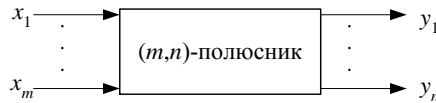


3. Основы теории логических сетей

3.1. Элементы

Понятие элемента относится к числу первичных в теории логических сетей, поэтому формальное определение элемента выходит за ее рамки, и представление о нем можно дать лишь на описательном уровне. Всякий элемент характеризуется множеством полюсов и множеством операторов $\{y_j = f_j(\cdot)\}$.



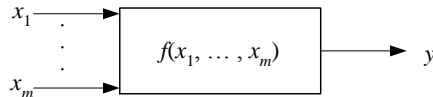
Реальные элементы обладают реальными физическими характеристиками:

- m – число входов;
- n – число выходов;
- k – нагрузочная способность выхода (максимальное число входов других элементов, которые можно подключить к выходу);
- τ – время задержки реакции элемента на входное воздействие.

По типу операторов элементы делятся на





- функциональные элементы;
- элементы памяти.

Функциональный элемент имеет $m \geq 1$ входов и один выход, то есть является m, n – полюсником. Оператор такого элемента есть булева функция $f(x_1, \dots, x_m)$ от переменных x_1, \dots, x_m , сопоставленных входам элемента.

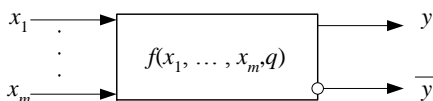


Общепринятые обозначения для основных функциональных элементов таковы

	инвертор (НЕ) – реализует инверсию входа;
--	---

	конъюнктор (И) – реализует конъюнкцию входов;
	элемент Шеффера (И-НЕ) – реализует инверсию конъюнкции входов;
	дизъюнктор (ИЛИ) – реализует дизъюнкцию входов;
	элемент Пирса (ИЛИ-НЕ) – реализует инверсию дизъюнкции входов.

Запоминающий элемент имеет $m \geq 1$ входов и один или два выхода, причем если выхода два, то значение на одном из них есть инверсия значения на другом.



Запоминающий элемент характеризуется множеством состояний $\{0,1\}$. Оператор $f(\cdot)$ запоминающего элемента связывает состояние $q \in \{0,1\}$ с набором значений входов x_1, \dots, x_m так, что для любого $q \in \{0,1\}$ существует набор значений переменных x_1, \dots, x_m , переводящий запоминающий элемент в состояние $q' \neq q$. Состояние элемента подается на его выход. Очевидно, запоминающий элемент есть автомат с двумя состояниями, но не всякий такой автомат есть запоминающий элемент.

Примерами запоминающих элементов могут служить *триггеры*. Это дискретные устройства с булевыми входными и выходными переменными, имеющие не более двух входов, один выход и ровно два внутренних состояния 0 и 1, то есть для триггеров $X = B$ или $X = B^2$, $Y = Q = B$.

Рассмотрим основные виды триггеров.

D –триггер (*delay* – задержать) задерживает вход на один такт. Он имеет один вход и описывается уравнениями

$$\begin{cases} y(t) = q(t); \\ q(t+1) = x(t), \end{cases}$$

или соотношением $y(t) = x(t-1)$. Очевидно, D –триггер есть автомат Мура вида $\langle B, B, B, \psi, \varphi \rangle$ с таблицей переходов

$D \backslash q$	0	1
0	0	0
1	1	1

T –триггер (*toggle* – переключать). Его выход совпадает с текущим состоянием, при поступлении на вход нуля состояние не меняется, при поступлении на вход единицы – инвертируется. T –триггер также можно рассматривать как автомат Мура вида $\langle B, B, B, \psi, \varphi \rangle$ с таблицей переходов–выходов

$T \backslash q$	0	1
0	0	1
1	1	0

RS –триггер (*reset* – сбросить, *set* – установить). RS –триггер имеет два входа, при поступлении на входы нулей состояние не меняется, при поступлении на вход R единицы состояние сбрасывается в ноль, при поступлении на вход S единицы состояние устанавливается в единицу. При поступлении единиц на оба входа состояние не определено, поэтому эту комбинацию подавать на входы не рекомендуется. Выход совпадает с текущим состоянием. RS –триггер рассматривается как частичный автомат Мура вида $\langle B^2, B, B, \psi, \varphi \rangle$, где $\varphi(q) = q$, а таблица переходов имеет вид

$RS \backslash q$	0	1
00	0	1
01	1	1
10	0	0
11	–	–

JK –триггер. Вход J аналогичен входу S , а вход K аналогичен входу R у RS –триггера за единственным исключением: если $J = K = 1$, то состояние триггера инвертируется.

$KJ \backslash q$	0	1
00	0	1
01	1	1
10	0	0
11	1	0

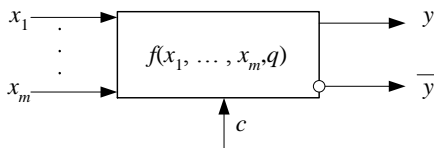
По способу отсчета времени запоминающие элементы делятся на

- синхронные;
- асинхронные.

У *синхронного* элемента имеется дополнительный вход c для указания момента времени, в который происходит изменение состо-

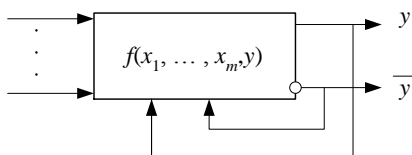
яния. Если $c = 0$ (нет синхросигнала), то состояние элемента не меняется независимо от того, меняется входной сигнал или нет. Если $c = 1$, то элемент переходит из состояния q в состояние q' , определяемое оператором $f(\cdot)$. Иначе говоря,

$$q' = f'(x_1, \dots, x_m, q, c) = \begin{cases} q, & \text{если } c = 0; \\ f(x_1, \dots, x_m, q), & \text{если } c = 1. \end{cases}$$



Пример. D –триггер – синхронный элемент.

Асинхронный элемент не имеет входа c и переходит из состояния q в состояние $q' = f(x_1, \dots, x_m, q)$ только при изменении входного сигнала. Иначе говоря, у синхронного элемента функция $f(\cdot)$ может быть произвольной, а у асинхронного нет, так как если в асинхронном элементе входной сигнал не меняется, то не меняется и состояние. Асинхронный элемент рассматривается как система с обратной связью.



Определение. Состояние q называется *устойчивым по отношению к входному символу x* (или просто *устойчивым*), если $f(x, q) = q$.

Определение. Состояние q называется *допустимым по отношению к входному символу x* (или просто *допустимым*), если под воздействием символа x автомат из состояния q перейдет в устойчивое состояние, иначе состояние называется *недопустимым*.

Пример. Рассмотрим некоторые триггеры.

RS –триггер – асинхронный элемент. Рассмотрим его таблицу функционирования. Отметим символом «*» устойчивые состояния,

символом «+» – допустимые, символом «–» – недопустимые.

R	S	q	q'
0	0	0	0 *
0	0	1	1 *
0	1	0	1 +
0	1	1	1 *
1	0	0	0 *
1	0	1	0 +
1	1	0	–
1	1	1	–

В RS –триггере все состояния, на которых он определен, являются устойчивыми либо допустимыми, поэтому он используется для синтеза более сложных триггеров.

JK –триггер. Асинхронный JK –триггер построить нельзя, так как при входе 11 состояние неустойчиво.

K	J	q	q'
0	0	0	0 *
0	0	1	1 *
0	1	0	1 +
0	1	1	1 *
1	0	0	0 *
1	0	1	0 +
1	1	0	1 –
1	1	1	0 –

В асинхронном элементе с временной задержкой τ переход из одного состояния в другое осуществляется за время τ . В синхронном элементе с временной задержкой τ синхронизирующие сигналы следуют через интервал времени $\tau' \geq \tau$, где τ' задается жестко, и переход синхронного элемента из одного состояния в другое осуществляется за время τ' . Таким образом, синхронный элемент работает не быстрее асинхронного.

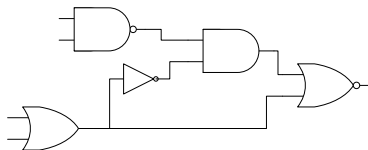
3.2. Логические сети

Определение. *Логической сетью* называется совокупность элементов, соединенных между собой некоторым образом посредством отождествления некоторых полюсов элементов.

Нарисуем на плоскости элементы сети так, чтобы они не перекрывали друг друга, и соединим линиями отождествленные полюсы

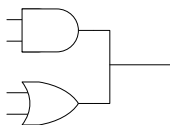
элементов. В результате получим геометрическую реализацию сети. Очевидно, что для одной сети существует множество геометрических реализаций, но по геометрической реализации сеть определяется однозначно.

Пример.



Не всякая логическая сеть задает реальное устройство.

Пример. В данной сети соединены выходы элементов.



Определение. Логическая сеть, задающая реальное физическое устройство, называется *правильной сетью*.

Далее правильную логическую сеть будем называть *сетью*.

Существует два класса сетей:

- комбинационные схемы;
- последовательностные схемы.

3.2.1. Комбинационные схемы

Дадим индуктивное **определение** комбинационной схемы.

База индукции. Всякий функциональный элемент e есть комбинационная схема, полюсами которой являются полюса элемента e .

Индуктивный переход. Если S – комбинационная схема, e – функциональный элемент, не содержащийся в S , и S' – результат включения в S элемента e путем отождествления каждого входа элемента e с не более чем одним полюсом схемы S , то S' – комбинационная схема. Входами схемы S' являются входы схемы S и входы элемента e , не отождествленные с полюсами схемы S ; выходами схемы S' являются выход элемента e и выходы схемы S , не отождествленные с входами элемента e .

Заключительная фраза. Других комбинационных схем нет.

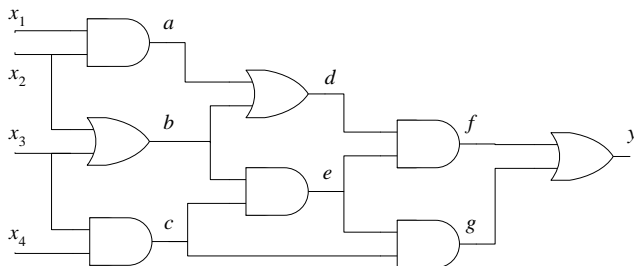
Отсюда следует, что в комбинационной схеме нет контуров.

3.2.1.1. Анализ комбинационной схемы

Рассмотрим произвольную комбинационную схему S с n входами. Сопоставим входам схемы булевы переменные x_1, \dots, x_n . Каждому полюсу схемы S припишем булеву функцию от переменных x_1, \dots, x_n по следующим правилам.

1. Если полюс есть вход схемы, то ему приписывается функция, равная сопоставленной ему переменной.
2. Если полюс есть выход элемента e с m входами, булева функция $h(y_1, \dots, y_m)$ есть оператор этого элемента, а $f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n)$ – функции, приписанные входам этого элемента y_1, \dots, y_m , то этому полюсу приписывается функция $f(x_1, \dots, x_n) = h(f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n))$.
3. Функция, приписанная полюсу i , приписывается всякому полюсу, отождествленному с i .

Пример. Рассмотрим комбинационную схему. Припишем выходам элементов символы функций и, начиная от выхода, будем строить суперпозицию этих функций до тех пор, пока не останутся только символы входных переменных; одновременно будем упрощать выражение, используя законы булевой алгебры.



$$\begin{aligned}
 y &= f \vee g; & f &= de; & d &= a \vee b; & a &= x_1 x_2; \\
 & & g &= ce; & e &= bc; & b &= x_1 \vee x_2; \\
 & & & & & & c &= x_2 x_3.
 \end{aligned}$$

$$y = f \vee g = de \vee ce = (d \vee c)e = (a \vee b \vee c)bc = bc = (x_1 \vee x_2)x_2x_3 = x_2x_3.$$

Следовательно, данную схему можно заменить одним конъюнктом.

Определение. Функция, приписанная полюсу схемы, называется функцией, реализуемой схемой на этом полюсе. Функция, припи-

санная выходу схемы, называется *функцией, реализуемой схемой*. Система функций, реализуемых схемой, называется *оператором схемы*. Этот оператор описывает поведение схемы.

Задача анализа комбинационной схемы. По заданной схеме определить ее оператор.

Общий метод решения этой задачи был рассмотрен выше. Задача имеет единственное решение.

3.2.1.2. Синтез комбинационной схемы

Задача синтеза. Задан оператор, то есть система булевых функций, и задан элементный базис, то есть набор функциональных элементов. Требуется построить схему в заданном элементном базисе, реализующую заданный оператор.

Данная задача в общем случае имеет множество решений. Обычно ставится задача синтеза схемы с некоторыми свойствами. Например, требуется построить схему с наименьшим числом элементов, или схему, принадлежащую некоторому классу схем. Последняя задача не всегда разрешима. Возможны следующие классы комбинационных схем.

Схемы с ограниченной глубиной. Под глубиной схемы понимается длина самой длинной последовательности элементов пути, ведущего от входов схемы к ее выходам.

Плоские схемы. Схема называется плоской, если есть ее геометрическая реализация на плоскости без пересечения соединений.

Древовидные схемы. Содержатся в классе плоских схем и являются удобными для диагностики.

Рассмотрим некоторые простейшие методы синтеза комбинационных схем.

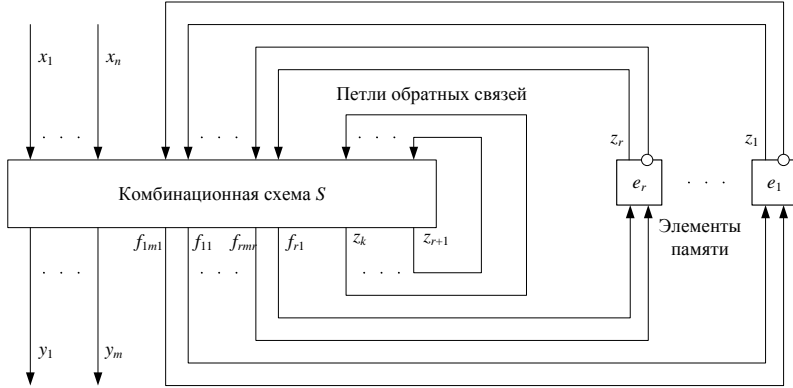
Синтез в базисе "И", "ИЛИ", "НЕ". Пусть мы имеем функциональное описание комбинационной схемы, где функции, реализованные схемой, заданы дизъюнктивными нормальными формами. Тогда можно применить двухъярусный метод синтеза схем: получаем необходимые инверсии входных переменных, затем на первом ярусе реализуем конъюнкции, на втором – дизъюнкции.

