System Requirements

Curtis Helle , Gezim Hoxha, Jacob Pledger, Jeremy Zaretski

0. Summary

K-maps are used regularly to minimize Boolean functions to ensure that the least amount of gates to create a circuit is used. The goal of the KSimple project is to allow a user that is familiar with K-maps to use the program to minimize Boolean function. This document is intended to convey the system requirements for KSimple, the K-Map solver. It includes behavioral models, a UML diagram showing classes and their inheritance structure, detailed class descriptions and behavior and finally a mockup of the program.

Table of Contents

1. Behavioral Models

1.1 Data flow model

THIS IS A PLACEHOLDER

1.2 State machine model

THIS IS A PLACEHOLDER

2. Class Object and Inheritance Model

THIS IS A PLACEHOLDER

3. Class Descriptions

3.1 PrefPane

Class:	PrefPane
Description:	A class that manages requesting variable names from the user to a maximum of 6. Gets the number of variables entered to determine size of the truth table and K-Map.
Inputs:	None
Source:	None
Outputs:	None.
Destination:	None.
Action:	Stores user chosen preferences: variable names and number of variables, he/she wishes to use.
Pre-condition:	None.
Post-condition:	None.
Side Effects:	None.

Method:	getVariableNames
Description:	Returns the variable names the user has chosen to use.
Inputs:	None.
Source:	Calling object.
Outputs:	Variable names; QVector<QString>.
Destination:	None.
Action:	Returns private data member variables.
Pre-condition:	Variables names have been set.
Post-condition:	No change to internals.
Side Effects:	None.

Method:	getNumberOfVariables
Description:	Returns the number of variables the user has chosen to use.
Inputs:	None
Source:	Calling object.
Outputs:	Number of variables; int.
Destination:	None.
Action:	Returns private data member numberOfVariables.
Pre-condition:	Valid size has been entered.
Post-condition:	No change to internals.
Side Effects:	None.

Method:	setVariableNames
Description:	Sets the variable names the user has chosen to use.
Inputs:	QVector<QString>
Source:	setPreferences method.
Outputs:	None.
Destination:	None.
Action:	Sets private data member variables.
Pre-condition:	Valid variable names have been entered.
Post-condition:	Variable names are stored in data structure.
Side Effects:	None.

Method:	setNumberOfVariables
Description:	Sets the number of variables the user has chosen to use.
Inputs:	Int
Source:	setPreferences method.
Outputs:	None.
Destination:	None.
Action:	Sets private data member numberOfVariables.
Pre-condition:	Valid number of variables is entered.
Post-condition:	Number of variable is stored in data structure.
Side Effects:	None.

Method:	setPreferences
Description:	Validates the preferences and sets the preferences.
Inputs:	None.
Source:	PrefPane sends a signal when user presses OK.
Outputs:	None.
Destination:	None.
Action:	Updates object based on preferences.
Pre-condition:	Valid preferences are entered.
Post-condition:	Effected objects are updated.
Side Effects:	None.

3.2 KMapWidget

Class:	KMapWidget
Description:	A data structure representing a K-map. Contains a vector data member for storage, and buttons for drawing.
Inputs:	None.
Source:	Set Method, or translated from a Truth Table or Boolean expression
Outputs:	None.
Destination:	None.
Action:	None
Pre-condition:	None.
Post-condition:	None.
Side Effects:	None.

Method:	set
Description:	Sets a space on the K-map to 1, 0, or X. A single mouse click on a box cycles the value from 0 to 1 to X and repeats.
Inputs:	1,0,X.
Source:	User via mouse.
Outputs:	1,0,X.
Destination:	1 location in the Karnaugh map vector.
Action:	Sets a single location in the Karnaugh map to 0, 1, or X. Each click cycles the character from 0 to 1, 1 to X, and X to 0, and repeats.
Pre-condition:	None.
Post-condition:	A single space in the K-map is filled in with either a 1,0, or X.
Side Effects:	None.

