15-354 Final Project: SAT Solver

Fan Pu Zeng
fzeng@andrew.cmu.edu

Boolean satisfiability (SAT) solvers have played an important role in software and hardware verification, automatic test pattern generation, planning, scheduling, and solving challenging problems in algebra [1]. This report will introduce my SAT solver, its implementation details, designs and algorithms used, some comparisons between using different heuristics for splitting and choosing of variable assignments in the unforced case, challenges faced, and future directions.

1 SAT Solver Overview

The SAT solver uses the classical Davis–Putnam–Logemann–Loveland (DPLL) algorithm, which is based off backtracking search. In order to speed up the search process, unit propagation (also known as boolean constraint propagation), and the usage of watchlists for ease of backtracking is also implemented. Both of these will be discussed in detail later.

2 Project Files Overview

An overview of the project files, and how to run them is given in this section to whet the appetites of the reader.

- `sat.py` This is the entry point of the project, and this is also where the DPLL algorithm is implemented in the SATSolver class
- `loader.py` This contains the Loader class that loaders a `.cnf` file into our proprietary CNF format
- `lib.py` This contains the definitions for the classes representing various SAT objects. Most notably, we have:

SAT solvers This report will talk about the design and algorithm, challenges faced, observations and takeaways, and future improvements on my implementation of a SAT solver.

3 Overall Design

References

[1] Frank Van Harmelen, Vladimir Lifschitz, and Bruce Porter. *Handbook of knowledge representation*. Elsevier, 2010.