Синтез в базисе "И-НЕ". Строим схему в базисе "И", "ИЛИ", "НЕ" и заменяем все элементы на элементы "И-НЕ".

Синтез в базисе "ИЛИ-НЕ". Получаем систему функций, двойственных функциям, реализуемым схемой. Затем для полученной системы строим схему в базисе "И", "ИЛИ", "НЕ" и заменяем все элементы на элементы "ИЛИ-НЕ".

3.2.2. Последовательностные схемы

Всякая логическая сеть, не являющаяся комбинационной схемой, является последовательностной схемой. Основное отличие последовательностных схем от комбинационных состоит в том, что в них присутствуют либо контуры (обратные связи), либо элементы памяти, либо то и другое.



3.2.2.1. Анализ последовательностной схемы

Рассмотрим произвольную последовательностную схему N с n входами, m выходами, r элементами памяти, $k - r$ петлями обратной связи и комбинационной схемой S . Сопоставим во взаимно-однозначное соответствие входам входные булевы переменные x_1, \dots, x_n , выходам – выходные булевы переменные y_1, \dots, y_m , неинверсным выходам элементов памяти – переменные z_1, \dots, z_r со значениями состояний этих элементов, петлям обратной связи – переменные z_{r+1}, \dots, z_k со значениями сигналов, передаваемых по линиям обратной связи. Припишем всем полюсам схемы S булевы функции по определенным выше правилам. Тогда

$$\begin{cases} y_1 = \varphi_1(x_1, \dots, x_n, z_1, \dots, z_k); \\ \dots\dots\dots; \\ y_m = \varphi_m(x_1, \dots, x_n, z_1, \dots, z_k); \\ z_{r+1} = \psi_{r+1}(x_1, \dots, x_n, z_1, \dots, z_k); \\ \dots\dots\dots; \\ z_k = \psi_k(x_1, \dots, x_n, z_1, \dots, z_k). \end{cases}$$

Переменные z_1, \dots, z_k назовем *внутренними переменными* схемы N , а набор их значений – ее *состоянием*.

Обозначим через $f_{ij}(x_1, \dots, x_n, z_1, \dots, z_k)$ функцию, приписанную тому выходу схемы S , который отождествлен с j -м входом i -го элемента памяти. Все введенные функции описывают поведение схемы S . Опишем через них поведение схемы N .

Пусть $g_i(u_1, \dots, u_{m_i}, z_i)$ есть функция i -го элемента памяти с входными переменными u_1, \dots, u_{m_i} и внутренней переменной z_i . Скомбинируем из функций $g_i(\cdot)$ и $f_{ij}(\cdot)$ новые функции $\psi_i(\cdot)$ в виде следующих суперпозиций:

$$\begin{aligned} \psi_i(x_1, \dots, x_n, z_1, \dots, z_k) &= \\ &= g_i(f_{i1}(x_1, \dots, x_n, z_1, \dots, z_k), \dots, f_{im_i}(x_1, \dots, x_n, z_1, \dots, z_k), z_i), \\ &\quad i = \overline{1, r}. \end{aligned}$$

Функции $\phi_j(\cdot)$, $j = \overline{1, m}$ и $\psi_i(\cdot)$, $i = \overline{1, k}$ описывают всю схему N . Функции $\phi_j(\cdot)$ называются *структурными функциями выходов*, а $\psi_i(\cdot)$ – *структурными функциями переходов* схемы N . В итоге мы можем записать систему канонических уравнений последовательностной схемы

$$\begin{cases} y_j = \phi_j(x_1, \dots, x_n, z_1, \dots, z_k), & j = \overline{1, m}; \\ z'_i = \psi_i(x_1, \dots, x_n, z_1, \dots, z_k), & i = \overline{1, k}. \end{cases}$$

Введем обозначения $x = (x_1, \dots, x_n)$, $y = (y_1, \dots, y_m)$, $z = (z_1, \dots, z_k)$, $\phi_N = (\phi_1, \dots, \phi_m)$, $\psi_N = (\psi_1, \dots, \psi_k)$, тогда систему можно переписать

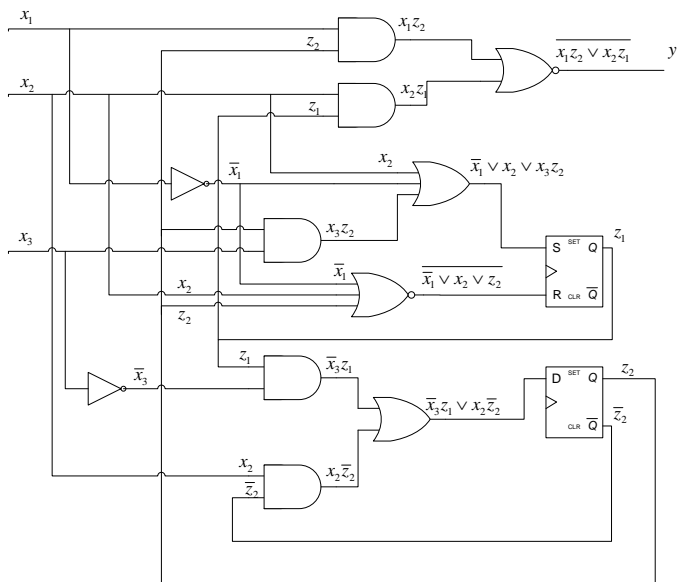
$$\begin{cases} y = \phi_N(x, z); \\ z' = \psi_N(x, z). \end{cases}$$

Эта система имеет такой же вид, как система канонических уравнений конечного автомата с входами, выходами и состояниями, представленными булевыми векторами, компоненты которых вычисляются как значения булевых функций.

Задача анализа последовательностной схемы. По заданной схеме определить ее оператор (конечный автомат).

Общий метод решения этой задачи дан выше. Задача имеет единственное решение.

Пример. Рассмотрим последовательностную схему. Здесь ее полюсам приписаны соответствующие функции.



Отсюда

$$\begin{cases} y = \overline{x_1 z_2 \vee x_2 z_1}; \\ S = \bar{x}_1 \vee x_2 \vee x_3 z_2; \\ R = \bar{x}_1 \vee x_2 \vee z_2; \\ D = \bar{x}_3 z_1 \vee x_2 \bar{z}_2. \end{cases}$$

Для выхода D -триггера имеем $z'_2 = D$. Получим функцию выхода RS -триггера. Рассмотрим для этого его таблицу истинности.

R	S	q	q'
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	—
1	1	1	—

Получим кратчайшую ДНФ для данной функции:

$$q' = \bar{R}q \vee S.$$

Отсюда

$$\begin{aligned} z_1' &= \bar{R}z_1 \vee S = \overline{(\bar{x}_1 \vee x_2 \vee z_2)} z_1 \vee (\bar{x}_1 \vee x_2 \vee x_3 z_2) = \\ &= (\bar{x}_1 \vee x_2 \vee z_2) z_1 \vee \bar{x}_1 \vee x_2 \vee x_3 z_2 = z_2 z_1 \vee \bar{x}_1 \vee x_2 \vee x_3 z_2. \end{aligned}$$

Окончательно имеем

$$\begin{cases} y = \overline{x_1 z_2 \vee x_2 z_1}; \\ z_1' = z_2 z_1 \vee \bar{x}_1 \vee x_2 \vee x_3 z_2; \\ z_2' = \bar{x}_3 z_1 \vee x_2 \bar{z}_2. \end{cases}$$

Это и есть оператор схемы.

3.2.2.2. Синтез последовательностной схемы

Далее будем рассматривать автоматы $A = \langle X, Q, Y, \psi, \phi \rangle$, где $X = B^n$, $Y = B^m$. С одной стороны, это оправдано тем, что всегда можно перейти от абстрактных алфавитов X , Y к двоичным алфавитам, с другой стороны, именно такие автоматы являются моделями реальных дискретных устройств.

Определение. Будем говорить, что схема N с m выходами и n входами *моделирует* автомат A , если для любого состояния q найдется набор σ значений внутренних переменных схемы z_1, \dots, z_n такой, что $\forall \alpha \in B^{n*} : \phi(\alpha, s) = \phi(\alpha, \sigma)$.

Задача синтеза последовательностной схемы. Задан автомат, и задан элементный базис. Требуется построить последовательностную схему в заданном элементном базисе, моделирующую заданный автомат.

Покажем принципиальную возможность синтеза последовательностной схемы, моделирующей заданный автомат.

Пусть задан автомат $A = \langle X, Q, Y, \psi, \phi \rangle$, где $X = B^n$, $Y = B^m$. Найдем наименьшее число k , удовлетворяющее условию $2^k \geq |Q|$ (вообще говоря, можно взять любое k , удовлетворяющее этому условию). Рассмотрим произвольное взаимно-однозначное отображение $\rho : Q \rightarrow B^k$. Иначе говоря, каждому состоянию автомата A поставим во взаимно-однозначное соответствие некоторый булев вектор (код) размерности k , то есть состояние некоторой последо-

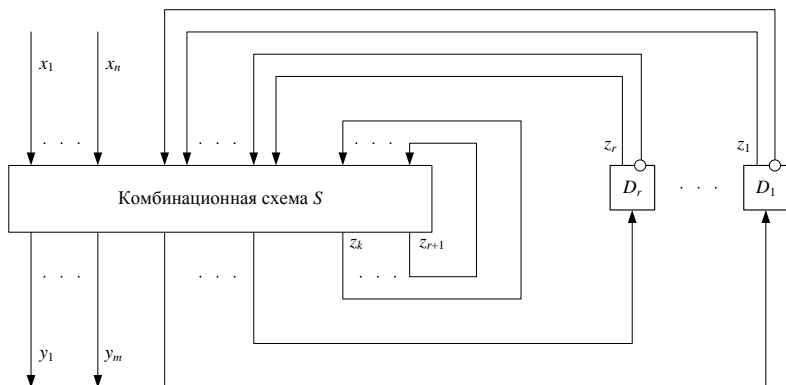
вательностной схемы N . Для любых $x \in X$ и $z \in B^k$ положим

$$\begin{cases} \psi_N(x, z) = \rho(\psi(x, s)); \\ \varphi_N(x, z) = \varphi(x, s), \end{cases}$$

где $\rho(s) = z$. Индукцией по длине слова нетрудно убедиться, что последовательностная схема N с функциями $\psi_N(\cdot)$, $\varphi_N(\cdot)$ моделирует заданный автомат. Систему ее канонических уравнений можно записать как

$$\begin{cases} y_j = \varphi_j(x_1, \dots, x_n, z_1, \dots, z_k), & j = \overline{1, m}; \\ z'_i = \psi_i(x_1, \dots, x_n, z_1, \dots, z_k), & i = \overline{1, k}. \end{cases}$$

Этой системе соответствует последовательностная схема в каноническом виде, изображенная на рисунке. Здесь через D_i обозначены D -триггеры. Остается построить комбинационную схему S , взяв для этого любой функционально полный базис (например, "И,ИЛИ,НЕ").



Пример. Рассмотрим автомат с двумя входами и одним выходом, заданный таблицей переходов и выходов.

$X \backslash Q$	1	2	3	4
00	1/0	3/0	3/1	4/1
01	2/0	2/1	1/0	1/1
10	3/0	3/0	2/0	2/0
11	3/1	4/1	2/0	4/0

Найдем объем памяти автомата. Поскольку $\log_2 4 = 2$, то $k = 2$.

Зададим кодирование $\rho: Q \rightarrow B^2$.

	1	2	3	4
z_1	0	0	1	1
z_2	0	1	0	1

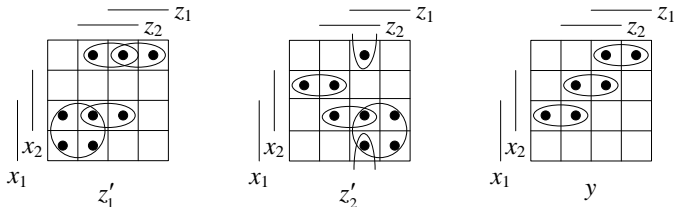
Положим

$$\begin{cases} \psi_N(x, z) = \rho(\psi(x, q)); \\ \varphi_N(x, z) = \varphi(x, q), \end{cases}$$

где $\rho(q) = z$. Выпишем таблицы истинности функций переходов и выходов. Таблицы функций $\psi_N = (\psi_1, \psi_2)$ и $\varphi_N = (\varphi)$ получаются из таблиц функций ψ и φ заменой всюду $q \in Q$ на $\rho(q)$. Зададим эти функции таблицами истинности и найдем их кратчайшие ДНФ.

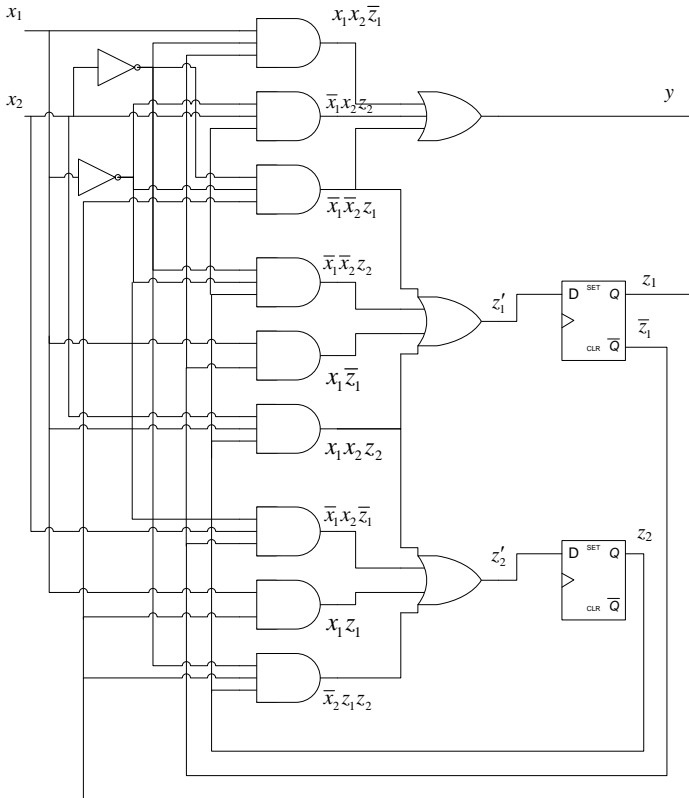
x_1	x_2	q	q'	z_1	z_2	z'_1	z'_2	y
0	0	1	1	0	0	0	0	0
0	0	2	3	0	1	1	0	0
0	0	3	3	1	0	1	0	1
0	0	4	4	1	1	1	1	1
0	1	1	2	0	0	0	1	0
0	1	2	2	0	1	0	1	1
0	1	3	1	1	0	0	0	0
0	1	4	1	1	1	0	0	1
1	0	1	3	0	0	1	0	0
1	0	2	3	0	1	1	0	0
1	0	3	2	1	0	0	1	0
1	0	4	2	1	1	0	1	0
1	1	1	3	0	0	1	0	1
1	1	2	4	0	1	1	1	1
1	1	3	2	1	0	0	1	0
1	1	4	4	1	1	1	1	0

Кратчайшая система ДНФ для этой системы функций найдена визуально по матрицам Грея и имеет вид



$$\begin{cases} z'_1 = x_1 \bar{z}_1 \vee x_1 x_2 z_2 \vee \bar{x}_1 \bar{x}_2 z_2 \vee \bar{x}_1 \bar{x}_2 z_1; \\ z'_2 = x_1 z_1 \vee x_1 x_2 z_2 \vee \bar{x}_1 x_2 \bar{z}_1 \vee \bar{x}_2 z_1 z_2; \\ y = \bar{x}_1 \bar{x}_2 z_1 \vee \bar{x}_1 x_2 z_2 \vee x_1 x_2 \bar{z}_1. \end{cases}$$

Теперь по этим каноническим уравнениям строим комбинационную схему, а затем последовательностную схему. Для этого выходы комбинационной схемы z'_1 и z'_2 отождествляются с входами D -триггеров, а выходы триггеров – с входами z_1 и z_2 комбинационной схемы.



