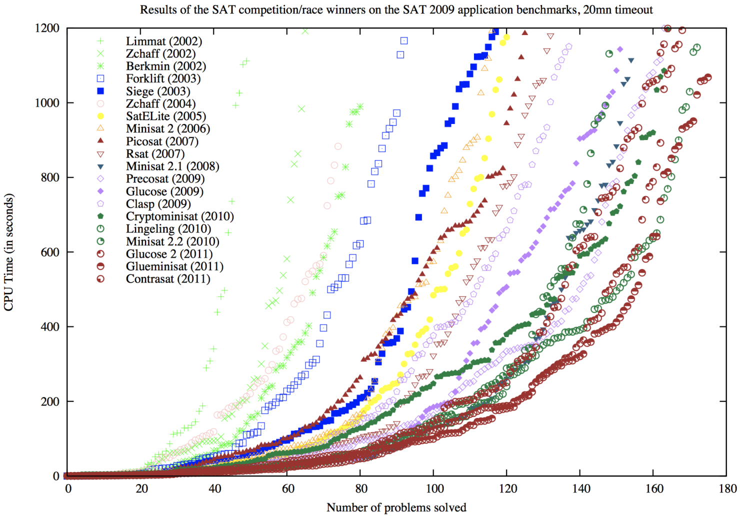
AAA528: Computational Logic

Lecture 2 — CDCL SAT Solvers

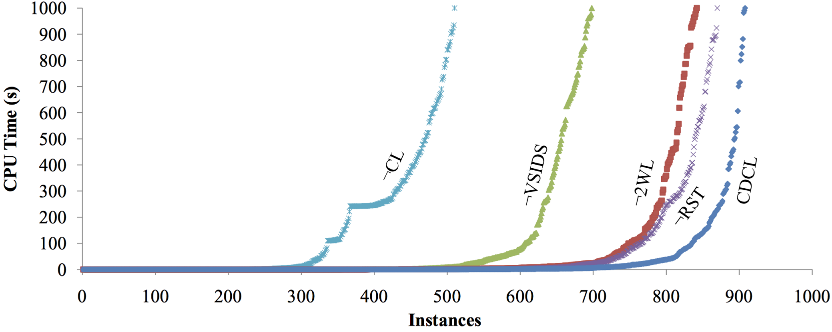
Hakjoo Oh
2025 Spring

Progress of SAT Solving



(Courtesy of D. Le-Berre)

Impact of CDCL



(Courtesy of Katebi et al. 2011)

Review: DPLL

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

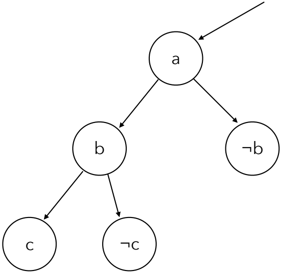
DPLL performs backtrack search, where each step involves

- 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:

$$(\neg P \lor Q) \land (R \lor \neg Q \lor S)$$

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

Example (Decision Level and Antecedents)

Consider the CNF formula:

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

- 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 $\alpha(x_2) = w_3$.
  - $\alpha(x)$: the *antecedent* of x, the unit clause used for implying x
- $\delta(x_4) = 1$ and $\delta(x_3) = 2$
  - $\delta(x) \in \{-1, 0, 1, \ldots, |X|\}$: the decision level of x

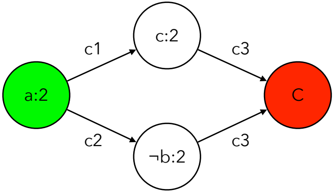
Implication Graph

- 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 a directed edge from l to l' where $\neg l \in c$. (if $l \xrightarrow{c} l'$, then $\neg l \in c$)
- A special node C (or κ) is called the conflict node. C is generated when unit propagation yields an unsatisfied clause (c). $\alpha(C) = c$.
- 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:

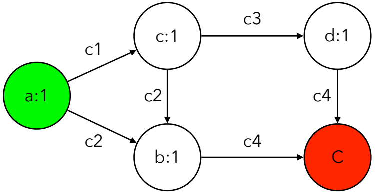


  - The root node denotes the decision literal.
  - $a \xrightarrow{c_1} c$: assignment $a = \top$ caused assignment $c = \top$ due to clause c_1 during BCP. Similar for $a \xrightarrow{c_2} \neg b$.
  - $c \xrightarrow{c_3} C$ and $b \xrightarrow{c_3} C$: assignments $c = \top$ and $b = \bot$ caused a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

Example 2

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

- 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:

