Heaven's Light is Our Guide

Rajshahi Universiy of Engineering and Technology

Course Code
ECE 2214

Course Title
Numerical Methods and Discrete Mathematics Sessional

Experiment Date: September 25, 2023
Submission Date: October 2, 2023

Lab Report 3: Checking if an expression is a tautology

Submitted to	**Submitted by**
Md. Nahiduzzaman	Md. Tajim An Noor
Lecturar	Roll: 2010025
Dept of ECE, Ruet	

Taking a propositional logic and finding if the logic is tautology or not

Md Tajim An Noor

Introduction

Given a logical expression e.g., $(p \vee \neg p)$, explaining how I would determine whether it is a tautology or not. Additionally, providing an example expression and checking whether they are tautology or not.

Tautology

A tautology is an assertion of Propositional Logic that is true in all situations; that is, it is true for all.[1] The statement "Either it is raining, or it is not raining" is a tautology. This statement is always true because it covers all possible cases: if it's raining, the first part of the statement is true, and if it's not raining, the second part of the statement is true. In either case, the statement as a whole is true.

Tools Used

- Python
- VS Code - for running python code
- MacTeX -LaTeXcompiler
- VS Code with LaTeX workshop extension as a text editor

Process

Code:

```
# this code is to check if a simple and/or logic with 2 variables
↪  is a tautology or not
cnt = 0
a = input()  # taking input string
a = a.replace(" ", "")
# spliting input into variables depending on the logic
if "|" in a:
    b, c = map(str, a.split("|"))
```

```
elif "&" in a:
    b, c = map(str, a.split("&"))

l1 = [True, False]
l2 = [True, False]
result = False

if "-" in b:
    l1 = [False, True]
if "-" in c:
    l2 = [False, True]


def checkTauto(a, b, c):  # function to check tautolog
    global cnt
    if "|" in a:
        if b or c:
            result = True
        else:
            result = False
            cnt += 1  # if false for any case, returns cnt > 0
    if "&" in a:
        if b and c:
            result = True
        else:
            result = False
            cnt += 1
    print("=> " + str(result))  # to print the truth table


def diffVar(a, b, c):
    # to tassign bool values to variables if they are different
    for i in l1:
        for j in l2:
            print(i, j)
            b = i
            c = j
            checkTauto(a, b, c)
    if cnt > 0:
        print("Not tautology")
    else:
        print("Is Tautology")


def sameVar(a, b, c):  # to tassign bool values to variables if they
↪   are same
```

```
    for i in l1:
        for j in l2:
            if b == c:  # if vars are same bool can't be different
                if i != j:
                    continue
            if b != c:  # if vars are same but inverse, bool can't
            ↪  be same
                if i == j:
                    continue
            print(i, j)
            b = i
            c = j
            checkTauto(a, b, c)
    if cnt > 0:  # if cnt>0 at least 1 case is false
        print("Not tautology")
    else:
        print("Is Tautology")


if b not in c:
    diffVar(a, b, c)
else:
    sameVar(a, b, c)
```

Output

```
a|b
True True
=> True
True False
=> True
False True
=> True
False False
=> False
Not tautology
```

```
a&b
True True
=> True
True False
=> False
False True
=> False
False False
=> False
Not tautology
```

```
b&b
True True
=> True
False False
=> False
Not tautology
```

```
a | -a
True False
=> True
False True
=> True
Is Tautology
```

Analysis

Code explanation

The above code is only for finding an expression's tautology if the expression has only two variables and if the expression only contains AND, OR, NOT logics. The two variables can be the same, but for more than two variables, the code will not work. In the code, first an input, the logical expression is taken. Then the spaces in that expression is removed, and then the variables are stored in separate variables. Then using a function, the variables are assigned their boolean values with all possible combinations, and evaluated using another function. At the same time, the truth table of the expression is printed. Here, an integer variable *cnt* is kept, initialized as 0, whenever a case is false, it increments the variable. Finally, if $cnt > 0$, the expression is evaluated as $Not\ tautology$. Otherwise, the expression will be evaluated as $Is\ tautology$.

Discussion

The code is only applicable for simple two variable logical expression. It's not efficient either. There are many other ways it can be done. Due to lack of knowledge, it was not possible for me to do it.
If the code was efficient and more usable for versatile expression, this experiment would've been more successful. For now, the code is satiable.

References

[1] "1.6: Tautologies and contradictions - mathematics libretexts," https://math.libretexts.org/, (Accessed on 10/01/2023).