

Министерство науки и высшего образования РФ  
Федеральное государственное автономное  
образовательное учреждение высшего образования  
**«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Институт космических и информационных технологий  
институт  
Программная инженерия  
кафедра

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №1**  
Конечные автоматы

тема

Преподаватель

подпись, дата

А. С. Кузнецов  
инициалы, фамилия

Студент КИ23-16/1Б, 032321684  
номер группы, зачетной книжки

подпись, дата

К. М. Дорошев  
инициалы, фамилия

Красноярск 2025

## **1 Цель**

Реализация и исследование детерминированных и недетерминированных конечных автоматов.

## **2 Задание**

Вариант – 10.

Для выполнения практической работы необходимо разработать в системе JFLAP конечные автоматы и произвести программную реализацию на языке Java для следующих автоматов:

- 1) Построить ДКА, допускающий в алфавите  $\{0, 1\}$  все цепочки нулей и единиц с одинаковыми парами символов на обоих краях цепочки.
- 2) Построить НКА, допускающий язык из цепочек из 0 и 1, в которых хотя бы на одной из последних пяти позиций стоит 1.

## **3 Ход выполнения**

Для начала была установлена программа JFLAP, в которой были построены конечные автоматы из условия задания. Каждый КА был сначала протестирован в JFLAP тестовыми цепочками, затем была написана программная реализация на Java, которая также была протестирована на корректность работы теми же самыми тестовыми цепочками.

### **3.1 Построение ДКА**

Необходимо было реализовать детерминированный конечный автомат (ДКА). Построить ДКА, допускающий в алфавите  $\{0, 1\}$  все цепочки нулей и единиц, где самый левый символ отличается от самого правого символа.

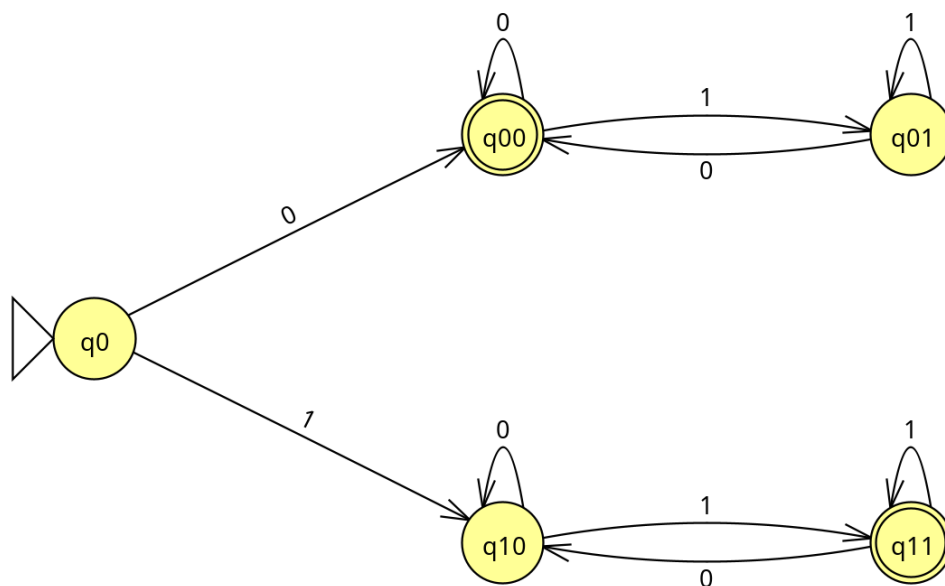


Рисунок 1 – ДКА в JFLAP

На рисунке 2 показан тест ДКА для цепочки «00100» которая удовлетворяет условию.

