Boolean Satisfiability Solving

Hao Zheng
zheng@cse.usf.edu

Computer Science and Engineering
University of South Florida
Basics about SAT

- Boolean satisfiability (SA) problem is to determine if there is an assignment to variables that makes a Boolean formula satisfied.

- If a Boolean formula is defined over $n$ variables, there exist $2^n$ possible assignments.

- SAT is a classic NP-complete problem.
  - Very unlikely polynomial algorithms exist.
  - Techniques and heuristics help improve performance.
Importance of SAT

- Many important applications need fast SAT engines.
  - Theorem proving.
  - Bounded model checking.
  - Circuit testing.
  - Logic synthesis.
  - AI planning.
  - Software verification.
Definitions and Notations

- **Conjunctive Normal Form (CNF):** A formula is a conjunction of clauses.
- **Clause:** A disjunction of literals.
- **Literals:** Instances of variables.
- **Example**

\[(1) (x_2 + \neg x_3) \cdot (\neg x_1 + x_3) \cdot (\neg x_2 + x_4) \cdot (x_1 + x_3 + x_4)\]

- A CNF formula is also referred to as a clause database.
- SAT solvers accept formulas in CNF for efficiency purposes.
- Any Boolean formulas can be converted to CNF polynomially.
Definitions and Notations (1)

- **Assignment**: a set of valuations of variables.
  - Written as $A = \{(x_i, v(x_i)), \ldots\}$.
  - Let $n$ be the number of variables in a formula.
  - $|A| < n$: partial assignment.
  - $|A| = n$: complete assignment.

- **Satisfying assignment**: make a formula evaluate to true.

- **Unsatisfying assignment**: make a formula evaluate to false.

Examples of assignments for formula (1):
- Satisfying: $A = \{(x_1, 0), (x_2, 1), (x_4, 1)\}$.
- Unsatisfying: $A = \{(x_1, 1), (x_2, 0), (x_3, 0), (x_4, 0)\}$.
Definitions and Notations (2)

With $A$, a clause databased is partitioned into
- **Satisfied**: evaluated to 1.
- **Unsatisfied**: evaluated to 0.
- **Unresolved**: truth value unknown.

Example: $A = \{(x_1, 1), (x_4, 1)\}$ for

\[c_1 : (x_2 + \neg x_3), \quad c_2 : (\neg x_1 + x_3),\]
\[c_3 : (\neg x_2 + x_4), \quad c_4 : (x_1 + x_3 + x_4)\]

- **Satisfied**: $c_3, c_4$.
- **Unresolved**: $c_1, c_2$. 
Definitions and Notations (3)

- **Free literals**: unassigned literals.
- **Unit clause**: a clause with only one free literal. A variable must be assigned to a specific value if its literal appears in a unit clause.
- **Decision Assignments**: Assignment to a variable if it does not appear in any unit clause. With choice of assignments.
- **Implication assignments**: variable assignments due to unit clauses. Without choice of assignments.
Basic Backtrack Search

- Trial-and-error style search.
- Iterate through
  - Make a decision assignment.
  - Deduce implications.
  - Backtrack to reverse a previous decision if a conflict arises.
- An assignment leading to a conflict is also called **conflicting** assignment.
- This is referred to as DPLL algorithm proposed by Davis, Logemann and Loveland.
Search is organized as a **decision tree**.

- Nodes are labeled with decision variables.
- Edges are labeled with decisions and implications.
- Each assignment is associated with a decision level.

\[
c_1 : (x_2 + \neg x_3), \quad c_2 : (\neg x_1 + x_3), \quad c_3 : (\neg x_2 + x_4), \quad c_4 : (x_1 + x_3 + x_4)
\]
Conflict Analysis

- Conflicts often occur, and unavoidable for UNSAT problems.
- Deduction and backtracking are costly.
- Conflict analysis helps search by
  - Generating **conflict-induced** clauses.
  - Obtaining a decision level for backtracking.
- Augmenting the original formula with conflict-induced clauses is called **learning**.
  - Helps to avoid hitting the same conflict again.
- **Non-chronological** backtracking helps to trim large search space.
Conflict Analysis (1)

- Conflicting assignment:
  \[ A_C = \{(x_1, 1), (x_9, 0), (x_{10}, 0), (x_{11}, 0)\} \].

- Negating the conjunction of \( A_C \) results in a conflict-induced clause \((\neg x_1 + x_9 + x_{10} + x_{11})\).

Current assignment:
\{(x_9,0)@1, (x_{10},0)@3, (x_{11},0)@3, (x_{12},1)@2, (x_{13},1)@2\}

Decision assignment:
\((x_1,1)@6\)

Clause database:
\[

c_1 = (\neg x_1 + x_2) \\
c_2 = (\neg x_1 + x_3 + x_9) \\
c_3 = (\neg x_2 + \neg x_3 + x_4) \\
c_4 = (\neg x_4 + x_5 + x_{10}) \\
c_5 = (\neg x_4 + x_6 + x_{11}) \\
c_6 = (\neg x_5 + \neg x_6) \\
c_7 = (x_1 + x_7 + \neg x_{12}) \\
c_8 = (x_1 + x_8) \\
c_9 = (\neg x_7 + \neg x_8 + \neg x_{13})
\]
Conflict Analysis (2)

- The largest decision level is 6 from $A_C$.
- This is where the search backtracks.

Now, assignment $(x_1, 0)$ is an implication due to

$$(-x_1 + x_9 + x_{10} + x_{11}).$$

A new conflict arises.

$$A_C = \{(x_9, 0), (x_{10}, 0), (x_{11}, 0), (x_{12}, 1), (x_{13}, 1)\}$$
Conflict Analysis (3)

- $A_C = \{(x_9, 0), (x_{10}, 0), (x_{11}, 0), (x_{12}, 1), (x_{13}, 1)\}$
- The largest decision level from $A_C$ is 3.
- The current decision level is 6.
- Non-chronological backtracking jumps over several levels.
- Trims large search space.