Дискретная математика

Презентации к лекциям (вариант для 2 часов в неделю)

Савицкий Игорь Владимирович

факультет ВМК МГУ

весна 2023

Лекция 1
Булевы функции. Существенные и фиктивные переменные. Формулы. Основные эквивалентности. Разложение по переменным. Совершенная ДНФ

Функции алгебры логики

Определение

Пусть A, B — множества.

- Декартово произведение множеств A и B — это множество всех упорядоченных пар, в которых первый элемент принадлежит A, а второй принадлежит B:

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

- n-я декартова степень множества A — это множество векторов длины n с элементами из A:

$$A^n = \underbrace{A \times A \times \ldots \times A}_{n \text{ раз}} = \{(a_1, \ldots, a_n) \mid a_i \in A,\ i = \overline{1, n}\},\ \ n \geqslant 1.$$

- Если A — конечное множество, то мощность A обозначается $|A|$ и равна количеству элементов A.

Функции алгебры логики

Понятие функции

- Функция (отображение) — это неопределяемое понятие.
- Содержательно функция — это правило f, которое любому подходящему математическому объекту (входу) x сопоставляет математический объект (выход) $f(x)$.
- Если все подходящие входы функции f составляют множество A, а все возможные выходы принадлежат множеству B, то говорят, что функция действует из множества A во множество B:

$$f\colon A \to B.$$

- Нас будут интересовать конкретные множества:
  $E_2 = \{0, 1\}$,
  $E_2^n = \{(a_1, \ldots, a_n) \mid a_1, \ldots, a_n \in \{0, 1\}\}$.

Функции алгебры логики

Определение

- Функция алгебры логики (булева функция) от n переменных — это функция $f\colon E_2^n \to E_2,\ n \in \mathbb{N}$.
- Множество всех булевых функций обозначается P_2.

Табличное и векторное задание булевых функциий

- Булева функция определяется своей таблицей: слева указываются все возможные значения переменных, а справа — значения функции на этих наборах переменных. Например:

x	y	$x \mathbin{\&} y$
0	0	0
0	1	0
1	0	0
1	1	1

- В таблице наборы значений переменных обычно упорядочены стандартно — лексикографически.
- В случае стандартного порядка наборов достаточно задать только вектор функции.
- Например, вектор функции $x \mathbin{\&} y$ — (0001).

Функции алгебры логики

Функции одной переменной

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

- Считаем, что $0 = \text{false}$ (ложь), $1 = \text{true}$ (истина).
- $0, 1$ — константы.
- x — тождественная функция.
- $\overline{x} = \neg x =$ «x не истинно» — отрицание.

Функции алгебры логики

Логические связки

x	y	$x \mathbin{\&} y$	$x \vee y$	$x \sim y$
0	0	0	0	1
0	1	0	1	0
1	0	0	1	0
1	1	1	1	1

- $x \mathbin{\&} y = x \cdot y = xy =$ «x истинно и y истинно» — конъюнкция (умножение по модулю 2).
- $x \vee y =$ «x истинно или y истинно» — дизъюнкция.
- $x \sim y =$ «x истинно тогда и только тогда, когда y истинно» — эквивалентность.

Функции алгебры логики

Логические связки (импликация)

x	y	$x \,\&\, y$	$x \vee y$	$x \sim y$	$x \to y$
0	0	0	0	1	1
0	1	0	1	0	1
1	0	0	1	0	0
1	1	1	1	1	1

- $x \to y = \overline{x} \vee y =$ «если x истинно, то y истинно» — импликация.
- Импликация — это логическое следование, не имеющее отношения к причинно-следственным связям между x и y.

  Если известно, что x и $x \to y$ истинны, то y тоже истинно.

  Но $x \to y$ может быть истинно и не из-за внутренней связи x и y, а, например, потому что y — заведомо истинное утверждение.

Функции алгебры логики

Прочие функции

x	y	$x \,\&\, y$	$x \vee y$	$x \sim y$	$x \to y$	$x \oplus y$	$x \mid y$	$x \downarrow y$
0	0	0	0	1	1	0	1	1
0	1	0	1	0	1	1	1	0
1	0	0	1	0	0	1	1	0
1	1	1	1	1	1	0	0	0

- $x \oplus y = \overline{x \sim y}$ — сложение по модулю 2 (исключающее «или»).
- $x \mid y = \overline{xy}$ — штрих Шеффера.
- $x \downarrow y = \overline{x \vee y}$ — стрелка Пирса.
- Мнемоническое правило для выражения функций $x \mid y$ и $x \downarrow y$: поворачиваем черту на 90° и кладём сверху в виде отрицания. Внизу остаётся либо часть $\vee$ от стрелки (дизъюнкция), либо ничего (конъюнкция).

Функции алгебры логики

Лемма (о числе слов)

В алфавите $A = \{a_1, \ldots, a_r\}$ *из* r *букв можно построить ровно* r^m *различных слов длины* m.

Доказательство

- Зафиксируем любое $r \in \mathbb{N}$ и проведём индукцию по m.
- База индукции $(m = 1)$: слов из одной буквы ровно $r = r^1$.
- Пусть утверждение леммы верно для $m - 1$.
- Шаг индукции. Для построения слова длины m можно взять слово длины $m - 1$ и дополнить его одной буквой.
- По индуктивному предположению всего r^{m-1} различных слов длины $m - 1$. Дополнить слово буквой можно r способами.
- Итого получаем $r^{m-1} \cdot r = r^m$ способов.

$\square$

Функции алгебры логики

Утверждение

Всего существует 2^{2^n} булевых функций от n переменных.

Доказательство

x_1	x_2	$\dots$	x_{n-1}	x_n	$f(x_1, \dots, x_n)$
0	0	$\dots$	0	0	
0	0	$\dots$	0	1	
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
1	1	$\dots$	1	0	
1	1	$\dots$	1	1	

- Функция задаётся таблицей. В её левой части — наборы из E_2^n.
- Наборы идут в лексикографическом порядке. По лемме их 2^n.
- В правой части имеется слово длины 2^n в алфавите $\{0, 1\}$. По лемме получается 2^{2^n} способов заполнить таблицу.

$\square$

Существенные и фиктивные переменные

Определение

Пусть $f(x_1, \dots, x_n) \in P_2$. Переменная x_i существенная, если существуют такие числа $a_1, \dots, a_{i-1}$, $a_{i+1}, \dots, a_n \in \{0, 1\}$, что

$$f(a_1, \dots, a_{i-1}, 0, a_{i+1}, \dots, a_n) \neq f(a_1, \dots, a_{i-1}, 1, a_{i+1}, \dots, a_n).$$

В противном случае переменная x_i фиктивная.

- Фиктивные переменные — это переменные, от значений которых выход функции не зависит.
- Например, у функции $f(x, y, z) = x \vee z$ переменная y является фиктивной, а переменные x, z — существенными.

Существенные и фиктивные переменные

Определение

Функции называются **конгруэнтными**, если одну из них можно получить из другой путём добавления и удаления фиктивных переменных и путём перестановки переменных.

- Например, функции $f(x, y, z) = x \to z$ и $f(x, y) = y \to x$ — это различные, но конгруэнтные функции.

Формулы

- Считаем, что задано счётное множество символов переменных и что для каждой функции из P_2 задан функциональный символ.

Определение (формула)

Пусть $Q \subseteq P_2$.

- Если f — символ функции из Q от k переменных, $x_{i_1}, \dots, x_{i_k}$ — символы переменных, то $f(x_{i_1}, \dots, x_{i_k})$ — формула над Q.
- Если f — символ функции из Q от k переменных, а $\Phi_1, \dots, \Phi_k$ — формулы над Q или символы переменных, то $f(\Phi_1, \dots, \Phi_k)$ — формула над Q.

- Формула — это строка из символов, устроенная по определённым правилам.
- Пример: $Q = \{f(x, y),\ g(x)\}, \quad \Phi = g(f(g(z), f(x, x)))$.

Формулы

Соглашения при записи формул

- Функция может иметь несколько функциональных символов.
- Вместо $\neg(\Phi)$ можно писать $(\neg\Phi)$: $\neg(x) = (\neg x)$.
- Вместо $f(\Phi_1, \Phi_2)$ можно писать $(\Phi_1\ f\ \Phi_2)$: $\&(x, y) = (x \mathbin{\&} y)$.
- Можно опускать скобки с учётом приоритета операций для некоторых стандартных функциональных символов:
  - $\neg$ имеет наибольший приоритет, далее идут $\cdot$ и $\&$, далее все остальные функциональные символы.
  - Операции с одинаковым приоритетом применяются слева направо.
- Функциональный символ $\cdot$ в записи $\Phi_1 \cdot \Phi_2$ можно опускать.
- Вместо $\neg(\Phi)$ можно писать $\overline{(\Phi)}$ или $\overline{\Phi}$.
- Символы функций-констант, можно записывать без переменных: 0 вместо $0(x)$; и 1 вместо $1(x)$.

Формулы

Определение (функция, реализуемая формулой)

Пусть $Q \subseteq P_2$, $(x_1, \dots, x_n)$ — произвольный упорядоченный набор различных символов переменных, а $X = \{x_1, \dots, x_n\}$.

- Если f — символ функции из Q от k переменных, $x_{i_1}, \dots, x_{i_k} \in X$,
  1. То $f(x_{i_1}, \dots, x_{i_k})$ — формула над Q.
  2. Она реализует функцию $h(x_1, \dots, x_n) = f(x_{i_1}, \dots, x_{i_k})$.
- Если f — символ функции из Q от k переменных, а $\Phi_1, \dots, \Phi_k$ — формулы над Q или символы переменных из X,
  1. То $f(\Phi_1, \dots, \Phi_k)$ — формула над Q.
  2. Она реализует функцию $h(x_1, \dots, x_n) = f(g_1(x_1, \dots, x_n), \dots, g_k(x_1, \dots, x_n))$,
  3. где $g_j(x_1, \dots, x_n)$ — функция, реализуемая формулой Φ_j, либо $g_j(x_1, \dots, x_n) = x_{i_j}$, если Φ_j — символ переменной x_{i_j}.

Формулы

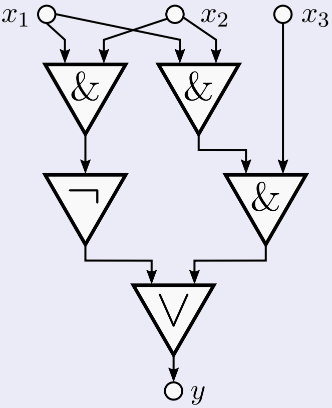
Пример

- Пусть $Q = \{f(x,y),\ g(x)\}$, $\Phi = g(f(g(z), f(x,x)))$.
- Выбираем набор переменных (x,y,z):
  1. Формула $g(z)$ реализует функцию $h_1(x,y,z) = g(z)$.
  2. Формула $f(x,x)$ реализует функцию $h_2(x,y,z) = f(x,x)$. Например, $h_2(0,1,1) = f(0,0)$.
  3. Формула $f(g(z), f(x,x))$ реализует функцию $h_3(x,y,z) = f(h_1(x,y,z), h_2(x,y,z))$.

     Эта запись значит, что $h_3(a_1,a_2,a_3)$ будет равно $f(b_1,b_2)$, где $b_1 = h_1(a_1,a_2,a_3)$, $b_2 = h_2(a_1,a_2,a_3)$.
  4. Формула $g(f(g(z), f(x,x)))$ реализует функцию $h(x,y,z) = g(h_3(x,y,z))$.
- Если $g(x) = \overline{x}$ и $f(x,y) = xy$, то это формула $\overline{\overline{z} \cdot (x \cdot x)}$.

Формулы

Графическое представление формулы



$$y = \overline{x_1 x_2} \vee (x_1 x_2) x_3$$

- Число входных полюсов равно числу различных символов переменных.
- Во входные полюса не входят дуги, но из них может исходить любое число дуг.
- Выходной полюс один, в него входит одна дуга, и из него не исходит дуг.
- В функциональный элемент входит по одной дуге на каждый аргумент его функции, а исходит ровно одна дуга.
- Формула реализует булеву функцию.

Формулы

- Функция, реализуемая формулой, зависит не только от самой формулы, но и от выбранного набора переменных $(x_1, \ldots, x_n)$.

Пример

- Формула над $\{\rightarrow, \vee\}$:
$$\Phi = x_2 \rightarrow x_1.$$
- Формула Φ при наборе (x_1, x_2, x_3) реализует функцию
$$f(x_1, x_2, x_3) = x_2 \rightarrow x_1 \quad (f(x, y, z) = y \rightarrow x).$$
- При наборе (x_2, x_1) формула Φ реализует функцию
$$f(x_2, x_1) = x_2 \rightarrow x_1 \quad (f(x, y) = x \rightarrow y).$$
- Эти функции обязательно конгруэнтны, но могут отличаться порядком переменных и количеством фиктивных переменных.

Формулы

- Каждая формула — это схема построения сложных функций путём подстановки простых функций друг в друга, а также путём переименования переменных.

Определение

Формулы Φ_1 и Φ_2 называются **эквивалентными**, если они реализуют одну и ту же функцию при некотором одинаковом выборе набора переменных. Обозначение: $\Phi_1 = \Phi_2$.

- Формулы $x \to y$ и $\overline{x} \vee y$ эквивалентны.
- Формулы $x \cdot \overline{x}$ и $y \cdot \overline{y}$ эквивалентны (при (x, y) реализуют 0).
- Формулы xy и $x \vee y$ не эквиваленты.
- Формулы $x \vee y$ и $y \vee z$ не эквиваленты:
  нельзя записать $x \vee y = y \vee z$.

Формулы

Замечания

- Булева функция — это вектор значений. Она содержит информацию о количестве своих переменных (определяется по длине вектора), но не содержит информации об их именах.
- Функции $f(x, y) = xy$ и $g(x, z) = xz$ одинаковы (имеют вектор значений (0001)). А функция $h(x, y, z) = xy$ отличается от этих функций (имеет вектор значений $(0000\,0011)$).
- Формула, наоборот, содержит информацию об именах своих переменных, но не о количестве переменных, от которых зависит реализуемая ей функция.
- Формулы $x \cdot y$ и $x \cdot z$ не эквиваленты. Например, при выборе набора (x, y, z) первая формула реализует функцию $(0000\,0011)$, а вторая — $(0000\,0101)$. Аналогично при других наборах.
- Формула $x \cdot y$ может реализовывать любую функцию вида $f(x_1, \ldots, x_n) = x_i x_j$, где $i, j \in \{1, \ldots, n\}$, $i \neq j$, $n \geqslant 2$.

Функции алгебры логики

Основные тождества

- Коммутативность: $xy = yx, \quad x \vee y = y \vee x, \quad x \oplus y = y \oplus x,$
$$x \sim y = y \sim x, \quad x \mid y = y \mid x, \quad x \downarrow y = y \downarrow x.$$
- Ассоциативность: $x \cdot (y \cdot z) = (x \cdot y) \cdot z,$
$$x \vee (y \vee z) = (x \vee y) \vee z, \quad x \oplus (y \oplus z) = (x \oplus y) \oplus z.$$
- Дистрибутивность конъюнкции относительно операций $\vee, \oplus$:
$$x(y \vee z) = xy \vee xz, \quad x(y \oplus z) = xy \oplus xz.$$
- Дистрибутивность дизъюнкции относительно конъюнкции:
$$x \vee yz = (x \vee y)(x \vee z).$$
- Правила де Моргана: $\overline{xy} = \overline{x} \vee \overline{y}, \quad \overline{x \vee y} = \overline{x}\,\overline{y}, \quad \overline{\overline{x}} = x.$

Функции алгебры логики

Основные тождества (продолжение)

- Простейшие поглощения:
  $x \vee x = x, \quad x \vee \overline{x} = 1, \quad x \vee 1 = 1, \quad x \vee 0 = x,$
  $x \cdot x = x, \quad x \cdot \overline{x} = 0, \quad x \cdot 1 = x, \quad x \cdot 0 = 0.$
- Выражение $\neg$ и $\sim$ через $\oplus$:
  $\overline{x} = x \oplus 1, \quad x \sim y = \overline{x \oplus y} = x \oplus y \oplus 1$
- Выражение различных функций через $\&, \vee, \neg$:
  $x \to y = \overline{x} \vee y, \quad x \mid y = \overline{xy}, \quad x \downarrow y = \overline{x \vee y}.$
  $x \oplus y = \overline{x}y \vee x\overline{y}, \quad x \sim y = \overline{x}\,\overline{y} \vee xy.$

Разложение функции по переменным

- Обозначим $x^\sigma = \begin{cases} \overline{x}, & \sigma = 0, \\ x, & \sigma = 1. \end{cases}$
- $x^\sigma = 1$ тогда и только тогда, когда $x = \sigma$.

Теорема (Разложение Шеннона)

Для любой функции $f(x_1, \ldots, x_n) \in P_2$ и любого $k \in \{1, \ldots, n\}$ верно

$$f(x_1, \ldots, x_n) = \bigvee_{\sigma_1, \ldots, \sigma_k \in \{0,1\}} x_1^{\sigma_1} \ldots x_k^{\sigma_k} f(\sigma_1, \ldots, \sigma_k, x_{k+1}, \ldots, x_n)$$

Доказательство

- Рассмотрим произвольный набор $(a_1, \ldots, a_n) \in E_2^n$. Вычислим значение правой части равенства на этом наборе.

$$\bigvee_{\sigma_1, \ldots, \sigma_k \in \{0,1\}} a_1^{\sigma_1} \ldots a_k^{\sigma_k} f(\sigma_1, \ldots, \sigma_k, a_{k+1}, \ldots, a_n)$$

Разложение функции по переменным

Доказательство (продолжение)

$$\bigvee_{\sigma_1,\ldots,\sigma_k\in\{0,1\}} a_1^{\sigma_1}\ldots a_k^{\sigma_k} f(\sigma_1,\ldots,\sigma_k,a_{k+1},\ldots,a_n)$$

- Если $a_i \neq \sigma_i$, то $a_i^{\sigma_i} = 0$, и всё слагаемое равно 0.
- Все слагаемые дизъюнкции будут равны нулю, кроме одного: слагаемого при $\sigma_1 = a_1, \ \ldots, \ \sigma_k = a_k$.
- Получаем $a_1^{a_1}\ldots a_k^{a_k} f(a_1,\ldots,a_k,a_{k+1},\ldots,a_n) = f(a_1,\ldots,a_n)$.
- Значит, на наборе $(a_1,\ldots,a_n)$ правая и левая части равенства совпадают.
- Поскольку совпадение имеет место для любого $(a_1,\ldots,a_n)\in E_2^n$, формулы в левой и правой части равенства эквивалентны.

$\square$

Разложение функции по переменным

Следствие (Разложение по одной перменной)

Для любой булевой функции $f(x_1, \ldots, x_n)$ *верно*

$$f(x_1, \ldots, x_n) = \overline{x}_1 f(0, x_2, \ldots, x_n) \vee x_1 f(1, x_2, \ldots, x_n).$$

Доказательство

- Это частный случай разложения Шеннона при $k = 1$.

□

- Пример: $x \oplus y = \overline{x}(0 \oplus y) \vee x \cdot (1 \oplus y) = \overline{x}y \vee x\overline{y}$.

Совершенная ДНФ

Теорема (о совершенной дизъюнктивной нормальной форме)

Для любой булевой функции $f(x_1, \ldots, x_n) \not\equiv 0$ *верно*

$$f(x_1, \ldots, x_n) = \bigvee_{\substack{\sigma_1, \ldots, \sigma_n \in \{0,1\} \\ f(\sigma_1, \ldots, \sigma_n) = 1}} x_1^{\sigma_1} \cdot \ldots \cdot x_n^{\sigma_n}.$$

Доказательство

- Разложим по всем переменным:

$$f(x_1, \ldots, x_n) = \bigvee_{\sigma_1, \ldots, \sigma_n \in \{0,1\}} x_1^{\sigma_1} \ldots x_n^{\sigma_n} f(\sigma_1, \ldots, \sigma_n)$$

- Слагаемые, в которых $f(\sigma_1, \ldots, \sigma_n) = 0$, нулевые. Исключив их, получим совершенную ДНФ.

$\square$

Совершенная ДНФ

- Совершенная ДНФ — это формула над $\{\&, \vee, \neg\}$, которую можно построить по таблице значений любой функции (кроме 0).
- Каждая строка в таблице, где функция равна 1, задаёт слагаемое совершенной ДНФ. Набор $(\sigma_1, \dots, \sigma_n)$ этого слагаемого — это набор значений переменных данной строки.
- Пример:

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

$$f(x,y) = \overline{x}\,\overline{y} \vee x\overline{y} \vee xy$$

Лекция 2
Полные системы. Полиномы Жегалкина. Замкнутые классы. Классы T_0, T_1, L

Полные системы

Определение

Множество $A \subseteq P_2$ называется **полной системой**, если любая булева функция реализуема формулой над A.

- Очевидно, P_2 является полной системой.

Полные системы

Теорема

Система $\{\vee, \&, \neg\}$ *является полной.*

Доказательство

- Пусть $f(x_1 \ldots, x_n) \in P_2$, $f(x_1 \ldots, x_n) \not\equiv 0$. Тогда f реализуема формулой (совершенной ДНФ) над $\{\vee, \&, \neg\}$.
- Пусть $f(x_1 \ldots, x_n) \equiv 0$. Тогда f реализуется $\overline{x}_1 \cdot x_1$ — формулой над $\{\vee, \&, \neg\}$.

$\square$

Полные системы

Лемма

Пусть $A \subseteq P_2$ — полная система и пусть все функции из A реализуются формулами над B. Тогда B — полная система.

Доказательство (схематично)

- Пусть $f(x_1 \ldots, x_n) \in P_2$. Тогда f реализуется формулой Φ над A.
- Каждую функциональный символ в формуле Φ заменим на реализующую его функцию формулу над B. Получим формулу над B, реализующую f.
- Тогда B — полная система.

$\square$

- Пример замены функционального символа:
  - Имеем выражение $x \vee y = xy \oplus x \oplus y$.
  - Тогда $xy \vee (x \oplus y)$ преобразуется в $(xy)(x \oplus y) \oplus (xy) \oplus (x \oplus y)$.

Полные системы

Теорема

Следующие системы полны в P_2:

1. $\{\vee, \neg\}$;
2. $\{\&, \neg\}$;
3. $\{|\}$;
4. $\{\&, \oplus, 1\}$.

Доказательство

1. Так как $x \cdot y = \overline{\overline{x} \vee \overline{y}}$, а $\{\vee, \&, \neg\}$ — полная система, получаем, что $\{\vee, \neg\}$ — полная система.
2. $x \vee y = \overline{\overline{x} \cdot \overline{y}}$. Получили $\{\vee, \&, \neg\}$.
3. $x \mid y = \overline{x \cdot y}$. Поэтому $\overline{x} = x \mid x$, $x \cdot y = \overline{x \mid y}$. Получили $\{\&, \neg\}$.
4. $\overline{x} = x \oplus 1$. Получили $\{\&, \neg\}$.

$\square$

Полиномы Жегалкина

Определение

- Монотонная элементарная конъюнкция над переменными $x_1, \dots, x_n$ — это формула вида 1 или $u_1 \cdot \ldots \cdot u_k$, где $u_1, \dots, u_k$ — различные переменные из $x_1, \dots, x_n$ без отрицаний.
- Полином Жегалкина над переменными $x_1, \dots, x_n$ — это формула вида 0 или $K_1 \oplus \ldots \oplus K_l$, где $K_1, \dots, K_l$ — различные монотонные ЭК над переменными $x_1, \dots, x_n$.

- Пример: $1 \oplus x_4 \oplus x_1 \oplus x_1x_4 \oplus x_1x_2x_3$.
- Элементарные конъюнкции и полиномы Жегалкина, различающиеся только расстановкой скобок и порядком множителей и слагаемых, считаем одинаковыми.

Полиномы Жегалкина

Теорема

Каждая функция $f(x_1, \ldots, x_n) \in P_2$ *может быть представлена полиномом Жегалкина, причём единственным образом.*

Доказательство (существование полинома Жегалкина)

- Если $f \equiv 0$, то её полином Жегалкина — 0.
- Если $f \not\equiv 0$, то представим f в виде совершенной ДНФ.

$$f(x_1, \ldots, x_n) = \bigvee_{\substack{\sigma_1, \ldots, \sigma_n \in \{0,1\} \\ f(\sigma_1, \ldots, \sigma_n) = 1}} x_1^{\sigma_1} \cdot \ldots \cdot x_n^{\sigma_n}.$$

- Каждая конъюнкция $x_1^{\sigma_1} \ldots x_n^{\sigma_n}$ принимает значение 1 только на наборе $(\sigma_1, \ldots, \sigma_n)$.

Полиномы Жегалкина

Доказательство (существование пол. Жегалкина, продолжение)

$$f(x_1, \ldots, x_n) = \bigvee_{\substack{\sigma_1,\ldots,\sigma_n \in \{0,1\} \\ f(\sigma_1,\ldots,\sigma_n)=1}} x_1^{\sigma_1} \cdot \ldots \cdot x_n^{\sigma_n}.$$

- Это значит, что на каждом наборе α не более одного слагаемого совершенной ДНФ обращается в 1.
- Тогда, поскольку на всех наборах, кроме набора $(1,1)$ функции $x \vee y$ и $x \oplus y$ совпадают, можно заменить $\bigvee$ на $\bigoplus$

$$f(x_1, \ldots, x_n) = \bigoplus_{\substack{\sigma_1,\ldots,\sigma_n \in \{0,1\} \\ f(\sigma_1,\ldots,\sigma_n)=1}} x_1^{\sigma_1} \cdot \ldots \cdot x_n^{\sigma_n}.$$

Полиномы Жегалкина

Доказательство (существование пол. Жегалкина, продолжение)

$$f(x_1, \ldots, x_n) = \bigoplus_{\substack{\sigma_1,\ldots,\sigma_n \in \{0,1\} \\ f(\sigma_1,\ldots,\sigma_n)=1}} x_1^{\sigma_1} \cdot \ldots \cdot x_n^{\sigma_n}.$$

- Заменим далее $\overline{x} = x \oplus 1$ (т.е. $x^\sigma = x \oplus \overline{\sigma}$):

$$f(x_1, \ldots, x_n) = \bigoplus_{\substack{\sigma_1,\ldots,\sigma_n \in \{0,1\} \\ f(\sigma_1,\ldots,\sigma_n)=1}} (x_1 \oplus \overline{\sigma}_1) \cdot \ldots \cdot (x_n \oplus \overline{\sigma}_n).$$

- Раскрывая скобки ($(x \oplus y)z = xz \oplus yz$) и приводя подобные слагаемые ($x \oplus x = 0$), получим полином Жегалкина.

