Семинары по Дискретной математике

Алексей Федосеев

Версия 1.0, июнь 2006 г.

Обо всех неточностях и дополнениях Вы можете сообщать автору на электронную почту: *aleksey@fedoseev.net*. Буду рад любым комментариям.

Содержание

1 Множества и отношения

Определение множества Множество — это совокупность элементов, которые объединены *отношением принадлежности*. Для любого элемента множества A можно заключить: $a \in A$.

Множества бывают конечными (из конечного числа элементов) и бесконечными. Конечные множества могут задаваться перечислением их элементов, например:

$$A = \{1, 2, 3, 4\}, B = \{\text{Иванов}, \text{Петров}, \text{Сидоров}\}$$

Бесконечные множества удобно задавать с помощью коллективизирующего условия: $A = \{x | P(x)\}$, где $P(x)$ принимает значение "истина" или "ложь". Например, $A = \{x | k \in \mathbb{N}, x = 2k\}$.

Множества считаются равными тогда и только тогда, когда они состоят из одних и тех же элементов:

$$A = B \Leftrightarrow (\forall x)(x \in A \Leftrightarrow x \in B)$$

Отношение включения — множество A включается в множество B, если все элементы A принадлежат B: $A \subseteq B \Leftrightarrow (\forall x)(x \in A \Rightarrow x \in B)$. Тогда $A = B \Leftrightarrow (A \subseteq B)\ \&\ (B \subseteq A)$.

Порядок элементов в множестве не играет роли, множество не содержит повторяющихся элементов: $\{1, 2, 3, 4\} = \{1, 2, 4, 3\} = \{1, 1, 2, 3, 4\}$. Не следует путать множества элементов и множества множеств: $\{1, 2, 3, 4\} \neq \{\{1, 2\}, \{3, 4\}\}$.

Пустое множество не содержит ни одного элемента $(\forall a)(a \notin A)$. При этом $\{\varnothing\} \neq \varnothing$.

Существует ли множество, содержащее само себя? Да, рассмотрим пример: $Z = \{X | \text{X — множество, содержащее не менее трёх элементов}\}$:

$$\left.\begin{array}{c} \{1, 2, 3\} \in Z \\ \{1, 2, 3, 4\} \in Z \\ \{1, 2, 3, 4, 5\} \in Z \end{array}\right\} \Rightarrow Z \in Z$$

Существует ли множество, не содержащее само себя? $Y = \{X | X \notin X\}$. Это парадокс Рассела.

Операции над множествами Существуют операции над множествами (получение новых множеств на основе существующих):

Объединение:	$A \cup B = \{x \mid x \in A \vee x \in B\}$
Пересечение:	$A \cap B = \{x \mid x \in A\ \&\ x \in B\}$
Разность:	$A \setminus B = \{x \mid x \in A\ \&\ x \notin B\}$
Симметрическая разность:	$A \triangle B = (A \cup B) \setminus (A \cap B)$

Если задано *универсальное множество* U, то может быть введена операция *дополнения*: $\overline{A} = U \setminus A$.

Указанные операции обладают рядом интересных свойств (см. литературу). Свойства оформляются в виде тождеств (всегда истинных равенств). Существует несколько способов доказательства теоретико-множественных тождеств:

1. метод двух включений;

2. метод характеристических функций;

3. метод эквивалентных преобразований.

Операции над множествами могут быть проиллюстрированы на *диаграммах Венна*. Например, пересечение множеств может быть представлено как тёмная область на рисунке 1.

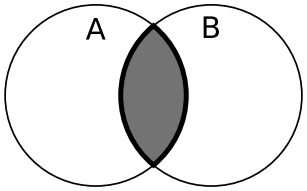

Рис. 1: Диаграмма Венна для пересечения множеств

Метод двух включений основывается на укзанном выше определении равенства множеств ($A = B \Leftrightarrow (A \subseteq B)\ \&\ (B \subseteq A)$). Необходимо доказать две леммы: если $x \in A \Rightarrow x \in B$ и $x \in B \Rightarrow x \in A$. Рассмотрим пример:

Доказать тождество $(A \cup B) \setminus (A \cap B) = (A \setminus B) \cup (B \setminus A)$. Докажем методом двух включений:

$\triangleleft$ Необходимость: $x \in (A \cup B) \setminus (A \cap B) \Rightarrow$
$(x \in (A \cup B)\ \&\ x \notin (A \cap B)) \Rightarrow \left\{ \begin{array}{l} x \in A \Rightarrow x \neq B \\ x \in B \Rightarrow x \neq A \end{array} \right\} \Rightarrow$
$(x \in A \setminus B) \vee (x \in B \setminus A) \Rightarrow x \in (A \setminus B) \cup (B \setminus A)$.
Достаточность: $x \in (x \in (A \cup B)\ \&\ x \notin (A \cap B)) \Rightarrow$
$(x \in A\ \&\ x \notin B) \vee (x \in B\ \&\ x \notin A) \Rightarrow$
$(x \in A \cup B)\ \&\ (x \notin A \cap B) \Rightarrow x \in (A \cup B) \setminus (A \cap B)$ $\triangleright$

Задания для самоподготовки. Доказать методом двух включений и проиллюстрировать результаты на диаграммах Венна:

- $(A \setminus B) \setminus C = A \setminus (B \setminus C)$;
- $(A \setminus B) \setminus C = (A \setminus C) \setminus (B \setminus C)$.

Метод характеристичских функций состоит в сопоставлении каждому множеству A функции $\chi_A(x) = \left\{ \begin{array}{l} 1, x \in A \\ 0, x \notin A \end{array} \right.$. Тогда функции, соответствующие операциям, будут равны:

$$\chi_{A \cap B} = \chi_A \chi_B;$$
$$\chi_{A \cup B} = \chi_A + \chi_B - \chi_A \chi_B;$$
$$\chi_{A \setminus B} = \chi_A (1 - \chi_B).$$

Рассмотрим пример доказательства приведённого выше тождества, $(A \cup B) \setminus (A \cap B) = (A \setminus B) \cup (B \setminus A)$:

Требуется показать, что $\chi_{(A\cup B)\setminus(A\cap B)} = \chi_{(A\setminus B)\cup(B\setminus A)}$.

$\triangleleft$ $\chi_{(A\cup B)\setminus(A\cap B)} = \chi_{A\cup B}(1 - \chi_{A\cap B}) = (\chi_A + \chi_B - \chi_A\chi_B)(1 - \chi_A\chi_B) =$
$= \chi_A - \chi_A\chi_B + \chi_B - \chi_A\chi_B - \chi_A\chi_B + \chi_A\chi_B = \underline{\chi_A + \chi_B - 2\chi_A\chi_B}$
$\chi_{(A\setminus B)\cup(B\setminus A)} = \chi_A(1 - \chi_B) + \chi_B(1 - \chi_A) - \chi_A(1 - \chi_B)\chi_B(1 - \chi_A) =$
$= \chi_A + \chi_B - 2\chi_A\chi_B - (\chi_A - \chi_A\chi_B)(\chi_B - \chi_A\chi_B) = \underline{\chi_A + \chi_B - 2\chi_A\chi_B}$ $\triangleright$

Задания для самоподготовки. Доказать методом характеристических функций и проиллюстрировать результаты на диаграммах Венна:

- $(A \triangle B) \triangle C = A \triangle (B \triangle C)$
- $A \cup (B \triangle C) = (A \cup B) \triangle (A \cup C)$

Метод эквивалентных преобразований заключается в последовательной подстановке известных тождеств в формулу для получения правой части доказываемого утверждения из левой или наоборот.

Задания для самоподготовки. Доказать методом эквивалентных преобразований и проиллюстрировать результаты на диаграммах Венна:

$$A \triangle B = (A \cup B) \cap (\overline{A} \cup \overline{B}).$$

Декартово произведение множеств Определим множество *упорядоченных пар* следующим образом:

$$A \times B = \{z | z = (x, y), (x \in A) \ \& \ (y \in B)\}$$

В общем случае операция декартова произведения *некоммутативна*: $A \times B \neq B \times A$. Также эта операция не обладает свойством *ассоциативности* $A \times (B \times C) \neq (A \times B) \times C \neq A \times B \times C$, действительно, если $a_i \in A, b_i \in B, c_i \in C$, то $(a_i, (b_i, c_i)) \neq ((a_i, b_i), c_i) \neq (a_i, b_i, c_i)$.

Графически декартово произведение можно изобразить в виде квадрата на плоскости (рисунок 2).

Умножение множества на себя называют декартовым квадратом (кубом и т.п.): $A^2 = A \times A$, $A^3 = A \times A \times A$, ... Декартов квадрат множества всех действительных чисел $\mathbb{R}^2$ образует декартову плоскость.

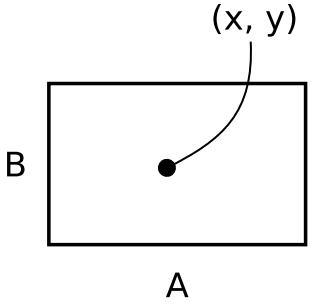

Рис. 2: Декартово произведение множеств

Задания для самоподготовки. Доказать методом двух включений и проиллюстрировать тождества графиком:

- $A \times (B \cup C) = (A \times B) \cup (A \times C)$;
- $A \times (B \setminus C) = (A \times B) \setminus (A \times C)$.

Показать, что тождество $\overline{A \times B} = \overline{A} \times \overline{B}$ в общем случае неверно. Записать верное тождество и доказать его методом двух включений.

Соответствия и операции над ними Пусть заданы два множества A и B, *соответствием* называют некоторое подмножество их декартова произведения: $\rho \subseteq A \times B$ (рисунок 3). Если для пары $(x, y) \in A \times B$ имеет место соответствие, то $(x, y) \in \rho$. Это также записывают в виде $x \rho y$.

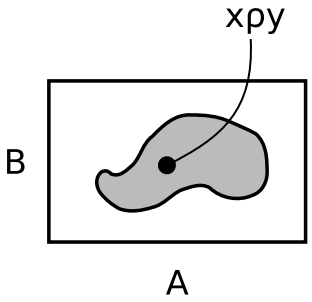

Рис. 3: Пример соответствия

Существует несколько способов задания соответствия: перечислением элементов (возможно для конечных множеств A и B), заданием предиката (например, $\rho = \{(x, y) | x \in A, y \in B, x < y + 5\}$), и в виде графа, часто соответствие иллюстрируют графиком (рисунок 4).

Для каждого соответствия рассматривают *область определения* $\operatorname{dom} \rho = \{x | x \in A \ \& \ (\exists y)((x, y) \in \rho)\}$ и *область значений* $\operatorname{rng} \rho = \{x | y \in B \ \& \ (\exists x)((x, y) \in \rho)\}$.

Вводится понятие *функциональности* соответствия. Соответствие функционально по первой компоненте тогда и только тогда, когда $\forall (x_1, y_1), (x_2, y_2) \in$

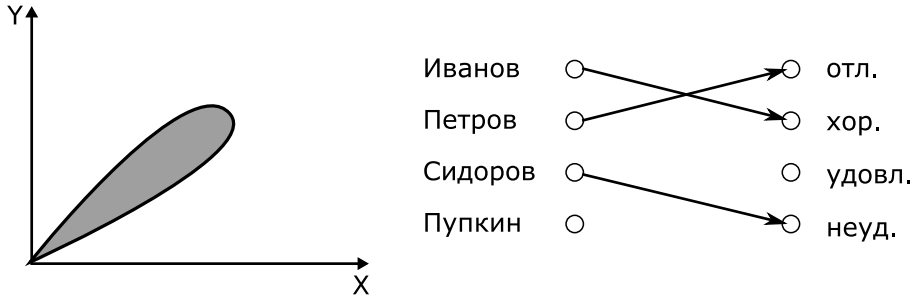

Рис. 4: Пример задания соответствия графически

$\rho : y_1 = y_2 \Rightarrow x_1 = x_2$, другими словами: для каждого y существует единственный x. Аналогично, соответствие функционально по второй компоненте тогда и только тогда, когда $\forall (x_1, y_1), (x_2, y_2) \in \rho : x_1 = x_2 \Rightarrow y_1 = y_2$, или для каждого x существует единственный y.

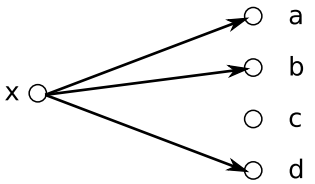

Рис. 5: Пример соответствия, не функционального по 2-ой компоненте

Отображение из A в B (функция) — это соответствие из A в B, которое всюду определено и функционально по второй компоненте.

Соответствие — это множество упорядоченных пар, поэтому все операции над множествами к ним применимы. Для соответствий вводятся две новые операции: *композиция* и *обратное соответствие*.

Композиция соответствий $\rho \subseteq A \times B$ и $\sigma \subseteq B \times C$ — новое соответствие $\tau \subseteq A \times C$, равное $\tau = \rho \circ \sigma = \{(x, y) | x \in A, y \in C, \exists z \in B : x \rho z \ \& \ z \sigma y\}$. Для конечных соответствий удобно пользоваться графовым представлений соответствий — тогда для получения композиции достаточно пройти по рёбрам графа.

Пример композиции соответствий представлен на рисунке 6 и иллюстрирует композицию соответствий оценок, полученных студентами, (ρ) и соответствия оценки получаемой стипендии. Результат композиции в этом случае — соответствие студентов и выплаты стипендии.

Под *степенью отношения* понимают его композицию с самим собой: $\rho^2 = \rho \circ \rho$.

Обратное соответствие образуется следующим образом: $\rho^{-1} \subseteq B \times A$, $\rho^{-1} = \{(x, y) | (y, x) \in \rho\}$. При этом $\operatorname{dom} \rho = \operatorname{rng} \rho^{-1}$ и $\operatorname{rng} \rho = \operatorname{dom} \rho^{-1}$.

Композиция и обратное соответствие обладают рядом свойств, которые могут быть доказаны методом двух включений. Рассмотрим свойство композиции $\rho \circ (\sigma \cup \tau) = \rho \circ \sigma \cup \rho \circ \tau$.

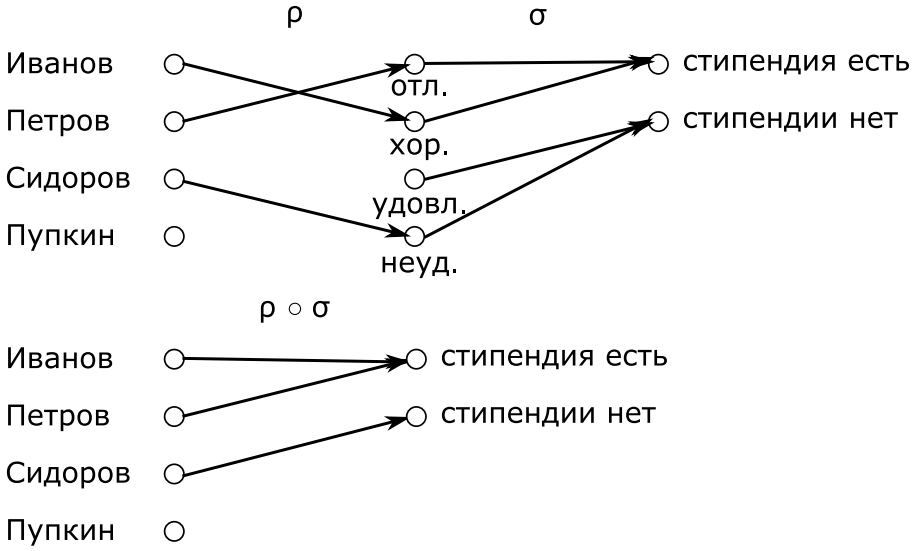

Рис. 6: Пример композиции соответствий

◁ Необходимость: $(x, y) \in \rho \circ (\sigma \cup \tau) \Rightarrow$
$(\exists z)((x, z) \in \rho \ \& \ (z, y) \in \sigma \cup \tau) \Rightarrow$
$(\exists z)((x, z) \in \rho \ \& \ ((z, y) \in \sigma \vee (z, y) \in \tau)) \Rightarrow$
$(\exists z)((x, z) \in \rho \ \& \ (z, y) \in \sigma) \vee (\exists z)((x, z) \in \rho \ \& \ (z, y) \in \tau) \Rightarrow$
$(x, y) \in \rho \circ \sigma \vee (x, y) \in \rho \circ \tau \Rightarrow (x, y) \in \rho \circ \sigma \cup \rho \circ \tau$
Достаточность доказывается аналогично ▷

Задания для самоподготовки. Доказать тождества для соответствий. Во втором примере привести контр пример, показывающий, что равенство не может иметь место.

- $(\rho \circ \sigma)^{-1} = \sigma^{-1} \circ \rho^{-1}$;
- $\rho \circ (\sigma \cap \tau) \subseteq \rho \circ \sigma \cap \rho \circ \tau$.

Задания для самоподготовки. Для соответствия $\rho \subseteq A \times B$: 1) построить график и граф; найти область определения и область значений; 2) найти ρ^{-1}, построить для него график и граф; 3) установить, является ли ρ отображением из A в B:

- $A = \{1, 2, 3, 4\}, B = \{p, q, r, s, t\}$, $\rho = \{(1, s), (1, t), (2, r), (2, s), (3, p), (3, q), (4, q), (4, t)\}$;
- $A = B = \{a, b, c, d, e\}, \rho = \{(b, b), (d, c), (c, e), (a, e), (e, a)\}$.

Свойства отображений Отображение называют *инъективным*, если оно функционально по первой компоненте.

На рисунке 7 показано неинъективное отображение $f(x) = x^2$ при $A = \mathbb{R}$, $B = \mathbb{R}^+$. Оно не является инъективным, так как значению $y = 1$ соответствуют два $x = 1$ и $x = -1$.

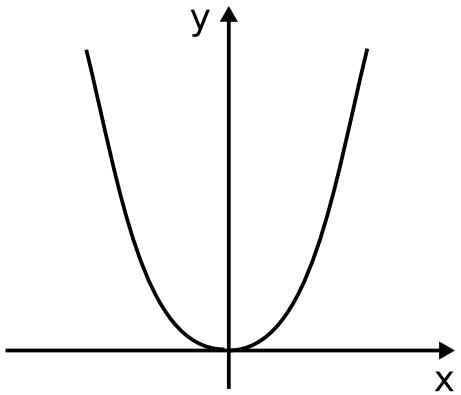

Рис. 7: Пример неинъективного отображения

Отображение $f : A \to B$ называют *сюръективным*, если область отображения совпадает с B.

На рисунке 8 показано несюръетивное отображение $f(x) = e^x$ при $A = \mathbb{R}$, $B = \mathbb{R}^+$. Оно не является сюръективным, так как область значений отображения равна $\operatorname{rng} f = (0, +\infty)$. Однако, оно инъективно, так как функионально по первой компоненте.

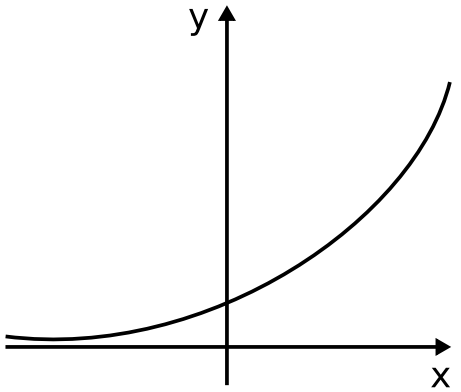

Рис. 8: Пример инъективного и несюръективного отображения

Если отображение инъективно и сюръективно, то оно называется *биективным*. Такое отображение задаёт *взаимно однозначное соответствие* между множествами A и B.

Пример биекции изображён на рисунке 9: если рассмотреть угол, под которым расположен луч, то он соответствует точке на действительной прямой и наоборот. Таким образом, задано биективное отображение интервала $(0, \pi)$ на $\mathbb{R}$.

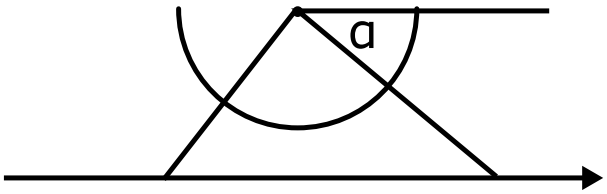

Рис. 9: Пример биективного отображения

Задания для самоподготовки. Для данных функций $f : A \to B$ проверить свойства инъективности, сюръективности и биективности.

- $A = M_{n\times n}(\mathbb{R})$(квадратные матрицы), $B = \mathbb{R}$, $f(x) = \det x$;
- $A = \mathbb{C} \setminus \{0\}$, $B = \mathbb{R}^+ \times [0, 2\pi)$, $f(z) = (|x|, \arg z)$;
- $A = C[a, b]$ (функции, непрерывные на отрезке $[a, b]$), $B = \mathbb{R}$, $f(x) = \int\limits_a^b x(t)\, dt$.

Бинарные отношения и их свойства *Бинарное отношение* (отношение) — соответствие множества самому себе: $\rho \subseteq A \times A$. Выделим специальное отношение, которое существует для любого множества A, называемое *диагональю*: $id_A = \{(x, x)|x \in A\}$.

Задания для самоподготовки. На множестве $\mathbb{Z}$ задано отношение $<$. Найти квадрат этого отношения. Как изменится ответ, если задать это отношение на $\mathbb{R}$.

Свойства бинарных отношений:

1. Отношение называют *рефлексивным*, если $(\forall x \in A)((x, x) \in \rho)$. Отношение рефлексивно тогда и только тогда, когда диагональ полностью включается в него $id_A \subseteq \rho$. В качестве примера рефлексивного отношения можно привести $\rho = \{(x, y)|x = y\}$.

