LogicSim Labs

Logic Gate Simulator

User’s Manual

Version 1.0

Revision History

Date	Version	Description	Author
05/May/2024	1.0	The User’s Manual describes the instructions for installation and use of the Logic Gate Simulator project.	Daniel Bobadilla, Fatima Avila, Lily Gray, David Sutherland

Table of Contents

Test Case

1. Purpose

The purpose of the document is to provide readers with a manual on how to use the program and to alert the user to any additional features the program can do. The User Manual also guides the reader on how to install the program.

2. Introduction

The Boolean Logic Simulator by LogicSim Labs evaluates Boolean logical expressions. This tool is meant to help speed up Boolean logic expression evaluation and can handle various logic operators and handles variations of inputs so long as they are within reasonable bounds.

Some key features include the ability to handle infix notation, operator precedence, and excess parentheses. It also includes robust error handling for invalid operators.

How to Install:
Assuming the user has found the project code from the GitHub project repository the steps are as follows.

1. Click the "Code" button on the repository which provides a link. Copy that link.
2. From the terminal Navigate to the folder that the user wants the project to be copied into.
3. Then, in the terminal type and enter "git clone [URL that user copied]"
4. From there, the project should be cloned into that folder and then the user should be able to navigate it.
5. Once the user is in the project folder, assuming they are using a GNU compiler, they must type and enter "g++ -o main main.cpp"
6. Finally, run the project from the terminal using "./main"

3. Getting started

To use the code for the Logic Gate Simulator project to evaluate logical expressions, first run the "make" command on your computer's terminal. This should prompt the user to enter q or quit to exit the program, otherwise type a valid expression with your keyboard and press the "enter" button. A valid input consists of at least one operand (T or F), or two with one operator between them (&, |, $, @). An exception is "!", which is directly associated to the operand to its right and does not need to be next to another operator. A properly structured argument will return the evaluation of the user's input.

How the user enters an input should depend on the simulator's order of operations. The logical not, "!", takes precedence when no parentheses are used. Otherwise, logical operators and their operands will be computed from left to right. A valid operator (&, @, |, $) should be used in between a T or an F (or t or f) except for "!", which should be placed in front of a T or an F or an expression in parentheses to negate the evaluation (T becomes F and F becomes T).

The user should use "$" when they want the result to be True when exactly one operand is true. They should use "@" when they want the evaluation of two operands to be F when one or both operands are false. "!" should be used when the user wants the input to the right of the "!" to be negated. "&" should be used when the user wants the program to return true only if both operands are true. Lastly "|" should be used when the user wants to return true when both or one operand(s) is true.

The program's output is the equivalent and simplified result of the Boolean algebraic expressions typed by the user. If the output is “True”, the expression overall evaluates to “True”, vice versa if the output is “False”. If an error is detected, an error due to an unrecognized character will be output to the screen. If the error is for a different reason, there is currently no functionality to help the user determine where the error comes from.
[Lily]

4. Advanced features

The program has the feature to quit whenever the user wants by typing q or Q and is case insensitive in terms of operands if they are characters “T” or “F”. The program is also capable of reusing the previous input by pressing the “up” arrow on the keyboard.

5. Troubleshooting

If the user runs into frequent errors, check to see if the operators being used are the right ones, i.e. ! (NOT), &(AND), $ (XOR), @ (NAND), and | (OR). Other reasons that the expected and actual results may conflict are misplaced parentheses and mismatched operators. Every operator must act on two operands (except for NOT). The user should check that any inputs agree with the valid input guidelines described in section 3.

6. Examples

Here are some examples of valid inputs:

1. T
2. T & T
3. T | F
4. ! T
5. F $ T
6. F @ F
7. (T & T) | (F & F)
8. ((T & F) | F) @ T

[David]

7. Glossary of terms

Boolean – a logical 0 or 1. 0 symbolizes a “False”, and 1 symbolizes a “True” result or input.
Operand – the variable that an operator is acting on.
Operator – a function that does an action to two or more operands.

8. FAQ

This section should answer frequently asked questions about the software.

What operators can the program accept?

The program can accept AND (&), OR (|), NOT (!), NAND (@), XOR ($).

Can the program accept brackets and curly brackets?

No, only normal parenthesis, you wouldn't have any need to use brackets or curly brackets.

What is the limit of how large the question can be?

It depends on your device, but any feasible question should be able to be handled.

Where can I download the program?

https://github.com/liliangray/EECS348_GroupProject