Полиномы Жегалкина

Доказательство (единственность полинома Жегалкина)

- Напомним, что элементарные конъюнкции / полиномы Жегалкина, различающиеся только порядком множителей или слагаемых мы считаем одинаковыми.
- Зафиксируем набор переменных $(x_1, \ldots, x_n)$ и рассмотрим монотонные элементарные конъюнкции и полиномы Жегалкина от этих переменных.
- Подсчитаем число различных монотонных элементарных конъюнкций. Сопоставим каждой монотонной ЭК набор из нулей и единиц $(a_1, \ldots, a_n)$, где

$$a_i = \begin{cases} 1, & x_i \text{ входит в ЭК}, \\ 0, & x_i \text{ не входит в ЭК}, \end{cases} \qquad i = \overline{1, n}.$$

Полиномы Жегалкина

Доказательство (единственность пол. Жегалкина, продолжение)

- Тогда число монотонных ЭК от переменных $(x_1, \ldots, x_n)$ равно числу строк длины n в алфавите $\{0, 1\}$, то есть, 2^n.
- Пронумеруем все монотонные ЭК и сопоставим каждому полиному Жегалкина набор из нулей и единиц $(b_1, \ldots, b_{2^n})$, где

$$b_i = \begin{cases} 1, & \text{ЭК с номером } i \text{ входит в полином,} \\ 0, & \text{ЭК с номером } i \text{ не входит в полином,} \end{cases} \qquad i = \overline{1, 2^n},$$

  а полиному Жегалкина 0 соответствует набор из 2^n нулей.
- Тогда число полиномов Жегалкина от переменных $(x_1, \ldots, x_n)$ равно числу строк длины 2^n в алфавите $\{0, 1\}$, то есть, 2^{2^n}.

Полиномы Жегалкина

Вектор коэффициентов полинома: пример

x_1	x_2	x_3	x_4	ЭК	Вектор полинома
0	0	0	0	1	1
0	0	0	1	x_4	1
0	0	1	0	x_3	0
0	0	1	1	x_3x_4	0
0	1	0	0	x_2	0
0	1	0	1	x_2x_4	0
0	1	1	0	x_2x_3	0
0	1	1	1	$x_2x_3x_4$	0
1	0	0	0	x_1	1
1	0	0	1	x_1x_4	1
1	0	1	0	x_1x_3	0
1	0	1	1	$x_1x_3x_4$	0
1	1	0	0	x_1x_2	0
1	1	0	1	$x_1x_2x_4$	0
1	1	1	0	$x_1x_2x_3$	1
1	1	1	1	$x_1x_2x_3x_4$	0

Полином:
$1 \oplus x_4 \oplus x_1 \oplus x_1x_4 \oplus x_1x_2x_3$

Полиномы Жегалкина

Доказательство (единственность пол. Жегалкина, продолжение)

- Всего существует 2^{2^n} полиномов от переменных $(x_1, \ldots, x_n)$ и 2^{2^n} булевых функций от n переменных.
- Каждый полином Жегалкина реализует ровно одну булеву функцию.
- Кроме того, в первой части доказательства показано, что каждая функция реализуется хотя бы одним полиномом Жегалкина.
- Если какая-либо функция реализуется двумя полиномами Жегалкина, то оставшихся $2^{2^n} - 2$ полиномов не хватит, чтобы реализовать все оставшиеся $2^{2^n} - 1$ функций.
- Поэтому каждая функция реализуется только одним полиномом Жегалкина.

$\square$

Замкнутые классы

Определение

Замыкание $[A]$ множества $A \subseteq P_2$ — это множество всех функций из P_2, которые можно реализовать с помощью формул над A.

Свойства замыкания

1. $A \subseteq [A]$;
2. $A \subseteq B \Rightarrow [A] \subseteq [B]$;
3. $[[A]] = [A]$.

- Первые два свойства очевидны из определения замыкания.
- Третье свойство: в формуле Φ над $[A]$ можно каждый используемый символ заменить на формулу над A. В результате получим формулу над A, эквивалентную Φ.

Замкнутые классы

Определение

Множество $A \subseteq P_2$ — замкнутый класс, если $[A] = A$.

Утверждение

Пусть A — замкнутый класс и $A \neq P_2$. Пусть $B \subseteq A$. Тогда B — не полная система.

Доказательство

- $B \subseteq A \Rightarrow [B] \subseteq [A] = A \neq P_2$.
- Поэтому $[B] \neq P_2$, то есть B — не полная система.

$\square$

Функции, сохраняющие константы

Класс T_0

- $T_0 = \{f(x_1, \ldots, x_n) \in P_2 \colon f(0, \ldots, 0) = 0\}$.

Теорема

Класс T_0 замкнут.

Доказательство

- Пусть $f(y_1, \ldots, y_m) \in T_0$ и $g_1(\tilde{x}_1), \ldots, g_m(\tilde{x}_m) \in T_0$.
- Рассмотрим $h(\tilde{x}) = f(g_1(\tilde{x}_1), \ldots, g_m(\tilde{x}_m))$. Тогда

$$h(0, \ldots, 0) = f(g_1(0, \ldots, 0), \ldots, g_m(0, \ldots, 0)) = f(0, \ldots, 0) = 0.$$

- Если на месте g_i в формуле переменная, то $g_i(\tilde{x}_i) = x_{ik} \in T_0$, и равенство остаётся справедливым.
- То есть, $h(\tilde{x}) \in T_0$. Поэтому класс T_0 замкнут.

$\square$

Функции, сохраняющие константы

Класс T_1

- $T_1 = \{f(x_1, \ldots, x_n) \in P_2 \colon f(1, \ldots, 1) = 1\}$.

Теорема

Класс T_1 замкнут.

Доказательство

- Пусть $f(y_1, \ldots, y_m) \in T_1$ и $g_1(\tilde{x}_1), \ldots, g_m(\tilde{x}_m) \in T_1$.
- Рассмотрим $h(\tilde{x}) = f(g_1(\tilde{x}_1), \ldots, g_m(\tilde{x}_m))$. Тогда

$$h(1, \ldots, 1) = f(g_1(1, \ldots, 1), \ldots, g_m(1, \ldots, 1)) = f(1, \ldots, 1) = 1.$$

- Если на месте g_i в формуле переменная, то $g_i(\tilde{x}_i) = x_{ik} \in T_1$, и равенство остаётся справедливым.
- То есть, $h(\tilde{x}) \in T_1$. Поэтому класс T_1 замкнут.

$\square$

Линейные функции

Класс L

- Булева функция $f(x_1, \dots, x_n)$ называется линейной, если её можно представить в виде $f(x_1, \dots, x_n) = a_0 \oplus a_1 x_1 \oplus \dots \oplus a_n x_n$, где $a_0, \dots, a_n \in \{0, 1\}$.
- L — класс всех линейных булевых функций.

Теорема

Класс L замкнут.

Доказательство

- Линейная функция — это константа, либо сумма нескольких переменных и, возможно, единицы.
- Подставляя такие выражения друг в друга и приводя подобные, будем получать выражения такого же вида.

$\square$

Лекция 3
Классы S, M. Лемма о несамодвойственной функции. Лемма о немонотонной функции

Самодвойственные функции

Определение

Пусть $f(x_1,\dots,x_n)\in P_2$. Тогда функцией, двойственной к f, называется функция $f^*(x_1,\dots,x_n)=\overline{f(\overline{x}_1,\dots,\overline{x}_n)}$.

Примеры

- $(x\vee y)^* = \overline{\overline{x}\vee\overline{y}} = \overline{\overline{x}}\cdot\overline{\overline{y}} = xy.$
- $(xy)^* = \overline{\overline{x}\,\overline{y}} = \overline{\overline{x}}\vee\overline{\overline{y}} = x\vee y.$
- $0^* = 0^*(x) = \overline{0}(\overline{x}) = \overline{0} = 1, \quad 1^* = 0.$
- $(x\oplus y)^* = \overline{\overline{x}\oplus\overline{y}} = (x\oplus 1)\oplus(y\oplus 1)\oplus 1 = x\oplus y\oplus 1 = x\sim y.$

Самодвойственные функции

Утверждение

Для любой булевой функции $f(x_1, \ldots, x_n)$ *верно*

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

Доказательство

- $f^{**}(x_1, \ldots, x_n) = \overline{f}^{*}(\overline{x}_1, \ldots, \overline{x}_n) = \overline{\overline{f}}(\overline{\overline{x}}_1, \ldots, \overline{\overline{x}}_n) = f(x_1, \ldots, x_n).$

$\square$

Самодвойственные функции

Теорема (Принцип двойственности)

Пусть $h(x_1, \ldots, x_n) = f(g_1(x_{11}, \ldots, x_{1n_1}), \ldots, g_m(x_{m1}, \ldots, x_{mn_m}))$.
Тогда $h^*(x_1, \ldots, x_n) = f^*(g_1^*(x_{11}, \ldots, x_{1n_1}), \ldots, g_m^*(x_{m1}, \ldots, x_{mn_m}))$.

Доказательство

- Напомним: $f^*(y_1, \ldots, y_m) = \overline{f}(\overline{y}_1, \ldots, \overline{y}_m)$.
- Имеем

$$\begin{aligned} h^*(x_1, \ldots, x_n) &= \overline{h}(\overline{x}_1, \ldots, \overline{x}_n) = \\ &= \overline{f}(g_1(\overline{x}_{11}, \ldots, \overline{x}_{1n_1}), \ldots, g_m(\overline{x}_{m1}, \ldots, \overline{x}_{mn_m})) = \\ &= \overline{f}(\overline{g}_1^*(x_{11}, \ldots, x_{1n_1}), \ldots, \overline{g}_m^*(x_{m1}, \ldots, x_{mn_m})) = \\ &= f^*(g_1^*(x_{11}, \ldots, x_{1n_1}), \ldots, g_m^*(x_{m1}, \ldots, x_{mn_m})). \end{aligned}$$

□

Самодвойственные функции

Следствие

Пусть $\Phi(f_1, \dots, f_n)$ *— формула, содержащая только функциональные символы функций* $f_1, \dots, f_n$ *и реализующая функцию* h.
Тогда формула $\Phi(f_1^*, \dots, f_n^*)$ *реализует функцию* h^*.

- Пример: $((x \oplus y) \vee z)^* = (x \sim y)z$.

Самодвойственные функции

Определение

- Функция $f \in P_2$ называется самодвойственной, если $f^* = f$.
- Класс самодвойственных функций обозначается S.

Примеры

- $(x)^* = \overline{(\overline{x})} = x$.
- $(\overline{x})^* = \overline{(\overline{\overline{x}})} = \overline{x}$.
- $x \oplus y \oplus z$.
- Медиана: $\mathrm{m}(x, y, z) = xy \vee xz \vee yz = xy \oplus xz \oplus yz$.

Самодвойственные функции

Утверждение

Класс S замкнут.

Доказательство

- Пусть $f(y_1, \ldots, y_m) \in S$ и $g_1(\tilde{x}_1), \ldots, g_m(\tilde{x}_m) \in S$.
- Рассмотрим $h(\tilde{x}) = f(g_1(\tilde{x}_1), \ldots, g_m(\tilde{x}_m))$. Тогда по принципу двойственности

$$h^*(x_1, \ldots, x_n) = f^*(g_1^*(x_{11}, \ldots, x_{1n_1}), \ldots, g_m^*(x_{m1}, \ldots, x_{mn_m})) =$$
$$= f(g_1(x_{11}, \ldots, x_{1n_1}), \ldots, g_m(x_{m1}, \ldots, x_{mn_m})) = h(x_1, \ldots, x_n).$$

- Если на месте g_i в формуле переменная, то $g_i(\tilde{x}_i) = x_{ik} \in S$, и равенство остаётся справедливым.
- То есть, $h(\tilde{x}) \in S$. Поэтому класс S замкнут.

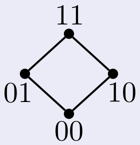
$\square$

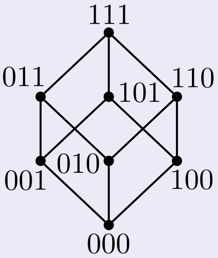
Монотонные функции

Определение

Пусть $\alpha = (a_1, \ldots, a_n)$, $\beta = (b_1, \ldots, b_n) \in E_2^n$. Будем считать, что $\alpha \leqslant \beta$, если $a_i \leqslant b_i$ при всех $i = \overline{1, n}$.

Примеры частичных порядков





$(0,0) \leqslant (0,1) \leqslant (1,1)$

$(0,1)$ и $(1,0)$ не сравнимы

$(0,0,0) \leqslant (0,1,0) \leqslant (0,1,1) \leqslant (1,1,1)$

$(0,1,0)$ и $(1,0,1)$ не сравнимы

Монотонные функции

Определение

- Булева функция $f(x_1, \ldots, x_n)$ называется монотонной если для любых двух наборов $\alpha, \beta \in E_2^n$ верно

$$\alpha \leqslant \beta \implies f(\alpha) \leqslant f(\beta).$$

- Множество всех монотонных булевых функций обозначается M.

Примеры

- $0, 1, x$ — монотонны.
- $\overline{x}$ — немонотонна.
- xy, $x \vee y$ — монотонны.
- $x \oplus y$ немонотонна.

Монотонные функции

Утверждение

Класс M замкнут.

Доказательство

- Добавление / удаление фиктивных переменных и перестановка переменных не влияет на монотонность функции.
- Поэтому в суперпозиции достаточно рассматривать функции с общим набором переменных.
- Пусть $f(y_1, \ldots, y_m) \in M$ и $g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n) \in M$.
- Рассмотрим $h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$.
- Пусть $\alpha, \beta \in E_2^n$, $\alpha \leqslant \beta$. Обозначим $c_i = g_i(\alpha)$ и $d_i = g_i(\beta)$ при $i = \overline{1, m}$. В силу монотонности g_i имеем $c_i \leqslant d_i$.
- Это значит, что $(c_1, \ldots, c_m) \leqslant (d_1, \ldots, d_m)$.

Монотонные функции

Утверждение

Класс M замкнут.

Доказательство (продолжение)

$$g_i(\alpha) = c_i \leqslant d_i = g_i(\beta)$$

- В силу монотонности f имеем

$$h(\alpha) = f(g_1(\alpha), \ldots, g_m(\alpha)) = f(c_1, \ldots, c_m) \leqslant f(d_1, \ldots, d_m) = \\ = f(g_1(\beta), \ldots, g_m(\beta)) = h(\beta).$$

- Если на месте g_i в формуле переменная, то $g_i(\tilde{x}) = x_k \in M$, и все рассуждения остаются справедливыми.
- То есть, $h(\tilde{x}) \in M$. Поэтому класс M замкнут.

$\square$

Теорема Поста

Лемма (о несамодвойственной функции)

Пусть $f(x_1, \ldots, x_n)$ *— булева функция и* $f \notin S$. *Тогда, подставляя в* f *на места всех переменных* x *и* $\overline{x}$, *можно получить одну из функций* $\varphi(x) \equiv 0$ *или* $\varphi(x) \equiv 1$.

Доказательство

- Напомним: $g \in S$, если $g(x_1, \ldots, x_m) = \overline{g}(\overline{x}_1, \ldots, \overline{x}_m)$.
- Пусть $f(x_1, \ldots, x_n) \notin S$. Тогда существуют $a_1, \ldots, a_n \in E_2$ такие, что $f(a_1, \ldots, a_n) = f(\overline{a}_1, \ldots, \overline{a}_n)$.
- Рассмотрим $\varphi(x) = f(x \oplus a_1, \ldots, x \oplus a_n)$. Эта функция получается из f подстановкой x и $\overline{x}$: $x \oplus 0 = x$, $x \oplus 1 = \overline{x}$.
- Ясно, что $\varphi(0) = f(a_1, \ldots, a_n) = f(\overline{a}_1, \ldots, \overline{a}_n) = \varphi(1)$. Значит, $\varphi(x)$ является константой.

$\square$

Теорема Поста

Лемма (о немонотонной функции)

Пусть $f(x_1, \ldots, x_n)$ — булева функция и $f \notin M$. Тогда, подставляя в f на места всех переменных $0, 1$ и x, можно получить функцию $\varphi(x) = \overline{x}$.

Доказательство

- Пусть $f(x_1, \ldots, x_n) \notin M$. Возьмём $a_1, \ldots, a_n,\ b_1, \ldots, b_n \in E_2$ такие, что $a_i \leqslant b_i$ при $i = \overline{1, n}$, а $f(a_1, \ldots, a_n) \nleqslant f(b_1, \ldots, b_n)$.
- Это значит, что $f(a_1, \ldots, a_n) = 1$, а $f(b_1, \ldots, b_n) = 0$.
- Для каждого $i = \overline{1, n}$ в силу $a_i \leqslant b_i$ возможны 3 случая:
  1. Если $a_i = b_i = 0$, то в f подставляем $x_i = 0$;
  2. Если $a_i = 0$, $b_i = 1$, то в f подставляем $x_i = x$;
  3. Если $a_i = b_i = 1$, то в f подставляем $x_i = 1$.
- Получим некоторую функцию $\varphi(x)$. При этом имеем $\varphi(0) = f(a_1, \ldots, a_n) = 1,\ \varphi(1) = f(b_1, \ldots, b_n) = 0$, т.е. $\varphi(x) = \overline{x}$.

$\square$

Лекция 4
Лемма о нелинейной функции. Теорема Поста. Базисы. Предполные классы

Теорема Поста

Лемма (о нелинейной функции)

Пусть $f(x_1, \ldots, x_n)$ *— булева функция и* $f \notin L$*. Тогда, подставляя в* f *на места всех переменных* $0, 1, x, y, \overline{x}, \overline{y}$*, можно получить одну из функций* $\varphi(x, y) = xy$ *или* $\varphi(x, y) = \overline{xy}$*.*

Доказательство

- Рассмотрим полином Жегалкина для $f(x_1, \ldots, x_n)$. Поскольку $f \notin L$, в нём есть нелинейное слагаемое.
- Выберем наименьшее по числу переменных нелинейное слагаемое. Не ограничивая общности, считаем, что это слагаемое $x_1 \ldots x_s$.
- Подставляем: $g(x, y) = f(\underbrace{x, \ldots, x, y}_{s}, 0, \ldots, 0)$.
- Поскольку слагаемое $x_1 \ldots x_s$ минимально, любое другое нелинейное слагаемое содержит переменную из $x_{s+1}, \ldots, x_n$.

Теорема Поста

Доказательство леммы о нелинейной функции (продолжение)

$$g(x, y) = f(\underbrace{x, \ldots, x, y}_{s}, 0, \ldots, 0)$$

- Таким образом, все другие нелинейные слагаемые обнулились.
- Тогда $g(x, y) = xy \oplus ax \oplus by \oplus c$. Выберем $h(x, y) = g(x \oplus b,\ y \oplus a) = f(x \oplus b, \ldots,\ x \oplus b,\ y \oplus a,\ 0,\ \ldots,\ 0)$.
- Поскольку $x \oplus 0 = x,\ \ x \oplus 1 = \overline{x}$, функция $h(x, y)$ получена из f подстановкой $0, 1, x, y, \overline{x}, \overline{y}$.
- Имеем

$$h(x, y) = (x \oplus b)(y \oplus a) \oplus a(x \oplus b) \oplus b(y \oplus a) \oplus c = xy \oplus (ab \oplus c).$$

- В зависимости от a, b, c это либо xy, либо $\overline{xy}$.

□

Теорема Поста

Замечание: «не ограничивая общности»

- Выше было доказано, что для $f \notin L$, у которой наименьшее по числу переменных нелинейное слагаемое имеет вид $x_1 \dots x_s$, выполняется условие леммы о нелинейной функции.
- В общем случае минимальное нелинейное слагаемое может иметь вид $x_{i_1} \dots x_{i_s}$.
- Тогда в выражении $g(x, y) = f(\underbrace{x, \dots, x, y}_{s}, 0, \dots, 0)$ только поменяются местами подстановки: переменная x будет подставлена на места $x_{i_1}, \dots, x_{i_{s-1}}$, а переменная y на место x_{i_s}.
- При этом ход рассуждений сохранится таким же, как и для доказанного случая, и утверждение леммы останется верным.
- В таких ситуациях выбирают удобный случай для иллюстрации доказательства, а «не ограничивая общности рассуждений» значит, что рассуждения применимы и к общему случаю.

Теорема Поста

Теорема (Поста о полноте)

Множество $A \subseteq P_2$ *является полной системой тогда и только тогда, когда оно не содержится целиком ни в одном из классов* T_0, T_1, S, L, M.

Доказательство

- $\Rightarrow$. Пусть N — один из классов T_0, T_1, S, L, M и $A \subseteq N$. Тогда N — замкнутый класс.
- Имеем $A \subseteq N$, тогда $[A] \subseteq [N] = N \neq P_2$. Поэтому A — не полная система.
- $\Leftarrow$. Пусть A не содержится целиком ни в одном из классов T_0, T_1, S, L, M.
- Это значит, что в A есть функции $f_0 \notin T_0,\ f_1 \notin T_1,\ f_S \notin S,\ f_L \notin L,\ f_M \notin M$.

Теорема Поста

Доказательство (продолжение)

$$f_0 \notin T_0,\ f_1 \notin T_1,\ f_S \notin S,\ f_L \notin L,\ f_M \notin M$$

- Покажем, что формулами над выбранными функциями можно получить все функции из P_2.
- а) Получение отрицания $\overline{x}$.
  - Рассмотрим $\varphi_0(x) = f_0(x, \ldots, x) \in [A]$.
  - $f_0 \notin T_0$, поэтому $\varphi_0(0) = f_0(0, \ldots, 0) = 1$. Тогда $\varphi_0(x) \in \{1, \overline{x}\}$.
  - Рассмотрим $\varphi_1(x) = f_1(x, \ldots, x) \in [A]$.
  - $f_1 \notin T_1$, поэтому $\varphi_1(1) = f_1(1, \ldots, 1) = 0$. Тогда $\varphi_1(x) \in \{0, \overline{x}\}$.
  - Если ни одна из функций не равна $\overline{x}$, то $\varphi_0(x) \equiv 1,\ \varphi_1(x) \equiv 0$.
  - Тогда по лемме о немонотонной функции подстановками 0, 1, x получаем из f_M функцию $\overline{x} \in [A]$.
  - В любом из случаев получили $\overline{x} \in [A]$.

Теорема Поста

Доказательство (продолжение)

$$f_0 \notin T_0,\ f_1 \notin T_1,\ f_S \notin S,\ f_L \notin L,\ f_M \notin M, \qquad \overline{x} \in [A]$$

- б) Получение констант 0 и 1.
  - По лемме о несамодвойственной функции подстановками $x, \overline{x}$ в f_S получаем константу: $\varphi(x) \equiv 0$ или $\varphi(x) \equiv 1$.
  - С помощью отрицания получаем вторую константу: $1 = \overline{0}$ или $0 = \overline{1}$.
- в) Получение конъюнкции xy.
  - По лемме о нелинейной функции подстановками $0, 1, x, \overline{x}, y, \overline{y}$ в f_L получаем функцию $\varphi(x, y) = xy$ или $\varphi(x, y) = \overline{xy}$.
  - Во втором случае получаем $xy = \overline{\varphi(x, y)}$.
- Получили $\{xy, \overline{x}\} \subseteq [A]$ — полную систему. Значит, $[A] = P_2$.

$\square$

Базисы

Определение

Множество $A \subseteq P_2$ называется **базисом** (в P_2), если выполнены два условия:

1. A — полная система;
2. Для любой $f \in A$ система $A \setminus \{f\}$ не является полной.

Теорема

Максимальное число функций в базисе в P_2 равно 4.

Доказательство

- Докажем, что из всякой полной системы можно выделить подмножество из не более 4 функций, которая также будет образовывать полную систему.
- Это будет значить, что система из большего числа функций не может являться базисом.

Базисы

Доказательство (в базисе не более 4 функций)

- Пусть A — полная система. Тогда по теореме Поста в A есть функции $f_0 \notin T_0,\ f_1 \notin T_1,\ f_S \notin S,\ f_L \notin L,\ f_M \notin M$.
- Тогда $\{f_0, f_1, f_S, f_L, f_M\}$ — полная система. Если хотя бы две из указанных функций совпадают, то искомая система из не более 4 функций найдена. Пусть все 5 функций различны.
- Рассмотрим $f_0(x_1, \ldots, x_n) \notin T_0$. Верно $f_0(0, \ldots, 0) = 1$.
- Если $f_0 \notin M$, то функцию f_M можно убрать из системы, и система $\{f_0, f_1, f_S, f_L\}$ полна.
- Если $f_0 \in M$, то $f_0(x_1, \ldots, x_n) \equiv 1$. Тогда $f_0 \notin S$, и система $\{f_0, f_1, f_L, f_M\}$ полна.
- В любом случае получаем систему из не более 4 функций.
- Таким образом, в базисе не может быть более 4 функций.

Базисы

Доказательство (существует базис из 4 функций)

- Покажем, что существует базис из 4 функций.
- Рассмотрим $A = \{0, 1, xy, x \oplus y \oplus z\}$.
- $1 \notin T_0,\ 0 \notin T_1,\ 0 \notin S,\ xy \notin L,\ x \oplus y \oplus z \notin M$.
  (последнее верно, т.к. $0, 1 \in M$, а $x \oplus 0 \oplus 1 = \overline{x} \notin M$)
- По теореме Поста система A полна.
- $A \setminus \{0\} \subseteq T_1, \quad A \setminus \{1\} \subseteq T_0,$
- $A \setminus \{xy\} \subseteq L, \quad A \setminus \{x \oplus y \oplus z\} \subseteq M.$
- По теореме Поста каждая из 4-х приведённых выше систем неполна. То есть, при удалении из A любой функции получается не полная система.
- Значит, A является базисом из 4 функций.

