Discrete Mathematics
21800147 YouYoung Kim

Homework 2

DNF Converter

A. Description

a. Intro

DNF converter is a program that converts the proposition formula into a disjunctive normal form. All well-formed propositional formula can be represented with tree like data structures. I assigned tree's node to each and all logic forms and variables. In this way, DNF form is that any other type of node of tree cannot locate above 'or' node. I put well-formed inputs into the data structure by creating data structures based on trees. For making clear and easy to implement, applying NNF forms are also necessary. Print out what the solution is if you are satisfiability with the results made and print "UNSAT" otherwise.

b. Approach

Firstly I divided the types into variable and operations. A variable will have a specific id that differentiates itself from others. For operations, there are 3 specifics: "and", "or", and "not". Operations should have at least one or more children, so I ended up thinking a tree like structured linked list. Then parsing algorithm breaks input string into smaller strings identifying a variable and operations. This process is recursively done to each smaller strings when those strings are identified as operations. At every step, a data structure tree will grow.

When parsing ends up successful, NNF is applied. The tree will be searched, and fixed when illegal 'not' operations are found. I used recursive algorithms dividing by types of propositional formula. I found that the condition for applying distribute law and commutative law is the last step. I applied Distribute law when 'and' node is a parent of any 'or' nodes. Commutative law was applied when a parent node and a child node is same operators, such as 'and'& 'and', or 'or'& 'or'. The tree is in DNF form; however; I wanted to provide a clear and sorted view to the user. Therefore, I added a quick sort algorithm to rearrange variables and operations.

Check the satisfiability with the resulting values from the DNF converter. It is a formula bound by 'or' operation, only one line of each line needs to be true. When meeting 'not' operation, send -1 as parameter to change the value to negative, and in other cases, send 1 to store the positive value as it is. Multiply the values when you meet 'and' operation to find out that the values in the same line are true. The last time meet 'or' operation, it is satisfied if the value multiplied by the value of the same line is greater than zero.

B. Discussion

DNF was almost same in shape to CNF. Almost the same algorithm had to change a few variables. The process of checking satisfaction was added to this task, which was actually connected by 'or' operation, so it was easy for the proposition to be true if only one line is true. If satisfaction had to be checked with CNF, the solution

would have been much less. Conjunctions would have fewer solutions than DNF because the proposition only turns true when both p and q are true.