2. Отношение называют *иррефлексивным*, если $(\forall x \in A)((x, x) \notin \rho$. Отношение иррефлексивно тогда и только тогда, когда диагональ полностью не принадлежит ему $id_A \cap \rho = \varnothing$. Если отношение не рефлексивно и не иррефлексивно, его называют *нерефлексивным*. Иррефлексивным отношением является $\rho = \{(x, y)|x \neq y\}$.

3. Отношение называют *симметричным*, если $(\forall x, y \in A)(x\rho y \Rightarrow y\rho x)$. При этом график отношения симметричен относительно диагонали, при этом отношение совпадает с обратным к нему $\rho = \rho^{-1}$. Примером такого отношения является равенство треугольников.

4. Отношение называют *антисимметричным*, если $(\forall x, y \in A)(x\rho y \ \& \ y\rho x \Rightarrow x = y)$. Если отношение не симметрично и не антисимметрично, его называют несимметричным. На графике отношения все симметричные относительно диагонали точки будут лежать на самой диагонали $\rho \cap \rho^{-1} \subseteq id_A$. Пример такого отношения — $x \leqslant y$.

5. Отношение называют *транзитивным*, если $(\forall x, y, z \in A)(x\rho y \ \& \ y\rho z \Rightarrow x\rho z)$. При этом квадрат отношения включается в него самого $\rho^2 \subseteq \rho$.

На рисунке 10 показаны примеры отношений с перечисленными свойствами.

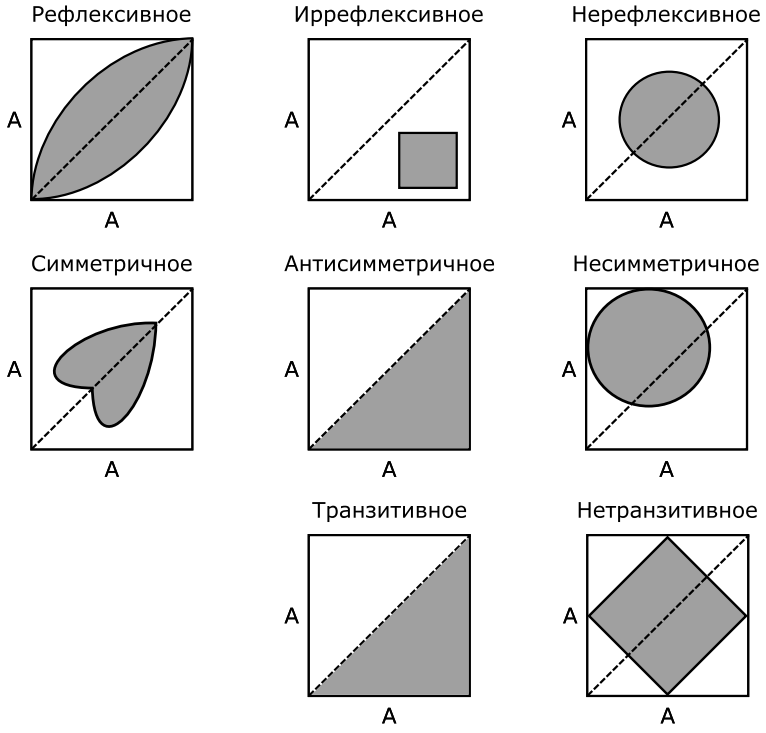

Рис. 10: Графики, иллюстрирующие свойства отношений

Задания для самоподготовки. Для заданного бинарного отношения $\rho \subseteq A^2$ 1) построить график; 2) найти ρ^{-1} и ρ^2 и построить их графики; 3) исследовать свойства Р, И, С, А, Т.

- $A = \{1, 2, 3, 4\}, \rho = \{(1, 1), (1, 3), \ (2, 1), (2, 4), \ (3, 2), (3, 3), (3, 4), (4, 1)\}$;
- $A = [0, 1], x\rho y \Leftrightarrow |x - y| \leqslant 1/3$;
- $A = [0, 1], x\rho y \Leftrightarrow y \geqslant x + 1/4, x \leqslant 1/2$.

Задания для самоподготовки. Для заданного бинарного отношения $\rho \subseteq A^2$ 1) найти ρ^{-1} и ρ^2; 2) исследовать свойства Р, И, С, А, Т.

- $A = M_{n\times n}(\mathbb{R}), P\rho Q \Leftrightarrow p_{ij} \leqslant q_{ij}, i, j \in \{1, \dots n\}$;
- $A = M_{n\times n}(\mathbb{R}), P\rho Q \Leftrightarrow \det P = \det Q$;
- $A = M_{n\times n}(\mathbb{R}), P\rho Q \Leftrightarrow \det P \leqslant \det Q$.

Рассмотрим пример бинарного отношения, которое изображено на рисунке 11. Исследуем свойства Р, И, С, А, Т для этого отношения.

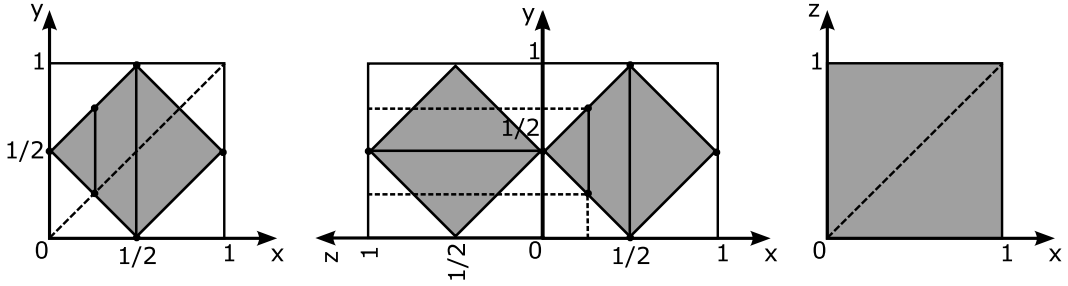

Рис. 11: Пример отношения

рефлексивность диагональ не включается полностью в это отношение, значит оно нерефлексивно;

симметричность график отношения симметричен относительно диагонали, значит это симметричное отношение;

транзитивность транзитивность будем проверять, учитывая тот факт, что для транзитивных отношений выполняется $\rho \circ \rho \subseteq \rho$.

Рассмотрим, какие значения принимает отношения при различных x. Для этого воспользуемся специальным обозначением *сечения*: $\rho(X) = \{y|(x, y) \in \rho, x \in X\} = \bigcup\limits_{x\in X} \rho(x)$. На рисунке 11 показаны значения, которые отношение принимает при различных x:

$$\rho\left(\frac{1}{2}\right) = [0, 1], \rho(0) = \rho(1) = \frac{1}{2}, \rho\left(\frac{1}{4}\right) = \left[\frac{1}{4}, \frac{3}{4}\right]$$

Отметим, что $\rho(x)$ для любого x содержит $\frac{1}{2}$, а значит:

$$\rho \circ \rho(0) = \rho \circ \rho(1) = \rho(\rho(0)) = \rho\left(\tfrac{1}{2}\right) = [0, 1],$$
$$\rho\left(\rho\left(\tfrac{1}{4}\right)\right) = \rho\left(\left[\tfrac{1}{4}, \tfrac{3}{4}\right]\right) = [0, 1],$$
$$\dots$$

Получается, что $\rho \circ \rho = [0, 1]^2$, а значит $\rho \circ \rho \not\subseteq \rho$, т.е. это отношение не транзитивно.

Таким образом, это отношение имеет следующие свойства:

Р	И	С	А	Т
-	-	+	-	-

Отношения порядка Отношения, обладающие свойствами *рефлексивности*, *антисимметричности* и *транзитивности* называют *отношениями порядка*.

Исследуем свойства следующего отношения: $\rho \subseteq \mathbb{N}^2, x\rho y \Leftrightarrow x|y$ (x делится нацело на y).

рефлексивность $(\forall x)(x = x \Rightarrow x|x)$, рефлексивно;

симметричность Симметричность в общем случае не выполняется (например, для чисел 3 и 1). $x|y \ \ \& \ \ y|x \Rightarrow x = y$, значит антисимметрично;

транзитивность

$$\left.\begin{array}{r} x|y \Leftrightarrow x = ky, k \in \mathbb{N} \\ y|z \Leftrightarrow y = mz, m \in \mathbb{N} \end{array}\right\} \Rightarrow x = (km)z,$$

т.е. выпоняется $x|z$, значит отношение транзитивно.

Обобщим свойства отношения в таблице:

Р	И	С	А	Т
+	-	-	+	+

В упорядоченных множествах два элемента называются *сравнимыми*, если для них можно определить, какой больше, а какой — меньше. В случае обычного числового порядка ($\leqslant$) все элементы оказываются сравнимыми, а в рассмотренном выше отношении порядка, например, элементы 2 и 3 не могут быть сравнены. Отношения порядка, при которых все элементы множества сравнимы, называются *линейными*.

Для отношений порядка на конечных множествах удобно использовать диаграмме Хассе: элементы множества представляются точками на плоскости, два элемента соединяются линией в том случае, если элементы сравнимы и между ними не может быть помещён ещё один элемент. При этом, большие элементы рисуют выше, а меньшие — ниже.

На рисунке 12 показана диаграмма Хассе для рассмотренного выше отношения порядка на различных множествах целых чисел. На диаграмме можно легко увидеть несравнимые элементы и линейный порядок (все элементы выстраиваются в линию).

Рассмотрим другое отношение порядка: отношения включения теории множеств $\subseteq$ на множестве 2^A всех подмножеств множества A. Внимательно рассмотрев его свойства, можно увидеть, что это отношение также рефлексивно, антисимметрично и транзитивно.

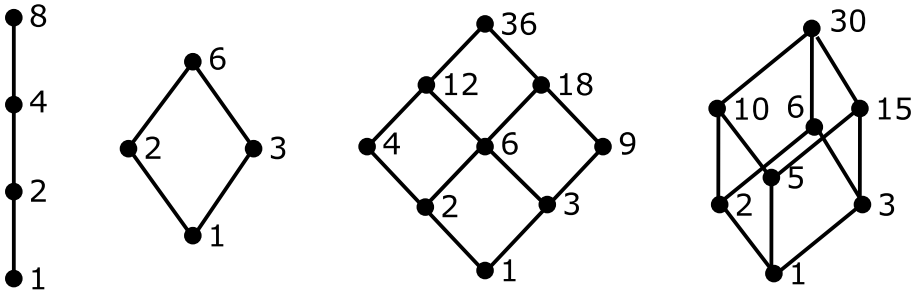

Рис. 12: Диаграммы Хассе для отношения порядка делимости

На рисунке 13 показан пример такого отношения на множестве, образованном тремя множествами A, B и C и их объединениями. Диаграмма Хассе в данном случае имеет вид кубика, а отношение порядка не является линейным — например, A и $B \cup C$ не могут быть сравнимы, так как нельзя сказать, чтобы одно из этих множеств включалось в другое.

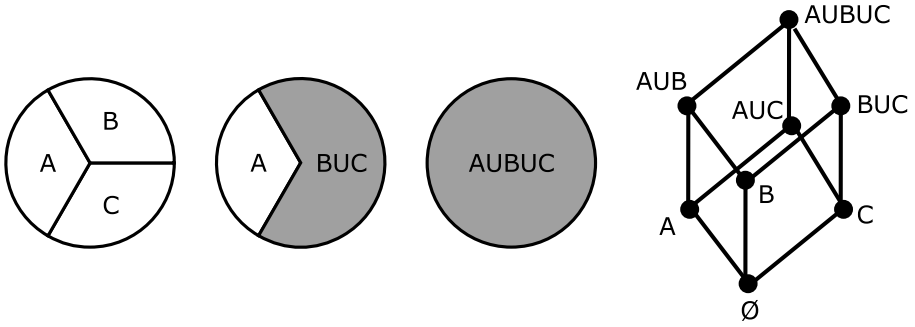

Рис. 13: Диаграмма Хассе для отношения порядка включения

Задания для самоподготовки. Рассмотрим множество слов русского алфавита. Каждое слово состоит из упорядоченного набора букв $x = x_1 x_2 \dots x_n$. Для букв будем использовать стандартный порядок их следования в алфавите. Рассмотрим следующие два отношения:

- $x \rho y \Leftrightarrow x_1 < y_1 \vee (x_1 = y_1 \ \& \ x_2 < y_2) \vee (x_1 = y_1 \ \& \ x_2 = y_2 \ \& \ x_3 < y_3) \vee \dots$;
- $x \rho y \Leftrightarrow x_1 \leqslant y_1 \ \& \ x_2 \leqslant y_2 \ \& \ x_3 \leqslant y_3 \ \& \ \dots$.

Докажите, что они являются отношениями порядка. Какое из них является линейным? Изобразите диаграммы Хесса для следующего набора слов: “бутылка”, “банан”, “бисквит”, “бивень”, “банджо”.

Отношения эквивалентности Отношения, обладающие свойствами *рефлексивности*, *симметричности* и *транзитивности* называют *отношениями эквивалентности*.

Примерами отношения эквивалентности является простое числовое равенство или отношение подобия для треугольников.

Исследуем свойства отношения: $A = M_{n\times n}(\mathbb{R}), \rho \subseteq A^2, P\rho Q \Leftrightarrow \det P = \det Q$, т.е. на множестве квадратных матриц из действительных чисел отношение выполняется, если определители двух матриц равны.

Свойства этого отношения вытекают из свойств числового равенства:

рефлексивность $(\forall X)(\det X = \det X)$, рефлексивно;

симметричность $(\forall X, Y)(\det X = \det Y)$, симметрично;

транзитивность $(\forall X, Y, Z)(\det X = \det Y \ \ \& \ \ \det Y = \det Z) \Rightarrow \det X = \det Z$, транзитивно.

Отличительным свойством отношений эквивалентности является существование *классов эквивалентности*. Классом эквивалентности элемента x по отношению $\rho \subseteq A^2$ называют множество всех элементов A, эквивалентных по данному отношению x:

$$[x]_\rho = \{y | y \in A, x\rho y\}$$

Например, в рассмотренном выше отношении (ограничимся сейчас $n = 2$) можно выделить множество всех матриц, определитель которых равен 1:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \rho \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \rho \begin{pmatrix} 0 & -1 \\ 1 & 1 \end{pmatrix}, \dots$$

Таким образом, для каждого действительного числа x можно построить класс эквивалентности матриц, определитель которых равен x:

$$\left[\begin{pmatrix} x & 0 \\ 0 & 1 \end{pmatrix}\right]_\rho = \left\{\begin{pmatrix} x & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 2x & 0 \\ 0 & 0.5 \end{pmatrix}, \dots\right\}$$

Можно доказать, что классы эквивалентности образуют *разбиение* начального множества A. Разбиением называется такой набор множеств A_1, A_2, ..., который удовлетворяет условиям: $A_1 \cup A_2 \cup \dots \cup A_n \cup \dots = A$ и $A_1 \cap A_2 \cap \dots \cap A_n \cap \dots = \varnothing$, т.е. множества разбиения не пересекаются и при объединении дают начальное множество.

Множество A можно представить в виде объединения классов эквивалентности некоторых его элементов: $A = [x]_\rho \cup [y]_\rho \cup \dots$. В нашем примере множество всех матриц можно разбить на показанные выше классы эквивалентности, эти классы не будет пересекаться, так как в каждом из них определители матриц отличаются.

Множество всех классов эквивалентности, задающих разбиение множества A, называют *фактор-множеством*, а процесс его построения *факторизацией*. Обозначается фактор-множество следующим образом:

$$A/_\rho = \left\{ \left[\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \right]_\rho, \left[\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right]_\rho, \left[\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \right]_\rho, \dots \right\} =$$
$$= \left\{ \left[\begin{pmatrix} x & 0 \\ 0 & 1 \end{pmatrix} \right]_\rho \middle| \, x \in \mathbb{R} \right\}$$

Рассмотрим ещё одно отношение: на множестве целых чисел при заданном $k \in \mathbb{N}$, $\rho \subseteq \mathbb{Z}^2, x\rho y \Leftrightarrow x \equiv_k y$, что означает — x и y равны по модулю k или имеют один остаток при делении на k. Выясним свойства отношения:

рефлексивность $(\forall x)(x = x) \Rightarrow x \equiv_k x$, рефлексивно;

симметричность $(\forall x, y)(x \equiv_k y \Rightarrow y \equiv_k k)$, симметрично;

транзитивность

$$\left.\begin{array}{r} x \equiv_k y \Leftrightarrow x = nk + r,\ y = mk + r \\ y \equiv_k z \Leftrightarrow y = mk + r,\ z = lk + r \end{array}\right\} \Rightarrow x = nk + r,\ z = lk + r,$$

транзитивно

Обобщим свойства отношения в таблице. Видно, что это отношение эквивалентности.

Р	И	С	А	Т
+	-	+	-	+

Построим существующие классы эквивалентности:

$[0]_{\equiv_k} = \{0, \pm k, \pm 2k, \pm 3k, \dots\}$,
$[1]_{\equiv_k} = \{1, \pm k + 1, \pm 2k + 1, \pm 3k + 1, \dots\}$,
$\dots$,
$[k-1]_{\equiv_k} = \{k - 1, \pm k + k - 1, \pm 2k + k - 1, \pm 3k + k - 1, \dots\}$

Всего k классов эквивалентности, которые задают разбиение всего множества целых чисел. Для $k = 2$ имеем следующие два класса эквивалентности:

$$[0]_{\equiv_2} = \{0, \pm 2, \pm 4, \dots\},$$
$$[1]_{\equiv_2} = \{\pm 1, \pm 3, \pm 5, \dots\},$$

а фактор-множество равно:

$$\mathbb{Z}/_{\equiv_2} = \{[0]_{\equiv_2}, [1]_{\equiv_2}\}.$$

Задания для самоподготовки. Доказать, что отношение $\rho \subseteq A^2$ является эквивалентностью. Построить фактор-множество $A/_\rho$. Чем однозначно определяется каждый класс эквивалентности?

- $A = V_3$ (множество векторов в пространстве), $x \rho y \Leftrightarrow (x - y, \vec{n}) = 0$, где $\vec{n} \neq 0$ — фиксированный вектор;
- $A = \mathbb{N}^2, (a, b)\rho(c, d) \Leftrightarrow ad = bc$.

Мощности множеств В основе понятия *мощности* лежит определение *равномощности* множеств. Множество A равномощно множеству B, если существует биекция $f : A \to B$. Биекция задаёт взаимо-однозначное соответствие между множествами, т.е. каждому элементу множества A может быть поставлен один элемент множества B и наоборот. То есть, эти множества как бы (в бесконечном случае так говорить не очень корректно) состоят из одного числа элементов.

Равномощность множеств обычно обозначается так: $A \sim B$. Не трудно показать, что это отношение является *эквивалентностью*. А значит, можно выделить классы эквивалентности равномощных множеств, которые совпадают для всех равномощных множеств: $|A| = |B|$, такой класс эквивалентности $|A|$ называют *мощностью* множества A.

В случае конечных множеств, под мощностью можно понимать натуральное число — число элементов множества (для мощностей с одинаковым числом элементов биекция строится тривиально). Однако, в бесконечном случае без построения биекции в доказательстве не обойтись.

Любое множество, равномощное множеству натуральных чисел $\mathbb{N}$, называют *счетным*. Мощность счетных множеств обозначается символом $\aleph_0$ ("алеф-ноль"). Любое множество, равномощное множеству натуральных чисел, можно "пронумеровать", записав в виде последовательности $a_1, a_2, \ldots a_n \ldots$.

Весьма оригинально, что множество всех положительных чётных чисел $\mathbb{N}_2$, включающееся в множество натуральных чисел $\mathbb{N}$, также счётно (равномощно $\mathbb{N}$). Это доказывается построением элементарной биекции: $f(n) = 2n, n \in \mathbb{N}$. Другой забавный пример — множество $\mathbb{N} \times \mathbb{N}$ (точки на плоскости с целыми положительными координатами) также счётно; биекция здесь чуть более сложна — она задаётся нумерацией всех точек плоскости из угла $(1, 1)$ по расширяющимся диагоналям (см. рисунок 14). Очевидно, что $\mathbb{Z}$ и $\mathbb{Q}$ также счётны.

Для счетных множеств существует ряд интересных теорем:

- любое бесконечное множество содержит счётное подмножество;
- любое подмножество счётного множества конечно или счётно;
- объединение не более чем счётного семейства счётных множеств счётно;

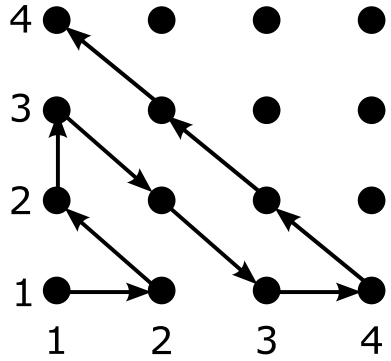

Рис. 14: Нумерация множества $\mathbb{N} \times \mathbb{N}$

- объединение конечного и счётного множества счётно.

Однако, множество всех действительных чисел не является счётным (это довольно элементарно доказал Кантор). Кроме того, равномощными являются следующие множества: всех двоичных последовательностей, всех подмножеств множества $\mathbb{N}$ и всех действительных чисел:

$$\{0,1\}^\omega \sim 2^{\mathbb{N}} \sim (0,1) \sim \mathbb{R}.$$

Мощность этих множеств называют *мощностью континуума*.

На самом деле, для любого множества A мы можем легко получить множество с большей мощностью. Для этого достаточно рассмотреть множество 2^A.

2 Алгебраические структуры

Алгебраическая структура (алгебра) — множество с операциями на нём. Обозначается следующим образом: $\mathcal{A}(A, \Omega)$, где Ω — набор операций, называемый *сигнатурой*.