Method:	display
Description:	Renders the K-map on its container.
Inputs:	None.
Source:	Calling objects.
Outputs:	None.
Destination:	None.
Action:	Outputs Karnaugh map information to the screen.
Pre-condition:	Preferences have been set.
Post-condition:	K-Map is drawn onto the screen.
Side Effects:	None.

Method:	toTruthTable
Description:	Converts the Karnaugh Map to a Truth Table.
Inputs:	None.
Source:	Calling object.
Outputs:	A vector of characters (0,1,X) in the correctly formatted order required by the Truth Table class.
Destination:	None.
Action:	Converts the Karnaugh Map into a vector of characters for a Truth Table and returns it.
Pre-condition:	That K-Map has valid data in it.
Post-condition:	Truth table is returned in the appropriate format.
Side Effects:	None.

Method:	solve
Description:	Creates K-Map groupings.
Inputs:	None.
Source:	Calling object.
Outputs:	None.
Destination:	None.
Action:	Calculates K-map groupings.
Pre-condition:	A K-Map has been allocated and the user has entered input to one of the three classes: Boolean Expression, Truth Table, K-Map.
Post-condition:	The groupings are created.
Side Effects:	None.

3.3 TruthTableWidget

Class:	TruthTableWidget
Description:	A class representing a truth table. Contains a vector of characters to store the inputs and outputs.
Inputs:	None.
Source:	None.
Outputs:	None.
Destination:	None.
Action:	None.
Pre-condition:	None.
Post-condition:	None.
Side Effects:	None.

Method:	set
Description:	Sets a space on the Truth Table to 1, 0, or X. A single mouse click on a box cycles the value from 0 to 1 to X and repeats.
Inputs:	1,0,X
Source:	User via mouse.
Outputs:	1,0,X
Destination:	1 location in the character vector data member.
Action:	Sets a single location in the Truth Table to 0, 1, or X. Each click cycles the character from 0 to 1, 1 to X, and X to 0, and repeats.
Pre-condition:	None.
Post-condition:	A single space in the Truth Table is filled in with either a 1,0, or X
Side Effects:	None

Method:	display
Description:	Renders the truth table to container.
Inputs:	None.
Source:	None.
Outputs:	None.
Destination:	None.
Action:	Draws the truth table onto the screen with appropriate rows and columns.
Pre-condition:	Truth table members have valid data.
Post-condition:	The truth table and its data are displayed on the screen.
Side Effects:	None.

Method:	toKMap

Description:	Converts the truth table to K-Map.
Inputs:	None.
Source:	Calling object.
Outputs:	A vector of characters (0,1,X) in the correctly formatted order required by the KMapWidget class.
Destination:	None.
Action:	Converts the Truth Table into a vector of characters for a K-Map.
Pre-condition:	The truth table has valid data.
Post-condition:	A Karnaugh Map is created and returned with the correct interpretation of the Truth Table.
Side Effects:	None.

Method:	toBooleanExpression
Description:	None.
Inputs:	None.
Source:	Calling object.
Outputs:	A string formatted as a Boolean expression (using alphabetic characters and ‘,+,*,^,()).
Destination:	None.
Action:	Converts the Truth Table into a string for a Boolean expression.
Pre-condition:	The truth table has valid data.
Post-condition:	A Boolean expression is created and returned with the correct interpretation of the Truth Table.
Side Effects:	None.

Method:	solve
Description:	Calls the solve methods for Boolean Expression and Karnaugh Map.
Inputs:	None.
Source:	None.
Outputs:	Solved K-Map and minimized Boolean expression.
Destination:	Boolean expression string data member and the screen/monitor.
Action:	Calls solve methods from the Boolean expression class and Karnaugh Map class.
Pre-condition:	There are allocated variables for a Boolean function and a Karnaugh Map, and the system has received user input into one of: Boolean Expression, Karnaugh Map, or Truth Table. A display must also be connected.
Post-condition:	The Boolean expression is minimized, and the K-Map groupings are output to the display.
Side Effects:	String data member in the Boolean Expression and vector data member in K-Map are changed.