Если в качестве элементов памяти используются не D -триггеры, оператор которых имеет вид $f(x, z) = x$, а некоторые

другие триггеры T_i с m_i входами и операторами $g_i \neq f$, то возникает задача отыскания *функций возбуждения* $f_{ij}(\cdot)$ триггеров T_i по заданным функциям $\psi_i(\cdot)$, $i = \overline{1, k}$, $j = \overline{1, m_i}$ (функций, которые необходимо подать на входы триггера, чтобы реализовать желаемый переход). Как отмечалось в предыдущем разделе, функции $f_{ij}(\cdot)$ и $\psi_i(\cdot)$ связаны соотношениями

$$\begin{aligned} \psi_i(x_1, \dots, x_n, z_1, \dots, z_k) &= \\ &= g_i(f_{i1}(x_1, \dots, x_n, z_1, \dots, z_k), \dots, f_{im_i}(x_1, \dots, x_n, z_1, \dots, z_k), z_i), \\ &\quad i = \overline{1, k}. \end{aligned}$$

Из этих соотношений вытекает общий метод построения функций возбуждения.

Пусть функция $\psi_i(\cdot)$ задана таблицей истинности. Разобьем строки этой таблицы на два класса. В первый поместим те строки, где $\psi_i(\cdot) = 0$, во второй – те строки, где $\psi_i(\cdot) = 1$. Каждый класс, в свою очередь, разобьем на два подкласса. В первый поместим те строки, где $z_i = 0$, во второй – те строки, где $z_i = 1$. Построим таблицу справа m_i столбцами, помеченными символами функций возбуждения $f_{ij}(\cdot)$. В результате таблица примет вид:

x_1	...	x_n	z_1	...	z_i	...	z_k	ψ_i	f_{i1}	...	f_{im_i}
			0		0			0			
					
			0		0			0			
			1		0			0			
					
			1		0			0			
			0		1			1			
					
			0		1			1			
			1		1			1			
					
			1		1			1			

Теперь для построения функций возбуждения триггера T_i нужно заполнить столбцы $f_{ij}(\cdot)$ так, чтобы выполнялось равенство

$$\begin{aligned} \psi_i(x_1, \dots, x_n, z_1, \dots, z_k) = \\ = g_i(f_{i1}(x_1, \dots, x_n, z_1, \dots, z_k), \dots, f_{im_i}(x_1, \dots, x_n, z_1, \dots, z_k), z_i). \end{aligned}$$

Для этого достаточно записать в строки первого подкласса те наборы значений переменных u_1, \dots, u_{m_i} , для которых $g_i(u_1, \dots, u_{m_i}, 0) = 0$, в строки второго подкласса – те наборы значений переменных u_1, \dots, u_{m_i} , для которых $g_i(u_1, \dots, u_{m_i}, 1) = 0$, в строки третьего, для которых $g_i(u_1, \dots, u_{m_i}, 0) = 1$, в строки четвертого подкласса – те наборы значений переменных u_1, \dots, u_{m_i} , для которых $g_i(u_1, \dots, u_{m_i}, 1) = 1$.

Рассмотрим сводную таблицу возбуждения функций изученных нами триггеров.

q	q'	D	T	R	S	K	J
0	0	0	0	–	0	–	0
0	1	1	1	0	1	–	1
1	0	0	1	1	0	1	–
1	1	1	0	0	–	0	–
–	0	0	×	1	0	1	0
–	1	1	×	0	1	0	1

Пример. Получим функции возбуждения триггеров для автомата из предыдущего примера.

x_1	x_2	z_1	z_2	z'_1	z'_2	T_1	T_2	R_1	S_1	R_2	S_2	K_1	J_1	K_2	J_2
0	0	0	0	0	0	0	0	–	0	–	0	–	0	–	0
0	0	0	1	1	0	1	1	0	1	1	0	–	1	1	–
0	0	1	0	1	0	0	0	0	–	–	0	0	–	–	0
0	0	1	1	1	1	0	0	0	–	0	–	0	–	0	–
0	1	0	0	0	1	0	1	–	0	0	1	–	0	–	1
0	1	0	1	0	1	0	0	–	0	0	–	–	0	0	–
0	1	1	0	0	0	1	0	1	0	–	0	1	–	–	0
0	1	1	1	0	0	1	1	1	0	1	0	1	–	1	–
1	0	0	0	1	0	1	0	0	1	–	0	–	1	–	0
1	0	0	1	1	0	1	1	0	1	1	0	–	1	1	–
1	0	1	0	0	1	1	1	1	0	0	1	1	–	–	1
1	0	1	1	0	1	1	0	1	0	0	–	1	–	0	–
1	1	0	0	1	0	1	0	0	1	–	0	–	1	–	0
1	1	0	1	1	1	1	0	0	1	0	–	–	1	0	–
1	1	1	0	0	1	1	1	0	1	0	–	–	1	0	–
1	1	1	1	1	1	0	0	0	–	0	–	0	–	0	–

3.2.2.3. Синхронные и асинхронные схемы, понятие состязания в асинхронной схеме

Последовательностные схемы делятся на два класса:

- синхронные схемы;
- асинхронные схемы.

Определение. Схема называется *синхронной*, если она не содержит петель обратной связи и всякий ее элемент памяти есть синхронный триггер, иначе схема называется *асинхронной*.

В синхронных схемах триггеры имеют дополнительный вход, по которому передается синхронизирующий сигнал c , позволяющий триггерам, и следовательно схеме, переходить из одного состояния в другое. Все триггеры синхронной схемы меняют свое состояние одновременно при наличии сигнала c . Этот факт снимает массу проблем, которые возникают в асинхронных схемах. Рассмотрим основные проблемы асинхронной схемотехники.

Неустойчивость. Рассмотрим асинхронный автомат. Состояние и выход асинхронного автомата изменяются только при изменении входа. При этом автомат пробегает последовательность неустойчивых состояний (*неустойчивый такт*) и достигает устойчивого состояния (*устойчивый такт*). При этом теряет смысл рассматривать выходы в неустойчивых состояниях. Выход есть функция лишь устойчивого состояния, то есть асинхронный автомат является частичным.

Если же функции переходов таковы, что под воздействием некоторого входа автомат не достигает устойчивого состояния, то возникает проблема неустойчивости. Для устойчивого состояния должно выполняться условие $\psi_{ik} = k$, то есть, чтобы автомат был устойчивым, для любого элемента ψ_{ij} таблицы переходов должна существовать цепочка

$$\psi_{ij} = k_1, \psi_{ik_1} = k_2, \dots, \psi_{ik_l} = k_l,$$

где k_l – устойчивое состояние. Таблица переходов, обладающая таким свойством, называется *устойчивой*.

Если автомат задан диаграммой переходов, то для любого входного символа $x \in X$ и любой вершины в графе должен существовать путь, где все дуги помечены символом x , в вершину с петлей, помеченной символом x . В графе не существует циклов, помеченных этим символом, кроме петель.

Пример. Слева – неустойчивая таблица переходов, так как под

воздействием символа b автомат пробегает цепочку состояний $1 \rightarrow 3 \rightarrow 1 \rightarrow 3 \rightarrow \dots$, то есть не попадает в устойчивое состояние. Справа – устойчивая таблица. Устойчивые состояния подчеркнуты.

$X \backslash Q$	1	2	3	$X \backslash Q$	1	2	3
a	2	1	<u>3</u>	a	2	3	<u>3</u>
b	3	3	1	b	<u>1</u>	1	2
c	3	<u>2</u>	1	c	3	<u>2</u>	2
d	2	3	2	d	<u>1</u>	3	<u>3</u>

Можно потребовать, чтобы при любом входном воздействии автомат сразу попадал в устойчивое состояние. Тогда мы получим нормальную таблицу переходов.

Определение. Таблица (функция) переходов асинхронного автомата называется *нормальной*, если

$$\forall x \in X, q \in Q: \psi(x, q) = q' \Rightarrow \psi(x, q') = q'.$$

Пример. Слева – устойчивая таблица из предыдущего примера, справа – полученная из нее нормальная таблица.

$X \backslash Q$	1	2	3	$X \backslash Q$	1	2	3
a	2	3	<u>3</u>	a	3	3	<u>3</u>
b	<u>1</u>	1	2	b	<u>1</u>	1	1
c	3	<u>2</u>	2	c	2	<u>2</u>	2
d	<u>1</u>	3	<u>3</u>	d	<u>1</u>	3	<u>3</u>

На таблицу выходов никаких ограничений не накладывается, но в клетках, соответствующих неустойчивым состояниям, ставятся прочерки.

Опасные состязания. Рассмотрим эту проблему на примере.

Пример. Пусть автомат задан таблицей переходов

$X \backslash Q$	1	2	3	4
a	<u>1</u>	3	<u>3</u>	1
b	2	<u>2</u>	4	<u>4</u>
c	<u>1</u>	1	1	1
d	<u>1</u>	<u>2</u>	1	<u>4</u>

Пусть кодирование $\rho: Q \rightarrow B^2$ имеет вид:

$Z \backslash Q$	1	2	3	4
z_1	0	0	1	1
z_2	0	1	0	1

Предположим, что последовательностная схема, моделирующая данный автомат, в какой-то момент времени находится в состоянии 01, соответствующем состоянию 2 автомата, а на вход при этом поступает сигнал a . Согласно таблице переходов, схема должна перейти в состояние 10, соответствующее состоянию 3 автомата. Пусть триггер T_1 переключился в состояние 1, а триггер T_2 еще остается в состоянии 1. Тогда схема на некоторое время остается в состоянии 11, соответствующее состоянию 4 автомата, которое вместе с сигналом a действует на ее входы. Под действием a схема должна перейти из состояния 11 в состояние 00. Представим, что это произошло. Тогда схема под действием неизменного сигнала a может остаться в состоянии 00, хотя должна была перейти в состояние 10. То есть, из-за наличия различных задержек в элементах памяти схема работает неверно, автомат не переходит в предписанное ему состояние. Такое явление называется *опасными состязаниями* или *гонками*.

Очевидно, что опасные состязания могут возникнуть только в том случае, когда несколько элементов памяти должны одновременно сменить свои значения. Состязание не всегда является опасным. Так, если схема в только что рассмотренном примере находится в состоянии 11, и на вход подается сигнал c , то согласно таблице переходов автомата схема должна перейти в состояние 00. Оба триггера должны сменить свое состояние, так что возможно состязание. Однако, в этом случае схема сработает правильно, независимо от того, какой триггер быстрее переключится. Это произойдет потому, что под действием сигнала c схеме как из состояния 01, так и из состояния 10, предписывается переход в состояние 00.

Если состязание приводит к переходам в не предписанные состояния, оно называется *опасным*, иначе – *неопасным*.

3.2.2.4. Уточнение задачи синтеза и сведение ее к задаче кодирования состояний автомата

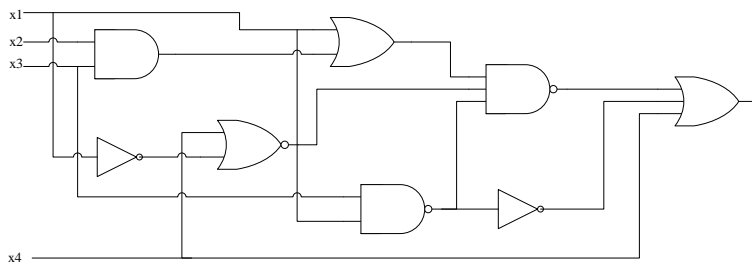
Задача синтеза асинхронных схем. Задан автомат и задан элементный базис, требуется построить асинхронную схему, моделирующую данный автомат и не имеющую опасного состязания элементов памяти.

Задача синтеза синхронных схем. Задан автомат и задан элементный базис, требуется построить наиболее простую синхронную схему, моделирующую данный автомат.

Один из способов решения такой задачи состоит в выборе наиболее подходящего отображения $\rho: Q \rightarrow B^k$, называемого *кодированием состояний автомата*. Оказывается, состояния автомата можно закодировать таким образом, что в моделирующей этот автомат асинхронной схеме опасных состязаний элементов возникать не будет. Далее такое кодирование называется *противогоночным*. Разные кодирования состояний автомата приводят к получению разных систем структурных функций переходов и выходов, которые, в свою очередь, реализуются схемами разной сложности. Экспериментальные данные говорят о том, что более простые структурные функции переходов и выходов, как правило, реализуются более простыми схемами. Таким образом, если учесть, что при заданном кодировании состояний автомата построение последовательностной схемы не содержит принципиальных трудностей, то можно утверждать, что решение задачи синтеза асинхронной схемы сводится к отысканию противогоночного кодирования состояний автомата, а решение задачи синтеза синхронной схемы сводится к отысканию кодирования, приводящего к получению более простых структурных функций переходов и выходов. Далее мы рассмотрим методы построения таких кодирований.

3.3. Задачи

Задача 1. Провести анализ комбинационной схемы.



Синтезировать как можно более простую комбинационную схему с тем же оператором.