Операция — произвольное отображение $\omega : A^n \to A$. Выделяют унарные операции ($n = 1$), бинарные операции ($n = 2$), тернарные операции ($n = 3$). Также иногда представляют особый интерес некоторые элементы множества A, при этом они могу добавляться в сигнатуру, превращаясь в так называемые *нулярные* операции.

Чаще всего встречаются бинарные операции, они могу обладать следующими свойствами:

коммутативность $(\forall x, y \in A)(x * y = y * x)$;

ассоциативность $(\forall x, y, z \in A)(x * (y * z) = (x * y) * z)$;

идемпотентность $(\forall x \in A)(x * x = x)$.

Примеры алгебр:

- $\mathcal{N}(\mathbb{N}, \{+, \cdot\})$ — сложение и умножение на множестве натуральных чисел;
- $\mathcal{B}_A(2^A, \{\cup, \cap, \setminus, \neg, \varnothing, A\})$ — операции над множеством всех подмножеств множества A, два элемента из 2^A: $\varnothing$ и A были специально выделены;
- $\mathcal{V}(V_3, \{+, \cdot\alpha(\alpha \in \mathbb{R})\})$ — множество векторов в пространстве с операциями сложения и умножения на число — эта агебра интересна тем, что её сигнатура бесконечна.

Примеры не-алгебр:

- $\mathcal{R}(\mathbb{R}, \{+, \cdot, /\})$ — не алгебра, так как деление определено не для всех пар чисел из $\mathbb{R}$;
- $\mathcal{M}(M \to M, \{\circ, ^{-1}\})$ — не алгебра, так как не для каждого отображения $M \to M$ существует обратное отображение.

Группы Любая алгебра с одной бинарной операцией $\mathcal{A}(A, *)$ называется *группоидом*. Примеры группоида: $(\mathbb{N}, +)$ — сложение натуральных чисел или $(V_3, \times)$ — множество векторов в пространстве с операцией векторного произведения.

Группоид, операция которого обладает *ассоциативностью*, называется *полугруппой*. Примером полугруппы является алгебра бинарных отношений на множестве M с операцией композиции $(2^{M \times M}, \circ)$ или та же алгебра $(\mathbb{N}, +)$. А вот алгебра векторов с векторным произведением полугруппой уже не является, так как векторное произведение неассоциативно.

Если $\mathcal{A}(A, *)$ — полугруппа и существует такое $\varepsilon \in A$, что $(\forall x \in A)(x * \varepsilon = \varepsilon * x = x)$, то такая полугруппа называется *моноидом*, а элемент ε — нейтральным элементом. Обычно, нейтральный элемент записывается в сигнатуре. Можно доказать, что если полугруппа имеет нейтральный элемент, то он является единственным. Примером моноида является множество натуральных чисел с операцией умножения $(\mathbb{N}, \cdot, 1)$. А полугруппа натуральных чисел со сложением должна быть дополнена нулём, чтобы стать моноидом: $(\mathbb{N}_0, +, 0)$. Также моноид образуют теоретико-множественные операции объединения и пересечения: $(2^A, \cup, \varnothing)$ и $(2^A, \cap, A)$.

Пусть $\mathcal{M}(M, *, \varepsilon)$ — произвольный моноид, тогда $x' \in M$ называется обратным элементом к $x \in M$, если $x * x' = x' * x = \varepsilon$. Моноид называется *группой*, если каждый элемент в нём имеет обратный. Можно доказать, что в группе для любого элемента существует единственный обратный элемент, или $(x')' = x$.

Примером группы является множество целых чисел с операцией сложения: $(\mathbb{Z}, +, 0)$, при этом обратным элементом к любому числу является его отрицание. Можно увидеть, откуда взялось название *полугруппы* — если мы возьмём две полугруппы натуральных и отрицательных

чисел $(\mathbb{N}, +)$ и $(\mathbb{N}^-, +)$ и объединим их с добавлением нейтрального элемента 0, то получим полную группу.

Также группой является множество всех невырожденных матриц $(\mathcal{M}^+, \cdot, E, ^{-1})$ с операцией умножения. В этом случае обратная матрица является обратным элементом к любой матрице (она существует, так как матрицы невырождены).

В группе можно решать уравнения вида $a * x = b$ или $x * a = b$, решением которых будет соответственно $x = a' * b$ и $x = b * a'$.

Группа, операция которой коммутативна, называется *абелева группа*.

Задания для самоподготовки. Задана алгебра $(X, *)$ с одной бинарной операцией. Проверить свойства этой операции и указать, к какому типу эта алгебра относится.

- $X = \mathbb{N}, x * y = x^y$;
- $X = \mathbb{N}, x * y = \text{НОК}(x, y)$;
- $X = \mathbb{R}, x * y = x$;
- X — множество диагональных матриц порядка n, $x * y$ — произведение матриц;
- $X = 2^A, x * y = A \setminus B$.

Группа вычетов Особый интерес представляют группа следующего вида: $\mathbb{Z}_K^+(\{0, 1, 2, \ldots, k-1\}, \oplus_k, 0)$, где $\oplus_k$ — операция сложения по модулю k (остаток от деления $x + y$ на k). Такая группа называется *аддитивной группой вычетов*. Аналогичная группа (при исключении нуля из множества алгебры) с операцией умножения по модулю k $\mathbb{Z}_K^*(\{1, 2, \ldots, k-1\}, \odot_k, 1)$ называется *мультипликативной группой вычетов*.

В мультипликативной группе вычетов элемент образует *замыкание* относительно операции умножения. Например, в $\mathbb{Z}_5^*(\{1, 2, 3, 4\}, \odot_5, 1)$:

$$
\begin{aligned}
&3^1 = 3\\
&3^2 = 3 \odot_5 3 = 4\\
&3^3 = 3 \odot_5 3 \odot_5 3 = 4 \odot_5 3 = 2\\
&3^4 = 2 \odot_5 3 = 1\\
&3^5 = 1 \odot_5 3 = 3 = 3^1
\end{aligned}
$$

Значит, 3^{2006} в этой группе вычисляется легко: $3^{2006} = 3^{2004} \odot_5 3^2 = (3^4)^{501} \odot_5 4 = 4$.

Группа перестановок Рассмотрим конечное множество $A = \{1, 2, \ldots, n\}$, состоящее из n элементов. Перестановкой называется биекция $A \to A$. Перестановка записывается в виде:

$$\begin{pmatrix} 1 & 2 & \dots & n \\ i_1 & i_2 & \dots & i_n \end{pmatrix},$$

где $i_1, \dots i_n \in A$.

На множестве перестановок можно определить операцию *композиции*:

$$\sigma = \begin{pmatrix} 1 & 2 & \dots & n \\ i_1 & i_2 & \dots & i_n \end{pmatrix}, \tau = \begin{pmatrix} 1 & 2 & \dots & n \\ j_1 & j_2 & \dots & j_n \end{pmatrix}, \tau \circ \sigma = \begin{pmatrix} 1 & 2 & \dots & n \\ k_1 & k_2 & \dots & k_n \end{pmatrix},$$

где $k_s = \tau(\sigma(s)) = \tau(i_s) = j_{i_s}$.

Для каждой перестановки можно получить *обратную перестановку*:

$$\sigma = \begin{pmatrix} 1 & 2 & \dots & n \\ i_1 & i_2 & \dots & i_n \end{pmatrix}, \sigma^{-1} = \begin{pmatrix} i_1 & i_2 & \dots & i_n \\ 1 & 2 & \dots & n \end{pmatrix}$$

Множество всех перестановок степени n образуют группу по операции композиции, при этом нейтральным элементом является тождественная перестановка:

$$(A \to A, \circ, \varepsilon), \varepsilon = \begin{pmatrix} 1 & 2 & \dots & n \\ 1 & 2 & \dots & n \end{pmatrix}$$

Такая группа называется *группой перестановок*.

Рассмотрим уравнение над этой группой:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix} \circ X \circ \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 2 & 1 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 4 & 2 \end{pmatrix}$$
$$X = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix}^{-1} \circ \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 4 & 2 \end{pmatrix} \circ \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 2 & 1 & 3 \end{pmatrix}^{-1} =$$
$$= \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix} \circ \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 4 & 2 \end{pmatrix} \circ \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 4 & 1 \end{pmatrix} =$$
$$= \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 2 & 4 \end{pmatrix} \circ \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 4 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{pmatrix}$$

Исследование группоидов Рассмотрим ещё один пример группоида и исследуем свойства его бинарной операции. $\mathcal{A}(\mathbb{R}, \circ)$, где $(\forall x, y)(x \circ y = y)$.

Очевидно, что эта операция *некоммутативна*, так как $x \circ y = y \neq y \circ x = x$. С другой стороны, *ассоциативность* может быть доказана следующим образом:

$$x \circ (y \circ z) = x \circ z = z$$
$$(x \circ y) \circ z = y \circ z = z$$

Значит, это полугруппа. При попытке найти нейтральный элемент, получим:

$$x \circ \varepsilon = x \Leftrightarrow \varepsilon = x,$$

это значит, что нейтральный элемент не существует, и данная алгебра - не моноид.

Кольца, тела и поля Алгебраическую структуру с двумя бинарными операциями $\mathcal{R}(R, +, \cdot, 0, 1)$ называется *кольцом*, если выполняются следующие соотношения:

- $a + (b + c) = (a + b) + c$;
- $a + b = b + a$;
- $a + 0 = a$;
- $(\forall a)(\exists -a : a + (-a) = 0)$;
- $a \cdot (b \cdot c) = (a \cdot b) \cdot c$;
- $a \cdot 1 = 1 \cdot a = a$;
- $a \cdot (b + c) = a \cdot b + a \cdot c$ и $(a + b) \cdot c = a \cdot c + b \cdot c$.

Другими словами, алгебра $(R, +, 0)$ является *абелевой группой*, алгебра $(R, \cdot, 1)$ — *моноидом*, а операция умножения дистрибутивна относительно сложения.

Можно доказать, что в кольце имеет место аннулирующее свойство нуля: $a \cdot 0 = 0 \cdot a = 0$.

В качестве примеров колец можно привести:

- кольцо действительных чисел с привычными операциями сложения и умножения — $(\mathbb{R}, +, \cdot, 0, 1)$. Действительно, по сложению имеем абелеву группу, тогда как по умножению — лишь моноид (обратный к нулю элемент не определён).
- кольцо квадратных матриц степени n — $(M_n, +, \cdot, 0, E)$, где 0 и E — соответственно нулевая и единичная матрица. Также только моноид по умножению, потому что обратная матрица существует не для каждой квадратной матрицы.

В некоторых кольцах существуют *делители нуля*, т.е. такие элементы $a, b \neq 0$, что $a \cdot b = 0$. Для обычной арифметики это кажется удивительным и не выполняется, тогда как для приведённого выше примера кольца матриц можно найти делители нуля:

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Кольцо без делителей нуля называется *областью целостности*. Пример такой алгебры — кольцо целых чисел с операциями сложения и умножения. Кольцо без делителей нуля, множество ненулевых элементов

которого является группой по умножению, называется *телом*. Тело с коммутативной операцией умножения называется *полем*.

Кольца рациональных, действительных и комплексных чисел с операциями арифметического сложения и умножения являются полями.

Задания для самоподготовки. Задана алгебра $(X, +, *)$ с двумя бинарными операциями. Проверить свойства этой операции и указать, к какому типу эта алгебра относится.

- $X = M_n^d$ — множество диагональных матриц степени n;
- $X = 2^A$, $+ — \triangle$, симметрическая разность множеств, а $* — \cap$, пересечение множеств;
- $X = \mathbb{R}[x]$ — множество многочленов от переменной x с действительными коэффициентами.

Кольцо вычетов Важным примером кольца является *кольцо вычетов* по модулю k: $\mathbb{Z}_k(\{0, 1, \ldots, k-1\}, \oplus_k, \odot_k, 0, 1)$. Для разных k это кольцо может обладать разными свойствами. Рассмотрим два случая: $k = 4$ и $k = 5$.

При $k = 4$ имеем следующие таблицы сложения и умножения по модулю 4:

$\oplus_4$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

$\odot_4$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

Видно, что элемент 2 является делителем нуля: $2 \odot_4 2 = 0$.

При $k = 5$ имеем следующие таблицы сложения и умножения по модулю 5:

$\oplus_5$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

$\odot_5$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Здесь делителей нуля нет, а для каждого ненулевого элемента можно найти обратный по умножению: $1^{-1} = 1, 2^{-1} = 3, 3^{-1} = 2, 4^{-1} = 4$. Т.е. данное кольцо вычетов является полем.

Можно доказать, что кольцо вычетов является полем тогда и только тогда, когда k — простое число.

В кольцах вычетов можно решать уравнения и системы уравнений. Например решим данные уравнения в кольце $\mathbb{Z}_7$, для простоты

умножение и сложение по модулю 7 будем обозначать стандартными знаками полюса и умножения:

$$\begin{cases} 2x+3y=1, \\ 3x-y=2 \end{cases} \begin{cases} y=3x-2, \\ 2x+3(3x-2)=1 \end{cases} \begin{cases} y=3x-2, \\ 4x=0, x=0 \end{cases}$$

$$\begin{cases} x=0, \\ y=5 \end{cases}$$

или в кольце вычетов $\mathbb{Z}_5$:

$$x^3+x^2+4x-1=0$$
$$x^3+x^2+4x+4=0$$
$$x^2(x+1)+4(x+1)=0$$
$$(x+1)(x^2+4)=0$$
$$x_1=4,$$
$$x_2^2=1, x_2=1$$

Задания для самоподготовки. Решить систему уравнений или алгебраическое уравнение в полях $\mathbb{Z}_5$ и $\mathbb{Z}_7$:

- $\begin{cases} -4x+3y=1, \\ x+2y=3 \end{cases}$
- $\begin{cases} -x+3y=1, \\ 3x+y=2 \end{cases}$
- $x^3-x+1=0.$

Подалгебры Пусть дана алгебра $\mathcal{A}(A,\Omega)$ и множество $B \subset A$, такое, что $\mathcal{B}(B,\Omega)$ — тоже алгебра с теми же операциями. Тогда $\mathcal{B}$ — подалгебра $\mathcal{A}$.

В качестве примеров подалгебр можно привести:

- группу $(\mathbb{Q},+,0)$ и её подгруппу $(\mathbb{Z},+,0)$;
- кольцо $(\mathbb{R},+,\cdot,0,1)$ и его подкольцо $(\mathbb{Z},+,\cdot,0,1)$.

А вот алгебра $(\mathbb{N}_0,+,0)$ не является подгруппой $(\mathbb{Z},+,0)$, так как не обладает свойствами группы (является лишь моноидом и соответственно подмоноидом $(\mathbb{Z},+,0)$).

Задания для самоподготовки. Для данной группы (кольца) M поверить, является ли H её подгруппой (подкольцом):

- $M = (\mathbb{C} \setminus \{0\}, \cdot), H = \{x \in \mathbb{R}, x > 0\}$;
- $M = (C[a,b], +, \cdot)$, C — класс функций, непрерывных на отрезке, $H = \{f \in C[a,b] : f(a) + f(b) = 0\}$;
- $M = (V_3, +), H = \{\vec{a} \in V_3; (\vec{a}, \vec{n}) = 0\}$ ($\vec{n}$ — заданный вектор).

Полукольца Определение *полукольца* аналогично определению *полугруппы* — снимается требование существования обратного элемента по сложению (из двух полуколец можно получить кольцо, как и в случае группы):

- $a + (b + c) = (a + b) + c$;
- $a + b = b + a$;
- $a + 0 = a$;
- $a \cdot (b \cdot c) = (a \cdot b) \cdot c$;
- $a \cdot 1 = 1 \cdot a = a$;
- $a \cdot (b + c) = a \cdot b + a \cdot c$ и $(a + b) \cdot c = a \cdot c + b \cdot c$;
- $a \cdot 0 = 0 \cdot a = 0$;

Если операция сложения в полукольце *идемпотентна*: $x + x = x$, то такое полукольцо называют *идемпотентным полукольцом*, иногда их называют просто *полукольцами*. Пример такого полукольца:

$$\mathcal{R}^+ = (\mathbb{R}^+ \cup \{+\infty\}, \min, +, +\infty, 0)$$

.

Действительно, операция взятия минимума идемпотентна и $+\infty$ является по ней нейтральным элементом.

Если операция умножения обладает также свойствами идемпотентности и коммутативности, то такое полукольцо называют *симметричным.*

Пример симметричного полукольца: алгебра подмножеств множества $mathcal A = (2^A, \cup, \cap, \varnothing, A)$.

Булевы алгебры Симметричное полукольцо, в котором для каждого элемента x существует дополнение $\overline{x}$ такое, что $x + \overline{x} = 1$ и $x \cdot \overline{x} = 0$, называется *булевой алгеброй*.

Самый распространённым примером булевой алгебры является двухэлементная булева алгебра: $\mathcal{B}(\{0,1\}, \vee, \wedge, 0, 1)$. Интересно, что приведённое выше симметричное полукольцо подмножеств множества A также является булевой алгеброй, в которой дополнение определяется как $\overline{X} = A \setminus X$.

Также представляет интерес n-компонентная булева алгебра:

$$\mathcal{B}(\{0,1\}^n, \vee, \wedge, \mathbf{0}, \mathbf{1}),$$

где операции $\vee$ и $\wedge$ определены на булевых векторах из n элементов, а $\mathbf{0}$ и $\mathbf{1}$ — соответственно нулевой и единичный вектор. Нетрудно показать, что это также булева алгебра.

В n-элементной булевой алгебре определяется стандартное отношение порядка (покомпонентное сравнение): $\overline{\alpha} \leqslant \overline{\beta} \Leftrightarrow \alpha_i \leqslant \beta_i, i = \overline{1,n}$. Диаграммы Хассе для этого отношения булевой алгебры первого, второго и третьего порядка показаны на рисунке 15.

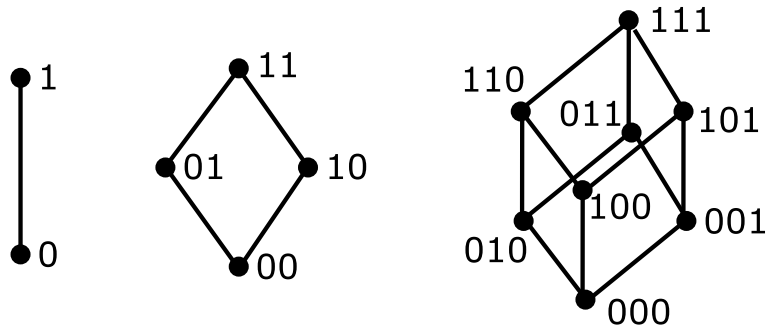

Рис. 15: Диаграмма Хассе для стандартного отношения порядка булевой алгебры

На базисе булевых алгебр (в первую очередь, двухэлементной булевой алгебры) строится теория *булевых функций*.

3 Булевы функции

Булева функция — это отображение вида $f : \{0,1\}^n \to 0,1$, где n — число булевых переменных. В общем случае это скалярная функция векторного переменного.

Важное свойство булевых функции — их число конечно для заданного n и равно 2^{2^n}. То есть, каждая функция может быть задана своей *таблицей истинности* или просто *пронумерована*.

При $n = 0$ существует только две функции-константы: 0 и 1.

Рассмотрим все функции для $n = 1$:

x	f_1	f_2	f_3	f_4
0	0	0	1	1
1	0	1	0	1

$f_1(x) = 0, f_4(x) = 1$ — константы, $f_2(x) = x$ — тождественная функция, $f_3(x) = \overline{x}$ — дополнение.

Рассмотрим все функции для $n = 2$:

x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

$f_1(x_1, x_2) = 0, f_{16}(x_1, x_2) = 1$ — константы, $f_2(x_1, x_2) = x_1 \wedge x_2$ — конъюнкция, $f_7(x_1, x_2) = x_1 \oplus x_2$ — исключающее или, $f_8(x_1, x_2) = x_1 \vee x_2$ — дизъюнкция, $f_9(x_1, x_2) = x_1 \downarrow x_2$ — стрелка Пирса, $f_{10}(x_1, x_2) = x_1 \sim x_2$ — эквивалентность, $f_{14}(x_1, x_2) = x_1 \to x_2$ — импликация, $f_{15}(x_1, x_2) = x_1 \mid x_2$ — штрих Шефера.

Таким образом, каждая булева функция может быть представлена в виде таблицы истинности, так и в виде формулы.

ДНФ Элементарная конъюнкция — формула вида $x_i^{\alpha_i} \wedge x_j^{\alpha_j} \wedge \ldots x_k^{\alpha_k}$, где при $\alpha_i = 0$ используется дополнение переменной, а при $\alpha_i = 1$ — сама переменная. Пример элементарной конъюнкции: $x_1\overline{x_3}x_4$.

Дизъюнктивная нормальная форма — это формула вида $K_1 \vee K_2 \vee \ldots K_m$, где K_i — элементарная конъюнкция.

Можно доказать, что любая функция может быть представлена в виде ДНФ.

Карта Карно Для иллюстрации значения булевой функции, а также для получения её ДНФ используются *карты Карно* — форма изображения булевой функции в виде таблицы, строки и столбцы которой обозначают отдельные переменные.

Например, карта Карно для функции от трёх переменных $f = (00101110)$ будет иметь вид:

$x_2\ x_3$ / x_1	0 0	0 1	1 1	1 0
0	0	0	0	1
1	1	1	0	1

Отметим, что значение переменных в соседних столбцах и строках карты Карно должны быть сравнимы.