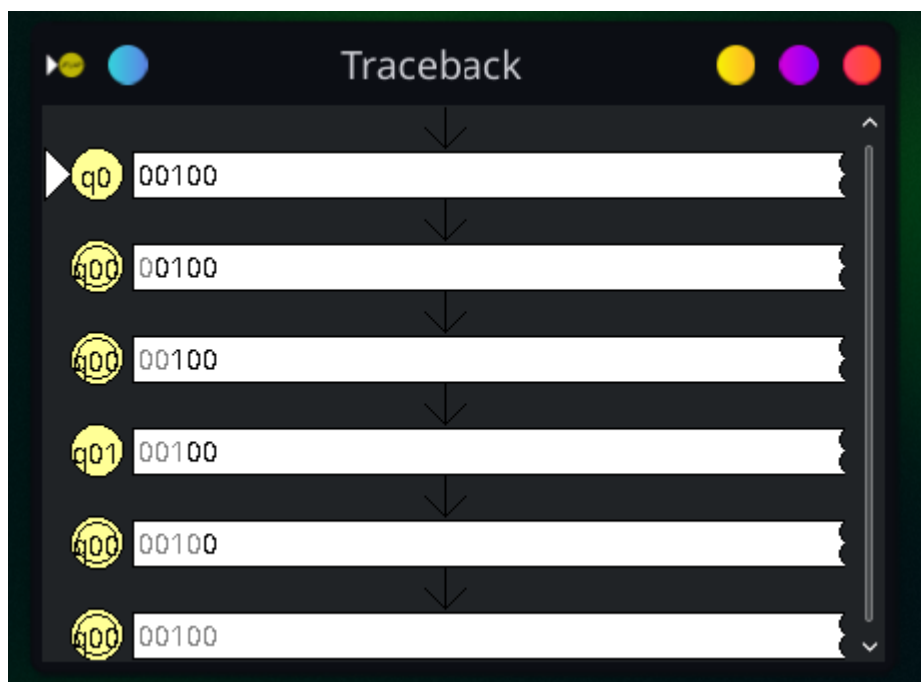


Рисунок 2 – Тест для цепочки «00100»

Также была проверка на отклонение строки, которая не удовлетворяет условию. На рисунке 3 показан тест ДКА для цепочки «110000».