$\square$

Базисы

Примеры базисов из разного количества функций

- Базис из одной функции: $\{x \mid y\}$.
- Базис из двух функций: $\{xy, \overline{x}\}$.
- Базис из трёх функций: $\{xy, x \oplus y, 1\}$.
- Базис из четырёх функций: $\{0, 1, xy, x \oplus y \oplus z\}$.

$\square$

Предполные классы

Определение

Множество $A \subseteq P_2$ называется предполным классом, если выполнены два условия:

1. A — не полная система;
2. Для любой $f \notin A$ система $A \cup \{f\}$ является полной.

Теорема

В P_2 существует в точности 5 предполных классов: T_0, T_1, S, L, M.

Доказательство

- Нужно доказать, что указанные пять классов являются предполными и что никаких других предполных классов нет.

Предполные классы

Доказательство (классы не вложены друг в друга)

- Докажем, что никакой из классов T_0, T_1, S, L, M не целиком содержится в другом из этих классов.
- Для каждой пары классов приведём пример функции, которая принадлежит одному из них, но не принадлежит другому.
  В таблице каждая функция принадлежит множествам, указанным в её строке и столбце.

	$P_2 \setminus T_0$	$P_2 \setminus T_1$	$P_2 \setminus S$	$P_2 \setminus L$	$P_2 \setminus M$
T_0		0	0	xy	$x \oplus y$
T_1	1		1	xy	$x \sim y$
S	$\overline{x}$	$\overline{x}$		$\mathrm{m}(x, y, z)$	$\overline{x}$
L	$\overline{x}$	$\overline{x}$	0		$\overline{x}$
M	1	0	0	xy	

- Напомним, что $\mathrm{m}(x, y, z) = xy \vee xz \vee yz = xy \oplus xz \oplus yz$.

Предполные классы

Доказательство (предполнота классов)

- Докажем, что каждый класс $K \in \{T_0, T_1, S, L, M\}$ является предполным.
- $[K] = K \neq P_2$, система K не является полной.
- Пусть $f \notin K$. Поскольку классы не вложены друг в друга, система $K \cup \{f\}$ не содержится целиком ни в одном из 4 других классов. Но, т.к. $f \notin K$, она не содержится целиком и в K.
- По теореме Поста $K \cup \{f\}$ — полная система.

Предполные классы

Доказательство (нет других предполных классов)

- Покажем, что любой предполный класс принадлежит множеству $\{T_0, T_1, S, L, M\}$.
- Пусть A — предполный класс. Значит, A — не полная система.
- По теореме Поста $A \subseteq K$ для некоторого $K \in \{T_0, T_1, S, L, M\}$.
- Предположим, что $A \neq K$. Тогда найдётся $f \in K \setminus A$.
- Получим, что $[A \cup \{f\}] \subseteq K$, то есть (по теореме Поста) $A \cup \{f\}$ — не полная система.
- Получили противоречие тому, что A — предполный класс. Противоречие означает, что $A = K$.
- Значит, $A \in \{T_0, T_1, S, L, M\}$.

$\square$

Прикладные задачи для булевых функций

- Построение эффективных схем для реализации булевых функций.
  - Используется для создания компактных и быстрых микросхем для электронных устройств.
- Анализ задач из класса NP и SAT-solver'ы.
  - Многие прикладные задачи принадлежат классу NP.
  - Для них неизвестен гарантированно быстрый способ решения, но есть методы, которые «обычно» работают быстро на практике.
  - Все задачи из класса NP сводятся к задачам анализа некоторых формул для булевых функций, которые решает SAT-solver.
- Анализ криптографических свойств булевых функций.
  - Преобразование и кодирование информации можно выражать с помощью булевых функций.
  - Сложность расшифровки кода зависит от свойств булевых функций.
  - Например, если используются линейные булевы функции, то код легко расшифровать (решить систему линейных уравнений).

Лекция 5
Графы. Изоморфизм, связность. Деревья

Графы

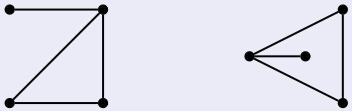
Определение

Граф (простой граф) — это пара $G = (V, E)$, где

- V (множество вершин) — непустое конечное множество.
- $E \subseteq \{\{u, v\} \mid u \neq v,\ u, v \in V\}$ (множество рёбер) — множество неупорядоченных пар различных элементов V.

- Элементы V называют вершинами, а элементы E — рёбрами.
- Ребро $\{u, v\}$ традиционно обозначают (u, v), несмотря на то, что это неупорядоченная пара

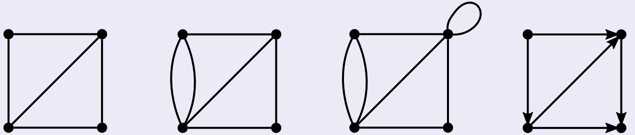
Геометрический способ изображения



Графы

Другие типы графов

- Граф, в котором дополнительно допущено повторение рёбер, называется мультиграфом (повторяющиеся рёбра называются параллельными).
- Мультиграф, в котором дополнительно допускаются пары из одинаковых элементов в качеств рёбер, называется псевдографом (пары из одинаковых элементов называются петлями).
- Граф (мультиграф, псевдограф), в котором рёбра являются упорядоченными парами, называется ориентированным.



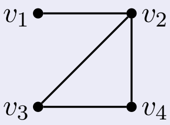
Простой граф Мультиграф Псевдограф Орграф

Степени вершин

Определение

- Две **вершины** графа называются **смежными**, если они входят в одно и тоже ребро (соединены ребром).
- **Вершина** и **ребро** графа называются **инцидентными**, если вершина входит в данное ребро.
- **Степень вершины** $\deg v$ в неориентированном графе — это число инцидентных ей рёбер (в псевдографе петля считается за два инцидентных ребра).

Пример



- Вершины v_1 и v_2 смежны.
  Вершины v_1 и v_3 не смежны.
- Вершина v_1 инцидентна ребру (v_1, v_2).
- $\deg v_1 = 1,\ \deg v_2 = 3,\ \deg v_3 = 2,\ \deg v_4 = 2.$

Степени вершин

Утверждение

Пусть $G = (V, E)$ *— граф с* p *вершинами* $v_1, \ldots, v_p$, *и* q *рёбрами. Тогда*

$$\sum_{i=1}^{p} \deg v_i = 2q.$$

Доказательство

- Каждое ребро графа инцидентно двум вершинам, т.е. добавляет $+1$ к степеням двух вершин графа.
- Значит, каждое из q рёбер добавляет $+2$ к сумме степеней вершин.
- Поскольку сумма степеней формируется только за счёт инцидентных вершинам рёбер, она получается равна $2q$.

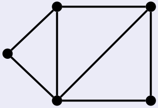
$\square$

Подграфы

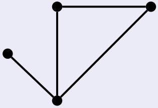
Определение

Граф $G_1 = (V_1, E_1)$ называется **подграфом** графа $G = (V, E)$, если $V_1 \subseteq V$ и $E_1 \subseteq E$.

Пример



G



G_1

Изоморфизм графов

Определение

- Графы $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$ называются **изоморфными**, если существует взаимно-однозначное отображение $\varphi\colon V_1 \to V_2$ такое, что для любых $u, v \in V_1$ верно

$$(u,\, v) \in E_1 \iff (\varphi(u),\, \varphi(v)) \in E_2.$$

- Отображение φ называется **изоморфизмом**.

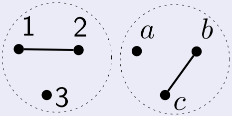
- Изоморфные графы обладают одними и теми же свойствами. В большинстве случаев они считаются одинаковыми.

Изоморфизм графов

Пример

Рассмотрим графы

- $G_1 = (V_1, E_1), \quad V_1 = \{1, 2, 3\},\ E_1 = \{(1,\ 2)\};$
- $G_2 = (V_2, E_2), \quad V_2 = \{a, b, c\},\ E_2 = \{(b,\ c)\}.$



- Можно построить следующий изоморфизм:

$$1 \leftrightarrow c,$$
$$2 \leftrightarrow b,$$
$$3 \leftrightarrow a.$$

Пути в графах

Определение

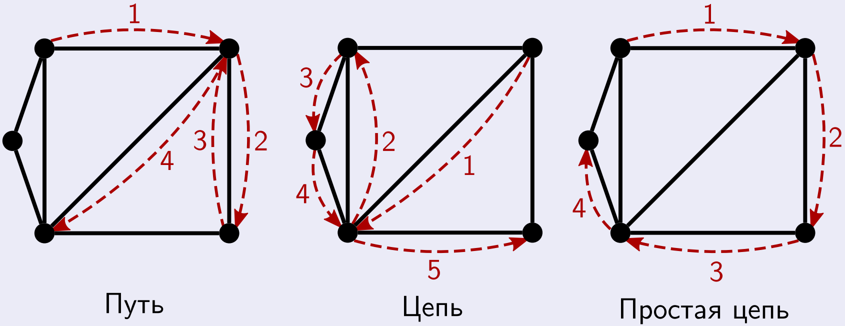
- **Путь** (маршрут) в графе $G = (V, E)$ — это любая последовательность вида

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

  где $v_i \in V,\ i = \overline{0, n}$.
  (это путь из v_0 в v_n; $n \geqslant 0$ — число рёбер в нём — длина пути)
- **Цепь** — это путь, в котором $v_0 \neq v_n$ и все рёбра разные.
- **Простая цепь** — это цепь, в которой все вершины различные.

Пути в графах

Примеры путей в графе



- Подпуть пути в графе G — это подпоследовательность, которая тоже является путём в G.

Пути в графах

Утверждение

Любой путь в графе G из v_0 в v_n, где $v_0 \neq v_n$, содержит подпуть из v_0 в v_n, который является простой цепью.

Доказательство

- Пусть в G задан путь L из v_0 в v_n, и $v_0 \neq v_n$.
- Если вершина v_0 встречается несколько раз, то удалим в L начальную часть до последнего вхождения вершины v_0.
- После этого, если вершина v_n встречается несколько раз, удалим заключительную часть пути после первого вхождения вершины v_n.
- Получим путь L_1 из v_0 в v_n, в котором вершины v_0 и v_n не повторяются.

Пути в графах

Утверждение

Любой путь в графе G из v_0 в v_n, где $v_0 \neq v_n$, содержит подпуть из v_0 в v_n, который является простой цепью.

Доказательство (продолжение)

- Если в L_1 вершины не повторяются, то рёбра тем более не повторяются, и L_1 является простой цепью.
- Иначе L_1 имеет вид $v_0 C_1 v C_2 v C_3 v_n$, где v — вершина графа, а C_1, C_2, C_3 — участки пути.
- Построим путь $L_2 = v_0 C_1 v C_3 v_n$, который является подпутём L_1, в котором вершина v повторяется на 1 раз меньше.
- Повторяем аналогично, исключая все повторяющиеся вершины. В конце концов получим простую цепь L_k.

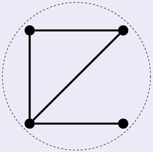
□

Связность графов

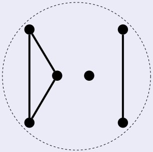
Определение

Граф G называется **связным**, если для любых вершин u, v графа G существует путь из u в v.

Примеры



Связный граф



Несвязный граф

Связность графов

Отношение существования пути

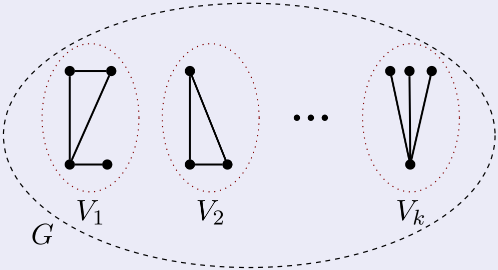
- Пусть $G = (V, E)$. Введём на множестве V отношение

$$u \to v \equiv (\text{в графе } G \text{ существует путь из } u \text{ в } v).$$

- Для этого отношения выполняются свойства:
  1. Рефлексивность: $\forall u \in V$ верно $u \to u$.
  2. Симметричность: $\forall u, v \in V$, если $u \to v$, то $v \to u$.
  3. Транзитивность: $\forall u, v, w \in V$, если $u \to v$ и $v \to w$, то $u \to w$.
- Таким образом, $\to$ — это отношение эквивалентности.
- Тогда V разбивается на непересекающиеся классы эквивалентности $V_1, \ldots, V_k$ так, что любые две вершины из одного класса соединены путём, а вершины из разных классов не соединены путём.

Связность графов

Компоненты связности



- Таким образом, граф G распадается на связные подграфы, которые друг с другом не соединены рёбрами.
- Эти подграфы называются компонентами связности графа G.
- Если граф G связен, то у него одна компонента связности.

Замкнутые пути

Определение

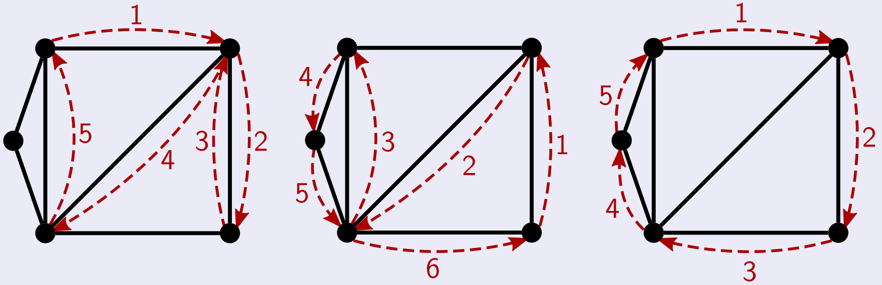
- Путь

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

  в графе G называется замкнутым, если $v_n = v_0$.
- Цикл — это замкнутый путь, в котором все рёбра разные.
- Простой цикл — это цикл, в котором все вершины различные (не считая $v_n = v_0$).

Замкнутые пути

Примеры замкнутых путей в графе



Замкнутый путь   Цикл   Простой цикл

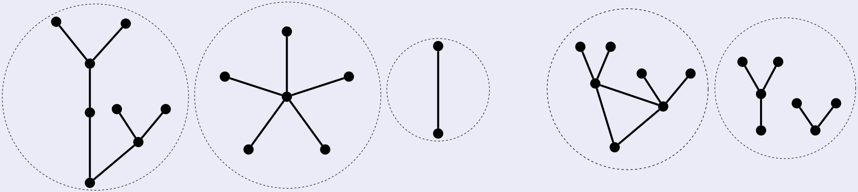
Деревья

Определение

Дерево — это связный граф без циклов.

- В определении дерева подразумеваются циклы, содержащие хотя бы одно ребро.

Примеры



Дерево Дерево Дерево Граф с циклом Лес

Деревья

Определение

Подграф $G_1 = (V_1, E_1)$ графа $G = (V, E)$ называется остовным деревом, если $V_1 = V$ и G_1 — дерево.

Лемма 1

Пусть граф $G = (V, E)$ связен и ребро (u, v) входит хотя бы в один цикл G. Тогда при удалении ребра (u, v) граф останется связным.

Доказательство

- Если ребро (u, v) входит в цикл, то u и v соединены путём, не содержащим ребра (u, v).
- Тогда во всех путях между вершинами G ребро (u, v) можно заменить на путь от u до v.
- При удалении ребра u, v пути между всеми вершинами сохранятся.

$\square$

Деревья

Теорема

В любом связном графе G существует подграф, являющийся остовным деревом.

Доказательство

- Если в графе G нет циклов, то он сам является остовным деревом.
- Иначе в G есть цикл, т.е. существует ребро, входящее в цикл. Удалим это ребро. По лемме 1 граф остаётся связным.
- Если в графе всё ещё есть циклы, продолжаем удаление рёбер аналогично, каждый раз получая связный граф с меньшим числом рёбер.
- Поскольку в графе конечное число рёбер, в конце концов получим связный граф без циклов — остовное дерево G.

$\square$

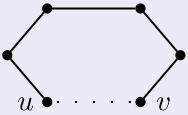
Деревья

Лемма 2

Если в связный граф $G = (V, E)$ добавить новое ребро (u, v), где $u, v \in V$, то в графе G образуется хотя бы один простой цикл.

Доказательство

- Пусть $u, v \in V$, $u \neq v$ и $(u, v) \notin E$. Поскольку граф G связный, в нём существует путь из u в v.
- По доказанному ранее утверждению в этом пути можно выделить подпуть, который является простой цепью из u в v.
- При добавлении в конец этой простой цепи ребра (v, u) получается простой цикл.



□

Деревья

Лемма 3

Если в графе $G = (V, E)$ ровно p вершин и q рёбер, то

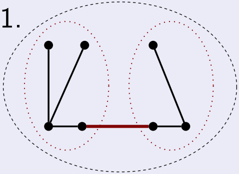
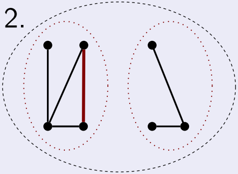
1. *в G не менее $p - q$ компонент связности;*
2. *если в G нет циклов, то в нём ровно $p - q$ компонент связности.*

Доказательство

- Будем строить граф G постепенно, начиная с графа из p изолированных вершин (без рёбер), и добавляя на каждом шаге одно ребро.
- У исходного графа без рёбер p компонент связности.

Деревья

Доказательство (число компонент связности, продолжение)

1. 
2. 

- При добавлении ребра:
  1. Ребро между разными компонентами связности $\Rightarrow$ число компонент уменьшается на 1;
  2. Ребро внутри компоненты связности $\Rightarrow$ число компонент не изменяется и по лемме 2 появляется цикл.
- При добавлении всех q рёбер число компонент уменьшится на q или менее. Получится число, не меньшее $p - q$.
- Если в G нет циклов, то при добавлении ребра возможен только первый случай. Поэтому получится ровно $p - q$ компонент.

$\square$

Лекция 6
Эквивалентные определения деревьев. Корневые деревья. Геометрическая реализация графов. Планарные графы. Граф K_5

Деревья

Теорема

Пусть $G = (V, E)$, $p = |V|$, $q = |E|$.
Следующие утверждения эквиваленты:

1. G — *дерево* (*т.е. связный граф без циклов*);
2. G — *граф без циклов и* $q = p - 1$;
3. G — *связный граф и* $q = p - 1$;
4. G — *связный граф, но при удалении любого ребра становится несвязным*;
5. G — *граф без циклов, но при добавлении любого ребра на тех же вершинах появляется цикл.*

Деревья

Доказательство (разные определения дерева)

- Будем доказывать $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 4 \Rightarrow 5 \Rightarrow 1$.
- $1 \Rightarrow 2$. G — связный граф без циклов.
  - По лемме 3 в графе должно быть ровно $p - q$ компонент связности.
  - Тогда имеем $1 = p - q$, то есть $q = p - 1$.
- $2 \Rightarrow 3$. Имеем граф без циклов и $q = p - 1$.
  - По лемме 3 в графе должно быть $p - q = 1$ компонент связности.
  - То есть граф G связный.
- $3 \Rightarrow 4$. Имеем связный граф и $q = p - 1$.
  - Удалим любое ребро G. Тогда останется $q' = p - 2$ рёбер.
  - По лемме 3 число компонент связности не меньше $p - q' = 2$.
  - Таким образом, при удалении любого ребра получается несвязный граф.

Деревья

Доказательство (разные определения дерева, продолжение)

- $4 \Rightarrow 5$. Имеем связный граф, который при удалении любого ребра становится несвязным.
  - Если бы в G был цикл, то, удаляя ребро из этого цикла, по лемме 1 получили бы связный граф, что невозможно.
  - Поэтому в G нет циклов.
  - По лемме 2 при добавлении в связный граф G ребра появляется цикл.
- $5 \Rightarrow 1$. Имеем граф без циклов, в котором при добавлении любого ребра появляется цикл.
  - Пусть G не связный. Тогда при добавлении ребра между разными компонентами связности цикла не добавится.
  - Это противоречит условию. Значит, G связный.

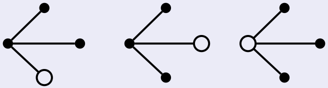
$\square$

Корневые деревья

Определение

- Корневое дерево — это дерево с выделенной вершиной.
- Выделенная вершина называется корнем.

Пример



1 2 3

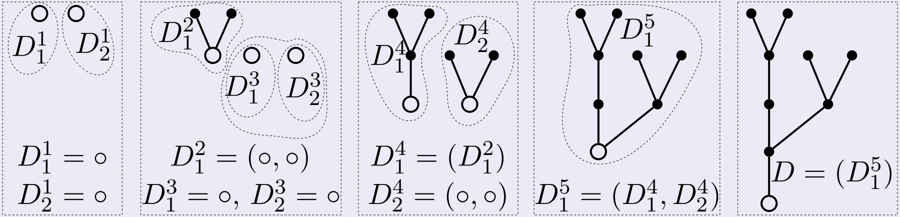
Корневые деревья 1 и 2 изоморфны, корневое дерево 3 неизоморфно им.

Корневые деревья

Определение (индуктивное)

1. Граф из одной вершины (корня), является **корневым деревом**.
2. Пусть $D_1 = (V_1, E_1), \ldots, D_m = (V_m, E_m)$ — корневые деревья с корнями $v_1, \ldots, v_m$ ($V_1, \ldots, V_m$ попарно не пересекаются). Тогда **корневым деревом** является граф $D = (V, E)$, где
   - $V = V_1 \cup \ldots \cup V_m \cup \{v_0\}$, где $v_0 \notin V_1 \cup \ldots \cup V_m$ — новая вершина.
   - $E = E_1 \cup \ldots \cup E_m \cup \{(v_0, v_1), \ \ldots, \ (v_0, v_m)\}$.
   - Корнем D выбирается v_0.

Пример

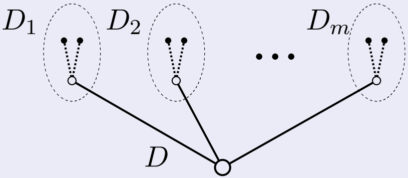


Корневые деревья

Определение

В индуктивном определении корневого дерева деревья $D_1, \ldots D_m$ называются поддеревьями дерева D.

Иллюстрация



Эквивалентность определений

Общее и индуктивное определения корневого дерева эквивалентны.

Корневые деревья

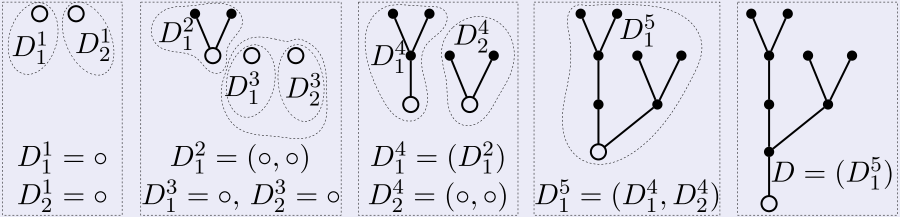
Определение

Упорядоченным корневым деревом называется корневое дерево, в котором

1. Задан порядок поддеревьев;
2. Каждое поддерево является упорядоченным корневым деревом.

Корневое дерево из одной вершины является упорядоченным корневым деревом.

Пример



$D_1^1 = \circ$
$D_2^1 = \circ$

$D_1^2 = (\circ, \circ)$
$D_1^3 = \circ,\ D_2^3 = \circ$

$D_1^4 = (D_1^2)$
$D_2^4 = (\circ, \circ)$

$D_1^5 = (D_1^4, D_2^4)$

$D = (D_1^5)$

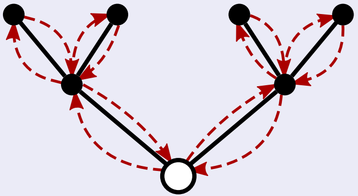
Корневые деревья

Теорема

Число различных упорядоченных корневых деревьев с q рёбрами не превосходит 4^q.

Доказательство

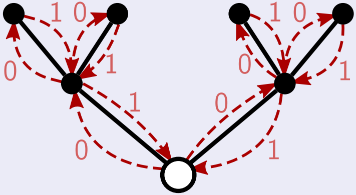
- Рассмотрим алгоритм обхода «в глубину» упорядоченного корневого дерева.
    1. Начать с корня. Пока есть не пройденные поддеревья выполнять:
    2. Перейти в корень очередного поддерева, обойти это поддерево «в глубину»;
    3. Вернуться в корень исходного дерева.



Корневые деревья

Доказательство теоремы о числе деревьев (продолжение)

- Вдоль каждого ребра дерева алгоритм обхода проходит дважды: один раз при заходе в поддерево и один раз при возврате.
- Каждый обход формирует последовательность из 0 и 1:
  1. Движение по ребру от корня добавляет 0;
  2. Движение по ребру к корню добавляет 1.



- Таким образом, каждому дереву сопоставляется последовательность из 0 и 1 длины $2q$ (код дерева).

Корневые деревья

Доказательство теоремы о числе деревьев (продолжение)

- По коду упорядоченное корневое дерево восстанавливается однозначно:
  1. Начинаем с одной вершины — корня.
  2. 0 указывает, что нужно добавить ребро из текущей вершины в новую вершину и перейти в неё («движемся вверх»).
  3. 1 указывает, что нужно вернуться на предыдущую вершину, на один ярус назад («движемся вниз»).
- Таким образом, каждому коду соответствует только одно упорядоченное корневое дерево.
- Поэтому число упорядоченных корневых деревьев с q рёбрами не превосходит числа последовательностей из 0 и 1 длины $2q$.
- Таких последовательностей всего $2^{2q} = 4^q$.

$\square$

Корневые деревья

Замечание

- Корневые деревья получаются из деревьев добавлением пометки «корень» на одну из вершин.
- Поэтому число деревьев с q рёбрами не превосходит числа корневых деревьев с q рёбрами.
- Упорядоченные корневые деревья получаются из корневых деревьев добавлением порядка поддеревьев.
- Поэтому число корневых деревьев с q рёбрами не превосходит числа упорядоченных корневых деревьев с q рёбрами.

Следствие

1. Число неизоморфных корневых деревьев с q рёбрами не превосходит 4^q.
2. Число неизоморфных деревьев с q рёбрами не превосходит 4^q.

