SAT Solver using DPLL Algorithm

Vedasree 201049
Jahnvi 200467

February 18, 2022

Introduction

We have used the DPLL algorithm for our SAT Solver. It is a backtracking based algorithm, that is, it tries to build a solution on assumptions till it reaches a point where no solution could be valid. There, it "backtracks" and goes forward with some different assumptions.

In our case, the DPLL algorithm, after removing as many clauses as possible by finding the "fixed" literals (literals whose value must be True or False), assumes a literal to be True and solves the new cnf. If unsatisfiable thus, it assumes the same literal to be False and solves this new cnf.
The choice of this literal is what determines the efficiency and runtime of a DPLL algorithm.

Implementation

Given a CNF formula in the DIMACS representation, we store the clauses in the form of a list of lists. Each row in the list is a clause, the elements of these rows are the literals of clauses.

Main function - DPLL

- This function takes the clauses and fixed literals list as input.
- It finds the unit literals (*clauses with single literal*) and stores in the list units.
- Now, it runs a for loop through all literals in the list units.
- Each iteration of for loop calls unit_propagate that removes all clauses with the literal and all occurrences of the literal -l from the clauses of the cnf.
- Now it checks for the pure literals (*Using function find_pure_literals*) in cnf and assigns True to all the pure literals (*Using function pure_literal_assign*).
- Now, if the output cnf has no clauses, we have no conditions to satisfy and hence, a solution exists and DPLL returns true.
- If the length of any clause is zero, we can do nothing to satisfy that clause and DPLL returns False.

- Since we assigned the semantics to pure literals and units (*which should be true*) now we choose a literal from remaining clauses and call DPLL assuming it is true. If this function returns true, then we return true.
- If this function returns false, we call DPLL assuming the value of this random literal False. If the DPLL is True, there exists a model. So, we return true and the model. Else, we return False and the cnf is Unsatisfiable.
- The literal chosen for the aforementioned purpose is the one that occurs the most number of times in the cnf. Choosing this literal to be True or False (appending it as a supposed unit literal) ensures the maximum number of deletions of clauses in the unit_propagate functional step. Thus, we have ensured that the code is optimised.

Function - find_pure_literals

- This function takes clauses as input and outputs a list pures that has pure literals.
- We create two lists each of size equal to number of literals.
- The first list stores value 1 if literal appears in clauses and the second list stores 1 if not of that literal appears in clauses of cnf.
- We traverse through all the clauses using for loop until we find the literal l. If we find the literal we break the loop and assign value 1 for the l*th* member of first list, else the value stays 0.
- Similarly , We traverse through all the clauses using for loop until we find the literal not of l. If we find the not of literal we break the loop and assign value 1 for the l*th* member of second list, else the value stays 0.
- if both l*th* member of first and second list are 1 it is not a pure literal. Else it is and we append it to list pures.

Function find_unit_literals

- This function takes clauses as input and updates the list fixed(*contains literals that are unit literals*)
- This function loops through all the clauses and checks if the length of clause is 1 or not
- If the length of clause is 1 and the literal is not in the list of units we append literal to the list units.

Function unit_propagate

- This function takes the clauses, literal to be used to propagate as input.
- It runs for loop through all the clauses in cnf and removes rows or clauses with literal l and removes the literal -l from the clauses that has not l as literal.

Function pure_literal_assign

- This function takes the clauses and the list of pure literals as input and updates the clauses and "fixed" literals as it is run.
- A literal is fixed when its truth value is decided for the cnf to be satisfiable. By this logic, all pure literals are fixed.
- A for loop on "pures" (list of pure literals) is run and their truth value is stored in ’fixed’.
- We introduced a new temporary list because iterating on a list while editing it leads to erroneous results.
- Any clause containing a pure literal is appended to this list. All these clauses are then removed from the cnf as they are satisfied as soon as the pure literal is set.