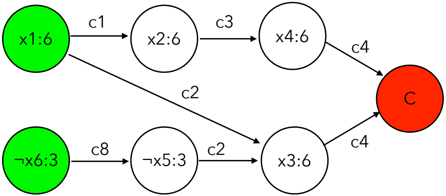


Example 3

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

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

- 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 (partial) implication graph:



Exercise

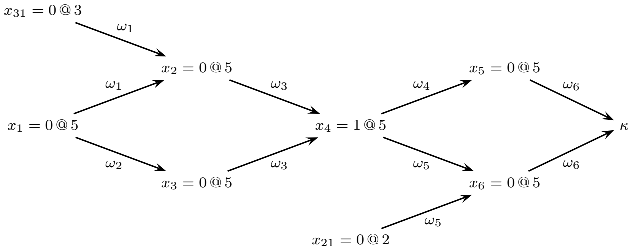
Consider the CNF formula:

$$
\begin{aligned}
\varphi_1 &= \omega_1 \wedge \omega_2 \wedge \omega_3 \wedge \omega_4 \wedge \omega_5 \wedge \omega_6 \\
&= (x_1 \vee x_{31} \vee \neg x_2) \wedge (x_1 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee x_4) \wedge \\
&\quad\; (\neg x_4 \vee \neg x_5) \wedge (x_{21} \vee \neg x_4 \vee \neg x_6) \wedge (x_5 \vee x_6)
\end{aligned}
$$

- Assume decision assignments $\boldsymbol{x_{21} = 0@2}$ and $\boldsymbol{x_{31} = 0@3}$
- The current decision assignment: $\boldsymbol{x_1 = 0@5}$.

The implication graph:

Conflict Clause



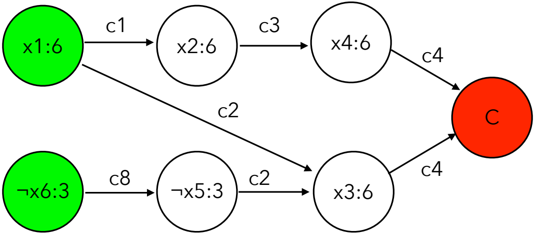
- From this failure, we learn that $\neg \boldsymbol{x_1} \wedge \neg \boldsymbol{x_{31}} \wedge \neg \boldsymbol{x_{21}}$ leads to a conflict.
- To avoid the conflict, the solver learns a *conflict clause*

$$\boldsymbol{c_9 : x_1 \vee x_{31} \vee x_{21}}$$

  and adds it to the formula. This process of adding conflict clauses is the solver's way to learn from its past mistakes.
- Conflict clauses prune the search space (and also have an impact on the decision heuristic).

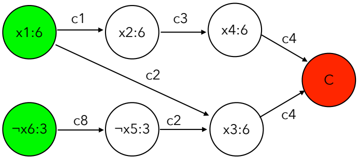
Exercise

Find a conflict clause from the failure:



Learning a Conflict Clause via Resolution

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



- Start from the unsatisfied clause: $c := c_4 = (\neg x_3 \vee \neg x_4)$
- Pick the implied literal with the current decision level (6) in c: e.g., x_3
- Pick any incoming edge (antecedent) 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 (no literal with the current decision level and incoming edge).

Learning a Conflict Clause via Resolution

The clause learning procedure:

$$\omega_L^{d,i} = \begin{cases} \alpha(\kappa) & \text{if } i = 0 \\ \omega_L^{d,i-1} \odot \alpha(l) & \text{if } i \neq 0 \land \xi(\omega_L^{d,i-1}, l, d) = 1 \\ \omega_L^{d,i-1} & \text{if } i \neq 0 \land \forall_l\, \xi(\omega_L^{d,i-1}, l, d) = 0 \end{cases}$$

- $\alpha(\kappa)$: all literals in the unsatisfied clause
- $\xi(\omega, l, d)$ is true if a clause ω has an implied literal l assigned at the current decision level d:

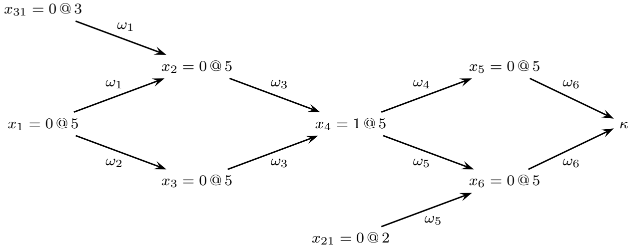
$$\xi(\omega, l, d) \iff l \in \omega \land \delta(l) = d \land \alpha(l) \neq \mathsf{NIL}$$

- When $i = 0$, the clause is set to the unsatisfied clause $\alpha(\kappa)$.
- At each step i, a literal l assigned at the current decision level d is selected and the intermediate clause is resolved with the antecedent of l.