Геометрическая реализация графов

Определение

Пусть $G = (V, E)$ — граф и $V = \{v_1, \ldots, v_p\}$, а $E = \{e_1, \ldots, e_q\}$. Будем говорить, что задана **геометрическая реализация графа** G в пространстве M, если

1. Каждой вершине v_i графа G сопоставляется точка a_i в пространстве M (разным вершинам разные точки);
2. Каждому ребру $e_k = (v_i, v_j)$ сопоставлена непрерывная кривая l_k (без самопересечения и самоналегания), соединяющая точки a_i и a_j, причём l_k не проходит через другие точки $a_s \notin \{a_i, a_j\}$.
3. Любые две различные кривые l_k и l_m не имеют общих точек, за исключением концов.

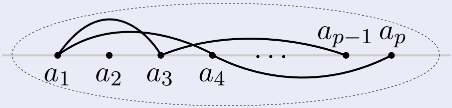
Геометрическая реализация графов

Теорема

Для любого конечного графа $G = (V, E)$ существует геометрическая реализация в трёхмерном евклидовом пространстве.

Доказательство

- Выберем произвольную прямую l в пространстве и разместим на ней точки вершин $a_1, \ldots, a_p$.
- Проведём $q = |E|$ плоскостей через прямую l, и разместим кривую каждого ребра в отдельной плоскости (вне прямой l).
- Проведённые таким образом кривые не могут пересекаться.



$\square$

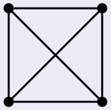
Планарные графы

Определение

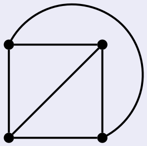
Граф называется планарным, если существует его геометрическая реализация на плоскости.

- Напомним, что у геометрической реализации графа не должно быть пересечений рёбер.

Пример



некорректная геометрическая реализация



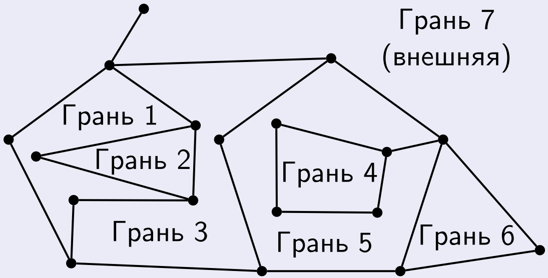
корректная геометрическая реализация

Планарные графы

Определение

- Грань геометрической реализации графа на плоскости — это максимальный по включению связный участок плоскости, не содержащий точек вершин и рёбер этой реализации.
- Ограниченные грани называются внутренними. Неограниченная грань называется внешней.

Пример



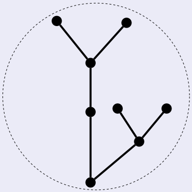
Планарные графы

Теорема (формула Эйлера для планарных графов)

Пусть $G = (V, E)$ *— связный планарный граф,* $p = |V|,\ q = |E|$.
Пусть r *— число граней в некоторой геометрической реализации* G *на плоскости. Тогда* $p - q + r = 2$.

Доказательство

- Индукция по q при фиксированном p.
- База: $q = p - 1$ (т.к. при $q < p - 1$ граф не связен).
- Тогда, в силу связности, G является деревом.

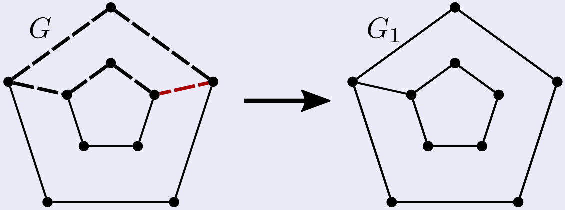


- Поскольку в дереве нет циклов, число граней $r = 1$.
- Тогда при $q = p - 1$ имеем $p - q + r = 2$.

Планарные графы

Доказательство формулы Эйлера (продолжение)

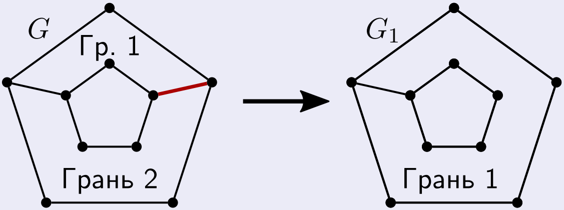
- Пусть равенство верно для некоторого $q = q_0 - 1 \geqslant p - 1$.
- Шаг индукции: докажем, что равенство верно для $q = q_0$.
- Имеем $q_0 \geqslant p$, т.е. G — не дерево. Тогда в G есть цикл.
- Удалим любое ребро e из цикла, а также кривую этого ребра из геометрической реализации G с r гранями.



- Получаем геометрическую реализацию графа G_1.
- Поскольку ребро выброшено из цикла, граф G_1 остался связным.

Планарные графы

Доказательство формулы Эйлера (продолжение)



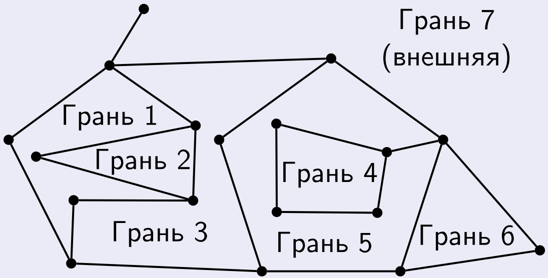
- Поскольку ребро было в цикле, оно разделяло две разные грани. После удаления ребра они соединяются в одну.
- Тогда у G_1 имеется p вершин, $q_0 - 1$ ребро и $r - 1$ граней.
- По индуктивному предположению (для $q_0 - 1$) имеем формулу Эйлера для G_1: $p - (q_0 - 1) + (r - 1) = 2$.
- Тогда $p - q_0 + r = 2$.

□

Планарные графы

- Каждая грань геометрической грани графа на плоскости отделена кривыми рёбер от других граней.
- В простых случаях граница грани представляет собой цикл.
- Для более сложных случаев можно ввести понятие обхода границы грани.

Пример

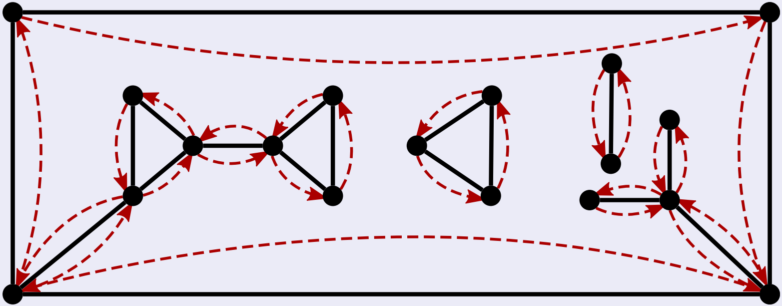


Планарные графы

Обход границы грани

Обход границы грани геометрической реализации графа G на плоскости — это минимальный по суммарной длине набор замкнутых маршрутов, содержащий все вершины и рёбра, точки которых являются граничными точками грани.

Пример



Планарные графы

Утверждение

Пусть $G = (V, E)$ *— планарный граф,* $|E| = q$*, геометрическая реализация* G *на плоскости имеет* r *граней и* $q_1, \ldots, q_r$ *— количества рёбер в обходе границы каждой грани. Тогда*

$$\sum_{i=1}^{r} q_i = 2q.$$

Доказательство

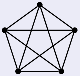
- Каждое ребро графа участвует в обходе границы двух граней графа, либо в обходе границы одной грани дважды.
- Значит, каждое из q рёбер добавляет $+2$ к сумме длин обходов.

□

Граф K_5

Определение

K_5 — это полный граф с 5 вершинами (каждая вершина соединена с каждой).



Теорема

Граф K_5 не планарен.

Доказательство

- Пусть существует геометрическая реализация графа K_5 на плоскости.
- Тогда в ней $p = 5,\ q = 10$. При этом K_5 связен.
- По формуле Эйлера $p - q + r = 2$, где r — число граней.
- Тогда K_5 имеет $r = 2 + q - p = 7$ граней.

Граф K_5

Доказательство (непланарность K_5, продолжение)

- K_5 имеет $q = 10$ рёбер и $r = 7$ граней.
- Пусть q_i — длина обхода границы i-й грани, $i = \overline{1, r}$.
- Каждая грань должна быть отделена от других как минимум одним циклом, а в цикле не может быть менее 3 ребёр.
- Поэтому $q_i \geqslant 3, \; i = \overline{1, r}$.
- Тогда имеем

$$21 = 3r \leqslant \sum_{i=1}^{r} q_i = 2q = 20.$$

- Противоречие означает, что граф K_5 не планарный.

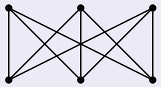
$\square$

Лекция 7
Граф $K_{3,3}$. Теорема Понтрягина-Куратовского. Число вершин и рёбер в планарном графе. Раскраски графов

Граф $K_{3,3}$

Определение

$K_{3,3}$ — это полный двудольный граф, у которого в каждой доле по 3 вершины (все вершины из разных долей соединены, все вершины из одной доли не соединены).



Теорема

Граф $K_{3,3}$ не планарен.

Доказательство

- Пусть существует геометрическая реализация графа $K_{3,3}$ на плоскости.
- Тогда в ней $p = 6, \ q = 9$. При этом $K_{3,3}$ связен.
- По формуле Эйлера $p - q + r = 2$, где r — число граней.
- Тогда $K_{3,3}$ имеет $r = 2 + q - p = 5$ граней.

Граф $K_{3,3}$

Доказательство (непланарность $K_{3,3}$, продолжение)

- $K_{3,3}$ имеет $q = 9$ рёбер и $r = 5$ граней.
- Пусть q_i — длина обхода границы i-й грани, $i = \overline{1, r}$.
- Каждая грань должна быть отделена от других как минимум одним циклом, а в цикле не может быть менее 3 ребёр.
- Но в графе $K_{3,3}$ нет циклов длины 3, т.е. в цикле не менее 4 ребёр.
- Поэтому $q_i \geqslant 4,\ \ i = \overline{1, r}$.
- Тогда имеем

$$20 = 4r \leqslant \sum_{i=1}^{r} q_i = 2q = 18.$$

- Противоречие означает, что граф $K_{3,3}$ не планарный.

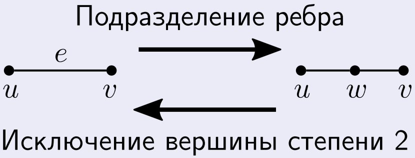
□

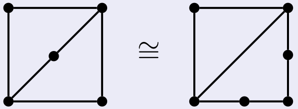
Теорема Понтрягина-Куратовского

Определение

- Подразделение ребра $e = (u, v)$ в графе G — это удаление ребра e и добавление новой вершины w с рёбрами (u, w) и (w, v).
- Граф H называется подразделением графа G, если H можно получить из G путём конечного числа подразделений рёбер.
- Графы G_1 и G_2 называются гомеоморфными, если существуют их подразделения, которые изоморфны между собой.

Примеры

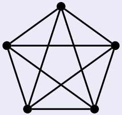




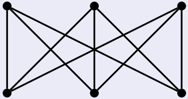
Теорема Понтрягина-Куратовского

Теорема (Понтрягин, Куратовский)

Граф планарен тогда и только тогда, когда он не содержит подграфов, гомеоморфных K_5 *или* $K_{3,3}$.



K_5



$K_{3,3}$

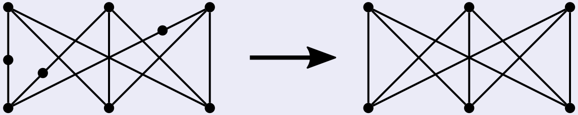
Доказательство

- $\Rightarrow$. Пусть в G есть подграф G_1, гомеоморфный K_5 или $K_{3,3}$.
- Рассмотрим геометрическую реализацию графа G на плоскости.
- Удалим из нее точки и линии, которые соответствуют вершинам и рёбрам, отсутствующим в G_1.

Теорема Понтрягина-Куратовского

Доказательство (продолжение)

- Получаем геометрическую реализацию G_1.
- Если в этой геометрической реализации считать вершины степени 2 частью линий, то получим геометрическую реализацию графа K_5 или $K_{3,3}$ на плоскости.



- Но графы K_5 и $K_{3,3}$ не планарны, поэтому такой геометрической реализации быть не может.
- Противоречие означает, что в G нет подграфа, гомеоморфного K_5 или $K_{3,3}$.
- $\Leftarrow$. Без доказательства.

$\square$

Число вершин и рёбер в планарном графе

Теорема

Пусть G — связный планарный граф, не являющийся деревом, имеющий p вершин и q рёбер и пусть в G нет циклов длины меньше k $(k \geqslant 3)$. Тогда

$$q \leqslant \frac{k}{k-2}(p-2).$$

Доказательство

- Рассмотрим геометрическую реализацию G на плоскости.
- Пусть $q_1, \ldots, q_r$ — это количества рёбер в обходе границы каждой грани этой реализации. Имеем $\sum\limits_{i=1}^{r} q_i = 2q$.
- Каждая грань должна быть отделена от других как минимум одним циклом, поэтому $q_i \geqslant k, \; i = \overline{1, r}$.
- Тогда $2q \geqslant kr$, т.е. $r \leqslant \frac{2q}{k}$.

Число вершин и рёбер в планарном графе

Доказательство теоремы о числе рёбер (продолжение)

- Имеем $r \leqslant \frac{2q}{k}$.
- По формуле Эйлера для связного графа имеем $p - q + r = 2$, т.е. $r = 2 - p + q$.
- Тогда $2 - p + q = r \leqslant \frac{2q}{k}$, т.е. $qk - 2q \leqslant (p-2)k$.
- Наконец, получаем $q \leqslant \frac{k}{k-2}(p-2)$.

$\square$

Число вершин и рёбер в планарном графе

Следствие

В планарном графе с p вершинами и q рёбрами, где $p \geqslant 3$, верно $q \leqslant 3(p-2)$.

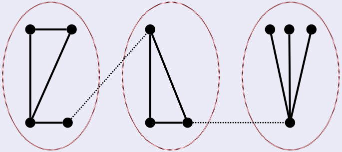
Доказательство

- Пусть G связен. Если G не дерево, то в G, так как в G нет циклов длины меньше 3, по доказанной теореме получаем $q \leqslant 3(p-2)$.
- Если G — дерево, то $q = p - 1$. Нужное неравенство выполняется при $p - 1 \leqslant 3(p-2)$, т.е. $5 \leqslant 2p$. При $p \geqslant 3$ это верно.
- Пусть G — не связный планарный граф и $p \geqslant 3$. Рассмотрим любую его геометрическую реализацию на плоскости.

Число вершин и рёбер в планарном графе

Доказательство следствия (продолжение)

- Добавляя в G рёбра (и соответствующие линии в геометрическую реализацию) можно получить связный граф G' с его геометрической реализацией на плоскости.



- Тогда имеем связный планарный граф G' с $p \geqslant 3$ вершинами и $q' > q$ рёбрами.
- Применяя к G' один из прошлых случаев, имеем $q < q' \leqslant 3(p-2)$.

$\square$

Число вершин и рёбер в планарном графе

Лемма

В любом планарном графе $G = (V, E)$ есть вершина степени 5 или менее.

Доказательство

- Если в G менее 3 вершин, то их степени не превосходят 1, и утверждение леммы очевидно.
- Иначе пусть $q = |E|$, а $V = \{v_1, \ldots, v_p\}$.
- Имеем $\sum\limits_{i=1}^{p} \deg v_i = 2q$ и $q \leqslant 3p - 6$. Тогда $\sum\limits_{i=1}^{p} \deg v_i \leqslant 6p - 12$.
- Пусть d_0 — минимальная степень вершины. Тогда $pd_0 \leqslant \sum\limits_{i=1}^{p} \deg v_i \leqslant 6p - 12$.
- Получаем $d_0 \leqslant 6 - \frac{12}{p}$, то есть, поскольку d_0 целое, $d_0 \leqslant 5$.

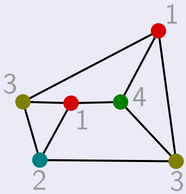
$\square$

Раскраски графов

Определение

- Пусть $G = (V, E)$ — граф. Пусть $C = \{c_1, \dots, c_k\}$ — произвольное множество, элементы которого называются цветами. Раскраска (вершинная) графа G — это отображение $\varphi\colon V \to C$,
- Раскраска называется правильной, если любые две смежные вершины раскрашены в разные цвета, т.е. для любого ребра $(u, v) \in E$ выполнено $\varphi(u) \neq \varphi(v)$.

Пример правильной раскраски



Раскраски графов

Теорема

Вершины любого планарного графа можно правильно раскрасить в 5 или меньшее число цветов.

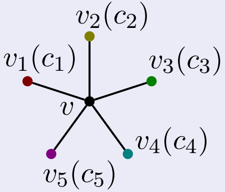
Доказательство

- Индукция по чилу вершин p. Для $p = 1$ утверждение очевидно.
- Предположим, что любой планарный граф с k вершинами правильно раскрашивается в 5 или менее цветов.
- Шаг индукции. Докажем, что это верно для планарного графа G с $k + 1$ вершиной.
- По лемме в G есть вершина v степени не больше 5.
- Рассмотрим геометрическую реализацию G на плоскости.
- Удалим из G вершину v и все рёбра, инцидентные ей. Получим граф G_1 с k вершинами и его геометрическую реализацию.

Раскраски графов

Доказательство теоремы о 5 красках (продолжение)

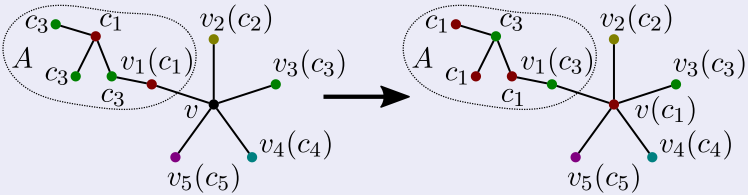
- Граф G_1 тоже планарен. По индуктивному предположению правильно раскрасим G_1 в 5 или менее цветов.
- а) Если у вершин, смежных с v, используется не более 4 цветов, то раскрашиваем v в любой оставшийся цвет. Получим правильную раскраску G в 5 или менее цветов.
- б) Иначе $\deg v = 5$ и вершины, смежные с v, раскрашены во все 5 цветов. Упорядочим эти вершины по часовой стрелке и обозначим $v_1, \ldots, v_5$, а их цвета — $c_1, \ldots, c_5$.



Раскраски графов

Доказательство теоремы о 5 красках (продолжение)

- Пусть A — это множество всех вершин графа G_1, до которых есть путь из v_1 по рёбрам G_1 и вершинам цветов c_1 и c_3.
- б1) Если $v_3 \notin A$, то поменяем в A цвета вершин $c_1 \leftrightarrow c_3$. Раскраска G_1 останется правильной.
- Теперь цвета v_1 и v_3 совпадают и равны c_3. Красим v в цвет c_1, получаем правильную раскраску G в 5 цветов.



Раскраски графов

Доказательство теоремы о 5 красках (продолжение)

- Почему раскраска G_1 останется правильной:
  1. Вершины вне A не затронуты перекраской. Поэтому любое ребро между вершинами не из A соединяет вершины разных цветов.
  2. Любое ребро внутри A соединяет вершины цветов c_1 и c_3. При перекраске цвета поменяются местами, но останутся различными.
  3. У вершин из A не может быть соседних вершин не из A цвета c_1 или c_3. Поэтому любое ребро между вершиной из A и вершиной не из A соединяет вершину цвета c_1/c_3 с вершиной цвета $c_2/c_4/c_5$. При перекраске это не поменяется.

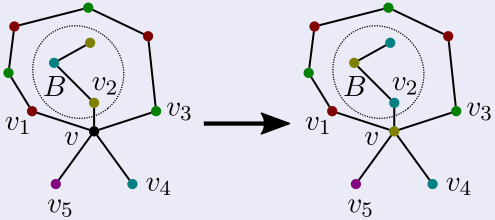
Раскраски графов

Доказательство теоремы о 5 красках (продолжение)

- 62) Пусть $v_3 \in A$. Тогда рассмотрим B — множество всех вершин графа G_1, до которых есть путь из v_2 по рёбрам G_1 и по вершинам цветов c_2 и c_4.
- Поскольку $v_3 \in A$, существует путь из v_1 в v_3 по рёбрам G_1 и вершинам цветов только c_1 и c_3.
- Этот путь вместе с рёбрами (v_3, v) и (v, v_1) образует цикл, причём точки вершин v_2 и v_4 лежат по разные стороны замкнутой линии этого цикла.

Раскраски графов

Доказательство теоремы о 5 красках (продолжение)



- Поэтому любой путь из v_2 в v_4 пересекает этот цикл, а значит, содержит вершину цвета c_1 или c_3.
- Тогда $v_4 \notin B$. Поменяем в B цвета вершин $c_2 \leftrightarrow c_4$. Раскраска G_1 останется правильной.
- Теперь цвета v_2 и v_4 совпадают и равны c_4. Красим v в цвет c_2, получаем правильную раскраску G в 5 цветов.

$\square$

Лекция 8
Алфавитное кодирование. Однозначные коды. Граф кода. Теорема Маркова

Алфавитное кодирование

Определение

- Алфавит (конечный алфавит) A — это непустое конечное множество. Элементы алфавита называются символами.
- Через A^* обозначается множество слов (конечной длины) в алфавите A, включая пустое слово Λ.
- Длина $|w|$ слова $w \in A^*$ — это количество символов в слове w. Длина пустого слова Λ есть нуль.

Определение

- Слово $u \in A^*$ является префиксом слова $w \in A^*$, если существует слово $v \in A^*$ такое, что $w = uv$.
- Слово $u \in A^*$ является суффиксом слова $w \in A^*$, если существует слово $v \in A^*$ такое, что $w = vu$.
- Префикс или суффикс называется собственным, если он не равен Λ и не совпадает со всем словом w.

Алфавитное кодирование

Определение

Кодирование из алфавита A в алфавит B — это произвольное отображение $\varphi\colon A^* \to B^*$.

Определение

- Алфавитное кодирование из A в B задаётся отображением $\varphi\colon A \to B^*$ и условием, что слова из A^* кодируются побуквенно:

$$\varphi(a_{i_1} \dots a_{i_s}) = \varphi(a_{i_1}) \dots \varphi(a_{i_s}), \quad \varphi(\Lambda) = \Lambda.$$

- Пусть $A = \{a_1, \dots, a_r\}$ и $B_i = \varphi(a_i)$, $i = \overline{1, r}$.
  Слова $B_i \in B^*$ называют кодовыми словами.
- Будем считать, что $B_i \neq B_j$ при $i \neq j$ и $i, j = \overline{1, r}$.
- Набор $\{B_1, \dots, B_r\}$ называют кодом.

Алфавитное кодирование

Пример

- Пусть $A = \{a, b, c\}$, $B = \{0, 1\}$ и кодирование φ имеет вид:

$$a \to 0$$
$$b \to 01$$
$$c \to 10$$

- $\varphi(ca) = 100$. $\varphi(ac) = \varphi(ba) = 010$.

Определение

Алфавитное кодирование называется однозначным, если для любых различных слов $u, v \in A^*$ выполнено $\varphi(u) \neq \varphi(v)$.

- Заметим, что однозначность кодирования $\varphi\colon A \to B^*$ не зависит от алфавита A. Она зависит только от кода $C = \{B_1, \ldots, B_r\}$.

Разновидности однозначных кодов

Определение

Код называется равномерным, если все кодовые слова имеют одинаковую длину.

Утверждение

Любой равномерный код является однозначным.

Доказательство

- Пусть кодовые слова имеют длину m, а мы имеем закодированное сообщение w.
- Чтобы декодировать w, нужно разбить его на слова длины m и для каждого из них определить закодированный символ.
- Это можно сделать только одним способом (если слово w действительно кодирует некоторое сообщение).

$\square$

Разновидности однозначных кодов

Определение

Код называется **префиксным**, если никакое кодовое слово не является префиксом другого кодового слова.

Утверждение

Любой префиксный код является однозначным.

Доказательство

- Имеем закодированное сообщение w. Поскольку код префиксный, только одно кодовое слово B_{i_1} является началом $w = B_{i_1} w_2$.
- Теперь только одно кодовое слово B_{i_2} является началом слова $w_2 = B_{i_2} w_3$.
- Продолжая аналогично, можно однозначно декодировать сообщение $w = B_{i_1} w_2 = B_{i_1} B_{i_2} w_3 = \ldots = B_{i_1} \ldots B_{i_k}$.

$\square$

Разновидности однозначных кодов

Определение

Код называется суффиксным, если никакое кодовое слово не является суффиксом другого кодового слова.

Утверждение

Любой суффиксный код является однозначным.

Доказательство

- Имеем закодированное сообщение w. Поскольку код суффиксный, только одно кодовое слово B_{i_1} является концом $w = w_2 B_{i_1}$.
- Теперь только одно кодовое слово B_{i_2} является концом слова $w_2 = w_3 B_{i_2}$.
- Продолжая аналогично, можно однозначно декодировать сообщение $w = w_2 B_{i_1} = w_3 B_{i_2} B_{i_1} = \ldots = B_{i_k} \ldots B_{i_1}$.

□

Граф кода

Вершины графа кода

- Имеем код $C = \{B_1, \ldots, B_r\} \subseteq B^*$.
- Пусть $S_1 = \{\beta_1, \ldots, \beta_k\}$ — множество всех слов $\beta_i \in B^*$ таких, что β_i является одновременно собственным префиксом некоторого слова из C и собственным суффиксом некоторого (возможно, другого) слова из C.
- $S = S_1 \cup \{\beta_0\}$, где $\beta_0 = \Lambda$.
- S — множество вершин графа кода.

Пример

- $C = \{0010,\ 1100,\ 01,\ 001,\ 11\} \subseteq \{0,1\}^*$.
- $(\underbrace{1}100,\ 0\underbrace{1}),\quad (\underbrace{0}1,\ 110\underbrace{0}),\quad (\underbrace{00}1,\ 11\underbrace{00})$.
- $S = \{\Lambda,\ 1,\ 0,\ 00\}$.

Граф кода

- Имеем код $C = \{B_1, \dots, B_r\}$ и построенное по нему множество вершин $S = \{\beta_0, \dots, \beta_k\}$.

