



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
им. Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 Информатика и вычислительная техника

ОТЧЕТ

по домашнему заданию

Дисциплина: Прикладная теория цифровых автоматов

Название: Минимизация конечного автомата.

Студент

ИУ6-45Б

(Группа)

Преподаватель

1.06.22

(Подпись, дата)

Б.М. Мякин

(И.О. Фамилия)

1.06.22

(Подпись, дата)

Ю.И. Бауман

(И.О. Фамилия)

Москва, 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
<u>ОСНОВНАЯ ЧАСТЬ</u>	4
Спецификация автомата.....	4
Программная реализация.....	7
ЗАКЛЮЧЕНИЕ.....	11
СПИСОК ИСТОЧНИКОВ.....	12

↑
показ

ВВЕДЕНИЕ

В настоящей работе выполнена минимизация цифрового автомата для придуманной задачи, а также реализован программный способ нахождения минимального количества вершин цифрового автомата, эквивалентного данному.

Существуют 2 способа реализации автомата: программный и аппаратный. Программная реализация выполняется на любом языке высокого уровня. Аппаратная реализация – предусматривает построение устройств памяти для запоминания текущего состояния автомата, в роли которых обычно используются триггеры. В настоящей работе использован программный способ реализации цифрового автомата, так как этот способ подразумевает вариативность реализации, возможность отладки и тестирования в процессе разработки программы. К программам (в отличие от аппаратной реализации цифровых автоматов) можно добавлять новые функции по мере изменения целей, под которые она разрабатывается

Задача: есть счетчик, в каждый момент времени к нему прибавляется одно из значений $\{0, 1\}$, требуется определить в каждый момент времени кратно ли двойке значение счетчика

Цель работы - закрепить навыки построения и минимизации конечных цифровых автоматов. Для реализации поставленной цели необходимо выполнить следующие задачи.

Задачи:

- Изучить задание в соответствии со своим вариантом;
- Описать автомат, соответствующий условию задачи;
- Построить минимальный автомат эквивалентный данному;
- Написать программу нахождения минимального количества вершин автомата, эквивалентного данному;

ОСНОВНАЯ ЧАСТЬ

Вариант 4

Задание: Минимизация конечного автомата

Придумаем задачу: ~~Задача:~~ есть счетчик, в каждый момент времени к нему прибавляется одно из значений $\{0, 1\}$, требуется определить, в каждый момент времени кратна ли двойке значение счетчика

- Построим автомат (~~не являющийся минимальным~~).
Проанализируем задачу и составим матрицу переходов.

Таблица 1 – матрица переходов

Состояния	δ		λ	
	1	0	1	0
Q0	1	0	0	1
Q1	2	1	1	0
Q2	3	2	0	1
Q3	0	3	1	0
Q4	0	1	0	1

Цифры в таблице 0, 1 в столбце δ означают входное значение 0 или 1, соответственно. Цифры 0, 1 в столбце λ означают выходное значение четное или нечетное соответственно.

Спецификация автомата

- Состояния автомата:
 q_4 - начальное состояние автомата; q_0 – суммарное значение суммы кратно 4;
 q_1 – суммарное значение дает остаток 1 при делении на 4; q_2 – суммарное значение дает остаток 2 при делении на 4; q_3 – суммарное значение дает остаток 3 при делении на 4.
- Входные сигналы:
1 – на вход автомата пришло значение 1
0 – на вход автомата пришло значение 0
- Выходные сигналы:
0 – суммарное значение четно
1 – суммарное значение нечетно

Построим описанный автомат, смотри рисунок 1.

