

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

ЛАБОРАТОРНА РОБОТА №6

З КУРСУ

«ТЕХНОЛОГІЯ ПРОЕКТУВАННЯ КОМП’ЮТЕРНИХ СИСТЕМ»

Виконав:

студент IV курсу

групи ІО-21

Журо Георгій Олександрович

Київ – 2015

**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.

**Опис програми:**

Для побудови аналітичної форми функцій переходів та функцій збудження тригерів необхідно натиснути кнопку «Build Functions» (Рис. 1). Результат для даної таблиці (Рис. 1) зображено на рисунку 2.

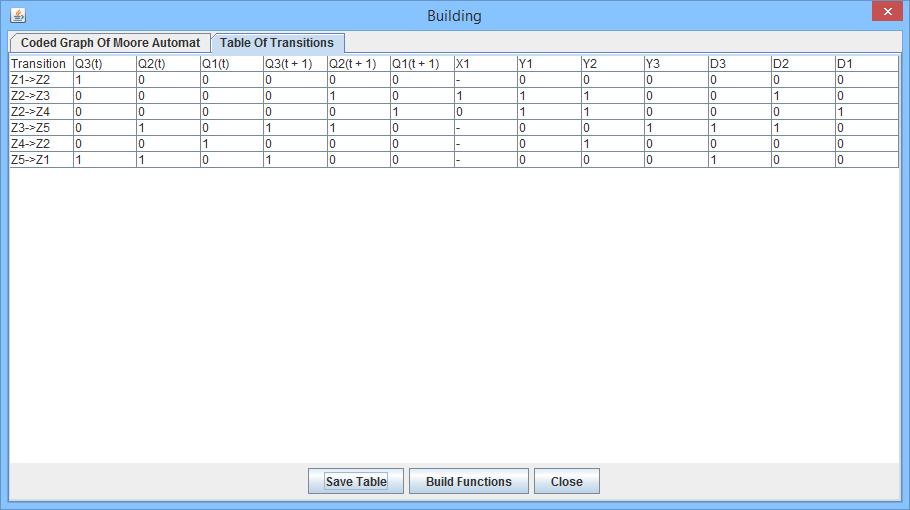


Рисунок 1

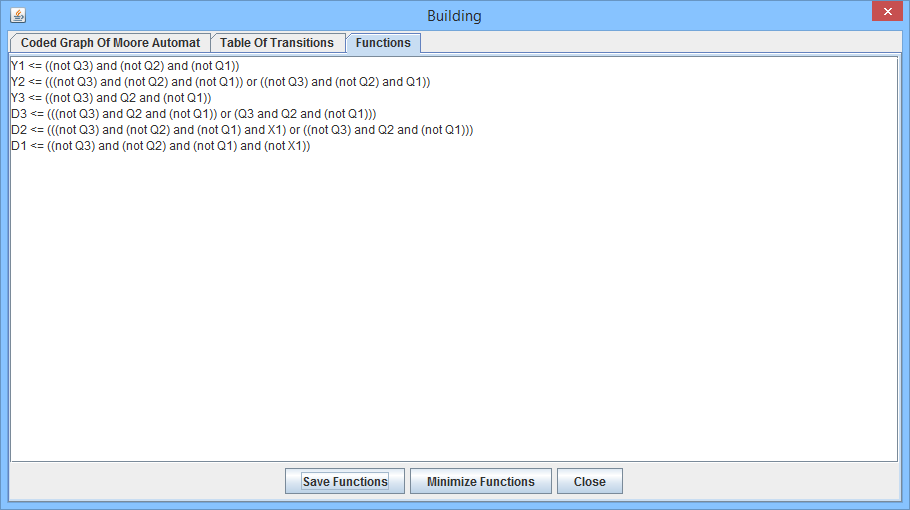


Рисунок 2

Для збереження аналітичного представлення не мінімізованих функцій необхідно натиснути кнопку «Save Functions» та в діалоговому вікні вибрати необхідний файл. Для мінімізації функцій необхідно натиснути кнопку «Minimize Functions». У результаті в новій вкладці будуть відображені мінімізовані функції (Рис. 3). Процедура мінімізації методом Квайна-МакКласкі реалізована в статичному методі minimizeFunctions() класу FunctionsWorker.

