Ордена трудового красного знамени
Федеральное Государственное Бюджетное Образовательное
Учреждение высшего образования
Московский Технический Университет Связи и Информатики

Лабораторная работа №1

Тема работы: «Регулярные грамматики и конечные автоматы»

Вариант №11

Выполнил: Ракитский А.А.
2 курс группа БСТ-1854

Москва 2019

1. Задание на лабораторную работу

Часть 1

Для заданной грамматики написать функции переходов, таблицу переходов, построить диаграмму переходов. В случае если грамматика порождает недетерминированный конечный автомат, привести его к детерминированному виду, построив новую диаграмму состояний и выписав правила получившейся грамматики. Для построения диаграмм состояний воспользоваться пакетом JFLAP.

Часть 2

Для получившегося детерминированного конечного автомата написать программу – оконное приложение, реализующее функцию лексического анализа, получающего на вход цепочку языка, отображающего переходы между состояниями конечного автомата и отвечающего на вопрос, принадлежит ли цепочка языку, заданному грамматикой.

Заданная грамматика:

11.	G({S, A, B},{0,1},P,S), где P: S → 1A | 0B A → 0B | 0S | 1 B → 1A | 1S | 0

2. Решение задания Часть 1

2.1. Функции переходов

F(A, 1) = B F(A, 1) = S
F(B, 0) = A F(B, 0) = S
F(H, 0) = B F(H, 1) = A
F(S, 0) = A F(S, 1) = B

2.2. Таблица переходов

	0	1
H	B	A
A	-	B, S
B	A,S	-
S	A	B

2.3. Диаграмма переходов

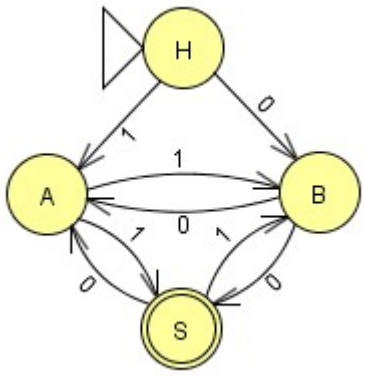

Рисунок 1 Диаграмма переходов НКА

Грамматика порождает недетерминированный конечный автомат.

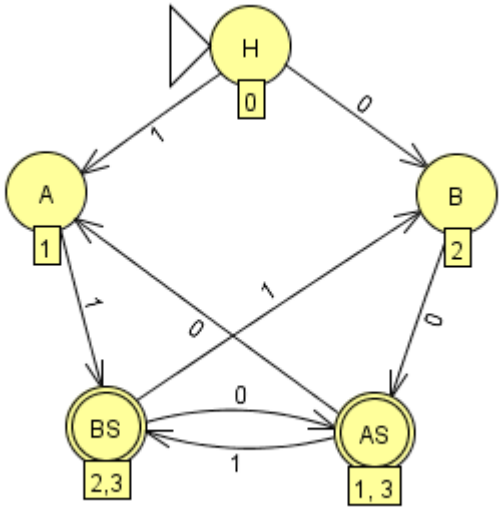

Рисунок 2 Диаграмма переходов ДКА

2.4. Правила грамматики ДКА

P:

A → 0AS | 1

B → 1BS | 0

BS → 1AS | 1A

AS → 0BS | 0B

3. Решение задания Часть 2

Код программы

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace TYAP_Lab1
{
    public partial class Form1 : Form
    {
        private enum State {H, A, B, AS, BS, ERROR}

        public Form1()
        {
            InitializeComponent();
        }

        //AnalyzeString() - анализируем строку
        //Reverse() - переворачиваем строку, так как правая грамматика
        //AdvanceState() - переходим в следующее состояние
        //ValidateString() - проверяем, принадлежит ли языку. Не уверен, что делать с пустой строкой
        //ShowResult() - выводим строку состояний

        private void button1_Click(object sender, EventArgs e)
        {
            AnalyzeString(textBox1.Text);
        }

        private void AnalyzeString(string input)
        {
            string result = "";
            State currentState = State.H;
            input = Reverse(input);
            foreach (char c in input)
            {
               if (currentState != State.ERROR)
               {
```

```
                currentState = AdvanceState(currentState, c);
                result += currentState.ToString() + "; ";
            }
            else
            {
                break;
            }
        }

