

Alexandria University
Faculty of Engineering
Computer and Systems Engineering Dept.
First Year



Discrete Mathematics
Fall 2013
Assigned: Saturday 2nd Novt, 2013
Due: Saturday 23rd Nov, 2013

Truth Table Construction

Second Year, Discrete Mathematics

Programming Assignment

The team's names:

- 1) Remon Hanna Wadie Youssef [no.23]
- 2) Mina Makram Louis Eldehaiby [no.71]

1) 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.

2) Data structure:

- a) ArrayList<String>.
- b) Stack<String>.
- c) String [] 2d and 1d.

3) Algorithms:

1) Algorithm of converting the expression into postfix:

a- convert the infix into array containing the variables and the operands:-

```
b- for i=0 → infix.size(){
    if (character(i)("~")) {
        stack.push("~");
    } else if (character(i)("^")) {
        if (stack.isEmpty() or stack.peek()=="(") {
            stack.push("^");
        } else {
            while (!stack.isEmpty() and (stack.peek()=="~" or
            stack.peek()=="^")) {
                postfix.add(stack.pop());
            }
            stack.push("^");
        }
    } else if (character(i)("v")) {
        if (stack.isEmpty() or stack.peek()=="(") {
            stack.push("v");
        } else {
```

```

        while (!stack.isEmpty() and (stack.peek()=="~")or
        stack.peek()=="^" or stack.peek()=="v")){
            postfix.add(stack.pop());
        }
        stack.push("v");
    }
} else if (character(i)="-->") {
    if (stack.isEmpty() or stack.peek()=="("){
        stack.push("-->");
    } else {
        while (!stack.isEmpty() and (stack.peek()=="~")or
        stack.peek()=="^"or stack.peek()=="v" or
        stack.peek()=="-->")) {
            postfix.add(stack.pop());
        }
        stack.push("-->");
    }
} else if (character(i)("<-->")) {
    if (stack.isEmpty() or stack.peek()=="(") {
        stack.push("<-->");
    } else {
        while (!stack.isEmpty() and (stack.peek()=="~")or
        stack.peek()=="^"or stack.peek()=="v"or
        stack.peek()=="-->" or stack.peek()=="<-->")) {
            postfix.add(stack.pop());
        }
        stack.push("<-->");
    }
} else if (character(i)("(") {
    stack.push(infix.get(i));
} else if (character(i)("")) {
    while (!stack.isEmpty() and !stack.peek()=="(") {
        postfix.add(stack.pop());
    }
    if (!stack.isEmpty())
        stack.pop();
} else {
    postfix.add(infix.get(i));
}
}

}

while (!stack.isEmpty()) {
    postfix.add(stack.pop());
}

```

now we have the postfix of the expression.

2) Algorithm for evaluating the postfix expression:

Create stack of booleans;

```
for (int i = 0; i < expression.size(); i++) {
    if (expression.get(i).equals("T")) { stack.push(true);}
    else if (expression.get(i).equals("F")) {stack.push(false);}
    else {
        if (expression.get(i).equals("~")) {stack.push(not s.pop());}
        else {
            // this is an operator
            // pop the two variables in the stack and evaluate the
            // operation then push the result in the stack
            result = evaluateOperation(s.pop(),
            s.pop(),expression.get(i));
            stack.push(result);
        }
    }
}
```

Return result;

4) Assumptions and details:

