

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

ЛАБОРАТОРНА РОБОТА №3

З КУРСУ

«ТЕХНОЛОГІЯ ПРОЕКТУВАННЯ КОМП’ЮТЕРНИХ СИСТЕМ»

Виконав:

студент IV курсу

групи ІО-21

Журо Георгій Олександрович

Київ – 2015

**Тема:** Автоматизація розмітки блок-схем алгоритмів.

**Мета:** Здобуття навичок з автоматизації процедури розмітки алгоритмів за методами Мілі та Мура.

**Завдання:**

1. В залежності від значення розрядів номера залікової книжки розмітити алгоритм, перевірений на наявність помилок у попередній роботі згідно методу.

**Метод розмітки:** Мура.

1. Згідно отриманої розмітки графічно відобразити граф переходів для алгоритму.
2. Розробити формат файлу для зберігання графу переході. Реалізувати функцію збереження/відновлення графу переходів.

**Тип формату:** Бінарний.

**Опис програми:**

Для того, щоб розмітити методом Мура блок-схему алгоритму, побудовану редактором блок-схем (лабораторна робота №1), та перевірену на наявність безкінечних циклів (лабораторна робота №2), необхідно вибрати пункт головного меню Execute -> Mark Algorithm For Moore Automat. Для алгоритму, що зображений на рисунку 1, розмічена таким чином блок-схема показана на рисунку 2.

