5'th semester
CSED - 2016

¬Boolean

DISCRETE MATHEMATICS

Teacher assistant: Amr Nabil Sharaf
Name: Omar Ahmed Hussin
Phone: 01121131933
E-Mail: omarahmedhussin1@gmail.com
ID: 40

Overview

This application is a very nice tool that solves the logical expressions & make it's truth table & test it for tautology or contradiction and the most important thing it provides multiple expressions evaluation as well as multiple expressions testing for equality.

Problem Statement

It's required to design and implement a program that takes a logic expression and draws the corresponding truth table. Then user can find the truth value of the expression given the values of its propositions, determine if the expression is a tautology or a contradiction, and test if two expressions are equivalent by comparing the truth table for both of them.

Application features

1. User-friendly GUI interface [**BOUNCE**]
2. Input validation
3. Standard logical connectives [<->,-> **BOUNCE**]
4. Visual view of the truth table
5. Save the truth table in a text file
6. Truth table evaluation [multiple expressions **BOUNCE**]
7. Testing for tautology and contradiction
8. Testing for equivalence [multiple expressions **BOUNCE**]

Data Structure

In this app the developer used collections of data structure, like, array lists and complex objects that holds number of array lists (ex: data shape class)

Algorithms

Complete search

Generating masks by searching for all possible masks that the expression can be evaluated with