Для каждой единицы в карте Карно можно выписать элементарную коньнкцию, с помощью которой она получается — например, единица при $x_1 = 0$ и $x_2x_3 = 10$ может быть представлена в виде конъюнкции $\overline{x_1}x_2\overline{x_3}$. Дизъюнкция всех таких конъюнкций, соответствующих единицам, даст нам ДНФ данной функции.

Минимизация ДНФ Минимальной булевой функцией называется функция, ДНФ которой содержит минимальное число элементарных конъюнкций (а если число конъюнкций равно, то число переменных в конъюнкциях меньше).

Алгоритм минимизации булевой функции состоит из следующих этапов:

1. Построение карты Карно для заданной функции.

2. Выделение *склеек*. Склейка — область, покрывающая только единицы на карте Карно, размер которой равен степени двух. Такая область может быть представлена в виде элементарной конъюнкции. При этом необходимо выделять склейки максимально большого размера. Дизъюнкция всех конъюнкций склеек даёт *сокращённую ДНФ*.

3. Определение *ядра*. Ядро — набор склеек, каждая из которых покрывает единицы, не покрываемые ни одной другой склейкой. Такие склейки называют *ядровыми*. Если в ядро входят все склейки, то минимальная ДНФ равна сокращённой, конец алгоритма.

4. Если существуют склейки, целиком попадающие в ядро, их можно выбросить. В результате получится *ДНФ Квайна*.

   (a) Если ядро покрывает все единицы, то минимизация закончена.

   (b) Если есть единицы, не покрываемые ядерными склейками, то строится *функция Патрика*: для каждой из таких единиц строится дизъюнкция вида $K_1 \vee K_2 \vee \dots K_m$, где K_i — конъюнкция склейки, покрывающецй данную единицу. Такие дизъюнкции объединяются в общую конъюнкцию, которая и носит название функции Патрика. После раскрытия скобок и сокращения получается дизъюнкция нескольких наборов конъюнкций, каждый из таких наборов представляет собой отдельную альтернативу, покрывающую все единицы, не попавшие в ядро. Каждая такая альтернатива, объединённая с ядром, даёт одну из возможных минимальных ДНФ, или одну из *тупиковых* ДНФ.

Рассмотрим пример минимизации булевой функции $f = (00101110)$. На рисунке 16 показана карта Карно для этой функции.

Всего можно выделить три склейки: K_1, K_2 и K_3. В ядро входят K_1 и K_3, тогда как K_2 может быть выброшена. В данном случае ДНФ Квайна и минимальная ДНФ совпадают: $f = K_1 \vee K_3 = x_2\overline{x_3} \vee x1\overline{x_2}$.

На рисунке 17 показана карта Карно похожей функции. В этом случае ядро K_1 и K_4 не покрывает все единицы функции.

Для единицы на пересечении K_2 и K_3 функция Патрика будет иметь вид: $K_2 \vee K_3$. Дальнейшее упрощение не возможно, имеем две альтернативы. Значит, тупиковые ДНФ можно записать в виде:

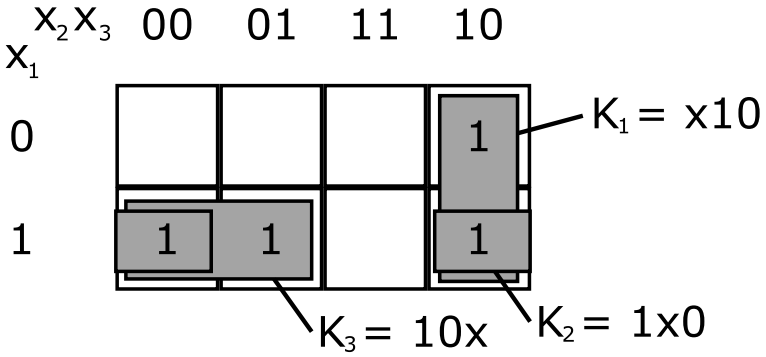

Рис. 16: Пример минимизации ДНФ с помощью карты Карно

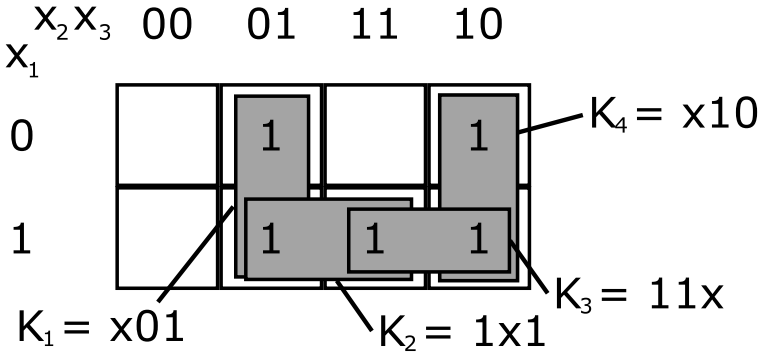

Рис. 17: Пример минимизации ДНФ с помощью карты Карно

$$f = K_1 \vee K_4 \vee \left\{ \begin{array}{l} K_2 \\ K_3 \end{array} \right. = \overline{x_2} x_3 \vee x_2 \overline{x_3} \vee \left\{ \begin{array}{l} x_1 x_3 \\ x_1 x_2 \end{array} \right.$$

Бывают ситуации, когда в ядро не входит ни одна склейка. Например, для функции, показанной на рисунке 18.

В этом случае функция Патрика будет иметь шесть сомножителей — по одному на единицу: Ф.П. $= (K_1 \vee K_2)(K_2 \vee K_3)(K_3 \vee K_4)(K_4 \vee K_5)(K_5 \vee K_6)(K_6 \vee K_1) = (K_1K_2 \vee K_2 \vee K_2K_3 \vee K_1K_3)(K_3K_4 \vee K_4 \vee K_3K_5 \vee K_4K_5)(K_5 \vee K_6) = (K_2 \vee K_1K_3)(K_4 \vee K_3K_5)(K_5 \vee K_6) = (K_2K_4 \vee K_1K_3K_4 \vee K_1K_3K_5 \vee K_2K_3K_5)(K_5 \vee K_6) = K_2K_4K_6 \vee K_1K_3K_5$.

В результате имеем две альтернативы, две тупиковые ДНФ:

$$f = \left\{ \begin{array}{l} K_1 \vee K_3 \vee K_5 \\ K_2 \vee K_4 \vee K_6 \end{array} \right. = \left\{ \begin{array}{l} \overline{x_1 x_2} \vee x_1 x_3 \vee x_2 \overline{x_3} \\ \overline{x_2} x_3 \vee x_1 x_2 \vee \overline{x_1 x_3} \end{array} \right.$$

Системы булевых функций. Базис Жегалкина Конечное множество булевых функций $\mathcal{F} = \{f_1, f_2, \ldots f_n\}$ называют *системой булевых функций*. Систему булевых функций называют *полной*, если *любая* булева функция может быть выражена в виде формулы над этой системой (другими словами, если она представима через функции данной системы).

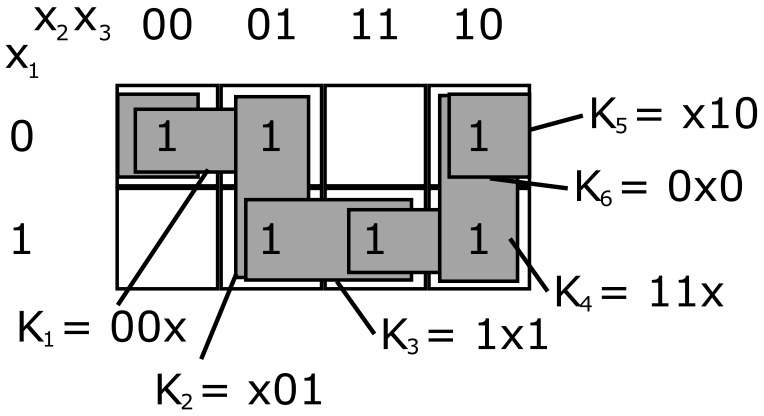

Рис. 18: Пример минимизации ДНФ с помощью карты Карно

Примером полной системы является так называемый *стандартный базис*, содержащий дизъюнкцию, конъюнкцию и отрицание: $\mathcal{F}_0 = \{\vee, \wedge, \overline{\ }\}$. Полнота этой системы легко доказывается тем, что любая булева функция может быть представлена в виде ДНФ или КНФ. А учитывая, что $x \vee y = \overline{\overline{x} \wedge \overline{y}}$ и $x \wedge y = \overline{\overline{x} \vee \overline{y}}$, полными являются даже системы $\{\vee, \overline{\ }\}$ и $\{\wedge, \overline{\ }\}$.

Другой распространённой полной системой булевых функций является *базис Жегалкина*: $\mathcal{F}_{\text{Ж}} = \{\oplus, \cdot, 1\}$, включающий исключающее или, конъюнкцию и константу 1. Можно доказать, что любая булева функция представима в виде так называемого *полинома Жегалкина*:

$$f(x_1, \dots x_n) = \sum_{\substack{i_1, i_2, \dots i_k \in \{1, \dots n\} \\ k \in \overline{1,n}}} a_{i_1, i_2, \dots i_k} x_{i_1} x_{i_2} \dots x_{i_k} \oplus a_0,$$

где $a_j \in \{0, 1\}$. В частном случае, полином Жегалкина имеет линейный вид:

$$f(x_1, \dots x_n) = \sum_{i \in \overline{1,n}} a_i x_i \oplus a_0.$$

Булева функция, представимая в виде линейного полинома Жегалкина, называется *линейной*.

Существует простой способ выражения любой булевой функции над базисом Жегалкина. Этот способ носит название *метода неопределённых коэффициентов*. Рассмотрим этот метод на примере:

Рассмотрим функцию $f(x_1, x_2, x_3) = (00111001)$. В общем виде полином Жегалкина для этой функции имеет вид:

$$f(x_1, x_2, x_3) = a_{123} x_1 x_2 x_3 \oplus a_{12} x_1 x_2 \oplus a_{23} x_2 x_3 \oplus a_{13} x_1 x_3 \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_3 x_3 \oplus a_0$$

Вычислим все коэффициенты начиная с a_0, последовательно подставляя в полином известные значения функции f:

$f(0,0,0) = a_0 = 0,$
$f(0,0,1) = a_3 \oplus a_0 = a_3 = 0,$
$f(0,1,0) = a_2 \oplus a_0 = a_2 = 1,$
$f(1,0,0) = a_1 \oplus a_0 = a_1 = 1,$
$f(0,1,1) = a_{23} \oplus a_2 \oplus a_3 \oplus a_0 = a_{23} \oplus 1 = 1 \Rightarrow a_{23} = 0,$
$f(1,0,1) = a_{13} \oplus a_1 \oplus a_3 \oplus a_0 = a_{13} \oplus 1 = 0 \Rightarrow a_{13} = 1,$
$f(1,1,0) = a_{12} \oplus a_1 \oplus a_2 \oplus a_0 = a_{12} = 0,$
$f(1,1,1) = a_{123} \oplus a_{12} \oplus a_{23} \oplus a_{13} \oplus a_1 \oplus a_2 \oplus a_3 \oplus a_0 = a_{123} \oplus 1 = 1 \Rightarrow a_{123} = 0$

Таким образом, функция имеет вид:

$$f(x_1, x_2, x_3) = x_1x_3 \oplus x_1 \oplus x_2$$

и не является линейной.

Теорема Поста Существует метод проверки полноты системы, называемый теоремой Поста. Она основывается на выделении пяти классов Поста булевых функций:

класс функций сохраняющих 0 $f \in T_0 \Leftrightarrow f(0,0,\dots 0) = 0,$

класс функций сохраняющих 1 $f \in T_1 \Leftrightarrow f(1,1,\dots 1) = 1,$

класс самодвойственных функций $f \in S \Leftrightarrow (\forall\alpha)(f(\overline{\alpha}) = \overline{f(\alpha)}),$

класс монотонных функций $f \in M \Leftrightarrow (\forall\alpha,\beta)(\alpha \leqslant \beta \Rightarrow f(\alpha) \leqslant f(\beta))$, где подразумевается стандартный порядок булевой алгебры (при котором существуют и несравнимые элементы!),

класс линейных функций $f = \sum\limits_{i\in\overline{1,n}} a_i x_i \oplus a_0$, т.е. функция представима в виде полинома Жегалкина первой степени.

Можно доказать, что все эти классы являются *замкнутыми*, т.е. любая комбинация функций из одного класса остаётся в том же классе.

> **Теорема (Поста):** система булевых функций полна тогда и только тогда, когда она целиком не лежит ни в одном из классов Поста.

Рассмотрим пример: $f(x_1, x_2, x_3) = x_1(\overline{x_1}|x_3)(\overline{x_1} \oplus \overline{x_3}) \to (x_2 \sim x_3), w = (11100110)$. Необходимо проверить полноту системы $\{f, w\}$.

Функция f принимает следующие значения: $f = (11111101)$. Проверка принадлежности первых четырёх классов поста может быть проведена очень быстро. Для проверки линейности построим представление данных функций в виде полинома Жегалкина:

$f(0,0,0) = a_0 = 1,$
$f(0,0,1) = a_3 \oplus 1 = 1 \Rightarrow a_3 = 0,$
$f(0,1,0) = a_2 \oplus 1 = 1 \Rightarrow a_2 = 0,$
$f(1,0,0) = a_1 \oplus 1 = 1 \Rightarrow a_1 = 0,$
$f(0,1,1) = a_{23} \oplus 0 \oplus 0 \oplus 1 = 1 \Rightarrow a_{23} = 0,$
$f(1,0,1) = a_{13} \oplus 0 \oplus 0 \oplus 1 = 1 \Rightarrow a_{13} = 0,$
$f(1,1,0) = a_{12} \oplus 0 \oplus 0 \oplus 1 = 0 \Rightarrow a_{12} = 1,$
$f(1,1,1) = a_{123} \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 = 1 \Rightarrow a_{123} = 1.$

Поучается, что $f = x_1x_2x_3 \oplus x_1x_2 \oplus 1$ нелинейна.

$w(0,0,0) = a_0 = 1,$
$w(0,0,1) = a_3 \oplus 1 = 1 \Rightarrow a_3 = 0,$
$w(0,1,0) = a_2 \oplus 1 = 1 \Rightarrow a_2 = 0,$
$w(1,0,0) = a_1 \oplus 1 = 0 \Rightarrow a_1 = 1,$
$w(0,1,1) = a_{23} \oplus 0 \oplus 0 \oplus 1 = 0 \Rightarrow a_{23} = 1,$
$w(1,0,1) = a_{13} \oplus 1 \oplus 0 \oplus 1 = 1 \Rightarrow a_{13} = 1,$
$w(1,1,0) = a_{12} \oplus 1 \oplus 0 \oplus 1 = 1 \Rightarrow a_{12} = 1,$
$w(1,1,1) = a_{123} \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 = 0 \Rightarrow a_{123} = 1.$

Поучается, что $w = x_1x_2x_3 \oplus x_1x_2 \oplus x_2x_3 \oplus x_1x_3 \oplus x_1 \oplus 1$ нелинейна.
Принадлежность классам Поста обобщим в таблице:

	T_0	T_1	S	M	L
f	-	+	-	-	-
w	-	-	-	-	-

Система функций $\{f, w\}$ является полной, более того, полной является уже система $\{w\}$.

Выразим стандартный базис $\{\wedge, \overline{\ }\}$ через функцию w. Воспользуемся тем, что w не сохраняет ни 0, ни 1, значит:

$$w(x,x,x) = \overline{x}.$$

Так как w — несамодвойственна, выразим константы через отрицание. Для этого найдём такой вектор α, что $w(\alpha) = w(\overline{\alpha})$. Видно, что это выполняется при $\alpha = (010)$. Значит, $w(\overline{x}, x, \overline{x}) = w(x, \overline{x}, x) = 1$. Тогда 0 можно получить с помощью отрицания.

w — нелинейна, выразим конъюнкцию из полинома Жегалкина этой функции. Можно получить, что $w(x_1, x_2, 0) = x_1x_2 \oplus x_1 \oplus 1$. Тогда

$$w(x_1, x_2 \oplus 1, 0) \oplus 1 = x_1(x_2 \oplus 1) \oplus x_1 \oplus 1 \oplus 1 = x_1x_2,$$

откуда $xy = \overline{w(x, \overline{y}, 0)}$.

Схемы из функциональных элементов Часто для формального задания булевых функций используется *схемы из функциональных элементов* (СФЭ) — графы специального вида, основу которых составляют собственно функциональные элементы, т.е. булевы функции, принимающие на вход переменные и выдающие результат.

На рисунке 19 изображены функциональные элементы для стандартных булевых функций. С их помощью, к примеру, функция исключающего или может быть реализована следующим образом (рисунок 20):

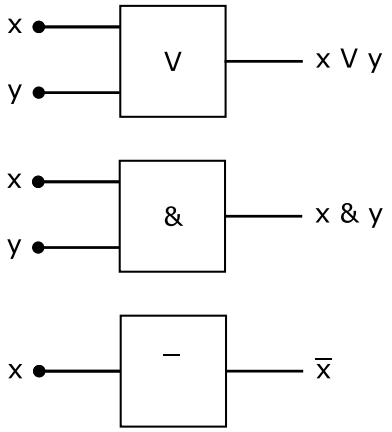

Рис. 19: Функциональные элементы для стандартных функций

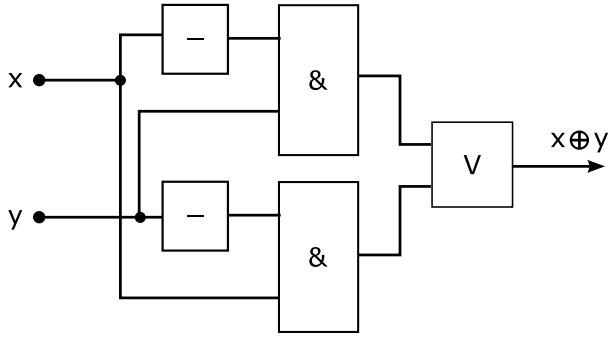

Рис. 20: СФЭ для функции исключающего или

Аналогичным образом могут быть построены схемы для булевых функций любой сложности. В электронике часто используются схемы, выполняющие сложение битовых переменных. Такие схемы называются *сумматорами*. Таблица истинности для булевых функций сложения двух $x_1 + x_2 = y_1 y_2$ и трёх $x_1 + x_2 + x_3 = y_1 y_2$ однобитовых (булевых) переменных содержит два столбца результата (результат имеет большее число разрядов, чем аргументы), т.е. мы имеем две булевых функции для каждой из операций сложения:

x_1	x_2	y_1	y_2
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

x_1	x_2	x_3	y_1	y_2
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

СФЭ для сумматора сложения трёх переменных представлен на рисунке 21. Удобство использования этого сумматора таково, что на его основе можно стоить сумматоры для сложения сколь угодно больших чисел. Например, на рисунке 22 показан сумматор для сложения двух трёхразрядных переменных $x_1x_2x_3 + y_1y_2y_3 = z_1z_2z_3z_4$:

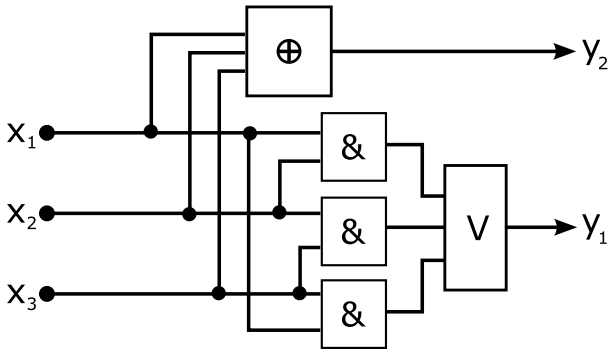

Рис. 21: СФЭ сумматора

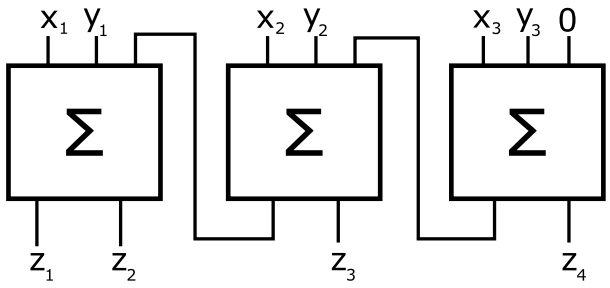

Рис. 22: СФЭ для сумматора трёхразрядных переменных

4 Графы

В теории графов выделяют два вида графов: *неориентированные* и *ориентированные.*

Неориентированные графы Неориентированным графом называют пару (V, E), где V — конечное множество *вершин*, а E — множество *рёбер* такое, что $E \subseteq \{\{v, w\} | v, w \in V \ \ \& \ \ v \neq w\}$. Важно, что рёбра могут соединять какие-то *две* вершины. Вершины можно представить в виде точек на плоскости, а дуги — линиями, их соединяющими. При этом для нас представляют интерес не координаты точек или длина и форма линий, а *связь* между дугами и вершинами.