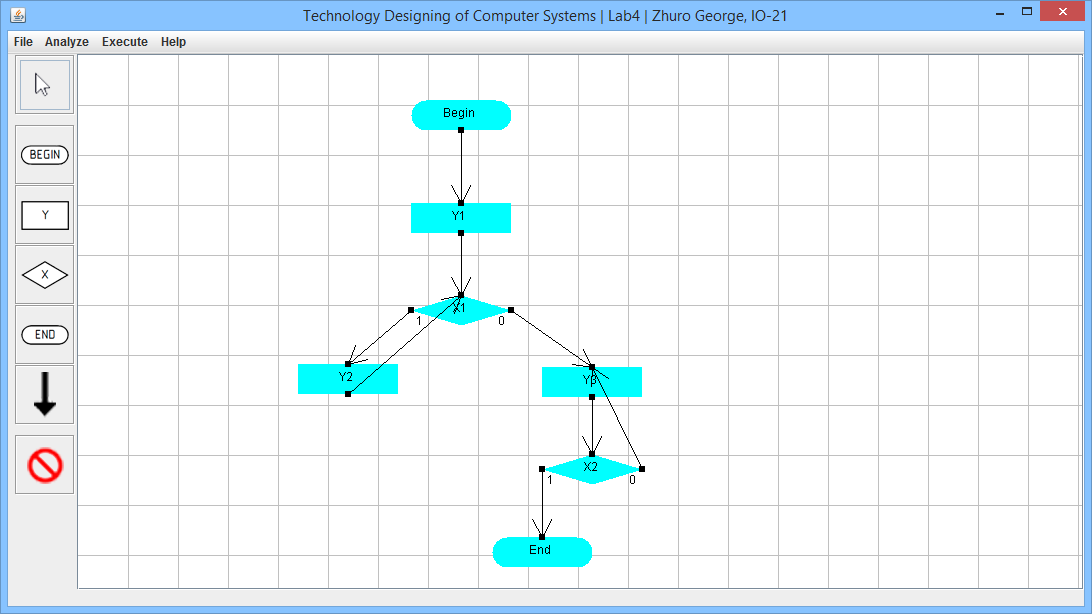


Рисунок 1

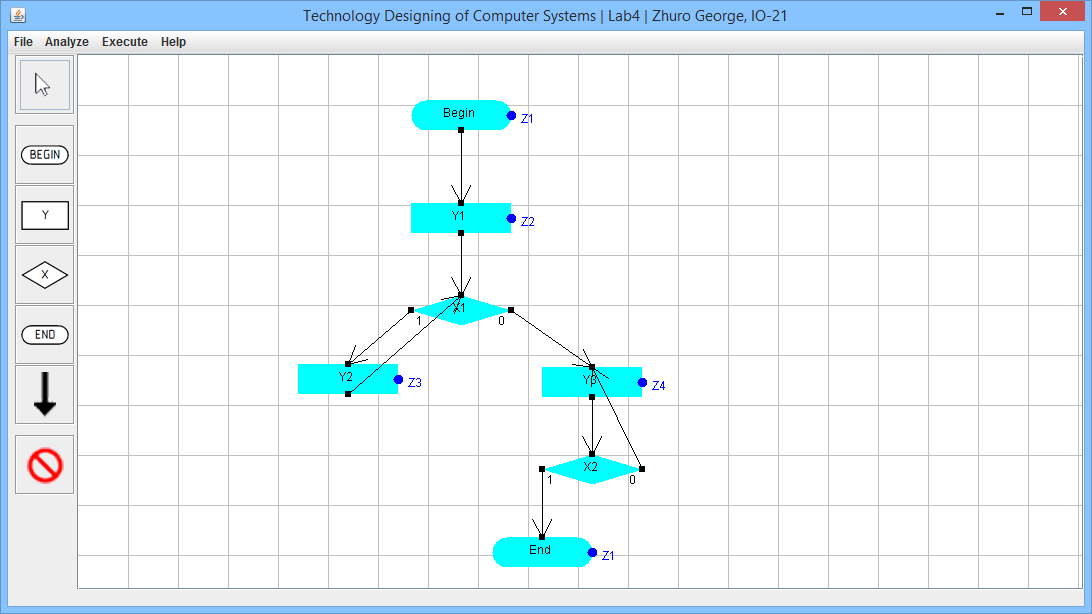


Рисунок 2

Після цього можна побудувати граф переходів для автомату Мура. Для цього необхідно вибрати пункт головного меню Execute -> Build Graph of Moore Automat… . Результат даної дії для даного алгоритму (рисунок 2) зображений на рисунку 3. Дана операція була реалізована за допомогою видалення з матриці зв’язків логічних вершин та запису в матрицю умов переходів, що визначені за допомогою рекурсивного алгоритму пошуку наступної кінцевої або операторної вершини від кожної операторної та початкової вершин алгоритму.

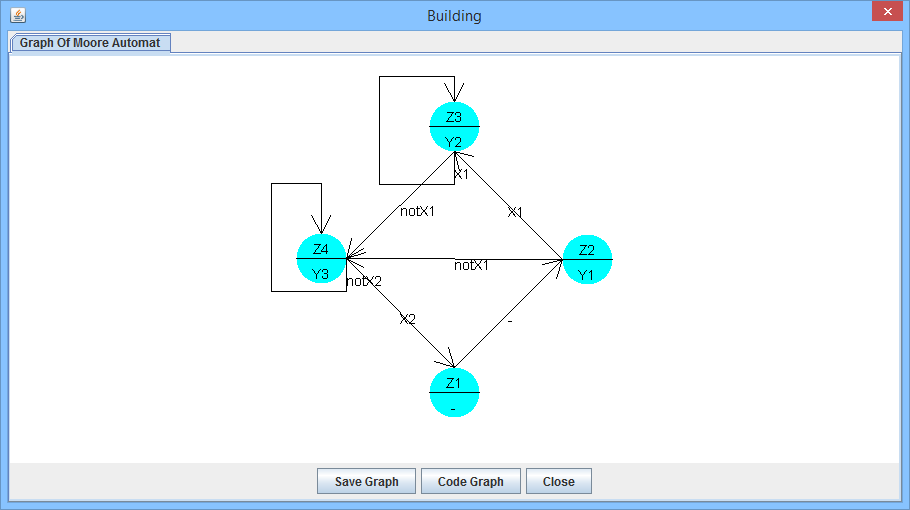


Рисунок 3

Для збереження отриманого графу переходів необхідно натиснути кнопку Save Graph та ввести ім’я файлу.

Для відновлення збереженого графу переходів необхідно скористатися пунктом головного меню File -> Open Graph Of Moore Automat. Граф зберігається у файл бінарного формату. Це реалізовано за допомогою серіалізації об’єкту класу MooreAutomat, що містить всі дані про граф переходів.