        ValidateString(currentState);
        ShowResult(result);
    }

    //Если я правильно понял, то потому, что у меня правая грамматика, терминалы будет "кушать" справа налево,
    //поэтому для более удобной итерации имеет смысл перевернуть строку
    private string Reverse(string s)
    {
        char[] charArray = s.ToCharArray();
        Array.Reverse(charArray);
        return new string(charArray);
    }

    private State AdvanceState(State state, char c)
    {
          State nextState;
        switch (state)
        {
            case State.H:
            {
                if (c == '0')
                    nextState = State.B;
                else if (c == '1')
                    nextState = State.A;
                else
                    nextState = State.ERROR;
                break;
            }
            case State.A:
            {
                if (c == '1')
                    nextState = State.BS;
                else
                    nextState = State.ERROR;
                break;
```

```
                }
                case State.B:
                {
                    if (c == '0')
                        nextState = State.AS;
                    else
                        nextState = State.ERROR;
                    break;
                }
                case State.AS:
                {
                    if (c == '0')
                        nextState = State.A;
                    else if (c == '1')
                        nextState = State.BS;
                    else
                        nextState = State.ERROR;
                    break;
                }
                case State.BS:
                {
                    if (c == '0')
                        nextState = State.AS;
                    else if (c == '1')
                        nextState = State.B;
                    else
                        nextState = State.ERROR;
                    break;
                }
                default:
                {
                    nextState = State.ERROR;
                    break;
                }
                }

                return nextState;
        }

        private void ValidateString(State endState)
        {
            //Если все успешно, и строка кончается на одном из конечных состояний, то она
принадлежит языку
            if (endState == State.AS || endState == State.BS)
            {
```

```
                checkBox1.Checked = true;
            }
            else
            {
                checkBox1.Checked = false;
            }
        }