На рисунке 23 показаны примеры неориентированных графов. Граф **(а)** определяется как:

$$V = \{\text{“Киев”}, \text{“Лондон”}, \text{“Москва”}, \text{“Рио”}\},$$
$$E = \{\{\text{“Киев”}, \text{“Москва”}\}, \{\text{“Лондон”}, \text{“Москва”}\}\}.$$

А вот в графе **(в)** все вершины соедины с каким-либо ребром:

$$V = \{1, 2, 3, 4, 5, 6, 7\},$$
$$E = \{\{1, 2\}, \{1, 3\}, \{1, 7\}, \{2, 4\}, \{2, 5\}, \{3, 6\}, \{4, 5\}, \{6, 7\}\}$$

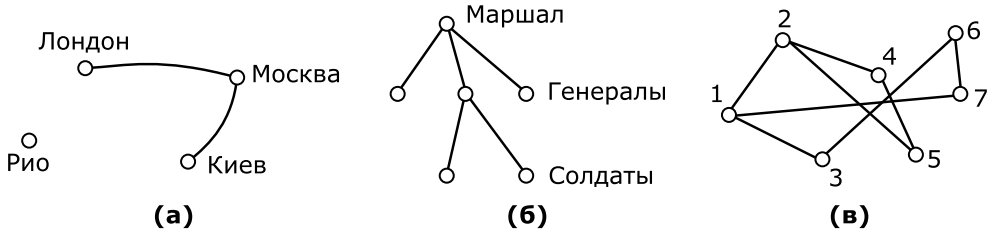

Рис. 23: Примеры неориентированных графов

Можно легко получить число возможных графов для заданного числа вершин $|V| = n$. Максимальное число рёбер в таком графе — число неупорядоченных пар из n, т.е. $C_n^2 = \frac{n(n-1)}{2}$. Тогда общее число неориентированных графов равно мощности множества подмножеств всех рёбер, т.е. $2^{\frac{n(n-1)}{2}}$.

Для каждой вершины вводится понятие *степени*:

$$dg(v) = |\{u | \{u, v\} \in E\}|,$$

т.е. число рёбер, соединённых с вершиной.

Цепью в графе называют произвольную конечную или бесконечную последовательность вершин, в которой каждая последующая соединена ребром с предыдущей. Для конечной цепи число рёбер в ней называют *длиной*. Цепь называется *простой*, если все её вершины, кроме, быть может, первой и последней, попарно различны и все её рёбра попарно различны. Простая цепь ненулевой длины, начало и конец которой совпадают, называется *циклом*. Неориентированный граф, не имеющий циклов называют *ациклличным*. На рисунке 24 показаны цепь, простая цепь и цикл.

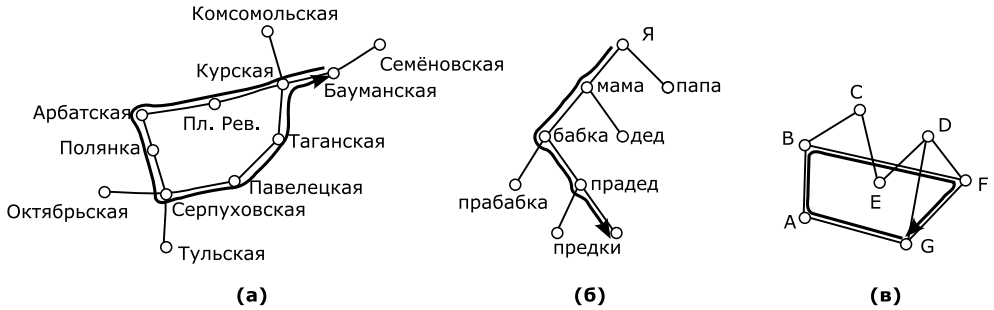

Рис. 24: Цепь, простая цепь и цикл

Подграфом $\mathcal{G}' = (V', E')$ графа $\mathcal{G} = (V, E)$ называют такой *граф*, что: $V' \subseteq V \ \ \& \ \ E' \subseteq E$. Так как $\mathcal{G}'$ — граф, в нём не могут быть только вершины, соединённые с рёбрами. Подграф называется *остовным*, если $V = V'$.

Неориентированный граф называется *связным*, если в нём любые две вершины соединены цепью. Максимальный связный подграф — *компонента связности* этого графа (или просто *компонента*). Компоненты не имеют ни общих вершин, ни общих рёбер.

Для двух вершин можно ввести бинарное отношение *достижимости*: uv выполняется, когда вершины u и v можно соединить цепью. Можно доказать, что это отношение является отношением эквивалентности. Тогда граф может быть факторизован, и элементами фактормножества будут являться компоненты данного графа. На рисунке 25 показан пример графа из нескольких компонент связности.

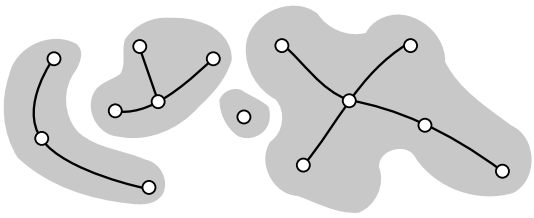

Рис. 25: Компоненты неориентированного графа

Особый интерес представляют особые виды графов. Ациклический связный граф называется *деревом*. Обычно в дереве одна из вершин выделяется в качестве *корня*. Более общим случаем дерева является *лес*: граф, состоящий из двух и более деревьев (ациклический граф). Вершины, степень которых равна 1 и не являющиеся корнем, называют *листьями*. Граф, состоящий только из корня и листьев, называют *кустом*. На рисунке 26 показаны дерево, куст и лес.

Помимо перечисления множеств V и E, неориентированные графы можно представлять в виде матрицы *смежности*: матрица имеет квадратный и симметрический вид, размером $n \times n$, где n — число вершин. Элементы матрицы равны:

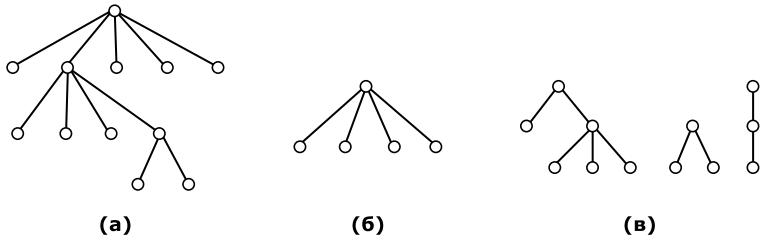

Рис. 26: Дерево, куст и лес

$$a_{i,j} = \begin{cases} 1, \text{если } \{v_i, v_j\} \in E, \\ 0, \text{если } \{v_i, v_j\} \notin E. \end{cases}$$

Для графа на рисунке 23, **(а)** матрица смежности будет иметь вид:

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

Другой матрицей, представляющей для нас интерес, является матрица *достижимости*, размер которой также $n \times n$. Элементы матрицы равны:

$$a_{i,j} = \begin{cases} 1, \text{если } v_i \Rightarrow^* v_j \\ 0, \text{иначе.} \end{cases}$$

В том же примере матрица достижимости отличается от матрицы смежности:

$$C = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Оринтированные графы Ориентированный граф отличается наличием направления на дугах, соединяющих вершины. *Ориентированный граф* определяется как пара (V, E), где V — множество вершин, как в неориентированном графе, а $E \subseteq V \times V$ — множество дуг. Каждая дуга — упорядоченная пара вершин, стрелка на которой направлена от первой вершины ко второй. Отметим также, что в случае неориентированного графа возможны *петли*, т.е. дуги вида (v, v). На рисунке 27 показан пример ориентированного графа. Для него множество вершин и дуг равно:

$V = \{$“Вася”, “Гога”, “Гриша”, “Маша”, “Миша”, “Оля”$\}$,
$E = \{$(“Вася”, “Маша”), (“Гога”, “Гриша”), (“Гога”, “Миша”),
(“Гриша”, “Оля”), (“Маша”, “Вася”), (“Миша”, “Оля”),
(“Оля”, “Оля”)$\}$

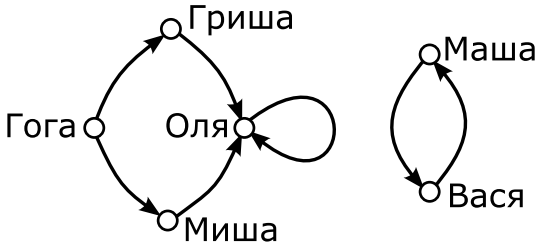

Рис. 27: Пример ориентированного графа

Число ориентированных графов для заданного числа вершин больше, чем неориентированных. Всего между n вершинами можно провести n^2 дуг (число упорядоченных пар). Значит число графов будет равно мощности множества подмножеств всех дуг: 2^{n^2}.

Для ориентированных графов степень вершины состоит из двух частей:

полустепень захода $dg^-(v) = |\{v|(w, v) \in E\}|$ — число дуг, входящих в вершину;

полустепень исхода $dg^+(v) = |\{v|(v, w) \in E\}|$ — число дуг, исходящих из вершины.

Например, для вершины “Оля” на рисунке 27 полустепень захода равна 3, а полустепень исхода — 1.

Аналогом цепи в ориентированных графах является *путь* — при этом движение возможно только в направлении стрелок. Также определяется и *простой путь*, а аналог цикла называется *контуром*.

В ориентированных графах определяется три вида связности:

связность Ориентированный граф называется *связным* (просто связным, связным односторонне), если $(\forall v, u \in V)(v \Rightarrow^* u \vee u \Rightarrow^* v)$, т.е. существует путь нулевой или большей длины из u в v или из v в u.

сильная связность Ориентированный граф называется *сильно связным* (связным двусторонне), если $(\forall v, u \in V)(v \Rightarrow^* u \ \ \& \ \ u \Rightarrow^* v)$, т.е. существуют пути нулевой или большей длины из u в v и из v в u.

слабая связность Ориентированный граф называется *слабо связным*, если соответствующий ему неориентированный граф (стереть все стрелочки и удалить петли) связен.

На рисунке 28 показаны связный, сильно и слабо связные подграфы исходных графов.

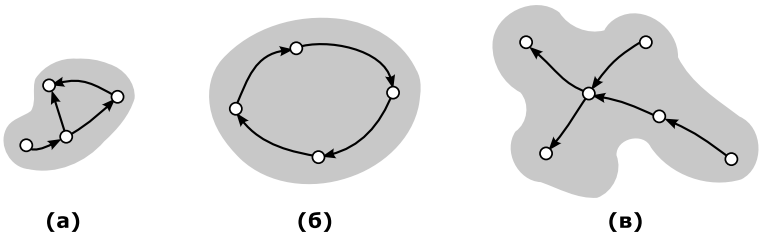

Рис. 28: Связность в ориентированных графах

Ориентированное дерево определяется следующим образом:

1. $dg^-(v) \leqslant 1$ для всех вершин;
2. $dg^-(v) = 0$ для единственной вершины, называемой корнем;
3. в графе нет контуров.

Куст и лес определяются аналогично. Граф более общего вида, не имеющий контуров, называется *сетью*.

Ориентированные графы также могут задаваться матрицей смежности. В этом случае элементы матрицы равны:

$$a_{i,j} = \begin{cases} 1, \text{если } (v_i, v_j) \in E, \\ 0, \text{если } (v_i, v_j) \notin E. \end{cases}$$

Такая матрица уже не является в общем случае симметричной. Для графа на рисунке 27 матрица будет равна:

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

Аналогично определяется и матрица достижимости. Для нашего примера такая матрица будет равна:

$$C = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Идемпотентные полукольца Идемпотентным называется полукольцо, $(A, \oplus, \odot, \mathbf{0}, \mathbf{1})$, в котором операция сложения идемпотентна: $(\forall a \in A)(a \oplus a = a)$.

Примером таких полуколец являются:

- *булево полукольцо* $\mathcal{B}_2 = (\{0, 1\}, \vee, \wedge, 0, 1)$;
- $\mathcal{R}^+ = ([0, +\infty], min, +, +\infty, 0)$.

В указанных полукольцах для каждого элемента можно ввести операцию *итерации*: $a^* = a^0 \oplus a^1 \oplus a^2 \oplus \dots$ — бесконечная сумма степеней элементов. Под степенью n элемента понимается умножение его на себя n раз, а нулевая степень равна единице полукольца по определению: $a^0 = \mathbf{1}$.

Для приведённых выше полуколец итерация для любых элементов находится очень просто:

- $\mathcal{B}_2$:

$$0^* = 0^0 \oplus 0^1 \oplus 0^2 \oplus \dots = 1 + 0 + 0 + \dots = 1$$
$$1^* = 1^0 \oplus 1^1 \oplus 0^2 \oplus \dots = 1 + 1 + 1 + \dots = 1$$

- $\mathcal{R}^+$: для любого a

$$a^* = a^0 \oplus a^1 \oplus \dots = \mathbf{1} \oplus a^1 \oplus a^2 = min\{0, a^1, a^2, \dots\} = 0$$

Идемпотентные полукольца интересны тем, что в них можно решать уравнения вида:

$$x = a \odot x \oplus b \Leftrightarrow x = a^* \odot b;$$
$$x = x \odot a \oplus b \Leftrightarrow x = b \odot a^*.$$

Интересно, что такие необычные для арифметики уравнения как $x = x$ или $x = x + \mathbf{1}$ имеют здесь единственные решения:

$$x = x \oplus \mathbf{1} \Leftrightarrow x = \mathbf{1}^* \odot \mathbf{1} = \mathbf{1};$$
$$x = x \Leftrightarrow x = \mathbf{1} \odot x \oplus \mathbf{0} \Leftrightarrow x = \mathbf{1}^* \odot \mathbf{0} = \mathbf{0}.$$

Нахождение матрицы достижимости Важной задачей в теории графов является нахождение матрицы достижимости по матрице смежности. Можно доказать, что матрица достижимости может быть получена как итерация матрицы смежности:

$$C = A^*,$$

где элементами матриц являются нули и единицы, о операции соответствуют операциям идемпотентного полукольца $\mathcal{B}_2$. В этом случае матрица C может быть получена при решении уравнения $X = A \odot X$, так как оно эквивалентно уравнению $X = A \odot X \oplus E$, где E — единичная матрица.

Таким образом, для получения матрицы C необходимо решить n систем уравнений следующего вида для $k = \overline{1,n}$:

$$\begin{cases} x_1 = a_{11} \odot x_1 \oplus a_{12} \odot x_2 \oplus \cdots \oplus a_{1n} \odot x_n \oplus \varepsilon_{1k} \\ x_2 = a_{21} \odot x_1 \oplus a_{22} \odot x_2 \oplus \cdots \oplus a_{2n} \odot x_n \oplus \varepsilon_{2k} \\ \cdots \\ x_n = a_{n1} \odot x_1 \oplus a_{n2} \odot x_2 \oplus \cdots \oplus a_{nn} \odot x_n \oplus \varepsilon_{nk} \end{cases},$$

где a_{ij} — элементы матрицы смежности, а ε_{ij} — элементы единичной матрицы. Решением каждой из систем будет k-й столбец матрицы C.

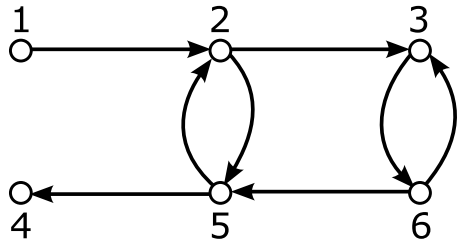

Рис. 29: Ориентированный граф

Рассмотрим пример. На рисунке 29 показан ориентированный граф из шести вершин, матрица смежности которого равна:

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

1-й столбец

$$\left\{\begin{array}{l} x_1 = 1 \odot x_2 \oplus 1 \\ x_2 = 1 \odot x_3 \oplus 1 \odot x_5 \\ x_3 = 1 \odot x_6 \\ x_4 = 0 \\ x_5 = 1 \odot x_2 \oplus 1 \odot x_4 \\ x_6 = 1 \odot x_3 \oplus 1 \odot x_5 \end{array}\right. \Leftrightarrow \left\{\begin{array}{l} x_1 = 1 \\ x_2 = 1 \odot x_3 \oplus 1 \odot (1 \odot x_2) \\ x_3 = 1 \odot x_6 \\ x_4 = 0 \\ x_5 = 1 \odot x_2 \\ x_6 = 1 \odot (1 \odot x_6) \oplus 1 \odot x_5 \end{array}\right. \Leftrightarrow$$

$$\Leftrightarrow \left\{\begin{array}{l} x_1 = 1 \\ x_2 = 1^* \odot (1 \odot x_3) = 1 \odot x_3 \\ x_3 = 1 \odot x_6 \\ x_4 = 0 \\ x_5 = 1 \odot x_2 \\ x_6 = 1^* \odot (1 \odot x_5) = 1 \odot x_5 \end{array}\right. \Leftrightarrow \left\{\begin{array}{l} x_1 = 1 \\ x_2 = 1 \odot (1 \odot (1 \odot (1 \odot x_2))) = x_2 \\ x_3 = 1 \odot (1 \odot (1 \odot x_2)) \\ x_4 = 0 \\ x_5 = 1 \odot x_2 \\ x_6 = 1 \odot (1 \odot x_2) \end{array}\right. \Leftrightarrow$$

$$\Leftrightarrow \left\{\begin{array}{l} x_1 = 1 \\ x_2 = 0 \\ x_3 = 0 \\ x_4 = 0 \\ x_5 = 0 \\ x_6 = 0 \end{array}\right.$$

2-й столбец

$$\left\{\begin{array}{l} x_1 = 1 \odot x_2 \\ x_2 = 1 \odot x_3 \oplus 1 \odot x_5 \oplus 1 \\ x_3 = 1 \odot x_6 \\ x_4 = 0 \\ x_5 = 1 \odot x_2 \oplus 1 \odot x_4 \\ x_6 = 1 \odot x_3 \oplus 1 \odot x_5 \end{array}\right. \Leftrightarrow \left\{\begin{array}{l} x_1 = 1 \odot x_2 \\ x_2 = 1 \\ x_3 = 1 \odot x_6 \\ x_4 = 0 \\ x_5 = 1 \odot x_2 = 1 \\ x_6 = 1^* \odot 1 \odot x_5 = 1 \odot x_5 \end{array}\right. \Leftrightarrow \left\{\begin{array}{l} x_1 = 1 \\ x_2 = 1 \\ x_3 = 1 \\ x_4 = 0 \\ x_5 = 1 \\ x_6 = 1 \end{array}\right.$$

3-6 столбцы Вычисляются аналогично.

Полученная матрица достижимости примет вид:

$$C = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Нахождение кратчайших путей Теория графов широко применяется при решении транспортных задач — нахождения кратчайших маршрутов в графах дорог, сетей и т.п. Для этого используются *размеченные* графы:

$\mathcal{G} = (V, E, \varphi)$, где $\varphi : E \mapsto S \setminus \{0\}$ — весовая функция над некоторым полукольцом $(S, \oplus, \odot, 0, 1)$. Т.е. каждому ребру (или дуге) сопоставляется число, символизирующее длину пути, сложность задачи или любую другую стоимость перемещения между вершинами.

Рассмотрим граф, размеченный над полукольцом $\mathcal{R}^+$ (см. рисунок 30).

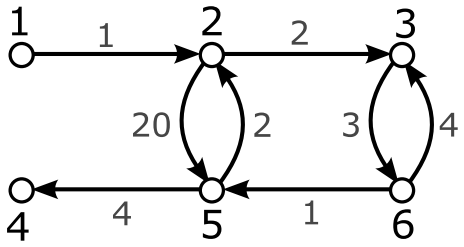

Рис. 30: Граф с размеченными дугами

Для такого графа можно записать матрицу A *метог дуг* (эта матрица аналогична матрице смежностей), где

$$a_{ij} = \begin{cases} \varphi((v_i, v_j)), \text{если } (v_i, v_j) \in E \\ +\infty, \text{иначе} \end{cases}$$

В нашем случае матрица меток дуг равна:

$$A = \begin{pmatrix} +\infty & 1 & +\infty & +\infty & +\infty & +\infty \\ +\infty & +\infty & 2 & +\infty & 20 & +\infty \\ +\infty & +\infty & +\infty & +\infty & +\infty & 3 \\ +\infty & +\infty & +\infty & +\infty & +\infty & +\infty \\ +\infty & 2 & +\infty & 4 & +\infty & +\infty \\ +\infty & +\infty & 4 & +\infty & 1 & +\infty \end{pmatrix}$$

Можно доказать, что на основе этой матрицы по аналогии можно получить *матрицу стоимостей* путей $C = A^*$, где c_{ij} — стоимость кратчайшего пути между вершинами i и j, т.е. сумма меток всех дуг для данного пути. Если же пути между вершинами i и j не существует, то соответствующий элемент матрицы C равен $+\infty$.

Матрица стоимостей C находится аналогично, с помощью решения n систем уравнений. Важно отметить, что в данном случае вся работа ведётся в полукольце $\mathcal{R}^+$, в котором нулём полукольца является $+\infty$, а единицей — 0!

