DNF Converter - 21800321 Jihee Park

1. Introduction

   This program changes the sentence that is propositional formula to the DNF formula. It means all propositional formula starts with ‘or’ gate except the situation when the formula is connected only ‘and’ or ‘not’ gate. Then, print out the solution assuming the formula is ‘True’, but if the formula cannot be ‘True’, print out ‘USAT’.

   A. Split the string by white space and make it tree-like form.
   B. Check the ‘not’ node and if there is, use De Morgan’s law.
   C. Check the ‘and’ node has ‘or’ node and if there is, use distribution’s law.
      Input: (gate element1 element2 ... element n (gate element1 ... element k) ... element m) Output)
   D. Make answers at all leaf node and if there is no solution, print out ‘USAT’.

2. Approach
   A. Get Tree
      a. Split the string by white space and if it is same with ‘(and’/ ‘(or’/ ‘(not’, make node to conform that type and go to next word.
      b. If the divided word is a form of ‘a_n’, make node to conform that type and return that node.
   B. NNF Converter
      a. If there was ‘not’ node in the past an, use De Morgan’s law and go to a child node.
      b. If the current node is not a ‘not’ and no ‘not’ node has been released before, it will go to a child node.
      c. If the node is null pointer, return null.
   C. DNF Converter
      a. (and a_1 a_2 … an (or b_1 b_2 … b_k) … a_m) > (or (and a_1 ...an b_1...a_m) … (and a_1 … a_n b_k … a_m))
      b. If the current node is ‘and’ node and there is ‘or’ node in child node, use distribution’s law as above.
      c. Go to the child node and if the node is null pointer, return null.
   D. Error
      a. Obtain the number of ‘)’ that should have been the last child node in the current node and output an error and exit the program.
      b. If a type of the current node is ‘and’/ ‘or’ and size is 1, output an error and exit the program.
   E. Make Answer
      a. In ‘or’ node, a leaf node makes true, a leaf node in ‘not’ node makes false and go to child node.
         - Check the all child node solution and if there is no positive answer and no leaf node in current node, print our ‘USAT’ and exit the program.
      b. In ‘and’ node, all node makes true and when there is an error like the situation when the same leaf node has two answer, check the root node is ‘and’.
         - If the root node is ‘and’ print out ‘USAT’ and exit the program.
         - Otherwise, go on and return -1.
      c. If there is no error in making solution, return 1.

3. Result
   A. Because I check many forms that cannot make solution and apply that to this program, this will answer the good solution.
   B. ANS
      Input: (or a1 (not (or (not (or a2 a3)) a4)))
      ➔ 1 ↳2 -4 ↳3 -4 ↳0 ↳1 2 3 -4
      Input: (and a1 (not a1))
      ➔ 1 -1 ↳0 ↳USAT
      Input: (or (and a1 (not a1)) (and a2 (not a2)))
      ➔ 1 -1 ↳2 -2 ↳0 ↳USAT

4. discussion
   A. I had a lot of interest in making solution without using z3, and I have a hard time with wrong result because I didn't think about a lot of cases.
   B. I think that the program would be better if I can make a program making a formula example.