3.4 EquationWidget

Class:	EquationWidget
Description:	A wraparound data structure for a string that contains the user-input Boolean expression. Contains a string data member for storage.
Inputs:	None.
Source:	None.
Outputs:	None.
Destination:	None.
Action:	None.
Pre-condition:	None.
Post-condition:	None.
Side Effects:	None.

Method:	getExpression
Description:	Returns the expression from the user in the required format.
Inputs:	None.
Source:	None.
Outputs:	A string in the required format.
Destination:	None.
Action:	Gets a Boolean expression from the user in the required format.
Pre-condition:	Valid expression has been entered.
Post-condition:	Returns the expression with no change to internal data.
Side Effects:	None.

Method:	toTruthTable
Description:	Converts the Boolean expression to a Truth Table.
Inputs:	None.
Source:	None.
Outputs:	A vector of characters (0,1,X) in the correctly formatted order required by the Truth Table class.
Destination:	None.
Action:	Converts the Boolean expression into a vector of characters for a Truth Table.
Pre-condition:	Valid expression exists.
Post-condition:	A Truth Table is created and returned with the correct interpretation of the Boolean expression.
Side Effects:	None.

Method:	display
Description:	Renders the Boolean expression to the screen.
Inputs:	None.
Source:	None.
Outputs:	None.
Destination:	None.
Action:	Draws expression on the screen in the appropriate container.
Pre-condition:	Expression must be valid.
Post-condition:	The Boolean expression is printed on the screen.
Side Effects:	None.

Method:	solve
Description:	Minimizes the Boolean expression.
Inputs:	Truth Table vector data member.
Source:	None.
Outputs:	Minimized Boolean expression.
Destination:	None.
Action:	Solves the Boolean expression using the Quine McCluskey algorithm, updates it in memory, and outputs it to the screen.
Pre-condition:	A Truth Table has been allocated and contains data based on user input and a display is connected.
Post-condition:	The Boolean expression is minimized, stored in memory, and output to the screen.
Side Effects:	The string data member in Boolean expression is changed.

3.5 QMManagement

Class:	QMManagement
Description:	Management class for QM algorithm and data control.
Inputs:	None.
Source:	None.
Outputs:	None.
Destination:	None.
Action:	None.
Pre-condition:	None.
Post-condition:	None.
Side Effects:	None.

Method:	minimize
Description:	The QM Algorithm.
Inputs:	Truth Table vector<strings> data member.
Source:	None.
Outputs:	Minimized Boolean expression data.
Destination:	None.
Action:	Solves the Boolean expression using the Quine McCluskey algorithm, updates it in memory, and outputs it to the screen.
Pre-condition:	A Truth Table has been allocated and contains data based on user input and a display is connected.
Post-condition:	The Boolean expression is minimized, stored in memory, and output to the screen.
Side Effects:	The string data member in Boolean expression is changed.

See attached document for doxygen-style class descriptions.

4. Coding Conventions

4.1Doxygen Comments

Doxygen comments are mandatory. Documentation can be found here:
http://www.doxygen.nl/manual.html

4.2Coding Conventions

All code shall be styled as follows:

-All class names shall start with a capital letter. If the class name contains multiple words, each word will begin with a capital. For example: class TruthTable

-The first letter of all function names shall begin with a lowercase letter. However, the beginning of any additional words in the function name shall be capitalized. For example: void toTruthTable(args)

-Class member variables shall always begin with an underscore followed by a lowercase, with the first letter of any additional words capitalized. For example: string _booleanExpression;

-Temporary or non-class member variables simply begin each word with a lowercase letter, with words separated by underscores. For example: int temp_var;

-For QT, signals and slots are named the same as functions, but prefixed with “sig_” and “slot_” respectively.

-Class definitions shall always be in the order of: public, protected, private

-Opening brackets shall come directly after the command requiring them, closing brackets must be lined up with the first letter of that command. For example:

```
while(true){ //incidentally, this is how we will be writing infinite loops

//some commands

}
```

-Indentation should be 3 spaces.

