

Heaven's Light is Our Guide
Rajshahi University of Engineering and Technology



Course Code
ECE 2214

Course Title
Numerical Methods and Discrete Mathematics Sessional

Experiment Date: October 2, 2023
Submission Date: October 9, 2023

Lab Report 4: Python codes to find logical contradiction, contraposition & equivalence

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

Contents

1	Introduction	3
1.1	Contradiction	3
1.2	Contraposition	3
1.3	Logical Equivalence	3
2	Tools Used	4
3	Process	4
3.1	Code:	4
3.1.1	Contradiction	4
3.1.2	Contraposition	4
3.1.3	Logical Equivalence	5
3.2	Output	6
4	Discussion	7

Python codes to find logical contradiction, contraposition & equivalence.

Md Tajim An Noor

1 Introduction

1.1 Contradiction

A statement that is always false is known as a contradiction. A statement or notion that is logically or intrinsically false is referred to as a contradiction. It occurs when a logical argument leads to a situation where two opposing statements cannot both be true at the same time. One common example of a contradiction in discrete mathematics is:

”There exists an integer n such that n is both even and odd.”

This statement is a contradiction because by the definition of even and odd integers, no integer can be both even and odd simultaneously. Even integers are divisible by 2, whereas odd integers are not.

1.2 Contraposition

Contraposition refers to the inference of going from a conditional statement into its logically equivalent contrapositive. The contrapositive of a statement is formed by negating both the hypothesis and the conclusion of the original conditional statement and reversing their order. Here is the general form of a contrapositive:

Original statement: If A , then B .

Contrapositive: If not B , then not A .

In other words, the contrapositive switches the roles of the antecedent (A) and the consequent (B) and negates both of them.

1.3 Logical Equivalence

Compound propositions that have the same truth values in all possible cases are called logically equivalent. We can also define this notion as follows:

The compound propositions p and q are called logically equivalent if $p \leftrightarrow q$ is a tautology. The notation $p \equiv q$ denotes that p and q are logically equivalent.[1]

2 Tools Used

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

3 Process

3.1 Code:

3.1.1 Contradiction

```
1 import sympy
2 from itertools import product
3 from sympy.logic.boolalg import truth_table
4 from sympy.abc import x, y
5
6 p, q = sympy.symbols("p q")
7
8 # Defining logical expressions
9 expr = ~(p | ~p)
10
11 table = truth_table(expr, [p, p])
12 contradiction = True
13 for t in table:
14     # printing the truth table
15     print("{0} -> {1}".format(*t))
16     if t[1] == True:
17         contradiction = False
18         break
19
20 if contradiction:
21     print("The statement is a contradiction.")
22 else:
23     print("The statement is not a contradiction.")
```

3.1.2 Contraposition

```
1 def find_contrapositive(sentence):
2     # Split the sentence into words
3     words = sentence.split()
4
5     # Find the position of "if" and "then" in the sentence
6     if_index = -1
```

```

7     then_index = -1
8     for i, word in enumerate(words):
9         if word.lower() == "if":
10             if_index = i
11         elif word.lower() == "then":
12             then_index = i
13
14     # Check if both "if" and "then" were found
15     if if_index != -1 and then_index != -1:
16         # Identify the p (if) and q (then) parts of the statement
17         p = " ".join(words[if_index + 1 : then_index])
18         q = " ".join(words[then_index + 1 :])
19
20         # Negating both the p and the q
21         np = "it is not true that " + p
22         nq = "it is not true that " + q
23
24         # Form the contrapositive statement
25         contrapositive = f"If {np}, then {nq}"
26
27         return contrapositive
28     else:
29         return "Invalid input sentence. Please use 'if' and 'then' in
        ↪ your sentence."
30
31
32 # Example usage:
33 original_sentence = "If he comes then I will go."
34 contrapositive_sentence = find_contrapositive(original_sentence)
35 print("Original:", original_sentence)
36 print("Contrapositive:", contrapositive_sentence)

```

3.1.3 Logical Equivalence

```

1 import sympy
2
3 # Define symbolic variables
4 p, q = sympy.symbols("p q")
5
6 # Define your logical expressions. Here we enter the logical
   ↪ expressions we want to check equivalence of.
7 # Here for testing, De Morgan's theorem is used.
8 expr1 = ~(p & q)
9 expr2 = ~p | ~q
10
11 # Check for logical equivalence: Simplifying the logical
   ↪ expressions and then storing the boolean value in a variable.

