SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Singapore University of Technology and Design

50.001: Introduction to Information Systems & Programming

2D Group 13

Zhang Peiyuan 1004539

Qiao Yingjie 1004514

Ryan Seh Hsien En 1004669

Chua Qi Bao 1004494

Lim Hng Yi 1004289

Peh Jing Xiang 1004276

Date of submission: 26 October 2020

SATSolverTest

Firstly, we create a File object(CNF File content) that will be read by the Scanner object. Since the first few lines of the CNF are comment lines, we check if the first character of the line has a c (present in all comment lines) that will cause the scanner to skip to the next line in the CNF file - this will save time and help the skipping comment lines faster by checking the first character only. If the line starts with p, we will read that line and store the number of variables n_v and number of clauses n_c.

Then we proceed to read the clauses of this cnf file. We start by creating a for loop which will be looping n_c (number of clauses) times. For each loop, we read the next integer until we hit a '0'. Otherwise, if the integer we read is positive, we will add the literal with the Posliteral.make() method to the clause. Whereas if it is negative, we will add the negative literal with Negliteral.make() to the clause.

Once we hit an 0 it means a whole clause is read, we will add the clause into the formula object which we created. We will repeat the steps till we have added n_c clauses.

We are referring to the pseudo code provided in the handout. Firstly, we check the 2 base cases when the environment is the final solution or when there is an empty clause. Then if the smallest clause is a unit clause, we set that clause to true. If the literal inside the unit clause is a Posliteral, we set the corresponding variable to true in the environment. If the literal is Negliteral, we set the variable to false. When there is no smallest clause, we just try making the first literal from that clause true first, and reduce the formula. If this works, we can get the final environment. If it does not work, we will set the literal to false. If it still does not work, it means the formula is not satisfactory.

After invoking the SATSolver.solve(formula), if the resulting environment is null, it means that it is unsatisfiable. Likewise, if it is not null, then it is satisfiable and we will write the boolean variables to the BoolAssignment.txt which will be created automatically when the resulting environment is not null.

To write to the BoolAssignment.txt file, we iterate through the number of variables, checking its boolean value. We start by creating a StringBuilder and attach a new line for every variable. If its boolean value evaluates to 'FALSE', we append its name at the end of the file with a new line together with 'FALSE'. Otherwise we attach its name and 'TRUE'.

Result of file 'test_2020.cnf':

Result: not satisfiable
Running time: 4959.670278ms
Specification: MacBook Pro (13-inch, 2018), 2.3 GHz Quad-Core Intel Core i5