Programmer's Documentation Guidelines

Created for classes C++ at MFF UK.

1. Description

SAT solver solves propositional formulas and decides, if they are satisfiable.

2. Tools

I used Visual Studio 2019, OS Windows 10 64-bit.

3. Data structures

There is diagram of structures, which I used in application. You can see structure of cnf formula in figure 1. There is vector of variables. These variables are represented with class Variable. , There is vector of literals, because there are negative and positive literals in clauses. Each variable has two literal observers, positive and negative. These literal observers provides interface to work with variable(setting assignment, getting assignment etc..). These vectors are created and filled at begining, so iterators, pointers and references stay same to the end. Clauses are solved in the same way. There is deference, how I store clauses. It is possible to need to add new clause during calculating. I used own resizable array for it, It does not change place of things in memory, when i add new item.

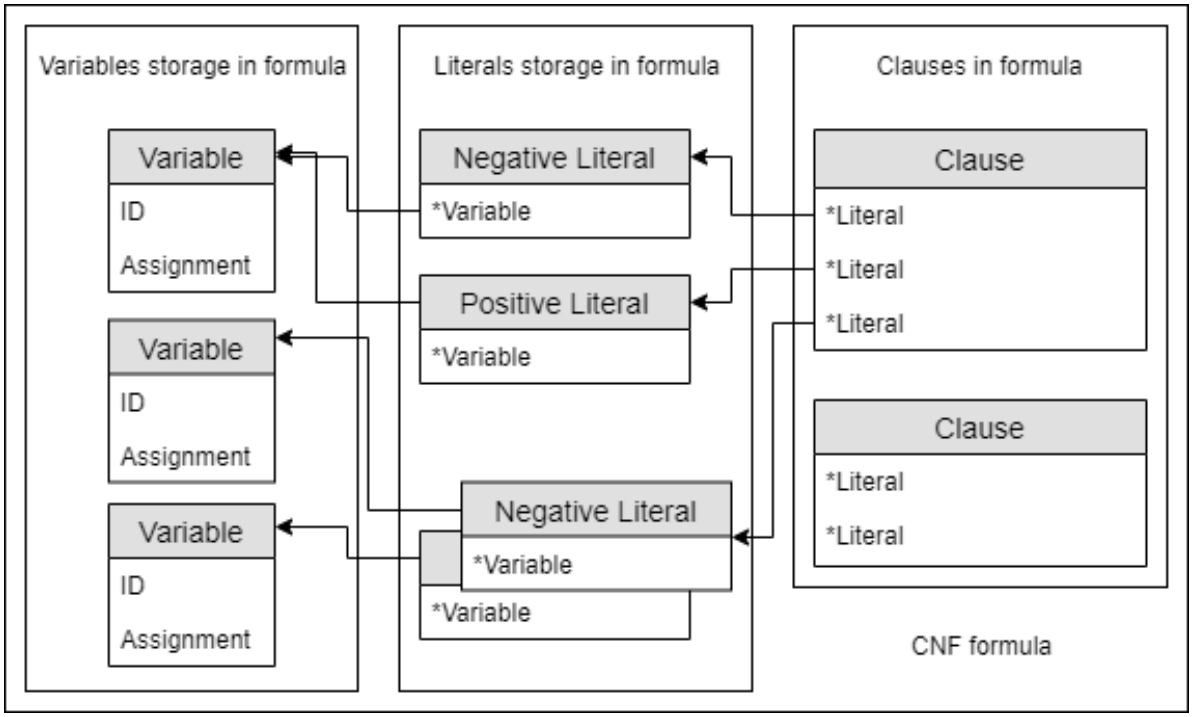

figure 1 - cnf formula

I was solving, how to store information about decisions, which I had made.I decided to make structure called Level. There are information about ID, decision, which I made and literals, which I assigned because of unit propagation. For better performance I have something like invertible index of variable. There is information, in which level variables were assigned. Next structure is

implication graph. For better deletion of vertices during backtrack, I store vertices in level as well. Each literal has own vertex. It is easy to do cut, because of orientation of edges.

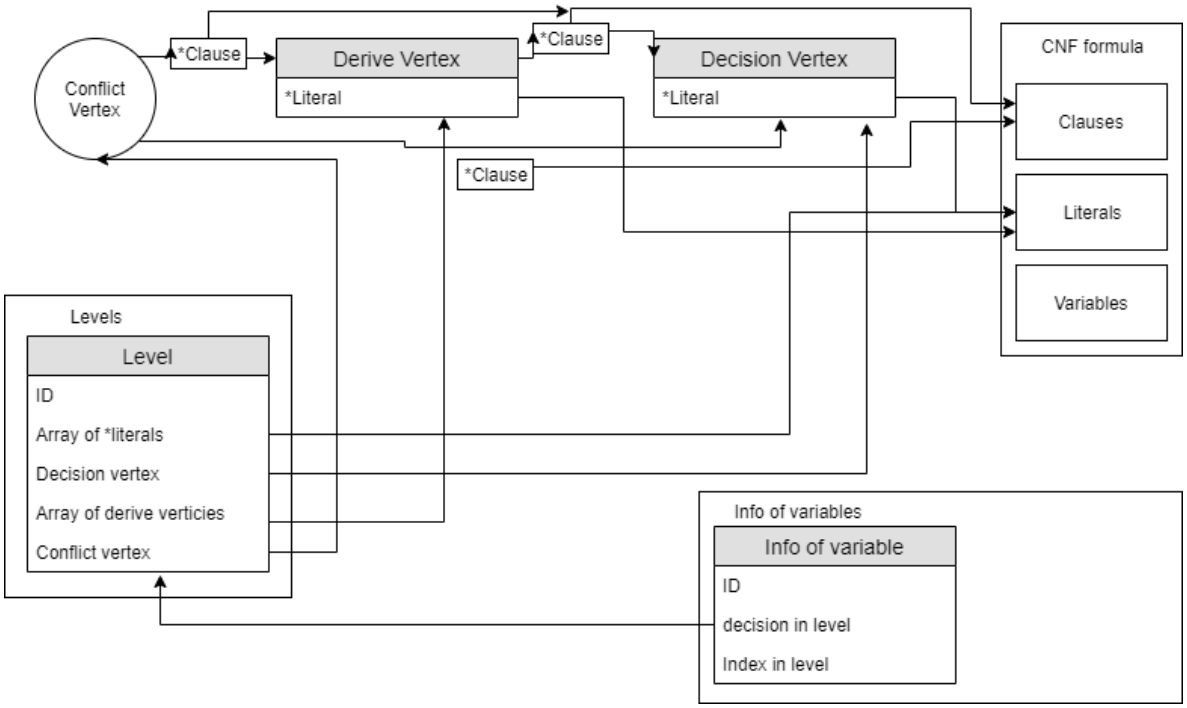

figure 2 - cdcl algorithm

4. Algorithms

I used standart CDCL algorithm with learning asserting clauses.

5. Architecture

Main method is **make**. It connects all parts of algorithm. For first step, **unit_propagation** is called. It finds all unit clauses and set assignments. Second step is **make_decision**. This method picks random variable, sets assignment and creates new decision level. After this, next **unit_propagation** is called. There are helpers for check, if clause has conflict etc.. but names of method describes functinality of it. So if there is conflict, conflict vertex is added into current decision level. When I add new vertex in the level, I must connect it to all verticies, where literals are in clause, which has conflict. After conflict, **analyze_conflict** is called. It finds asserting clause and level, where algorithm will backtrack. **Backtracking** is simple deletation of levels and unassigning variables. This process is in loop until there is no variable with undefined assignment or all possible choices for assignment are checked.