1-й столбец

$$\begin{cases} x_1 = 1 \odot x_2 \oplus 0 \\ x_2 = 2 \odot x_3 \oplus 20 \odot x_5 \\ x_3 = 3 \odot x_6 \\ x_4 = +\infty \\ x_5 = 2 \odot x_2 \oplus 4 \odot x_4 \\ x_6 = 4 \odot x_3 \oplus 1 \odot x_5 \end{cases} \Leftrightarrow \begin{cases} x_1 = 0 \\ x_2 = 2 \odot x_3 \oplus 20 \odot (2 \odot x_2) \\ x_3 = 3 \odot x_6 \\ x_4 = +\infty \\ x_5 = 2 \odot x_2 \\ x_6 = 4 \odot (3 \odot x_6) \oplus 1 \odot x_5 \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} x_1 = 0 \\ x_2 = 2 \odot x_3 \oplus 22 \odot x_2 \\ x_3 = 3 \odot x_6 \\ x_4 = +\infty \\ x_5 = 2 \odot x_2 \\ x_6 = 7 \odot x_6 \oplus 1 \odot x_5 \end{cases} \Leftrightarrow \begin{cases} x_1 = 0 \\ x_2 = 22^* \odot 2 \odot x_3 = 2 \odot x_3 \\ x_3 = 3 \odot x_6 \\ x_4 = +\infty \\ x_5 = 2 \odot x_2 \\ x_6 = 7^* \odot 1 \odot x_5 = 1 \odot x_5 \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} x_1 = 0 \\ x_2 = 2 \odot (3 \odot (2 \odot (2 \odot x_2))) = 10 \odot x_2 \\ x_3 = 3 \odot (2 \odot (2 \odot x_2)) \\ x_4 = +\infty \\ x_5 = 2 \odot x_2 \\ x_6 = 1 \odot (2 \odot x_2) \end{cases} \Leftrightarrow \begin{cases} x_1 = 0 \\ x_2 = +\infty \\ x_3 = +\infty \\ x_4 = +\infty \\ x_5 = +\infty \\ x_6 = +\infty \end{cases}$$

2-й столбец

$$\begin{cases} x_1 = 1 \odot x_2 \\ x_2 = 2 \odot x_3 \oplus 20 \odot x_5 \oplus 0 \\ x_3 = 3 \odot x_6 \\ x_4 = +\infty \\ x_5 = 2 \odot x_2 \oplus 4 \odot x_4 \\ x_6 = 4 \odot x_3 \oplus 1 \odot x_5 \end{cases} \Leftrightarrow \begin{cases} x_1 = 1 \odot x_2 \\ x_2 = 0 \\ x_3 = 3 \odot x_6 \\ x_4 = +\infty \\ x_5 = 2 \odot x_2 \oplus 4 \odot x_4 \\ x_6 = 4 \odot (3 \odot x_6) \oplus 1 \odot x_5 \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} x_1 = 1 \\ x_2 = 0 \\ x_3 = 3 \odot x_6 \\ x_4 = +\infty \\ x_5 = 2 \\ x_6 = 7^* \odot 1 \odot x_5 = 1 \odot x_5 = 3 \end{cases} \Leftrightarrow \begin{cases} x_1 = 1 \\ x_2 = 0 \\ x_3 = 6 \\ x_4 = +\infty \\ x_5 = 2 \\ x_6 = 3 \end{cases}$$

3-6 столбцы Вычисляются аналогично.

Полученная матрица стоимостей будет иметь вид:

$$C = \begin{pmatrix} 0 & 1 & 3 & 11 & 7 & 6 \\ +\infty & 0 & 2 & 10 & 6 & 5 \\ +\infty & 6 & 0 & 8 & 4 & 3 \\ +\infty & +\infty & +\infty & 0 & +\infty & +\infty \\ +\infty & 2 & 4 & 4 & 0 & 7 \\ +\infty & 3 & 4 & 5 & 1 & 0 \end{pmatrix}$$

Гомоморфизм и автоморфизм графов Пусть заданы два графа $\mathcal{G}_1 = (V_1, E_1)$ и $\mathcal{G}_2 = (V_2, E_2)$, отображение множества вершин первого графа во второе $h : V_1 \mapsto V_2$ — *гомоморфизм* графа $\mathcal{G}_1$ в $\mathcal{G}_2$, если $(\forall v, w \in V_1)(\{v, w\} \in E_1 \Rightarrow \{h(v), h(w)\} \in E_2)$. Если отображение h биективно, то графы называются *изоморфными*.

Примеры гомоморфизма и не гомоморфизма представлены на рисунке 31.

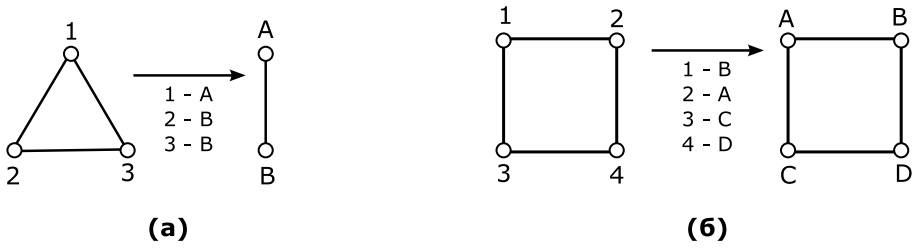

Рис. 31: Пример гомоморфизма и не гомоморфизма

Аналогичным образом гомоморфизм определяется для ориентированных графов. Рассмотрим соответствующий пример: для заданного графа (см. рисунок 32) необходимо найти все двухвершинные гомоморфные образы.

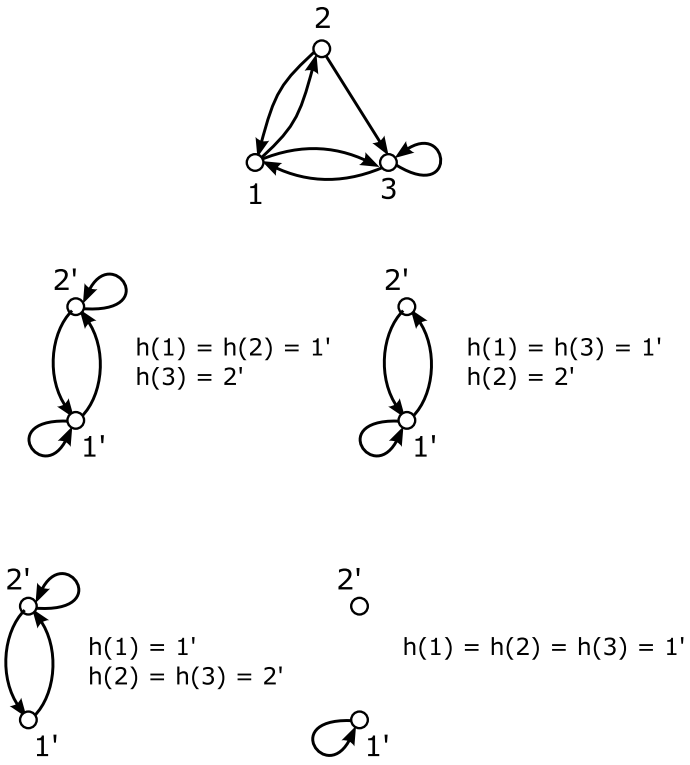

Рис. 32: Гомоморфизм в ориентированном графе

Изоморфизм графа на себя (перестановка вершин) называется *автоморфизмом*.

Для анализа возможных существующих автоморфизмов в неориентированных графах удобно использовать дополнения к графам: граф с тем же набором вершин и набором дуг, не принадлежащих исходному графу. На рисунке 33 показан исходный граф и его дополнение.

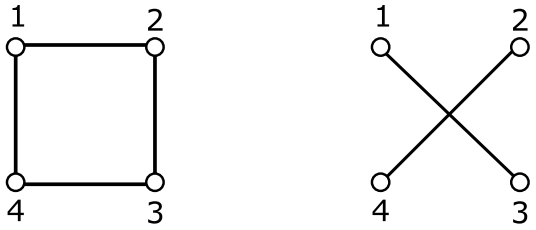

Рис. 33: Автоморфизм в неориентированном графе

Можно увидеть, что любая перестановка из множества дополнении к графу оставит граф неизменным:

$$\left\{\varepsilon, \begin{pmatrix} 1 & 3 \\ 3 & 1 \end{pmatrix}, \begin{pmatrix} 2 & 4 \\ 4 & 2 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{pmatrix}\right\}.$$

Такие *циклические перестановки* можно записать в сокращённом виде:

$$\{\varepsilon, (13), (24), (13)(24)\}.$$

Видно, что все эти перестановки и их комбинации задают всевозможные автоморфизмы оригинального графа.

Алгоритмы обхода графов. Поиск в глубину, поиск в ширину
Часто возникает задача *обхода* вершин и рёбер графа, т.е. посещение всех вершин или рёбер, причём только по одному разу. Такой обход может использоваться, например, в поиске информации на графе. Существуют два широко известных алгоритма обхода вершин графа: поиск в глубину и поиск в ширину. В обоих из них поиск ведётся от какой-то вершины, избранной в качестве начальной.

Поиск в глубину использует в своей работе рекурсию (или стэк в явном виде). Основная идея алгоритма состоит в “погружении” в граф от начальной вершины на максимально возможную глубину, а затем постепенный возврат из рекурсивных вызовов.

С помощью рекурсии алгоритм поиска в глубину можно представить следующим образом:

```
bool visited[N] = {false, ... }; // массив флагов посещённых вершин
edge edges[M];                   // массив рёбер

// рекурсивная процедура поиска
void deep_search(vertex_t v) {
```

```
    visited[v] = true; // отметить вершину как посещённую
    ...                // сделать с вершиной то, ради чего обход затевался

    for(size_t i = 0; i < M; i++) {
        edge e = edges[i];
        if(e.from == v && !visited[e.to]) {
             deep_search(e.to);
        }
    }
}
```

Если стэк использовать явным образом, можно избавиться от рекурсивного вызова:

```
bool visited[N] = {false, ... }; // массив флагов посещённых вершин
edge edges[M];                   // массив рёбер
stack<vertex_t> s;               // стэк

// нерекурсивная процедура поиска
void deep_search(vertex_t v0) {

    s.push(v0);

    while(!s.empty()) {
        vertex_t v = s.pop();

        if(visited[v]) continue; // пропустить посещённую вершину

        visited[v] = true; // отметить вершину как посещённую
        ...                // сделать с вершиной что-то

        for(size_t i = 0; i < M; i++) {
            edge e = edges[i];
            if(e.from == v && !visited[e.to]) {
                s.push(e.to);
            }
        }
    }
}
```

В обоих случаях поиск запускается вызовом процедуры с параметром начальной вершины.

Рассмотрим пример поиска в глубину на неориентированном графе, изображённый не рисунке 34. Остовный подграф, получившийся при обходе вершин обозначен пунктиром. Очевидно, что этот подграф является деревом с корнем в начальной вершине, а его вид будет зависеть от порядка обхода вершин во внутреннем цикле рекурсивной процедуры.

Порядок следования вершин и состояние стэка на каждом шаге:

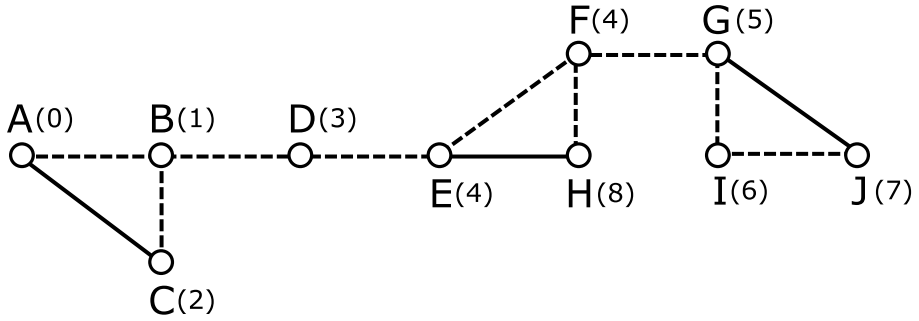

Рис. 34: Поиск в глубину в неориентированном графе

Шаг	**Текущая вершина**	**Стэк в конце шага**
0	∅	A
1	A	B A
2	B	C D B A
3	C	D B A
4	D	E D B A
5	E	F H E B A
6	F	G F H E B A
7	G	I J G F H E B A
8	I	J J I G F H E B A
9	J	I G F H E B A
10	H	E B A
конец	A	∅

Аналогично производится поиск в глубину на ориентированном графе (см. рисунок 35).

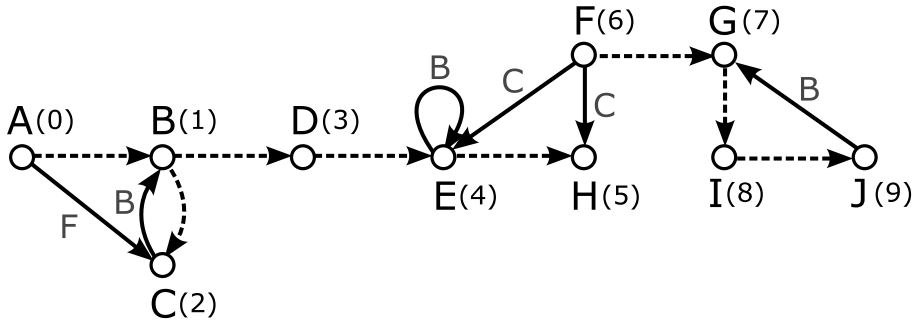

Рис. 35: Поиск в глубину в ориентированном графе

С помощью алгоритма поиска в глубину можно классифицировать дуги ориентированного графа следующим образом:

остовные дуги получаются непосредственно при поиске в глубину, образуют дерево (или в общем случае лес), на рисунке обозначены пунктиром;

прямые дуги соединяют предка и потомка в остовном пути, на рисунке обозначены буквой **F**;

обратные дуги соединяют потомка и предка в остовном пути, на рисунке обозначены буквой **B**, интересно то, что каждая из таких дуг задаёт элементарный контур в графе;

перекрёстные дуги все остальные дуги, на рисунке обозначены буквой **C**.

Вторым широко распространённым алгоритмом является *поиск в ширину*. Это алгоритм использует *очередь* для хранения списка обрабатываемых вершин. В результате работы алгоритма также получается остовное дерево, которое в общем случае отличается от дерева поика в глубину. Можно предложить следующий алгоритм поиска в ширину:

```
bool visited[N] = {false, ... };   // массив флагов посещённых вершин
edge edges[M];                     // массив рёбер
queue<vertex_t> q;                 // очередь

// процедура поиска
void wide_search(vertex_t v0) {

    q.push(v0);

    while(!q.empty()) {
        vertex_t v = q.pop();

        if(visited[v]) continue; // пропустить посещённую вершину
        visited[v] = true;       // отметить вершину как посещённую
        ...                      // сделать с вершиной что-то

        for(size_t i = 0; i < M; i++) {
            edge e = edges[i];
            if(e.from == v && !visited[e.to])
               q.push(e.to);
        }
    }
}
```

Рассмотрим пример поиска в ширину на неориентированном графе, изображённый не рисунке 36. Остовный подграф, получившийся при обходе вершин обозначен пунктиром. Порядок следования вершин и состояние очереди на каждом шаге представлены в таблице:

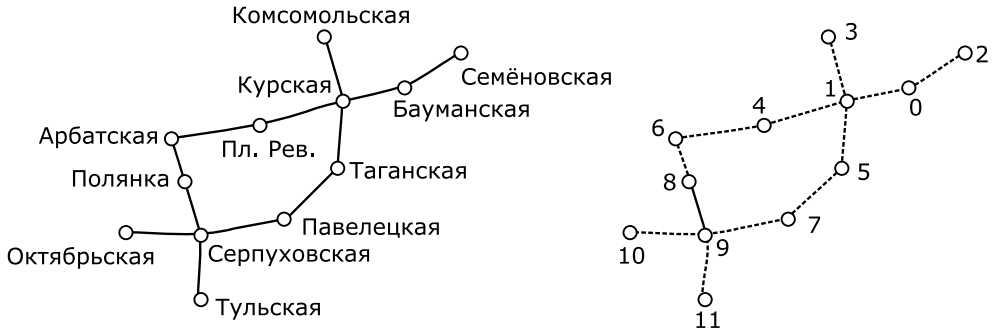

Рис. 36: Поиск в ширину в неориентированном графе

Шаг	**Текущая вершина**	**Очередь в конце шага**
0	∅	Баум.
1	Бауманская	Курск. Семён.
2	Курская	Семён. Комсом. Пл.Рев. Таган.
3	Семёновская	Комсом. Пл.Рев. Таган.
4	Комсомольская	Пл.Рев. Таган.
5	Пл. Революции	Таган. Арбат.
6	Таганская	Арбат. Павел.
7	Арбатская	Павел. Полян.
8	Павелецкая	Полян. Серпух.
9	Полянка	Серпух. Серпух.
10	Серпуховская	Серпух. Октяб. Тульск.
11	Октябрьская	Тульская
12	Тульская	∅

5 Регулярные языки и конечные автоматы

Языки Искусственные языки (например, языки программирования) рассматриваются в особом разделе дискретной математики — *Теории формальных языков*. В рамках этих семинаров языки будут изучаться с *синтаксической* точки зрения (проверка корректности фраз на рассматриваемом языке).

В основе определения языка лежит понятие *алфавита*. Произвольное конечное непустое множество $V = \{a, b, \dots\}$ называется алфавитом, его элементы — *буквами*. Упорядоченный набор букв $a_1 a_2 \dots a_n, a_i \in V$ в данном алфавите называют *словом* (или *цепочка*). *Длина* слова — число букв в нём. По сути, все слова длины n — декартова n-ная степень V^n алфавита. Также по определению вводится понятие пустого слова λ: $V^0 = \{\lambda\}$. Все слова в алфавите V можно обозначить как $V^* = V^0 \cup V^1 \cup V^2 \cup \dots$.

Языком $\mathcal{L}$ в алфавите V называется некоторое подмножество всех слов этого алфавита: $\mathcal{L} \subseteq V^*$.

Рассмотрим пример. Возьмем алфавит $V = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ всех десятичных цифр. В этом случае всего существует одно слово длины

0 (пустое слово λ), 10 слов длины 1 (“0”, “1”, “2” и т.п.), 100 слов длины 2 (“00”, “01”, . . . , “99”) и т.п.. Множество всех слов в этом алфавите — множество всех строк, состоящих из десятичных цифр. Это множество, естественно, бесконечно.

Рассмотрим язык всех двоичных слов: $\mathcal{L}_2 = \{0000, 011, 101111010, \dots\}$. Очевидно, что это множество образует язык в алфавите V, так как $\mathcal{L}_2 \subset V^*$. Кстати говоря, язык всех двоичных *чисел* $\mathcal{L}_B = \{0, 1, 10, \dots 101111010, \dots\}$ является подмножеством языка двоичных слов: $\mathcal{L}_B \subset \mathcal{L}_2 \subset V^*$.

Возьмём на множестве всех слов бинарную операцию *конкатенации* (склеивания) строк:

$$a_1 a_2 \dots a_n \cdot b_1 b_2 \dots b_m = a_1 a_2 \dots a_n b_1 b_2 \dots b_m.$$

Очевидно, что эта операция ассоциативна, но в общем случае некоммутативна. Кроме того, пустое слово λ является нейтральным элементом по этой операции: $a \cdot \lambda = \lambda \cdot a = a$. Можно заключить, что алгебра $(V^*, \cdot, \lambda)$ — моноид.

Операция конкатенации может быть расширена и на два языка:

$$\mathcal{L}_1 \cdot \mathcal{L}_2 = \{c | c = a \cdot b, a \in \mathcal{L}_1, b \in \mathcal{L}_2\}$$

Например, для двух языков в алфавите $V = \{a, b\}$ конкатенация будет равна:

$$\begin{aligned}
&\mathcal{L}_1 = \{ab, aab, a\} \\
&\mathcal{L}_2 = \{b, bb\} \\
&\mathcal{L}_1 \cdot \mathcal{L}_2 = \{abb, aabb, ab, abbb, aabbb\}
\end{aligned}$$

Все свойства конкатенации сохраняются, а нейтральным элементом будет язык $\{\lambda\}$.

Также к языкам можно применять и стандартные теоретико-множественные операции, например, *объединение*. Можно заметить, что множество всех языков в алфавите V с операциями объединения и конкатенации образуют идемпотентное полукольцо:

$$\mathcal{L}(2^{V^*}, \cup, \cdot, \varnothing, \{\lambda\}).$$

Не трудно убедиться, что все аксиомы идемпотентного полукольца при этом выполняются.

В этом полукольце также существует операция *итерации* — бесконечного объединения степеней языков:

$$\mathcal{L}^* = \mathcal{L}^0 \cup \mathcal{L}^1 \cup \mathcal{L}^2 \cup \mathcal{L}^3 \dots,$$

где степень n языка определяется как n-кратная конкатенация этого языка с самим собой, а 0-ая степень по определению язык $\{\lambda\}$.

Например, для языка $\mathcal{L} = \{ab, b\}$ в алфавите $V = \{a, b\}$ итерация будет равна языку:

$$\mathcal{L}^* = \mathcal{L}^0 \cup \mathcal{L}^1 \cup \mathcal{L}^2 \cup \mathcal{L}^3 \cdots =$$
$$= \{\lambda\} \cup \{ab, b\} \cup \{ab, b\} \cdot \{ab, b\} \cup \{ab, b\} \cdot \{ab, b\} \cdot \{ab, b\} \cdots =$$
$$= \{\lambda, ab, b\} \cup \{abab, abb, bab, bb\} \cup \{abab, abb, bab, bb\} \cdot \{ab, b\} \cup \cdots =$$
$$\{\lambda, b, ab, bb, abb, bab, bbb, abab, abbb, babb, ababab, \dots\}$$

Обычно итерация языка обозначают кратко: $\{ab, b\}^*$.

Также введем понятие *позитивной итерации*:

$$\mathcal{L}^+ = \mathcal{L}^1 \cup \mathcal{L}^2 \cup \mathcal{L}^3 \dots,$$

Регулярные языки и выражения *Регулярным языком* в алфавите $V = \{a, b, c, \dots\}$ называется *замыкание* элементарных языков $[\{a\}, \{b\}, \{c\}, \dots]_{\{\cup, \cdot, \ ^*\}}$ по операциям объединения, конкатенации и итерации.

Другими словами, регулярные языки — это языки, которые могут быть получены из букв алфавита с помощью последовательных применений операций объединения, конкатенации или итерации, и никакие другие.

Существует множество регулярных языков. Например, язык двоичных слов, приведённый выше или язык всех адресов электронной почты в алфавите $\{a, b, \dots, z, -, @\}$.

