Programmer’s Documentation Guidelines

This document is the intellectual property of the author.

Created for classes C++ at MFF UK.

## Description

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

## Developers features

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

## Data structures

There is diagram of structures, which i used in application. There is vector of variables. These variables are represented with class Variable. Because there are negative and positive literals in clauses, there is vector of literals. 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 at begining, so iterators, pointers and references stay same to the end. Clauses are solved in same way. There is deference, how i store clauses. Becauses durring calculating, it is possible to need to add new clause. I used own resizable array for it, It does not change place of things in.

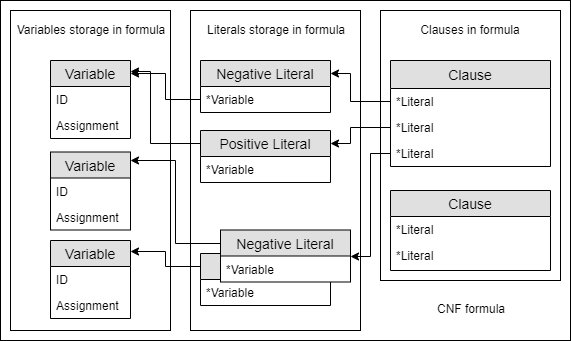


figure - cnf formula

I was solving, how to store information about decisions, which i had made. So I made structure called Level. There is 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 are information, in which level variables were assigned. Next structure is implication graph. For better deletion of vertices during backtrack, i store verticies in level as well. Each literal has own vertex. It is easy to do cut, because of oriantion of edges.

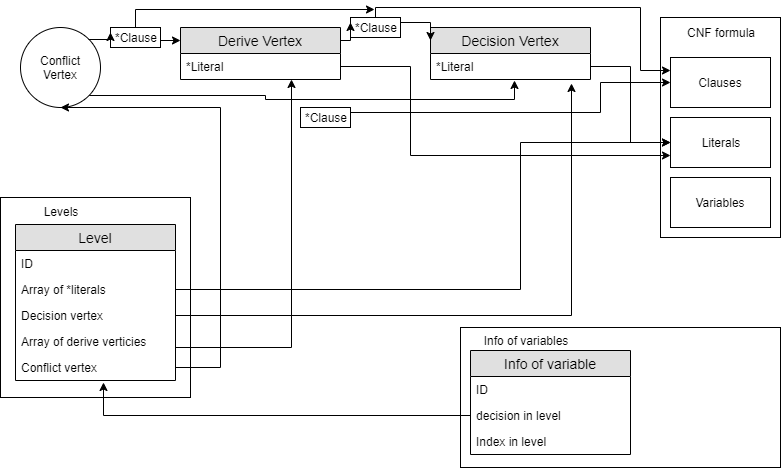


figure - cdcl algorithm

## Algorithms

I used standart CDCL algorithm with learning asserting clauses.

## Architecture

Main method is **make**. It conects 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 pick random variable, sets assignment and create 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. Whene it adds new vertex in level, it must connect to all vertex, 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.