-For any additional required reference to coding conventions go to:
http://www.yolinux.com/TUTORIALS/LinuxTutorialC++CodingStyle.html

5. Mockup

This mockup illustrates how the system interface will look.

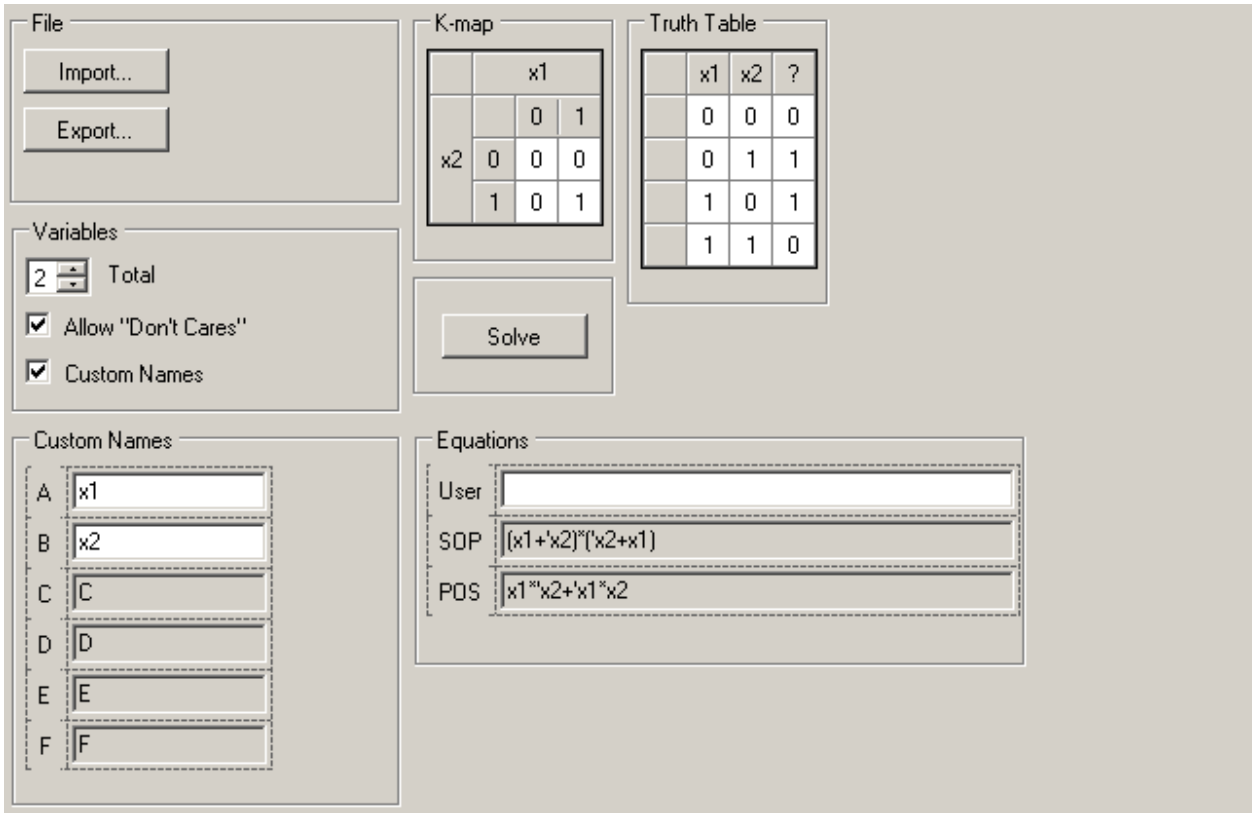

Figure 1 - Initial mockup of the system.

5. References

The following link provides documentation to all releases of QT. We will be using QT 4.5 to develop the system. The labs at the school currently run QT 4.2.

QT Documentation: http://doc.trolltech.com/