Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №5**

*з предмету: «ТПКС»*

*по темі:*

«Автоматизація синтезу таблиці переходів»

Виконав: студент ФІОТ

групи ІО-92

Петрук В.О.

Київ 2012р.

**Мета:** Здобуття навичок з аналізу графових структур і автоматизації процедури

побудови таблиці переходів.

**Завдання**

1. Представити номер залікової книжки в двійковому вигляді:

(9219)10=(10010000000011)2

2. В залежності від молодшого розряду номера залікової книжки визначити тип

тригера:

n2 n1 Тип тригира

1 1 JK

3. Розробити модуль генерації таблиці переходів і функцій збудження тригерів на

основі закодованого графу переходів.

4. Реалізувати засоби відображення таблиці та її збереження у файлі.

**Опис програми**

Розробка модуля генерації таблиці переходів виконується на основі раніше закодованого графу переходів автомата Мура. Виконується аналіз переходів графу та зміни станів при переходах. На основі цих даних визначаються можливі переходи алгоритму та значення функцій перемикання тригерів на кожному переході.

Таблиця переходів відображається в графічному інтерфесі користувача в окремому вікні разом з графом автомата Мура(рис. 1).

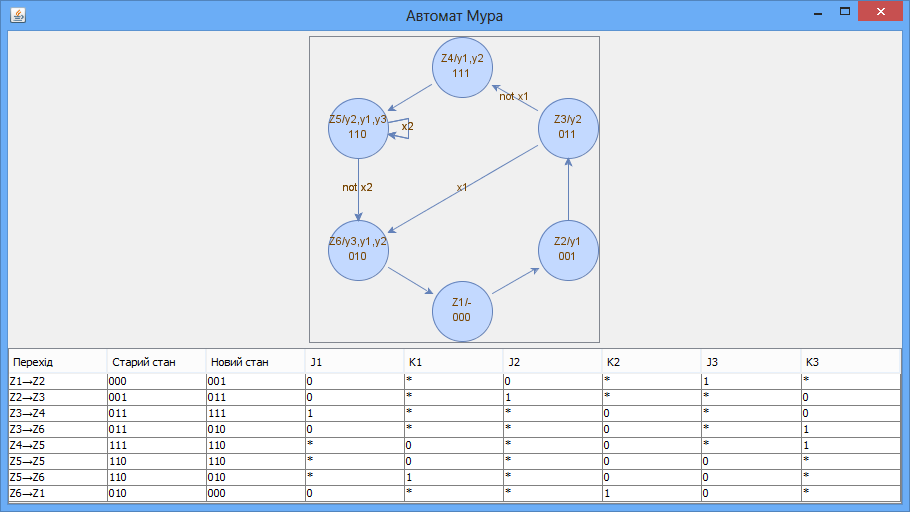


Рисунок 1 – Відображення графу та таблиці переходів автомата Мура

Для збереження таблиці переходів автомату Мура у вигляді текстового файлу необхідно вибрати пункти меню Файл -> «Зберегти таблицю переходів. У діалоговому вікні необхідно ввести ім’я файлу. Користувачу пропонується зберегти файл у форматі .mt (від Moore Table).

**Код програми**

Представлені лише класи та методи, які безпосередньо були реалізовані або змінені для виконання даної лабораторної роботи.

package lab1.file;

import lab1.graph.MooreMachine;

import lab1.gui.GraphPanel;

import lab1.minimisation.Minimizator;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.\*;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.HashSet;

import java.util.Set;

public class OpenMoorListener extends JFrame implements ActionListener {

private String[] columnNames;

private String[][] data;

private ArrayList<String> dataArrayL;

private String splitLine = ":";

private int markingSize = 4;

private int columnCount;

private Set<String> signals;

/\*\*

\* Конструктор обробника кнопки відкриття файлу.

\*/

public OpenMoorListener() {

setTitle("Автомат Мура");

setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

setSize\_Position();

setLayout(new BorderLayout());

}

/\*\*

\* Обробник кнопки відкриття файлу.

\*/

public void actionPerformed(ActionEvent e) {

JFileChooser chooser = new JFileChooser();

chooser.setCurrentDirectory(new File("."));

chooser.setSelectedFile(new File(".moor"));

int result = chooser.showOpenDialog(null);

if (result != JFileChooser.CANCEL\_OPTION) {

String fileName = chooser.getSelectedFile().getPath();

try {

createMoorFrame(fileName);

setTitle("Автомат Мура: " + fileName);

} catch (IOException e1) {

JOptionPane.showMessageDialog(null, ("Не вдалося відкрити файл " + fileName));

} catch (ClassNotFoundException e1) {

e1.printStackTrace(); //To change body of catch statement use File | Settings | File Templates.