**Лістинг програми:**

package automat.moore;

import gsa.GSAModel;

import java.io.\*;

public class MooreAutomat implements Serializable {

private String[] stateNames;

private int[][] yNumbers;

private int[][] connectionMatrix;

private int[][] xNumbers;

private boolean[][] xValues;

public MooreAutomat(String[] stateNames, int[][] yNumbers, int[][] connectionMatrix, int[][] xNumbers, boolean[][] xValues) {

this.stateNames = stateNames;

this.yNumbers = yNumbers;

this.connectionMatrix = connectionMatrix;

this.xNumbers = xNumbers;

this.xValues = xValues;

}

public MooreAutomat(GSAModel model) {

int[] nodesType = model.getNodesType();

int[][] nodesConnectionMatrix = model.getConnectionMatrix();

int[][] signalMatrix = model.getSignalMatrix();

int count = 0;

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

if (nodesType[i] == 2) {

count++;

}

}

stateNames = new String[nodesType.length - count - 1];

connectionMatrix = new int[nodesType.length - count - 1][];

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

connectionMatrix[i] = new int[nodesType.length - count - 1];

for (int j = 0; j < connectionMatrix[i].length; j++) {

connectionMatrix[i][j] = -1;

}

}

yNumbers = new int[nodesType.length - count - 1][];

int z1 = -1;

int z = 2;

int j = 0;

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

if (((nodesType[i] == 0) || (nodesType[i] == 3)) && (z1 == -1)) {

z1 = j;

stateNames[j] = "Z1";

yNumbers[j] = null;

j++;

}

else {

if (nodesType[i] == 1) {

stateNames[j] = "Z" + String.valueOf(z++);

yNumbers[j] = new int[signalMatrix[i].length];

for (int k = 0; k < signalMatrix[i].length; k++) {

yNumbers[j][k] = signalMatrix[i][k];

}

j++;

}

}

}

xNumbers = null;

xValues = null;

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

if ((nodesType[i] == 0) || (nodesType[i] == 1)) {

for (j = 0; j < nodesConnectionMatrix[i].length; j++) {

if (nodesType[j] == 1) {

if (nodesConnectionMatrix[i][j] > 0) {

int[][] newXNumbers;

if (xNumbers != null) {

newXNumbers = new int[xNumbers.length + 1][];

for (int x = 0; x < xNumbers.length; x++) {

newXNumbers[x] = xNumbers[x];

}

}

else {

newXNumbers = new int[1][];

}

newXNumbers[newXNumbers.length - 1] = null;

xNumbers = newXNumbers;

boolean[][] newXValues;

if (xValues != null) {

newXValues = new boolean[xValues.length + 1][];

for (int x = 0; x < xValues.length; x++) {

newXValues[x] = xValues[x];

}

} else {

newXValues = new boolean[1][];

}

newXValues[newXValues.length - 1] = null;

xValues = newXValues;

connectionMatrix[getStateNumber(nodesType, i)][getStateNumber(nodesType, j)] = xNumbers.length - 1;

}

}

else {

if (nodesType[j] == 3) {

if (nodesConnectionMatrix[i][j] > 0) {

int[][] newXNumbers;

if (xNumbers != null) {

newXNumbers = new int[xNumbers.length + 1][];

for (int x = 0; x < xNumbers.length; x++) {

newXNumbers[x] = xNumbers[x];

}

} else {

newXNumbers = new int[1][];

}

newXNumbers[newXNumbers.length - 1] = null;

xNumbers = newXNumbers;

boolean[][] newXValues;

if (xValues != null) {

newXValues = new boolean[xValues.length + 1][];

for (int x = 0; x < xValues.length; x++) {

newXValues[x] = xValues[x];

}

} else {

newXValues = new boolean[1][];

}

newXValues[newXValues.length - 1] = null;

xValues = newXValues;

connectionMatrix[getStateNumber(nodesType, i)][getStateNumber(nodesType, j)] = xNumbers.length - 1;

}

}

else {

if (nodesType[j] == 2) {

if (nodesConnectionMatrix[i][j] > 0) {

int[] newXNumbers = null;

boolean[] newXValues = null;

stepToOperatorNode(getStateNumber(nodesType, i), j, newXNumbers, newXValues, nodesType, nodesConnectionMatrix, signalMatrix);

}

}

}

}

}

}

}

}

public static void writeToFile(File file, MooreAutomat automat) throws IOException {

ObjectOutputStream output = new ObjectOutputStream(new FileOutputStream(file));

output.writeObject(automat);

output.close();

}

public static MooreAutomat readFromFile(File file) throws IOException, ClassNotFoundException {

ObjectInputStream input = new ObjectInputStream(new FileInputStream(file));

MooreAutomat automat = (MooreAutomat) input.readObject();

input.close();

return automat;

}

private void stepToOperatorNode(int start, int from, int[] stepXNumbers, boolean[] stepXValues, int[] nodesType,

int[][] nodesConnectionMatrix, int[][] signalMatrix) {

for (int i = 0; i < nodesConnectionMatrix[from].length; i++) {

if (nodesConnectionMatrix[from][i] == 1) {

int[] stepNewXNumbers;

if (stepXNumbers != null) {

stepNewXNumbers = new int[stepXNumbers.length + 1];

