Software Analysis and Verification - Fall 2025

Ruzica Piskac

November 4, 2025

The deadline for Project 3 is November 20, 11:59PM.

You are encouraged to work with a partner. There is no starter code provided for this assignment. You may use any programming language that you wish.

1 SAT Solver

Implement the basic DPLL Algorithm for SAT. Your program should accept input in the CNF version of the DIMACS format. The input file is a formula in conjunctive normal form, where each clause is given on a separate line as a list of indices denoting propositional variables, where a negative index indicates a negated propositional variable.

If a given formula is satisfiable, your SAT solver should produce `SAT` and an assignment to all variables such that the formula evaluates to true under this assignment. If a given formula is unsatisfiable, your SAT solver can simply produce `UNSAT`.

This is a group project. There are various extensions to the basic algorithm that make SAT solvers scale well in practice. In the lectures, you were given pointers to relevant literature describing many of these extensions. Your SAT implementation should contain at least n extensions, where n is the number of people in your group. Since your goal is to win the SAT competition, try to figure out which extensions work best in practice. Of course, for bonus points you are welcome to implement more extensions. For every extension that you implement, you should be able to describe what this feature does and how it improved the original DPLL algorithm.

Here are some links that you might need:

- "Conflict-Driven Clause Learning SAT Solvers", from the Handbook of Satisfiability, `http://gauss.ececs.uc.edu/SAT/articles/FAIA185-0131.pdf`
- The SAT Competition is a good source of benchmarks. However, you may need to back a couple of years to find ones small enough that your solver can handle: `http://www.satcompetition.org`
- DIMACS format: `http://www.satcompetition.org/2011/format-benchmarks2011.html`