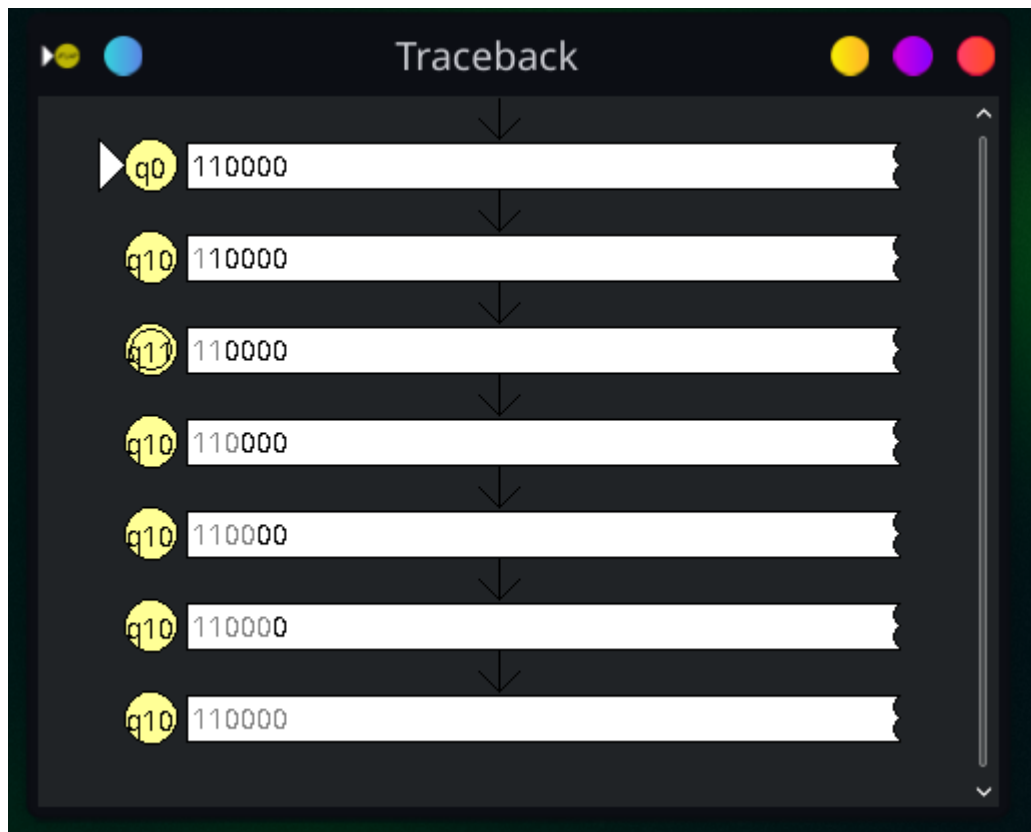


Рисунок 3 – Тест для цепочки «110000»

Дальше был написан код на Java для реализации данного ДКА, он показан на рисунке 4.

```
" Press ? for help
.. (up a dir)
~/Projects/test/avtomat-1/
├── .git/
├── docs/
├── jflap/
├── reports/
├── src/
│   ├── Main.java
│   ├── Makefile
│   └── README.md
└── ~

39 import java.util.*;
38
37 class DFA {
36     private final Map<String, Map<String, String[]>> transitions;
35
34     public DFA() {
33         transitions = new HashMap<>();
32
31         addTransition(null, null, "0", new String[]{"0", "0"});
30         addTransition(null, null, "1", new String[]{"1", "1"});
29
28         addTransition("0", "0", "0", new String[]{"0", "0"});
27         addTransition("0", "0", "1", new String[]{"0", "1"});
26         addTransition("1", "1", "0", new String[]{"1", "0"});
25         addTransition("1", "1", "1", new String[]{"1", "1"});
24
23         addTransition("0", "1", "0", new String[]{"0", "0"});
22         addTransition("0", "1", "1", new String[]{"0", "1"});
21         addTransition("1", "0", "0", new String[]{"1", "0"});
20         addTransition("1", "0", "1", new String[]{"1", "1"});
19     }
18
17     private void addTransition(String firstChar, String lastChar, String input, String[] nextState) {
16         String key = firstChar + "," + lastChar;
15         transitions.computeIfAbsent(key, k -> new HashMap<>()).put(input, nextState);
14     }
13
12     public boolean accepts(String inputChain) {
11         int n = inputChain.length();
10         if (n < 2) {
9             return false; // строки короче 2 символов не принимаем
8         }
7
6         String firstTwo = inputChain.substring(0, 2);
5         String lastTwo = inputChain.substring(n - 2);
4
3         return firstTwo.equals(lastTwo);
2     }
1 }
40

<fzf/Projects/test/avtomat-1> NORMAL +7 -6 -13 src/Main.java java utf-8[unix] 34% 40/115% :1 [8]trailing
```

Рисунок 4 – Код для ДКА на Java

После компиляции, сборки и запуска программы были повторно проведены тесты, показанные на рисунке 5.

```

00 -> ACCEPT
01 -> ACCEPT
10 -> ACCEPT
11 -> ACCEPT
0000 -> ACCEPT
00100 -> ACCEPT
0 -> REJECT
1 -> REJECT
0001 -> REJECT
0110 -> REJECT
110000 -> REJECT

```

Рисунок 5 – Тесты для ДКА на Java

### 3.2 Построение НКА

Также необходимо было реализовать недетерминированный конечный автомат (НКА) допускающий язык из цепочек из 0 и 1, в которых хотя бы на одной из последних пяти позиций стоит 1. На рисунке 6 показана реализация ДКА первого задания в системе JFLAP.