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

stepNewXNumbers[j] = stepXNumbers[j];

}

} else {

stepNewXNumbers = new int[1];

}

stepNewXNumbers[stepNewXNumbers.length - 1] = signalMatrix[from][0];

boolean[] stepNewXValues;

if (stepXValues != null) {

stepNewXValues = new boolean[stepXValues.length + 1];

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

stepNewXValues[j] = stepXValues[j];

}

} else {

stepNewXValues = new boolean[1];

}

stepNewXValues[stepNewXValues.length - 1] = true;

if (nodesType[i] == 2) {

stepToOperatorNode(start, i, stepNewXNumbers, stepNewXValues, nodesType, nodesConnectionMatrix, signalMatrix);

} else {

int[][] newXNumbers;

if (xNumbers != null) {

newXNumbers = new int[xNumbers.length + 1][];

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

newXNumbers[j] = xNumbers[j];

}

} else {

newXNumbers = new int[1][];

}

newXNumbers[newXNumbers.length - 1] = stepNewXNumbers;

xNumbers = newXNumbers;

boolean[][] newXValues;

if (xValues != null) {

newXValues = new boolean[xValues.length + 1][];

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

newXValues[j] = xValues[j];

}

} else {

newXValues = new boolean[1][];

}

newXValues[newXValues.length - 1] = stepNewXValues;

xValues = newXValues;

connectionMatrix[start][getStateNumber(nodesType, i)] = newXNumbers.length - 1;

}

} else {

if (nodesConnectionMatrix[from][i] == 2) {

int[] stepNewXNumbers;

if (stepXNumbers != null) {

stepNewXNumbers = new int[stepXNumbers.length + 1];

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

stepNewXNumbers[j] = stepXNumbers[j];

}

} else {

stepNewXNumbers = new int[1];

}

stepNewXNumbers[stepNewXNumbers.length - 1] = signalMatrix[from][0];

boolean[] stepNewXValues;

if (stepXValues != null) {

stepNewXValues = new boolean[stepXValues.length + 1];

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

stepNewXValues[j] = stepXValues[j];

}

} else {

stepNewXValues = new boolean[1];

}

stepNewXValues[stepNewXValues.length - 1] = false;

if (nodesType[i] == 2) {

stepToOperatorNode(start, i, stepNewXNumbers, stepNewXValues, nodesType, nodesConnectionMatrix, signalMatrix);

} else {

int[][] newXNumbers = new int[xNumbers.length + 1][];

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

newXNumbers[j] = xNumbers[j];

}

newXNumbers[newXNumbers.length - 1] = stepNewXNumbers;

xNumbers = newXNumbers;

boolean[][] newXValues = new boolean[xValues.length + 1][];

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

newXValues[j] = xValues[j];

}

newXValues[newXValues.length - 1] = stepNewXValues;

xValues = newXValues;

connectionMatrix[start][getStateNumber(nodesType, i)] = newXNumbers.length - 1;

}

}

}

}

}

private int getStateNumber(int[] nodesType, int n) {

int z1 = -1;

int j = 0;

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

if (nodesType[i] == 1) {

j++;

}

else {

if (((nodesType[i] == 0) || (nodesType[i] == 3)) && (z1 == -1)) {

z1 = j;

j++;

}

}

}

if ((nodesType[n] == 0) || (nodesType[n] == 3)) {

if (z1 == -1) {

return n;

}

else {

return z1;

}

}

else {

return j;

}

}

public String[] getStateNames() {

return stateNames;

}

public int[][] getyNumbers() {

return yNumbers;

}

public int[][] getConnectionMatrix() {

return connectionMatrix;

}

public int[][] getxNumbers() {

return xNumbers;

}

public boolean[][] getxValues() {

return xValues;

}

}

**Висновки:** В результаті виконання даної лабораторної роботи я здобув навички з автоматизації процедури розмітки алгоритму методом Мура. Я реалізував побудову графу за допомогою видалення логічних вершин з матриці зв’язків та запису в неї умов переходів, які я визначив за допомогою рекурсивного алгоритму пошуку наступної операторної або кінцевої вершини від початкової та кожної операторної вершини. Також я здобув навички з візуалізації графів та роботи з бінарними файлами. Програма була написана на мові Java. Для візуалізації графу переходів був використаний пакет javax.swing.\*.