Satisfiability, Boolean Modeling and Computation
Spring 2016

Matti Järvisalo

Lecture 3: Conflict-Driven Clause Learning (CDCL)
January 26, 2016
The Practical SAT-based Approach

1. **Encoding**: how to represent your problem in CNF
   - Possibly through many steps via more high-level representations

2. **Preprocessing**: automated simplification (rewriting) of the CNF
   - reformulating the encoding

3. **SAT solving**: how to determine satisfiability fast in practice
The Practical SAT-based Approach

1. Encoding: how to represent your problem in CNF
   - Possibly through many steps via more high-level representations
2. Preprocessing: automated simplification (rewriting) of the CNF
   - reformulating the encoding
3. SAT solving: how to determine satisfiability fast in practice
Three Types of SAT Instances,
Three Approaches to Solving

- Three categories of SAT instances
  - **Application/Industrial:** CDCL ("intelligent" search)
  - **Crafted:** CDCL / DPLL / local search
    ("search over whole search space as fast as you can")
  - **Random:** DPLL / local search / survey propagation
DPLL: Example

\[(x \lor y \lor z)\]
\[(x \lor y \lor \neg z)\]
\[(x \lor \neg y \lor z)\]
\[(x \lor \neg y \lor \neg z)\]
\[(\neg x \lor y \lor z)\]
\[(\neg x \lor y \lor \neg z)\]
\[(\neg x \lor \neg y \lor z)\]
\[(\neg x \lor \neg y \lor \neg z)\]

In every branch

- Need to assign two variables
- Then assigning third gives a conflict by unit propagation
DPLL: Example

Looking at the leftmost branch

- First clause unit propagates $z = 1$
- Second clause becomes falsified

\[
\begin{align*}
(x \lor y \lor z) \\
(x \lor y \lor \neg z) \\
(x \lor \neg y \lor z) \\
(x \lor \neg y \lor \neg z) \\
(\neg x \lor y \lor z) \\
(\neg x \lor y \lor \neg z) \\
(\neg x \lor \neg y \lor z) \\
(\neg x \lor \neg y \lor \neg z)
\end{align*}
\]

$x = 0$

$y = 0$

$z = 1$

DPLL will simply backtrack by

- Undoing $y = 0$, $z = 1$
- Assigning $y = 1$ (the other branch for $y$)
Conflict-Driven Clause Learning (CDCL)

CDCL behaves differently

- Not standard backtracking search
- Instead of branching on a variable by $x = 0$ and $x = 1$:
  Makes \textit{decisions}: individually assign $x = 0$ or $x = 1$
- At a conflict:
  - \textit{Learn a clause} describing the conflict
  - Undo some number of assignments and continue

$\begin{align*}
(x \lor y \lor z) \\
(x \lor y \lor \neg z) \\
(x \lor \neg y \lor z) \\
(x \lor \neg y \lor \neg z) \\
(\neg x \lor y \lor z) \\
(\neg x \lor y \lor \neg z) \\
(\neg x \lor \neg y \lor z) \\
(\neg x \lor \neg y \lor \neg z) \\
\end{align*}$

$\begin{align*}
x &= 0 \\
y &= 0 \\
z &= 1
\end{align*}$
Conflict-Driven Clause Learning (CDCL)

CDCL behaves differently

- Not standard backtracking search
- Instead of branching on a variable by $x = 0$ and $x = 1$:
  Makes decisions: individually assign $x = 0$ or $x = 1$
- At a conflict:
  - Learn a clause describing the conflict
  - Undo some number of assignments and continue

\[
\begin{aligned}
(x \lor y \lor z) & \\
(x \lor y \lor \neg z) & \\
(x \lor \neg y \lor z) & \\
(x \lor \neg y \lor \neg z) & \\
(\neg x \lor y \lor z) & \\
(\neg x \lor y \lor \neg z) & \\
(\neg x \lor \neg y \lor z) & \\
(\neg x \lor \neg y \lor \neg z) & \\
\end{aligned}
\]

$\text{decision} \rightarrow x = 0$

$\text{decision} \rightarrow y = 0$

$\text{unit prop} \rightarrow z = 1$

$\text{conflict} \rightarrow \times$
Conflict-Driven Clause Learning (CDCL)

CDCL behaves differently

- Not standard backtracking search
- Instead of branching on a variable by $x = 0$ and $x = 1$: Makes *decisions*: individually assign $x = 0$ or $x = 1$
- At a conflict:
  - Learn a clause describing the conflict
  - Undo some number of assignments and continue

\[
\begin{align*}
(x \lor y \lor z) & \quad decision \rightarrow x = 0 \\
(x \lor y \lor \neg z) & \quad decision \rightarrow y = 0 \\
(x \lor \neg y \lor z) & \quad unit \ prop \rightarrow z = 1 \\
(x \lor \neg y \lor \neg z) & \quad conflict \rightarrow x
\end{align*}
\]

learned:

\[
(x \lor y)
\]
Conflict-Driven Clause Learning (CDCL)

