<Discrete Mathematics Homework #2>

21800001 Narin Kang

1. Introduction

The second personal task is changing a formula to DNF. DNF is short for 'Disjunctive Normal Form', which means that each clause of formula connected by conjunction, while the biggest clauses are connected by disjunction. Also each clause is a combination of smaller clauses. If we divide each clause into smaller parts that cannot be split more, it becomes an atomic propositional variable. The atomic propositional variable can have the original value or its negation.

2. Approach

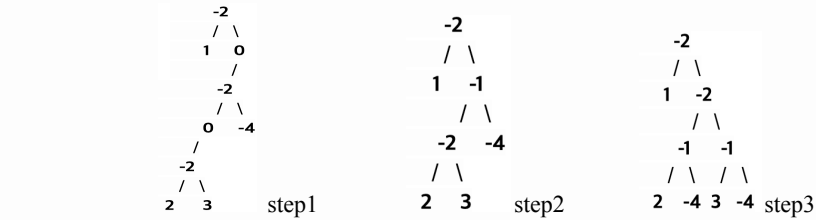
Because the clause consists of a repetitive structure with a clause, we can solve this problem by using recursion. Also because each section can be divided into left and right based on conjunction or disjunction, a tree structure can be used for problem solving.

First step to solve this problem is to convert the given formula into a tree structure. Because input is prefix format, operand(or, and, not...) is always located next to every '(' character. Using this feature, if the '(' character comes to the input, call the function recursively and create a new node and store -1 if word is 'and', -2 if 'or', 0 if 'not' and the integer after 'a' letter if operand. Then connect the next clause to the left node and next to the right node.

Secondly, change the given tree structure to negative normal form. This process is done by Depth-First Search. From the top node of the tree, if the value in one node is zero and the value of its child nodes have a positive integer value, multiply each child node by -1 and then connect the parent node of the ‘0’ node to the child nodes. Also, if the value in one node is zero and the value of its child nodes have negative integer value, change the value to -2 if each child node has -1, change to -1 if it has -2. Then connect the nodes as described earlier. If the negation is taken twice, it's the same as it was originally, so if the node with 0 value appears as an even number time, don't change the value but only connect the nodes.

The final step is to convert NNF to DNF. If any two nodes are connected with 'conjunctions, i.e. if there is a node which has a value of -1, store every value of the left child nodes and the right child nodes if the values mean operand, not operator. Then the left and right values are paired one by one to create a new small tree connected by a new node which has -1 for its value. Each tree is then connected by a new node which has -2 for its value recursively upto the position of the original tree.

The tree shapes and values after completing each step are as follows:



3. Result

input : (or a1 (not (or (not (or a2 a3)) a4)))
output: 1
2 -4
3 -4

4. Discussion

While solving HW2, I initially thought that I could distinguish operator and operator by just the difference in sign. However, in the process of converting the tree to NNF, -1 can be multiplied with each operand. So it was necessary to add a variable to the tree struct that distinguishes whether the node's value is an operator or an operand.