University of Puerto Rico - Mayagüez Campus

Department of Electrical and Computer Engineering

Logically

Final Report

Gustavo A. Hernández Ortiz

Yorki G. Serrano Natal

Javier Ramirez Zayas

Jean O. Candelaria Pagán

ICOM-4036-060

Contents

Introduction

Boolean algebra is a branch from algebra in which variables can have a value of 1 or 0 and the and the expressions are composed of different logical operators such as and, or, and & not. These logical expressions can quickly become complex. The process of designing a logic circuit usually involves creating a truth table, drawing a logic circuit with the proper gates and trying to simplify an expression to obtain a simpler circuit. In other cases, logic expressions are used to represent the relations of different data sets and this is represented using venn diagrams.

Logically is a programming language that looks to simplify the process of designing logic circuits and to provide all the necessary tools used when dealing with logical expressions. Instead of spending a lot of time writing tables and karnaugh maps, users can obtain the same results by typing some simple commands. It's simple and intuitive syntax allows users to quickly learn how to write a program and start creating and analyzing logical functions.

Reference Manual

Creating Expressions

The syntax for creating logical expressions is very straightforward. Expressions are written by defining individual logical gates:

name = OP name name… - The name in the left side of the assignment is the variable name for the desired expression, it can be an input of a previously created logic gate but it cannot be a name given to a previously created one. The names on the left side of the assignment are the inputs of the gate. These can be existing expression, it is a list of names separated by spaces. The NOT operator can only receive one input at a time. OP is the name of the logic gate that will be used for the expression. Available gates are:

- *AND*
- *NAND*
- *OR*
- *NOR*
- *XOR*
- *XNOR*
- *NOT*

Expression Functions

Functions are operations you can do with a declared logical expressions. The functions available in Logically are:

DISPLAY name **-** displays the resulting logical expression with the given name.

DEL name **-** deletes the specified expression and all its sub expressions. If it's the input of another expression, it will be replaced with a simple input variable.

TABLE name **-** generates the truth table for the specified logical expression.

CKT name **-** draws the circuit diagram for a given expression. Supports logic circuits with 2 or 3 input variables.

VENN name **-** draws the venn diagram of a given logical expression. Can draw diagrams with 2 or 3 variables.

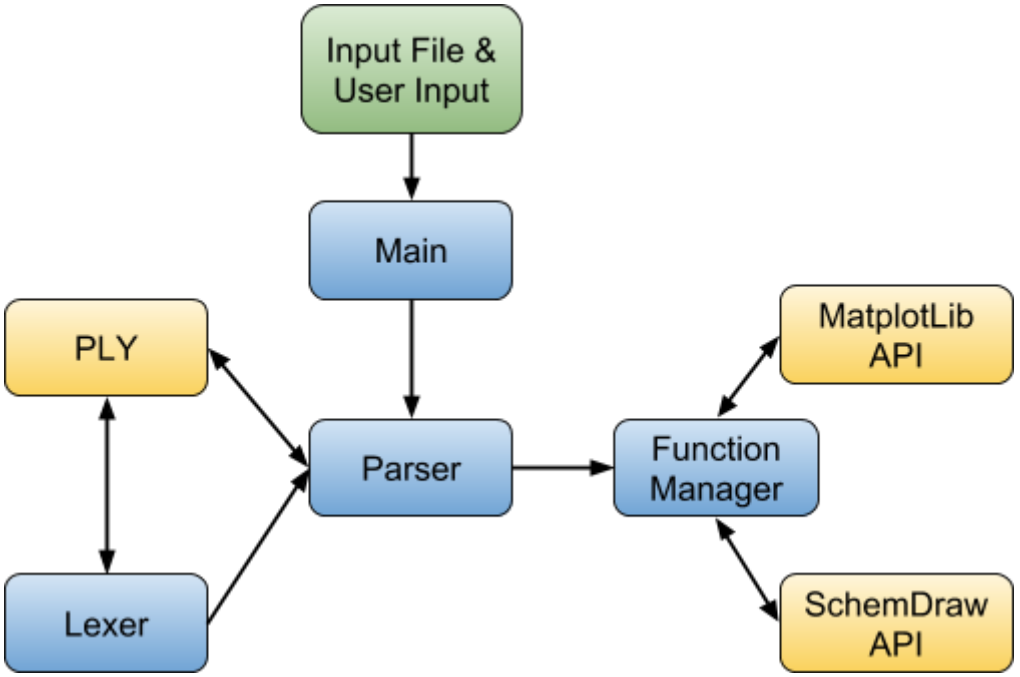
SIMPLIFY name **-** simplifies the specified expression to an equivalent using its minterms.

