Implementation for SAT Solver

Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

DPLL Algorithm Rules

Data in this slide is referenced completely from :

https://ethz.ch/content/dam/ethz/special-interest/infk/chair-program-method/pm/documents/Education/Courses/SS2018/Program%20Verification/01-SATSolving.pdf

- If A is T then return (sat, M)
- If A contains an empty clause ⊥ then return unsat
- *Pure Literal Rule* : If p occurs only *positively (negatively)* in A, delete clauses of A in which p occurs, update M to M∪{p} (to M∪{-p})
- *Unit Propagation*: If 1 is a *unit clause* in A:
  - update M to M∪{1}, and
  - remove all clauses from A which have 1 as disjunct, and
  - update all clauses in A containing ~1 as a disjunct by *removing* that disjunct.
- *Decision*: If p occurs both *positively* and *negatively* in clauses of A:
  - Apply the algorithm to (M∪{p}, A∧p): if we get (sat, M′) then return this
  - otherwise, apply the algorithm to (M∪{–p}, A∧ –p) and return the result
  - Here, p or –p are called *decision literals.*

Our Implementation Step-1

Processing the data

- Number of occurrence of a particular literals were stored and also the difference in the frequency of positive and negative occurrences.
- In order to uniquely identify the positive and negative occurrence of the literal, the positive one was stored in the form of 2n and negative one was stored as 2n+1, where n is the literal, which also helped while assigning the values to these literals.
- A particular literal was accessed using the index value.

Our Implementation Step-2

Functions used in the implementation

1. apply_transform(formula F , literal_value)
   - This function is used to make changes to the entire set of clauses as per the value assigned to a single literal.
   - If a particular clause has the literal (whose value has been assigned) with same polarity then that entire clause is removed from the formula.
   - If the literal occurs in opposite polarity then that particular entry from the clause is removed only.
   - At any stage if the formula gets empty we return sat, or if one of the clause becomes empty we return unsat.
2. unit_propagate(formula F)
   - It searches for a unit clause in the formula. If it finds a unit clause, then that particular literal is assigned the value according to its polarity and apply_transform function is called to make changes in the entire formula.
   - At any stage if the formula gets empty we return sat, or if one of the clause becomes empty we return unsat.

Our Implementation Step-2

Functions used in the implementation

- DPLL (formula F)
  - This is a recursive function implementing the complete DPLL Algorithm using the previous two functions.
  - It firstly calls for unit_propogate and then checks for its result. If the result is sat, we return and show the model or if it is unsat we return unsat.
  - Then it chooses the decision literal , in this case we have chosen the decision literals to be the literal with maximum occurrence. The value is assigned to this literal (if positive occurrence is more, then firstly true is assigned and vice versa). Then apply_transform function is called to extend this assignment to all the clauses. If this value does not result in any conflict(we do not get unsat from the apply_transform function) we find next literal and then recursively follow this step to form a branch until we get our result. If the assignment of a particular value results in conflicts, we assign it other value and then follow the recursion.
  - The function returns (sat, model) only if the formula gets empty, that is all the clauses gets erased, otherwise it returns unsat in all other scenario.

Assumptions and Limitations

- Assumptions:
  - The input formula needs to be given in DIMACS format.
  - The input needs to be given to the input terminal instead of reading the input from the CNF file.
- Limitations:
  - Takes some time for big input formulas, that is, with large number of literals and clauses.