Задача 2. Синтезировать схему, моделирующую систему булевых функций

$$\begin{cases} \varphi_f = 001011011011111; \\ \varphi_g = 1111110100011111. \end{cases}$$

в базисах «И, ИЛИ, НЕ», «И-НЕ», «ИЛИ-НЕ».

Задача 3. Синтезировать синхронный D -триггер, используя RS -триггер в качестве элемента памяти.

Задача 4. Синтезировать сеть, используя:

- базис «И, ИЛИ, НЕ» и JK -триггер;
- базис «И-НЕ» и RS -триггер;
- базис «ИЛИ-НЕ» и D -триггер;
- базис «ИЛИ-НЕ» и T -триггер.

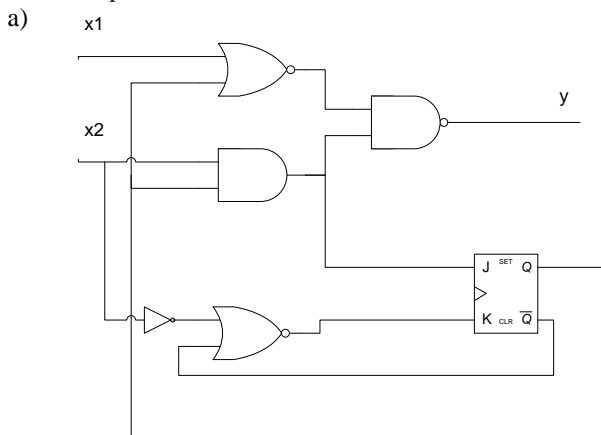
X/Q	0	1
00	1/0	0/0
01	0/1	1/1
10	1/1	0/1
11	1/1	1/0

Задача 5. Выбрать для конечного автомата из предыдущей задачи элементный базис, обеспечивающий наиболее простую схему.

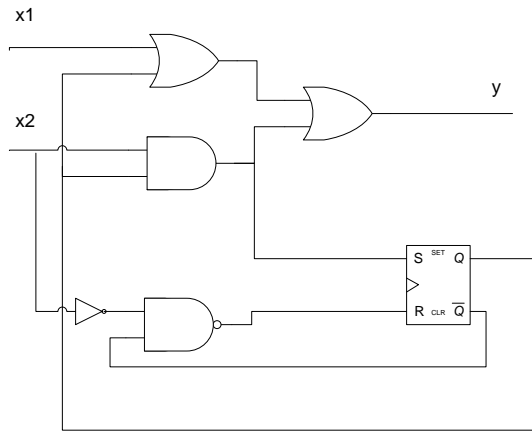
Задача 6. Построить наиболее простую схему сумматора двух двоичных чисел.

Задача 7. Синтезировать схемы, моделирующие конечные автоматы, построенные в двух предыдущих разделах, выбрав наилучший элементный базис.

Задача 8. Провести анализ последовательностных схем



b)



4. Методы противогоночного кодирования

4.1. Соседнее кодирование

Определение. Кодирование состояний автомата, при котором переход из одного состояния кодирующей его схемы в другое осуществляется путем изменения значения ровно одной внутренней переменной, называется *соседним*.

Очевидно, при соседнем кодировании состояний автомата опасное состязание элементов памяти возникать не будет, то есть соседнее кодирование является противогоночным. Для того, чтобы кодирование было соседним, необходимо, чтобы состояния автомата, связанные переходом, были закодированы соседними булевыми векторами.

Пример. Требуется осуществить соседнее кодирование состояний автомата, заданного таблицей переходов.

$X \backslash Q$	1	2	3	4
a	1	3	3	1
b	2	2	4	4
c	1	1	1	1
d	1	2	1	4

Здесь отмечены устойчивые состояния. Чтобы закодировать состояния этого автомата, требуется как минимум две внутренних переменных. Однако, этого недостаточно для осуществления соседнего кодирования, так как в автомате имеется цепочка переходов $1 \xrightarrow{b} 2 \xrightarrow{a} 3 \xrightarrow{c} 1$, образующая «треугольник». Более того, кодами большей длины закодировать данные состояния также нельзя по той же причине – не существует трех булевых векторов, каждые два из которых являются соседними.

Определение. *Элементарным переходом* называется переход из одного состояния автомата в другое, при котором меняется ровно одна внутренняя переменная.

Для решения задачи соседнего кодирования прямые переходы заменяются последовательностью элементарных переходов. Как показано выше, это может потребовать введения некоторых дополнительных состояний.

Рассмотрим некоторые методы получения таких промежуточных состояний.

4.1.1. Метод универсального соседнего кодирования с использованием связанных q -множеств

Определение. Кодирование состояний автомата, независящее от вида функций переходов и выходов, называется *универсальным*.

Определение. Назовем q -множеством (обозначается R_q) всякую совокупность значений внутренних переменных, приписанную состоянию q автомата.

Определение. Множество булевых векторов (в частности, q -множество) называется *связным*, если для любых векторов α и β из этого множества существует цепочка векторов этого же множества, содержащая α и β , в которой любые два рядом стоящих набора являются соседними по Хэммингу.

Пример. Множество $\{000, 001, 011, 111\}$ является связным.

Поставим задачу приписывания состояниям автомата q -множеств, обладающих следующими свойствами:

- всякое q -множество связно;
- если существует переход из состояния i в состояние j , то $R_i \cup R_j$ связно.

Очевидно, что объединение связанных множеств $R_i \cup R_j$ будет связно, если существует пара соседних векторов, один из которых принадлежит R_i , а другой – R_j .

Решим поставленную задачу на рассмотренном примере.

$X \backslash Q$	1	2	3	4
a	1	3	3	1
b	2	2	4	4
c	1	1	1	1
d	1	2	1	4

В заданном автомате переходами связаны состояния (1,2), (2,3), (3,4), (1,4), (1,3). Положим для начала

$$R_1 = \{000\}, R_2 = \{010\}, R_3 = \{110\}, R_4 = \{111\}.$$

1			2
		4	3

Такой выбор обеспечивает выполнение требований для первых трех пар состояний. Попробуем расширить множества так, чтобы требования выполнялись для последних двух пар.

$$R_1 = \{000, 001\}, R_2 = \{010, 011\}, R_3 = \{110, 100\}, R_4 = \{111, 101\}.$$

1a	1b	2b	2a
3b	4b	4a	3a

Как сами эти множества, так и их попарные расширения связаны. Рассмотрим исходную таблицу переходов

$X \backslash Q$	1	2	3	4
a	1	3	3	1
b	2	2	4	4
c	1	1	1	1
d	1	2	1	4

Преобразуем ее в таблицу элементарных переходов, введя дополнительные состояния.

$X \backslash Q$	000 $1a$	001 $1b$	010 $2a$	011 $2b$	100 $3a$	110 $3b$	101 $4a$	111 $4b$
a	1a	1a	3b	2a	3a	3a	1b	4a
b	2a	1a	2a	2a	4a	4b	4a	4a
c	1a	1a	1a	1b	1a	1a	1b	3b
d	1a	1a	2a	2a	1a	1a	4a	4a

Любому переходу $i \rightarrow j$ в исходном автомате соответствует последовательность элементарных переходов в расширенном автомате, например, переходу из состояния 2 в состояние 3 под действием символа a соответствует цепочка переходов $2a \rightarrow 3b \rightarrow 3a$.

Если q -множества, удовлетворяющие условиям, построить трудно, то можно воспользоваться следующим алгоритмом, который осуществляет кодирование с длиной кода, не превышающей удвоенной минимальной длины кода.

Алгоритм универсального соседнего кодирования с использованием связанных q -множеств

Начало. Дан автомат с n состояниями. Требуется найти q -множества для соседнего кодирования.

Шаг 1. Определяем длину кода $2N$, где $N \geq \log_2 n$.

Шаг 2. Строим квадратную матрицу Грея для $2N$ переменных.

Двумя перпендикулярными линиями, проведенными через середины сторон, разбиваем матрицу на четыре части.

Шаг 3. Верхнюю левую часть заполняем построчно числами $1, \dots, 2^{N-1}$, этими же числами, но по столбцам, заполняем нижнюю левую часть. Нижнюю правую часть заполняем построчно числами $2^{N-1} + 1, \dots, 2^N$, этими же числами, но по столбцам, заполняем верхнюю правую часть.

Шаг 4. К одному q -множеству R_q относим все наборы, помеченные номером q .

Конец.

Пример. Пусть $N = 2$. Матрица Грея примет вид

	1	1	3	4
	2	2	3	4
1	1	2	3	3
1	2	4	4	

- $R_1 = \{0000, 0001, 1100, 1000\}$;
- $R_2 = \{0100, 0101, 1101, 0101\}$;
- $R_3 = \{0011, 0111, 1111, 1110\}$;
- $R_4 = \{0010, 0110, 1011, 1010\}$.

Выше нам удалось найти кодирование для четырех состояний с меньшей длиной кода.

4.1.2. Метод соседнего кодирования с совместным использованием кодов

В отличие от предыдущего метода, где каждый набор (код) присылался какому-либо состоянию, здесь применяется более гибкий подход, а именно: с каждым состоянием автомата связывается единственный код, а остальные используются для организации промежуточных переходов между двумя разными состояниями. Отсюда происходит термин «совместное использование кодов». Метод применяется к автоматам, функция переходов которых является нормаль-

ной. Метод является приближенным в том смысле, что не гарантирует получения соседнего кодирования $\rho: Q \rightarrow B^k$ с минимальным значением k .

Зафиксируем входной символ $x \in X$ и состояние $q \in Q$ автомата. Построим множество $K_{x,q} = \{p \in Q: \psi(x, p) = q\}$. Построенное таким образом множество назовем K -множеством.

Метод состоит из четырех этапов.

Этап I. Определение длины кода.

Этап II. Построение K -множеств.

Этап III. Кодирование состояний.

Этап IV. Связывание K -множеств.

Этап V. Преобразование таблицы переходов в таблицу элементарных переходов.

Рассмотрим подробно эти этапы.

Этап I. Первоначально длина кода k – это минимальное целое число, удовлетворяющее условию $k \geq \log_2 |Q|$. Если для этой размерности после реализации последующих этапов решение не получится, то k увеличивается на 1 и снова выполняются этапы III, IV, пока не будет получено решение.

Пример. Требуется осуществить соседнее кодирование состояний автомата, заданного таблицей переходов.

$X \backslash Q$	1	2	3	4	5	6
a	1	1	3	4	4	4
b	2	2	2	2	5	5
c	6	5	3	3	5	6

Длина кода $k \geq \log_2 6$, значит, $k = 3$.

Этап II. Для каждой пары (x, q) найдем K -множество $K_{x,q}$. Затем построим матрицу связей F , строкам и столбцам которой сопоставим состояния автомата. Элемент матрицы $F(i, j)$ равен числу K -множеств, содержащих пару (i, j) , если $i \neq j$, и называется *коэффициентом связи* пары (i, j) . В клетки матрицы $F(i, i)$ ничего не записываем. Для каждого состояния подсчитаем *вес* – сумму коэффициентов связи (то есть сумму недиагональных элементов соответствующей строки матрицы F), и запишем в последний столбец матрицы.

Пример. Продолжим рассмотрение предыдущего примера

$X \backslash Q$	1	2	3	4	5	6
a	1	1	3	4	4	4
b	2	2	2	2	5	5
c	6	5	3	3	5	6

Выпишем K – множества, содержащие больше одного элемента.

$$\begin{aligned}
 K_{a,1} &= \{1, 2\} & K_{a,4} &= \{4, 5, 6\} \\
 K_{b,2} &= \{1, 2, 3, 4\} & K_{b,5} &= \{5, 6\} \\
 K_{c,3} &= \{3, 4\} & K_{c,5} &= \{2, 5\} & K_{c,6} &= \{1, 6\}
 \end{aligned}$$

Матрица связей имеет вид

	1	2	3	4	5	6	Σ
1		2	1	1	0	1	5
2	2		1	1	1	0	5
3	1	1		2	0	0	4
4	1	1	2		1	1	6
5	0	1	0	1		2	4
6	1	0	0	1	2		4

Этап III. Осуществим предварительное кодирование состояний. Цель этого этапа – закодировать состояния так, чтобы как можно большее число K – множеств оказались связными. Приведем один из эвристических алгоритмов такого кодирования.

Пусть S – множество кодов, приписанных каким-либо состояниям, $R = B^k \setminus S$ – множество кодов, не приписанных никаким состояниям.

Шаг 1. Состоянию с наибольшим весом припишем код $0\dots 0$. Положим $S = \{0\dots 0\}$.

Шаг 2. Выпишем R' – подмножество множества R , каждый элемент которого является соседним по крайней мере одному коду из S :

$$R' = \{\alpha \in R : \exists \beta \in S : d(\alpha, \beta) = 1\},$$

где $d(\alpha, \beta)$ – расстояние по Хэммингу между векторами α, β

Для каждого такого кода $\alpha \in R'$ найдем не закодированное состояние p_α , наиболее сильно связанное с состояниями, закодированными соседними кодами, то есть состояние, обеспечивающее максимум критерия – суммы коэффициентов связи

$$L(\alpha, p) = \sum_{q: d(\alpha, p(q))=1} F(p, q).$$

Шаг 3. Из всех сумм коэффициентов связи $L(\alpha, p_\alpha)$ выберем максимальную сумму $L(\alpha', p_{\alpha'})$. Положим $\rho(p_{\alpha'}) = \alpha'$ и добавим код α' в множество S . Если не все состояния закодированы, перейдем на шаг 2.

Пример. Продолжим рассмотрение предыдущего примера, для которого получена матрица связей

	1	2	3	4	5	6	Σ
1		2	1	1	0	1	5
2	2		1	1	1	0	5
3	1	1		2	0	0	4
4	1	1	2		1	1	6
5	0	1	0	1		2	4
6	1	0	0	1	2		4

Процесс кодирования будем демонстрировать на матрице Грея.

Шаг 1. Закодируем состояние 4 кодом 000, так как сумма коэффициентов связей для него максимальна (равна 6).

4			

Шаг 2. Выделим на матрице клетки, соседние с занятой.

4	•		•
•			

Из матрицы связей видно, что

$$L(100,1) = F(4,1) = 1;$$

$$L(100,2) = F(4,2) = 1;$$

$$L(100,3) = F(4,3) = 2 \rightarrow \max;$$

$$L(100,5) = F(4,5) = 1;$$

$$L(100,6) = F(4,6) = 1.$$

Для остальных выделенных кодов получатся те же значения, поскольку все они соседние с кодом 000, приписанным состоянию 4.