}

}

}

/\*\*

\* Десереалізує обєкт класу MSA з файлу.

\*

\* @param fileName

\* @throws java.io.IOException

\* @throws ClassNotFoundException

\*/

private void createMoorFrame(String fileName) throws IOException, ClassNotFoundException {

BufferedReader moorFile = new BufferedReader(new FileReader(fileName));

ArrayList<String> newMoorTable = new ArrayList<String>();

String line = "";

while (null != (line = moorFile.readLine())) {

newMoorTable.add(line);

}

// for (int i = 0; i < newMoorTable.size(); i++) {

// System.out.println(newMoorTable.get(i));

// }

createMoorFrame(newMoorTable);

}

// відображення вікна з графом автомата Мура, створеного на основі табличної форми графу

public void createMoorFrame(ArrayList<String> newMoorTable) {

for (int i = 0; i < newMoorTable.size(); i++) {

System.out.println(newMoorTable.get(i));

}

MooreMachine mooreMachine = new MooreMachine(newMoorTable);

GraphPanel graphPanel = new GraphPanel(mooreMachine);

add(graphPanel, BorderLayout.NORTH);

columnCount = 3 + markingSize \* 2;

String[] conditions = graphPanel.getConditions();

columnCount += conditions.length;

//рахуємо сигнали

signals = new HashSet<String>();

ArrayList<String> vertexNamesSignals = graphPanel.getVertexNamesPlus();

for (int i = 0; i < vertexNamesSignals.size(); i++) {

String[] temp = vertexNamesSignals.get(i).split("/");

for (int j = 0; j < temp[1].length(); j++) {

String[] thisSignals = temp[1].split(",");

for (int k = 0; k < thisSignals.length; k++) {

if (!thisSignals[k].equals("-")) {

signals.add(thisSignals[k]);

}

}

}

}

ArrayList<String> signalsArray = new ArrayList<String>();

for (String s : signals) {

signalsArray.add(s);

}

columnCount += signals.size();

//column names

columnNames = new String[columnCount];//3 - size of marking

columnNames[0] = "Перехід";

columnNames[1] = "Старий стан";

columnNames[2] = "Новий стан";

for (int i = 0; i < conditions.length; i++) {

columnNames[i + 3] = conditions[i];

}

for (int i = 0; i < signalsArray.size(); i++) {

columnNames[i + 3 + conditions.length] = signalsArray.get(i);

}

int count1 = 0;

for (int i = 4 + conditions.length + signalsArray.size(); i <= columnNames.length; i += 2) {

columnNames[i - 1] = "J" + ( markingSize - count1);

columnNames[i] = "K" + ( markingSize-count1);

count1++;

}

ArrayList<String> vertexNames = graphPanel.getVertexNames();

HashMap<String, Integer> vertexMarking = graphPanel.getVertexMarking();

ArrayList<String> moorGraphTable = graphPanel.getMoorGraphTable();

int[][] vertexesConnections = createConnectionMatrix(moorGraphTable, mooreMachine);

dataArrayL = new ArrayList<String>();

//переходи

int count = 0;

for (int i = 0; i < vertexesConnections.length; i++) {

dataArrayL.add(vertexNames.get(i) + "→");

for (int j = 0; j < vertexesConnections.length; j++) {

if (vertexesConnections[i][j] == 1) {

String temp = dataArrayL.get(count);

temp += vertexNames.get(j);

dataArrayL.remove(count);

dataArrayL.add(temp);

dataArrayL.add(vertexNames.get(i) + "→");

count++;

}

if ((j == vertexesConnections.length - 1)) {

dataArrayL.remove(count);

}

}

}

// умови

String[] condTrans = graphPanel.getConditionTransitions();

ArrayList<Integer> fromVertex = graphPanel.getFromVertex();

ArrayList<Integer> toVertex = graphPanel.getToVertex();

//форматуємо умови

String[] formatCondTrans = new String[condTrans.length];

for (int i = 0; i < dataArrayL.size(); i++) {

String[] temp = dataArrayL.get(i).split("→");

for (int j = 0; j < fromVertex.size(); j++) {

if ((temp[0].equals("Z" + (fromVertex.get(j) + 1)) & (temp[1].equals("Z" + (toVertex.get(j) + 1))))) {

formatCondTrans[i] = condTrans[j];

break;

}

}

}

//старий, новий стани

for (int i = 0; i < dataArrayL.size(); i++) {

String[] temp = dataArrayL.get(i).split("→");

String oldMark = editMarking(Integer.toBinaryString(vertexMarking.get(temp[0])));

