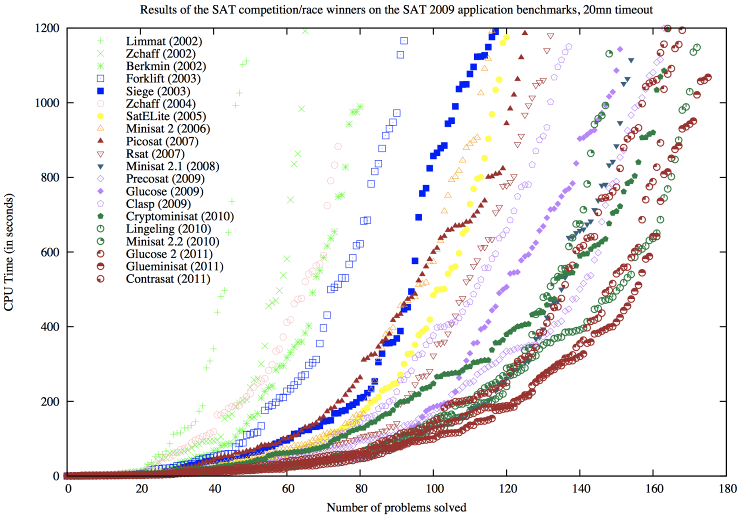
AAA528: Computational Logic

Lecture 2 — CDCL SAT Solvers

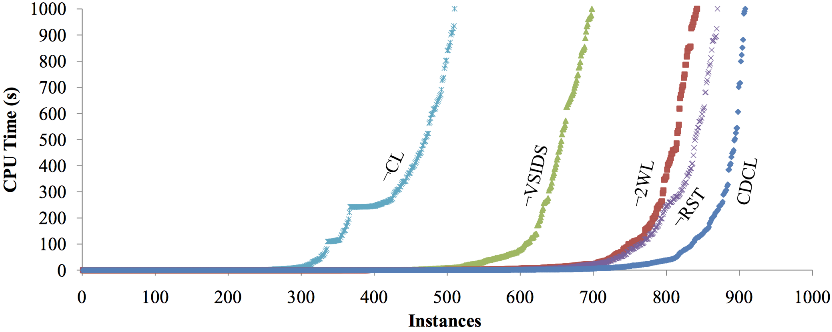
Hakjoo Oh
2018 Fall

Progress of SAT Solving



(Courtesy of D. Le-Berre)

Impact of CDCL



(Courtesy of Katebi et al. 2011)

Review: DPLL

$$
\begin{array}{l}
\text{let rec } \textbf{DPLL } F = \\
\quad \text{let } F' = \textbf{BCP}(F) \text{ in} \\
\quad \text{if } F' = \top \text{ then true} \\
\quad \text{else if } F' = \bot \text{ then false} \\
\quad \text{else} \\
\quad\quad \text{let } P = \textbf{Choose}(\textbf{vars}(F')) \text{ in} \\
\quad\quad (\textbf{DPLL } F'\{P \mapsto \top\}) \vee (\textbf{DPLL } F'\{P \mapsto \bot\})
\end{array}
$$

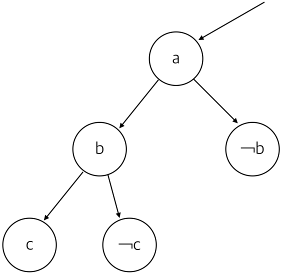
DPLL performs backtrack search, where each step involves

- deciding a variable to branch on,
- propagating logical implication of this decision, and
- backtracking in the case of conflict.

Modern SAT Solving

Three major features of CDCL SAT solvers:

- Non-chronological backtracking
  - DPLL always backtracks to the most recent decision level.



- Learning from past failures (covered in this lecture)
  - DPLL revisits bad partial assignments that share the same root cause.
- Heuristics for choosing variables and assignments
  - DPLL chooses arbitrary variables.

Decision Variable and Level

DPLL performs a search on a binary tree.

- Decision variable: the assigned variable
- Decision level: the depth of the binary tree at which the decision is made, starting from 1.
    - The assignments implied by a decision (via BCP) are associated with the level of the decision.

Example:

$$(\neg P \vee Q) \wedge (R \vee \neg Q \vee S)$$

- Choose P and assign $P = \top$: P is the decision variable at level 1.
- With BCP, Q is assigned $\top$ at level 1.
- Choose R and assign $R = \top$ at decision level 2.
- BCP deduces $S = \top$. the decision level of S is 2.

