SAT Solver

Functional Programming
Instructor: Prof. Amey karkare

Outline

Abstract

- Aim is to solve SAT problem given in DIMACS CNF format.
- We perform Exhaustive DPLL to perform model counting as well as list out all the solutions
- We also provide a satisfiability check using DPLL as well as DP algorithm.

Tools Used

Alex

- Alex is a tool for generating lexical analysers in Haskell, given a description of the tokens to be recognised in the form of regular expressions. It is similar to the tool lex or flex for C/C++.
- We have used alex lexer to identify the start of formula, literals, end of clause, white spaces, etc.

Happy Parser

- Happy is a parser generator system for Haskell, similar to the tool 'yacc' for C. Like 'yacc', it takes a file containing an annotated BNF specification of a grammar and produces a Haskell module containing a parser for the grammar.
- We have used happy parser to parse the cnf formula given in Dimacs form.

SAT Problem

- A propositional formula ϕ is satisfiable iff there exists a substitution of truth values for its variables that makes it true.
- The SAT problem is to, given a formula ϕ, either find such a satisfying assignment or prove that none exists and thus that ϕ is unsatisfiable

DP (Davis Putnam) Algorithm

- The Davis-Putnam algorithm was developed by Martin Davis and Hilary Putnam for checking the validity of a first-order logic formula using a resolution-based decision procedure for propositional logic.

Algorithm

- Perform Unit propogation.
- Perform Pure literal Elimination.
- Perform Resolution Propagation, select a variable (propositional symbol) and add all resolvents over that variable.
- Iterate until there are no variables left.
- If you end up with no clauses left then the original set is satisfiable.
- If you end up with empty clause then the original set is not satisfiable.

Resolution Propagation

Resolution Propagation

- Given a literal Resolution Propagation gives the resolvents and add them to the original formula.
- For two clauses (C OR p) and (D OR (NOT p)) the resolution on p yield the resolvent (C OR D).
- The heuristic used for getting the most efficient literal for resolution propagation is Dynamic Largest Individual Sum (DLIS).

DPLL Algorithm

- Davis–Putnam–Logemann–Loveland (DPLL) algorithm is a complete, backtracking-based search algorithm for deciding the satisfiability of propositional logic formulae in conjunctive normal form, i.e. for solving the CNF-SAT problem.

Algorithm

- **To check for satisfiability:**
- If empty clauses are present, return False
- If consistent set of clauses are present, return True (corresponds to satisfying assignment)
- Perform unit propagation for every single literal clause recursively.
- Perform Pure Literal Elimination recursively.
- Choose a literal randomly or by using some heuristic
- Assign a truth value to it, simplify the formula and then recursively check if the simplified formula is satisfiable; if this is the case, the original formula is satisfiable; otherwise, the same recursive check is done assuming the opposite truth value.

Exhaustive DPLL Search

- The trace of an exhaustive DPLL search can be viewed as a compilation of the propositional theory.
- With little modifications, DPLL can be converted to OBDD, d-DNNF, FBDD etc.
- We have developed Exhaustive DPLL search algorithm in Haskell.

Features in Exhaustive DPLL Search

- Different Variable Ordering can be passed by the user on which the branching variable is decided.
- User can get the Binary tree and analyze the effect of ordering and assignment.
- User can check the assignment is SAT or UNSAT
- User can get the Number of models in a formula.
- User can get all possible SAT assignments.
- User can get the runtime to compare different algorithms.

Some Heuristics used in above Algorithms

Unit Propagation

- We search for a clause containing only 1 variable if there is one then we assume that literal to be true and remove all the clauses containing that literal.
- We also remove occurrences of its negation from all the remaining clauses.

Pure literal Elimination

- A literal is called pure if its negation appears nowhere in the formula.
- Setting that literal to true will satisfy some number of clauses automatically and simplify the formula