Other Functions

HELP **-** displays available functions and how to define expressions.

EXIT **-** closes Logically.

Translator Architecture



Module Interfaces

The main module of the program, also called logically,py, is the module that takes care of receiving the user input or the input file depending on how the program is executed. This module passes its data to the parser which then works together with the lexer and PLY to analyze then recognize the tokens in the input and any formed syntax expressions. The lexer's sole purpose to identify these tokens while the parser not only looks for formed expressions but organizes the logical expressions given by the user and builds a tree to organize them. The parser is the only module with access to the function tree and takes care of modifying it.

The function manager module receives a function and an expression subtree from the parser to either create a truth table, draw a venn diagram, draw a circuit or simplifying a logical expression. Depending on the type of function specified by the user this module will take care of the operation and in use the corresponding API. In the case of venn diagrams it will use the Matplotlib API and when drawing circuits it will use the SchemDraw API.

Language Development

Development Environment

During the development of Logically, the following tools were used:

- Python 3.4 - the language used for building Logically.
- PyCharm - the main IDE used for development of Logically.
- Github - used VCS and helpful for keeping track of changes that were done to the source code.

Testing Methodology

To test the different parts of the programming language, bottom-up testing was done. The lower modules were tested individually and then tested once again when incorporated with the higher level modules.

In the case of the lexer, this was tested by sending the lexer different tokens and invalid characters to observe if it was labeling them correctly. The parser was tested by forming different expressions that were and weren't rules of the language and observing if they were accepted as correct input. There were also cases where the input might be a valid expression but it caused a logical error such as deleting an expression that doesn't exist or trying to draw the circuit for a variable that is an input signal. The main module was tested by giving a file as a parameter that did not have a ".lly" extension.

The function manager module was tested by testing every function individually and passing different logical expressions as parameters to each of them. After they worked correctly we connected them to the parser and tested all the modules excluding the main module. After this process was done, every module was connected and tested one last time.

Test Programs

The following programs were used to test Logically:

Program #1 - Used for testing expression building, table generation and circuit drawing.

```
out = OR a b
a = AND c d
c = NOT e
b = AND f e
f = NOT d
TABLE out
CKT out
```

Program #2 - Used for testing the simplifier.

```
out = AND x y z
x = AND b c
y = AND c d
z = OR c d
SIMPLIFY out
```

Program #3 - Used for testing venn diagrams.

```
exp = OR a x
x = AND a b
VENN exp
```

Program #4 - Used for testing expression modifying and lexer

```
var = NAND y t i
exp = OR var w
DISPLAY exp
DEL exp
```

Program #5 - Used for testing parser and logical errors.

```
test = OR a b c d e f g h
DISPLAY test
DISPLAY exp
DEL
DELETE
out = AND
```

Conclusion

Throughout the process of creating Logically we discovered that the development of a programming language is not particularly easy but it is not as complicated as one would think. It was very important for us to have a clear idea of what we wanted the finished product to be in order to work efficiently. There were various obstacles during development mostly because of the different libraries and tools that had to be integrated. Developing the syntax and implementing it using PLY was one of the tasks with the least of issues. Some of the problems were compatibility issues with the version of Python and this was solved rather quickly. The other tools did not have issues with compatibility but their implementation was very complicated because these tools required translating the users input into many lines of code and partially automating the use of the libraries. Throughout the development some changes were made to the API's used as well as the features included in the language. While creating logically we were able to work well as a group, we managed to become more proficient using Python and we learned how the development process of a programming language unfolds. The finished result was much like what we initially had planned. Not only did we implement the initial features we had planned but we were able to include an extra one, which was expression simplifying. We think that the end result of Logically can be a useful tool for students who are studying logic circuits since it simplifies the process of creating and testing logical functions as well as drawing the necessary logic circuits.