Example

Consider the CNF formula:

$$\begin{aligned}\phi &= w_1 \wedge w_2 \wedge w_3 \\ &= (x_1 \vee \neg x_4) \wedge (x_1 \vee x_3) \wedge (\neg x_3 \vee x_2 \vee x_4)\end{aligned}$$

- Assume the decision assignment: $x_4 = 0@1$.
- Unit propagation yields no additional implications.
- The second decision: $x_1 = 0@2$.
- Unit propagation yields implied assignments $x_3 = 1@2$ and $x_2 = 1@2$.
- $\alpha(x_3) = w_2$ and $\alpha(x_2) = w_3$.

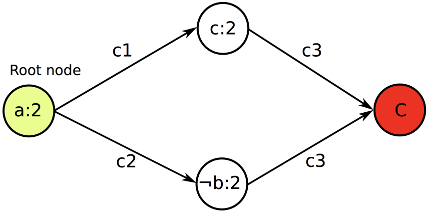
Implication Graph

- An implication graph is a labelled directed acyclic graph $G(V, E)$
- Nodes (V) are the literals in the current partial assignment. Each node is labelled with the literal and the decision level at which it is assigned.
  - $x_i : dl$: x_i was assigned to $\top$ at decision level dl.
  - $\neg x_i : dl$: x_i was assigned to $\bot$ at decision level dl.
- E denotes the set of directed edges labelled with clauses: $l \xrightarrow{c} l'$. Edges from $l_1, \ldots, l_k$ to l labelled with c mean that assignments $l_1, \ldots, l_k$ caused assignment l due to clause c during BCP.
  - If l' is implied from c, then there is an directed edge from l to l' where $\neg l \in c$.
- A special node C (or κ) is called the conflict node.
- Edge to conflict node labeled with c: current partial assignment contradicts clause c.

Example 1

$$c_1 : (\neg a \vee c) \quad c_2 : (\neg a \vee \neg b) \quad c_3 : (\neg c \vee b)$$

- Assume a is assigned $\top$ at decision level 2.
- The implication graph:

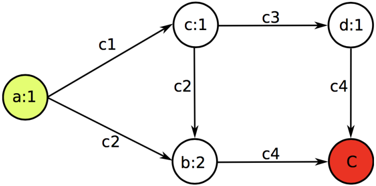


  - The root node denotes the decision literal.
  - $a \xrightarrow{c_1} c$: assignment $a = \top$ caused assignment $c = \top$ due to clause c_1 during BCP. Similar for $a \xrightarrow{c_2} \neg b$.
  - $c \xrightarrow{c_3} C$ and $b \xrightarrow{c_3} C$: assignments $c = \top$ and $b = \bot$ caused a contradiction due to clause c_3.

Example 2

$$c_1 : (\neg a \vee c) \quad c_2 : (\neg c \vee \neg a \vee b) \quad c_3 : (\neg c \vee d) \quad c_4 : (\neg d \vee \neg b)$$

- Assume a is assigned $\top$ at decision level 1.
- During BCP,
  - $a = \top$ causes $c = \top$ due to c_1: $a \xrightarrow{c_1} c$.
  - $a = \top$ and $c = \top$ cause $b = \top$ due to c_2: $a \xrightarrow{c_2} b$ and $c \xrightarrow{c_2} b$.
  - $c = \top$ causes $d = \top$ due to c_3: $c \xrightarrow{c_3} d$.
  - Assignments $b = \top$ and $d = \top$ cause a contradiction due to c_4: $b \xrightarrow{c_4} C$ and $d \xrightarrow{c_4} C$.
- The implication graph:

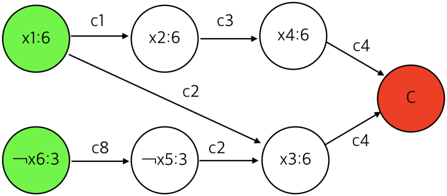


Example 3

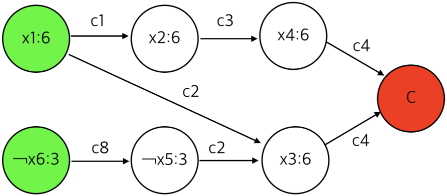
Consider a formula that contains the following clauses, among others:

$$c_1 : (\neg x_1 \vee x_2) \quad c_2 : (\neg x_1 \vee x_3 \vee x_5) \quad c_3 : (\neg x_2 \vee x_4) \quad c_4 : (\neg x_3 \vee \neg x_4)$$
$$c_5 : (x_1 \vee x_5 \vee \neg x_2) \quad c_6 : (x_2 \vee x_3) \quad c_7 : (x_2 \vee \neg x_3) \quad c_8 : (x_6 \vee \neg x_5)$$

