Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №6**

*з предмету: «ТПКС»*

*по темі:*

«Автоматизація мінімізації булевих функцій»

Виконав: студент ФІОТ

групи ІО-92

Петрук В.О.

Київ 2012р.

**Мета:** Здобуття навичок автоматизації процедури мінімізації булевих функцій

методом Квайна-МакКласкі.

**Завдання**

1. Розробити процедуру мінімізації булевих функцій методом Квайна-МакКласкі.

2. Розробити засоби аналізу ефективності мінімізації по кількості елементів,

кількості входів/виходів, довжині критичного шляху.

3. На основі розробленої процедури (п.1) реалізувати модуль мінімізації булевих

функцій переходів і функцій збудження тригерів з таблиці, побудованої в

попередній роботі. Передбачити відображення немінімізованих/мінімізованих

функцій та ефективності їх мінімізації (п.2).

4. Реалізувати засоби збереження результатів мінімізації у файлі.

**Опис програми**

**Мінімізація методом Квайна-Мак-Класкі**

Дані для мінімізації беруться з структурної таблиці і представляються у вигляді колекції рядків. Мінімізація проводиться по одиницям (ДДНФ).

**Алгоритм мінімізації**

Проглядаються всі пари рядків. Якщо знайдені такі, що відрізняються в єдиному розряді, кожна з них відзначається на видалення, а в список мінімізованих додається рядок, де за вказаним симвоом, що відрізняється ставиться \*. Після проходу по списку перевіряється чи були мінімізації. Якщо так, всі зазначені масиви видаляються, додаються мінімізовані значення і процедура реуурсивно повторюється. В іншому випадку функція не підлягає подальшій мінімізації.

**Код програми**

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

package lab1.minimisation;

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Set;

public class Minimizator {

private String[] columnNames;

private String[][] data;

private String[] conditions;

private int markingSize;

private Set<String> signals;

private ArrayList<String> unminimizedFunctions;

private ArrayList<String> unminimizedFunctionsShow;

private ArrayList<String[]> minimizedFunctions;

private ArrayList<String> minimizedFunctionsShow;

private int pos;

private ArrayList<String> endFunctions;

//конструктор класа мінімізації методом Квайна-МакКласкі

public Minimizator(String[] columnNames, String[][] data, String[] conditions, int markingSize, Set<String> signals) {

this.columnNames = columnNames;

this.data = data;

this.conditions = conditions;

this.markingSize = markingSize;

this.signals = signals;

}

//повретає колекцію не мінімізований функцій

public ArrayList<String> getUnMinimizatedFunctions() {

unminimizedFunctions = new ArrayList<String>();

unminimizedFunctionsShow = new ArrayList<String>();

for (int i = 3 + conditions.length; i < columnNames.length; i++) {

unminimizedFunctions.add("" + columnNames[i] + " = ");

unminimizedFunctionsShow.add("");

}

for (int i = 3; i < unminimizedFunctions.size() + 3; i++) {

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

if (data[j][i + conditions.length].equals("1")) {

String temp = unminimizedFunctions.get(i - 3) + data[j][1];

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

temp += data[j][3 + k];

}

temp += " v ";

unminimizedFunctions.add(i - 3, temp);

unminimizedFunctions.remove(i - 3 + 1);

}

}

}

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

String temp = unminimizedFunctions.get(i);

temp = temp.substring(0, temp.length() - 3);

unminimizedFunctions.add(i, temp);

unminimizedFunctions.remove(i + 1);

}

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

int count = 0;

boolean flag = false;

for (int j = 0; j < unminimizedFunctions.get(i).length(); j++) {

if (flag) {

String sign = "Q";

if (count < markingSize) {

sign += (markingSize - count);

} else {

sign = columnNames[3 + count - markingSize];

}

if (unminimizedFunctions.get(i).charAt(j) == '1') {

String temp = unminimizedFunctionsShow.get(i) + sign;

unminimizedFunctionsShow.add(i, temp);

unminimizedFunctionsShow.remove(i + 1);

count++;

} else {

if (unminimizedFunctions.get(i).charAt(j) == '0') {

String temp = unminimizedFunctionsShow.get(i) + "!" + sign;

unminimizedFunctionsShow.add(i, temp);

unminimizedFunctionsShow.remove(i + 1);

count++;

} else {

if (unminimizedFunctions.get(i).charAt(j) == '\*') {

String temp = unminimizedFunctionsShow.get(i) + "";

unminimizedFunctionsShow.add(i, temp);

unminimizedFunctionsShow.remove(i + 1);

count++;

} else {

String temp = unminimizedFunctionsShow.get(i) + unminimizedFunctions.get(i).charAt(j);

unminimizedFunctionsShow.add(i, temp);

unminimizedFunctionsShow.remove(i + 1);

count = 0;

}

}

}

} else {

String temp = unminimizedFunctionsShow.get(i) + unminimizedFunctions.get(i).charAt(j);

unminimizedFunctionsShow.add(i, temp);

unminimizedFunctionsShow.remove(i + 1);

if (unminimizedFunctions.get(i).charAt(j) == '=') {

flag = true;

}

}

}

}

System.out.println();

System.out.println("Немінімізовані функції");