CDCL behaves differently

- Not standard backtracking search
- Instead of branching on a variable by $x = 0$ and $x = 1$:
  Makes decisions: individually assign $x = 0$ or $x = 1$
- At a conflict:
  - Learn a clause describing the conflict
  - Undo some number of assignments and continue

\[
\begin{align*}
(x \lor y \lor z) \\
(x \lor y \lor \neg z) \\
(x \lor \neg y \lor z) \\
(x \lor \neg y \lor \neg z) \\
(\neg x \lor y \lor z) \\
(\neg x \lor y \lor \neg z) \\
(\neg x \lor \neg y \lor z) \\
(\neg x \lor \neg y \lor \neg z) \\

\text{learned:} \\
(x \lor y) \\
\end{align*}
\]

decision $\rightarrow$ $x = 0$
Conflict-Driven Clause Learning (CDCL)

CDCL behaves differently

- Not standard backtracking search
- Instead of branching on a variable by \( x = 0 \) and \( x = 1 \):
  - Makes *decisions*: individually assign \( x = 0 \) or \( x = 1 \)
- At a conflict:
  - Learn a clause describing the conflict
  - Undo some number of assignments and continue

\[
\begin{align*}
(x \lor y \lor z) & \quad & \text{decision} \rightarrow & \quad & x = 0 \\
(x \lor y \lor \neg z) & \quad & \text{unit prop} \rightarrow & \quad & y = 1 \\
(x \lor \neg y \lor z) & \quad & \text{unit prop} \rightarrow & \quad & z = 1 \\
(x \lor \neg y \lor \neg z) & \quad & \text{conflict} \rightarrow & \quad & \times \\
\end{align*}
\]
Conflict-Driven Clause Learning (CDCL)

CDCL behaves differently

- Not standard backtracking search
- Instead of branching on a variable by $x = 0$ and $x = 1$:
  Makes *decisions*: individually assign $x = 0$ or $x = 1$
- At a conflict:
  - *Learn a clause* describing the conflict
  - Undo some number of assignments and continue

\[
(x \lor y \lor z) \\
(x \lor y \lor \neg z) \\
(x \lor \neg y \lor z) \\
(x \lor \neg y \lor \neg z) \\
(\neg x \lor y \lor z) \\
(\neg x \lor y \lor \neg z) \\
(\neg x \lor \neg y \lor z) \\
(\neg x \lor \neg y \lor \neg z)
\]

*learned:*

\[
(x \lor y), (x)
\]

\[
\left(\begin{array}{c}
\text{decision} 
\rightarrow \\
\text{unit prop} 
\rightarrow \\
\text{unit prop} 
\rightarrow \\
\text{conflict} 
\rightarrow
\end{array}\right)
\]

\[
\left(\begin{array}{c}
x = 0 \\
y = 1 \\
z = 1 \\
\times
\end{array}\right)
\]
Conflict-Driven Clause Learning (CDCL)

CDCL behaves differently

- Not standard backtracking search
- Instead of branching on a variable by $x = 0$ and $x = 1$: Makes decisions: individually assign $x = 0$ or $x = 1$
- At a conflict:
  - Learn a clause describing the conflict
  - Undo some number of assignments and continue

\[
\begin{align*}
(x \lor y \lor z) \\
(x \lor y \lor \neg z) \\
(x \lor \neg y \lor z) \\
(x \lor \neg y \lor \neg z) \\
(\neg x \lor y \lor z) \\
(\neg x \lor y \lor \neg z) \\
(\neg x \lor \neg y \lor z) \\
(\neg x \lor \neg y \lor \neg z)
\end{align*}
\]

Learned:
\[
(x \lor y) \quad (x)
\]

unit prop $\rightarrow$ $x = 1$
Conflict-Driven Clause Learning (CDCL)

CDCL behaves differently

- Not standard backtracking search
- Instead of branching on a variable by $x = 0$ and $x = 1$:
  Makes decisions: individually assign $x = 0$ or $x = 1$
- At a conflict:
  - Learn a clause describing the conflict
  - Undo some number of assignments and continue

\[
\begin{align*}
(x \lor y \lor z) \\
(x \lor y \lor \neg z) \\
(x \lor \neg y \lor z) \\
(x \lor \neg y \lor \neg z) \\
(\neg x \lor y \lor z) \\
(\neg x \lor y \lor \neg z) \\
(\neg x \lor \neg y \lor z) \\
(\neg x \lor \neg y \lor \neg z) \\
\text{learned:} \\
(x \lor y) , (x)
\end{align*}
\]

\[
\begin{align*}
\text{unit prop } &\longrightarrow & x = 1 \\
\text{decision } &\longrightarrow & z = 0
\end{align*}
\]
Conflict-Driven Clause Learning (CDCL)

CDCL behaves differently

- Not standard backtracking search
- Instead of branching on a variable by \( x = 0 \) and \( x = 1 \): Makes *decisions*: individually assign \( x = 0 \) or \( x = 1 \)
- At a conflict:
  - Learn a clause describing the conflict
  - Undo some number of assignments and continue

\[
\begin{align*}
(x \lor y \lor z) \\
(x \lor y \lor \neg z) \\
(x \lor \neg y \lor z) \\
(x \lor \neg y \lor \neg z) \\
(\neg x \lor y \lor z) \\
(\neg x \lor y \lor \neg z) \\
(\neg x \lor \neg y \lor z) \\
(\neg x \lor \neg y \lor \neg z)
\end{align*}
\]