Граф кода

$G_C = (S, E)$ — ориентированный псевдограф с пометками на дугах. Пусть $\beta_i, \beta_j \in S$.

1. $\beta_i, \beta_j \neq \Lambda$. Если $B_{i_1}, \dots, B_{i_l} \in C$ $(l \geqslant 0)$ и $\beta_i B_{i_1} \dots B_{i_l} \beta_j \in C$, то в G_C есть дуга (β_i, β_j) с пометкой $B_{i_1} \dots B_{i_l}$ (Λ при $l = 0$).

$$\cdots \to \underset{\beta_i}{\bullet} \xrightarrow{B_{i_1} \dots B_{i_l}} \underset{\beta_j}{\bullet} \to \cdots$$

2. Если $\beta_i = \Lambda$, $\beta_j \neq \Lambda$ или $\beta_i \neq \Lambda$, $\beta_j = \Lambda$, то пункт 1 работает с условием $l \geqslant 1$.
3. Если $\beta_i = \beta_j = \Lambda$, то пункт 1 работает с условием $l \geqslant 2$.

Т.е. среди $\beta_i, B_{i_1}, \dots, B_{i_l}, \beta_j$ должно быть хотя бы два непустых слова.

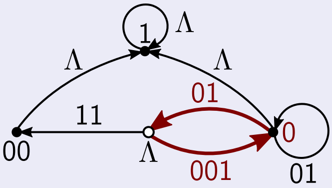
Граф кода

Теорема

Код C является однозначным тогда и только тогда, когда в графе кода G_C нет ориентированных циклов (включая петли), проходящих через вершину Λ.

Пример

- $C = \{0010,\ 1100,\ 01,\ 001,\ 11\} \subseteq \{0,1\}^*$.
- $S = \{\Lambda,\ 1,\ 0,\ 00\}$.



G_S

Код C не однозначный.

Граф кода

Доказательство теоремы о графе кода

- $\Rightarrow$ Пусть в графе G_C есть ориентированный цикл, проходящий через вершину Λ.
- Тогда существует ориентированный цикл, который начинается и оканчивается в вершине $\beta_0 = \Lambda$ и не содержит Λ в промежуточных точках маршрута.
- Пусть этот цикл обходит некоторые вершины $\Lambda, \beta_1, \dots, \beta_m, \Lambda$ в указанном порядке.
- Выпишем все пометки вершин и дуг цикла в порядке его обхода:

$$w = \beta_0 B_{i_1} \dots B_{i_p} \beta_1 B_{j_1} \dots B_{j_q} \beta_2 \dots \beta_m B_{l_1} \dots B_{l_s} \beta_0$$

- По построению графа кода $p, s \geqslant 1$ и имеем $\beta_0 B_{i_1} \dots B_{i_p} \beta_1, \quad \beta_1 B_{j_1} \dots B_{j_q} \beta_2, \quad \dots, \quad \beta_m B_{l_1} \dots B_{l_s} \beta_0 \quad \in \quad C.$

Граф кода

Доказательство теоремы о графе кода (продолжение)

- Тогда (учитывая $\beta_0 = \Lambda$) есть два способа декодировать слово:

$$w = \underbracket{\beta_0 \overbracket{B_{i_1}} \dots \overbracket{B_{i_p}} \beta_1} \overbracket{\underbracket{B_{j_1}} \dots \underbracket{B_{j_q}} \beta_2} \dots \overbracket{\beta_m \underbracket{B_{l_1}} \dots \underbracket{B_{l_s}} \beta_0}.$$

- Указанное выше разделение верно при нечётных m. При чётных m

$$w = \underbracket{\beta_0 \overbracket{B_{i_1}} \dots \overbracket{B_{i_p}} \beta_1} \overbracket{\underbracket{B_{j_1}} \dots \underbracket{B_{j_q}} \beta_2} \dots \underbracket{\beta_m \overbracket{B_{l_1}} \dots \overbracket{B_{l_s}} \beta_0}\,.$$

- Если $m = 0$ (цикл состоит из одной петли), то имеем $p \geqslant 2$ и

$$w = \underbracket{\beta_0 \overbracket{B_{i_1}} \dots \overbracket{B_{i_p}} \beta_0}\,.$$

- Таким образом, в любом из случаев код C не однозначный.

Граф кода

Доказательство теоремы о графе кода (продолжение)

- $\Leftarrow$. Пусть код C не однозначный.
- Тогда существует слово, которое декодируется несколькими способами. Возьмём самое короткое такое слово w.
- Рассмотрим два способа разбить это слово на кодовые слова:

$$w = a_1\, a_2\, a_3\, a_4\, a_5\, a_6\, a_7\, a_8\, a_9\, a_{10}\, a_{11}\, a_{12}\, a_{13}\, a_{14}\, a_{15}\, a_{16}\, a_{17}\,.$$

- Поскольку это самое короткое слово, точки разбиения первым и вторым способом не могут совпадать.
- Обозначим слова между точками разных разбиений $\beta_1, \ldots, \beta_m$.

$$w = a_1\, a_2\, a_3\, \beta_1\, a_7\, a_8\, \beta_2\, \beta_3\, a_{11}\, a_{12}\, a_{13}\, \beta_4\, a_{16}\, a_{17}\,.$$

Граф кода

Доказательство теоремы о графе кода (продолжение)

- Нетрудно видеть, что все слова β_i являются собственными префиксами и собственными суффиксами некоторых кодовых слов, то есть, вершинами G_C.
- Между двумя словами β_i и β_{i+1} может находиться несколько точек одного разбиения:

$$\ldots\ \beta_i\, B_{j_1} \ldots B_{j_q}\, \beta_{i+1}\ \ldots$$

- В таком случае в G_C есть дуга из β_i в β_{i+1} с пометкой, которая является соединением нуля или более кодовых слов.

Граф кода

Доказательство теоремы о графе кода (продолжение)

- Начальный отрезок:

$$\overbrace{\underbrace{B_{j_1}}\ldots\underbrace{B_{j_q}}\,\beta_1}\ldots$$

- В таком случае в G_C есть дуга из Λ в β_1 с пометкой, которая является соединением одного или более кодовых слов.
- Конечный отрезок:

$$\ldots\overbrace{\beta_m\,\underbrace{B_{j_1}}\ldots\underbrace{B_{j_q}}}$$

- В таком случае в G_C есть дуга из β_m в Λ с пометкой, которая является соединением одного или более кодовых слов.

Граф кода

Доказательство теоремы о графе кода (продолжение)

- Если все точки принадлежат одному разбиению, то имеем

$$\overline{\underline{B_{j_1}}\dots\underline{B_{j_q}}}$$

- В таком случае в G_C есть дуга из Λ в Λ с пометкой, которая является соединением двух или более кодовых слов.
- В любом из случаев получаем, что в G_C существует ориентированный путь из Λ в Λ.
- Выбрасывая участки пути между повторяющимися вершинами, получим ориентированный путь из Λ в Λ без повторений вершин. Такой путь является циклом.

$\square$

Теорема Маркова

Теорема (Марков)

Пусть $\varphi\colon a_i \to B_i$, $i = \overline{1, r}$ *— алфавитное кодирование из* A *в* B.

Пусть $l_i = |B_i|$, $i = \overline{1, r}$ *и* $L = \sum\limits_{i=1}^{r} l_i$.

Пусть W *— максимальное количество кодовых слов, которые можно поместить подряд в каком-либо кодовом слове:* $B_j = C' B_{i_1} \ldots B_{i_W} C''$.

Тогда, если φ *не однозначно, то существуют два разных слова* $u', u'' \in A^*$ *такие, что* $\varphi(u') = \varphi(u'')$ *и*

$$|u|,\ |u''| \leqslant \left\lfloor \frac{(W+1)(L-r+2)}{2} \right\rfloor.$$

Теорема Маркова

Доказательство

- Рассмотрим код $C = \{B_1, \ldots, B_r\}$ и граф кода G_C.
- Поскольку код не однозначный, существует ориентированный цикл в G_C, проходящий через вершину $\beta_0 = \Lambda$.
- Выбрасывая участки цикла между одинаковыми вершинами, получим ориентированный путь из Λ в Λ без повторений вершин

$$\beta_0\beta_1 \ldots \beta_m\beta_0.$$

- Выпишем пометки вершин и дуг пути в порядке его обхода:

$$w = \beta_0 B_{i_1} \ldots B_{i_p} \beta_1 B_{j_1} \ldots B_{j_q} \beta_2 \ldots \beta_m B_{l_1} \ldots B_{l_s} \beta_0$$

- По построению G_C слово w неоднозначно расшифровывается:

$$w = \beta_0 B_{i_1} \ldots B_{i_p} \beta_1 B_{j_1} \ldots B_{j_q} \beta_2 \ldots \beta_m B_{l_1} \ldots B_{l_s} \beta_0.$$

Теорема Маркова

Доказательство

- Выберем $u', u'' \in A^*$ — две расшифровки w. Тогда $\varphi(u') = \varphi(u'')$.
- $|u'|$ — это число кодовых слов в первом разбиении w,
  $|u''|$ — во втором.
- Слова $\beta_1, \ldots, \beta_m$ различны и непусты и являются собственными префиксами каких-то кодовых слов B_j.
- У каждого слова B_i всего $l_i - 1$ собственных префиксов. Всего различных собственных префиксов не более $\sum\limits_{i=1}^{r} (l_i - 1) = L - r$.
- Таким образом, $m \leqslant L - r$.

Теорема Маркова

Доказательство

- Рассмотрим участок w между тремя подряд идущими словами β_j:

$$\beta_i B_{i_1} \dots B_{i_p} \beta_{i+1} B_{j_1} \dots B_{j_q} \beta_{i+2}.$$

- Этот участок в любом разбиении содержит не более $W+1$ кодовых слов.
- Подсчитаем, на сколько участков указанного вида можно разбить

$$w = \beta_0 D_0 \beta_1 D_1 \beta_2 \dots \beta_m D_m \beta_0.$$

- Участки будут содержать пары (D_0, D_1), (D_2, D_3) и т.д. Если m чётно, то последний участок берём неполным: $\beta_m D_m \beta_0$ (в нём тоже не более $W+1$ кодовых слов).

Теорема Маркова

Доказательство

- Имеем пар: $\lceil (m+1)/2 \rceil \leqslant \lceil (L-r+1)/2 \rceil \leqslant (L-r+2)/2$.
- Тогда всего кодовых слов в любом разбиении не больше

$$\frac{(W+1)(L-r+2)}{2}$$

- Поскольку число кодовых слов целое, оно не превосходит

$$\left\lfloor \frac{(W+1)(L-r+2)}{2} \right\rfloor .$$

- Итак,

$$|u'|, |u''| \leqslant \left\lfloor \frac{(W+1)(L-r+2)}{2} \right\rfloor .$$

$\square$

Лекция 9
Неравенство Макмиллана. Существование префиксного кода с заданными длинами кодовых слов. Оптимальные коды

Неравенство Макмиллана

Теорема (неравенство Макмиллана)

Пусть $\varphi\colon a_i \to B_i \in B^*,\ i = \overline{1,r}$ *— кодирование из алфавита* $A = \{a_1, \ldots, a_r\}$ *в алфавит* B*,* $q = |B|$*, а* $l_i = |B_i|,\ i = \overline{1,r}$.

Если φ *однозначно, то* $\sum\limits_{i=1}^{r} \frac{1}{q^{l_i}} \leqslant 1$.

Доказательство

- Пусть $x = \sum\limits_{i=1}^{r} \frac{1}{q^{l_i}}$, а $n \in \mathbb{N}$. Нужно доказать, что $x \leqslant 1$.
- Рассмотрим $x^n = \left(\sum\limits_{i_1=1}^{r} \frac{1}{q^{l_{i_1}}}\right) \cdot \left(\sum\limits_{i_2=1}^{r} \frac{1}{q^{l_{i_2}}}\right) \cdot \ldots \cdot \left(\sum\limits_{i_n=1}^{r} \frac{1}{q^{l_{i_n}}}\right)$.
- Для раскрытия скобок мы выбираем по одному слагаемому из каждой скобки и перемножаем. Суммируем по всем способам выбрать так слагаемые.

Неравенство Макмиллана

Доказательство неравенства Макмиллана (продолжение)

$$x^n = \left(\sum_{i_1=1}^{r} \frac{1}{q^{l_{i_1}}}\right) \cdot \left(\sum_{i_2=1}^{r} \frac{1}{q^{l_{i_2}}}\right) \cdot \ldots \cdot \left(\sum_{i_n=1}^{r} \frac{1}{q^{l_{i_n}}}\right) =$$

$$= \sum_{i_1=1}^{r} \sum_{i_2=1}^{r} \cdots \sum_{i_n=1}^{r} \frac{1}{q^{l_{i_1}}} \cdot \frac{1}{q^{l_{i_2}}} \cdot \ldots \cdot \frac{1}{q^{l_{i_n}}} = \sum_{i_1=1}^{r} \cdots \sum_{i_n=1}^{r} \frac{1}{q^{l_{i_1}+\ldots+l_{i_n}}}$$

- Обозначим $l_{\max} = \max(l_1, \ldots, l_r)$.
- В полученной сумме все слагаемые имеют вид $\frac{1}{q^k}$, где $k \in \mathbb{N}$, причём $k \leqslant n \cdot l_{\max}$.
- Обозначим через c_k число слагаемых вида $\frac{1}{q^k}$ в этой сумме.

Неравенство Макмиллана

Лемма

В условиях теоремы о неравенстве Макмиллана $c_k \leqslant q^k$.

Доказательство леммы

- $F_k = \{(i_1, \ldots, i_n) \mid i_1, \ldots, i_n \in \{1, \ldots, r\},\ l_{i_1} + \ldots + l_{i_n} = k\}$.
- Тогда $c_k = |F_k|$. Каждому набору $(i_1, \ldots, i_n) \in F_k$ сопоставим слово $\psi(i_1, \ldots, i_n) = B_{i_1} \ldots B_{i_n} \in B^*$ длины $l_{i_1} + \ldots + l_{i_n} = k$.
- Поскольку кодирование φ однозначно, все слова $\psi(i_1, \ldots, i_n)$ различны.
- Таким образом, каждому набору из F_k соответствует слово в алфавите B длины k, причём разным наборам разные слова.
- Тогда $c_k = |F_k| \leqslant |B^k| = q^k$.

$\square$

Неравенство Макмиллана

Доказательство неравенства Макмиллана (продолжение)

- В сумме сгруппируем все слагаемые вида $\frac{1}{q^k}$ при каждом k.

$$x^n = \left(\sum_{i_1=1}^{r} \frac{1}{q^{l_{i_1}}}\right) \cdot \left(\sum_{i_2=1}^{r} \frac{1}{q^{l_{i_2}}}\right) \cdot \ldots \cdot \left(\sum_{i_n=1}^{r} \frac{1}{q^{l_{i_n}}}\right) =$$

$$= \sum_{i_1=1}^{r} \cdots \sum_{i_n=1}^{r} \frac{1}{q^{l_{i_1}+\ldots+l_{i_n}}} = \sum_{k=1}^{n \cdot l_{\max}} \frac{c_k}{q^k} \leqslant \sum_{k=1}^{n \cdot l_{\max}} \frac{q^k}{q^k} = n \cdot l_{\max}.$$

- Итак, мы получили $x^n \leqslant n \cdot l_{\max}$, т.е. $x \leqslant \sqrt[n]{n \cdot l_{\max}}$ при $n \in \mathbb{N}$.
- Тогда $\lim\limits_{n\to\infty} x \leqslant \lim\limits_{n\to\infty} \sqrt[n]{n \cdot l_{\max}}$.
- Имеем $\lim\limits_{n\to\infty} x = x$ и $\lim\limits_{n\to\infty} \sqrt[n]{n \cdot l_{\max}} = 1$, т.е. $x \leqslant 1$.

$\square$

Неравенство Макмиллана

Комментарий к доказательству неравенства Макмиллана

- Значение $x = \sum\limits_{i=1}^{r} \frac{1}{q^{l_i}}$ описывает набор кодовых слов: слагаемое $\frac{1}{q^{l_i}}$ соответствует кодовому слову B_i.
- Значение

$$x^n = \left(\sum_{i_1=1}^{r} \frac{1}{q^{l_{i_1}}}\right) \cdot \ldots \cdot \left(\sum_{i_n=1}^{r} \frac{1}{q^{l_{i_n}}}\right) = \sum_{i_1=1}^{r} \cdots \sum_{i_n=1}^{r} \frac{1}{q^{l_{i_1}+\ldots+l_{i_n}}} = \sum_{k=1}^{n \cdot l_{\max}} \frac{c_k}{q^k}$$

  описывает набор всех слов, которые можно составить из n кодовых слов: слагаемое $\frac{1}{q^{l_{i_1}+\ldots+l_{i_n}}}$ соответствует слову $B_{i_1} \ldots B_{i_n}$.
- В силу однозначности кода все указанные слова различны, а значит их количество ограничено: c_k не больше, чем q^k.
- Поэтому можно оценить $x^n \leqslant n \cdot l_{\max}$, откуда следует и $x \leqslant 1$.

Существование префиксного кода с заданными длинами

Теорема

Пусть даны натуральные числа $l_1, \ldots, l_r$, q *и алфавиты* $A = \{a_1, \ldots, a_r\}$ *и* $B = \{b_1, \ldots, b_q\}$. *Пусть* $\sum\limits_{i=1}^{r} \frac{1}{q^{l_i}} \leqslant 1$.
Тогда существует префиксное кодирование $\varphi \colon a_i \to B_i \in B^*$, $i = \overline{1, r}$ *такое, что* $|B_i| = l_i$, $i = \overline{1, r}$.

Доказательство

- Пусть $l_{\max} = \max(l_1, \ldots, l_r)$, а при $k = \overline{1, l_{\max}}$ пусть d_k — число таких $i \in \{1, \ldots, r\}$, что $l_i = k$ (т.е. число кодовых слов, которые должны иметь длину k).
- Тогда имеем $\sum\limits_{i=1}^{r} \frac{1}{q^{l_i}} = \sum\limits_{k=1}^{l_{\max}} \frac{d_k}{q^k} \leqslant 1$.
- Мы хотим построить префиксный код в алфавите B такой, что в нём d_1 слов длины $1, \ldots, d_{l_{\max}}$ слов длины $l_{\max}$.

Существование префиксного кода с заданными длинами

Доказательство (продолжение)

- При каждом $m = \overline{1, l_{\max}}$ имеем $\sum\limits_{k=1}^{m} \frac{d_k}{q^k} \leqslant \sum\limits_{k=1}^{l_{\max}} \frac{d_k}{q^k} \leqslant 1$.
- Домножая на q^m и перенося часть слагаемых в правую часть, получим $d_m \leqslant q^m - (d_1 q^{m-1} + \ldots + d_{m-1} q)$.
- Будем строить префиксный код в порядке возрастания длин слов.
- База: $m = 1$. $d_1 \leqslant q = |B|$. Поэтому можно выбрать d_1 разных слов длины 1. Выбираем их произвольно. Они не являются префиксами друг друга.
- Пусть выбрали d_1 слов длины 1, $\ldots$, d_{m-1} слов длины $m-1$, и все выбранные слова не являются префиксами друг друга.
- Шаг для m. Нужно добавить в код d_m различных слов длины m так, чтобы добавленные ранее слова не являлись префиксами новых слов.

Существование префиксного кода с заданными длинами

Доказательство (продолжение)

- Всего существует q^m слов длины m. Уже выбранные более короткие слова запрещают выбирать часть из этих q^m слов:
  - Каждое слово P длины l, включённое в код, запрещает добавлять слова вида $w = Pv$, где $|w| = m,\ |v| = m - l$.
  - Таким образом, слово длины l запрещает добавлять q^{m-l} слов.
  - Все d_l слов длины l запрещают добавлять суммарно $d_l q^{m-l}$ слов.
  - Все уже выбранные слова запрещают не более $d_1 q^{m-1} + d_2 q^{m-2} + \ldots + d_{m-1} q$ слов.
- Тогда не запрещённых слов остаётся не менее $q^m - (d_1 q^{m-1} + d_2 q^{m-2} + \ldots + d_{m-1} q) \geqslant d_m$.
- Тогда среди них можно выбрать d_m слов длины m и включить в код так, чтобы никакие слова не являлись префиксами друг друга.
- Повторяя для $m = 1, \ldots, l_{\max}$, получим искомый префиксный код.

□

Существование префиксного кода с заданными длинами

Следствие

Пусть в алфавите B существует однозначный код с длинами кодовых слов $l_1, \ldots, l_r$. Тогда в алфавите B существует префиксный код с теми же длинами кодовых слов.

Доказательство

- Пусть $q = |B|$. Тогда в силу существования однозначного кода выполняется неравенство Макмиллана:

$$\sum_{i=1}^{r} \frac{1}{q^{l_i}} \leqslant 1.$$

- Тогда по доказанной теореме существует префиксный код с длинами кодовых слов $l_1, \ldots, l_r$.

□

Оптимальные коды

Содержательные соображения

- Пусть имеется алфавит $A = \{a_1, \ldots, a_r\}$, набор вероятностей $P = (p_1, \ldots, p_r)$ и кодирование $\varphi\colon a_i \to B_i \in \{0,1\}^*$, кодирующее тексты в алфавите A словами из нулей и единиц.
- Считаем, что символ a_i встречается в текстах с вероятностью p_i, т.е. в тексте длины N символ a_i встречается $\approx p_i N$ раз.
- Такое условие выполняется, например, для длинных текстов на естественных языках: каждый символ a_i во всех достаточно длинных текстах встречается с примерно одинаковой частотой p_i.
- Пусть исходный текст имеет длину N. После кодирования каждый символ a_i заменится словом длины $l_i = |B_i|$.
- Поскольку каждый символ a_i встречается $\approx p_i N$ раз, общая длина текста после кодирования будет $\approx \sum\limits_{i=1}^{r} l_i p_i N = N \cdot \sum\limits_{i=1}^{r} p_i l_i$.

Оптимальные коды

Определение

Пусть $A = \{a_1, \ldots, a_r\}$, $B = \{0, 1\}$, $P = (p_1, \ldots, p_r)$ — набор вероятностей:

$$p_i > 0,\ i = \overline{1, r}, \quad \sum_{i=1}^{r} p_i = 1.$$

Пусть $\varphi\colon a_i \to B_i \in B^*$, $i = \overline{1, r}$ — префиксное кодирование, $l_i = |B_i|$.

- Цена кодирования φ относительно набора вероятностей P есть

$$c_P(\varphi) = \sum_{i=1}^{r} p_i l_i.$$

- При заданных A, B, P кодирование φ называется оптимальным, если $c_P(\varphi) \leqslant c_P(\psi)$
  для любого префиксного кодирования ψ из алфавита A в B.

Оптимальные коды

Утверждение

Для любого набора вероятностей $P = (p_1, \ldots, p_r)$ существует оптимальное кодирование.

Доказательство

- Рассмотрим произвольное равномерное кодирование φ с r словами длины r. Тогда $c_P(\varphi) = \sum\limits_{i=1}^{r} p_i r = r$.
- Рассмотрим все кодирования ψ такие, что $c_P(\psi) = \sum\limits_{i=1}^{r} p_i l_i \leqslant r$.
- Тогда $l_i \leqslant \frac{r}{p_i}$, $i = \overline{1, r}$ и таких кодирование конечное число.
- Значит, среди них существует кодирование наименьшей цены. Оно и будет оптимальным.

$\square$

Оптимальные коды

- В условиях следующих лемм используем обозначения из определения оптимального кодирования.

Лемма 1

Пусть φ — оптимальное префиксное кодирование и $p_i > p_j$.
Тогда $l_i \leqslant l_j$.

Доказательство

- Пусть $l_i > l_j$. Поменяем местами слова B_i и B_j и получим префиксное кодирование ψ.
- $c_P(\varphi) - c_P(\psi) = (p_i l_i + p_j l_j) - (p_i l_j + p_j l_i) = (p_i - p_j)(l_i - l_j) > 0.$
- То есть $c_P(\psi) < c_P(\varphi)$ и φ не оптимально. Противоречие.

$\square$

Оптимальные коды

Лемма 2

Пусть φ — оптимальное префиксное кодирование, $r \geqslant 2$ и $l_{\max} = \max(l_1, \ldots, l_r)$.
Пусть слово $B'a$, где $a \in \{0, 1\}$, является кодовым у φ и $|B'a| = l_{\max}$.
Тогда в коде присутствует и кодовое слово $B'\overline{a}$.

Доказательство

- Пусть у φ есть кодовое слово $B'a$ (с вероятностью $p_s > 0$) максимальной длины и нет кодового слова $B'\overline{a}$.
- Построим новое кодирование ψ, заменив слово $B'a$ на B'.
- Покажем, что ψ — это префиксное кодирование.
- Поскольку $B'a$ — кодовое слово φ, и φ — префиксное кодирование, у φ нет кодового слова B'. Значит, и у ψ нет кодового слова B', а значит все его кодовые слова разные.

Оптимальные коды

Доказательство леммы 2 (продолжение)

- В φ никакие кодовые слова не являлись префиксами друг друга.
- При переходе от кодирования φ к кодированию ψ слово $B'a$ было заменено на более короткое слово B'. Поэтому только слово B' может оказаться префиксом какого-то другого кодового слова.
- Поскольку $|B'| = l_{\max} - 1$ и $B = \{0, 1\}$, слово B' может быть префиксом только слов $B'a$ и $B'\overline{a}$.
- Но по предположению кодового слова $B'\overline{a}$ нет у φ (значит и у ψ), а кодовое слово $B'a$ было удалено при переходе к ψ.
- Таким образом, код ψ является префиксным.
- $c_P(\varphi) - c_P(\psi) = p_s l_{\max} - p_s(l_{\max} - 1) = p_s > 0$.
- То есть $c_P(\psi) < c_P(\varphi)$ и φ не оптимально. Противоречие.