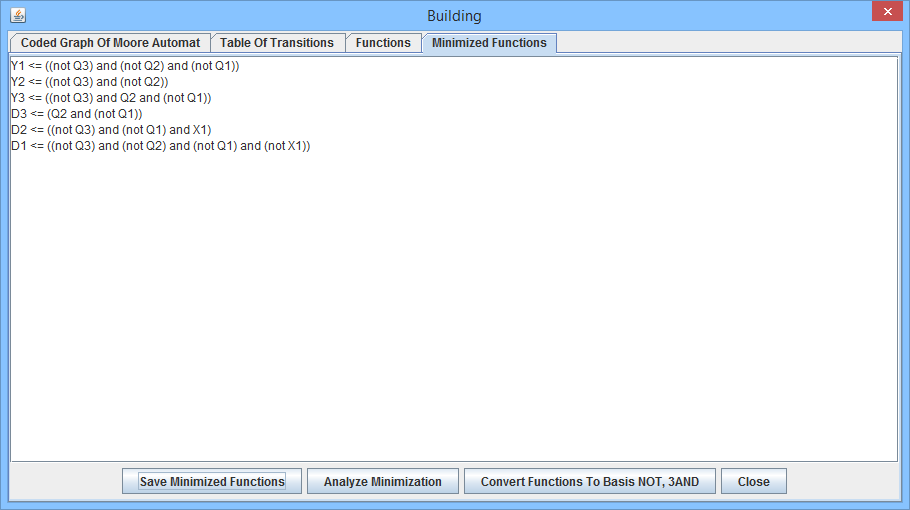


Рисунок 3

Для збереження мінімізованих функцій необхідно натиснути кнопку «Save Functions» та в діалоговому вікні вибрати необхідний файл. Результат показаний на рисунку 4.

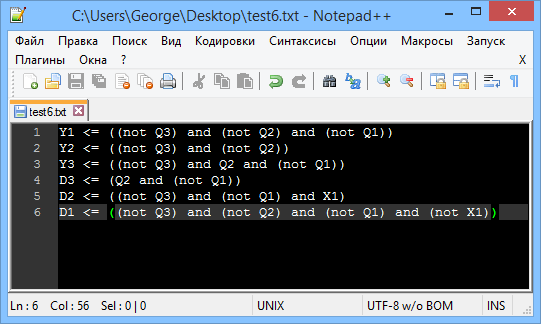


Рисунок 4

Для перегляду параметрів ефективності мінімізації функцій необхідно натиснути кнопку «Analyze Minimization» (Рис. 5).

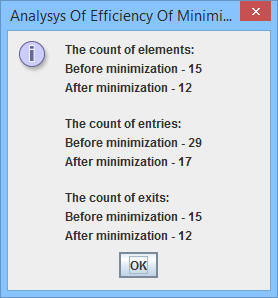


Рисунок 5

**Лістинг програми:**

package automat.functions;

import java.util.ArrayList;

public class Function {

private String name;

private ArrayList<Implicant> implicants;

private int boolFunction;

public Function(String name, ArrayList<Implicant> implicants, int boolFunction) {

this.name = name;

this.implicants = implicants;

this.boolFunction = boolFunction;

}

public String getName() {

return name;

}

public ArrayList<Implicant> getImplicants() {

return implicants;

}

public int getBoolFunction() {

return boolFunction;

}

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(name);

builder.append(" = ");

String boolFunctionString;

if (boolFunction <= 2) {

boolFunctionString = BoolFunction.getBoolFunctionString(boolFunction);

} else {

builder.append(BoolFunction.getBoolFunctionString(4));

boolFunctionString = BoolFunction.getBoolFunctionString(boolFunction - 2);

}

if (implicants.size() > 1) {

builder.append("(");

}

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

builder.append(implicants.get(i).toString());

builder.append(" ");

builder.append(boolFunctionString);

builder.append(" ");

}

builder.append(implicants.get(implicants.size() - 1).toString());

if (implicants.size() > 1) {

builder.append(")");

}

return builder.toString();