String newMark = editMarking(Integer.toBinaryString(vertexMarking.get(temp[1])));

String tempS = dataArrayL.get(i) + splitLine + oldMark + splitLine + newMark + splitLine;

dataArrayL.add(i, tempS);

dataArrayL.remove(i + 1);

}

//додаємо умови

for (int i = 0; i < dataArrayL.size(); i++) {

String tempSplitLine = splitLine;

for (int j = 0; j < conditions.length; j++) {

if (j == conditions.length - 1) {

tempSplitLine = "";

}

String temp = dataArrayL.get(i);

if (formatCondTrans[i].contains("!" + conditions[j])) {

temp += "0" + tempSplitLine;

} else {

if (formatCondTrans[i].contains(conditions[j])) {

temp += "1" + tempSplitLine;

} else {

temp += "\*" + tempSplitLine;

}

}

dataArrayL.add(i, temp);

dataArrayL.remove(i + 1);

}

}

//додаємо сигнали

for (int i = 0; i < signalsArray.size(); i++) {

for (int j = 0; j < dataArrayL.size(); j++) {

String temp = dataArrayL.get(j);

String[] tempArr = temp.split("→");

//шукаємо відповідну вершину з сигналами

for (int k = 0; k < vertexNamesSignals.size(); k++) {

if (vertexNamesSignals.get(k).contains(tempArr[0])) {

if (vertexNamesSignals.get(k).contains(signalsArray.get(i))) {

temp += splitLine + "1";

} else {

temp += splitLine + "0";

}

dataArrayL.add(j, temp);

dataArrayL.remove(j + 1);

break;

}

}

}

}

//тригери

for (int i = 0; i < dataArrayL.size(); i++) {

String[] line = dataArrayL.get(i).split(splitLine);

String[] oldMark = line[1].split("");//01..markingsize

String[] newMark = line[2].split("");

for (int j = 1; j <= markingSize; j++) {

if (oldMark[j].equals("0")) {

if (newMark[j].equals("0")) {

String tempS = dataArrayL.get(i) + splitLine + "0" + splitLine + "\*";

dataArrayL.add(i, tempS);

dataArrayL.remove(i + 1);

} else {

String tempS = dataArrayL.get(i) + splitLine + "1" + splitLine + "\*";

dataArrayL.add(i, tempS);

dataArrayL.remove(i + 1);

}

} else {

if (newMark[j].equals("0")) {

String tempS = dataArrayL.get(i) + splitLine + "\*" + splitLine + "1";

dataArrayL.add(i, tempS);

dataArrayL.remove(i + 1);

} else {

String tempS = dataArrayL.get(i) + splitLine + "\*" + splitLine + "0";

dataArrayL.add(i, tempS);

dataArrayL.remove(i + 1);

}

}

}

}

//конвертування

data = new String[dataArrayL.size()][columnCount];

for (int i = 0; i < dataArrayL.size(); i++) {

data[i] = dataArrayL.get(i).split(splitLine);

}

Minimizator minimizator = new Minimizator(columnNames, data, conditions, markingSize, signals);

minimizator.getUnMinimizatedFunctions();

JTable table = new JTable(data, columnNames);

JScrollPane scrollPane = new JScrollPane(table);

add(scrollPane, BorderLayout.CENTER);

setVisible(true);

}

// встановлення розмірів та позициї вікна

private void setSize\_Position() {

Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

int frameWidth = screenSize.width \* 2 / 3;

int frameHeight = screenSize.height \* 2 / 3;

int leftCornerH = (screenSize.height - frameHeight) / 2;

int leftCornerW = (screenSize.width - frameWidth) / 2;

setBounds(leftCornerW, leftCornerH, frameWidth, frameHeight);

setPreferredSize(new Dimension(frameWidth, frameHeight));

setLayout(new BorderLayout());

}

//створює матрицю переходів автомата

private int[][] createConnectionMatrix(ArrayList<String> moorGraphTable, MooreMachine mooreMachine) {

ArrayList<String[]> lines = new ArrayList<String[]>();

if (moorGraphTable.size() != 2) {

for (int i = 2; i < moorGraphTable.size(); i++) {

lines.add(moorGraphTable.get(i).split(splitLine));

}

}

String[] line2 = moorGraphTable.get(1).split(splitLine);

int countOfConditions = 0;

for (int i = 0; i < line2.length; i++) {

if (line2[i].equals("")) {

countOfConditions++;

}

if (line2[i].equals("-")) {

break;