$\square$

Оптимальные коды

Лемма 3

Пусть φ *— оптимальное префиксное кодирование и* $p_1 \geqslant p_2 \geqslant \ldots \geqslant p_r$. *Тогда можно переставить кодовые слова* φ *так, что получится оптимальное префиксное кодирование, в котором частотам* p_{r-1} *и* p_r *соответствуют кодовые слова, отличающиеся только в последнем символе.*

Доказательство

- По лемме 2 в коде φ есть два слова $B_i = B'a$ и $B_j = B'\overline{a}$ максимальной длины, где $i < j$.
- Поменяем эти слова местами со словами B_{r-1} и B_r. Получим префиксное кодирование ψ.
- Если $i = r - 1$ или $j = r$, то приведённая ниже оценка всё равно останется верной.

Оптимальные коды

Доказательство леммы 3 (продолжение)

- Учитывая $l_i = l_j \geqslant l_{r-1}, l_r$, а также $p_i \geqslant p_{r-1}$ и $p_j \geqslant p_r$, имеем

$$
\begin{aligned}
&c_P(\varphi) - c_P(\psi) = \\
&= (p_i l_i + p_{r-1} l_{r-1}) + (p_j l_j + p_r l_r) - (p_i l_{r-1} + p_{r-1} l_i) - (p_j l_r + p_r l_j) = \\
&\qquad = (p_i - p_{r-1})(l_i - l_{r-1}) + (p_j - p_r)(l_j - l_r) \geqslant 0
\end{aligned}
$$

- То есть $c_P(\psi) \leqslant c_P(\varphi)$. Поскольку φ оптимально, то и ψ оптимально.

$\square$

Лекция 10
Теорема редукции. Коды, исправляющие ошибки. Оценка функции $M_r(n)$

Оптимальные коды

Лемма 4

Пусть есть два набора вероятностей и два кодирования $(p' + p'' = p_k)$:

$P:$	p_1	p_2	$\dots$	p_{k-1}	p_k
$\varphi:$	B_1	B_2	$\dots$	B_{k-1}	B_k

$P':$	p_1	p_2	$\dots$	p_{k-1}	p'	p''
$\varphi':$	B_1	B_2	$\dots$	B_{k-1}	B_k0	B_k1

1. *Если одно из кодирований* φ, φ' *является префиксным, то и второе тоже является префиксным.*
2. $c_{P'}(\varphi') = c_P(\varphi) + p_k$.

Доказательство

- 1.1) Пусть φ — префиксное кодирование. Тогда ни одно из слов $B_1, \dots, B_{k-1}, B_k$ не является префиксом другого.
- Какое-то из слов $B_1, \dots, B_{k-1}$ может быть началом B_k0 или B_k1 только если совпадает с ним. Но тогда B_k было бы его началом.

Оптимальные коды

Доказательство леммы 4 (продолжение)

- Поскольку само слово B_k не является префиксом $B_1, \ldots, B_{k-1}$, то B_k0 или B_k1 тем более не являются их префиксами.
- B_k0 и B_k1 также не могут быть префиксами друг друга. Таким образом, кодирование φ' префиксное.
- 1.2) Пусть теперь φ' префиксное кодирование. Тогда ни одно из слов $B_1, \ldots, B_{k-1}$ не является префиксом другого.
- Никакое из слов $B_1, \ldots, B_{k-1}$ не может быть началом B_k, так как не является началом B_k0.
- Если B_k является началом какого-либо из слова B_i, то оно либо должно совпадать с ним (тогда B_i является началом B_k0), либо одно из слов B_k0, B_k1 является началом B_i.
- Последнее тоже невозможно, поэтому кодирование φ префиксное.

Оптимальные коды

Доказательство леммы 4 (продолжение)

- 2) Подсчитаем искомое соотношение:

$$c_{P'}(\varphi') - c_P(\varphi) = (p'|B_k0| + p''|B_k1|) - p_k|B_k| =$$
$$= (p' + p'')(|B_k| + 1) - p_k|B_k| = p_k(|B_k| + 1) - p_k|B_k| = p_k.$$

$\square$

Теорема редукции

Теорема (редукции)

Пусть есть два набора вероятностей и два кодирования $(p' + p'' = p_k)$:

$P:$	p_1	p_2	$\dots$	p_{k-1}	p_k
$\varphi:$	B_1	B_2	$\dots$	B_{k-1}	B_k

$P':$	p_1	p_2	$\dots$	p_{k-1}	p'	p''
$\varphi':$	B_1	B_2	$\dots$	B_{k-1}	B_k0	B_k1

1. *Если* φ' *— оптимальное префиксное кодирование для* P', *то* φ *— оптимальное префиксное кодирование для* P.
2. *Если* φ *— оптимальное префиксное кодирование для* P *и* $p_1 \geqslant \ldots \geqslant p_{k-1} \geqslant p' \geqslant p''$, *то* φ' *— оптимальное префиксное кодирование для* P'.

Теорема редукции

Доказательство

- 1) φ' — оптимальное префиксное кодирование. По лемме 4 кодирование φ тоже префиксное.
- Предположим, что кодирование φ не оптимально. Тогда существует префиксное кодирование ψ:

$$\begin{array}{|c|cccc|c|}\hline P: & p_1 & p_2 & \ldots & p_{k-1} & p_k \\ \psi: & D_1 & D_2 & \ldots & D_{k-1} & D_k \\ \hline\end{array}$$

  такое, что $c_P(\psi) < c_P(\varphi)$.
- Построим кодирование ψ':

$$\begin{array}{|c|cccc|cc|}\hline P': & p_1 & p_2 & \ldots & p_{k-1} & p' & p'' \\ \psi': & D_1 & D_2 & \ldots & D_{k-1} & D_k0 & D_k1 \\ \hline\end{array}$$

- По лемме 4 ψ' — тоже префиксный код.

Теорема редукции

Доказательство (продолжение)

- По лемме 4 имеем $c_{P'}(\varphi') = c_P(\varphi) + p_k$ и $c_{P'}(\psi') = c_P(\psi) + p_k$.
- Тогда $c_{P'}(\psi') = c_P(\psi) + p_k < c_P(\varphi) + p_k = c_{P'}(\varphi')$.
  Противоречие тому, что код φ' оптимальный.
- Тогда φ должен быть оптимальным префиксным кодом.
- 2) φ — оптимальное префиксное кодирование
  и $p_1 \geqslant \ldots \geqslant p_{k-1} \geqslant p' \geqslant p''$.
- По лемме 4 кодирование φ' тоже префиксное.
- Предположим, что кодирование φ' не оптимально.
  Тогда выберем оптимальное префиксное кодирование ψ' для P'.
  Для него будет верно $c_{P'}(\psi') < c_{P'}(\varphi')$.
- По лемме 3, с учётом $p_1 \geqslant \ldots \geqslant p_{k-1} \geqslant p' \geqslant p''$, переставим в ψ' слова так, чтобы частотам p', p'' соответствовали кодовые слова, отличающиеся только в последнем символе.

Теорема редукции

Доказательство (продолжение)

- Получим оптимальное префиксное кодирование ψ':

P':	p_1	p_2	$\dots$	p_{k-1}	p'	p''
ψ':	D_1	D_2	$\dots$	D_{k-1}	D_k0	D_k1

- Построим кодирование ψ (тоже префиксное по лемме 4):

P:	p_1	p_2	$\dots$	p_{k-1}	p_k
ψ:	D_1	D_2	$\dots$	D_{k-1}	D_k

- По лемме 4 имеем $c_{P'}(\varphi') = c_P(\varphi) + p_k$ и $c_{P'}(\psi') = c_P(\psi) + p_k$.
- Тогда $c_P(\psi) = c_{P'}(\psi') - p_k < c_{P'}(\varphi') - p_k = c_P(\varphi)$. Противоречие тому, что код φ оптимальный.
- Тогда φ' должен быть оптимальным префиксным кодом.

$\square$

Теорема редукции

Построение оптимального префиксного кода

- Строим оптимальный код рекурсивным алгоритмом.
- Имеем набор вероятностей $P_k = (p_1, \ldots, p_k)$.
- Если $k = 2$, то оптимальное кодирование имеет слова 0 и 1.
- Иначе переставим вероятности и получим набор $P'_k = (p'_1, \ldots, p'_k)$ такой, что $p'_1 \geqslant \ldots \geqslant p'_k$.
- Далее (рекурсивно) ищем оптимальное кодирование для вероятностей $P_{k-1} = (p'_1,\ \ldots,\ p'_{k-2},\ p)$, где $p = p'_{k-1} + p'_k$.
- Если $\{B_1, \ldots, B_{k-1}\}$ — оптимальный код для P_{k-1}, то по теореме редукции $\{B_1,\ \ldots,\ B_{k-2},\ \ B_{k-1}0,\ B_{k-1}1\}$ — оптимальный код для P'_k.
- Переставляя слова в этом коде, получим оптимальный код для P_k.

Коды, исправляющие ошибки

Содержательные соображения

- При передаче цифровой информации часто возникают искажения.
- Чтобы бороться с этим, на практике используют не оптимальные коды, а коды с избыточной информацией.
- Эта дополнительная информация позволяет восстановить исходное сообщение, даже если при передаче появились ошибки.
- Рассматриваем равномерные двоичные коды со словами длины n.
- Считаем, что допускаются ошибки типа «замещение»: замены символов $0 \leftrightarrow 1$ в пересылаемом сообщении.
- В этом случае сообщение с ошибками можно разделить на части длины n (при отсутствии ошибок — на кодовые слова), и исправлять ошибки в каждом кодовом слове отдельно.
- Считаем, что ошибки возникают редко; тогда можно рассчитывать, что в каждом кодовом слове их будет немного.

Коды, исправляющие ошибки

Определение

Пусть дан равномерный двоичный код $C = \{B_1, \ldots, B_k\}$, в котором все слова составлены из нулей и единиц и имеют длину n.

- **Ошибка** в кодовом слове — это замена в нём 0 на 1 или 1 на 0.
- Код C **исправляет** r **ошибок**, если по любому слову B', которое получается из некоторого слова B_i добавлением не более r ошибок, можно однозначно восстановить исходное кодовое слово.
  - Т.е. добавлением r ошибок два разных кодовых слова B_i и B_j не могут быть преобразованы одно и то же слово B'.
- Код C **обнаруживает** r **ошибок**, если по слову B', которое получается из некоторого слова B_i добавлением не более r ошибок, можно однозначно определить, есть ли в B' ошибки.
  - Т.е. добавление r ошибок в кодовое слово B_i не может привести к получению другого кодового слова B_j.

Коды, исправляющие ошибки

Определение

Пусть $\alpha, \beta \in E_2^n = \{0,1\}^n$. Расстояние Хэмминга $\rho(\alpha,\beta)$ — это число разрядов с отличающимися значениями у векторов α и β.

- Например, $\rho((1\underline{01}0), (1\underline{10}0)) = 2$.
- Отображение ρ является метрикой на $\{0,1\}^n$: для него выполняются аксиомы метрики.

Аксиомы метрики

1. Неотрицательность: $\rho(\alpha,\beta) \geqslant 0$,
   и индикация тождества: $\rho(\alpha,\beta) = 0 \iff \alpha = \beta$;
2. Симметричность: $\rho(\alpha,\beta) = \rho(\beta,\alpha)$;
3. Неравенство треугольника: $\rho(\alpha,\gamma) \leqslant \rho(\alpha,\beta) + \rho(\beta,\gamma)$.

Коды, исправляющие ошибки

Определение

Шар (замкнутый шар) радиуса r в E_2^n с центром в $\alpha \in E_2^n$ — это множество
$$\overline{S}_r^n(\alpha) = \{\beta \in E_2^n \colon \rho(\alpha, \beta) \leqslant r\}.$$

Определение

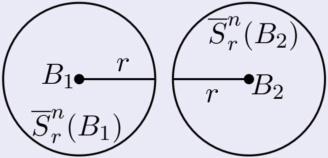
Пусть $C = \{C_1, \ldots, C_k\}$ — равномерный двоичный код.
Кодовым расстоянием кода C называется число
$$\rho_C = \min_{\substack{C_i, C_j \in C \\ C_i \neq C_j}} \rho(C_i, C_j).$$

Коды, исправляющие ошибки

Утверждение

Код C исправляет r ошибок тогда и только тогда, когда $\rho_C \geqslant 2r + 1$.

Доказательство



- Слова, которые могут получится из $B_i \in C$ добавлением не более r ошибок, образуют шар радиуса r с центром в B_i.
- Код исправляет r ошибок $\iff$ эти шары не пересекаются.
- Это значит, что минимальное расстояние между их центрами $\rho_C > 2r$, т.е. $\rho_C \geqslant 2r + 1$.

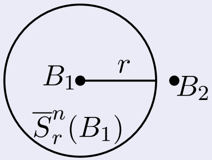
□

Коды, исправляющие ошибки

Утверждение

Код C обнаруживает r ошибок тогда и только тогда, когда $\rho_C \geqslant r + 1$.

Доказательство



- Слова, которые могут получится из $B_i \in C$ добавлением не более r ошибок, образуют шар радиуса r с центром в B_i.
- Код обнаруживает r ошибок $\iff$ эти шары не содержат кодовых слов (кроме своих центров).
- Это значит, что минимальное расстояние между кодовыми словами $\rho_C > r$, т.е. $\rho_C \geqslant r + 1$.

$\square$

Оценка функции $M_r(n)$

Исправление ошибок и количество кодовых слов

- Рассматриваем равномерные двоичные коды со словами длины n. Такой код может иметь от 2 до 2^n кодовых слов.
- Если код имеет 2^s кодовых слов, то каждое из них содержит s бит информации. То есть, для передачи N бит информации нужно будет посылать закодированное сообщение длины $n \cdot \lceil \frac{N}{s} \rceil \approx \frac{n}{s} \cdot N$.
- Таким образом, чем больше число 2^s кодовых слов в коде, тем более компактно кодируются сообщения.
- Но чем больше кодовых слов, тем сложнее добиться большого кодового расстояния, и тем меньше ошибок исправляет код.
- Поэтому имеет смысл рассмотреть число $M_r(n)$: максимальное число кодовых слов в коде, исправляющем r ошибок.
- Отметим: обычно вероятность наличия ошибок зависит от длины сообщения. Поэтому мы фиксируем длину кодовых слов, после чего можно определить, сколько ошибок должен исправлять код.

Оценка функции $M_r(n)$

- $S_r(n) = |\overline{S}_r^n(\alpha)|$ — число элементов в шаре радиуса r в E_2^n.

Утверждение

$$S_r(n) = C_n^0 + C_n^1 + \ldots + C_n^r.$$

Доказательство

- Шар состоит из $r+1$ слоёв (сфер): центр α; наборы, отличные от α в одном разряде; в двух разрядах; …; в r разрядах.
- Число наборов, отличающихся от α в i разрядах, равно числу способов выбрать эти разряды среди n разрядов (без повторений, порядок выбора не важен) — C_n^i.
- Центр шара — это один разряд $(1 = C_n^0)$.
- Всего получаем $S_r(n) = C_n^0 + C_n^1 + \ldots + C_n^r$.
  Это число не зависит от α.

□

Оценка функции $M_r(n)$

Определение

$M_r(n)$ — это максимальное число кодовых слов в коде, исправляющем r ошибок, с кодовыми словами длины n.

Теорема

$$\frac{2^n}{S_{2r}(n)} \leqslant M_r(n) \leqslant \frac{2^n}{S_r(n)}.$$

Доказательство

- 1) Если код с m кодовыми словами исправляет r ошибок, то шары радиуса r в центрах с кодовыми словами не пересекаются.
- Число точек в шаре равно $S_r(n)$, а во всех m шарах $m \cdot S_r(n)$.
- Тогда $m \cdot S_r(n) \leqslant |E_2^n| = 2^n$, т.е. $m \leqslant \frac{2^n}{S_r(n)}$.
- Поскольку это верно для любого кода, то и $M_r(n) \leqslant \frac{2^n}{S_r(n)}$.

Оценка функции $M_r(n)$

Доказательство (продолжение)

- 2) Построим код C, исправляющий r ошибок, в котором не менее $\frac{2^n}{S_{2r}(n)}$ кодовых слов.
- На первом шаге включим в код C произвольный набор α_1.
- Пусть в C уже включены $\alpha_1, \ldots, \alpha_i$, и он исправляет r ошибок.
- Тогда в качестве α_{i+1} выбираем любое слово, которое находится на расстоянии не менее $2r + 1$ от каждого слова $\alpha_1, \ldots, \alpha_i$.
- После включения α_{i+1} в C он также будет исправлять r ошибок.
- Повторяем, пока можно выбрать новое слово.
  Пусть в итоге получилось m кодовых слов.
- Все наборы из E_2^n находятся на расстоянии $2r$ или менее от каких-то кодовых слов, т.е. попадают хотя бы в один шар радиуса $2r$ с центром в кодовом слове.

Оценка функции $M_r(n)$

Доказательство (продолжение)

- Тогда m шаров радиуса $2r$ покрывают все 2^n наборов и E_2^n, т.е. $m \cdot S_{2r}(n) \geqslant 2^n$ (отметим, что шары могут пересекаться).
- Тогда $m \geqslant \frac{2^n}{S_{2r}(n)}$, а значит $M_r(n) \geqslant m \geqslant \frac{2^n}{S_{2r}(n)}$.

$\square$

Лекция 11
Коды Хэмминга. Автоматы.
Схемы из функциональных элементов

Коды Хэмминга

Длина двоичной записи

- Пусть $n \geqslant 1$ — длина кодовых слов и k: $2^{k-1} \leqslant n \leqslant 2^k - 1$.
- Это наибольшее k такое, что $k - 1 \leqslant \log_2 n$, т.е. $k = \lfloor \log_2 n \rfloor + 1$.
- И наименьшее k такое, что $\log_2(n+1) \leqslant k$, т.е. $k = \lceil \log_2(n+1) \rceil$.
- k — это количество двоичных разрядов, достаточное для записи любых чисел от 1 до n.

Двоичные записи чисел

Пусть $m \in \{1, \ldots, n\}$. Запись $m = (m_{k-1} \ldots m_1 m_0)_2$ будет означать, что двоичная запись числа m имеет вид $m_{k-1} \ldots m_1 m_0$, где $m_i \in \{0, 1\}$, $i = \overline{0,\ k-1}$.

Коды Хэмминга

Множества D_p

Пусть k: $2^{k-1} \leqslant n \leqslant 2^k - 1$. Тогда при $p = \overline{0,\ k-1}$ обозначаем

$$D_p = \{m \in \{1, \ldots, n\} : m = (m_{k-1} \ldots m_1 m_0)_2,\ \text{где } m_p = 1\}$$

Примеры

- $D_0 = \{1, 3, 5, 7, 9, \ldots\}$.
- $D_1 = \{2, 3,\ 6, 7,\ 10, 11,\ \ldots\}$.
- $D_2 = \{4, 5, 6, 7,\ 12, 13, 14, 15,\ \ldots\}$.

Коды Хэмминга

Определение

Пусть $k\colon 2^{k-1} \leqslant n \leqslant 2^k - 1$. Кодом Хэмминга порядка n называется множество всех наборов $(a_1, \ldots, a_n) \in E_2^n$, удовлетворяющих системе уравнений (все сложения производятся по модулю 2):

$$\begin{cases} \bigoplus\limits_{j \in D_0} a_j = 0 \\ \bigoplus\limits_{j \in D_1} a_j = 0 \\ \ldots \\ \bigoplus\limits_{j \in D_{k-1}} a_j = 0 \end{cases}$$

Коды Хэмминга

Теорема

Код Хэмминга порядка n содержит 2^{n-k} наборов, где $k = \lfloor \log_2 n \rfloor + 1$, и исправляет одну ошибку.

Доказательство

- 1) Число наборов в коде Хэмминга равно числу решений системы

$$\begin{cases} \bigoplus\limits_{j \in D_0} a_j = 0 \\ \bigoplus\limits_{j \in D_1} a_j = 0 \\ \ldots \\ \bigoplus\limits_{j \in D_{k-1}} a_j = 0 \end{cases} \iff \begin{cases} a_1 = \bigoplus\limits_{j \in D_0 \setminus \{a_1\}} a_j \\ a_2 = \bigoplus\limits_{j \in D_1 \setminus \{a_2\}} a_j \\ \ldots \\ a_{2^{k-1}} = \bigoplus\limits_{j \in D_{k-1} \setminus \{a_{2^{k-1}}\}} a_j \end{cases}$$

Коды Хэмминга

Доказательство свойств кода Хэмминга (продолжение)

- Заметим, что переменные $a_1, a_2, a_4, \ldots, a_{2^{k-1}}$ не встречаются в правых частях уравнений.
- Чтобы получить все решения системы, можно произвольным образом выбирать все a_j, где $j \neq 2^l,\ l = \overline{0,\ k-1}$. Оставшиеся a_j определяются однозначно по уравнениям.
- Количество произвольно выбираемых a_j равно $n-k$. Значит, система имеет 2^{n-k} решений. Столько же и наборов в коде.
- 2) Докажем, что код Хэмминга исправляет одну ошибку. Приведём алгоритм получения исходного кодового слова по кодовому слову с одной ошибкой.
- Пусть передавалось слово $\alpha = a_1 \ldots a_n$. Пусть произошла одна ошибка в разряде d и получено слово $\beta = b_1 \ldots b_n,\ a_d \neq b_d$.
- Поскольку $d \in \{1, \ldots, n\}$, можно записать $d = (d_{k-1} \ldots d_0)_2$.

Коды Хэмминга

Доказательство свойств кода Хэмминга (продолжение)

- Опишем, как по слову β найти разряд ошибки d.
- Вычисляем $\delta_p = \bigoplus\limits_{j \in D_p} b_j,\ p = \overline{0,\, k-1}$.

Утверждение

В условиях теоремы о коде Хэмминга $(\delta_{k-1} \dots \delta_0)_2 = d$.

Доказательство

- Поскольку α является кодовым словом, имеем $\bigoplus\limits_{j \in D_p} a_j = 0$.
- Тогда $\delta_p = 1 \iff \bigoplus\limits_{j \in D_p} b_j \neq 0 = \bigoplus\limits_{j \in D_p} a_j$, т.е. α и β отличаются в разряде из D_p. Это возможно тогда и только тогда, когда $d \in D_p$, т.е., по определению D_p, $d_p = 1$. Получается $\delta_p = d_p,\ p = \overline{0,\, k-1}$.
- Тогда $(\delta_{k-1} \dots \delta_0)_2 = (d_{k-1} \dots d_0)_2 = d$. □

Коды Хэмминга

Доказательство свойств кода Хэмминга (продолжение)

- Итак, если в слове β есть ровно одна ошибка, то указанный алгоритм позволяет найти разряд d этой ошибки.
- Заменяя b_d на $\overline{b}_d$, можно получить исходное кодовое слово α.
- Если в слове β нет ошибок, то β является кодовым словом. Тогда $\bigoplus\limits_{j \in D_p} b_j = 0$, и указанный алгоритм получит $\delta_p = 0, \ p = \overline{0, k-1}$.
- В этом случае исходное кодовое слово α совпадает со словом β.
- Таким образом, при наличии в кодовом слове не более одной ошибки, можно однозначно получить исходное кодовое слово.

$\square$

Коды Хэмминга

Теорема

$$\frac{2^n}{2n} \leqslant M_1(n) \leqslant \frac{2^n}{n+1}.$$

Доказательство

- Напомним, что $M_r(n)$ — это максимально возможное число кодовых слов в коде, исправляющем r ошибок.
- 1) Раньше была доказана общая оценка: $\frac{2^n}{S_{2r}(n)} \leqslant M_r(n) \leqslant \frac{2^n}{S_r(n)}$, где $S_r(n) = C_n^0 + C_n^1 + \ldots + C_n^r$. Тогда $S_1(n) = C_n^0 + C_n^1 = 1 + n$.
- Подставим в правое неравенство $r = 1$. Получаем $M_1(n) \leqslant \frac{2^n}{n+1}$.
- 2) Рассмотрим код Хэмминга порядка n. Пусть у него m кодовых слов. Ранее доказано, что $m = 2^{n-k}$, где $k = \lfloor \log_2 n \rfloor + 1$.
- Число k можно оценить сверху: $k = \lfloor \log_2 n \rfloor + 1 \leqslant \log_2 n + 1$.
- Тогда m оценивается снизу: $m \geqslant 2^{n-(\log_2 n+1)} = \frac{2^n}{2n}$.

Коды Хэмминга

Доказательство оценки $M_1(n)$ (продолжение)

- Поскольку $M_1(n)$ — максимальное число слов в коде, исправляющем одну ошибку, а код Хэмминга исправляет одну ошибку, имеем $M_1(n) \geqslant m \geqslant \frac{2^n}{2n}$.

$\square$

Замечание

1. Нижнюю оценку $\frac{2^n}{2n}$ нельзя получить из $\frac{2^n}{S_{2r}(n)} \leqslant M_r(n)$. Поскольку $S_2(n) \approx \frac{n^2}{2}$, получится худшая оценка $\frac{2^n}{n^2/2}$. Лучшая оценка получается именно благодаря коду Хэмминга.
2. При $n = 2^k - 1$ код Хэмминга имеет $\frac{2^n}{n+1}$ слов, и $M_1(n) = \frac{2^n}{n+1}$.

Автоматы

Бесконечные последовательности

Пусть A — конечное непустое множество.

- A^∞ — это множество счётно-бесконечных последовательностей вида $a_{i_1}a_{i_2}\ldots$, где $a_{i_n} \in A$ при $n \in \mathbb{N}$.
- Пусть $a = a_{i_1}a_{i_2}\ldots \in A^\infty$. Обозначим $a(t) = a_{i_t}$ при $t \in \mathbb{N}$.
- Для введения индексации с нуля пишем $a = a(0)a(1)\ldots \in A^\infty$.
- Конкатенация слова $u = u_1 \ldots u_k \in A^*$ и последовательности $a = a(1)a(2)\ldots \in A^\infty$ — это последовательность

  $$u * a = ua = u_1 \ldots u_k a(1)a(2)\ldots \in A^\infty.$$

  При этом для любого $a \in A^\infty$ определяем $\Lambda a = a$.
- Бесконечное повторение слова $u = u_1 \ldots u_k \in A^*$, $u \neq \Lambda$ есть

  $$u^\omega = u_1 \ldots u_k u_1 \ldots u_k \ldots \in A^\infty.$$

Автоматы

Определение автомата: структура

Конечный автомат — это система $\mathcal{A} = (A, B, Q, F, G, q_0)$, где

- $A \neq \varnothing$ — конечный входной алфавит,
- $B \neq \varnothing$ — конечный выходной алфавит,
- $Q \neq \varnothing$ — конечное множество состояний,
- $F\colon A \times Q \to B$ — функция выходов,
- $G\colon A \times Q \to Q$ — функция переходов,
- $q_0 \in Q$ — начальное состояние.

