Cython Testing Writeup

Jordan Berkompas, Jared Cline, Brady Davis,
Daniel Millson, Daniel Milstead, Austin Youngerman

Spring 2020

The first section of this writeup will include the general testing process, including preliminaries, running the program, and reading the code. The second section will go over what each member contributed individually to this process.

Testing Process

Preliminaries - Installation and Running:

Installation lacked some details. While there were instructions on what the Makefile does and how to run it, there were no directions on which executable to run (we found that it was ./create)

When trying to run this on machines outside of the lab (such as our virtual machine and personal computers), we noticed that the project relied on libraries which were not installed on those machines. While the compiled executables ran as expected, we couldn't run the Makefile, as it relied on libraries such as libcppunit.

The GUI can only be run on Unix machines in the lab, which indicates that, at this moment, this doesn't support cross-platform behavior. Like the comment above, the GUI relies on libraries which are installed on the lab, but doesn't indicate exactly which libraries, so we couldn't run it on our local machines. There are instructions on how to run it over VNC/X11, but we didn't try this. We noticed that the Makefile bundled in the GUI directory includes directions for installing in macOS and MSYS2, but we believe this Makefile should be joined with the one in the parent directory. On trying to install on a macOS device, it didn't work.

Running The Program:

When running the program, we first tested the following commands:

- NOT true
- NOT false
- true AND true
- true AND false
- true OR true

- true OR false
- false OR false

and similar commands. These were all interpreted correctly, and gave the expected outputs of "0" for false and "1" for true. We tried to combine multiple binary operations in the following ways:

- true AND true AND false
- true AND false OR true
- true OR false AND true

These commands did not run as expected. While there is no grouping of the actual binary connectives, we expected that these would evaluate in some expected order (i.e. left to right). However, since AND and OR don't neccessarily commute, their program should probably return some message about grouping and associativity, though it did not. If evaluated from left to right, then "true AND true AND false" should return false, and "true AND false OR true" should return true. However, when the output was generated for these statements, each gate/subexpression was evaluated in isolation, without passing the result forward. So the output for "true AND true AND false" was "1 0", and the output for "true AND false OR true" was "0 1".

We noticed unusual results with the following commands:

- true AND NOT false
- false OR NOT false

We expect "true AND NOT false" to actually evaluate "true AND true", which would result in true or "1". However (and similar to the argument above of not passing results of subexpressions), the output was actually "0 1". We believe the "1" was from evaluating "NOT false", but we don't understand the "0" result. A similar result is seen when evaluating "false OR NOT false".

Parentheses do not behave as expected (or have any functionality at all beyond being interpreted as strings). For example, the following commands;

- false OR (
- (AND true
- true XOR (

are not considered invalid. "false OR (" evaluates to "0", "(AND true" evaluates to "0", and "true XOR (" evaluates to "1". This is, of course, unexpected. We believe that this is both from lack of error checking, and from interpreting anything which is not a gate or a true/false value as a string. Oddly enough, all string values evaluate to false. We tried the following:

- bob AND true
- steve OR bob

- dymacek XOR marmorstein

all of which evaluated to "0", indicating that these are all interpreted as false. Interestingly enough, when trying a non-true/false value with NOT, such as:

- NOT dymacek
- NOT bob
- NOT (

These are actually determined to be invalid inputs, and this causes the program to quit. We will revisit this when examining the code. Similarly we noticed that "True" and "False", with capital letters, are actually invalid as well. The program is case sensitive, where gates must be entirely capitalized and true/false values must be all lowercase.

There is more unexpected behavior. When just evaluating:

- true
- false

there is no output at all! Other commands such as:

- NOT false AND NOT false
- NOT false OR NOT true

gave us 3 outputs each, when it should be 1. We noticed that it interprets anything on either side of a binary connective which is not a true-false value as a string. In thise case, when it encounters "AND" and "OR", it interprets the "NOT" on the right as a string, not as a NOT gate.

The GUI is still under development, so we won't comment on that. It does run in the Computer Science lab, so we saw that it was still under development.

Code:

We first note the lack of comments throughout some of their important files (gates.h, gates.cpp). The comments which ARE there are not descriptive.

Regarding their gate functionality, the only error checking is done in the NOT gate, where it looks to see if the input is a 0 or 1. Otherwise, it throws in anvalid input error and quits. The other gates do no such error checking, and happily evaluate strings as false.

We believe that the next step for them (which they might be working on) is to not pass the original string value, but to evaluate subexpressions and pass those as inputs to the left/right sides of a statement. In this first section, we will discuss attempts to install and run the program.

Individual Conributions

Jordan Berkompas

I worked on testing cross-platform capabilities for the gui, found out where exactly they described what to download depending on your OS. Helped test different cases with the backend, revealed that they can't handle having NOT in any other gate combination

Jared Cline

I tested some of the possible inputs for ./create. I found that the operators are computed by looking at the tokens to either side of the operator rather than being parsed as a full statement. And that unrecognized strings are evaluated to false.

Brady Davis

I was responsible for downloading the necessary software and libraries to use the GUI on a MacOS. After downloading the specific libraries and compiling the code using the README, I found that the GUI did not appear as what was expected. Instead, I received a linking error saying that a library was missing even though I had installed everything successfully. Thus, proving that libraries which are being used are only Linux specific and can only be ran from the lab

Daniel Millson

I was responsible for coming up with some test cases to check the functionality of the code. As well, I read through the code to understand the reason for various unexpected behaviors in the program. Specifically, I looked to understand the parse structure of the different logic gates, and how parameters were being passed. I also looked at functionality on different platforms, including my system at home and the virtual machine.

Daniel Milstead

I was responsible for testing inputs with all the given gates. I helped find that the GUI appeared, however did not have much functionality. Found that strings that were given as input, evaluated as false, unless NOT was used before, in which case, the program would exit and displayed that an invalid input was given.

Austin Youngerman

Bad Setup

I downloaded and tested the dependencies for Ubuntu/Linux. The first problem I noticed was that they included their compiled binaries, so I removed those and attempt to recompile them from my own machine. Immediately, the makefile didn't function properly due to them using the "compile" wrapper inside of it. The makefile failed in all commands stated in the README.

Failed Compile

I could not compile or run the front end due to missing SDL dependencies/improper environment, and it is not functioning out of the box. There are maybe 1 or 2 people in the major that have ever used X11 or a VNC server.

Overall lack of documentation

Changelog.txt is entirely empty No comments in a multitude of files regarding critical sections of code, specifically the test cases.