Шаг 3. Максимум критерия для кода 100 достигается на состоянии 3, значит, кодируем состояние 3 кодом 100. Возвращаемся на шаг 2.

Шаг 2. Выделим на матрице клетки, соседние с занятыми.

4	•		•
3	•		•

Из матрицы связей видно, что

$$\begin{aligned}
 L(001,1) &= F(4,1) = 1 \rightarrow \max; & L(101,1) &= F(3,1) = 1 \rightarrow \max; \\
 L(001,2) &= F(4,2) = 1 \rightarrow \max; & L(101,2) &= F(3,2) = 1 \rightarrow \max; \\
 L(001,5) &= F(4,5) = 1 \rightarrow \max; & L(101,5) &= F(3,5) = 0; \\
 L(001,6) &= F(4,6) = 1 \rightarrow \max. & L(101,6) &= F(3,6) = 0.
 \end{aligned}$$

Результаты для кодов 010 и 110 аналогичны результатам для кодов 001 и 101 соответственно.

Шаг 3. Максимальные значения критерия для кодов 001 и 101 одинаковы, значит, выбираем любой код, например, 001. Кодируем состояние 1 кодом 001 (можно выбрать любое состояние, значения критерия для всех состояний одинаковы). Возвращаемся на шаг 2.

Шаг 2. Выделим на матрице клетки, соседние с занятыми.

4	1	•	•
3	•		•

Из матрицы связей видно, что

$$\begin{aligned}
 L(011,2) &= F(1,2) = 2 \rightarrow \max; & L(010,2) &= F(4,2) = 1 \rightarrow \max; \\
 L(011,5) &= F(1,5) = 0; & L(010,5) &= F(4,5) = 1 \rightarrow \max; \\
 L(011,6) &= F(1,6) = 1. & L(010,6) &= F(4,6) = 1 \rightarrow \max.
 \end{aligned}$$

$$L(101,2) = F(1,2) + F(3,2) = 2 + 1 = 3 \rightarrow \max;$$

$$L(101,5) = F(1,5) + F(3,5) = 0 + 0 = 0;$$

$$L(101,6) = F(1,6) + F(3,6) = 1 + 0 = 1.$$

$$L(110,2) = F(3,2) = 1 \rightarrow \max;$$

$$L(110,5) = F(3,5) = 0;$$

$$L(110,6) = F(3,6) = 0.$$

Шаг 3. Максимальное значение критерия получено для кода 101. Кодируем состояние 3 кодом 101. Возвращаемся на шаг 2.

Шаг 2. Выделим на матрице клетки, соседние с занятыми.

4	1	•	•
3	2	•	•

$$L(011,5) = F(1,5) = 0; \quad L(010,5) = F(4,5) = 1 \rightarrow \max;$$

$$L(011,6) = F(1,6) = 1 \rightarrow \max. \quad L(010,6) = F(4,6) = 1 \rightarrow \max.$$

$$L(111,5) = F(2,5) = 1 \rightarrow \max; \quad L(110,5) = F(3,5) = 0 \rightarrow \max;$$

$$L(111,6) = F(2,6) = 0. \quad L(110,6) = F(3,6) = 0 \rightarrow \max.$$

Шаг 3. Максимальное значение критерия получено для кодов 011, 010 и 111, для них оно одинаково и равно 1. Кодируем состояние 6 кодом 011. Возвращаемся на шаг 2.

Шаг 2. Выделим на матрице клетки, соседние с занятыми.

4	1	6	•
3	2	•	•

$$L(010,5) = F(4,5) + F(6,5) = 1 + 2 = 3 \rightarrow \max;$$

$$L(111,5) = F(2,5) + F(6,5) = 1 + 2 = 3 \rightarrow \max;$$

$$L(110,5) = F(4,5) = 1.$$

Шаг 3. Максимальное значение критерия получено для кодов 010, 111, для них оно одинаково и равно 3. Кодируем состояние 5 кодом 010. Все состояния закодированы.

4	1	6	5
3	2		

Этап IV. Проверим, все ли K -множества являются связными. Несвязные множества свяжем, добавив дополнительные состояния с кодами из множества R , соблюдая правило

$$\forall x \in X, \forall q, p \in Q: K_{x,q} \cap K_{x,p} = \emptyset.$$

Если не все K -множества связаны, то либо увеличим длину кода на 1, либо вернемся на этап III и получим другое кодирование.

Пример. В предыдущем примере было получено кодирование

4	1	6	5
3	2		

Рассмотрим K -множества

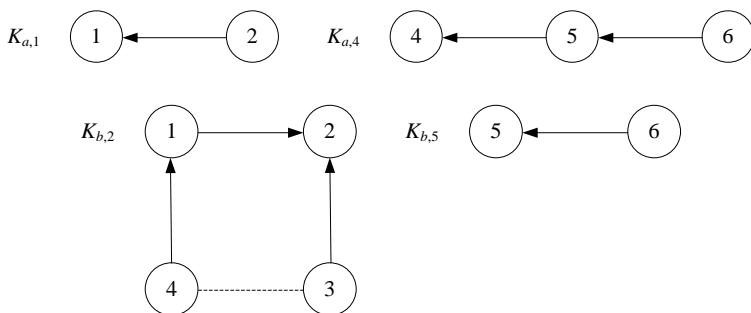
$$\begin{aligned} K_{a,1} &= \{1, 2\} & K_{a,4} &= \{4, 5, 6\} \\ K_{b,2} &= \{1, 2, 3, 4\} & K_{b,5} &= \{5, 6\} \\ K_{c,3} &= \{3, 4\} & K_{c,5} &= \{2, 5\} & K_{c,6} &= \{1, 6\} \end{aligned}$$

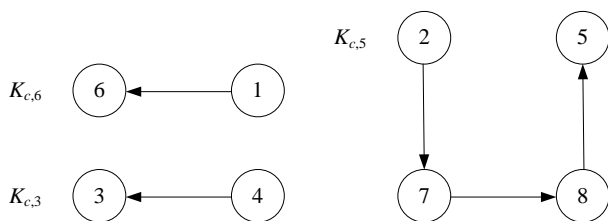
Несвязным является лишь множество $K_{c,5} = \{2, 5\}$. Расширяем его за счет состояний 7 и 8, получаем $K_{c,5} = \{2, 5, 7, 8\}$.

4	1	6	5
3	2	7	8

Этап V. Заменяем каждый переход $p \xrightarrow{x} q$ последовательностью элементарных переходов $p \xrightarrow{x} p_1 \xrightarrow{x} \dots \xrightarrow{x} p_l \xrightarrow{x} q$, где $p_1, \dots, p_l \in K_{x,q}$. Для этого множество $K_{x,q}$ можно представить в виде графа, вершинами которого являются состояния, а каждое ребро соединяет два состояния, закодированных соседними кодами. Затем находится остовное дерево графа, и его ребра ориентируются по направлению к состоянию q . Цепь $\langle p, q \rangle$ задает последовательность элементарных переходов.

Пример. Строим графы K -множеств.





Пунктирными линиями отмечены ребра, удаленные при построении остова дерева.

Рассмотрим исходную таблицу переходов.

$X \backslash Q$	1	2	3	4	5	6
a	1	1	3	4	4	4
b	2	2	2	2	5	5
c	6	5	3	3	5	6

Добавим в нее состояния 7 и 8 и преобразуем таблицу переходов в таблицу элементарных переходов.

$X \backslash Q$	1	2	3	4	5	6	7	8
a	1	1	3	4	4	5	—	—
b	2	2	2	1	5	5	—	—
c	6	7	3	3	5	6	8	5

4.2. Кодирование, разделяющее переходы

Определение. Автоматом с нормальными функциями переходов и выходов называется автомат, для которого

$$\forall x \in X, \forall q \in Q: \begin{cases} \psi(x, \psi(x, q)) = \psi(x, q); \\ \varphi(x, \psi(x, q)) = \varphi(x, q). \end{cases}$$

Иными словами, если под действием символа x автомат из состояния q переходит в состояние $\psi(x, q)$ и выдает сигнал $\varphi(x, q)$, то при неизменном входном сигнале состояние и выход не меняются.

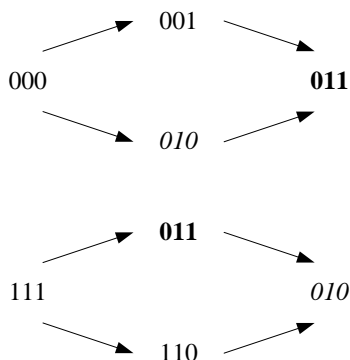
4.2.1. Постановка задачи противогоночного кодирования состояний автомата с нормальными функциями переходов и выходов

Рассмотрим автомат с нормальными функциями переходов и выходов $A = \langle X, Q, Y, \psi, \varphi \rangle$.

Пусть в автомате существуют два перехода по входному символу x : $q \xrightarrow{x} q'$ и $p \xrightarrow{x} p'$. Выберем кодирование

q	000
q'	011
p	111
p'	010

Учитывая, что триггеры меняют свое значение не обязательно одновременно, рассмотрим все варианты переходов, т.е. возможные последовательности элементарных переходов.



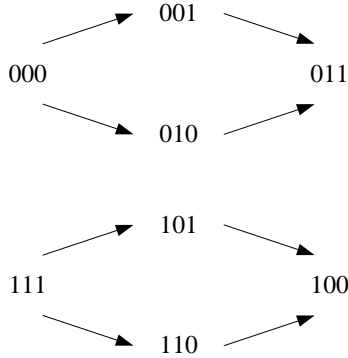
Здесь одинаковыми шрифтами выделены одинаковые коды.

Из рисунка видно, что при переходе из состояния $q(000)$ в $q'(011)$ автомат может попасть в состояние $p'(010)$, но это состояние является устойчивым, следовательно, автомат может остаться в состоянии $p'(010)$ вместо того, чтобы перейти в состояние $q'(011)$, как это предписано таблицей переходов. Аналогично можно показать, что автомат может попасть из состояния $p(111)$ в состояние $q'(011)$ вместо предписанного $p'(010)$. Назовем это явление *перепутыванием переходов*.

Чтобы перепутывания переходов не происходило, надо, чтобы никакие цепочки элементарных переходов $q \xrightarrow{x} q'$ и $p \xrightarrow{x} p'$ не содержали одинаковых кодов. Например, пусть кодирование имеет вид

q	000
q'	011
p	111
p'	100

Рассмотрим все последовательности элементарных переходов



Они не содержат одинаковых кодов, поскольку первая компонента кода принимает значение 0 в кодах состояний q и q' (а значит, и во всех промежуточных векторах из цепочки элементарных переходов $q \xrightarrow{x} q'$), и значение 1 – в кодах состояний p и p' (а значит, и во всех промежуточных векторах из цепочки элементарных переходов $p \xrightarrow{x} p'$). Поэтому вектора из цепочек $q \xrightarrow{x} q'$ и $p \xrightarrow{x} p'$ всегда ортогональны по первой компоненте. При таком кодировании переходы происходят в предписанные состояния.

Обозначим через $[\alpha, \beta]$ интервал минимальной мощности, содержащий булевы векторы α, β и обобщим проведенные рассуждения в следующем утверждении.

Утверждение. Кодирование $\rho: Q \rightarrow B^k$ является противогоночным тогда и только тогда, когда для любой пары различных переходов $q \xrightarrow{x} q'$ и $p \xrightarrow{x} p'$, где $q \neq p$ и $q' \neq p'$, верно

$$[\rho(q), \rho(q')] \cap [\rho(p), \rho(p')] = \emptyset. \quad (*)$$

Определение. *Кодирующей матрицей* назовем булеву матрицу с различными столбцами, каждая строка которой соответствует ком-

поненте кода, а каждый столбец – состоянию автомата.

Обозначим кодирующую матрицу через C . Столбцы матрицы C суть коды состояний автомата.

Для выполнения условия (*) необходимо и достаточно, чтобы для каждой пары переходов $q \xrightarrow{x} q'$ и $p \xrightarrow{x} p'$, где $q \neq p$ и $q' \neq p'$, в матрице C имелась строка, содержащая 0 в столбцах q, q' и 1 в столбцах p, p' .

Теперь можно поставить задачу: для автомата с нормальными функциями переходов и выходов найти кодирующую матрицу C , обеспечивающую условие (*), с минимальным числом строк.

4.2.2. Матрица условий

Выясним, каким образом выполнить при кодировании условие (*). Сопоставим каждой паре переходов $q \xrightarrow{x} q'$ и $p \xrightarrow{x} p'$ троичный вектор, содержащий 0 в компонентах q, q' и 1 в компонентах p, p' , остальные компоненты вектора оставим неопределенными. Совокупность таких векторов образует *матрицу условий* R .

Пример. Составим матрицу условий для автомата

$X \backslash Q$	1	2	3	4	5	6
a	1	1	3	4	4	4
b	2	2	2	2	5	5
c	6	5	3	3	5	6

Построим таблицу, в первый столбец запишем пары переходов, следующим 6 столбцам припишем состояния автомата. Запишем 0 в столбцах, соответствующих состояниям первого перехода, 1 – второго, и прочерк в остальных столбцах.

переходы	1	2	3	4	5	6
$a: 1 \rightarrow 1, 3 \rightarrow 3$	0	–	1	–	–	–
$2 \rightarrow 1, 3 \rightarrow 3$	0	0	1	–	–	–
$1 \rightarrow 1, 4 \rightarrow 4$	0	–	–	1	–	–
$1 \rightarrow 1, 5 \rightarrow 4$	0	–	–	1	1	–
$1 \rightarrow 1, 6 \rightarrow 4$	0	–	–	1	–	1
$2 \rightarrow 1, 4 \rightarrow 4$	0	0	–	1	–	–
$2 \rightarrow 1, 5 \rightarrow 4$	0	0	–	1	1	–
$2 \rightarrow 1, 6 \rightarrow 4$	0	0	–	1	–	1
$3 \rightarrow 3, 4 \rightarrow 4$	–	–	0	1	–	–

b:	3→3, 5→4	–	–	0	1	1	–
	3→3, 6→4	–	–	0	1	–	1
	1→2, 5→5	0	0	–	–	1	–
	1→2, 6→5	0	0	–	–	1	1
	2→2, 5→5	–	0	–	–	1	–
	2→2, 6→5	–	0	–	–	1	1
	3→2, 5→5	–	0	0	–	1	–
	3→2, 6→5	–	0	0	–	1	1
	4→2, 5→5	–	0	–	0	1	–
c:	4→2, 6→5	–	0	–	0	1	1
	1→6, 2→5	0	1	–	–	1	0
	1→6, 5→5	0	–	–	–	1	0
	6→6, 2→5	–	1	–	–	1	0
	6→6, 5→5	–	–	–	–	1	0
	1→6, 3→3	0	–	1	–	–	0
	1→6, 4→3	0	–	1	1	–	0
	6→6, 3→3	–	–	1	–	–	0
	6→6, 4→3	–	–	1	1	–	0
	2→5, 3→3	–	0	1	–	0	–
	5→5, 4→3	–	–	1	1	0	–
	2→5, 3→3	–	0	1	–	0	–
	5→6, 4→3	–	–	1	1	0	–

Определение. Троичный вектор α *покрывает* троичный вектор β , если α получается из β заменой некоторых неопределенных компонент на определенные, и, возможно, инвертированием компонент вектора.

