CS402 Introduction to Logic for Computer Science

# DPLL-based SAT Solver Implementation Summary

20160042 Inyong Koo

## 1 Introduction

The *Davis-Putnam-Logemann-Loveland (DPLL)* algorithm is a procedure that combines search and deduction to decide the satisfiability of CNF formula. The objective of this project is to design a DPLL-based SAT solver and implement it using python.

I implemented the *conflict-driven clause learning (CDCL)* algorithm, which is the improved version of DPLL algorithm. The main idea of CDCL is explained with detail on the course lecture note (Lecture 6). My implementation follows the DIMACS input/output requirements. Though I tried to handle errors of the file format, I assumed that all my test cases are in valid CNF format.

My SAT solver is a single file(`solvepy3.py`), written in Python 3.7. No other module from Python standard library was imported.

## 2 Program

Please refer to attached python file and its comments. I will only address some important code lines in this summary.

### 2.1 Classes (Line 14 - 56)

I defined two classes: the clause set and the assignment.

The clause set class contains the given formula and the number of variables. I sorted variables by index for each clause, and clauses by the number of variables for the formula. This is due to my decision strategy.

In assignment class, I categorized assignments into three seperate groups. The decision assignments are the assignments fixed by unit propagation. If there is a conflict within decision assigments, the formula is unsatisfiable. The shoot assignments are assignments added by decision policy. They might cause conflicts, and go through learning procedure. The implied assignments are assignments acquired by unit propagation after assigning some shoot assignments. The implied assignments are stored with the index of clause where it was derived.

Now we will look some important parts of the program: main scripts, `DPLL` module and its three submodules - `UnitPropagation`, `ClauseLearning`, and `DecisionStrategy`.

### 2.2 Main Scripts (Line 219 - 234)

Program read and `ParseCNF` module parses CNF format file into a clause set. The clause set contains the given formula and the number of variables. `DPLL` module receives the clause set input and find a satisfying assignment. If such assignment exists, program prints out "`s SATISFIABLE`" and the assignments in lines starting with a character "`v`". Otherwise, it prints out "`s UNSATISFIABLE`".

In assignment class, I categorized assignments into three seperate groups. The decision assignments are the assignments fixed by unit propagation. If there is a conflict within decision assigments, the formula is unsatisfiable. The shoot assignments are assignments added by decision policy. They might cause conflicts, and go through learning procedure. The implied assignments are assignments acquired by unit propagation after assigning some shoot assignments.

### 2.3 DPLL Module (Line 198 - 217)

Starting with the empty assignment, conduct unit propagation to logically acquire assignments. If the assignments satisfies the formula, return the satisfying assignment. If the formula is not satisfied, it means there are conflict or undecided clauses.

If there is any conflict clause, check if the conflict clause makes the formula unsatisfiable. If the conflict clause is composed of only decision variables, the formula is unsatisfiable. If not, conduct learning procedure to backtrack some shoot assignments and add a learned clause.

If there isn't any conflict clause, we follow the decision strategy to make a new shoot assignment.

### 2.4 Unit Propagation Module (Line 125 - 139)

A clause with single variable, or a unit clause, is decisive, since we can determine the value of the variable to make the clause true. We search for unit clauses in the formula, and make assignments for the literals. (If the formula is under assignment of shoot assignments, the induced clauses are not reliable. Make an implied assignment. Otherwise, make a decision assignment.) Formula after assignments may create more unit clauses, so repeat (propagate) the procedure until there are no more unit clauses.

### 2.5 Clause Learning Module (Line 159 - 173)

The conflict clause contains wrong implied and shoot assignments. (Otherwise, the conflict clause is composed of only decision assignments, and the formula is unsatisfiable.) First, remove implied variables from the conflict clause using resolution.

Then the resolved clause will only have shoot variables. we can backtrack shoot variables to satisfy the 'learned' clause. Keep only shoot variables existing in the learned clause, except one. The remaining one will be implied to satisfy the learned clause.

The learned clause will be added to the clause set at the DPLL module.

### 2.6 Decision Strategy Module (Line 187 - 196)

The decision strategy is simple. Find first occuring unassigned variable from the clause set. Since clause set is 'sorted' to have clauses with small number of variables at front, the decision strategy will gaurantee us to acquire an implied assignment in minimum number of shoot variables. This is efficient because when a conflict occurs, we won't have to backtrack too much. A conflict may be resolved in relatively short time.

## 3 Correctness

The correctness of my SAT solver was verified through some of the benchmarks from SATLIB - benchmark problems. I used some DIMACS benchmark instances such as AIM and JNH.

My concern is on time efficiency though. My SAT solver could not determine formula with more than 200 variables in 5 minutes. I wonder if there is any way to make my SAT solver more efficient.