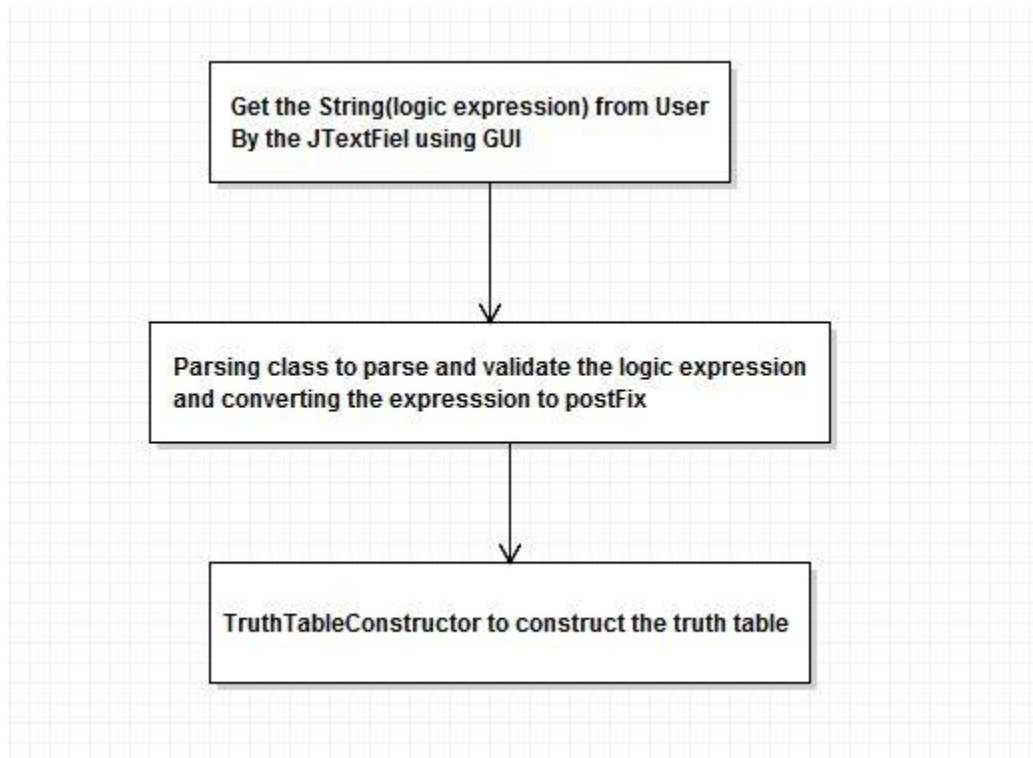
- 1) The user can enter any number of spaces in the expression.
- 2) We assume the symbols of the operators as shown in this table:

Operator	Symbol
And	\wedge
Or	\vee
Not	\sim
Implication	\rightarrow
Biconditional	\leftrightarrow

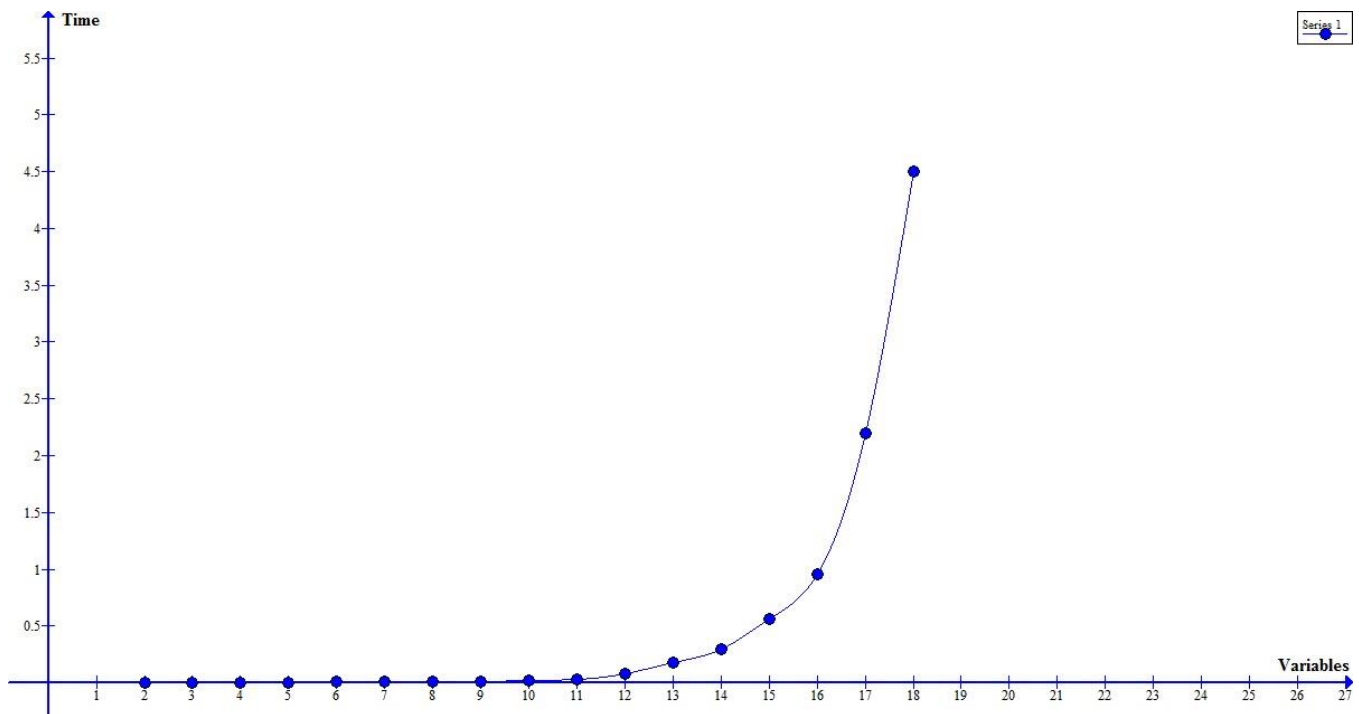
- 3) Any proposition is one character (A – Z) or (a – z) except (v).