Пример. В паре $\alpha = 10-1-0$;
 $\beta = 01-----$ вектор α покрывает вектор β .

Чтобы матрица C обеспечивала выполнение условия (*) для всех пар переходов, требуется, чтобы для каждой строки матрицы условий R существовала покрывающая ее строка матрицы C . Отношение покрытия транзитивно, т.е. если α покрывает β и β покрывает γ , то α покрывает γ . Следовательно, можно сократить матрицу условий, удалив из нее строки, покрываемые другими строками. Заметим также, что если α покрывает β , то и вектор, полученный из вектора α инверсией всех компонент, также покрывает

вает вектор β .

Пример. Сократим матрицу условий из предыдущего примера. Удаляемые строки отметим символом \leq , приписав также справа в скобках обозначение строки, покрывающей данную строку. В результате матрица условий принимает вид

1	2	3	4	5	6		
0	–	1	–	–	–	\leq	(α)
0	0	1	–	–	–	α	
0	–	–	1	–	–	\leq	(β)
0	–	–	1	1	–	\leq	(β)
0	–	–	1	–	1	\leq	(γ)
0	0	–	1	–	–	\leq	(β)
0	0	–	1	1	–	β	
0	0	–	1	–	1	γ	
–	–	0	1	–	–	\leq	(δ)
–	–	0	1	1	–	δ	
–	–	0	1	–	1	ε	
0	0	–	–	1	–	\leq	(β)
0	0	–	–	1	1	ζ	
–	0	–	–	1	–	\leq	(β)
–	0	–	–	1	1	\leq	(ζ)
–	0	0	–	1	–	\leq	(η)
–	0	0	–	1	1	η	
–	0	–	0	1	–	\leq	(θ)
–	0	–	0	1	1	θ	
0	1	–	–	1	0	κ	
0	–	–	–	1	0	\leq	(κ)
–	1	–	–	1	0	\leq	(κ)
–	–	–	–	1	0	\leq	(κ)
0	–	1	–	–	0	\leq	(λ)
0	–	1	1	–	0	λ	
–	–	1	–	–	0	\leq	(λ)
–	–	1	1	–	0	\leq	(λ)
–	0	1	–	0	–	\leq	(μ)
–	0	1	1	0	–	μ	
–	–	1	–	0	–	\leq	(μ)
–	–	1	1	0	–	\leq	(μ)

Окончательно сокращенная матрица условий R' принимает вид

1	2	3	4	5	6	
0	0	1	–	–	–	α
0	0	–	1	1	–	β
0	0	–	1	–	1	γ
–	–	0	1	1	–	δ
–	–	0	1	–	1	ε
0	0	–	–	1	1	ζ
–	0	0	–	1	1	η
–	0	–	0	1	1	θ
0	1	–	–	1	0	κ
0	–	1	1	–	0	λ
–	0	1	1	0	–	μ

4.2.3. Точный метод решения задачи противогоночного кодирования

Начало. Задан автомат с нормальными функциями переходов и выходов.

Шаг 1. Построим матрицу условий R .

Шаг 2. Сократим матрицу условий, получим матрицу R' .

Шаг 3. Найдем множество M булевых векторов, покрывающих строки матрицы R' . Сократим множество, оставив по одному из противоположных векторов, получим множество M' .

Шаг 4. Построим булеву матрицу, столбцам которой сопоставим строки матрицы R' , а строкам – векторы множества M' . На пересечении строки и столбца запишем единицу, если вектор, приписанный строке, покрывает вектор, приписанный столбцу.

Шаг 5. Найдем кратчайшее покрытие матрицы. Векторы, сопоставленные строкам покрытия, запишем в кодирующую матрицу C . При необходимости добавим в матрицу C строки так, чтобы все ее столбцы были различны.

Пример. Рассмотрим автомат из подраздела 4.1.1

$X \backslash Q$	1	2	3	4
a	1	3	3	1
b	2	2	4	4
c	1	1	1	1
d	1	2	1	4

Шаг 1. Строим матрицу условий R .

переходы		1	2	3	4
a :	$1 \rightarrow 1, 2 \rightarrow 3$	0	1	1	–
	$1 \rightarrow 1, 3 \rightarrow 3$	0	–	1	–
	$4 \rightarrow 1, 2 \rightarrow 3$	0	1	1	0
	$4 \rightarrow 1, 3 \rightarrow 3$	0	–	1	0
b :	$1 \rightarrow 2, 3 \rightarrow 4$	0	0	1	1
	$1 \rightarrow 2, 4 \rightarrow 4$	0	0	–	1
	$2 \rightarrow 2, 3 \rightarrow 4$	–	0	1	1
	$2 \rightarrow 2, 4 \rightarrow 4$	–	0	–	1
d :	$1 \rightarrow 1, 2 \rightarrow 2$	0	1	–	–
	$1 \rightarrow 1, 4 \rightarrow 4$	0	–	–	1
	$2 \rightarrow 2, 4 \rightarrow 4$	–	0	–	1
	$3 \rightarrow 1, 2 \rightarrow 2$	0	1	0	–
	$3 \rightarrow 1, 4 \rightarrow 4$	0	–	0	1

Шаг 2. Выделяем строки, покрываемые другими строками

1	2	3	4	
0	1	1	–	$\leq (\alpha)$
0	–	1	–	$\leq (\alpha)$
0	1	1	0	α
0	–	1	0	$\leq (\alpha)$
0	0	1	1	β
0	0	–	1	$\leq (\beta)$
–	0	1	1	$\leq (\beta)$
–	0	–	1	$\leq (\beta)$
0	1	–	–	$\leq (\alpha)$
0	–	–	1	$\leq (\beta)$
–	0	–	1	$\leq (\beta)$
0	1	0	–	γ
0	–	0	1	δ

Получаем сокращенную матрицу условий R'

1	2	3	4
0	1	1	0
0	0	1	1
0	1	0	–
0	–	0	1

Шаг 3. Находим множество M всех векторов, покрывающих хотя бы одну строку из R' . Для этого заменяем все неопределенные компоненты векторов из R' всеми возможными комбинациями нулей и единиц. В таблице слева перечислены строки из R' , справа – покрывающие их векторы.

R'	M
0110	0110
0011	0011
010–	0100 0101
0–01	0001 0101

Шаг 4. Строим булеву матрицу, столбцам которой сопоставим строки матрицы R' , а строкам – векторы множества M' . Из одинаковых либо противоположных векторов M оставляем один. На пересечении строки и столбца записываем единицу, если вектор, приписанный строке, покрывает вектор, приписанный столбцу

	0110	0011	010–	0–01
0110	1			
0011		1		
0100			1	
0101			1	1
0001				1

Шаг 5. Кратчайшее покрытие матрицы составляют строки $\{0110, 0011, 0101\}$, значит, кодирующая матрица имеет вид

1	2	3	4
0	1	1	0
0	0	1	1
0	1	0	1

Для больших мощностей множеств R' и M' задача является очень трудоемкой, поэтому рассмотрим приближенный метод противогоночного кодирования. Он обеспечивает отсутствие опасных состязаний, но не гарантирует, что длина кода будет минимальной.

4.2.4. Приближенный метод решения задачи противогоночного кодирования

Начало. Задан автомат с нормальными функциями переходов и выходов.

Шаг 1. Построим матрицу условий R .

Шаг 2. Сократим матрицу условий, получим матрицу R' .

Шаг 3. Выберем строку матрицы R' , содержащую наибольшее число определенных компонент. Доопределим ее так, чтобы покрыть как можно больше строк из R' , и добавим в кодирующую матрицу C .

Шаг 4. Удалим из R' строки, покрываемые строкой, добавленной в C . Если R' не пуста, перейдем на шаг 3.

Шаг 5. При необходимости добавим в матрицу C строки так, чтобы все ее столбцы были различны.

Пример. Продемонстрируем алгоритм на примере из подраздела 4.2.2. Автомат задан таблицей переходов

$X \backslash Q$	1	2	3	4	5	6
a	1	1	3	4	4	4
b	2	2	2	2	5	5
c	6	5	3	3	5	6

Для него в подразделе 4.2.2 уже выполнены шаги 1–2, т. е. получена матрица сокращенная матрица условий R' .

1	2	3	4	5	6	
0	0	1	—	—	—	α
0	0	—	1	1	—	β
0	0	—	1	—	1	γ
—	—	0	1	1	—	δ
—	—	0	1	—	1	ε
0	0	—	—	1	1	ζ
—	0	0	—	1	1	η
—	0	—	0	1	1	θ
0	1	—	—	1	0	κ
0	—	1	1	—	0	λ
—	0	1	1	0	—	μ

Шаг 3. Рассмотрим строку β . Она имеет 4 доопределения. В таблице выписаны доопределения и строки матрицы R' , которые они покрывают.

000110	β, δ
000111	$\beta, \gamma, \delta, \varepsilon, \zeta, \eta$
001110	α, β, λ
001111	$\alpha, \beta, \gamma, \zeta$

Из таблицы видно, что наибольшее количество строк покрывает доопределение 000111, добавляем эту строку в кодирующую матрицу, которая принимает вид

1	2	3	4	5	6
0	0	0	1	1	1

Шаг 4. Удаляем из матрицы R' строки, покрываемые вектором 000111. Матрица принимает вид

1	2	3	4	5	6	
0	0	1	—	—	—	α
—	0	—	0	1	1	θ
0	1	—	—	1	0	κ
0	—	1	1	—	0	λ
—	0	1	1	0	—	μ

Матрица не пуста, идем на шаг 3.

Шаг 3. Рассмотрим строку θ . Она имеет 4 доопределения. В таблице выписаны доопределения и строки матрицы R' , которые они покрывают.

000011	Θ
001011	α, θ
100011	θ, λ
101011	Θ

Из таблицы видно, что наибольшее количество строк покрывает два доопределения, выбираем 001011, добавляем эту строку в кодирующую матрицу, которая принимает вид

1	2	3	4	5	6
0	0	0	1	1	1
0	0	1	0	1	1

Шаг 4. Удаляем из матрицы R' строки, покрываемые вектором 001011. Матрица принимает вид

1	2	3	4	5	6	
0	1	—	—	1	0	к
0	—	1	1	—	0	λ
—	0	1	1	0	—	μ

Матрица не пуста, идем на шаг 3.

Шаг 3. Рассмотрим строку к. Она имеет 4 доопределения.

010010	к, μ
010110	к
011010	к
011110	к, λ

Выбираем 010010, добавляем эту строку в кодирующую матрицу, которая принимает вид

1	2	3	4	5	6
0	0	0	1	1	1
0	0	1	0	1	1
0	1	0	0	1	0

Шаг 4. Удаляем из матрицы R' строки, покрываемые вектором 001011. Матрица принимает вид

1	2	3	4	5	6	
0	—	1	1	—	0	λ

Матрица не пуста, идем на шаг 3.

Шаги 3–5. Рассмотрим строку λ. Она имеет 4 доопределения. Можно выбрать любое, так как требуется покрыть лишь одну строку. К тому же, столбцы кодирующей матрицы уже различны, и это требование выполнять не нужно. Добавляем строку 001100 в коди-

рующую матрицу, которая принимает вид

1	2	3	4	5	6
0	0	0	1	1	1
0	0	1	0	1	1
0	1	0	0	1	0
0	0	1	1	0	0

Матрица R' пуста, все столбцы матрицы C различны, значит, получено противогоночное кодирование.

4.3. Задачи

Задача 1. Выполнить противогоночное кодирование состояний асинхронного автомата несколькими методами, сравнить результаты по длине кода.

a)

$X \backslash Q$	1	2	ψ 3	4	5	6
a	1	1	1	4	4	6
b	2	2	6	5	5	6
c	5	4	3	4	5	3
d	4	3	3	4	3	4

b)

$X \backslash Q$	1	2	ψ 3	4	5	6
a	2	2	4	4	4	6
b	3	3	3	5	5	6
c	1	2	3	1	3	1
d	1	3	3	4	5	4

c)

$X \backslash Q$	1	2	ψ 3	4	5	6
a	6	6	3	4	4	6
b	1	2	1	2	2	2
c	1	1	5	5	5	5
d	4	3	3	4	5	3

Задача 2. Синтезировать наиболее простые последовательностные схемы, реализующие автоматы из задачи 1. В качестве элемента памяти использовать RS -триггер.

5. Кодирование в синхронных схемах

Рассмотрим сначала критерии качества кодирования состояний синхронной схемы. Кодирование считается «хорошим», если:

- Функции возбуждения триггеров имеют «простой» вид. Например, если функции являются ДНФ, то предпочтительными являются ДНФ наименьшей длины и ранга.
- Переключение триггеров с 0 на 1 или с 1 на 0 происходит как можно реже.

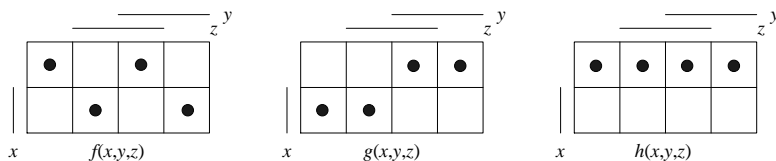
Первое требование связано с тем, что схема, построенная по таким формулам, содержит небольшое число элементов и связей. Второе требование кроме того направлено на так называемое энергосберегающее кодирование: чем меньше переключаются триггеры, тем меньше энергии потребляет схема.

Примитивным решением этой задачи является перебор всех возможных кодировок и выбор из них наилучшей по данным критериям. Но так как перебор очень велик, то рассмотрим некоторые эвристические методы.

5.1. Кодирование состояний, упрощающее функции возбуждения триггеров

Практика показывает, что более чем проще структурные функции переходов, тем проще построенная по ним схема. Здесь предполагается, что функции переходов заданы ДНФ, а за сложность ДНФ принимается ее ранг. Будем считать, что функция тем проще, чем больше пар соседних наборов в ее характеристических множествах M_f^0 и M_f^1 (множества наборов, на которых функция принимает значение 0 и 1), потому что в этом случае велика вероятность того, что наборы из множества M_f^1 образуют интервалы малого ранга.