}

}

package automat.functions;

import java.util.ArrayList;

class Implicant implements Cloneable {

private ArrayList<String> names;

private ArrayList<Boolean> values;

private int boolFunction;

public Implicant(ArrayList<String> names, ArrayList<Boolean> values, int boolFunction) {

this.names = names;

this.values = values;

this.boolFunction = boolFunction;

}

public ArrayList<String> getNames() {

return names;

}

public ArrayList<Boolean> getValues() {

return values;

}

public int getBoolFunction() {

return boolFunction;

}

public void setNames(ArrayList<String> names) {

this.names = names;

}

public void setValues(ArrayList<Boolean> values) {

this.values = values;

}

public String toString() {

StringBuilder builder = new StringBuilder();

String boolFunctionString;

if (boolFunction <= 2) {

boolFunctionString = BoolFunction.getBoolFunctionString(boolFunction);

} else {

builder.append(BoolFunction.getBoolFunctionString(4));

boolFunctionString = BoolFunction.getBoolFunctionString(boolFunction - 2);

}

builder.append("(");

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

if (!values.get(i)) {

builder.append(BoolFunction.getBoolFunctionString(4));

}

builder.append(names.get(i));

builder.append(" ");

builder.append(boolFunctionString);

builder.append(" ");

}

if (!values.get(names.size() - 1)) {

builder.append(BoolFunction.getBoolFunctionString(4));

}

builder.append(names.get(names.size() - 1));

builder.append(")");

return builder.toString();

}

public Implicant clone() {

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

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

cloneNames.add(new String(names.get(i)));

}

ArrayList<Boolean> cloneValues = new ArrayList<Boolean>();

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

if (values.get(i) != null) {

cloneValues.add(new Boolean(values.get(i)));

}

else {

cloneValues.add(null);

}

}

return new Implicant(cloneNames, cloneValues, boolFunction);

}

}

package face;

import automat.functions.Function;

import automat.functions.FunctionsWorker;

import automat.moore.\*;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.ArrayList;

class BuildFrame extends JDialog {

private MainFrame mainFrame;

private JTabbedPane tabbedPane;

private GraphPanel graphPanel;

private CodedGraphPanel codedGraphPanel;

private JButton codeGraphButton;

private AutomatTableModel tableModel;

private JButton buildTableButton;

private String functionsString;

private ArrayList<Function> functions;

private JButton buildFunctionsButton;

private ArrayList<Function> minimizedFunctions;

private String minimizedFunctionsString;

private JButton minimizeFunctionsButton;

public BuildFrame(MainFrame frame, Rectangle bounds, MooreAutomat automat) {

super(frame);

mainFrame = frame;

setBounds(bounds);

setMinimumSize(bounds.getSize());

setResizable(true);

setModal(true);

setTitle("Building");

tabbedPane = new JTabbedPane();

add(tabbedPane);

JPanel mooreGraphPanel = new JPanel();

mooreGraphPanel.setLayout(new BorderLayout());

graphPanel = new GraphPanel(new GraphModel(automat));

JPanel mooreGraphButtonsPanel = new JPanel();

JButton saveGraphButton = new JButton(new SaveGraphAction(this));

saveGraphButton.setText("Save Graph");

codeGraphButton = new JButton(new CodeGraphAction(this));

codeGraphButton.setText("Code Graph");

JButton closeButton = new JButton(new AbstractAction() {

public void actionPerformed(ActionEvent e) {

setVisible(false);

}

});

closeButton.setText("Close");

mooreGraphButtonsPanel.add(saveGraphButton);

mooreGraphButtonsPanel.add(codeGraphButton);

mooreGraphButtonsPanel.add(closeButton);

mooreGraphPanel.add(mooreGraphButtonsPanel, BorderLayout.SOUTH);

mooreGraphPanel.add(graphPanel);

tabbedPane.addTab("Graph Of Moore Automat", mooreGraphPanel);

}

public BuildFrame(MainFrame frame, Rectangle bounds, CodedMooreAutomat automat) {

super(frame);

mainFrame = frame;

setBounds(bounds);

setMinimumSize(bounds.getSize());

setResizable(true);

setModal(true);

setTitle("Building");

tabbedPane = new JTabbedPane();

add(tabbedPane);

