CS202A
SAT SOLVER

Definition of a sat-solver : In computer science and formal methods, a SAT solver is a computer program that aims to solve the Boolean satisfiability problem. On input a formula over Boolean variables, such as "(x or y) and (x or not y)", a SAT solver outputs whether the formula is satisfiable, meaning that there are possible values of x and y which make the formula true, or unsatisfiable, meaning that there are no such values of x and y. In this case, the formula is satisfiable when y is true, so the solver should return "satisfiable".

Implementation

Regarding our problem statement, we have used the DPLL algorithm and also processed our input clauses at various stages.

Taking Input/Output:

Taken input from *.cnf files in the ./test/cnf folder. And stored the clauses at each stage as a vector of clauses in which each clause is a vector of literals.
These input and output files can be changed from the code.
DPLL is the major crux for our assignment, and it becomes most important to understand its functionality. Learning about it was our first task.: the Davis–Putnam–Logemann–Loveland (DPLL) algorithm is a complete, backtracking-based search algorithm for deciding the satisfiability of propositional logic formulae in conjunctive normal form, i.e. for solving the CNF-SAT problem."
In simple words we keep on assuming a proposition as true or false and check if satisfiability is possible after making such recursive assumptions.
We implement this by using trees which store a vector having the truth values of all the propositions.

Soundness

The termination of this algorithm always holds since the number of
variables are being reduced in every recursive call.
If satisfiable is returned, then all involved unit clauses yield a satisfying
assignment. Otherwise, it is a big case analysis yielding contradiction for all cases. Hence Unsatisfiable.
So DPLL is a Complete method to establish satisfiability.

DPLL Algorithm Properties

→ DPLL is complete, correct, and guaranteed to terminate.
→DPLL constructs a model if one exists.
→ In general, DPLL requires exponential time
→In all SAT competitions so far, DPLL-based procedures have shown the
best performance.

Our implementation of this sat solver incorporates the following properties too while execution:

→ ***Input Optimisation***
Any clause having a proposition and its negation is not inserted in the list of clauses as it is true irrespective of valuations of other propositions. If a clause contains a proposition multiple times then it is added to the list only one time.

→ ***Pre-Processing***

If a clause contains a single proposition, then the value of the proposition is assigned as that has to be true for the formula to be satisfiable. The clauses which contain such propositions are removed from the list of propositions and the clauses containing negation are removed from the rest of the list of the set of clauses. They are the clauses that contain literals of the form 'a' and '-a'. They have to be true due to the binary nature of a proposition.

→ ***Unit propagation***

When we assign a proposition true then clauses having that proposition are removed from that list. Also, the propositions of size 2 having negation of the proposition automatically assign value to other propositions. So again, we can remove some clauses from the list and again get values to some of the propositions. So, this step is recursively done unless the list gets exhausted, a contradiction occurs, or we don't have an assigned proposition to propagate.

→ ***Weightage function***

We have assigned extra weightage to the propositions occurring in clauses of size 2 as they automatically assign value to other propositions or make a proposition true during further solving. So we calculated weighted frequency for each element as 2*(frequency of the proposition occurring in the clause of size 2) + frequency of the proposition. This helped choosing the proposition which simplifies the greatest number of clauses at the same time during the splitting of tree into right and left trees.

→ ***Deterministic Frequency***

Once the maximum frequency of all the clauses in the list becomes 1 then we need not take decisions anymore and we assign values to the propositions to make them true if possible or return UNSAT if we already determined values of propositions do not satisfy the list of literals.

References

https://en.wikipedia.org/wiki/Unit_propagation

https://en.wikipedia.org/wiki/SAT_solver

https://en.wikipedia.org/wiki/DPLL_algorithm