- В теории алгоритмов существует много разных вариантов конечных автоматов.
- Мы рассматриваем вариант, который называют конечными автоматами-преобразователями или автоматами Мили.

Автоматы

Определение автомата: функционирование

- На вход автомату подаётся бесконечное слово $x \in A^\infty$. На выходе получается бесконечное слово $y \in B^\infty$.
- Автомат работает в дискретном времени: $t = 1, 2, \ldots$ На каждом такте t автомату подаётся очередной символ $x(t)$.
- После каждого такта t автомат находится в состоянии $q(t)$ из Q. До первого такта он находится в состоянии $q(0) = q_0$.
- На каждом такте t автомат меняет своё состояние $q(t)$ и выдаёт символ выхода $y(t)$ согласно канонический уравнениям:

$$\begin{cases} y(t) = F(x(t), q(t-1)), \\ q(t) = G(x(t), q(t-1)), \\ q(0) = q_0. \end{cases}$$

$$\begin{aligned} \text{Вход: } x &= x(1)x(2)\ldots \\ \text{Выход: } y &= y(1)y(2)\ldots \\ \text{Состояния: } q &= q(0)q(1)q(2)\ldots \end{aligned}$$

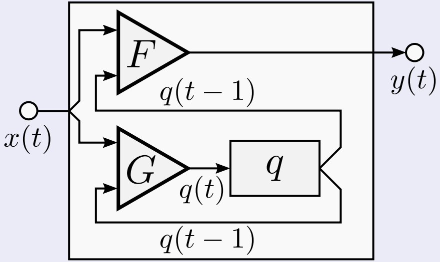
Автоматы

Определение автомата: функционирование

- Автомат однозначно генерирует последовательности y и q:
  - Изначально задано $q(0) = q_0$;
  - $t = 1$: автомат получает на вход $x(1)$ и вычисляет
    $y(1) = F(x(1), q(0))$,
    $q(1) = G(x(1), q(0))$;
  - $t = 2$: автомат получает на вход $x(2)$ и вычисляет
    $y(2) = F(x(2), q(1))$,
    $q(2) = G(x(2), q(1))$;
  - Аналогично продолжается при $t = \overline{3, \infty}$.
- Таким образом, автомат $\mathcal{A}$ реализует функцию $\varphi\colon A^\infty \to B^\infty$, которая для каждого x выдаёт выдаёт генерируемую автоматом последовательность $y = \varphi(x)$.
- Если автомату на вход подано конечное слово $x(1) \ldots x(k)$, автомат выдаст конечное слово $y(1) \ldots y(k)$ той же длины.

Автоматы

Схема работы автомата



$t = 1, 2, \dots$ — время

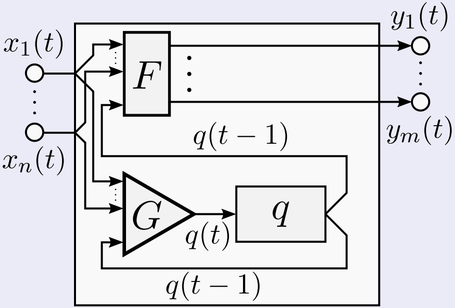
В начальный момент $q = q_0$

$$\begin{cases} y(t) = F(x(t),\, q(t-1)), \\ q(t) = G(x(t),\, q(t-1)), \\ q(0) = q_0. \end{cases}$$

Автоматы

- Если $A = C^n$, то у автомата несколько входов $x_1, \dots, x_n \in C^\infty$.
- Если $B = D^m$, то у автомата несколько входов $y_1, \dots, y_m \in D^\infty$.

Автомат с несколькими входами и выходами



$t = 1, 2, \dots$ — время

В начальный момент $q = q_0$

$$x(t) = (x_1(t), \dots, x_n(t))$$

$$y(t) = (y_1(t), \dots, y_m(t))$$

$$\begin{cases} y(t) = F(x(t),\, q(t-1)), \\ q(t) = G(x(t),\, q(t-1)), \\ q(0) = q_0. \end{cases}$$

Автоматы

Определение

Отображение $\varphi\colon A^\infty \to B^\infty$ называется автоматным, если оно реализуется некоторым автоматом с входным алфавитом A и выходным алфавитом B.

Пример

- $A = B = Q = \{0, 1\}$, канонические уравнения

$$\begin{cases} y(t) = q(t-1), \\ q(t) = x(t), \\ q(0) = 0. \end{cases}$$

$$\begin{array}{rl} \text{Вход: } x = & x(1)x(2)x(3)\dots \\ \text{Выход: } y = & 0\ \ x(1)x(2)\dots \\ \text{Состояния: } q = & 0\ x(1)x(2)x(3)\dots \end{array}$$

- Этот автомат реализует функцию единичная задержка: $\mathfrak{z}(x) = 0x$.

Автоматы

Определение

Диаграмма Мура для автомата (A, B, Q, F, G, q_0) — это ориентированный псевдограф с множеством вершин Q.

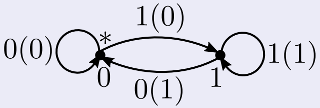
- В этом графе для каждой пары $(a, q) \in A \times Q$ проводится дуга из вершины q в вершину $q' = G(a, q)$ с пометкой $a(b)$, где $b = F(a, q)$.
- Вершина q_0 (начальное состояние) помечается символом $*$.

- Диаграмма Мура однозначно задаёт автомат.

Пример

- Единичная задержка: $A = B = Q = \{0, 1\}$

$$\begin{cases} y(t) = q(t-1), \\ q(t) = x(t), \\ q(0) = 0. \end{cases}$$



Схемы из функциональных элементов

Определение

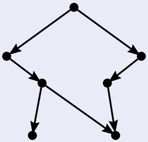
Пусть $G = (V, E)$ — ориентированный граф и $v \in V$.

- Полустепень захода $\deg^{-}(v)$ — количество дуг, заходящих в v.
- Полустепень исхода $\deg^{+}(v)$ — количество дуг, исходящих из v.
- Вершина v называется истоком, если $\deg^{-}(v) = 0$.

Определение

Ориентированный граф G называется ациклическим, если в нём нет ориентированных циклов.

Пример



Схемы из функциональных элементов

Определение СФЭ: структура

Схема из функциональных элементов (СФЭ) — это ориентированный ациклический граф, в котором

- Полустепень захода каждой вершины равна 0, 1 или 2;
- Каждому истоку (входу) приписана переменная x_i (входная переменная);
- Каждой вершине v, в которую входит 1 дуга, сопоставлена булева функция $\overline{x}$;

  Каждой вершине v, в которую входит 2 дуги, сопоставлена булева функция xy или $x \vee y$;

  Вершина с приписанной функцией называется функциональным элементом;
- Некоторым вершинам (выходам) приписаны переменные y_j (выходные переменные). Входы тоже могут быть выходами.

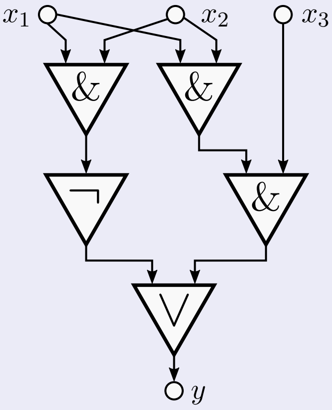
Схемы из функциональных элементов

Определение СФЭ: функционирование

- Пусть схема имеет входные переменные $x_1, \ldots, x_n$ и выходные переменные $y_1, \ldots, y_m$.
- Для каждой вершины v определим булеву функцию $f_v(x_1, \ldots, x_n)$, которая реализуется в вершине v:
    1. Если v — исток, которому приписана переменная x_i, то $f_v(x_1, \ldots, x_n) = x_i$.
    2. Пусть вершине v приписана функция $\overline{x}$ и в неё заходит одна дуга из вершины u, причём функция f_u уже определена. Тогда $f_v = \overline{f_u(x_1, \ldots, x_n)}$.
    3. Пусть вершине v приписана функция $g(x, y)$ (xy или $x \vee y$) и в неё заходят две дуги из вершин u_1, u_2, причём функции f_{u_1}, f_{u_2} уже определены. Тогда $f_v = g(f_{u_1}(x_1, \ldots, x_n), f_{u_2}(x_1, \ldots, x_n))$.
- Схема реализует набор функций, которые реализуются в вершинах-выходах.

Схемы из функциональных элементов

Пример: формула

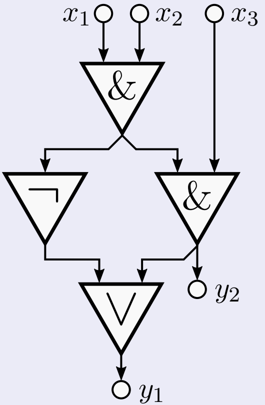


$$y = \overline{x_1 x_2} \vee (x_1 x_2) x_3$$

- Число входов равно числу различных символов переменных.
- Во входы не заходят дуги, но из них может исходить любое число дуг.
- Выход один (обозначается дугой, ведущей в дополнительную вершину).
- В функциональный элемент входит по одной дуге на каждый аргумент его функции, а исходит ровно одна дуга.
- Формула без входов представляет собой дерево.
- Формула реализует булеву функцию.

Схемы из функциональных элементов

Пример: СФЭ



$$y_1 = \overline{x_1 x_2} \vee (x_1 x_2) x_3$$
$$y_2 = x_1 x_2 x_3$$

Отличия от формулы:

- Из каждого функционального элемента может исходить любое число дуг.
- СФЭ может не являться деревом, но в ней нет ориентированных циклов.
- Выходов может быть несколько.
- СФЭ реализует набор булевых функций (булев оператор)

$$F\colon \{0,1\}^n \to \{0,1\}^m.$$

- В примере $F = (\overline{x_1 x_2} \vee x_1 x_2 x_3,\ x_1 x_2 x_3)$.

Схемы из функциональных элементов

Замечания

- Мы рассматриваем СФЭ с функциональными элементами $\&, \vee, \neg$. В общем случае рассматривают и СФЭ с другими функциональными элементами.
- Формулу (над множеством $\{xy,\ x \vee y,\ \overline{x}\}$) можно рассматривать как частный случай СФЭ с функциональными элементами $\&, \vee, \neg$.
- Поскольку $\{xy,\ x \vee y,\ \overline{x}\}$ — полная система, любая булева функцией реализуется формулой над $\{xy,\ x \vee y,\ \overline{x}\}$, а значит и схемой из функциональных элементов с элементами $\&, \vee, \neg$.

Лекция 12
Схемы из функциональных элементов с задержками. Автоматность осуществляемых ими отображений. Моделирование автоматной функции СФЭ с задержками

Схемы из функциональных элементов с задержками

Понятие схемы с задержками

Схема из функциональных элементов с задержками (СФЭЗ) определяется так же, как СФЭ, со следующими отличиями:

- Новый элемент $\mathfrak{z}$: некоторым вершинам, в которые входит одна дуга, может быть сопоставлена автоматная функция $\mathfrak{z}(x)$.
- В графе могут быть ориентированные циклы, но каждый ориентированный цикл должен проходить хотя бы через одну вершину, которой приписана функция задержки $\mathfrak{z}(x)$.

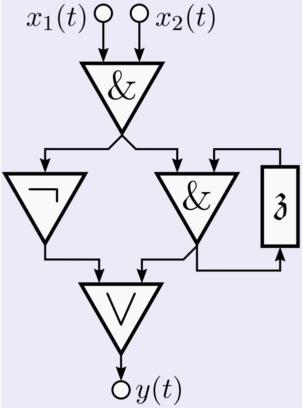
Схемы из функциональных элементов с задержками

Функионирование схемы с задержками

- Пусть схема имеет входные переменные $x_1, \ldots, x_n$ и выходные переменные $y_1, \ldots, y_m$.
- Считаем, что идёт дискретное время: $t = 1, 2, \ldots$. В каждый момент времени на каждый вход поступает символ 0 или 1.
- Считаем, что функциональные элементы срабатывают мгновенно, преобразовывая сигналы 0 и 1 в соответствии с приписанными им функциями.
- Элемент задержки работает как функция $\mathfrak{z}(x)$: в первый момент выдаёт 0; в каждый момент времени запоминает вход и выдаёт его на следующем такте.
- Схема с задержками реализует набор функций над бесконечными словами: $(E_2^n)^\infty \to (E_2^m)^\infty$, которые берутся из выходов схемы.

Схемы из функциональных элементов с задержками

Пример: схема из функциональных элементов и задержек



Отличия от СФЭ:

- Допускаются элементы, реализующие функцию единичной задержки.
- Допускаются ориентированные циклы, но каждый из них должен иметь задержку.
- Схема функцию

$$F\colon (\{0,1\}^n)^\infty \to (\{0,1\}^m)^\infty.$$

- В примере

$$\begin{cases} y(t) = \overline{x_1(t)x_2(t)} \vee x_1(t)x_2(t)q(t-1), \\ q(t) = x_1(t)x_2(t)q(t-1), \\ q(0) = 0. \end{cases}$$

Автоматность отображения СФЭ с задержками

Теорема

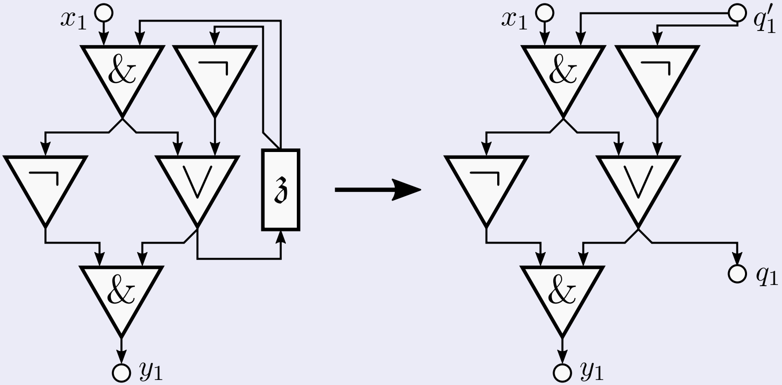
Любая схема из функциональных элементов с задержками осуществляет автоматное отображение.

Доказательство

- Пусть дана СФЭЗ Σ со входными переменными $x_1, \ldots, x_n$, выходными переменными $y_1, \ldots, y_m$ и задержками $R_1, \ldots, R_r$.
- Для $i = \overline{1, r}$: пусть в задержку R_i идёт дуга из v_i. Удалим эту дугу, а саму вершину R_i превратим во вход с входной переменной q_i'. Вершину v_i пометим выходной переменной q_i.
- В Σ каждый цикл проходит через задержку. Поэтому после указанных преобразований будет получен граф без циклов.
- Т.е. будет получена СФЭ Σ со входами $x_1, \ldots, x_n, q_1', \ldots, q_r'$ и выходами $y_1, \ldots, y_m, q_1, \ldots, q_r$.

Автоматность отображения СФЭ с задержками

Доказательство (продолжение)



- В примере задержка замена на вход q'_1, дуга в задержку удалена, а функциональному элементу $\vee$ приписан новый выход q_1 (отображён дугой, выходящей из $\vee$, с кругом на конце).

Автоматность отображения СФЭ с задержками

Доказательство (продолжение)

- По определению функционирования СФЭ получаем зависимости

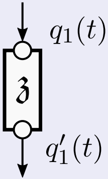
$$\begin{cases} y_i = f_i(x_1, \ldots, x_n,\, q'_1, \ldots, q'_r),\ \ i = \overline{1,m}, \\ q_j = g_j(x_1, \ldots, x_n,\, q'_1, \ldots, q'_r),\ \ j = \overline{1,r}, \end{cases}$$

  где $f_1, \ldots, f_m, g_1, \ldots, g_r$ — булевы функции.
- Эта зависимость одинакова в каждый момент времени функционирования схемы:

$$\begin{cases} y_i(t) = f_i(x_1(t), \ldots, x_n(t),\, q'_1(t), \ldots, q'_r(t)),\ \ i = \overline{1,m}, \\ q_j(t) = g_j(x_1(t), \ldots, x_n(t),\, q'_1(t), \ldots, q'_r(t)),\ \ j = \overline{1,r}. \end{cases}$$

Автоматность отображения СФЭ с задержками

Доказательство (продолжение)



- В исходной СФЭ с задержками $q'_j(t) = \mathfrak{z}(q_j(t))$, т.е. $q'_j(t) = q_j(t-1)$ и $q'_j(1) = 0$. Последнее запишем как $q_j(0) = 0$.
- Тогда функционирование СФЭ с задержками описывается

$$\begin{cases} y_i(t) = f_i(x_1(t), \ldots, x_n(t),\, q_1(t-1), \ldots, q_r(t-1)),\ \ i = \overline{1,m}, \\ q_j(t) = g_j(x_1(t), \ldots, x_n(t),\, q_1(t-1), \ldots, q_r(t-1)),\ \ j = \overline{1,r}, \\ q_1(0) = \ldots = q_r(0) = 0. \end{cases}$$

Автоматность отображения СФЭ с задержками

Доказательство (продолжение)

- Введём переменные:
  - $X = (x_1, \ldots, x_n)$ принимает значения из E_2^n;
  - $Y = (y_1, \ldots, y_m)$ принимает значения из E_2^m;
  - $Q = (q_1, \ldots, q_r)$ принимает значения из E_2^r.
- Обозначим $F(X, Q) = (f_1(X, Q), \ldots, f_m(X, Q))\colon E_2^n \times E_2^r \to E_2^m$ и $G(X, Q) = (g_1(X, Q), \ldots, g_r(X, Q))\colon\ E_2^n \times E_2^r \to E_2^r$.
- Тогда функционирование СФЭ с задержками описывается

$$\begin{cases} Y(t) = F(X(t),\, Q(t-1)), \\ Q(t) = G(X(t),\, Q(t-1)), \\ Q(0) = (0, \ldots, 0). \end{cases}$$

- А это канонические уравнения автомата $\mathcal{A} = (E_2^n,\, E_2^m,\, E_2^r,\, F,\, G,\, q_0)$, где $q_0 = (0, \ldots, 0)$.

□

Моделирование автоматной функции СФЭ с задержками

Определение

Пусть даны 2 автомата
$\mathcal{A}_1 = (A_1, B_1, Q_1, F_1, G_1, q_{01})$ и $\mathcal{A}_2 = (A_2, B_2, Q_2, F_2, G_2, q_{02})$
и инъективные отображения $K_1 \colon A_1 \to A_2$ и $K_2 \colon B_1 \to B_2$.

Будем говорить, что автомат $\mathcal{A}_2$ моделирует автомат $\mathcal{A}_1$ при отображениях K_1, K_2, если для любой входной последовательности $a_1 a_2 \ldots \in A_1^\infty$ верно:

если $\mathcal{A}_1$ отображает $a_1 a_2 \ldots \in A_1^\infty$ в $b_1 b_2 \ldots \in B_1^\infty$,
то $\mathcal{A}_2$ отображает $K_1(a_1) K_1(a_2) \ldots \in A_2^\infty$ в $K_2(b_1) K_2(b_2) \ldots \in B_2^\infty$.

- Моделирование позволяет закодировать входной и выходной алфавит (например, словами из нулей и единиц), но при этом сохранить функциональность автомата.
- Мы будем моделировать произвольные автоматы с помощью СФЭ с задержками, которые работают в алфавитах E_2^n, E_2^m.

Моделирование автоматной функции СФЭ с задержками

Теорема

Для любого автомата $\mathcal{A} = (A, B, Q, F, G, q_0)$ *существует моделирующая его СФЭ с задержками.*

Доказательство

- Выберем натуральные числа n, m, r так, что $|A| \leqslant 2^n,\ |B| \leqslant 2^m,\ |Q| \leqslant 2^r$.
- Выберем произвольные инъективные отображения

$$K_1\colon A \to E_2^n,$$
$$K_2\colon B \to E_2^m,$$
$$K_3\colon Q \to E_2^r,$$

  причём так, что $K_3(q_0) = \theta^r = (0, \dots, 0)$.
- В силу выбора n, m, r такие отображения существуют.

Моделирование автоматной функции СФЭ с задержками

Доказательство (продолжение)

- Рассмотрим функции F' и G' такие, что для любых $a \in A$ и $q \in Q$
  $F'(K_1(a),\, K_3(q)) = K_2(F(a,q)), \quad F'\colon E_2^n \times E_2^r \to E_2^m,$
  и $G'(K_1(a),\, K_3(q)) = K_3(G(a,q)), \quad G'\colon E_2^n \times E_2^r \to E_2^r.$
- Эти условия могут не полностью задавать функции F' и G'. Они заданы на наборах, которые являются кодами пар $(a,q) \in A \times Q$. На оставшихся наборах доопределим их произвольно.
- Тогда автомат $\mathcal{A}' = (E_2^n, E_2^m, E_2^r, F', G', \theta^r)$ моделирует $\mathcal{A}$.
- Выпишем канонические уравнения $\mathcal{A}'$:

$$\begin{cases} Y(t) = F'(X(t),\, Q(t-1)), \\ Q(t) = G'(X(t),\, Q(t-1)), \\ Q(0) = \theta^r. \end{cases}$$

Моделирование автоматной функции СФЭ с задержками

Доказательство (продолжение)

- Переменная X принимает значения из E_2^n, поэтому можно считать её вектором булевых переменных. Аналогично для Y и Q:

$$X(t) = (x_1(t), \ldots, x_n(t)),$$
$$Y(t) = (y_1(t), \ldots, y_m(t)),$$
$$Q(t) = (q_1(t), \ldots, q_r(t)),$$

- Тогда канонические уравнения $\mathcal{A}'$ можно записать в виде

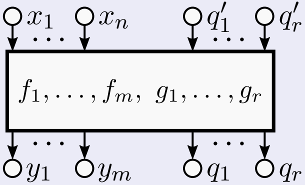
$$\begin{cases} y_i(t) = f_i(x_1(t), \ldots, x_n(t), q_1(t-1), \ldots, q_r(t-1)), \ i = \overline{1, m}, \\ q_j(t) = g_j(x_1(t), \ldots, x_n(t), q_1(t-1), \ldots, q_r(t-1)), \ j = \overline{1, r}, \\ q_1(0) = \ldots = q_r(0) = 0, \end{cases}$$

где f_i, $i = \overline{1, m}$, g_j, $j = \overline{1, r}$ — некоторые булевы функции.

Моделирование автоматной функции СФЭ с задержками

Доказательство (продолжение)

- Теперь построим СФЭ с задержками, реализующую эти уравнения.
- Сначала построим СФЭ, реализующую набор булевых функций $(f_1, \ldots, f_m,\ g_1, \ldots, g_r)$. Она будет иметь входы $x_1, \ldots, x_n,\ q'_1, \ldots, q'_r$ и выходы $y_1, \ldots, y_m,\ q_1, \ldots, q_r$.

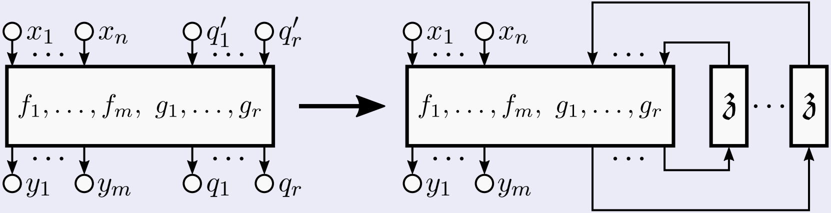


- Эта схема в каждый момент времени вычисляет $y_1(t), \ldots, y_m(t)$ и $q_1(t), \ldots, q_r(t)$ по $x_1(t), \ldots, x_n(t)$ и $q'_1(t), \ldots, q'_r(t)$.

Моделирование автоматной функции СФЭ с задержками

Доказательство (продолжение)

- Для всех $i = \overline{1, r}$, в соответствии с каноническими уравнениями, $q'_i(t) = q_i(t-1)$, причём $q'_i(1) = 0$. Это значит, что $q'_i(t) = \mathfrak{z}(q_i(t))$.
- Чтобы реализовать эти зависимости в схеме, нужно каждый выход q_i соединить со входом q'_i через задержку. При этом пометки входных и выходных переменных на этих вершинах удаляются.



- Получили СФЭ с задержками, реализующую то же отображение, что и автомат $\mathcal{A}'$, а значит, моделирующую автомат $\mathcal{A}$.

□

Моделирование автоматной функции СФЭ с задержками

Моделирование реальных систем

- Автомат-преобразователь — это модель системы, которая работает неопределённо долгое время, в каждый момент получает определённый входные сигналы и выдаёт некоторые результаты.
- Процессор компьютера является автоматом-преобразователем:
  - Состояния (конечная память) — регистры.
  - Входные сигналы — данные из оперативной памяти и с внешних устройств (клавиатуры, мыши).
  - Выходные сигналы — данные для записи в оперативную память, позиция чтения/записи в оперативной памяти, вывод на внешние устройства (дисплей).
- Автомат — вычислительно слабое устройство, так как имеет лишь конечную память. Компьютер является универсальным за счёт наличия (условно) бесконечной оперативной памяти.
- Моделирование автомата с помощью СФЭ с задержками позволяет строить процессоры из простых элементов.

Лекция 13
Теорема Мура. Схемный сумматор

Отличимые состояния автомата

- Будем рассматривать автоматы без начального состояния: $\mathcal{A} = (A, B, Q, F, G)$.

Работа автомата на конечных словах

- Расширим функции F и G на множество $A^* \times Q$.
- Пусть $v \in A^*$ (слово в алфавите A) и $q \in Q$.
  - $F(v, q)$ — это слово, которое выдаст автомат при работе над словом v, начатой из состояния q. Ясно, что $|F(v, q)| = |v|$.
  - $G(v, q)$ — это состояние, в котором окажется автомат после работы над словом v, начатой из состояния q.
- Нетрудно видеть, что если $|v| = 1$, то эти функции совпадают с функциями F и G из определения автомата.