Пример. Рассмотрим три булевы функции, заданных матрицами Грея. Эти функции имеют одинаковые мощности характеристических множеств M^0 и M^1 .



В характеристических множествах функции $f(x, y, z)$ нет ни одной пары соседних наборов. Характеристические множества функции $g(x, y, z)$ содержат по две пары соседних наборов, а множества функции $h(x, y, z)$ – по четыре пары. Очевидно, что ранг и длина кратчайшей ДНФ максимальны у функции $f(x, y, z)$ и минимальны у функции $h(x, y, z)$:

$$\check{\mathbf{A}}\check{\mathbf{I}}\hat{\mathbf{O}}_f = \bar{x}\bar{y}\bar{z} \vee x\bar{y}z \vee \bar{x}yz \vee xy\bar{z};$$

$$\check{\mathbf{A}}\check{\mathbf{I}}\hat{\mathbf{O}}_g = x\bar{y} \vee \bar{x}y;$$

$$\check{\mathbf{A}}\check{\mathbf{I}}\hat{\mathbf{O}}_h = \bar{x}.$$

Рассмотрим условия, которым должно удовлетворять кодирование, чтобы обеспечить большое число соседних наборов в характеристических множествах структурных функций переходов.

Рассмотрим произвольное состояние автомата $q \in Q$. Пусть переход в это состояние под действием символа $x \in X$ происходит из состояний q_1, q_2, \dots, q_r , то есть эти состояния образуют K -множество $K_{x,q} = \{q_1, \dots, q_r\}$. Пусть коды этих состояний образуют интервал $I_{x,q}$ или покрываются интервалом $I_{x,q}$, не содержащим коды других состояний. Сопоставим состоянию q код $\rho(q) = \sigma_1 \dots \sigma_k$. Тогда при входном наборе x интервал $I_{x,q}$ принадлежит множествам $M_{\psi_i}^{\sigma_i}$, $i = \overline{1, k}$. Таким образом, желательно, чтобы $\forall x \in X, \forall q \in Q$ множество $K_{x,q}$ образовывало интервал, либо покрывалось интервалом, не содержащим коды других состояний.

Пусть автомат под действием соседних наборов α, β переходит из состояния q в состояния q_1, q_2 соответственно. Пару $\{q_1, q_2\}$ в этом случае называют *парой последователей*. Пусть $\rho(q) = \gamma$ – код состояния q . Тогда наборы $\alpha\gamma, \beta\gamma$ также будут соседними. Пусть $\rho(q_1) = \delta = \delta_1 \dots \delta_k$, $\rho(q_2) = \varepsilon_1 \dots \varepsilon_k$. Тогда соседние наборы $\alpha\gamma \in M_{\psi_i}^{\delta_i}$, $\beta\gamma \in M_{\psi_i}^{\varepsilon_i}$, $i = \overline{1, k}$. Если же $\delta_i = \varepsilon_i = \eta$, то соседние наборы $\alpha\gamma, \beta\gamma$ оба принадлежат множеству $M_{\psi_i}^{\eta}$. Таким образом, желательно так закодировать все пары последователей q_1, q_2 , чтобы их коды имели

как можно меньше ортогональных компонент, то есть были бы соседними.

Эти требования противоречивы, поэтому поставим задачу следующим образом: закодировать состояния автомата так, чтобы вышеизложенные требования выполнялись максимально. Рассмотрим некоторые эвристические методы таких кодирований.

5.1.1. Метод Хамфри

Начало. Задан автомат $A = \langle X, Q, \psi \rangle$. Требуется закодировать его состояния кодами длины k .

Шаг 1. Расширим множество Q , добавив в него $2^k - |Q|$ фиктивных состояний.

Шаг 2. Каждое K – множество расширим за счет фиктивных состояний так, чтобы общее число состояний было равно целой степени двойки. Расширенную таким образом группу состояний назовем интервалоподобной.

Шаг 3. Нарисуем матрицу Грея для k переменных. Выпишем на ней последовательно все состояния автомата, соблюдая следующие правила.

- Сначала выписываются интервалоподобные группы в порядке невозрастания их мощностей. При этом если хотя бы одно состояние рассматриваемой группы уже выписано, то вся группа не выписывается. Интервал, образованный группой на матрице, выделяется.
- Далее выписываются на матрице пары последователей в произвольном порядке, но если хотя бы одно состояние из пары уже выписано ранее, то пара не выписывается. Интервал, образованный парой на матрице, выделяется.
- Далее в произвольном порядке выписываются остальные состояния, в том числе фиктивные.

Шаг 4. Внутри всех выделенных интервалов, на множествах интервалов одинаковой мощности и на множестве отдельных состояний выполняем все возможные перестановки с тем, чтобы максимизировать число пар последователей, расположенных в соседних клетках матрицы.

Конец. Каждому нефиктивному состоянию приписывается код клетки матрицы, в которой оно расположено.

Пример. Закодируем методом Хамфри состояния автомата, заданного таблицей переходов.

$X \backslash Q$	1	2	3	4	5	6	7	8	9	10	11
0	1	1	5	7	8	9	10	11	1	1	1
1	2	3	4	6	7	9	9	10	1	1	1

Шаг 1. Расширим множество Q , добавив в него 5 фиктивных состояний, которые в дальнейшем будем обозначать символом \times .

Шаг 2. Выпишем все K -множества, мощность которых больше единицы, и расширим их за счет фиктивных состояний до интервалоподобных групп.

$$K_{1,0} = \{1, 2, 9, 10, 11\} = \{1, 2, 9, 10, 11, \times, \times, \times\}$$

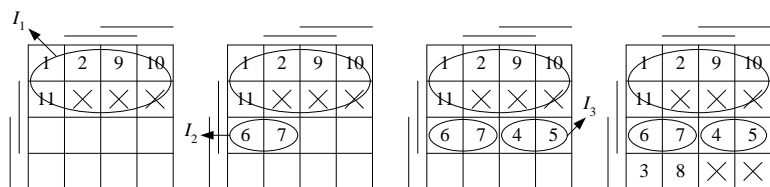
$$K_{1,1} = \{9, 10, 11\} = \{9, 10, 11, \times\}$$

$$K_{1,9} = \{6, 7\} = \{6, 7\}$$

Выпишем все пары последователей.

$$\{1, 2\}, \{1, 3\}, \{4, 5\}, \{6, 7\}, \{7, 8\}, \{8, 9\}, \{9, 10\}, \{10, 11\}$$

Шаг 3. Нарисуем матрицу Грея для 4 переменных. Выпишем на ней последовательно все состояния автомата: сначала интервалоподобные группы, затем пары последователей, затем отдельные состояния. Сначала выпишем группу наибольшей мощности $\{1, 2, 9, 10, 11, \times, \times, \times\}$. Она образует интервал I_1 . Группу $\{9, 10, 11, \times\}$ не выписываем, так как она пересекается с уже выписанной группой. Затем выписываем группу $\{6, 7\}$, она образует интервал I_2 . Из пар последователей можно выписать только пару $\{4, 5\}$, так как остальные пары пересекаются с множеством уже выписанных состояний. Эта пара образует интервал I_3 . Далее выписываем остальные состояния 3 и 8 и два фиктивных состояния. Процесс выписывания состояний показан на рисунке.



Шаг 4. Рассмотрим все пары последователей. Коды состояний в парах $\{1, 2\}$, $\{1, 3\}$, $\{4, 5\}$, $\{6, 7\}$, $\{7, 8\}$, $\{9, 10\}$ являются соседними. Найдем такую перестановку состояний, чтобы пары кодов $\{8, 9\}$, $\{10, 11\}$ также были соседними. При этом переставлять можно состояния внутри интервалов, интервалы одинаковой мощности и от-

дельные состояния. Такому условию удовлетворяет, например, следующая перестановка.

1	2	10	9
×	×	11	×
4	5	6	7
3	×	×	8

Каждому состоянию приписывается код клетки матрицы, в которой оно расположено. Таким образом, кодирование имеет вид

1	0000
2	0001
3	1000
4	1100
5	1101
6	1111
7	1110
8	1010
9	0010
10	0011
11	0111

Примечание. Наиболее трудоемким шагом алгоритма является шаг 4. Его можно не выполнять, но его выполнение может только улучшить кодирование.

5.1.2. Метод Армстронга

Так же, как и в предыдущем методе, предполагается, что имеются некоторые пары состояний, которые желательно закодировать соседними кодами.

Определение. *Весом* пары состояний $\{p, q\}$ (обозначается $w(p, q)$) назовем количество раз, которые эта пара встречается в K -множествах, в сумме с количеством раз, когда эта пара является парой последователей.

Пример. Рассмотрим автомат, заданный таблицей переходов.

$X \backslash Q$	1	2	3	4	5	6
00	1	1	5	4	2	2
01	3	3	3	6	6	6
10	2	5	4	4	4	6

Перечислим K -множества автомата, содержащие более одного состояния.

$$K_{00,1} = \{1, 2\}, \quad K_{00,2} = \{5, 6\};$$

$$K_{01,3} = \{1, 2, 3\}, \quad K_{01,6} = \{4, 5, 6\};$$

$$K_{10,4} = \{3, 4, 5\}.$$

Теперь перечислим пары состояний, сопоставив каждой паре количество раз, когда она встречается в K -множествах.

{1,2}	{1,3}	{2,3}	{3,4}	{3,5}	{4,5}	{4,6}	{5,6}
2	1	1	1	1	2	1	2

Перечислим пары последовательностей. Сопоставим каждой паре количество раз, когда она является парой последовательностей.

{1,3}	{3,5}	{4,6}	{2,6}	{1,2}	{1,5}	{4,5}	{2,4}
2	1	1	3	1	1	1	1

Построим матрицу весов, строкам и столбцам которой сопоставим состояния автомата. Элементу (p, q) сопоставляем вес этой пары.

	1	2	3	4	5	6
1		3	3	0	1	0
2	3		1	1	0	3
3	3	1		1	2	0
4	0	1	1		3	2
5	1	0	2	3		2
6	0	3	0	2	2	

Определение. Будем говорить, что состояния (p, q) связаны, если $w(p, q) > 0$.

Чем больше вес пары, тем важнее закодировать ее состояния соседними кодами. Таким образом, целью кодирования является минимизация критерия

$$W = \sum_{p, q \in Q} w(p, q) \times d(\rho(p), \rho(q)),$$

где $\rho(q)$ – код состояния q , $d(\rho(p), \rho(q))$ – расстояние по Хэммингу между кодами состояний p и q .

Данная задача является весьма сложной и требует перебора. Армстронг предложил располагать состояния автомата в строку таким образом, чтобы минимизировать величину

$$W' = \sum_{p,q \in Q} w(p,q)l(p,q),$$

где $l(p,q)$ – расстояние между состояниями (число промежуточных состояний, увеличенное на единицу) в этой строке. Если закодировать состояния в строке кодом Грея, то для $l(p,q)=1$ и $l(p,q)=2$ величина $l(p,q)$ совпадает с расстоянием по Хэммингу между кодами состояний. Точный метод минимизации величины W' также требует полного перебора, но Армстронг предложил алгоритм, при котором эта величина близка к минимальной. Алгоритм реализуется в два этапа.

Алгоритм Армстронга

Начало. Задан автомат $A = \langle X, Q, \psi \rangle$. Требуется закодировать его состояния кодами длины k .

Этап I. Предварительное упорядочение.

Шаг 1. Рассмотрим все пары состояний автомата. Построим таблицу, строкам и столбцам которой сопоставим состояния автомата. Элементу (p,q) сопоставим вес этой пары. Добавим к таблице два столбца, в один из них запишем $w'(p)$ – вес состояния p , то есть сумму элементов в строке таблицы, а во второй $w''(p)$ – вес соседей, то есть сумму весов состояний q , для которых $w(p,q) > 0$.

Шаг 2. Упорядочим состояния в порядке невозрастания их весов. Состояния одинакового веса упорядочим в порядке невозрастания весов соседей. Упорядоченную таким образом последовательность состояний обозначим через S .

Этап II. Окончательное упорядочение

Шаг 3. Заготовим несколько пустых строк. Первое состояние из последовательности S запишем в первую строку. Если второе состояние связано с первым, запишем второе состояние в первую строку, иначе – во вторую строку.

Шаг 4. Пусть имеется t строк, содержащих множества состояний N_1, \dots, N_t . Рассмотрим очередное состояние q . Если оно не связано ни с одним из уже рассмотренных состояний, записываем его в новую пустую строку с номером $t+1$. Иначе для каждой строки подсчитаем характеристику

$$R_i(q) = \frac{1}{|N_i|} \left(\sum_{p \in N_i} w(p, q) \right).$$

Разместим состояние q в строке с номером j , для которой характеристика $R_j(q) = \max_{1 \leq i \leq l} R_i(q)$, (то есть максимальна среди всех характеристик), справа или слева, в зависимости от того, какое размещение дает минимум критерия

$$W' = \sum_{p \in N_j} w(p, q) l(p, q).$$

Шаг 5. Рассмотрим множества N_1, \dots, N_l (кроме N_j) в порядке невозрастания характеристик $R_i(q)$. Очередное множество N_i добавляем в строку с номером j . Возможны четыре варианта: множество N_i добавляется справа или слева, в прямом или обратном порядке. Из этих вариантов выбирается тот, который дает минимум критерия

$$W' = \sum_{p \in N_i} w(p, q) l(p, q).$$

Шаг 6. Если не все состояния рассмотрены, перейдем на шаг 4. Иначе объединяем все оставшиеся строки в одну и кодируем состояния по порядку кодом Грея длины k .

Пример. Рассмотрим автомат из предыдущего примера.

