Assignment of CS202

SAT Solver using DPLL

Abhishek Pardhi, Akshat Garg

200026, 200084

B.Tech student

`apardhi20@iitk.ac.in, akgarg20@iitk.ac.in`

Indian Institute of Technology
Kanpur

Department of Computer Science & Engineering

February 18, 2022

Contents

1 Question 1

1.1 *DPLL* Algorithm

We've used ***Davis–Putnam–Logemann–Loveland (DPLL) algorithm*** to check the *satisfiability* of any given set of formulas. The algorithm comprises of four main parts:

- **Unit Propagation:** In here we look out for unit clauses throughout the set of clauses. We execute this part of code as many times until we are left with no unit clause. If there's a literal **p** in an unit clause then we remove all those clauses which have **p** as a literal in it or we just remove the literal ¬**p** from clauses having ¬**p** as a literal.
- **Pure Literals Elimination:** If a propositional variable occurs with only one polarity in the formula, we can safely assume that literal to be true and remove all those clauses which contains this literal.
- **Tautology Removal:** We remove all those clauses which contain both **p** and ¬**p** as literals.
- **Set of literals is empty:** We are assured that it is SAT and return 1.
- **Empty clause:** If there exit any empty clause in our set then we are assured that it is UNSAT and return 0.
- **Maximum Occurrence:** We first find the variable which occurs the most in our set and then we add **p** to our set of clauses if number of positive literal occurrences of that variable is more otherwise we add ¬**p**.

Then we execute this code recursively until we return **SAT** or **UNSAT**. Also, we kept updating the ***model*** and at the end we assigned it to the ***finalModel***.

1.2 Functions

The functions which we have used in our code are as follows:

1. ***readInput()*** - We first took input from a **cnf** file. We have made a list of lists **x** which is our set of clauses and **y** is a clause list which contains integers as its elements. **n** and **l** are number of variables and number of clauses respectively.
   *This function returns set **x**, number of variables **x** and number of clauses **y**.*
2. ***maximumOccurence()*** - We made two arrays, ***pos*** and ***both*** which stores the number of occurrences of positive literals and both positive & negative literals respectively. We then first check which variable occurred most frequently and then returned positive literal if occurrence of positive literal(***y***) of that variable is more than or equal to the negative literal otherwise we returned the negative literal(***-y***).

3. ***DPLL()*** -
    - **Unit Propagation:** We ran a while loop keeping condition on while loop 1 and we break the *while* loop only when there isn't any unit clause left. Inside this while loop we ran two *for* loops, in the first *for* loop we find all those variables for which there exist a unit clause in our set.
    - **Pure Literals Elimination:** We kept an array which denotes the presence of a variable in the *cnf.* For a variable (let $\boldsymbol{x}$) we assigned -1 in the array if it is not present in any clause. Then using for loops, we assigned 1 if $\boldsymbol{x}$ is present or 0 if $\neg\mathbf{x}$ is present. We checked if $\boldsymbol{x}$ and $\neg\mathbf{x}$ are both present in the *cnf* input, if so, we assign it 2. So, finally if a *cnf* input contains only $\boldsymbol{x}$: it is assigned 1 in the array, if it contains only $\neg\mathbf{x}$, then it is assigned 0 in the array, otherwise it is assigned 2. A literal which is NOT assigned 2 is called ***pure literal***. Next using for loops, we checked all the clauses containing pure literals and removed those clauses and added the pure literal to our model.
    - **Set of literals is empty:** Simply return *True* if all clauses are empty so it will be satisfied by any assignment of variables.
    - **Empty clause:** For a clause to be true, it requires at least one literal to be true and so an empty clause has no literal to make it true. Since the clause is false, it returns ***False***.

1.3 Note

Change the line number 5 of the code according to your file directory/name.