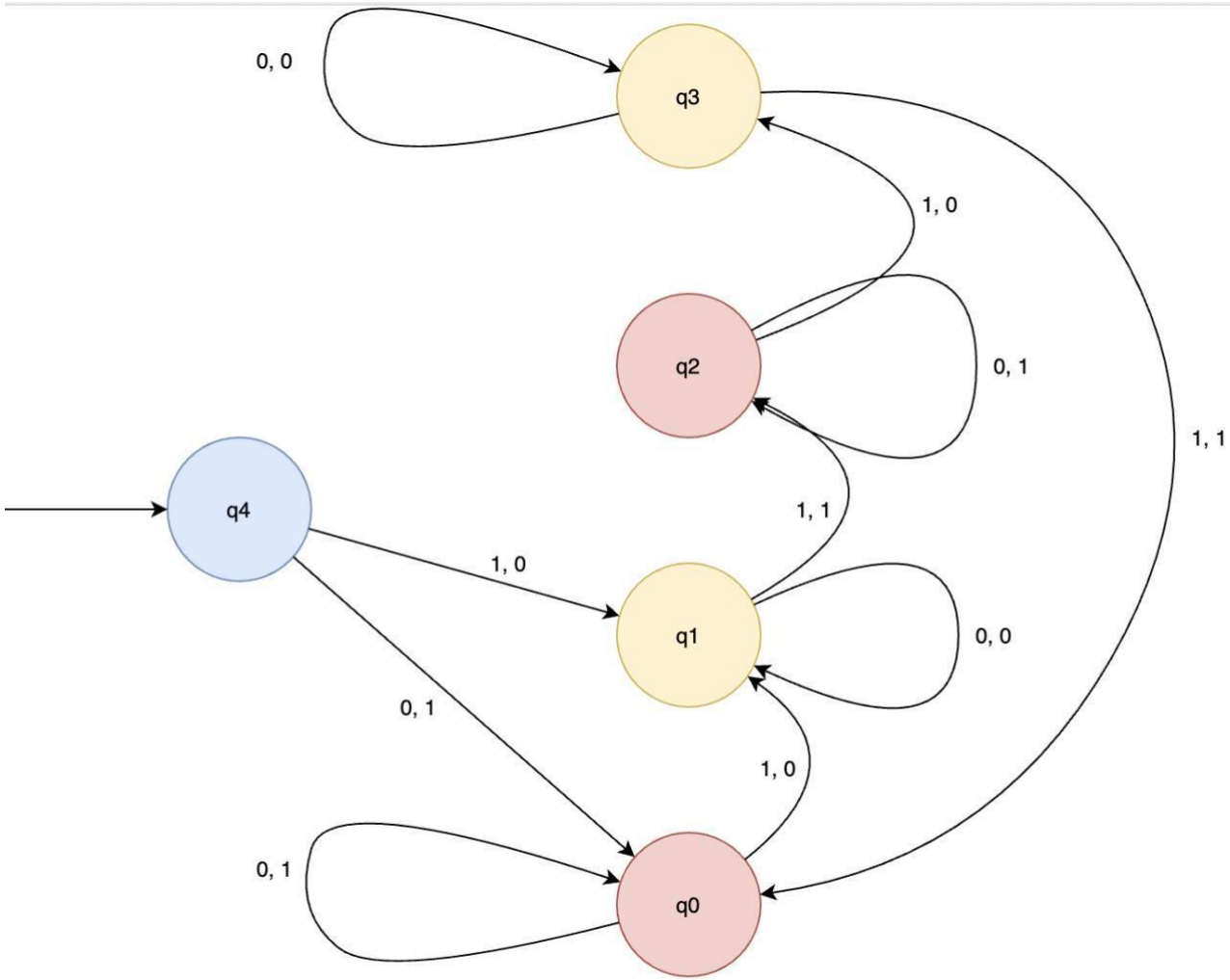


Рисунок 1 – Описанный автомат

Таблица 2 – матрица переходов

Состояния					π_2	
	δ		λ		δ	δ
	1	0	1	0	a	b
Q0	1	0	0	1	B ₁	A ₁
Q1	2	1	1	0	A ₁	B ₁
Q2	3	2	0	1	B ₁	A ₁
Q3	0	3	1	0	A ₁	B ₁
Q4	0	1	0	1	B ₁	A ₁

Начальное разбиение представляет собой один блок, включающий в себя все состояния $\pi_0 = \{A_0 = \langle 0, 1, 2, 3, 4 \rangle\}$.

Разбиение π_1 объединяет в один блок те состояния, которые нельзя различить при подаче цепочек длиной в 1 символ. Функция выходов λ при подаче сигналов 1 и 0 не может различить состояния 0, 2, 4 так как для каждого из этих состояний при подаче на вход 0 автомат выдает 1, при 1 - 0.

Состояния 1, 3 попадают в другой блок, но их также нельзя различить входной цепочкой длиной в 1 символ: $\pi_1 = \{A_1 = \langle 0, 2, 4 \rangle; B_1 = \langle 1, 3 \rangle\}$.

Следующее разбиение π_2 – неразличимые состояния цепочек длиной 2.