$X \backslash Q$	1	2	3	4	5	6
00	1	1	5	4	2	2
01	3	3	3	6	6	6
10	2	5	4	4	4	6

Шаг 1 частично выполнен в предыдущем примере, здесь только добавляем к таблице два последних столбца. Рассчитываем веса состояний, суммируя элементы строк таблицы.

$$w'(1) = 3 + 3 + 0 + 1 + 0 = 7;$$

$$w'(2) = 3 + 1 + 1 + 0 + 3 = 8;$$

$$w'(3) = 3 + 1 + 1 + 2 + 0 = 7;$$

$$w'(4) = 0 + 1 + 1 + 3 + 2 = 7;$$

$$w'(5) = 1 + 0 + 2 + 3 + 2 = 8;$$

$$w'(6) = 0 + 3 + 2 + 2 + 0 = 7.$$

Рассчитываем веса соседей для каждого состояния – сумму весов состояний, связанных с данным.

$$w''(1) = w'(2) + w'(3) + w'(5) = 8 + 7 + 8 = 23;$$

$$w''(2) = w'(1) + w'(3) + w'(4) + w'(6) = 7 + 8 + 7 + 7 = 29;$$

$$w''(3) = w'(1) + w'(2) + w'(4) + w'(5) = 7 + 8 + 7 + 8 = 30;$$

$$w''(4) = w'(2) + w'(3) + w'(5) + w'(6) = 8 + 7 + 8 + 7 = 30;$$

$$w''(5) = w'(1) + w'(3) + w'(4) + w'(6) = 7 + 8 + 7 + 7 = 29;$$

$$w''(6) = w'(2) + w'(4) + w'(5) = 8 + 7 + 8 = 23.$$

Итоговая таблица имеет вид

	1	2	3	4	5	6	w'	w''
1		3	3	0	1	0	7	23
2	3		1	1	0	3	8	29
3	3	1		1	2	0	7	30
4	0	1	1		3	2	7	30
5	1	0	2	3		2	8	29
6	0	3	0	2	2		7	23

Шаг 2. Упорядочим состояния по невозрастанию весов, состояния одинакового веса – по невозрастанию весов соседей. Результатом будет строка

$$S = \{2, 5, 3, 4, 1, 6\}.$$

Шаг 3. Помещаем состояние 2 в первую строку, затем помещаем состояние 5 во вторую строку, поскольку оно не связано с состоянием 2.

$$N_1 = \{2\}, \quad N_2 = \{5\}.$$

Шаг 4. Состояние 3 сильнее связано с состоянием 5, значит, помещаем его во вторую строку рядом с состоянием 5. Поскольку состояние 3 связано также с состоянием 2, объединяем первую и вторую строки, размещая состояние 2 рядом с состоянием 3.

$$N_1 = \{2, 3, 5\}.$$

Шаг 6. Не все состояния просмотрены, идем на шаг 4.

Шаг 4. Состояние 4 связано с состояниями 2, 3, 5. Размещению $N_1 = \{4, 2, 3, 5\}$ соответствует значение критерия

$$W = w(4,2)l(4,2) + w(4,3)l(4,3) + w(4,5)l(4,5) = 1 \cdot 1 + 1 \cdot 2 + 3 \cdot 3 = 12.$$

Размещению $N_1 = \{2, 3, 5, 4\}$ соответствует значение критерия

$$W = w(4,2)l(4,2) + w(4,3)l(4,3) + w(4,5)l(4,5) = 1 \cdot 3 + 1 \cdot 2 + 3 \cdot 1 = 8.$$

Минимум критерия достигается на втором размещении, окончательно имеем

$$N_1 = \{2, 3, 5, 4\}.$$

Шаг 6. Не все состояния просмотрены, идем на шаг 4.

Шаг 4. Состояние 1 связано с состояниями 2, 3 и 5. Размещению $N_1 = \{1, 2, 3, 5, 4\}$ соответствует значение критерия

$$W = w(1, 2)l(1, 2) + w(1, 3)l(1, 3) + w(1, 5)l(1, 5) = 3 \cdot 1 + 3 \cdot 2 + 1 \cdot 3 = 12.$$

Размещению $N_1 = \{2, 3, 5, 4, 1\}$ соответствует значение критерия

$$W = w(1, 2)l(1, 2) + w(1, 3)l(1, 3) + w(1, 5)l(1, 5) = 3 \cdot 4 + 3 \cdot 3 + 1 \cdot 2 = 23.$$

Минимум критерия достигается на первом размещении, окончательно имеем

$$N_1 = \{1, 2, 3, 5, 4\}.$$

Шаг 6. Не все состояния просмотрены, идем на шаг 4.

Шаг 4. Состояние 6 связано с состояниями 2, 4 и 5. Размещению $N_1 = \{6, 1, 2, 3, 5, 4\}$ соответствует значение критерия

$$W = w(6, 2)l(6, 2) + w(6, 4)l(6, 4) + w(6, 5)l(6, 5) = 3 \cdot 2 + 2 \cdot 5 + 2 \cdot 4 = 24.$$

Размещению $N_1 = \{1, 2, 3, 5, 4, 6\}$ соответствует значение критерия

$$W = w(6, 2)l(6, 2) + w(6, 4)l(6, 4) + w(6, 5)l(6, 5) = 3 \cdot 4 + 2 \cdot 1 + 2 \cdot 2 = 18.$$

Минимум критерия достигается на втором размещении, окончательно имеем

$$N_1 = \{1, 2, 3, 5, 4, 6\}.$$

Шаг 6. Все состояния просмотрены, закодируем их кодом Грея.

1	000
2	001
3	011
5	010
4	110
6	111

5.2. Кодирование состояний, минимизирующее число переключений триггеров

Рассмотрим теперь, как можно минимизировать число переключений триггеров. Очевидно, что для этого необходимо выполнить следующее требование: если существует переход из состояния q в состояние q' , то эти состояния необходимо кодировать соседними

кодами. Естественно, все требования такого рода выполнить вряд ли удастся, поэтому будем действовать следующим образом: чем чаще встречается переход из состояния в состояние, тем предпочтительнее кодировать эти состояния соседними кодами.

Определение. *Весом* пары состояний p и q (обозначается $w(p, q)$) назовем число переходов из состояния q в состояние p и наоборот (или число дуг, соединяющих соответствующие вершины диаграммы переходов-выходов).

Целью кодирования является минимизация критерия

$$W = \sum_{p, q \in Q} w(p, q) \cdot d(\rho(p), \rho(q)),$$

где $\rho(q)$ – код состояния q , $d(\rho(p), \rho(q))$ – расстояние по Хэммингу между кодами состояний p и q

Рассмотрим один из эвристических алгоритмов такого кодирования.

Алгоритм кодирования состояний, минимизирующего число переключений триггеров

Начало. Задан автомат $A = \langle X, Q, \psi \rangle$. Требуется закодировать его состояния кодами длины k .

Шаг 1. Построим таблицу, строкам и столбцам которой сопоставим состояния автомата. Элементу (p, q) сопоставим вес этой пары. В последний столбец запишем $w'(p)$ – *цену состояния* p , то есть число ненулевых элементов в строке таблицы. Образует множество закодированных состояний \tilde{Q} , первоначально $\tilde{Q} = \emptyset$.

Шаг 2. Выберем из таблицы пару с наибольшим весом w . Если таких пар несколько, то выберем пару с наибольшей суммой характеристик w' . Закодируем состояния соседними кодами (например, $0\dots 00, 0\dots 01$). Добавим закодированные состояния в множество \tilde{Q} .

Шаг 3. Из пар, имеющих общее состояние с состояниями из \tilde{Q} , выберем пару с максимальным весом. Если таких пар несколько, выберем пару с максимальной суммой весов состояний w' .

Шаг 4. Закодируем теперь еще не закодированное состояние p из выбранной пары. Для этого выделим коды, ближайшие по Хэм-

мингу к кодам уже размещенных состояний $q \in \tilde{Q}$, для которых $w(q, p) > 0$. Поочередно припишем состоянию p все такие коды α и вычислим величины:

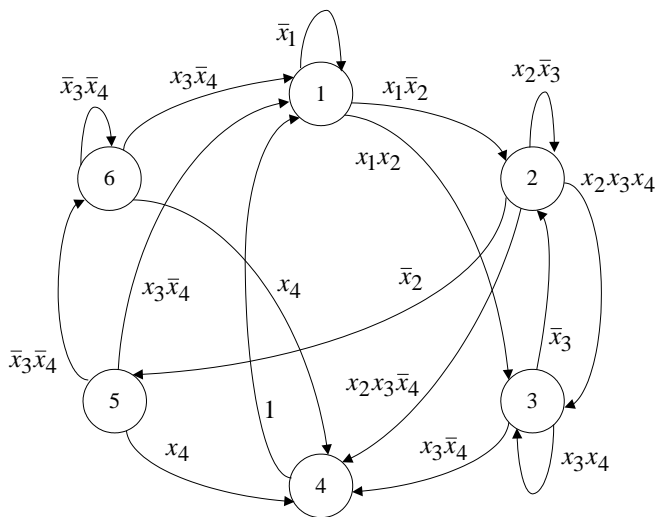
$$S(\alpha) = \sum_{q \in \tilde{Q}} w(q, p) d(p(q), \alpha).$$

Окончательно закодируем p кодом α , доставляющим максимум суммы $S(\alpha)$, добавляем p в множество \tilde{Q} .

Шаг 5. Если $\tilde{Q} \neq Q$, перейдем на шаг 3.

Конец.

Пример. Пусть автомат задан диаграммой переходов.



Шаг 1. Строим таблицу, строкам и столбцам которой сопоставим состояния автомата. Элементу (p, q) сопоставляем вес этой пары. В последний столбец записываем $w'(p)$ – вес состояния p , то есть число ненулевых элементов в строке таблицы.

	1	2	3	4	5	6	w'
1		1	1	1	1	1	5
2	1		2	1	1	0	4
3	1	2		1	0	0	3
4	1	1	1		1	1	5
5	1	1	0	1		1	4
6	1	0	0	1	1		3

Полагаем $\tilde{Q} = \emptyset$.

Шаг 2. Выбираем из таблицы пару с наибольшим весом — это $\{2,3\}$. Кодировем состояния пары соседними кодами. Процесс кодирования будем иллюстрировать матрицами Грея.

2			
3			

Шаг 3. Общее состояние с $\{2,3\}$ имеют пары $\{1,2\}$, $\{1,3\}$, $\{2,4\}$, $\{2,5\}$, $\{3,4\}$. Суммы весов состояний соответственно равны 9, 8, 9, 8, 8, значит, выбираем пару $\{1,2\}$.

Шаг 4. В нашем случае еще не закодировано состояние 1. Возможные коды α выделены на матрице Грея слева.

2	•		
3	•		

2	1		
3			

$$S(001) = w(2,1)d(000,001) + w(3,1)d(100,001) = 1 \cdot 1 + 1 \cdot 2 = 3;$$

$$S(101) = w(2,1)d(000,101) + w(3,1)d(100,101) = 1 \cdot 2 + 1 \cdot 1 = 3.$$

Отсюда следует, что состояние 1 можно закодировать любым из выделенных кодов. Итоговое размещение показано на матрице справа. Добавляем состояние 1 в \tilde{Q} .

Шаг 5. Не все состояния размещены, идем на шаг 3.

Шаг 3. Пары $\{1,4\}$, $\{1,5\}$, $\{1,6\}$, $\{2,4\}$, $\{2,5\}$, $\{3,4\}$ имеют вес 1 и суммы весов состояний соответственно 10, 9, 8, 9, 8, 8, значит, выбираем пару $\{1,4\}$.

Шаг 4. Размещаем состояние 4. Подходящие коды выделены на матрице Грея слева, на матрице справа показано окончательное размещение состояния 4.

$$S(011) = w(1,4)d(001,011) + w(2,4)d(000,011) + w(3,4)d(100,011) = 1 \cdot 1 + 1 \cdot 2 + 1 \cdot 3 = 6;$$

$$S(010) = w(1,4)d(001,010) + w(2,4)d(000,010) + w(3,4)d(100,010) = 1 \cdot 2 + 1 \cdot 1 + 1 \cdot 2 = 5;$$

$$S(101) = w(1,4)d(001,101) + w(2,4)d(000,101) + w(3,4)d(100,101) = 1 \cdot 1 + 1 \cdot 2 + 1 \cdot 1 = 4 \rightarrow \min;$$

$$S(110) = w(1,4)d(001,110) + w(2,4)d(000,110) + w(3,4)d(100,110) = 1 \cdot 3 + 1 \cdot 2 + 1 \cdot 1 = 5.$$

Шаг 5. Не все состояния размещены, идем на шаг 3.

Шаг 3. Рассматриваем пары $\{1,5\}$, $\{1,6\}$, $\{2,5\}$, $\{4,5\}$, $\{4,6\}$ с весом 1 и суммами весов состояний 9, 8, 8, 9, 8. Выбираем из них пару $\{1,5\}$.

Шаг 4. Кодлируем состояние 5, учитывая, что $w(3,5) = 0$.

$$S(011) = w(1,5)d(001,011) + w(2,5)d(000,011) + w(4,5)d(101,011) = 1 \cdot 1 + 1 \cdot 2 + 1 \cdot 2 = 5 \rightarrow \min;$$

$$S(010) = w(1,5)d(001,010) + w(2,5)d(000,010) + w(4,5)d(101,010) = 1 \cdot 2 + 1 \cdot 1 + 1 \cdot 3 = 6;$$

$$S(111) = w(1,5)d(001,111) + w(2,5)d(000,111) + w(4,5)d(101,111) = 1 \cdot 2 + 1 \cdot 3 + 1 \cdot 1 = 6.$$

Шаг 5. Не все состояния размещены, идем на шаг 3.

Шаги 3–4. Далее опять нужно выделить пары по изложенным правилам, но ясно, что осталось разместить лишь состояние 6, связанное переходами с состояниями 1, 4 и 5.

$$S(010) = w(1,6)d(001,010) + w(4,6)d(101,010) + w(5,6)d(011,010) = 1 \cdot 2 + 1 \cdot 3 + 1 \cdot 1 = 6;$$

$$S(111) = w(1,6)d(001,111) + w(4,6)d(101,111) + w(5,6)d(011,111) = 1 \cdot 2 + 1 \cdot 1 + 1 \cdot 1 = 4 \rightarrow \min.$$

Шаг 5. Все состояния размещены. Кодирование имеет вид:

1	001
2	000
3	100
4	101
5	011
6	111

5.3. Задачи

Задача 1. Закодировать состояния автомата с помощью алгоритмов Хамфри и Армстронга.

a)	$X \backslash Q$	1	2	3	4	5	6	7	8	9	10
	0	1	1	1	7	8	9	9	9	9	1
	1	2	3	4	6	7	9	6	10	5	5

$X \backslash Q$	1	2	3	4	5	6
00	3	3	4	4	4	2
01	2	3	4	6	5	4
11	1	1	2	6	6	5

Задача 2. Записать логические уравнения функции перехода автоматов из задачи 1 для полученных кодирований. Сравнить кодирования по критерию простоты схемы, моделирующей автомат.

Задача 3. Для автоматов из задачи 1 получить кодирование, минимизирующее число переключений триггеров.