- Assume that at decision level 3 the decision was $\neg x_6$, which implied $\neg x_5$ due to c_8.
- Assume further that the solver is now at decision level 6 and assigns $x_1 = \top$. At decision levels 4 and 5, variables other than $x_1, \ldots, x_6$ were assigned and not relevant to these clauses.
- The implication graph:



Learning a Conflict Clause

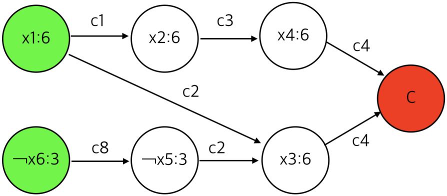


- To avoid the conflict, the solver learns a *conflict clause*

$$c_9 : (x_5 \vee \neg x_1)$$

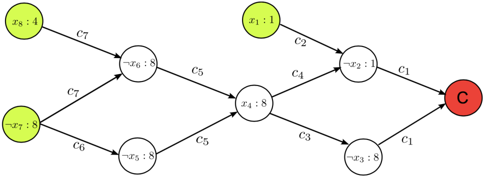
  and adds it to the formula.
- This process of adding conflict clauses is the solver's way to learn from its past mistakes.

Learning a Conflict Clause via Resolution



- Start from the unsatisfied clause: $c := c_4 = (\neg x_3 \vee \neg x_4)$
- Pick the implied literal with level 6 in the clause: x_3
- Pick any incoming edge of x_3: $c_2 = (\neg x_1 \vee x_3 \vee x_5)$
- Resolve c_4 and c_2: $c := (\neg x_1 \vee \neg x_4 \vee x_5)$
- Pick the implied literal with level 6: $\neg x_4$
- PIck the incoming edge of x_4: $c_3 = (\neg x_2 \vee x_4)$
- Resolve c_3 and c: $c := (\neg x_1 \vee \neg x_2 \vee x_5)$
- Pick the implied literal with level 6: $\neg x_2$
- Pick the incoming edge: $c_1 = (\neg x_1 \vee x_2)$
- Resolve c_1 with c: $c := (\neg x_1 \vee x_5)$. No more resolutions.

Heuristic for Deriving Better Conflict Clause



Learn smaller conflict clause $x_2 \vee \neg x_4$.

1. Find first unique implication point (UIP): $x_4 : 8$.
   - All paths from current decision node to the conflict node must go through UIP. First UIP is closest to conflict node.
2. The clause labelling incoming edge to C: $c_1 = (x_2 \vee x_3)$
3. Find the last assigned literal in c_1: $\neg x_3$
4. Pick any incoming edge to $\neg x_3$: $c_3 = (\neg x_4 \vee \neg x_3)$
5. Resolve c_1 and c_3: $x_2 \vee \neg x_4$
6. Set the current clause to resolvent and repeat (2)–(5) until negation of first UIP is found

Backtracking Level

- Backtracking level d deletes all assignments made after level d (assignments made d not deleted)
- A good strategy is to backtrack to the second highest decision level d' for literals in the conflict clause c.
- At the level d', c is always unit (exactly one unassigned literal).

Summary

Algorithm 2.2.1: DPLL-SAT

Input: A propositional CNF formula $\mathcal{B}$
Output: "Satisfiable" if the formula is satisfiable and "Unsatisfiable" otherwise

```
function DPLL
    if BCP() = "conflict" then return "Unsatisfiable";
    while (TRUE) do
        if ¬DECIDE() then return "Satisfiable";
        else
            while (BCP() = "conflict") do
                backtrack-level := ANALYZE-CONFLICT();
                if backtrack-level < 0 then return "Unsatisfiable";
                else BackTrack(backtrack-level);
```

- Conflict-Driven Clause Learning
- Variable selection heuristics: DLIS, VSIDS, ...
- Slides based on the lecture (See for other heuristics/optimizations):

  `http://www.cs.utexas.edu/~isil/cs389L/lecture3-6up.pdf`