```

```

12 equivalent = sympy.simplify_logic(expr1) ==
    ↳ sympy.simplify_logic(expr2)
13 # From the previous statement, we get if the expressions are
    ↳ logically equivalent.
14 if equivalent:
15     print("The expressions are logically equivalent.")
16 else:
17     print("The expressions are not logically equivalent.")

```

3.2 Output

The figure consists of three screenshots of a Jupyter Notebook interface, each showing a different logical expression and its corresponding truth table output.

Top Left Screenshot: The code defines `expr = p & q`. The output shows the truth table for `p & q`:

p	q	p & q
0	0	False
0	1	False
1	0	False
1	1	True

The statement is not a contradiction.

Top Right Screenshot: The code defines `expr = p & ~p`. The output shows the truth table for `p & ~p`:

p	~p	p & ~p
0	1	False
0	0	False
1	1	False
1	0	False

The statement is a contradiction.

Bottom Screenshot: The code defines `expr = ~(p | ~p)`. The output shows the truth table for `~(p | ~p)`:

p	~p	p ~p	~(p ~p)
0	1	1	False
0	0	0	False
1	1	1	False
1	0	1	False

The statement is a contradiction.

Figure 1: Outputs for Contradiction

The figure consists of three screenshots showing original and contrapositive statements.

Top Screenshot:

Original: If it rains, then I will go out.
 Contrapositive: If it is not true that it rains,, then it is not true that I will go out.

Middle Screenshot:

Original: If it rains, then I won't go out.
 Contrapositive: If it is not true that it rains,, then it is not true that I won't go out.

Bottom Screenshot:

Original: If he comes then I will go.
 Contrapositive: If it is not true that he comes, then it is not true that I will go.

Figure 2: Outputs for Contraposition

The figure consists of three screenshots of a Python IDE, likely JupyterLab, showing the results of logical equivalence tests using SymPy. Each screenshot displays a code editor with two expressions, `expr1` and `expr2`, and a terminal window below it.

- Top Left Screenshot:** The code defines `expr1 = ~(p | q)` and `expr2 = ~p & ~q`. The terminal output shows "The expressions are logically equivalent."
- Top Right Screenshot:** The code defines `expr1 = ~(p & q)` and `expr2 = ~p | ~q`. The terminal output shows "The expressions are logically equivalent."
- Bottom Center Screenshot:** The code defines `expr1 = ~(p | q)` and `expr2 = ~p | ~q`. The terminal output shows "The expressions are not logically equivalent."

Figure 3: Outputs for Logical Equivalence

4 Discussion

To solve the three problems, the python library *Sympy* was used. SymPy is an open source computer algebra system written in pure Python. It is built with a focus on extensibility and ease of use, through both interactive and programmatic applications.[2] Using SymPy, the input logical sequence can easily be processed. otherwise there would be too many cases to take care of. Also it'll be harder to process big logical inputs.

Using SymPy's built-in functions, the inputs, which are logical sentences, are processed to a simplification, and then using other functions, the logical operations were done.

References

- [1] K. H. Rosen, *DISCRETE MATHEMATICS AND ITS APPLICATIONS, SEVENTH EDITION*. McGraw-Hill.
- [2] R. P. M. GRANGER, F. BONAZZI, H. GUPTA, S. VATS, F. JOHANSSON, and M. J. FABIAN PEDREGOSA, "SymPy: Symbolic computing in python."