CS202 Assignment-2 Report

Udit Prasad & Samarpreet Singh

SAT Solver using DPLL algorithm (in C++) -

Firstly, we have read input from the file containing the formula in DIMACS CNF format, and from it, we obtain a list of clauses present in it (we are storing it in the form of 2d vector).
Then we've sorted each clause in increasing order (for using binary search on the vector(optimisation)).
For optimizing the DPLL implementation, we have used the concepts of

- pure literal elimination
- unit propagation

For these , we have used functions

- pure_elimination() - for finding literals occurring only in negated form or non-negated form in every clause.
- unit_propagation() - for checking the unit length clauses beforehand, for optimisation.

The function solver() takes the list of clauses as a parameter and returns if the formula is satisfiable or not. For this, we have used the rules defined in dpll algorithm:
(i) Empty formula implies SAT.
(ii) Any individual empty clause implies UNSAT.

For further optimization, we take smallest length clauses first using smallest_clause() function, and check which literal in it is occurring in maximum number of clauses with the help of "which_literal_to_use()" function, and choose that literal for proceeding with the algorithm.

(iii) For every clause, we remove it from the formula if the selected literal is present in it.
(iv) If its negation is present in any clause, we just delete its occurrence from that clause.

(iii) & (iv) are implemented using "reshape_clauses()" function.

(v) Now we evaluate with backtracking if the CNF formula is satisfiable either by adding this literal or its negation in it, else we return UNSAT. For the SAT case, we add the required literal in the list named model (vector declared globally).

There might be some don’t care cases as well, whose valuation doesn’t have any effect on satisfiability. Their non-negated form is assigned TRUE.

In this way, we have implemented the whole SAT solver.