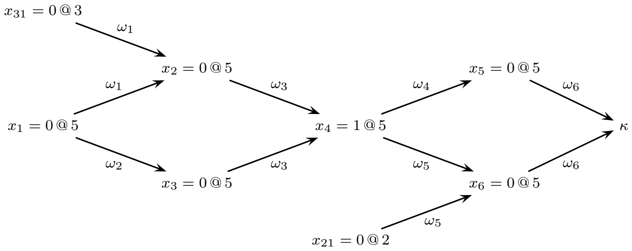
Exercise

Apply the clause learning procedure to the example:

$$
\begin{aligned}
\varphi_1 &= \omega_1 \wedge \omega_2 \wedge \omega_3 \wedge \omega_4 \wedge \omega_5 \wedge \omega_6 \\
&= (x_1 \vee x_{31} \vee \neg x_2) \wedge (x_1 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee x_4) \wedge \\
&\quad\ (\neg x_4 \vee \neg x_5) \wedge (x_{21} \vee \neg x_4 \vee \neg x_6) \wedge (x_5 \vee x_6)
\end{aligned}
$$



Heuristic for Deriving Smaller Conflict Clause



Goal: Learning a smaller conflict clause $\neg \boldsymbol{x_4} \vee \boldsymbol{x_{21}}$.

1. Find first unique implication point (UIP): $\boldsymbol{x_4 = 1@5}$.
    - All paths from current decision node to the conflict node must go through UIP. First UIP is closest to conflict node.
2. Stop clause learning at the first UIP.

Clause Learning with UIPs

- Observation: In the implication graph, there is a UIP at decision level d, when the number of literals in $\omega_L^{d,i}$ assigned at decision level d is 1.
- Let $\sigma(\omega, d)$ be the number of literals in ω assigned at decision level d:

$$\sigma(\omega, d) = |\{l \in \omega \mid \delta(l) = d\}|$$

- The clause learning procedure with UIPs:

$$w_L^{d,i} = \begin{cases} \alpha(\kappa) & \text{if } i = 0 \\ w_L^{d,i-1} & \text{if } i \neq 0 \land \sigma(w_L^{d,i-1}, d) = 1 \\ w_L^{d,i-1} \odot \alpha(l) & \text{if } i \neq 0 \land \xi(w_L^{d,i-1}, l, d) = 1 \end{cases}$$

- Example:

$$\begin{array}{ll} w_L^{5,0} = \{x_5, x_6\} & \text{Literals in } \alpha(\kappa) \\ w_L^{5,1} = \{\neg x_4, x_6\} & \text{Resolve with } \alpha(x_5) = \omega_4 \\ w_L^{5,2} = \{\neg x_4, x_{21}\} & \text{No more resolution applicable} \end{array}$$

CDCL Algorithm

Algorithm 1 Typical CDCL algorithm

```
CDCL(φ, ν)
if (UnitPropagation(φ, ν) == CONFLICT)
   then return UNSAT
dl ← 0                                          ▷ Decision level
while (not AllVariablesAssigned(φ, ν))
      do (x, v) = PickBranchingVariable(φ, ν)               ▷ Decide stage
         dl ← dl + 1                 ▷ Increment decision level due to new decision
         ν ← ν ∪ {(x, v)}
         if (UnitPropagation(φ, ν) == CONFLICT)             ▷ Deduce stage
            then β = ConflictAnalysis(φ, ν)                 ▷ Diagnose stage
                 if (β < 0)
                    then return UNSAT
                    else Backtrack(φ, ν, β)
                         dl ← β             ▷ Decrement decision level due to backtracking
return SAT
```

- ConflictAnalysis analyzes the most recent conflict, learns a new clause from the conflict, and returns a backtracking level.
- Backtrack backtracks to the decision level computed by ConflictAnalysis.

Summary

- Conflict-Driven Clause Learning
- Modern CDCL SAT solvers involves a number of additional issues:
  - Variable selection heuristics
  - Lazy data structures
  - Periodic restart of backtrack search
  - Deletion policies for learnt clauses
  - ...
- Slides are based on the following references:
  - Decision Procedures. Springer
  - Handbook of Satisfiability. IOS Press
  - `http://www.cs.utexas.edu/~isil/cs389L/lecture3-6up.pdf`