Rensselaer Polytechnic Institute

User Guide for Decision Procedure Program

Validity and Tautology Decision Procedure

Brian Delaney
4-25-2023

Contents

Project Description

Link to GitHub Repository hosting code, this user guide, and an example input file.

The goal of this project is to create a decision procedure that can be used to determine if a propositional logic argument is valid or if a logic statement is a logical tautology. The program achieves this using resolution. In order for an argument to be valid, there must not exist the possibility that all the premises are true and the conclusion is false. To test this, resolution is applied to the premises with a negated conclusion. If the resolution finds that there are no possible truth values for the literals in the argument that produce true premises and a false conclusion, then it must be the case that the argument is valid. The same process is applied for tautologies. Tautologies are defined as statements that are always true, so a negated tautology should always be false. The same resolution procedure can be applied to the negation of the statement in question. If there are no possible truth values for the literals in the statement that produce a true negated statement, then the negated statement is always false. Therefore, the initial statement is always true and by definition a logical tautology.

This implementation of resolution follows a multistep process to reduce the statement(s) in question.

1. Modify each statement to be in Negation Normal Form.
2. Modify the NNF statements to be in Conjunctive Normal Form.
3. Each conjunct forms a new clause.
4. Perform resolution on the clauses.
    a. If clauses with only one literal in them exist, try to eliminate other literals.
    b. If clauses are considered tautological clauses, reduce them to eliminate the literals and their compliment that make the clause tautological.
    c. If ever an empty clause set exists, return TRUE.
    d. If no more can be done, return FALSE

As a special note, you can in fact test tautologies using the validity checker. Simply inputting a conclusion with no premises is a logical argument, and if the argument is valid, then the conclusion is a tautology as well. The argument could be made that then including a separate procedure for checking for tautologies is redundant. While this is true, this tool is not meant to be an efficient means of checking validity and logical tautology. There are certainly far better implementations of tools to check for that. This program is meant to be an educational tool where an interested party may focus on logical validity and tautologies separately and explore the relationship between the two to discover this fact on their own.

Valid Characters

The program will only function properly if valid symbols are used to represent propositional logic statements.

- "~" – Negation
- "$" – Conditional
- "%" – Biconditional
- "&" – And
- "|" – Or
- "(" – Open parenthesis to define a new scope
- ")" – Closed parenthesis to conclude a scope
- "A", "B", …
    - Literals must be single character capital letters

There should not exist any spacing between letters in a logical statement. See Figure 1 for an example of how logical statements should be formulated.

Start Up Procedure

The main.py file can be found on the GitHub repository.

1. Download the main.py file
2. Open the command prompt
    a. Win + R
    b. Type “cmd”
    c. Click OK
3. Navigate to the directory that the main.py file is stored in
    a. Type “cd C:\Users\” and finish it out with the path to the directory containing main.py
4. Type “python main.py” and press enter
5. The program will then prompt you to choose “user” mode or “file” mode.

User Mode Instructions

1. When prompted to pick between reading from a file and using user input, type “user”
2. The program will prompt you for an output file. Type the name that you would like the output file to have. Both examples below are valid as types of input (i.e. you do not need to include the “.txt” file extension):
    a. “output”
    b. “output.txt”
3. You will then be prompted by a mode. You can choose any of the following:
    a. “Validity”
        i. This will prepare the program to receive logical premises and a conclusion for an argument to check for validity.
        ii. Now input the premises and conclusion to the argument. You do not need to do anything to signify what the conclusion is. The program will take the last input premise as the conclusion. Note that only inputting one premise will be an argument conclusion without any premises.
        iii. After inputting the conclusion, input “done” so that the program knows that the argument has been concluded.
        iv. The program will then tell you if the argument you put in is valid or invalid.
        v. The program will then default back to step 3.
    b. “Tautology”
        i. This will prepare the program to receive a logical statement to check if the statement is a logical tautology.
        ii. Now input the statement that you are going to evaluate. You do not need to tell the program that you are finished inputting statements since the program will only check one statement at a time.
        iii. The program will tell you whether the statement is a logical tautology or not.
        iv. The program will then default back to step 3
    c. “Exit”
        i. This will exit the program.
4. After you exit the program, a fuller receipt of the analysis done will be put into the output file. There, you can see the argument or tautology in question and see the full output of all the resolution clauses and which literals are being removed. If the resolution finds an empty set, you will see it make note of that. If no empty sets are ever found, then the program will finish and display the clause sets remaining.

File Mode Instructions

1. When prompted to pick between reading from a file and using user input, type “file”
2. The program will prompt you for an input file. Type the name of the input file with your desired test cases. See Input File Instructions for instructions on how to structure your input files. Both examples below are valid as types of input (i.e. you do not need to include the “.txt” file extension):
    a. “input”
    b. “input.txt”
3. The program will prompt you for an output file. Type the name that you would like the output file to have. Both examples below are valid as types of input (i.e. you do not need to include the “.txt” file extension):
    a. “output”
    b. “output.txt”
4. The program will then work through each test case in the input file. The displayed output will be the number of the test followed by the conclusion made by the program.
5. The program will then exit on its own.
6. After program exit, a fuller receipt of the analysis done will be put into the output file. There, you can see the argument or tautology in question and see the full output of all the resolution clauses and which literals are being removed. If the resolution finds an empty set, you will see it make note of that. If no empty sets are ever found, then the program will finish and display the clause sets remaining.

Input File Instructions

An example of a properly formatted input file is available on the GitHub repository as "test_cases.txt". A small snippet from that file is seen in Figure 1 below.

```
VALIDITY/A|~(B&C),~B,~(A|C),A
TAUTOLOGY/P|~P
```

Figure 1. Input File Snippet

Each test case gets its own line (i.e. a file with 10 tests in it will have 10 lines of text). Each line will contain the type of test being done followed by propositional logic statements. Those logical statements must have valid characters. See Valid Characters section for further clarification.

- For a test of Validity
  - The first text on each line should be "VALIDITY"
  - VALIDITY should be immediately followed by a "/" character. This lets the program know where the premises will begin.
  - Input premises and a conclusion. To separate premises and the conclusion from one another, use commas to denote the start of a new statement.
  - There is no need to put any sort of ending character or special designation for the conclusion.
  - Return to a new line for the next test.
- For a test of a logical Tautology
  - The first text on each line should be "TAUTOLOGY"
  - TAUTOLOGY should be immediately followed by a "/" character. This lets the program know where the statement will begin.
  - Input the statement.
  - There is no need to put any sort of ending character.
  - Return to a new line for the next test.