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

System.out.println(unminimizedFunctionsShow.get(i) + " ");

}

getMinimizedFunctions(unminimizedFunctions);

return unminimizedFunctionsShow;

}

//повретає колекцію мінімізованих функцій

public ArrayList<String> getMinimizedFunctions(ArrayList<String> unminimizedFunctions) {

//розбиваємо на підмасиви

String[] names = new String[unminimizedFunctions.size()];

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

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

String[] arrays = unminimizedFunctions.get(i).split("v");

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

arrays[j] = arrays[j].trim();

}

names[i] = arrays[0];

arrays[0] = arrays[0].split("=")[1].trim();

minimizedFunctions.add(arrays);

}

System.out.println("Мінімізовані функції");

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

endFunctions = new ArrayList<String>();

String[] tempArray = minimize(minimizedFunctions.get(i));

tempArray = new String[endFunctions.size()];

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

tempArray[j] = endFunctions.get(j);

}

minimizedFunctions.add(i, tempArray);

minimizedFunctions.remove(i + 1);

}

minimizedFunctionsShow = new ArrayList<String>();

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

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

String line = "";

for (int j = 0; j < minimizedFunctions.get(i).length; j++) {

line += minimizedFunctions.get(i)[j] + " v ";

}

newMinim.add(line);

}

//-------------

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

minimizedFunctionsShow.add(names[i].split("=")[0] + " = ");

int count = 0;

for (int j = 0; j < newMinim.get(i).length(); j++) {

String sign = "Q";

if (count < markingSize) {

sign += (markingSize - count);

} else {

sign = columnNames[3 + count - markingSize];

}

if (newMinim.get(i).charAt(j) == '1') {

String temp = minimizedFunctionsShow.get(i) + sign;

minimizedFunctionsShow.add(i, temp);

minimizedFunctionsShow.remove(i + 1);

count++;

} else {

if (newMinim.get(i).charAt(j) == '0') {

String temp = minimizedFunctionsShow.get(i) + "!" + sign;

minimizedFunctionsShow.add(i, temp);

minimizedFunctionsShow.remove(i + 1);

count++;

} else {

if (newMinim.get(i).charAt(j) == '\*') {

String temp = minimizedFunctionsShow.get(i) + "";

minimizedFunctionsShow.add(i, temp);

minimizedFunctionsShow.remove(i + 1);

count++;

} else {

String temp = minimizedFunctionsShow.get(i) + newMinim.get(i).charAt(j);

minimizedFunctionsShow.add(i, temp);

minimizedFunctionsShow.remove(i + 1);

count = 0;

}

}

}

}

}

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

String temp = minimizedFunctionsShow.get(i);

temp = temp.substring(0, temp.length() - 3);

minimizedFunctionsShow.add(i, temp);

minimizedFunctionsShow.remove(i + 1);

}

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

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

}

saveToFile();

return minimizedFunctionsShow;

}

// мінімізація окремої функції

private String[] minimize(String[] currentFunction) {

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

int[] status = new int[currentFunction.length];

for (int i = 0; i < currentFunction.length - 1; i++) {

for (int j = i + 1; j < currentFunction.length; j++) {

if (isSimilar(currentFunction[i], currentFunction[j])) {

status[i] = 1;

status[j] = 1;

String nextRFunc = "";

for (int k = 0; k < pos; k++) {

nextRFunc += currentFunction[i].charAt(k);

}

nextRFunc += "\*";

for (int k = pos + 1; k < currentFunction[i].length(); k++) {

nextRFunc += currentFunction[i].charAt(k);

}

nextRankFunctions.add(nextRFunc);

}

}

}

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

if (status[i] != 1) {

endFunctions.add(currentFunction[i]);

}

}

String[] newFunctions = new String[nextRankFunctions.size()];

if (nextRankFunctions.size() != 0) {

//конвертуємо в масив

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

newFunctions[i] = nextRankFunctions.get(i);

}

newFunctions = minimize(newFunctions);

}

return newFunctions;

}

// чи відрізняються терми лише на 1 розряд?

private boolean isSimilar(String s1, String s2) {

boolean flag = false;

for (int i = 0; i < s1.length(); i++) {

if (s1.charAt(i) != s2.charAt(i)) {

pos = i;

if (flag) {

return false;

} else {

flag = true;

}

}

}

return flag;

}

//функція збереження мінімізованих та немінімізованих функцій у файл

private void saveToFile() {

try {

BufferedWriter newMoorFile = new BufferedWriter(new FileWriter("minimiz.mim"));

newMoorFile.write("Не мінімізовані функції\n");

for (String s: unminimizedFunctionsShow) {

newMoorFile.write(s+"\n");

}

newMoorFile.write("Мінімізовані функції\n");

for (String s: minimizedFunctionsShow) {

newMoorFile.write(s+"\n");

}

newMoorFile.close();

} catch (IOException e) {

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

}

}

}

**Висновок**

Розроблена та реалізована процедура мінімізації булевих функцій методом Квайна-МакКласкі на основі таблиці переходів графа автомата Мура. Результати мінімізації відображаються у графічному інтерфейсі користувача.

Розроблено процедуру збереження мінімізованих функцій в текстовий файл.