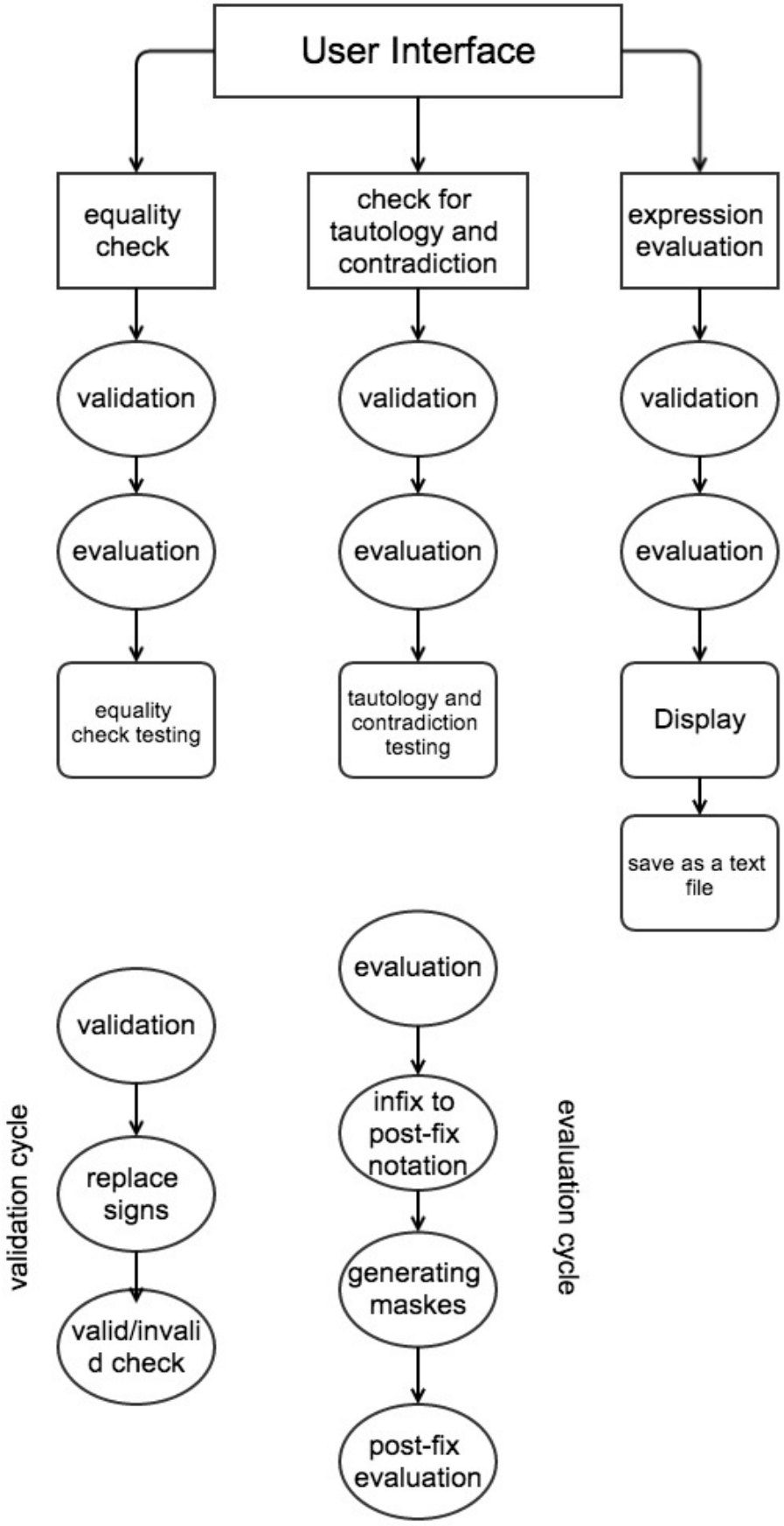
Infix to post-fix notation

Converted the expression to postfix notation in order to evaluate it

Post-fix evaluation

Used this algorithm to evaluate the post-fix expression with the mask we generated

Application design



Application design description

1) Validation cycle

Simply the validation procedure starts with taking every single expression and first replaces the logical connective with characters, which the program can deal with and here is the decision making for this step.

- & to &
- | to |
- ! to !
- <-> to *
- -> to >

Then the program takes the string after replacing signs operation then check for it's validity, by testing every fault can happen in the expression, (ex: two symbols without logical connective)

2) Evaluation cycle

After check for validity now the expressions are ready to be evaluated so the following procedure occurs.

(i) Change from infix to post-fix notation:

By using the algorithm described in this web-site
http://scriptasylum.com/tutorials/infix_postfix/algorithms/infix-postfix/

(ii) Generating masks

I made a class that generate a mask from a given integer by change it into binary representation into Boolean array and the algorithm described in this web-site
http://www.wikihow.com/Convert-from-Decimal-to-Binary

(iii) Post-fix notation evaluation

The evaluation for the expression by using every mask generated above and applies it to evaluate the expression, the algorithm described in this web site.
http://scriptasylum.com/tutorials/infix_postfix/algorithms/postfix-evaluation/

The precedence order of the logical connectives is taken from Wikipedia, the link is also here, http://en.wikipedia.org/wiki/Logical_connective

Operator	*Precedence*
¬	1
∧	2
∨	3
→	4
↔	5

3) Expressions evaluation

As in the flow chart it the validation and evaluation cycles take place first then after evaluation it's time to display the truth table to the user, so the program takes the data (symbols and masks and expressions and expressions values) turn it into 2 dimensional string array in order to represent it in a java table object.

Second step is the save option which enables the user to save the truth table any where as a text file, the application used the java file chooser applet to handle the path and saving, and also there is a save method that print the data in the target file.

In order to handle the java table and file chooser I visited couple of links helped me a lot organize the GUI interface and make it user-friendlier.

4) Tautology and Contradiction check

In this case after validation and evaluation cycles, it takes only one expression to do this test and iterate over the expression values to determine if it a tautology or contradiction or none.

5) Equality check

This function takes multiple expressions and checks its corresponding truth values it is equal or not by validation the evaluation then iterate to check the equality.

Application assumptions

- The logical operator that we deal with is as follows (without the brackets)
  - AND is represented by [&]
  - OR is represented by [|]
  - NOT is represented by [!]
  - IMPLIES is represented by [->]
  - BICONDITIONAL is represented by [<->]
- The symbols of the operands are only one latter and lowercase letter is different from uppercase which means 'a' is different from 'A'.
- The application is written in java and it needs a java runtime environment to run.

Application complexity

1. Validation complexity
   - O(2n*m) n = the length of the expression , m = the number of the expression

2. Evaluation complexity
   - $O(2n*2^m*k)$ n = the length of the expression, m = the number of symbols, k = number of expression