Ulep, Jeiwinfrey P.

BSCS – 2A

I. Introduction

Truth tables are a fundamental aspect of Boolean logic, systematically representing the behavior of logical operations such as AND, OR, and NOT. They are essential for programmers to evaluate and reason about complex logical expressions. Boolean algebra underpins logic systems, including propositional and predicate logic, and plays a crucial role in designing efficient, reliable code that governs decision-making in programming. For example, the truth table for conjunction (AND) indicates that the result is true only if both operands are true, while disjunction (OR) is true when at least one operand is true (Brandt, 2023).

A deep understanding of truth tables allows programmers to rigorously define the outcomes of logical operations, which is vital for developing algorithms and proofs within discrete structures (Brandt, 2023). Mastery of Boolean algebra and truth tables ensures that code accurately reflects intended logical behavior, resulting in robust and efficient software designs (Pang & Membrey, 2017).

In conclusion, a thorough understanding of truth tables is essential for grasping the principles of logic, programming, and computer systems' fundamental operations. Truth tables facilitate the evaluation of complex logical expressions, aiding in the design of accurate algorithms that maintain logical consistency in decision-making. This foundational knowledge enhances comprehension of data processing at a fundamental level, empowering developers to create robust software solutions and effectively address intricate real-world problems through rigorous logical reasoning. This competence is indispensable for navigating the complexities of computer science and software development.

II. Description of Variables

Libraries:

- **import tkinter as tk: This is for the GUI toolkit.**
- **from tkinter import ttk: This is for the themed Tk Gui.**
- **from tkinter import font, filedialog: This is for the font and file selection.**
- **from tabulate import tabulate: This is to create formatted tables.**

Variables:

- **truthTable:** A dictionary that stores truth values for the variables and the result of the logical equation.
- **logEq:** The logical equation input by the user, which is evaluated to generate the truth table.
- **p, q:** Represents the basic logical variables used in logical equations. They are assigned a 0 (False) or 1 (True).
- **def implication:** A function that returns the result of implication operation for the elements p and q.
- **def negation:** A function that returns the logical negation of p and q. Yes, it only has p in the function line but it also negates the variable q because negation is a unary operator.
- **def conjunction:** A function that returns the result of conjunction operation for the elements p and q. (AND)
- **def disjunction:** A function that returns the result of disjunction operation for the elements p and q. (OR)
- **def equivalence.** A function that determines whether p is equal to q. (==)
- **variables:** A list of variables extracted from the logical equation. It checks if which variable is present in the logical equation, whether ‘p’, or ‘q’, or ‘p’ and ‘q’.
- **st:** It’s a stack to keep track of opening parentheses encountered in the logical equation. It ensures that each closing parenthesis matches the most recent opening parenthesis.
- **allowedChars:** Characters allowed/valid in the logical equation.
- **def validLogEq:** It ensures that the logical equation contains valid operands and operators, and that the parentheses are balanced.
- **def evaluation:** Function to evaluate the logical equation.
- **precedence:** Precedence determines the order in which operators are evaluated in an expression for it to get the truth values right.
- **operand:** A list used to store operands (variables and their negations) during the evaluation of the logical equation.
- **operator:** A list used to store operators (e.g., conjunction, disjunction) during the evaluation of the logical equation.
- **def inputFromTxtFile:** Function to input logical equation from a text file.

III. Flow of the Program

1. Input of the logical equation (Either from the text box or from txt file).
2. Remove all blank spaces and convert the alphabet characters to lowercase.
3. Check if the logical equation is valid (Character checking which includes parentheses, operators, and letters p and q).

   - If invalid, display error message and exit

4. Extract variables from logical equation (To know how many columns to show in the table, whether p, q, or ‘p’ and ‘q’).
5. Generate the truth table for variables.
6. Populate the truth table for each variable.
7. Evaluate logical equation.

   - Initialize operand and operator stacks.

   - Iterate through each character in logical equation.

   - If '(' is found: push to operator stack.

   - If ')' is found: Pop from the parentheses stack to get the position of the most recent (.

   - Perform calculations for the expression within the parentheses.

   - If operand (p, q): push to operand stack.

   - If operator (^, v, ->, <->, ~).

   - Handle precedence and perform calculations using calculate function.

   - Push operator to operator stack.

   - Perform remaining calculations in operator stack.

   - Store final result.

8. Print truth table.
9. Display result in result label.

IV. Error Handling

1. Check for Invalid Characters: Ensure that the logical equation contains only allowed characters.
2. Check Parentheses Balance: Ensure that every opening parenthesis has a corresponding closing parenthesis.
3. Validate Operator Placement: Ensure that operators are placed correctly with respect to operands and parentheses.

Common Errors:

Operator at the Beginning or End: An operator should not be at the beginning or end of the logical equation.
Example: ^pq or pq^

Consecutive Identical Operators: Two exact same operators should not be placed consecutively. Note: negation (~) after operators work.
Example: p^^q or pvvq
Operator without Operands: An operator should not be placed without the required number of operands.
Example: p^ or ^q

Negation Operator Placement: The negation operator ~ should be followed by an operand or an opening parenthesis.
Example: p~q or p~v

Implication and Equivalence Operators: These operators should be placed between two operands.
Example: p-> or <->q

Absence of Parenthesis Match: All parentheses opening should always have a corresponding closing parenthesis, same with closing parenthesis. All should have a match/partner.
Example: (p^(pvq) or (pvq

4. Handle Empty Input: Ensure that the input is not empty.
5. Provide Clear Error Messages: Inform the user about the specific error encountered.

V. User's Manual

a. Installation

- Open a terminal or command prompt.
- Run the following command:

```
pip install tabulate
```

b. Running the Application

- Navigate to the directory containing the ulepJeiwinfreyTruthTable.py file.
- Run the script using Python.

c. Using the Application

- Enter Logical Equation (a or b).

  a) In the text box, enter the logical equation by inputting/typing your logical equation using the allowed characters.

  b) Click the "Input from TXT File" button, then it pops up a file selection. After, select your .txt file containing your logical equation. The content will be loaded into the input box.

- Evaluate the Logic Equation for the result of Truth Table.

  a) Click the "Evaluate" button to process the logical equation you inputted in the text box.
  b) The result will be displayed automatically if the logical equation input is from a .txt file.

- The Truth Table result will be displayed in the output below the buttons.