Kpi-best

МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра обчислювальної техніки

Лабораторна робота №6

з Теорії Проектування Комп’ютерних Систем-1

Виконав студент групи ІО-11

Ротенберг О.В

Київ — 2014

***Тема:*** Автоматизація мінімізації булевих функцій.

***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.

Лістинг: package redactor.gui.graph.mimimisation;

import java.util.ArrayList;

import java.util.List;

public class Mimimisator {

String[] header;

List<int[]> structureTable;

int markerSize;

List<KavaiForOneFunction> mimimisated;

public Mimimisator(TransitionData transitionData, int markerSize){

header = transitionData.getHeader();

structureTable = transitionData.getListOfStates();

this.markerSize = markerSize;

mimimisated = new ArrayList<KavaiForOneFunction>();

}

public void executeMimimisation(){

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

int index = 4;

while ((header[index].charAt(0) == 'X') ||

(header[index].charAt(0) == 'x')) {

cond.add(header[index]);

index++;

}

String[] conditionValues = new String[cond.size()] ;

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

conditionValues[i] = cond.get(i);

for (int i = 4+conditionValues.length; i< header.length; i++){

String name = header[i];

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

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

if (structureTable.get(j)[i] == 0 ) { //minimisation by 0

int[] temp = new int[1+conditionValues.length];

temp[0] = structureTable.get(j)[2];

for (int k = 1; k<temp.length; k++)

temp[k] = structureTable.get(j)[3+k] ;

information.add(temp);

}

}

KavaiForOneFunction kavaiForOneFunction = new KavaiForOneFunction(name,

conditionValues,information,markerSize);

kavaiForOneFunction.mimimisateFunction();

// kavaiForOneFunction.printInfo();

mimimisated.add(kavaiForOneFunction);

}

}

public List<KavaiForOneFunction> getMimimisated() {

return mimimisated;

}

}

package redactor.gui.graph.mimimisation;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Iterator;

import java.util.List;

public class KavaiForOneFunction {

private String name;

private String[] header;

private List<int[]> arguments;

private int argumentsSize;

private List<int[]> mimimisated;

private String notMinFunction;

private String minFunction;

private int numberOfInputsBeforeMimimi;

private int numberOfInputsAfterMimimi;

private int numberOfOutputsBeforeMimimi;

private int numberOfOutputsAfterMimimi;

public KavaiForOneFunction(String name, String[] conditionHeader,

ArrayList<int[]> QandXinformation,

int markerSize) {

this.name = name;

header = new String[markerSize + conditionHeader.length];

argumentsSize = QandXinformation.size();

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

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

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

header[i] = "Q" + (markerSize - i);

}

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

header[i] = conditionHeader[i - markerSize];

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

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

int q = current[0];

int[] temp = new int[header.length];

for (int j = markerSize - 1; j >= 0; j--) {

temp[j] = q % 2;

q /= 2;

}

for (int j = markerSize; j < temp.length; j++)

temp[j] = current[j - markerSize + 1];

arguments.add(temp);

}

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

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

int[] temp = Arrays.copyOf(current, current.length + 1);

if (!contains(mimimisated, temp))

mimimisated.add(temp);

}

notMinFunction = generateFunctionAsString();

numberOfInputsBeforeMimimi = countInputs();

numberOfOutputsBeforeMimimi = countOutputs();

}

public void mimimisateFunction() {

boolean needMoreMinimisation = true;

while (needMoreMinimisation) {

// System.out.println(generateFunctionAsString());

needMoreMinimisation = false;

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

for (int i = 0; i < mimimisated.size() - 1; i++) {

int[] previous = mimimisated.get(i);

for (int j = i + 1; j < mimimisated.size(); j++) {

int[] current = mimimisated.get(j);

int index = differenceByOne(previous, current);

if (index != -1) {

// System.out.println("dif in one: "+i+" "+j);

int[] temp = Arrays.copyOf(current, current.length);

temp[index] = -1;

temp[temp.length - 1] = 0;

mimimisatedInThisIteration.add(temp);

current[current.length - 1] = 1;

previous[current.length - 1] = 1;