JPanel mooreCodedGraphPanel = new JPanel();

mooreCodedGraphPanel.setLayout(new BorderLayout());

codedGraphPanel = new CodedGraphPanel(new GraphModel(automat));

JPanel mooreGraphButtonsPanel = new JPanel();

JButton saveCodedGraphButton = new JButton(new SaveCodedGraphAction(this));

saveCodedGraphButton.setText("Save Graph");

buildTableButton = new JButton(new BuildTableAction(this));

buildTableButton.setText("Build Table Of Transitions");

JButton closeButton = new JButton(new AbstractAction() {

public void actionPerformed(ActionEvent e) {

setVisible(false);

}

});

closeButton.setText("Close");

mooreGraphButtonsPanel.add(saveCodedGraphButton);

mooreGraphButtonsPanel.add(buildTableButton);

mooreGraphButtonsPanel.add(closeButton);

mooreCodedGraphPanel.add(mooreGraphButtonsPanel, BorderLayout.SOUTH);

mooreCodedGraphPanel.add(codedGraphPanel);

tabbedPane.addTab("Coded Graph Of Moore Automat", mooreCodedGraphPanel);

}

private class CodeGraphAction extends AbstractAction {

private BuildFrame frame;

public CodeGraphAction(BuildFrame frame) {

this.frame = frame;

}

public void actionPerformed(ActionEvent e) {

JPanel mooreCodedGraphPanel = new JPanel();

mooreCodedGraphPanel.setLayout(new BorderLayout());

codedGraphPanel = new CodedGraphPanel(new GraphModel(graphPanel.getModel().getAutomat()));

JPanel mooreGraphButtonsPanel = new JPanel();

JButton saveCodedGraphButton = new JButton(new SaveCodedGraphAction(frame));

saveCodedGraphButton.setText("Save Graph");

buildTableButton = new JButton(new BuildTableAction(frame));

buildTableButton.setText("Build Table Of Transitions");

JButton closeButton = new JButton(new AbstractAction() {

public void actionPerformed(ActionEvent e) {

setVisible(false);

}

});

closeButton.setText("Close");

mooreGraphButtonsPanel.add(saveCodedGraphButton);

mooreGraphButtonsPanel.add(buildTableButton);

mooreGraphButtonsPanel.add(closeButton);

mooreCodedGraphPanel.add(mooreGraphButtonsPanel, BorderLayout.SOUTH);

mooreCodedGraphPanel.add(codedGraphPanel);

tabbedPane.addTab("Coded Graph Of Moore Automat", mooreCodedGraphPanel);

tabbedPane.setSelectedIndex(1);

codeGraphButton.setEnabled(false);

}

}

private class SaveTableAction extends AbstractAction {

private BuildFrame frame;

public SaveTableAction(BuildFrame frame) {

this.frame = frame;

}

public void actionPerformed(ActionEvent e) {

JFileChooser chooser = mainFrame.getChooser();

chooser.resetChoosableFileFilters();

chooser.addChoosableFileFilter(new TextFileFilter());

int result = chooser.showSaveDialog(frame);

if (result == JFileChooser.APPROVE\_OPTION) {

if (!chooser.getSelectedFile().getName().endsWith(TextFileFilter.TEXT\_FILE\_EXTENSION)) {

chooser.setSelectedFile(new File(chooser.getSelectedFile().getAbsolutePath() + TextFileFilter.TEXT\_FILE\_EXTENSION));

}

try {

tableModel.writeToFile(chooser.getSelectedFile());

} catch (IOException e1) {

JOptionPane.showMessageDialog(frame, "Error! Can't create file.",

"Error", JOptionPane.ERROR\_MESSAGE);

}

}

}

}

private class BuildTableAction extends AbstractAction {

private BuildFrame frame;

public BuildTableAction(BuildFrame frame) {

this.frame = frame;

}

public void actionPerformed(ActionEvent e) {

JPanel tablePanel = new JPanel();

tablePanel.setLayout(new BorderLayout());

tableModel = new AutomatTableModel((CodedMooreAutomat) codedGraphPanel.getModel().getAutomat());

JTable table = new JTable(tableModel);

