Arpit Kumar Rai – 200190
Het Hitendrakumar Patel – 200440

Date – 18/02/2022

Assignment – 2

Source Code: SAT_solver.cpp

The C++ code involves the use of the library “bits/stdc++.h” and using basic STL classes like vector and set.

Data Structures Used:

1) The vector “model” is used to represent a Model of the formula.

2) The vector “assume” is used to keep track of the currently assigned literals.

3) The vector “literal_count” is used to store the frequency of each literal in the formula in a sorted manner.

4) The set “modelSet” stores the literals of the current model.

5) The vector “clauses” is used to represent the formula in CNF.

checkLiteral():

A function used to check whether a literal is present in the current model.

For this we use the “.find()” function which searches for the integer in the set “modelSet”. If it returns “modelSet.end()” then literal is not present in the current model.

checkSAT():

A function used to check if the formula is Satisfiable on the given model.

For this we check if there is at least one literal in every clause that is True. We run a loop on “clauses” and use “checkLiteral” at every iteration to check whether a literal is present in the current model.

checkUNSAT():

A function used to check if the formula is Unsatisfiable on the given model.

For this we check if there is at least one clause in which all the literals are False. We run a loop on “clauses” and use “checkLiteral” at every iteration to check whether a literal is present in the current model.

unitPropagation():

A function to perform unit resolution in a given formula. It considers the current model and finds all the clauses in which exactly one literal is unassigned and the rest of the literals are assigned False. It then concludes the unassigned literal to be True for the formula to remain satisfiable. This process is recursively continued until no more unit literals are present.

The function also checks for satisfiability of the current model and returns if the formula becomes satisfiable on the current model.

SATsolver():

This is the main function that solves the Satisfiability of a given formula using the above functions.

Firstly it finds the literal that is unassigned and has maximum frequency (as the vector “literal_count contains the literals in a sorted manner).

Now it assumes the above literal to be assigned True. This is followed by checking if the previous assumption makes the formula Unsatisfiable. If it does not make the formula Unsatisfiable we apply “unitPropagation” with the current model. We make recursive calls to the function “SATsolver” until we obtain a Model or conclude that the formula is Unsatisfiable.

Lastly we assume the above literal to be assigned False. This is followed by checking if the previous assumption makes the formula Unsatisfiable. We apply same procedure as in the previous step here.

Assumptions:

1. The tester has a working system with C++ compiler installed along with following libraries: “bits/stdc++.h”

Limitations:

“SAT_solver.cpp” takes approximately 200 seconds for number of variables equal to 150 (whose output is UNSAT).