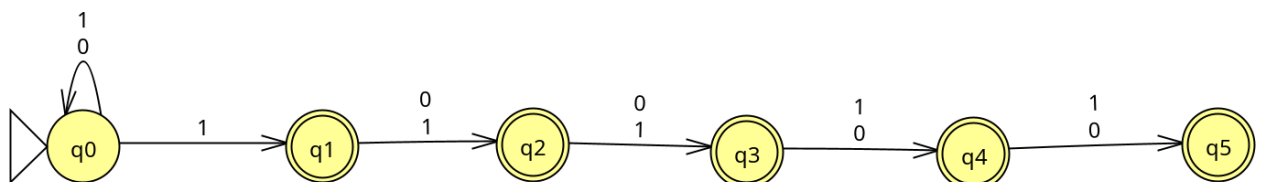


Рисунок 6 - НКА в JFLAP.

Дальше необходимо было провести все самые главные тесты для этой цепочки, они показаны на рисунке 7.

Input	Result
1	Accept
01	Accept
00001	Accept
101000	Accept
11111	Accept
0001000	Accept
1010101	Accept
000001	Accept
0	Reject
00	Reject
00000	Reject
100000	Reject
1100000	Reject

Рисунок 7 – Тесты для НКА.

Все тесты были пройдены успешно. На рисунке 8 показан код на Java для реализации данного НКА.

```

1  }
2  }
3  }
4  }
5  }
6  }
7  }
8  }
9  }
10 }
11 }
12 }
13 }
14 }
15 }
16 }
17 }
18 }
19 }
20 }
21 }
22 }
23 }
24 }
25 }
26 }
27 }
28 }
29 }
30 }
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }

class NFA {
    private final Set<Integer> currentStates;
    private final Map<Integer, Map<String, Set<Integer>>> transitions;
    private final int windowSize;

    public NFA() {
        this.windowSize = 5;
        this.currentStates = new HashSet<>();
        this.transitions = new HashMap<>();

        for (int i = 0; i <= windowSize; i++) {
            transitions.put(i, new HashMap<>());
        }

        currentStates.add(0);

        for (int state = 0; state < windowSize; state++) {
            transitions.get(state).computeIfAbsent("0", k -> new HashSet<>()).add(state + 1);
            transitions.get(state).computeIfAbsent("1", k -> new HashSet<>()).add(windowSize);
        }

        transitions.get(windowSize - 1).get("0").clear();
        transitions.get(windowSize - 1).get("0").add(windowSize - 1);

        transitions.get(windowSize).computeIfAbsent("0", k -> new HashSet<>()).add(windowSize);
        transitions.get(windowSize).computeIfAbsent("1", k -> new HashSet<>()).add(windowSize);
    }

    public boolean accepts(String inputChain) {
        Set<Integer> states = new HashSet<>(currentStates);

        for (char c : inputChain.toCharArray()) {
            Set<Integer> nextStates = new HashSet<>();
            String symbol = String.valueOf(c);

            for (int state : states) {
                if (transitions.get(state).containsKey(symbol)) {
                    nextStates.addAll(transitions.get(state).get(symbol));
                }
            }

            states = nextStates;
            if (states.isEmpty()) break;
        }

        return states.contains(windowSize);
    }
}

```

Рисунок 8 – Код для НКА на Java

После компиляции, сборки и запуска программы были повторно проведены тесты, показанные на рисунке 9.

```
NFA results:
1 -> ACCEPT
01 -> ACCEPT
00001 -> ACCEPT
101000 -> ACCEPT
11111 -> ACCEPT
0001000 -> ACCEPT
1010101 -> ACCEPT
000001 -> ACCEPT
0 -> REJECT
00 -> REJECT
00000 -> REJECT
100000 -> REJECT
1100000 -> REJECT
```

Рисунок 9 – Тесты для НКА на Java

#### **4 Выводы**

В ходе данной практической работы был изучен материал о детерминированных и недетерминированных конечных автоматах, были выполнены все задания, построены ДКА и НКА в системе JFLAP и реализован программный код на Java.