< DNF Converter>

1. Introduction

DNF Converter is a program that transforms a propositional formula into a Disjunctive Normal Form (DNF). DNF is a two-level propositional formulas which has form of $(a_{11} \wedge a_{12} \wedge \cdots \wedge a_{1n1}) \vee (a_{21} \wedge a_{22} \wedge \cdots \wedge a_{1n2}) \vee \ldots (a_{m1} \wedge a_{m2} \wedge \ldots \wedge a_{mnm})$ where a_{ij} is p or $\neg p$ for an atomic propositional variable p. Every propositional formula has an equivalent DNF form. So, our program will consume the propositional formula and produce the result through several procedures. After changing into DNF, we will print whether each a_{ij} is True or False. If a_{ij} is mean True, -a_{ij} is False.

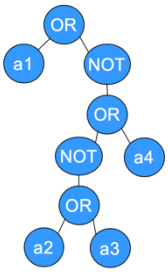
2. Approach

When given a propositional formula, program should change and print it in DNF format.
All input must be prefixed in the parentheses. At output expression, the numbers in the same row are all connected by disjunction, and each row is connected by conjunction. And the program must print out an error message if the given input does not follow that rule.

2.1) Solution Design

Our DNF Converter follow this procedure to solve the problem. Each procedure has detailed description to help reader understand.

[1] Parse the proposition formula
First, input will be split by space so that each expression goes into binary tree structure. For example, if the propositional formula (or a1 (not (or (not (or a2 a3)) a4))) converts into tree structure, the form or tree will be like this.



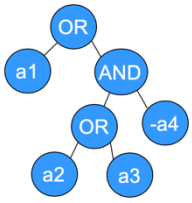
<Graph 1. Tree structure>

Each propositional variable will be a node and has it's own value. (AND: 1, OR: -1, NOT: 0)
When each expression goes into binary tree, there are some rules. First, if the node data is zero, the subtree must be created on the left node. Second, if a node data has a value 1 or -1, it has to check if the left node or right node is null. If the left node has a null value, make a subtree on the left node, and if the left node is full, go to the right node and make a subtree.

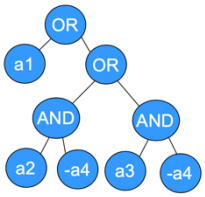
[2] Negate all child nodes of 'NOT' nodes and delete it
After the all variable goes into tree, the NOT node must be eliminated. Before that, all child nodes of NOT node must be negated. AND becomes OR, OR becomes AND, and each numerical variable will be negative integer. After all nodes negated, NOT node will be deleted in tree.

[3] Apply the distributive law
To apply distributive law, Let's see the structure of tree. To interpret the final expressions, we should make the form of disjunctive clauses. If you see graph 2, you can see that there is OR in the child node of AND.

We must place OR node to the root, OR should not be among the child node of the AND node. Then, apply the distributive.



<Graph 2. Before the distributive law>.



<Graph 3. After the distributive law>

When the parent node of the tree with applied NNS is AND, such as the tree of Graph2, if the child node has OR, the OR node and the AND node are changed to make OR come to the root and the child nodes come to AND, thus apply a distributive law. We apply the distributed law while watching all nodes using the recursive function. The function that applies the distributive law as much as the depth of the tree is executed, so the distribution law is applied to all nodes.

[4] Interpret the final expressions and print the result.
Now we can extract disjunctive clauses from tree. All numerical value node who has AND and OR node for the parent will be the line of output result. With the graph above, the output will be a1 for one line, a2, -a4 for second line and a3, -a4 for third line.

[5] Satisfiability checking result

For a given formula, print out satisfiability checking result. If the formula is satisfiable, print out a solution. Represent a solution as a list of integers (1~i) is either i or -i. If integer is True for the solution, it prints i. If it is false, prints -i. The case of unsatisfiable is the value of child nodes of the AND node the absolute value is the same and denotation is different, this case our program will print "UNSAT". For our program, if all the child nodes of AND and OR node are true, except unsatisfiable case, then they are satisfiable. So, we think all child nodes of AND and OR node is true.

3. Result

```
Enter the expression : (or a1 (not (or (not (or a2 a3)) a4)))
1
2 -4
3 -4
0
1 2 3 -4
```

```
Enter the expression : (and a1 (not (or (not (or a2 a3)) a4)))
1 2 -4
1 3 -4
0
1 2 3 -4
```

```
Enter the expression : (and a1 (not a1))
1 -1
0
UNSAT
```

4. Discussion

The program return error message "Wrong input.. Exit program!" for the wrong input that isn't propositional formula. But if we do not input the correct format when inputting the propositional form, the strange value is outputted. For example, if enter the expression "(and a1 not(a1))", output is "1 63". It is necessary not only to check whether the input is the propositional formula, but also to check whether the propositional formula is input in the correct format. However, the code to distinguish it is not completed, so should be careful when entering it.