Воспользуемся теоремой, в соответствии с которой в один блок π_{k+1} попадут те состояния p и q , для которых справедливо $(\forall x \in X) \delta(p, x) \approx_k \delta(q, x)$, причем эти состояния должны быть из предыдущего разбиения. При построении π_2 и соответствующей таблицы переходов вместо значения состояния $\delta(p, x)$ будем писать номер блока разбиения π_1 , в который попадает $\delta(p, x)$. Итак, $\pi_2 = \{A_2 = \langle 0, 2, 4 \rangle; B_2 = \langle 1, 3 \rangle\}$.

Итак:

$$\pi_0 = \{A_0 = \langle 0, 1, 2, 3 \rangle\}$$

$$\pi_1 = \{A_1 = \langle 0, 2, 4 \rangle; B_1 = \langle 1, 3 \rangle\}.$$

$$\pi_2 = \{A_2 = \langle 0, 2, 4 \rangle; B_2 = \langle 1, 3 \rangle\}.$$

$\pi_2 = \pi_1$ Процесс закончен.

Построим минимальный автомат:

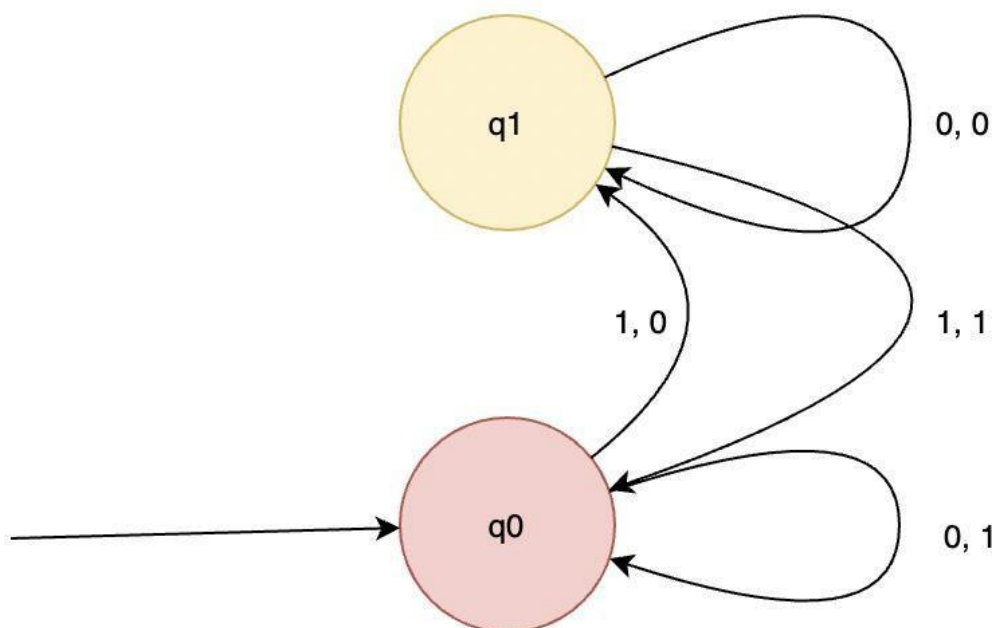


Рисунок 2 – минимальный конечный автомат

Теперь напишем программу, которая выведет нам последовательно все разбиения и минимальное количество вершин в автомате.

Программу реализуем на языке C++.

```
#include <iostream>
#include <vector>
#include <set>
#include <utility>
#include <map>

using namespace std;

const unsigned int SIZE = 2;
class Automat {
public:
    Automat(vector < pair <int, int> >& transitions,
            vector < pair <int, int> >& output) :
        transitions(transitions),
        output(output) {}
    void Minimise() {
        map < pair <int, int>, vector <int> >
prevSetOfValues;
        map < pair <int, int>, vector <int> > setOfValues;
        map < int, int > valueWithNumberOfSet;
        for (int i = 0; i < output.size(); ++i) {
            pair <int, int> values = output[i];
            setOfValues[values].push_back(i);
        }
        int pi = 1;
        int counter = 0;
        cout << "Partition: " << pi << endl;
        for (auto partition : setOfValues) {
            for (auto value : partition.second) {
                cout << value << " ";
                valueWithNumberOfSet[value] = counter;
            }
            ++counter;
            cout << endl;
        }
        ++pi;
        do {
            prevSetOfValues = setOfValues;
            setOfValues.clear();
            for (int i = 0; i < transitions.size(); ++i) {
                pair <int, int> values = {
valueWithNumberOfSet[transitions[i].first],
```

```

valueWithNumberOfSet[transitions[i].second]
        };
        setOfValues[values].push_back(i);
    }
    int counter = 0;
    cout << "Partition: " << pi << endl;
    for (auto partition : setOfValues) {
        for (auto value : partition.second) {
            cout << value << " ";
            valueWithNumberOfSet[value] = counter;
        }
        ++counter;
        cout << endl;
    }
    ++pi;
    } while (prevSetOfValues.size() !=
setOfValues.size());
    cout << "Resulted automat has: " <<
setOfValues.size() << " peaks" << endl;
    }
private:
    vector < pair <int, int> > transitions;
    vector < pair <int, int> > output;
};
int main() {
    int n;
    cout << "Enter quantity of elements:" << endl;
    cin >> n;
    vector < pair <int, int> > transitions(n);
    vector < pair <int, int> > output(n);
    cout << "Enter matrix delta:" << endl;
    for (int i = 0; i < n; ++i) {
        cin >> transitions[i].first >>
transitions[i].second;
    }
    cout << "Enter matrix lambda:" << endl;
    for (int i = 0; i < n; ++i) {
        cin >> output[i].first >> output[i].second;
    }
    Automat automat(transitions, output);
    automat.Minimise();
    return 0;
}