}

}

String[] line1 = moorGraphTable.get(0).split(splitLine);

int countOfVertexes = line1.length - countOfConditions;

int[][] vertexesConnections = new int[countOfVertexes][countOfVertexes];

for (int i = 0; i < countOfVertexes; i++) {

for (int j = 0; j < Math.pow(2, countOfConditions); j++) {

vertexesConnections[i][getVertexPos(lines.get(j)[countOfConditions + i], moorGraphTable.get(0).split(splitLine), countOfConditions, countOfVertexes)] = 1;

}

}

return vertexesConnections;

}

//визначає номер вершини за її позначенням

private int getVertexPos(String verName, String[] line1, int countOfConditions, int countOfVertexes) {

int count = 0;

for (int i = 0; i < countOfVertexes; i++) {

if (line1[countOfConditions + i].equals(verName)) {

break;

}

count++;

}

return count;

}

//форматування кодування

private String editMarking(String marking) {

if (marking.length() == markingSize) {

return marking;

}

int add = 0;

if (marking.length() == 1) {

add = 1;

}

for (int i = 0; i <= markingSize - marking.length() + add; i++) {

marking = "0" + marking;

}

return marking;

}

//повертає значення заголовків тадлиці переходів

public String[] getColumnNames() {

return columnNames;

}

//повертає значення заголовків тадлиці переходів

public ArrayList<String> getData() {

return dataArrayL;

}

}package lab1.gui;

import lab1.file.\*;

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class LSAMenuBar {

/\*\*

\* Створює випадаюче меню для управління проектом.

\*

\* @param mainPanel

\* @return

\*/

public JMenuBar getMenuBar(MainPanel mainPanel) {

JMenuBar jMenuBar = new JMenuBar();

JMenu fileMenu = new JMenu("Файл");

Icon newIcon = new ImageIcon("src/lab1/icons/new.png");

JMenuItem newProjectItem = new JMenuItem("Створити проект", newIcon);

newProjectItem.setToolTipText("Створити новий проект");

newProjectItem.addActionListener(new NewProjectListener(mainPanel));

Icon openIcon = new ImageIcon("src/lab1/icons/open.gif");

JMenuItem openProjectItem = new JMenuItem("Відкрити проект", openIcon);

openProjectItem.setToolTipText("Відкрити раніше створений проект");

openProjectItem.addActionListener(new OpenButtonListener(mainPanel));

Icon saveIcon = new ImageIcon("src/lab1/icons/save.png");

JMenuItem saveProjectItem = new JMenuItem("Зберегти проект", saveIcon);

saveProjectItem.setToolTipText("Зберегти проект");

saveProjectItem.addActionListener(new SaveButtonListener(mainPanel));

Icon openMoorIcon = new ImageIcon("src/lab1/icons/openMoor.gif");

JMenuItem openMoor = new JMenuItem("Відкрити автомат Мура", openMoorIcon);

openMoor.setToolTipText("Відкрити раніше створений автомат Мура з текстового файлу");

openMoor.addActionListener(new OpenMoorListener());

Icon saveMoorIcon = new ImageIcon("src/lab1/icons/saveMoor.png");

JMenuItem saveMoor = new JMenuItem("Зберегти автомат Мура", saveMoorIcon);

saveMoor.setToolTipText("Зберегти автомат Мура в текстовий файл");

saveMoor.addActionListener(new SaveMoorListener(mainPanel));

Icon saveMoorTableIcon = new ImageIcon("src/lab1/icons/saveMoorTable.png");

JMenuItem saveMoorTable = new JMenuItem("Зберегти таблицю переходів", saveMoorTableIcon);

saveMoorTable.setToolTipText("Зберегти таблицю автомата Мура в текстовий файл");

saveMoorTable.addActionListener(new SaveMoorTableListener(mainPanel));

JMenuItem exitItem = new JMenuItem("Вийти");

exitItem.setToolTipText("Не збережені дані будуть втрачені!!!");

exitItem.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

System.exit(0);

}

});

fileMenu.add(newProjectItem);

fileMenu.add(openProjectItem);

fileMenu.add(saveProjectItem);

fileMenu.addSeparator();

fileMenu.add(openMoor);

fileMenu.add(saveMoor);

fileMenu.addSeparator();

fileMenu.add(saveMoorTable);

fileMenu.addSeparator();

fileMenu.add(exitItem);

jMenuBar.add(fileMenu);

return jMenuBar;

}

}

**Висновок**

На основі раніше закодованого графу переходів автомата Мура розроблено та реалізовано алгоритми побудови таблиці переходів графу автомата Мура. Таблиця відображена в графічному інтерфейсі користувача.

Розроблено процедуру збереження таблиці переходів автомата Мура в текстовий файл.