SAT Solver

By: Mohit Gupta(200597),Naman Singla(200619)

Introduction :

The satisfiability problem for propositional logic is to decide whether a propositional logic formula φ is satisfiable. A SAT solver is a tool to solve the given set of formulas to generate a satisfiable solution, if possible, or return none if the formula is unsatisfiable. Here we take the formula in cnf format.

Aim :

To design a SAT solver function which works as follows

```
Function solve ( [constraints] ) {
        Set of operations
        return a model if SAT or return none if UNSAT
}
```

Here the input constraints are provided in DIMACS SAT format.
DIMACS SAT format is a standard text format for CNF formulae.

Eg. :
```
c
c start with comments
c
c
p cnf 5 3
1 -5 4 0
-1 5 3 4 0
-3 -4 0
```

Algorithm :

We have used the DPLL algorithm to solve the SAT problem. The algorithm is as follows:

```
SAT (Formula F, Interpreation I):
        if ( I ⇒ F ) return true
        if ( I ⇒ ¬F ) return false
        F,I = unit_propagation(F,I)
        if I is inconsistent return false
        F,I = pure_literal(F,I)
        if F = ∅ return true
        choose the best xi that I does not assign
        if sat(F, I ∪ { xi=true }) return true
        if sat(F, I ∪ { xi=false }) return true
        return false.
```

The basic eliminations used are :

1. If x_1 = x_2 = … = x_{n-1} = false and one of the clauses is (or x_1, x_2, … , x_n), then we infer that x_n is true.
2. If x_1 = false, one of the clauses is (or x_1, x_2, … , x_n) then we modify the given clause to (or x_2, … , x_n).
3. If x_1 = true then we drop all clauses containing x_1 independent of other propositions in the clauses.
4. If one of the clauses is (or x_4) i.e., a unit clause, then we infer that x_4 is true and drop the clause.(Unit propagation).

Once this is done we used the DPLL algorithm:

1. We assume one of the unassigned propositions as “true”.
2. Then proceed with the above algorithm and again assume further propositions recursively until we either reach zero clauses or a contradiction.
3. In case of contradiction,backtrack and change the last assumed value to “false” and rerun the process.
4. In case of zero clauses, the present state becomes the required modal for satisfiability.
5. If for all possibilities we reach a contradiction is achieved, the given cnf formula is unsatisfiable.

Assumptions :

1. Given constraints/ formula is given in cnf format.
2. All cnf files are present in the same testcase folder.

Limitations :

1. It takes a lot of time as the size of clauses or the no. of propositions increases.