a_n \to b_1 \dots b_m$, *где* a_j *и* b_j *— какие–то конкретные буквы нашего языка. Нормальным алгоритмом называют конечный упорядоченный список подстановок. При этом некоторые из подстановок могут быть помечены знаком* $\circ$.

По определению, нормальный алгоритм принимает на вход произвольное слово A нашего языка и преобразует его в слово B по правилам.
1) Читать слово A слева направо до тех пор, пока не встретиться подслово, равное левой части первой подстановки.
2) Если такое подслово существует, то заменить это подслово на правую часть подстановки; после этого вернуться к началу нового слова и к началу списка подстановок; иключение — если подстановка помечена знаком $\circ$, то закончить работу.
3) Если такое подслово не существует, то вернуться к началу слова, перейти к следующей подстановке и повторить пнукт 1).
4) Если ни одна подстановка не применима, то закончить работу.

Пример 65. *Пусть алфавит языка состоит из символов* $\{|, +\}$. *Синтаксис языка пуст — любые слова считаются формулами. Рассмотрим нормальный алгоритм*

$$\Big\{ + \to \wedge$$

(символом $\wedge$ *обозначают пустое слово). Несложно видеть, что этот алгоритм занимается сложением натуральных чисел в одноричной системе.*

Уверенность в том, что любой разумный алгортим можно записать в виде нормального алгоритма называют *тезисом Черча.*

Все формулы (синтаксически верные слова) делят на истинные и ложные.

Определение 98. *Конечный набор правил, которые определяют истинные формулы, называют семантикой данного языка.*

Пример 66. *Пусть алфавит языка состоит из букв* $|$, $+$ *и* $=$. *Синтаксис языка состоит из одного правила: каждая формула содержит ровно один знак* $=$. *Семантика языка состоит из одного правила: число знаков* $|$ *слева от знака* $=$ *совпадает с числом знаков* $|$ *после.*

Легко видеть, что мы задали язык равенств натуральных чисел.

То, что мы построили называют языком первого порядка. Построение языка предикатов с использованием кванторов $\forall$ и $\exists$ гораздо сложнее. Я остановлюсь только на одной

особенности, которая также является одним из принципов конструктивизма. Это отказ от закона исключенного третьего «либо утверждение истинно, либо ложно».

Действительно, изучая нашу способность осуществлять конструктивные процессы, мы формулируем утверждения в стиле «мы умеем строить объект, удовлетворяющий требованиям...». Но каково отрицание такого утверждения? Вариант «мы не умеем строить объект, удовлетворяющий требованиям...» не подходит: сейчас не умеем, а потом может научимся. Математические утверждения не должны зависеть от времени! Такая ситуация не может возникнуть в языке первого порядка. Там есть алгоритмически проверяемые правила семантики, т.е. есть алгоритм, который по каждому высказыванию определяет, истинно оно или ложно. Квантор $\exists$ алгоритмичен по определению: если что–то существует, значит мы умеем это что–то строить. Но как быть с квантором $\forall$?

Пример 67. *Рассмотрим предикат: любое натуральное число либо четно, либо нечетно. Этот предикат построен из двух предикатов «число четно» и «число нечетно». Каждый из них конструктивен — в языке с алфавитом $\{|\}$ легко написать алгоритм, проверяющий четность числа*

$$\begin{cases} ||\to \wedge, \\ |\to \textit{нечетно.} \\ \wedge \to \textit{четно.} \end{cases}$$

Просто из вида алгоритма мы понимаем, что он в конце либо выдаст ответ «четно», либо ответ «нечетно» (для строгого доказательства надо применить принцип математической индукции).

Такие предикаты называют *разрешимыми.*

Пример 68. *Рассмотрим другой предикат: любое натуральное число либо четно, либо несовершенно. На настоящий момент никто еще не научился конструировать нечетное совершенное число, но никто и не смог доказать, что такое число нельзя сконструировать. В нашем распоряжении есть только алгоритм, который по каждому данному натуральному числу осуществляет проверку четности и совершенности.*

Если этот алгоритм примет на вход совершенное нечетное число, он его найдет. Если же такого числа в природе вообще нет, то алгоритм «зависнет» — никогда не остановит работу. Такие предикаты называют *полуразрешимыми.*

Таким образом, мы вовсе не уверены в истинности утверждения: «либо A, либо не A», если A есть предикат «существует нечетное совершенное число».