ITCS – 6150 Project: Tautology checking RS method

Team:

1. **Vinit Shah – 801005774**
2. **Satwik Rao – 801257585**
3. **Koosha Sharifani – 801260796**
4. **Xiangcheng Wu – 801205324**

Introduction:

This project implements a formula verification method called RS proof to determine whether a given formula in the propositional calculus is a Tautology.

Formula F is a propositional tautology if and only if all the end sequences in diagram D(F) are fundamental.

Our Group has implemented both in Python and C++.

Instructions on how to run the program:

1. The Python version (tautology_checker.py)

The program will take the expression as the input and will give output whether the given expression is a tautology or not.

The program will validate the expression and if it is a valid expression, then it will check for tautology. If the expression is a tautology, then it will print all the leaves and fundamental nodes. If the formula is found not a tautology, then it will just indicate that the expression is not a tautology.

Command to run the program: python **tautology_checker.py**

In the next step, we have to input the expression.

When it prompts to provide the Propositional Formula, use variables from a set of a, b, c, and d

- For NOT operation use ~
- For AND operation use &
- For OR operations use |
- For => operations use >>
- For <=> operation use <<

SAMPLE INPUT:

~(a >> c) >> (~(c | d) >> (a & ~c))

~(a >> c) >> (~(c | d) >> (a & c))

Note: Program will run only for a single input expression, if we want to input multiple expressions, we have to run the program multiple times.

Please don't use any other brackets other than parentheses '**(**', '**)**'.

SCREENSHOTS:

Example 1:

```
$ python tautology_checker.py
Enter the Formula: ~(a >> c) >> (~(c | d) >> (a & ~c))
Propositional Formula (~(a => c) => (~(c v d) => (a ^ ~c))) is a Tautology
Leaves of the tree: [['~a', '~c', 'c', 'c', 'd'], ['~a', '(c v d)', 'c', 'a']]
Fundamental nodes: [['~a', '~c', 'c', 'c', 'd'], ['~a', '(c v d)', 'c', 'a']]
```

Example 2:

```
$ python tautology_checker.py
Enter the Formula: ~(a >> c) >> (~(c | d) >> (a & c))
Propositional Formula (~(a => c) => (~(c v d) => (a ^ c))) is NOT a Tautology
```

2. The C++ version (RS_Strategy.cpp)

INPUT: A propositional formula

~ for Negation (# for placeholder in the code. Do not enter it.)

^ for Conjunction

v for Disjunction

> for Implication

(* It supports both () and [].)

(* Spaces are not allowed between characters.)

(* Do not input two ~ continuously. Separate them with parentheses like this: ~(~a).)

INPUT EXAMPLE:

(a>b)>((b>c)>(a>c))

a>(avb)

~(a>c)>[~(cvd)>(a^c)]

~(a>c)>[~(cvd)>(a^~c)]

OUTPUT: 1. The binary tree representation for this sequence

2. The binary tree representation and normal representation for each leaf

(The tree grows from the left side to the right side.)

3. Whether the according leaf is fundamental

4. Whether this formula is a tautology

Please follow the instructions shown on the terminal.

* This project requires compilers supporting C++11 or higher versions. If you are using command lines:

g++ RS_Strategy.cpp -o RS_Strategy -std=c++11

* This project does not contain any function for checking grammar error.

SCREENSHOTS:

Example 1: a>(avb)

```
**************************************************************************

Please Enter a Formula (q for quit):

(Do not input two ~ continuously. Seperate them with parentheses like this: ~(~a).)

a>(avb)

The binary tree for this formula (# is a placeholder for negation):
          b
     v
          a
 >
     a

------------------------------

The binary tree for this leaf (# is a placeholder for negation):
          b
     v
          a
 v
     ~    a

          #

This leaf: a, b, ~a,  (fundamental)

------------------------------

### This formula is a tautology. ###
```

Example 2: ~(a>c)>[~(cvd)>(a^c)]

```
**********************************************************************

Please Enter a Formula (q for quit):

(Do not input two ~ continuously. Seperate them with parentheses like this: ~(~a).)

~(a>c)>[~(cvd)>(a^c)]

The binary tree for this formula (# is a placeholder for negation):
          ^     c
                a
     >
                     d
                v
          ~          c
                #
 >
                c
          >
                a
     ~
          #

------------------------------

The binary tree for this leaf (# is a placeholder for negation):
          a
     v
                d
          v
                c
 v
          c
     v
          ~     a
                #

This leaf: c, c, d, a, ~a,  (fundamental)

------------------------------

The binary tree for this leaf (# is a placeholder for negation):
          c
     v
                d
          v
                c
 v
          c
     v
          ~     a
                #

This leaf: c, c, d, c, ~a,  (NOT fundamental)

------------------------------

### One leaf is not fundamental. This formula is NOT a tautology. ###
```