CS202A Assignment 2

Aryan Vora 200204
Dishay Mehta 200341

18^{th} February 2022

1 Introduction

We have implemented a SAT Solver using the Davis–Putnam–Logemann–Loveland, henceforth called DPLL, algorithm. We have implemented the same in C++. The task is to take a CNF Formula from a file in DIMACS format and return whether a model exists for it or not, if it exists, we must return the model too.

2 Reading The File

We first read the formula and get it into a normalized form in a 2D array. Each row in the array corresponds to a clause in the formula and in that array, if the value at the i^{th} index is 1/-1 then it means that x_{i+1} is in the clause in it's positive/negated form respectively. If it is 0, then the corresponding term does not appear in that clause.

3 Searching For A Model

In it's simplest form, the algorithm can be thought of as a backtracking algorithm in which we first set x_i to be True in the Partial Interpretation and try to find a model, if we can't find one, then we set it as False in the Partial Interpretation and try the same. If we can't find an interpretation using these two, then we can surely say that the formula is unsatisfiable. This is implemented using recursion where we try to set each X_i to True or False.

4 Optimizations

It can be clearly seen that the above is a sure-shot method to find a model if it exists, but it is very time-consuming in the raw form. We introduce some optimization techniques to improve the speed.

4.1 General Optimizations

We can see that if a clause has only one unassigned literal then it is True then we can just ignore that entire clause in the formula as it is already True.

4.2 Unit Propagation

If there is a clause which has only one unassigned literal, then that literal can be assigned a value immediately such that the clause becomes True.

4.3 Pure Literal

If a proposition occurs with the same sign in all the clauses in which it occurs, then it is called *pure*. We assign the pure literal the value such that all clauses containing it become True. Hence, we can ignore all the clauses in which the literal occurs.

4.4 Choice of x_i to branch

We choose an x_i such that the number of clauses in which x_i occurs in it's negated form is close to the number of clauses in which it occurs in it's positive form.

5 Running The Solver

Refer to the README file for instructions to run the solver.