4) When the user want to evaluate the expression in particular values he should enter the values in Upper Case Latters ('T' or 'F').

5)The design:



6)Program Capacity Analysis:



The maximum number of propositions the program can support is 18.

6)Extra features:

- Load Truth Table from a text file.
- The expression can contain this operands (**Implication** (\rightarrow) and **Biconditional** (\leftrightarrow)).

7)User guide:

a)Enter the expression in this text field:

Enter The Expression :

Show Table

b) Press on Button Show Table .

c) The truth table of the expression will be generated like that:

Enter The Expression :

$T \vee H^A \sim A$

Show Table

Save

Load

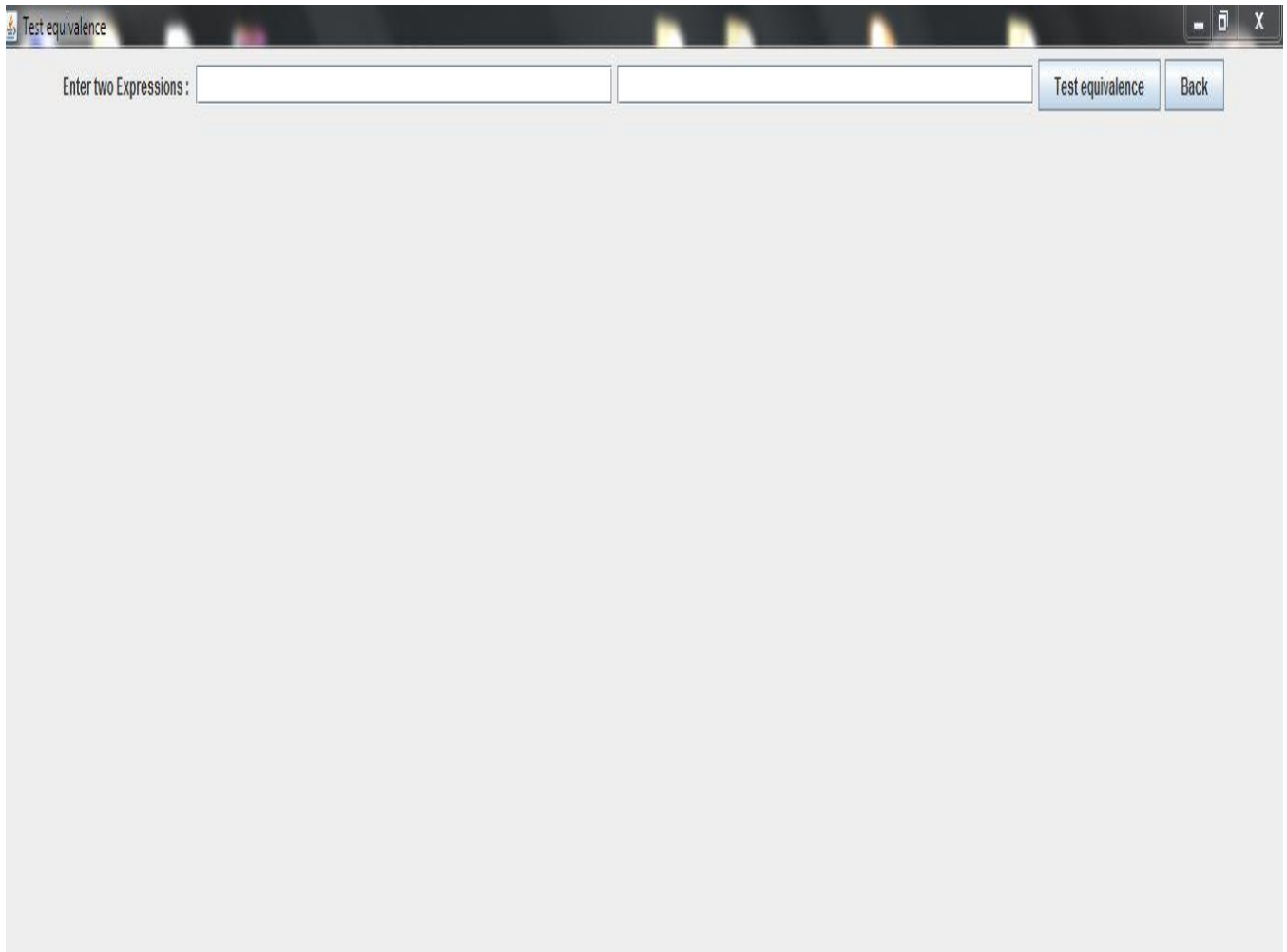
Test equivalence

T	H	A	Output
F	F	F	F
F	F	T	F
F	T	F	T
F	T	T	F
T	F	F	T
T	F	T	T
T	T	F	T
T	T	T	T

d)To save the truth table press on button (Save) and choose the location of the file to save in it.

e)To load truth table from a file press on button (Load) then choose the file that you want to load it.

f)To test the equivalence of two expressions press on button (Test equivalence) then you show this window



Test equivalence

Enter two Expressions :

Test equivalence Back

then enter the two expressions you want to test in these text fields the press on the button (Test equivalence) then you will show the truth table of each expression and the result like this:

Test equivalence

Enter two Expressions: $A \wedge B \vee C$ $A \wedge B \vee C$ Test equivalence Back

A	B	C	Output
F	F	F	F
F	F	T	T
F	T	F	F
F	T	T	T
T	F	F	F
T	F	T	T
T	T	F	T
T	T	T	T

A	B	C	Output
F	F	F	F
F	F	T	T
F	T	F	F
F	T	T	T
T	F	F	F
T	F	T	T
T	T	F	T
T	T	T	T