Примером нерегулярного языка является $\mathcal{L} = \{a^n b^n | n \in \mathbb{N}\} = \{ab, aabb, aaabbb, \dots\}$ в алфавите $V = \{a, b\}$. Очень важное условие — равенство числа букв a и b нарушает регулярность языка. Тогда как язык $\{a^n b^m | n, m \in \mathbb{N}\}$ конечно является регулярным, так как может быть получен как $\{a\}^+ \cdot \{b\}^+$.

Для более удобного представления регулярных языков можно использовать *регулярные выражения*. В регулярных выражениях могут фигурировать буквы алфавита, пустое слово λ, бинарные операции объединения и конкатенации и унарная операция итерации, так что регулярному выражению может быть однозначно поставлен в соответствие регулярный язык (обратное неверно). При этом достаточно заменить следующие элементы регулярного выражения:

$$\varnothing \to \varnothing$$
$$\forall a \in V : a \to \{a\}$$
$$\lambda \to \{\lambda\}$$
$$\alpha \to P, \beta \to Q \Rightarrow \begin{cases} (\alpha + \beta) \to P \cup Q \\ (\alpha \cdot \beta) \to P \cdot Q \\ \alpha^* \to P^*, \alpha^+ \to P^+ \end{cases}$$

В регулярных выражениях по определению используется следующие приоритеты операций (по убыванию приоритета): * и $^+$, $\cdot$, $+$.

Рассмотрим примеры на связь между регулярными языками и регулярными выражениями:

- $(a+bc)^* \to \{a, bc\}^* = \{\{a, bc\}^n | n \in \mathbb{N}_0\}$, примерами слов этого языка являются *abcaaa* или *bca*, тогда как слово *bacbc* не принадлежит данному языку.

- $ab^*+b^+(aa)^* \to \{a\}\{b\}^* \cup \{b\}^+\{aa\}^* = \{ab^n, b^m(aa)^k | n,k \geqslant 0, m > 0\}$, примерами слов этого языка являются $abbb$ или bb, тогда как слово aba не принадлежит данному языку.

- рассмотрим алфавит $V = \{a, b, \ldots, z, @, ., -\}$ символов латиницы и некоторых служебных символов. Регулярное выражение $(a+b+\cdots+z)^+@(a+b+\cdots+z)^+.(a+b+\cdots+z)^+$ задаёт в общем виде язык возможных адресов электронной почты, например: $hkhfskdjfh@qewq.ert$ или $aleksey@fedoseev.net$.

- в алфавите десятичных цифр $V = \{0,1,2,3,4,5,6,7,8,9\}$ регулярное выражение $(0+1)^*$ задаёт язык двоичных слов, рассмотренный выше.

Один язык может задаваться множеством регулярных выражений, например, язык $\{\{a,b\}^n ca^m | n \geqslant 0, m > 0\}$ — выражением $(a+b)^* ca^+$ или $(a^*b^*)^* caa^*$.

На практике регулярные выражения часто используются для поиска известных последовательностей символов в тексте и разбиении строк регулярного языка на составные части. См. команду grep в Unix или regular expressions в современных библиотеках и языках программирования.

Конечные автоматы Отдельный раздел математики рассматривает *автоматы* — абстрактные машины, выполняющие определённые действия на основании входных данных. В этом курсе рассматриваются *конечные автоматы*. Другими известными автоматами являются *автоматы с магазинной памятью*, используемые в теории компиляторов, а также универсальная *машина Тьюринга*.

Конечный автомат представляет собой абстрактное устройство (см. рисунок 37), состоящее из:

- входной ленты с символами из алфавита V;

- считывающего устройства;

- блока управления, который может находиться в одном из состояний.

Считывающее устройство может двигаться вдоль ленты в одном направлении, тем самым проходя всю цепочку символов на ленте. Работа автомата состоит из дискретных *шагов*, на каждом шаге устройство управления находится в определённом состоянии. При переходе на следующий шаг считывающее устройство может продвинуться на один символ, а также может измениться состояние блока управления (или, другими словами, состояние самого автомата). Блок управления содержит список правил переходов, согласно которому его состояние изменяется с учётом входных символов.

Формально, конечный автомат — это пятёрка вида

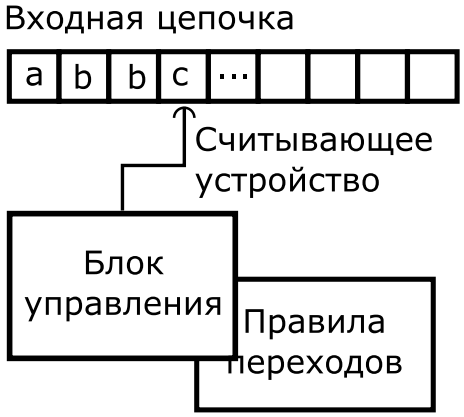

Рис. 37: Схема конечного автомата

$$\mathcal{F}(V, Q, q_0, Q_F, \delta),$$

где V — алфавит входных символов, Q — конечное множество состояний, $q_0 \in Q$ — начальное состояние автомата, $Q_F \subseteq Q$ — множество конечных состояний автомата, δ —множество правил переходов.

Каждое правило переходов имеет вид $pa \to q$, где $p, q \in Q$ — некоторые состояния, которые могут совпадать, $a \in V \cup \{\lambda\}$ — входной символ или пустое слово. Правило имеет следующий смысл:

- если $a \neq \lambda$, то это правило применимо, только если на данном шаге блок управления находится в состоянии p, а считывающее устройство наблюдает символ a на входной цепочке. Результатом действия этого правила является изменение состояния автомата на q и сдвиг считывающего устройства на один символ;

- если $a = \lambda$, то это правило применимо, только если на данном шаге блок управления находится в состоянии p, символ на ленте при этом не анализируется (это возможно также, если считывающее устройство дошло до конца ленты). Результатом действия этого правила является изменение состояния автомата на q.

Таким образом, автомат меняет своё состояние от начального к одному из конечных, последовательно читая символы на ленте. Говорят, что цепочка входных символов *допускается* конечным автоматом, если в результате работы автомата она прочитывается целиком, а автомат после этого находится в одном из конечных состояний. В противном случае, говорят, что цепочка *не допускается* данным автоматом.

Для отображения конечного автомата часто используют ориентированный граф, вершины которого соответствуют состояниям автомата, а дуги — правилам переходов. При этом дуга из вершины q_1 в вершину q_2 имеет метку a тогда и только тогда, когда автомат содержит правило перехода $q_1 a \to q_2$, т.е. метками дуг могут быть символы входного алфавита или символ λ.

На рисунке 38 показан автомат $\mathcal{F}_1$, который формально определяется как пятёрка $(\{a,b\},\{q_1,q_2,q_3,q_4\},q_1,\{q_4\},\delta)$ со следующими правилами переходов δ:

$$
\begin{array}{ll}
q_1 a \to q_2 & (1)\\
q_1 b \to q_2 & (2)\\
q_2 b \to q_3 & (3)\\
q_2 b \to q_4 & (4)\\
q_4 \lambda \to q_2 & (5)\\
q_4 a \to q_3 & (6)\\
q_4 a \to q_4 & (7)
\end{array}
$$

Начальные и конечные вершины отображаются специальным образом (с входящими и выходящими стрелками) — это всего лишь способ обозначения вершин, а не нарушение определения графа, согласно которому не бывает дуг, не связывающих вершины.

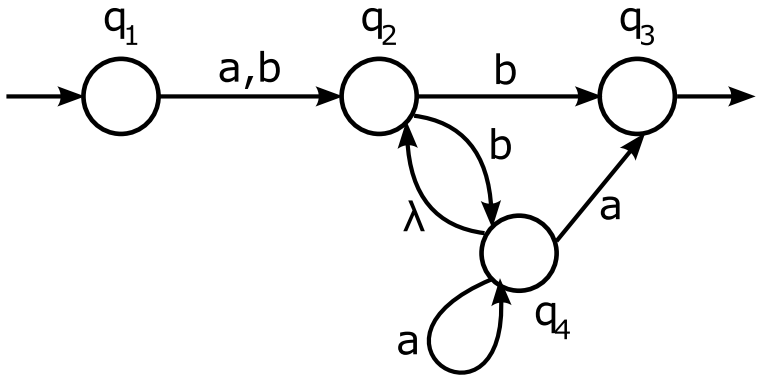

Рис. 38: Конечный автомат, в виде ориентированного графа

Рассмотрим пример работы этого автомата. Пусть на вход подаётся цепочка $abbaa$, последовательность шагов представлена в таблице:

Шаг	**Состояние**	**Входная цепочка**	**Правило**
1	q_1	$abbaa$	1
2	q_2	$bbaa$	4
3	q_4	baa	5
4	q_2	baa	4
5	q_4	aa	7
6	q_4	a	6
7	q_3	λ	конец

Входная цепочка *допускается* конечным автоматом, но это видно лишь при такой последовательности шагов. Однако, в ходе работы встретилось несколько альтернатив, связанных с наличием λ-переходов и правил с одинаковыми левыми частями. Т.е. блок управления должен быть снабжён дополнительным устройством выбора в таких спорных ситуациях (назовём его “чёртиком”), чтобы работа автомата не зашла в тупик.

Рассмотрим пример другой цепочки — aab:

Шаг	**Состояние**	**Входная цепочка**	**Правило**
1	q_1	aab	1
2	q_2	ab	ошибка

Эта цепочка не допускается данным конечным автоматом.

Задание для самоподготовки. Для заданного конечного автомата $\mathcal{F}(\{a,b\},\{q_1,q_2,q_3,q_4,q_5\},q_1,\{q_5\},\delta)$, где δ:

$$\begin{gathered} q_1a \to q_2 \\ q_2b \to q_3 \\ q_3a \to q_3 \\ q_3b \to q_3 \\ q_3b \to q_4 \\ q_4a \to q_5 \\ q_5\lambda \to q_1, \end{gathered}$$

проверить допустимость цепочек $abba$, $abbbaabaaba$ и $abbbab$. Можно ли выписать регулярное выражение, задающее все допускаемые данным автоматом цепочки?

Теорема Клини На самом деле, между регулярными языками и конечными автоматами существует непосредственная связь. В теореме Клини доказывается, что язык допускается некоторым конечным автоматом тогда и только тогда, когда он регулярный. Таким образом, мы можем для каждого регулярного языка построить конечный автомат, который будет проверять принадлежность цепочек данному языку, и, обратно, для любого конечного автомата можно получить язык допускаемых слов, и этот язык будет регулярным.

Рассмотрим первую задачу — получение конечного автомата по известному регулярному языку (или выражению). Мы можем выписать *элементарные* автоматы, которые допускают цепочки из одинарных букв алфавита и расширить их автоматами для операций (см. рисунок 39).

Рассмотрим пример: регулярный язык $\mathcal{L}'$ задан регулярным выражением $ab(a+b)^*b^+$. Соответствующий автомат представлен на рисунке 40.

Для получения регулярного языка, допускаемого конечным автоматом, в общем случае необходимо решить систему уравнений в полукольце языков (аналогично решалась задача поиска кратчайших расстояний в графе): $\mathcal{L}(2^{V^*},\cup,\cdot,\varnothing,\{\lambda\})$. Не трудно убедиться, что это идемпотентное полукольцо с итерацией, в котором мы можем решать уравнение вида $x = ax + b$ (см. выше). Чтобы получить систему уравнений, необходимо построить матрицу переходов для графа конечного автомата.

Рассмотрим пример, в котором получить регулярное выражение без составления конечного автомата и системы уравнений достаточно сложно. Рассмотрим алфавит $\{0,1\}$, необходимо получить регулярное

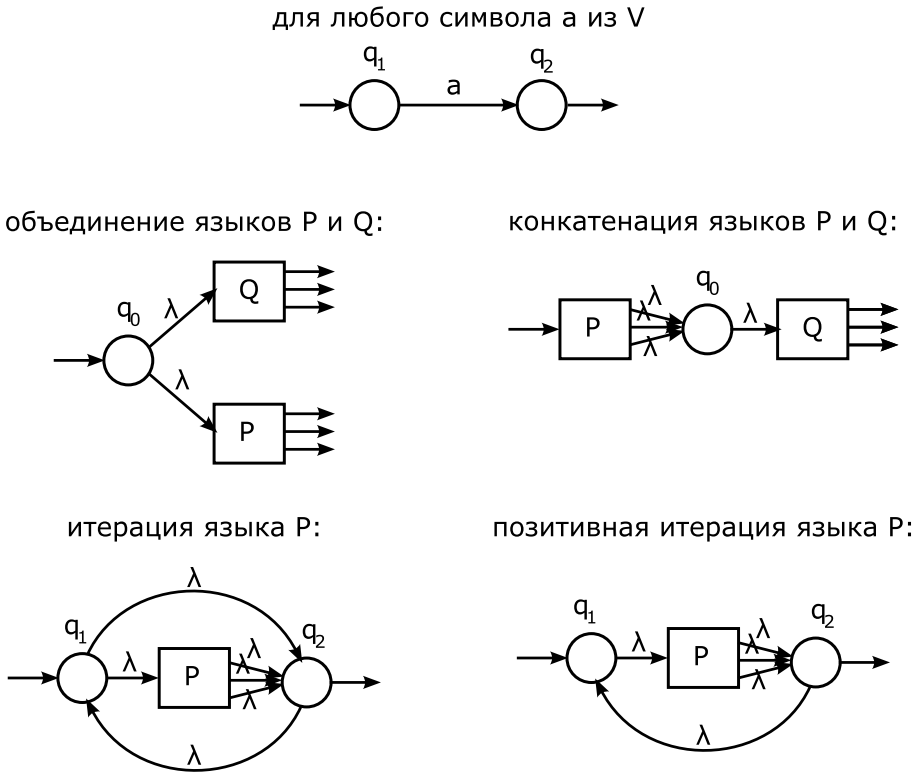

Рис. 39: Получение конечного автомата по регулярному языку

выражения для языка, содержащего только чётное число единиц и чётное число нулей. Попытки придумать регулярное выражение “в лоб” (например, $((00)^* + (11)^*)^*$) дадут только частные случаи. Построим конечный автомат, допускающий цепочки такого языка: очевидно, что возможно только четыре состояния: “чётное число нулей и единиц”, “чётное число нулей, нечётное — единиц”, “нечётное число нулей и единиц” и “нечётное число нулей и чётное число единиц”, автомат со всеми возможными переходами показан на рисунке 41.

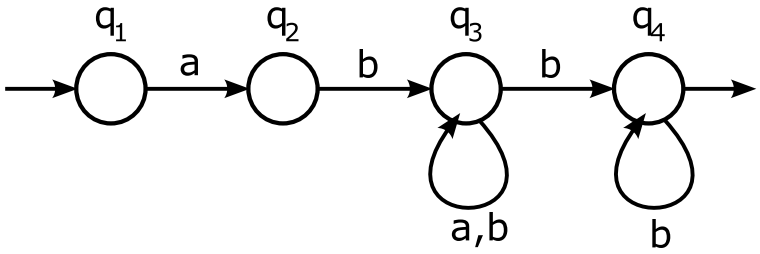

Рис. 40: Конечный автомат, соответствующий языку $\mathcal{L}'$

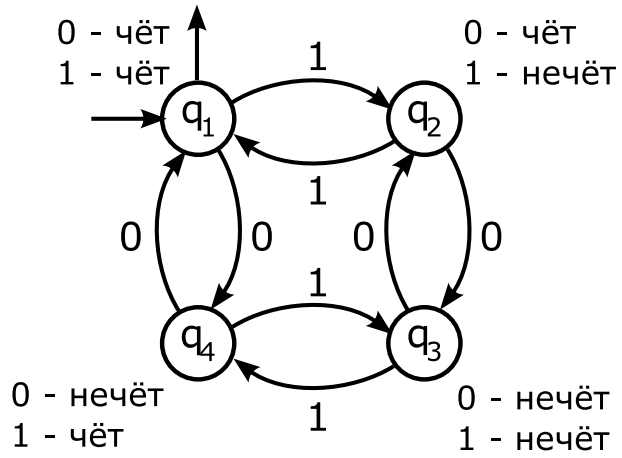

Рис. 41: Конечный автомат, допускающий цепочки с чётным числом нулей и единиц

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

Каждому состоянию q_i конечного автомата ставится в соответствие переменная x_i. Значение этой переменной суть есть регулярное выражение, задающее все слова, которые допускаются конечным автоматом, стартующем из данного состояния. Очевидно, что язык, допускаемый конечным автоматом получается сложением переменных всех стартовых состояний.

Каждой строке матрицы соответствует одно уравнение, при этом всем уравнениям, соответствующим конечным состояниям автомата, необходимо прибавить λ.

В нашем примере получается следующая система уравнений, которая имеет решение:

$$\begin{cases} x_1 = 1 \cdot x_2 + 0 \cdot x_4 + \lambda \\ x_2 = 1 \cdot x_1 + 0 \cdot x_3 \\ x_3 = 0 \cdot x_2 + 1 \cdot x_4 \\ x_4 = 0 \cdot x_1 + 1 \cdot x_3 \end{cases} \qquad \begin{cases} x_1 = 1 \cdot x_2 + 0 \cdot (0 \cdot x_1 + 1 \cdot x_3) + \lambda \\ x_2 = 1 \cdot x_1 + 0 \cdot x_3 \\ x_3 = 0 \cdot x_2 + 1 \cdot (0 \cdot x_1 + 1 \cdot x_3) \end{cases}$$

$$\begin{cases} x_1 = 00 \cdot x_1 + 1 \cdot x_2 + 01 \cdot x_3 + \lambda \\ x_2 = 1 \cdot x_1 + 0 \cdot x_3 \\ x_3 = 11 \cdot x_3 + 0 \cdot x_2 + 10 \cdot x_1 \end{cases} \qquad \begin{cases} x_3 = (11)^*(0 \cdot x_2 + 10 \cdot x_1) \\ x_1 = 00 \cdot x_1 + 1 \cdot x_2 + 01 \cdot x_3 + \lambda \\ x_2 = 1 \cdot x_1 + 0 \cdot x_3 \end{cases}$$

$$\begin{cases} x_1 = 00 \cdot x_1 + 1 \cdot x_2 + 01(11)^*(0 \cdot x_2 + 10 \cdot x_1) + \lambda \\ x_2 = 1 \cdot x_1 + 0(11)^*(0 \cdot x_2 + 10 \cdot x_1) \end{cases}$$

$$\begin{cases} x_1 = 00 \cdot x_1 + 1 \cdot x_2 + 01(11)^*0 \cdot x_2 + 01(11)^*10 \cdot x_1 + \lambda \\ x_2 = 0(11)^*0 \cdot x_2 + (1 + 0(11)^*10) \cdot x_1 \end{cases}$$

$$\begin{cases} x_2 = (0(11)^*0)^*(1 + 0(11)^*10) \cdot x_1 \\ x_1 = (00 + 01(11)^*10) \cdot x_1 + (1 + 01(11)^*0) \cdot x_2 + \lambda \end{cases}$$

$$x_1 = (00 + 01(11)^*10) \cdot x_1 + (1 + 01(11)^*0)(0(11)^*0)^*(1 + 0(11)^*10) \cdot x_1 + \lambda$$

$$x_1 = (00 + 01(11)^*10 + (1 + 01(11)^*0)(0(11)^*0)^*(1 + 0(11)^*10)) \cdot x_1 + \lambda$$

$$x_1 = (00 + 01(11)^*10 + (1 + 01(11)^*0)(0(11)^*0)^*(1 + 0(11)^*10))$$

Таким образом, регулярное выражение для языка, допускаемого автоматом, равно:

$$00 + 01(11)^*10 + (1 + 01(11)^*0)(0(11)^*0)^*(1 + 0(11)^*10),$$

а сам язык:

$$L = \{00, 01(11)^k10, \{1, 01(11)^l\}(0(11)^m0)^n\{1, 0(11)^o10\} | k, l, m, n, o \in \mathbb{N}_0\}.$$

Детерминизация конечных автоматов Выше уже было сказано о том, что при работе конечного автомата возможны "конфликтные ситуации", когда существует два правила с одинаковой левой частью или λ-переходы — в этом случае необходимо "знать заранее", какой путь следует выбрать, чтобы цепочка всё же была прочитана. Такой подход очень неудобен в реализации, поэтому возникает задача *детерминизации* конечного автомата, т.е. приведение его к такому виду, что "конфликтные ситуации не возможны", а язык, допускаемый автоматом, не изменяется.

Существует формальный алгоритм детерминизации автомата, который состоит из двух шагов:

- удаление λ-переходов;
- собственно детерминизация.

Для данного автомата $M = (V, Q, q_0, F, \delta)$, автомат без λ-переходов будет иметь вид:

$$M' = (V, Q', q_0, F', \delta'),$$

где множество вершин $Q' = \{q_0\} \cup \{q | \exists r)(r \to q \ \ \& \ \ \mu(r,q) \neq \lambda\}$, т.е. начальная вершина плюс все вершины, в которые входит хотя бы одна не-λ-дуга; множество конечных вершин $F' = (F \cap Q') \cup \{q | q \in Q' \ \ \& \ \ q \Rightarrow_\lambda^+ q_f \in F\}$, т.е. множество оставшихся конечных вершин плюс все вершины, достижимые из конечных по λ-путям (одна или набор λ-дуг); множество правил перехода δ' — оставляются только не-λ-переходы, к ним добавляются переходы, следующими за λ-путями из данной вершины (т.е. для $\forall p, q, r : p \Rightarrow_\lambda^+ q \to_a r, a \in V$ в δ' добавляется правило $pa \to r$).

Рассмотрим пример: на рисунке 42 показаны автоматы с λ-переходами и без — при этом они допускают один и тот же язык.