}

}

}

if (mimimisatedInThisIteration.size() > 0) {

needMoreMinimisation = true;

for (Iterator<int[]> i = mimimisated.iterator(); i.hasNext(); ) {

int[] temp = i.next();

if (temp[temp.length - 1] == 1)

i.remove();

}

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

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

if (!contains(mimimisated, current))

mimimisated.add(current);

}

mimimisatedInThisIteration.clear();

}

}

minFunction = generateFunctionAsString();

numberOfInputsAfterMimimi = countInputs();

numberOfOutputsAfterMimimi = countOutputs();

// System.out.println("finally: "+ minFunction);

}

public boolean contains(List<int[]> list, int[] item) {

boolean flag = false;

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

int[] temp = list.get(i);

flag = true;

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

if (temp[j] != item[j])

flag = false;

}

return flag;

}

/\*\*

\* doesn't check last array value (because it's only a marker)

\*

\* @param arg1

\* @param arg2

\* @return

\*/

public int differenceByOne(int[] arg1, int[] arg2) {

ArrayList<Integer> differencePositions = new ArrayList<Integer>();

for (int i = 0; i < arg1.length - 1; i++)

if (arg1[i] != arg2[i])

differencePositions.add(i);

if (differencePositions.size() == 1)

return differencePositions.get(0);

return -1;

}

public void printTransTableForThis() {

System.out.println("name = " + name);

System.out.println("arguments");

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

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

System.out.println();

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

int[] temp = arguments.get(i);

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

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

System.out.println();

}

}

public String generateFunctionAsString() {

String function = name + " = (";

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

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

for (int j = 0; j < current.length - 1; j++) {

switch (current[j]) {

case 0:

function += "!" + header[j];

break;

case 1:

function += header[j];

break;

}

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

function += " v ";

}

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

if (i < mimimisated.size() - 1)

function += ") & (";

else

function += ")";

}

return function;

}

public int countInputs() {

int counter = 0;

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

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

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

if (current[j] != -1)

counter++;

}

return counter;

}

public int countOutputs() {

return mimimisated.size();

}

public String getName() {

return name;

}

public String getNotMinFunction() {

return notMinFunction;

}

public int getNumberOfInputsBeforeMimimi() {

return numberOfInputsBeforeMimimi;

}

public String getMinFunction() {

return minFunction;

}

public int getNumberOfInputsAfterMimimi() {

return numberOfInputsAfterMimimi;

}

public int getNumberOfOutputsBeforeMimimi() {

return numberOfOutputsBeforeMimimi;

}

public int getNumberOfOutputsAfterMimimi() {

return numberOfOutputsAfterMimimi;

}

}

package redactor.gui.graph.mimimisation;

import redactor.Data;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.ArrayList;

import java.util.List;

public class TransitionAndMimimiTextCreator {

TransitionData transitionData;

List<KavaiForOneFunction> mimimiList;

public TransitionAndMimimiTextCreator(TransitionData transitionData, List<KavaiForOneFunction> mimimiList) {

this.transitionData = transitionData;

this.mimimiList = mimimiList;

}

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();

}

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

KavaiForOneFunction current = mimimiList.get(i);

out.print(current.getName()+"\t");

out.println("before mimimi: "+current.getNotMinFunction());

out.println("after mimimi: "+current.getMinFunction()+"\n");

out.println("inputs before mimimi = "+current.getNumberOfInputsBeforeMimimi());

out.println("inputs after mimimi = "+current.getNumberOfInputsAfterMimimi());

out.println("outputs before mimimi = "+current.getNumberOfOutputsBeforeMimimi());

out.println("outputs after mimimi = "+current.getNumberOfOutputsAfterMimimi());

out.println("---------------");

}

}

out.close();

} catch (IOException e) {

e.printStackTrace();

return false;

}

return true;

}

}

**Висновок**

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

Розроблено процедуру збереження мінімізованих функцій в текстовий файл.