```


Примеры работы программы:

Результат работы программы на указанном выше автомате представлен на рисунке 3.

```

Консоль отладки Microsoft Visual Studio
5
Enter matrix delta:
1      0
2      1
3      2
0      3
0      1
Enter matrix lambda:
0      1
1      0
0      1
1      0
0      1
Partition: 1
0 2 4
1 3
Partition: 2
1 3 4
0 2
Resulted automat has: 2 peaks
  
```

Рисунок 3 – результат работы программы

Так же проверим как работает программа на примере из лекций. Для этого уменьшим значение каждой вершины на 1. (1 -> 0, 2 -> 1 и т.д.). Результат смотри на рисунках 4, 5.

Таблица 3 – пример из лекции

Состояния					π_2		π_3		π_4	
	δ		λ		δ	δ	δ	δ	δ	δ
	a	b	a	b	a	b	a	b	a	b
1	3	6	1	0	A_1	A_1	C_2	A_2	D_3	A_3
2	4	8	0	1	B_1	A_1	B_2	D_2	C_3	E_3
3	1	4	1	0	A_1	B_1	A_2	B_2	A_3	C_3

4	7	9	0	1	B ₁	A ₁	B ₂	C ₂	C ₃	D ₃
5	9	1	1	0	A ₁	A ₁	C ₂	A ₂	D ₃	A ₃
6	3	5	1	0	A ₁	A ₁	C ₂	A ₂	D ₃	A ₃
7	4	3	0	1	B ₁	A ₁	B ₂	C ₂	C ₃	D ₃
8	4	2	1	0	B ₁	B ₁	B ₂	B ₂	C ₃	B ₃
9	5	7	1	0	A ₁	B ₁	A ₂	B ₂	A ₃	C ₃

Консоль отладки Microsoft Visual Studio

```

Enter quantity of elements:
9
Enter matrix delta:
2 5
3 7
0 3
6 8
8 0
2 4
3 2
3 1
4 6
Enter matrix lambda:
1 0
0 1
1 0
0 1
1 0
1 0
0 1
1 0
1 0

```

Рисунок 4 – результат работы программы на примере из лекции.

```

Partition: 1
1 3 6
0 2 4 5 7 8
Partition: 2
7
u 1 3 6
2 8
0 4 5
Partition: 3
1
7
3 6
0 4 5
2 8
Partition: 4
7
1
3 6
2 8
0 4 5
Resulted automat has: 5 peaks

```

Рисунок 5 – результат работы программы на примере из лекции.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данного домашнего задания построен автомат для задачи определения четности числа (формулировку задания смотри выше). Далее построенный автомат был минимизирован. Кроме того, написана программа на C++, которая позволяет определить кол-во вершин в минимальном автомате, который эквивалентен данному.

что еще

СПИСОК ИСТОЧНИКОВ

1. ГОСТ 7.32-2017. СИБИБД. Отчет о научно-исследовательской работе. Структура и правила оформления = System of standards on information, librarianship and publishing. The research report. Structure and rules of presentation : Национальный стандарт РФ : Введ. 01.07.2018. - М. : Стандартиформ, 2017. - [32 л.]. - URL: <https://docs.cntd.ru/document/1200157208> (дата обращения: 04.05.2022). - Текст: электронный.