        private void ShowResult(string result)
        {
            textBox2.Text = result;
        }
    }
}
```

4. Результат работы программы

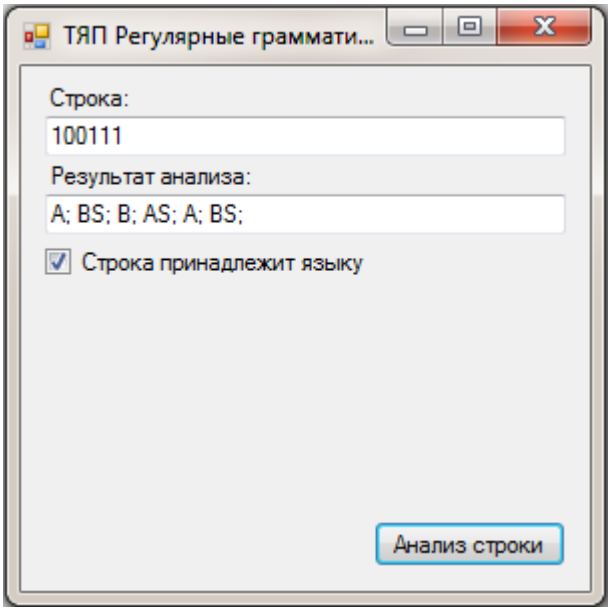

Рисунок 3 Результат анализа строки, принадлежащей языку

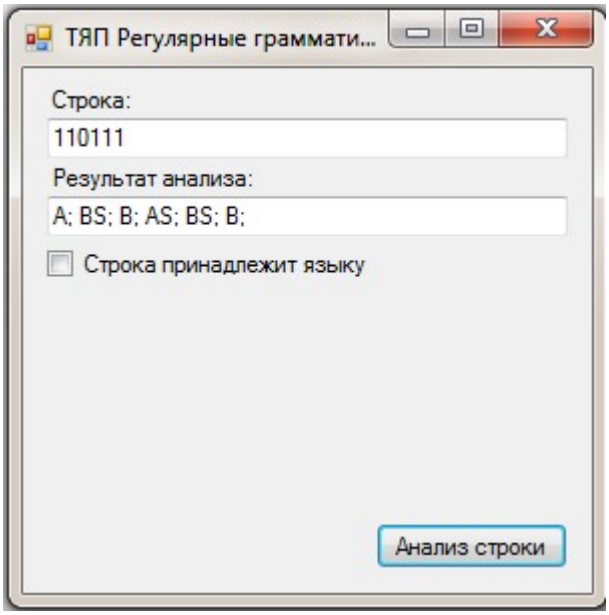

Рисунок 4 Результат анализа строки, не принадлежащей языку

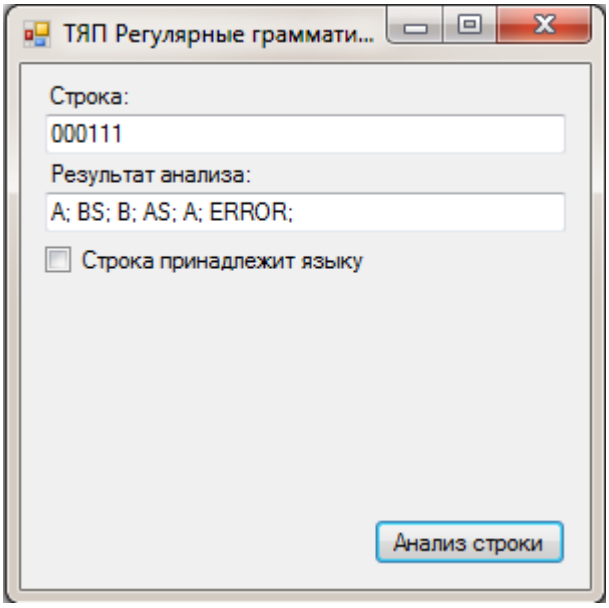

Рисунок 5 Результат анализа строки с попыткой некорректного перехода

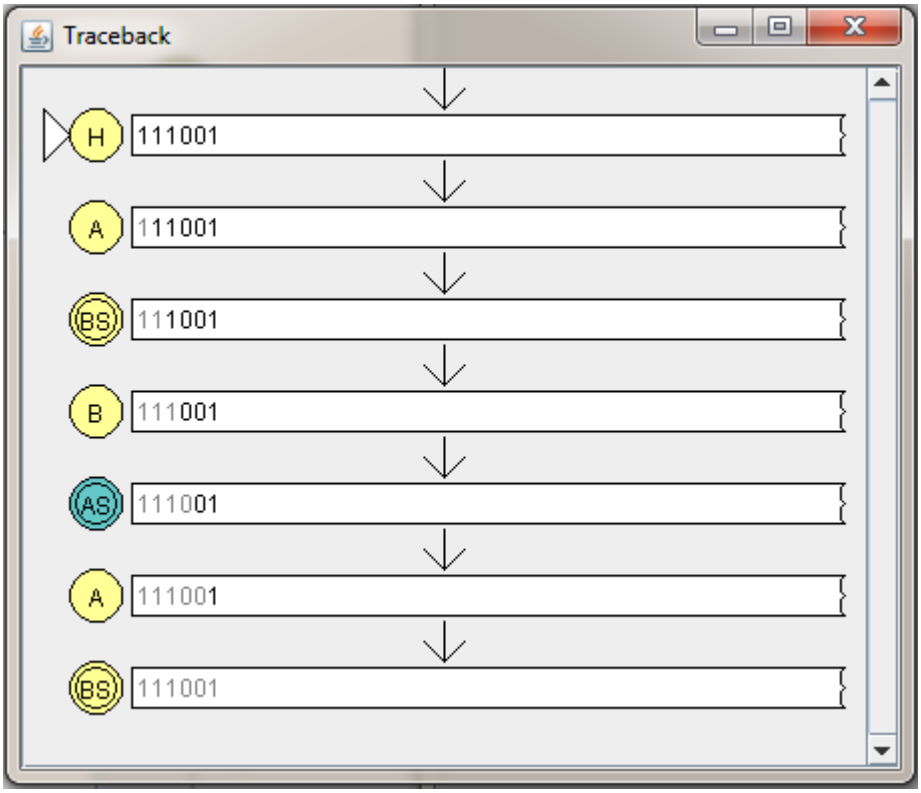

Рисунок 6 Сравнение результатов работы программы с JFLAP

5. Выводы

- Были записаны функции переходов
- Была записана таблица переходов
- НКА был приведен к ДКА
- Были составлены диаграммы переходов для НКА и ДКА
- Были составлены правила грамматики для ДКА
- Было разработано приложение, выполняющее анализ строки на принадлежность заданному языку
- Программа была протестирована с помощью встроенного функционала JFLAP