JPanel tableButtonsPanel = new JPanel();

JButton saveTableButton = new JButton(new SaveTableAction(frame));

saveTableButton.setText("Save Table");

buildFunctionsButton = new JButton(new BuildFunctionsAction(frame));

buildFunctionsButton.setText("Build Functions");

JButton closeButton = new JButton(new AbstractAction() {

public void actionPerformed(ActionEvent e) {

setVisible(false);

}

});

closeButton.setText("Close");

tableButtonsPanel.add(saveTableButton);

tableButtonsPanel.add(buildFunctionsButton);

tableButtonsPanel.add(closeButton);

tablePanel.add(tableButtonsPanel, BorderLayout.SOUTH);

tablePanel.add(table);

tabbedPane.addTab("Table Of Transitions", tablePanel);

tabbedPane.setSelectedIndex(tabbedPane.getTabCount() - 1);

buildTableButton.setEnabled(false);

}

}

private class SaveFunctionsAction extends AbstractAction {

private BuildFrame frame;

public SaveFunctionsAction(BuildFrame frame) {

this.frame = frame;

}

public void actionPerformed(ActionEvent e) {

JFileChooser chooser = mainFrame.getChooser();

chooser.resetChoosableFileFilters();

chooser.addChoosableFileFilter(new TextFileFilter());

int result = chooser.showSaveDialog(frame);

if (result == JFileChooser.APPROVE\_OPTION) {

if (!chooser.getSelectedFile().getName().endsWith(TextFileFilter.TEXT\_FILE\_EXTENSION)) {

chooser.setSelectedFile(new File(chooser.getSelectedFile().getAbsolutePath() + TextFileFilter.TEXT\_FILE\_EXTENSION));

}

try {

PrintWriter output = new PrintWriter(new FileWriter(chooser.getSelectedFile()));

output.print(functionsString);

output.close();

} catch (IOException e1) {

JOptionPane.showMessageDialog(frame, "Error! Can't create file.",

"Error", JOptionPane.ERROR\_MESSAGE);

}

}

}

}

private class BuildFunctionsAction extends AbstractAction {

private BuildFrame frame;

public BuildFunctionsAction(BuildFrame frame) {

this.frame = frame;

}

public void actionPerformed(ActionEvent e) {

JPanel functionsPanel = new JPanel();

functionsPanel.setLayout(new BorderLayout());

JTextArea functionsArea = new JTextArea();

functionsArea.setEditable(false);

functions = FunctionsWorker.getFunctions(tableModel);

StringBuilder builder = new StringBuilder();

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

builder.append(functions.get(i).toString());

builder.append("\n");

}

functionsString = builder.toString();

functionsArea.setText(functionsString);

JPanel functionsButtonPanel = new JPanel();

JButton saveFunctionsButton = new JButton(new SaveFunctionsAction(frame));

saveFunctionsButton.setText("Save Functions");

minimizeFunctionsButton = new JButton(new MinimizeFunctionsAction(frame));

minimizeFunctionsButton.setText("Minimize Functions");

JButton closeButton = new JButton(new AbstractAction() {

public void actionPerformed(ActionEvent e) {

setVisible(false);

}

});

closeButton.setText("Close");

functionsButtonPanel.add(saveFunctionsButton);

functionsButtonPanel.add(minimizeFunctionsButton);

functionsButtonPanel.add(closeButton);

functionsPanel.add(functionsButtonPanel, BorderLayout.SOUTH);

functionsPanel.add(new JScrollPane(functionsArea));

tabbedPane.addTab("Functions", functionsPanel);

tabbedPane.setSelectedIndex(tabbedPane.getTabCount() - 1);

buildFunctionsButton.setEnabled(false);

}

}

}

**Висновки:** У результаті виконання даної лабораторної роботи я здобув навички автоматизації процедури мінімізації булевих функцій методом Квайна-МакКласкі. Мною були реалізовані процедури побудови функцій переходів та функцій збудження тригерів, мінімізації цих функцій, аналізу ефективності їх мінімізації по різним параметрам, а також була реалізована можливість збереження результатів у текстовому файлі. Всі процедури були реалізовані на мові програмування Java.