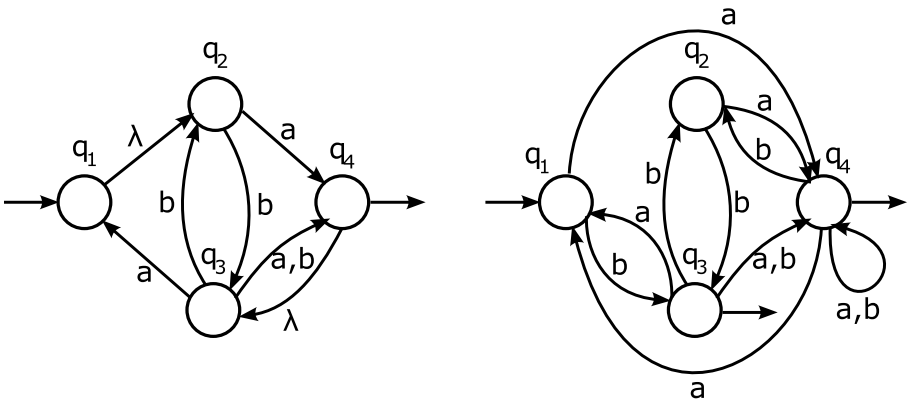

Рис. 42: Удаление λ-переходов

Детерминизация производится построением новых вершин и переходов. При этом в качестве вершин графа автомата могут теперь выступать не отдельные состояния изначального автомата, а*наборы* состояний, например $\{q_0, q_2, q_3\}$.

В качестве нового исходного состояния принимается состояние $\{q_0\}$. Далее для всех букв алфавита строятся новые вершины, в которые можно попасть из данной вершины. Для нашего примера все новые состояния и переходы будут выглядеть так:

$$\delta'(\{q_1\}, a) = \{q_4\}$$
$$\delta'(\{q_1\}, b) = \{q_3\}$$
$$\delta'(\{q_3\}, a) = \{q_1, q_4\}$$
$$\delta'(\{q_3\}, b) = \{q_2, q_4\}$$
$$\delta'(\{q_4\}, a) = \{q_1, q_4\}$$
$$\delta'(\{q_4\}, b) = \{q_2, q_4\}$$
$$\delta'(\{q_1, q_4\}, a) = \{q_1, q_4\}$$
$$\delta'(\{q_1, q_4\}, b) = \{q_2, q_3, q_4\}$$
$$\delta'(\{q_2, q_4\}, a) = \{q_1, q_4\}$$
$$\delta'(\{q_2, q_4\}, b) = \{q_2, q_3, q_4\}$$
$$\delta'(\{q_2, q_3, q_4\}, a) = \{q_1, q_4\}$$
$$\delta'(\{q_2, q_3, q_4\}, b) = \{q_2, q_3, q_4\}$$

Новыми конечными вершинами являются все вершины, содержащие в себе хоть одну из конечных вершин исходного автомата: $\{q_4\}$,

$\{q_1, q_4\}$, $\{q_2, q_4\}$, $\{q_2, q_3, q_4\}$.

Новые вершины можно переименовать. Граф детерминизированного автомата показан на рисунке 43.

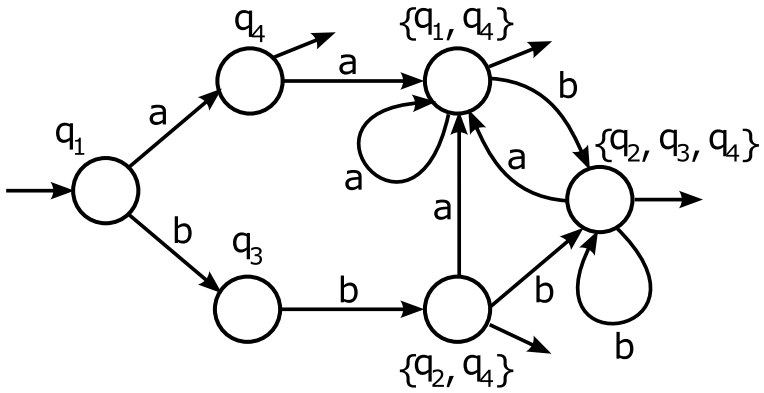

Рис. 43: Детерминизированный автомат

Существует теорема, согласно которой для каждого автомата может быть построен эквивалентный (допускающий тот же язык) автомат. Это позволяет для любого регулярного языка построить детерминизированный автомат.

Из этой теоремы есть интересное следствие — с помощью детерминизированного автомата можно получать автоматы *не допускающие* заданную цепочку. Для этого необходимо в детерминизированном автомате, допускающем данную цепочку, взять в качестве конечных вершин дополнение относительно всех вершин: $F' = Q \setminus F$.

Рассмотрим пример — необходимо получить автомат (в алфавите $\{a, b\}$), не допускающий идущих подряд букв ab. На рисунке 44 показан исходный автомат, допускающий все слова, содержащие подряд ab, затем тот же автомат, но детерминизированный. Третьим показан автомат, не досускающий цепочки, содержащие подряд ab. В данном случае две правые вершины в последнем автомате можно объединить как *неверные состояния*, в том смысле, что попав туда, автомат заведомо не прочитает заданную цепочку.

Помимо отрицания языка (язык — множество слов, а значит и все множественные операции к нему применимы), благодаря теореме о детерминизации, можно заключить следующее про регулярные языки:

$\overline{L} = V^* \setminus L$ — дополнение к языку является регулярным, $L_{or} = L_1 \cup L_2$ — объединение двух регулярных языков по определению регулярно. А значит, $L_{and} = L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ пересечение двух регулярных языков является регулярным языком.

Лемма о разрастании Для регулярных языков существует интересная *лемма*:

Если $L \subseteq V^*$ — регулярный язык, то $\exists k_L \in \mathbb{N} : \forall x \in L \;\; \&$

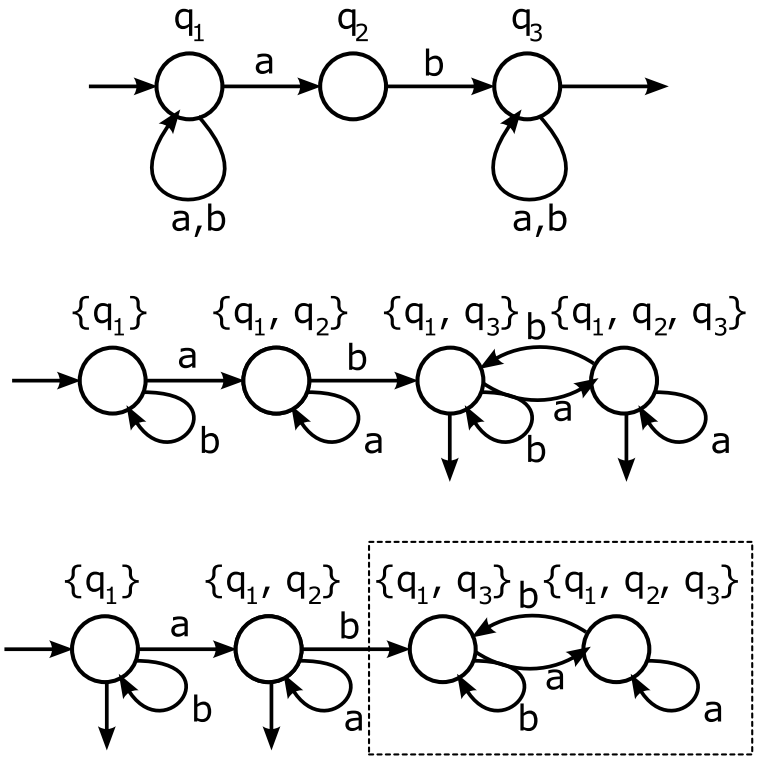

Рис. 44: Детерминизация и дополнение конечного автомата

$|x| \geqslant k_L x = uvw, \text{где} |v| > 0 \ \ \& \ \ |v| \leqslant k_L \ \ \& \ \ (\forall n \geqslant 0)(uv^n w \in L)$.

Другими словами, в любом регулярном языке можно найти такую непустую последовательность символов, что её исключение или повторение любое число раз оставит новое слово в данном языке. Мы можем "накачивать" часть слова, и оставаться в рамках языка.

Рассмотрим пример. В регулярном языке, задаваемом выражением $a^*(a+b)^+$ можно найти такое разбиение цепочки $x = a^n(a^m b^k)^l : u = \lambda, v = a, w = (a^m b^k)^l$. "Накачивая" цепочку v мы остаёмся в рамках языка.

Однако, эта лемма часто применяется для доказательства нерегулярности языков. Если удаётся показать, что она не выполняется для любой цепочки x в языке L, то можно заключить, что этот язык не регулярный.

Например, язык $L = \{a^n b^n | n \in \mathbb{N}\}$. Слова в этом языке имеют вид $aaa \ldots aabbb \ldots bb$, с одинаковым числом символов a и b. Мы можем разбить слово следующим образом:

- $u = \lambda, v = a^n b^n, w = \lambda$, но в этом случае, накачивая v, слово $(a^n b^n)(a^n b^n) \ldots (a^n b^n)$ уже не будет принадлежать нашему языку. По аналогии, можно отвергнуть все разбиения, в которых v содержит как символы a, так и символы b;

- $u = \lambda, v = a^n, w = b^n$, но тогда слова вида $a^{2n}b^n$, $a^{3n}b^n$ и т.п. уже не будут принадлежать языку, так как число символов a и b.

Аналогично можно рассмотреть все остальные случаи. Очевидно, что лемма не выполняется, а это значит, что язык этот нерегулярный.

Для доказательства нерегулярности других языков иногда требуется использовать свойства регулярных языков. Например, требуется доказать нерегулярность языка всех двойных слов $L_d = \{yy | y \in V^*\}$. Формулировка языка не позволяет напрямую применить лемму о разрастании (в y может попадать что угодно).

В данном случае можно воспользоваться тем, что пересечение регулярных языков регулярно. Допустим, язык L_d регулярен, тогда его пересечение с языком, задаваемым выражением: a^*ba^*b, равное $L_d \cup \{a^nba^mb | n, m \in \mathbb{N}_0\} = \{a^kba^kb | k \in \mathbb{N}_0\}$ должно быть регулярным, а уже регулярность последнего можно легко опровергнуть леммой о разрастании — также как и в предыдущем примере.

О грамматиках Мы рассматривали языки как множество слов (собственно определение языка), как способ простого описания этого множества (регулярные выражения) и как средство по их распознаванию (конечные автоматы). Все эти описания в определенном смысле эквивалентны.

Существует еще один способ задания языков — с точки зрения генерации всех слов языка (т.е. способ, обратный распознаванию и автоматам). При этом вводится понятие *грамматики*:

$$\mathcal{G} = (V, N, S, P),$$

где V — алфавит символов языка (называемый также *терминальным алфавитом*), N — алфавит служебных символов (*нетерминальный алфавит*) — он не пересекается с терминальным алфавитом $V \cap N = \varnothing$, $S \in N$ — аксиома и P — конечное множество правил вывода вида

$$P : \alpha \to \beta,$$

где $\alpha \in (V \cup N)^* N (V \cup N)^*$, т.е. множество слов из символов обоих алфавитов, но содержащее по крайней мере один нетерминальный символ, $\beta \in (V \cup N)^*$. Набор правил вида $\alpha \to \beta_1, \alpha \to \beta_2, \ldots$ обычно сокращают как $\alpha \to \beta_1 | beta_2 | \ldots$.

Каждое слово языка получается из последовательного применения правил, начиная от аксиомы. В этом случае говорят, что язык *порождается* грамматикой.

Рассмотрим пример грамматики арифметических выражений с одной переменной:

$$\mathcal{G}_0 = (\{x, (,), +, *\}, \{E, T, F\}, E, P), P :$$

$$E \to E + T \,|\, T$$
$$T \to T * F \,|\, F$$
$$F \to (E) \,|\, x$$

Используя данную грамматику мы можем построить например такое выражение, применяя правило к самому левому нетерминалу:

$$E \vdash E + T \vdash E + T + T \vdash T + T \vdash T * F + T \vdash F * F + T \vdash (E) * F + T \vdash$$
$$\vdash (E + T) * F + T \vdash (T + T) * F + T \vdash (F + T) * F + T \vdash (x + T) * F + T \vdash$$
$$\vdash (x + F) * F + T \vdash (x + x) * F + T \vdash (x + x) * x + T \vdash (x + x) * x + F \vdash$$
$$\vdash (x + x) * x + x$$

Видно, что в нетерминальные символы вкладывается определённый синтаксический смысл. Например, в грамматике языка программирования нетерминальными символами являются понятия “выражение”, “оператор”, “условие” и т.п.

Существует классификация грамматик — в зависимости от того, какой вид имет правила вывода, языки принадлежат к разным классам. Например, если во всех правилах вывода $\alpha \to \beta$ цепочки $\alpha \in N$, а $\beta \in (V \cup N)^*$, то такие грамматики называются *контекстно-свободными* (это большинство искусственных языков). Нас больше всего интересует случай, когда правила имеют вид:

$$A \to aB,$$
$$A \to a,$$
$$A \to B,$$
$$A \to \lambda,$$

где $A, B \in N$, а $a \in V$. Такие грамматики называют *регулярными*.

Существует теорема о том, что для любого регулярного языка можно построить не только конечный автомат, который допускает данный язык, но и регулярную грамматику, которая порождает данный язык.

Чтобы получить регулярную грамматику по конечному автомату необходимо:

- в качестве терминального алфавита взять входной алфавит V;
- множество нетерминальных символов будет содержать состояния конечного автомата $N = \{q_1, q_2, \ldots, q_n\}$;
- аксиома совпадает с начальным состоянием автомата;
- правила вывода могут быть получены из правил переходов следующим образом:
  - если $qa \to r \in \delta$, т.е. существует переход по символу a, то необходимо добавить правило вывода $q \to ar$;

– если и только если $qa \to q_f \in \delta$, где $q_f \in F$ — оно из завершающих состояний, то добавляется правило вывода $q \to a$.

Обратный процесс аналогичен.

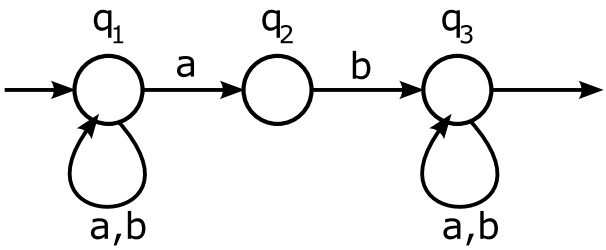

Рис. 45: Пример конечного автомата

Например, для конечного автомата, изображённого на рисунке, грамматика будет иметь вид:

$$\mathcal{G} = (\{a, b\}, \{q_1, q_2, q_3\}, q_1, P), P :$$

$$\begin{array}{l} q_1 \to aq_1|bq_1|aq_2 \\ q_2 \to bq_3|b \\ q_3 \to aq_3|bq_3|a|b \end{array}$$

Минимизация автоматов с выходом Помимо конечных автоматов, что мы рассматривали ранее, существуют *автоматы с выходом*, т.е. автоматы, преобразующие слово входного языка в слово выходного языка. Такие автоматы обычно применяются при кодировании/декодировании информации.

Будем задавать конечный автомат с выходом следующей пятёркой:

$$\mathcal{F}_o(A, B, Q, q_0, f, g),$$

где A — входной алфавит, B — выходной алфавит (может в частном случае совпадать со входным), Q — множество состояний автомата, $q_0 \in Q$ — начальное состояние, $f : Q \times A \to Q$ — функция переходов, аналогичная δ в определении конечного автомата, $g : Q \times A \to B$ — функция выходов — какой символ будет получен на выход на каждом шаге работы автомата. Конечным состоянием в этом случае мы полагаем любое из состояний автомата. Также, мы будем рассматривать только детерминированные автоматы с выходом.

Рассмотрим пример. Пусть необходимо построить автомат, заменяющий каждую третью единицу в двоичном слове на ноль. В виде графа он может быть представлен следующим образом:

Работа автомата на примере цепочки “10100110” показана в таблице:

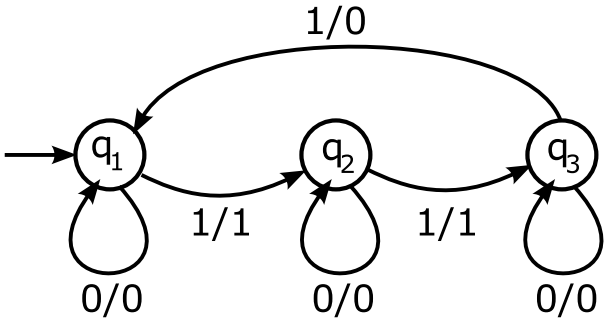

Рис. 46: Конечный автомат с выходом, заменяющий каждую третью единицу

Шаг	**Состояние**	**Входная цепочка**	**Выходная цепочка**
1	q_1	10100110	λ
2	q_2	0100110	1
3	q_2	100110	10
4	q_3	00110	101
5	q_3	0110	1010
6	q_3	110	10100
7	q_1	10	101000
8	q_2	0	1010001
9	q_2	λ	10100010

Для таких автоматов существует задача *минимизации* — получения эквивалентного (дающего тот же выход по заданному входу) автомата с меньшим числом состояний. Существует алгоритм минимизации автомата с выходом.

В основе этого алгоритма лежит понятие *эквивалентных состояний*. Два состояния считаются эквивалентными, если они генерируют один и тот же выход по заданному входу. Таким образом, можно разбить всё множество состояний на классы эквивалентности.

Алгоритм минимизации состоит из следующих шагов:

1. удаление недостижимых вершин, т.е. всех таких вершин, в которые нельзя попасть из начальной при любой входной цепочке;

2. разбиение множества всех вершин на начальные классы эквивалентности I_1, I_2, I_3, I_4 (очевидно, что их может быть не более четырёх);

3. для каждого состояния и входного символа получить класс эквивалентности, в одно из состояний которого мы попадаем по данному переходу. Эти результаты можно объединить в таблице:

	q_1	q_2	$\ldots$	q_n
a_1	I_k	I_j	$\ldots$	$\ldots$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
a_m	I_l	I_p	$\ldots$	$\ldots$

4. на основании полученных переходов разбиваем существующие классы эквивалентности I_j, так, чтобы все состояния, входящие в один класс, переходили в одни и те же классы эквивалентности по заданному входу. Другими словами, выделяем в полученной таблице одинаковые столбцы и группируем согласно ним состояния автомата — если эти группы пересекаются с классами эквивалентности, то их необходимо разбить;

5. повторить два предыдущих этапа пока на очередной итерации не получится набор классов эквивалентности, аналогичный предыдущему шагу;

6. заменить каждый из классов эквивалентности новым состоянием.

Рассмотрим пример. На рисунке 47 показан автомат с выходом.

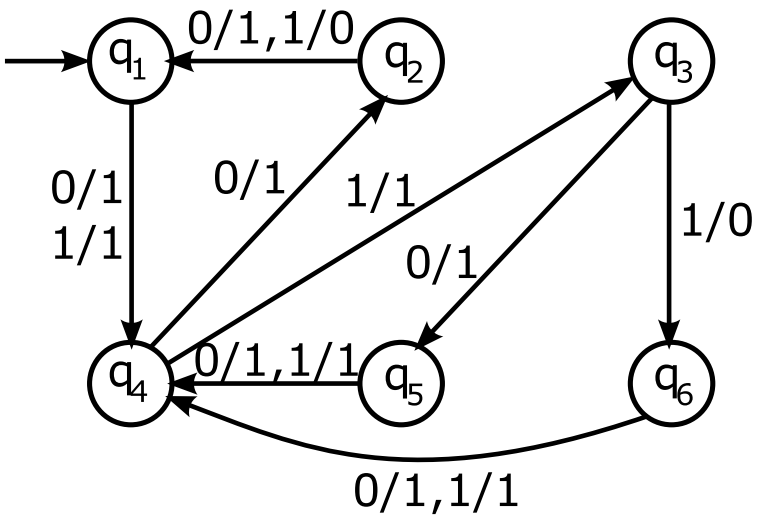

Рис. 47: Пример конечного автомата с выходом

Недостижимых вершин в этом автомате нет, так что переходим ко второму этапу. Можно выделить два класса эквивалентности: $I_1 = \{q_1, q_4, q_5, q_6\}$ и $I_2 = \{q_2, q_3\}$.

Далее строим таблицу переходов:

	q_1	q_2	q_3	q_4	q_5	q_6
0	I_1	I_1	I_1	I_2	I_1	I_1
1	I_1	I_1	I_1	I_2	I_1	I_1

Видно, что q_4 должно быть отделено в свой собственный класс эквивалентности. Класс эквивалентности I_2 не меняется: $I_1 = \{q_1, q_5, q_6\}$, $I_2 = \{q_2, q_3\}$ и $I_3 = \{q_4\}$.

Снова строим таблицу:

	q_1	q_2	q_3	q_4	q_5	q_6
0	I_3	I_1	I_1	I_2	I_3	I_3
1	I_3	I_1	I_1	I_2	I_3	I_3

Все классы остаются неизменными, значит мы дошли до конца алгоритма. Результат минимизации показан на рисунке 48.

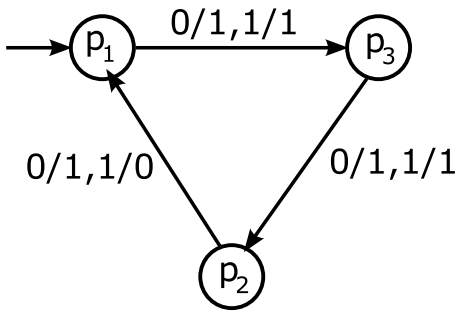

Рис. 48: Пример конечного автомата с выходом