Kpi-best

МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра обчислювальної техніки

Лабораторна робота №5

з Теорії Проектування Комп’ютерних Систем-1

Виконав студент групи ІО-11

Ротенберг О.В

Київ — 2014

***Тема:*** Автоматизація синтезу таблиці переходів.

***Мета:*** Здобуття навичок з аналізу графових структур і автоматизації процедури побудови таблиці переходів.

1. Представити номер залікової книжки в двійковому вигляді:

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

тригера:

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

4. Реалізувати засоби відображення таблиці (п.3) та її збереження у файлі.

Тип триггера: Т.

Таблица переходов для Т-триггера:

0 ->0 = 0

0 ->1 = 1

1 -> 0 = 0

1 ->1 = 1

Лістинг:

package redactor.gui.graph.mimimisation;

import java.util.ArrayList;

public class TransitionData {

ArrayList<String[]> transitions;

int numberOfTriggers;

ArrayList<int[]> listOfStates;

ArrayList<String> conditions;

ArrayList<String> operations;

String[] header;

int rowLength;

public TransitionData(ArrayList<String[]> transitions, int numberOfTriggers) {

this.transitions = transitions;

this.numberOfTriggers = numberOfTriggers;

listOfStates = new ArrayList<int[]>();

conditions = new ArrayList<String>();

operations = new ArrayList<String>();

findAllConditions();

findAllOperations();

rowLength = numberOfTriggers + 4 + conditions.size() + operations.size();

header = new String[rowLength];

generateHeader();

fillStateTable();

printTable();

}

public void fillStateTable() {

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

int[] trans = new int[rowLength];

trans[0] = Integer.valueOf(transitions.get(i)[0]);

trans[1] = Integer.valueOf(transitions.get(i)[1]);

trans[2] = Integer.valueOf(transitions.get(i)[2]);

trans[3] = Integer.valueOf(transitions.get(i)[3]);

listOfStates.add(trans);

}

generateTriggerStates();

generateConditionStates();

generateOperationState();

}

public void generateHeader() {

header[0] = "Zold";

header[1] = "Znew";

header[2] = "Mold";

header[3] = "Mnew";

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

header[4 + i] = conditions.get(i);

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

header[4 + conditions.size() + i] = operations.get(i);

for (int i = 0; i < numberOfTriggers; i++)

header[4 + conditions.size() + operations.size() + i] = "T" + (i + 1);

}

public void generateTriggerStates() {

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

int[] current = listOfStates.get(i);

int temp = current[3];

for (int j = 0; j < numberOfTriggers; j++) {

current[current.length - j - 1] = temp % 2;

temp /= 2;

}

}

}

public boolean suchElementAlreadyExist(String element, ArrayList<String> list) {

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

if (element.equals(list.get(i)))

return true;

return false;

}

public void generateConditionStates(){

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

int[] currentLine = listOfStates.get(i);

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

currentLine[4+j] = -1;

String conditionText = transitions.get(i)[4];

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

String temp = "";

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

if (conditionText.charAt(j) != ')')

temp += conditionText.charAt(j);

else {

separatedConditions.add(temp);

temp = "";

}

}

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

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

if (separatedConditions.get(k).contains(conditions.get(j))){

if (separatedConditions.get(k).contains("!"))

currentLine[4+j] = 0;

else currentLine[4+j] = 1;

}

}

}

}

}

public void generateOperationState(){

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

int[] currentLine = listOfStates.get(i);

String operationText = transitions.get(i)[5];

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

if (operationText.contains(operations.get(j)))

currentLine[4+conditions.size()+j] = 1;

}

}

}

public void findAllConditions() {

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

String current = transitions.get(i)[4];

if (!current.equals("-")) {

String condition = "";

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

if ((current.charAt(j) != '(') && (current.charAt(j) != '!') && (current.charAt(j) != ')'))

condition += current.charAt(j);

if (current.charAt(j) == ')') {

if (!suchElementAlreadyExist(condition, conditions))

conditions.add(condition);

condition = "";

}

}

}

}

}

public void findAllOperations() {

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

String current = transitions.get(i)[5];

if (!current.equals("-")) {

String operation = "";

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

if (current.charAt(j) != ',')

operation += current.charAt(j);

else {

if (!suchElementAlreadyExist(operation, operations))

operations.add(operation);

operation = "";

}

}

if (!suchElementAlreadyExist(operation, operations))

operations.add(operation);

}

}

}

public void printTable() {

System.out.println("header");

for (int i = 0; i < rowLength; i++)

System.out.print(header[i] + " ");

System.out.println();

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

int[] current = listOfStates.get(i);

System.out.print(current[0] + " --> " + current[1] + "; "

+ Integer.toBinaryString(current[2]) + " --> " + Integer.toBinaryString(current[3]) + "|| ");

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

if (current[4+j] != -1)

System.out.print(current[4+j]+" ");

else System.out.print("-" + " ");

System.out.print("|| ");

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

System.out.print(current[4+conditions.size()+j]+" ");

System.out.print("|| ");

for (int j = 0; j<numberOfTriggers; j++)

System.out.print(current[4+conditions.size()+operations.size()+j]+" ");

System.out.println();

}

}

public ArrayList<int[]> getListOfStates() {

return listOfStates;

}

public String[] getHeader() {

return header;

}

public int getNumberOfTriggers() {

return numberOfTriggers;

}

}

package redactor.gui.graph.mimimisation;

import redactor.Data;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.ArrayList;

public class TransitionTextCreator {

TransitionData transitionData;

public TransitionTextCreator(TransitionData transitionData) {

this.transitionData = transitionData;

}

public boolean writeToFile(final String path) {

try {

PrintWriter out = new PrintWriter(new FileOutputStream(path));

if (transitionData != null) {

String[] header = transitionData.getHeader();

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

out.print(header[i]+"\t");

ArrayList<int[]> table = transitionData.getListOfStates();

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

int[] current = table.get(i);

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

out.print(current[j]+"\t");

out.println();

}

}

out.close();

} catch (IOException e) {

e.printStackTrace();

return false;

}

return true;

}

}

**Висновок**

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

Розроблено процедуру збереження таблиці переходів автомата Мілі в текстовий файл.