Computing Lab - 1 (2021) Tutorial on SAT Solvers

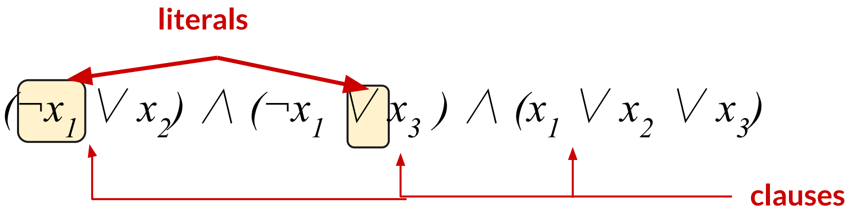
Sourav Das

Topics

- Introduction on SAT
- DIMACS Format
- Using the SAT Solver API’s
- Encoding a given problem in SAT.

Introduction to SAT

- In Boolean logic, a formula is in conjunctive normal form (CNF) or clausal normal form if it is a conjunction of one or more clauses, where each clause is a disjunction of literals



- The input of the SAT Solvers are a set of clauses in CNF
- We need to model the problem with a set of literals, and express the constraints in terms of clauses made from those literals.
- Tseytin Transformation takes an input of any arbitrary combinatorial logic circuit and produces a Boolean formula in CNF, which can be solved by the SAT Solver

DIMACS Format

- Each file starts with a header of the form “p cnf <no_of_variables> <no_of_clauses>”
- After that <no_of_clauses> lines follow stating each stating a clause
- Literals with positive polarity are marked with their corresponding index, whereas literals with negative polarity are marked with their respective negative index. (For eg. x_{15} represented as 15 and $\neg x_{15}$ represented as -15).
- The lines are terminated by 0
- Comment lines in the Dimacs format starts with c

Example format:

p cnf 3 3 //header
-1 2 0 // $\neg x_1 \vee x_2$
-1 3 0 // $\neg x_1 \vee x_3$
1 2 3 0 // $x_1 \vee x_3 \vee x_2$

c This is a sample DIMACS Format File

$$(\neg x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

Tseytin Transformation

- It breaks the given formula into smaller sub-formulas at the cost of adding new variables.
- Consider the formula
  - $\phi := ((p \lor q) \land r) \to (\neg s)$

    $x_1 \leftrightarrow (\neg s)$
    $x_2 \leftrightarrow (p \lor q)$
    $x_3 \leftrightarrow x_2 \land r$
    $x_4 \leftrightarrow x_3 \to x_1$
    $T(\phi) = x_4 \land (x_4 \leftrightarrow x_3 \to x_1) \land (x_3 \leftrightarrow x_2 \land r) \land (x_2 \leftrightarrow (p \lor q)) \land (x_1 \leftrightarrow (\neg s))$

Tseytin Transformation : https://en.wikipedia.org/wiki/Tseytin_transformation

Using SAT Solver API's

- You will be provided with a sample header file of "togasat" (Sat solver with C++ API)
- Using togasat in C++
    - Include togasat header file
    - Command to Initialise the SAT Solver
        - togasat::Solver solver;
    - Clause Formation is a vector<int> in C++
    - Command to add the clause in Solver
        - solver.addClause(clause);
    - Invoking the SAT Solver (Returns 0: SAT, 1; UNSAT; 2: UNKNOWN)
        - togasat::lbool status = solver.solve();
    - Finally getting the result
        - solver.printAnswer();

Togasat Github Link: https://github.com/togatoga/togasat

Encoding a given problem in SAT

- Suppose you are asked to sort 3 number using Boolean Satisfiability problem.
- Key Idea in using sat solvers is to represent the given problem in CNF using boolean variables.
- Add constraints to the SAT solvers to prune the search space
- How many variables do you require for this problem?
    - You have 3 numbers N_1; N_2; N_3 and for sorting you need a permutation order of these numbers.
    - So the 3 numbers goes to 3 places say P_1; P_2; P_3
    - So we can say that the number N_1 can be either in place P_1 ; P_2 ; or P_3. Since this is a boolean satisfiability problem we add 3 variables N_1P_1; N_1P_2; N_1P_3 , where N_xP_y = 1 if N_x is placed in position P_y; 0 otherwise.
    - So a total of 9 variables to start with.

Encoding a given problem in SAT

- What constraints do you think you need to add?
  - Each variable N_x must be placed in either of P_1 ; P_2 ; or P_3
    - $(N_xP_1 \lor N_xP_2 \lor N_xP_3)$
  - Each variable N_x must be placed in exactly one position only
    - $N_xP_1 \rightarrow \neg N_xP_2 \land \neg N_xP_3 \land (\forall_y (y != x) \neg N_yP_1)$
    - $N_1P_1 \rightarrow (\neg N_1P_2 \land \neg N_1P_3 \land \neg N_2P_1 \land \neg N_3P_1)$
- What more do you need to do?
  - Add constraints based on ordering i.e
    - $(N_xP_1 \land N_yP_2) \rightarrow N_xLN_y$ (where N_xLN_y is true if $N_x <= N_y$ because we need a sorted list)
    - $(N_xP_2 \land N_yP_3) \rightarrow N_xLN_y$
    - So this results in 6 more variables for our problem (N_1LN_2 ; N_1LN_3 ; N_2LN_1 ; N_2LN_3 ; N_3LN_1 ; N_3LN_2)
    - Finally based on the input values we need to add the last 6 constraints by doing pairwise comparison.

Thank You.