The 2 expressions are equivalent

h) If you want to back press on button (Back).

i) If there is a tautology or a contradiction in this expression you will know from the upper right label like this:

Enter The Expression: $T \vee \sim T$ Show Table Save Load Test equivalence A Tautology in this expression

T	Output
F	T
T	T

Enter The Expression: $\sim (T \vee \sim T)$ Show Table Save Load Test equivalence A Contradiction in this expression

T	Output
F	F
T	F

i)If you want to evaluate the expression in some values press on button (Evaluate Expression) then you will go to this window:

Truth Table Construction developed by Mina Makram and Remon Hanna

Enter the Expression

true

a f y J

enter the expression and press on button (Enter Values) then enter the values in the text fields in uppercase values ('T' or 'F') then press evaluate then you will show the output in JLabel in the upper right of the frame.

6)Sample runs:

Test equivalence

Enter The Expression :

Show Table Save Load Test equivalence A Tautology in this expression

T	g	Y	Output
F	F	F	T
F	F	T	T
F	T	F	T
F	T	T	T
T	F	F	T
T	F	T	T
T	T	F	T
T	T	T	T

Enter The Expression : $F \vee G^{\wedge} \sim (\sim S^{\wedge} \sim K) \leftrightarrow H$

Show Table

Save

Load

Test equivalence

F	G	S	K	H	Output
F	F	F	F	F	T
F	F	F	F	T	F
F	F	F	T	F	T
F	F	F	T	T	F
F	F	T	F	F	T
F	F	T	F	T	F
F	F	T	T	F	T
F	F	T	T	T	F
F	T	F	F	F	T
F	T	F	F	T	F
F	T	F	T	F	F
F	T	F	T	T	T
F	T	T	F	F	F
F	T	T	F	T	T
F	T	T	T	F	F
F	T	T	T	T	T
T	F	F	F	F	F
T	F	F	F	T	T
T	F	F	T	F	F
T	F	F	T	T	T
T	F	T	F	F	F
T	F	T	F	T	T
T	F	T	T	F	F
T	F	T	T	T	T
T	T	F	F	F	F
T	T	F	F	T	T
T	T	F	T	F	F
T	T	F	T	T	T
T	T	T	F	F	F
T	T	T	F	T	T
T	T	T	T	F	F
T	T	T	T	T	T