Отличимые состояния автомата

Определение

- Два состояния q_i и q_j автомата $\mathcal{A} = (A, B, Q, F, G)$ отличимы словом v, если $F(v, q_i) \neq F(v, q_j)$.
- Два состояния q_i и q_j автомата $\mathcal{A}$ отличимы, если они отличимы хотя бы одним словом $v \in A^*$.

- Слово v называют экспериментом, отличающим q_i и q_j, а его длину — длиной эксперимента.

Теорема Мура

Лемма

Пусть в автомате $\mathcal{A} = (A, B, Q, F, G)$ состояния q_u и q_v отличимы некоторым словом длины p и не отличимы никакими словами меньшей длины.
Тогда для любого $k \in \{1, \ldots, p\}$ в Q существуют два состояния, которые отличимы некоторым словом длины k и не отличимы никакими словами длины меньшей k.

Доказательство

- По условию существует слово $\alpha = a(1) \ldots a(p) \in A^*$ такое, что $F(\alpha, q_u) \neq F(\alpha, q_v)$.
- Пусть $F(\alpha, q_u) = \beta = b(1) \ldots b(p)$
  и $F(\alpha, q_v) = \gamma = c(1) \ldots c(p)$. Здесь $\beta, \gamma \in B^*$, $\beta \neq \gamma$.
- Поскольку q_u и q_v не отличимы словами длины меньшей p, слова β и γ должны отличаться в последнем символе, т.е. $b(p) \neq c(p)$.

Теорема Мура

Доказательство леммы (продолжение)

- Пусть $k \in \{1, \dots, p\}$. Рассмотрим состояния
  $q'_u = G(a(1)\dots a(p-k),\, q_u)$
  и $q'_v = G(a(1)\dots a(p-k),\, q_v)$.
- Напомним, что $b(p) \neq c(p)$. По принципу работы автомата
  $F(a(p-k+1)\dots a(p),\, q'_u) = b(p-k+1)\dots b(p)$
  и $F(a(p-k+1)\dots a(p),\, q'_v) = c(p-k+1)\dots c(p)$.
- Тогда q'_u и q'_v отличимы словом $a(p-k+1)\dots a(p)$ длины k.
- Докажем, что q'_u и q'_v не отличимы словами длины меньшей k.
- Пусть q'_u и q'_v отличимы словом α' длины меньшей k. Это значит, что $F(\alpha', q'_u) \neq F(\alpha', q'_v)$.
- Тогда $F(a(1)\dots a(p-k)\alpha',\, q_u) = b(1)\dots b(p-k)F(\alpha', q'_u)$
  и $F(a(1)\dots a(p-k)\alpha',\, q_v) = c(1)\dots c(p-k)F(\alpha', q'_v)$.

Теорема Мура

Доказательство леммы (продолжение)

- Значит, состояния q_u и q_v отличимы словом $a(1)\dots a(p-k)\alpha'$ длины меньшей p, что невозможно по условию теоремы.
- Противоречие означает, что q'_u и q'_v не отличимы словами длины меньшей k.

$\square$

Теорема Мура

Теорема (Мур)

Пусть в автомате $\mathcal{A} = (A, B, Q, F, G)$ число состояний $|Q| = r$ и состояния q_i и q_j отличимы. Тогда состояния q_i и q_j отличимы некоторым словом длины не более $r - 1$.

Доказательство

- Пусть α — слово минимальной длины, на котором состояния q_i и q_j отличимы. Обозначим $p = |\alpha|$.
- Нужно доказать, что $p \leqslant r - 1$.
- При $m = \overline{0, p}$ рассмотрим бинарное отношение на множестве Q

$$R_m(q', q'') \equiv (q' \text{ и } q'' \text{ не отличимы словами длины } m).$$

- Отметим, что если q' и q'' не отличимы словами длины m, то они не отличимы и словами меньшей длины.

Теорема Мура

Доказательство (продолжение)

- Все отношения R_m являются отношениями эквивалентности:
  1. $R_m(p,p)$: любое состояния не отличимо само от себя;
  2. $R_m(p,q) = R_m(q,p)$: в определении R_m порядок состояний не играет роли;
  3. $R_m(p,q) \mathbin{\&} R_m(q,s) \to R_m(p,s)$: если p и q дают одинаковые результаты на словах определённой длины, и q и s дают на них одинаковые результаты, то p и s будут давать одинаковые (те же самые) результаты.
- Отношение R_0 является тождественно истинным: любые состояния не отличимы словами длины 0.
- Рассмотрим разбиение Q на классы эквивалентности относительно R_m. Число этих классов обозначим $s(m)$. Ясно, что $s(0) = 1$.

Теорема Мура

Доказательство (продолжение)

- Если два состояния отличимы словами длины $m-1$, то они тем более отличимы словами длины m.
- Это значит, что если два состояния принадлежат разным классам эквивалентности R_{m-1}, то они принадлежит разным классам эквивалентности R_m.
- Таким образом, при увеличении m классы эквивалентности могут только «распадаться» на несколько классов, и $s(m) \geqslant s(m-1)$.
- Напомним, что состояния q_i и q_j отличимы словом длины p и не отличимы словами меньшей длины.
- Тогда, если $m \leqslant p$, то по лемме существуют состояния q'_m, q''_m, отличимые словами длины m, но не отличимые словами длины $m-1$, т.е. такие, что $\neg R_m(q'_m, q''_m)$ и $R_{m-1}(q'_m, q''_m)$.

Теорема Мура

Доказательство (продолжение)

- Это значит, что существуют два состояния, которые принадлежат одному классу эквивалентности R_{m-1} и разным классам R_m: при переходе от $m-1$ к m число классов возросло.
- Итак, при $1 \leqslant m \leqslant p$ имеем строгое неравенство $s(m-1) < s(m)$:

$$1 = s(0) < s(1) < \ldots < s(p-1) < s(p) \leqslant r.$$

- Тогда $s(1) \geqslant 2,\ s(2) \geqslant 3,\ \ldots,\ s(p) \geqslant p+1$, а значит, $r \geqslant p+1$.
- Отсюда получаем $p \leqslant r-1$.

$\square$

- Отметим, что отношение «состояния q' и q'' неотличимы» совпадает с $R_{r-1}(q', q'')$ и является отношением эквивалентности.

Теорема Мура

- Количество состояний у автомата — это мера памяти, которую он использует для вычислений.

Алгоритм оптимизации автомата

- Строим диаграмму Мура для автомата.
- Удаляем из диаграммы состояния (и дуги), недостижимые по ориентированным путям из начального состояния.
- Разбиваем множество состояний на классы эквивалентности по отношению неотличимости (пользуемся теоремой Мура).
- «Склеиваем» каждый класс эквивалентности в одно состояние с сохранением дуг.
- Удаляем дубликаты дуг.

- Можно показать, что указанный алгоритм строит автомат с минимальным числом состояний для данной функции.

Схемный сумматор

Определение

Схемный сумматор порядка n — это СФЭ с $2n$ входами $x_1, \dots, x_n$, $y_1, \dots, y_n$ и $n+1$ выходом $z_0, z_1, \dots, z_n$ такая, что для любых входных значений выполняется $(z_0 z_1 \dots z_n)_2 = (x_1 \dots x_n)_2 + (y_1 \dots y_n)_2$.

$$
\begin{array}{r}
+\begin{array}{r} x_1 \dots x_n \\ y_1 \dots y_n \end{array} \\
\hline
z_0 z_1 \dots z_n
\end{array}
$$

Схемный сумматор

Теорема

Существует сумматор порядка n, имеющий сложность (число функциональных элементов) не более $9n - 5$.

Доказательство

- Рассмотрим сложение «в столбик»:

$$\begin{array}{r} x_1 \ldots x_n \\ + \quad y_1 \ldots y_n \\ \hline z_0 z_1 \ldots z_n \end{array}$$

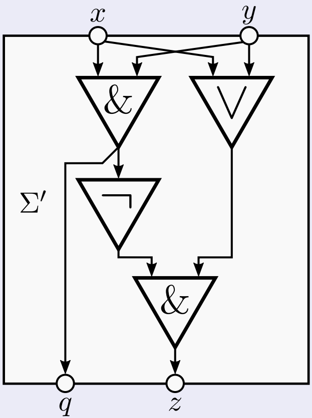
- Обозначим перенос в разряд k с помощью q_k:

$$\begin{array}{r} q_0 q_1 \ldots q_{n-1} \\ x_1 \ldots x_{n-1} x_n \\ + \quad y_1 \ldots y_{n-1} y_n \\ \hline z_0 z_1 \ldots z_{n-1} z_n \end{array}$$

Схемный сумматор

Доказательство (продолжение)

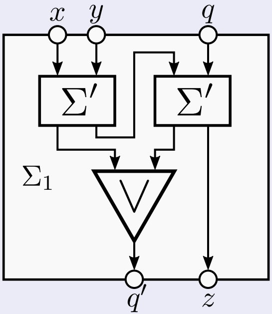
- Тогда мы имеем
  - $z_n = x_n \oplus y_n, \qquad q_{n-1} = x_n y_n;$
  - $z_i = x_i \oplus y_i \oplus q_i, \quad q_{i-1} = x_i y_i \vee x_i q_i \vee y_i q_i, \quad i = \overline{1, n-1};$
  - $z_0 = q_0.$
- Построим полусумматор: СФЭ Σ' с двумя входами x, y и двумя выходами q, z.
- Полусумматор имеет сложность 4 и вычисляет функции $q = xy$ и $z = x \oplus y$.
- Сложность 4 имеем благодаря соотношению $x \oplus y =$
  $= (\overline{x} \vee \overline{y}) \cdot (x \vee y) = \overline{xy} \cdot (x \vee y).$



Схемный сумматор

Доказательство (продолжение)

- Построим ячейку сумматора: СФЭ Σ_1 с тремя входами x, y, q и двумя выходами q', z.

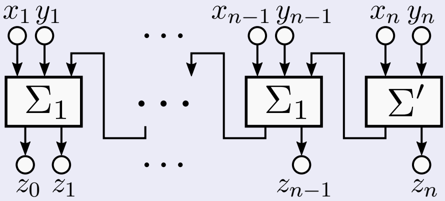


- Ячейка сумматора имеет сложность 9 и вычисляет две функции:
$q' = xy \vee (x \oplus y)q = xy \vee (x \vee y)q = xy \vee xq \vee yq$;
$z = x \oplus y \oplus q$.

Схемный сумматор

Доказательство (продолжение)

- Наконец, построим сумматор:



- Сумматор составлен из $n-1$ ячеек сумматора и одного полусумматора. Итоговая сложность $9(n-1)+4=9n-5$.

□

Лекция 14
Схемный вычитатель. Схемный умножитель. Теорема Карацубы

Схемный вычитатель

Определение

Вычитатель порядка n — это СФЭ с $2n$ входами $x_1, \dots, x_n$, $y_1, \dots, y_n$ и n выходами $z_1, \dots, z_n$ такая, что для любых входных значений таких, что $(x_1 \dots x_n)_2 \geqslant (y_1 \dots y_n)_2$,
выполняется $(z_1 \dots z_n)_2 = (x_1 \dots x_n)_2 - (y_1 \dots y_n)_2$.

$$\begin{array}{r} \_\ x_1 \dots x_n \\ y_1 \dots y_n \\ \hline z_1 \dots z_n \end{array}$$

- На входных значениях с $(x_1 \dots x_n)_2 < (y_1 \dots y_n)_2$ вычитатель может выдавать произвольные значения на выходе.

Схемный вычитатель

Теорема

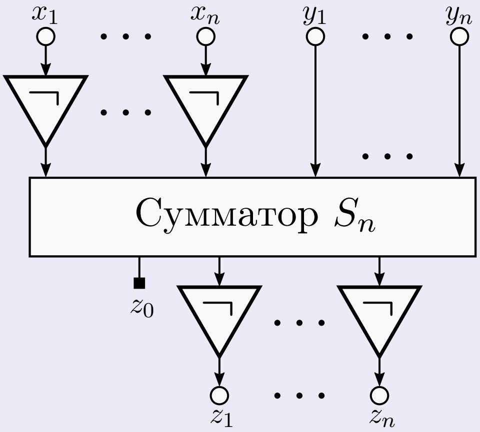
Существует вычитатель порядка n, имеющий сложность (число функциональных элементов) не более $11n - 5$.

Доказательство

- Обозначим $X = (x_1 \ldots x_n)_2$ и $Y = (y_1 \ldots y_n)_2$.
- $X - Y = -(-X + Y) = (2^n - 1) - ((2^n - 1 - X) + Y)$.
- $2^n - 1 = (\underbrace{1 \ldots 1}_{n})_2$. Поэтому $2^n - 1 - (x_1 \ldots x_n)_2 = (\overline{x}_1 \ldots \overline{x}_n)_2$.
- Тогда, чтобы реализовать вычитатель, достаточно у сумматора порядка n инвертировать значения на входах $x_1, \ldots, x_n$ и на выходах $z_1, \ldots, z_n$.
- Поскольку $X \geqslant Y$, для записи разности достаточно n битов, и на выходе z_0 у сумматора всегда будет 0. Не используем этот выход.

Схемный вычитатель

Доказательство (продолжение)



Схемный вычитатель

Доказательство (продолжение)

- Сложность сумматора $9n - 5$, дополнительно добавлено $2n$ отрицаний. Итоговая сложность $11n - 5$.

$\square$

- Если $X < Y$, то за счёт отбрасывания z_0 построенный вычитатель будет выдавать остаток от деления $X - Y$ на 2^n.
- Вычитатели в компьютерах обычно обрабатывают случай $X < Y$ таким же образом.
- Сумматоры в компьютерах обычно не используют выход z_0.
- Такой принцип работы позволяет сумматору и вычитателю автоматически поддерживать работу с отрицательными целыми числами в дополнительном коде.

Схемный умножитель

Определение

Умножитель порядка n — это СФЭ с $2n$ входами $x_1, \ldots, x_n$, $y_1, \ldots, y_n$ и $2n$ выходами $z_1, \ldots, z_{2n}$ такая, что для любых входных значений выполняется $(z_1 \ldots z_{2n})_2 = (x_1 \ldots x_n)_2 \cdot (y_1 \ldots y_n)_2$.

$$\begin{array}{r} \times \begin{array}{r} x_1 \;\; \ldots \; x_{n-1} \; x_n \\ y_1 \;\; \ldots \; y_{n-1} \; y_n \end{array} \\ \hline z_1 z_2 \ldots z_n z_{n+1} \ldots z_{2n-1} z_{2n} \end{array}$$

- Если $0 \leqslant X < 2^n$ и $0 \leqslant Y < 2^n$, то $0 \leqslant XY < 2^{2n}$. Поэтому умножитель имеет $2n$ выходов.

Схемный умножитель

Умножение «в столбик»

- Обозначим $u_{ij} = y_i \cdot x_j \in \{0,1\}$.

$$
\begin{array}{ccccccccc}
\times & & & & & x_1 & \ldots & x_{n-1} & x_n \\
 & & & & & y_1 & \ldots & y_{n-1} & y_n \\
\hline
 & & & & & u_{11} & \ldots & u_{1(n-1)} & u_{1n} \\
 & & & & u_{21} & u_{22} & \ldots & u_{2n} & 0 \\
+ & & & \iddots & \iddots & \iddots & \iddots & \iddots & \vdots \\
 & u_{n1} & u_{n2} & \ldots & u_{nn} & 0 & & \ldots & 0 \\
\hline
 & z_1 & z_2 & \ldots & z_n & z_{n+1} & \ldots & z_{2n-1} & z_{2n}
\end{array}
$$

- Вычисление всех $u_{i,j}$, $i,j = \overline{1,n}$, требует n^2 элементов $\&$. Реализация нуля требует двух элементов: $0 = \overline{x}_1 \cdot x_1$.
- Сложение n чисел длины не более $2n$ можно произвести $n-1$ сумматорами S_{2n}, каждый имеет сложность $18n-5$.
- Общая сложность $n^2 + 2 + (n-1)(18n-5) = 19n^2 - 23n + 7$.

Теорема Карацубы

O-символика

- Пусть $f(n)$ и $g(n)$ — функции натурального аргумента. Говорят, что $f(n) = \underline{O}(g(n))$, если существует константа $C > 0$ такая, что $f(n) \leqslant C \cdot g(n)$.
- Например:
  - Сложность умножения в столбик $19n^2 - 23n + 7 = \underline{O}(n^2)$.
  - $n^{1{,}5} = \underline{O}(n^{1{,}6})$.

Теорема (А. А. Карацуба)

Существует умножитель порядка n сложности $\underline{O}(n^{\log_2 3})$.

- Значение $\log_2 3$ немного меньше, чем $1{,}6$. Поэтому сложность умножителя Карацубы можно оценить как $\underline{O}(n^{1{,}6})$.
- Эта сложность заметно меньше, чем сложность умножения в столбик $\underline{O}(n^2)$.

Теорема Карацубы

- Доказательство теоремы Карацубы будет использовать несколько лемм, которые мы сформулируем и докажем ниже.

Обозначение

$M(n)$ — минимальная сложность умножителя порядка n.

Лемма 1

Существует константа $c_1 > 0$ такая, что $M(n+1) \leqslant M(n) + c_1 n$.

Доказательство

- Пусть $X = (x_0 \underbrace{x_1 \ldots x_n}_{X_1})_2 = x_0 \cdot 2^n + X_1$.

  и $\quad Y = (y_0 \underbrace{y_1 \ldots y_n}_{Y_1})_2 = y_0 \cdot 2^n + Y_1$.
- Т.е. $X_1 = (x_1 \ldots x_n)_2,\ Y_1 = (y_1 \ldots y_n)_2$.

Теорема Карацубы

Доказательство леммы 1 (продолжение)

- Тогда

$$X \cdot Y = (x_0 \cdot 2^n + X_1) \cdot (y_0 \cdot 2^n + Y_1) = \\ = x_0 y_0 \cdot 2^{2n} + (x_0 Y_1 + y_0 X_1) \cdot 2^n + X_1 Y_1.$$

- Для вычисления этого значения с помощью СФЭ требуется:
  - Умножитель порядка n для вычисления $X_1 Y_1$;
  - $2n$ элементов $\&$ для вычисления $x_0 Y_1$ и $y_0 X_1$, ещё 1 для $x_0 y_0$;
  - Сумматор порядка n для вычисления $x_0 Y_1 + y_0 X_1$;
  - Два сумматора порядка $2n + 2$ для вычисления внешних сложений;
  - Умножение на 2^l (битовый сдвиг) делается через подачу входа в старшие биты результата и нулей в младшие l бит (сложность 0);
  - Достаточно реализовать константу 0 в схеме один раз: $0 = x_1 \overline{x}_1$ (сложность 2). Используем 0 для заполнения младших битов при сдвигах и старших битов при подаче в сумматоры коротких чисел.

Теорема Карацубы

Доказательство леммы 1 (продолжение)

- Напомним, что сложность сумматора порядка n не превосходит $9n - 5$.
- Тогда существует константа $c_1 > 0$ такая, что суммарная сложность всей схемы, кроме схемы, кроме блока $X_1 Y_1$, не превосходит $c_1 n$.
- Блок $X_1 Y_1$ можно реализовать со сложностью $M(n)$.
- Получаем $M(n+1) \leqslant M(n) + c_1 n$.

$\square$

Теорема Карацубы

Лемма 2 (основная)

Существует константа $c_2 > 0$ *такая, что* $M(2n) \leqslant 3M(n) + c_2 n$.

Доказательство

- Пусть $X = (\underbrace{x_1 \dots x_n}_{X_1} \underbrace{x_{n+1} \dots x_{2n}}_{X_2})_2 = X_1 \cdot 2^n + X_2$.

  и $\quad Y = (\underbrace{y_1 \dots y_n}_{Y_1} \underbrace{y_{n+1} \dots y_{2n}}_{Y_2})_2 = Y_1 \cdot 2^n + Y_2$.

- Т.е. $X_1 = (x_1 \dots x_n)_2,\ X_2 = (x_{n+1} \dots x_{2n})_2,$

  $Y_1 = (y_1 \dots y_n)_2,\ Y_2 = (y_{n+1} \dots y_{2n})_2.$

Теорема Карацубы

Доказательство основной леммы 2 (продолжение)

- Тогда

$$
\begin{aligned}
X \cdot Y = (X_1 \cdot 2^n + X_2) \cdot (Y_1 \cdot 2^n + Y_2) = \\
= X_1 Y_1 \cdot 2^{2n} + (X_1 Y_2 + X_2 Y_1) \cdot 2^n + X_2 Y_2 = \\
= X_1 Y_1 \cdot 2^{2n} + [(X_1 + X_2) \cdot (Y_1 + Y_2) - X_1 Y_1 - X_2 Y_2] \cdot 2^n + X_2 Y_2.
\end{aligned}
$$

- Для вычисления этого значения с помощью СФЭ требуется:
  - Умножитель порядка $n+1$ для вычисления $(X_1 + X_2)(Y_1 + Y_2)$;
  - Два умножителя порядка n для вычисления $X_1 Y_1$ и $X_2 Y_2$ (их выходы используются по 2 раза);
  - Два сумматора порядка n для вычисления $X_1 + X_2$ и $Y_1 + Y_2$;
  - Два вычитателя порядка $2n+2$ и два сумматора порядка $4n$;
  - 0 элементов для реализации битовых сдвигов (умножение на 2^l), 2 элемента для реализации константы 0.

Теорема Карацубы

Доказательство основной леммы 2 (продолжение)

- Напомним, что сложность сумматора порядка n не превосходит $9n - 5$, а вычитателя $11n - 5$.
- Поэтому существует $c > 0$ такое, что суммарная сложность сумматоров и вычитателей (и константы 0) не превосходит cn.
- Тогда, с учётом леммы 1,

$$M(2n) \leqslant M(n+1) + 2M(n) + cn \leqslant 3M(n) + (c_1 + c)n.$$

- Выбирая $c_2 = c_1 + c$, получаем $M(2n) \leqslant 3M(n) + c_2 n$.

$\square$

Теорема Карацубы

Лемма 3

Существует константа $c_3 > 0$ такая, что $M(2^k) \leqslant c_3 \cdot 3^k$ при $k \in \mathbb{N}$.

- Неравенство в лемме эквивалентно $M(n) \leqslant c_3 \cdot n^{\log_2 3}$ при $n = 2^k$.

Доказательство

- Введём функцию $g(k) = \frac{M(2^k)}{3^k}$.
- В силу леммы 2 имеем

$$g(k) = \frac{M(2^k)}{3^k} = \frac{M(2 \cdot 2^{k-1})}{3^k} \leqslant \frac{3M(2^{k-1}) + c_2 2^{k-1}}{3^k} =$$

$$= \frac{M(2^{k-1})}{3^{k-1}} + \frac{c_2}{3}\left(\frac{2}{3}\right)^{k-1} = g(k-1) + \frac{c_2}{3}\left(\frac{2}{3}\right)^{k-1}.$$

Теорема Карацубы

Доказательство леммы 3 (продолжение)

- Т.е. $g(k) \leqslant g(k-1) + \frac{c_2}{3}\left(\frac{2}{3}\right)^{k-1}$.
- Тогда $g(k-1) \leqslant g(k-2) + \frac{c_2}{3}\left(\frac{2}{3}\right)^{k-2}$ и т.д.
- Получаем, с учётом $\sum\limits_{i=0}^{\infty} q^i = \frac{1}{1-q}$ при $q < 1$,

$$g(k) \leqslant g(k-1) + \frac{c_2}{3}\left(\frac{2}{3}\right)^{k-1}$$

$$\leqslant g(k-2) + \frac{c_2}{3}\left(\frac{2}{3}\right)^{k-1} + \frac{c_2}{3}\left(\frac{2}{3}\right)^{k-2} \leqslant \ldots$$

$$\ldots \leqslant g(0) + \frac{c_2}{3}\left(\left(\frac{2}{3}\right)^{k-1} + \ldots + \frac{2}{3} + 1\right) \leqslant$$

$$\leqslant g(0) + \frac{c_2}{3}\sum_{i=0}^{\infty}\left(\frac{2}{3}\right)^{i} = g(0) + c_2.$$

Теорема Карацубы

Доказательство леммы 3 (продолжение)

- Для реализации 1-разрядного умножителя нужно использовать один элемент $\&$. Поэтому $M(1) = 1$ и $g(0) = \frac{M(2^0)}{3^0} = 1$.
- Выбирая $c_3 = c_2 + 1$, получим $g(k) \leqslant c_3$, т.е. $M(2^k) \leqslant c_3 \cdot 3^k$.

$\square$

Теорема Карацубы

Теорема (А. А. Карацуба)

Существует умножитель порядка n *сложности* $\underline{O}(n^{\log_2 3})$*.*

Доказательство

- При $n \geqslant 2$ выберем натуральное число k такое, что $2^{k-1} < n \leqslant 2^k$.
- Числа длины n можно умножать на умножителе порядка 2^k, если подавать в старшие разряды нули.
- Напомним, что константу 0 необходимо реализовать только один раз, и это требует 2 элементов: $0 = x_1 \overline{x}_1$.
- Получаем $M(n) \leqslant M(2^k) + 2$.
- Тогда в силу леммы 3 имеем

$$M(n) \leqslant M(2^k) + 2 \leqslant c_3 \cdot 3^k + 2 = 3c_3 \cdot 3^{k-1} + 2 =$$
$$= 3c_3(2^{k-1})^{\log_2 3} + 2 \leqslant 3c_3 n^{\log_2 3} + 2 \leqslant 3(c_3 + 1)n^{\log_2 3}.$$

□

Литература

1. Лекции В. Б. Алексеева: Плейлист на YouTube
2. Алексеев В. Б. Лекции по дискретной математике. — М.: ВМК МГУ, 2004.
   https://mk.cs.msu.ru/images/b/b5/Lectdm.doc
3. Алексеев В. Б. Лекции по дискретной математике. — М.: Инфра-М, 2012.
4. Яблонский С. В. Введение в дискретную математику. — М.: Наука, 1986. — 384 с.
5. Гаврилов Г. П., Сапоженко А. А. Задачи и упражнения по дискретной математике. — М.: Физматлит, 2005. — 416 с.