*learned:*

\[
(x \lor y) , (x) , (z)
\]

*unit prop* \( \rightarrow \) \( x = 1 \)

*decision* \( \rightarrow \)

\[
\begin{align*}
&z = 0 \\
y = 1 \\
\times
\end{align*}
\]

*conflict* \( \rightarrow \)
Conflict-Driven Clause Learning (CDCL)

CDCL behaves differently

- Not standard backtracking search
- Instead of branching on a variable by $x = 0$ and $x = 1$: Makes *decisions*: individually assign $x = 0$ or $x = 1$
- At a conflict:
  - Learn a clause describing the conflict
  - Undo some number of assignments and continue

$$(x \lor y \lor z)$$
$$(x \lor y \lor \neg z)$$
$$(x \lor \neg y \lor z)$$
$$(x \lor \neg y \lor \neg z)$$
$$(\neg x \lor y \lor z)$$
$$(\neg x \lor y \lor \neg z)$$
$$(\neg x \lor \neg y \lor z)$$
$$(\neg x \lor \neg y \lor \neg z)$$

learned:

$$(x \lor y) \ , \ (x) \ , \ (z)$$

$unit prop \rightarrow \ x = 1$
$unit prop \rightarrow \ z = 1$
$unit prop \rightarrow \ y = 1$
$conflict \rightarrow \ x$
Conflict-Driven Clause Learning (CDCL)

CDCL behaves differently

- Not standard backtracking search
- Instead of branching on a variable by $x = 0$ and $x = 1$: Makes *decisions*: individually assign $x = 0$ or $x = 1$
- At a conflict:
  - Learn a clause describing the conflict
  - Undo some number of assignments and continue

\[(x \lor y \lor z)\]  
\[(x \lor y \lor \neg z)\]  
\[(x \lor \neg y \lor z)\]  
\[(x \lor \neg y \lor \neg z)\]  
\[\neg x \lor y \lor z\]  
\[\neg x \lor y \lor \neg z\]  
\[\neg x \lor \neg y \lor z\]  
\[\neg x \lor \neg y \lor \neg z\]

learned:
\[(x \lor y), (x), (z), \emptyset\]

Termination:
Learned the empty clause
Conflict-Driven Clause Learning

*Not* standard backtracking search

*Not* “DPLL with clause learning”

Basic idea:

**While** *(true)*

- **If** (unit propagations derives a conflict)
  - **If** *(no decisions have been currently made)* Return UNSAT
  - **Else**
    - *Learn a conflict clause C* and add it to F
    - *Backjump* based on C (i.e. undo some assignments)

- **Else If** (all clauses are satisfied) Return SAT

- **Else** Assign a value to some unassigned variable (i.e. *make a decision*)
Key Aspects

- Learning: How to learn a conflict clause?
- Backjumping: How to do non-chronological backtracking?
- Decisions: What type of a decision heuristic to use?
- Reofs: How to do unit propagation fast?
Terminology

- Decision: forced assignment of a value to a variable (a *guess*)
- Implied assignment: assignment of a value to a variable by unit propagation
- Decision level:
  - For a decision:
    \[1 + \text{the number of currently forced decisions made before the decision}\]
  - For an implied assignment:
    the decision level at which the assignment was made
CDCL: Search and Clause Learning

\[(x_1 \lor x_4) \land (x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land (\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land F_{\text{extra}}\]
CDCL: Search and Clause Learning

\[(x_1 \lor x_4) \land \]
\[(x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land \]
\[(\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land \]
\[\mathcal{F}_{\text{extra}} \]

\[x_5 = 1\]
CDCL: Search and Clause Learning

\((x_1 \lor x_4) \land (x_3 \lor \overline{x}_4 \lor \overline{x}_5) \land (\overline{x}_3 \lor \overline{x}_2 \lor \overline{x}_4) \land F_{\text{extra}}\)

\(x_5 = 1\)
\(x_2 = 1\)
CDCL: Search and Clause Learning

\[(x_1 \lor x_4) \land (x_3 \lor \overline{x}_4 \lor \overline{x}_5) \land (\overline{x}_3 \lor \overline{x}_2 \lor \overline{x}_4) \land \mathcal{F}_{\text{extra}}\]

\[x_5 = 1\]
\[x_2 = 1\]
CDCL: Search and Clause Learning

\[
\begin{align*}
(x_1 \lor x_4) \land (x_3 \lor \overline{x}_4 \lor \overline{x}_5) \land (\overline{x}_3 \lor \overline{x}_2 \lor \overline{x}_4) \land \\
\mathcal{F}_{\text{extra}}
\end{align*}
\]
CDCL: Search and Clause Learning

\[
\begin{align*}
(x_1 \lor x_4) \land \\
(x_3 \lor \overline{x}_4 \lor \overline{x}_5) \land \\
(\overline{x}_3 \lor \overline{x}_2 \lor \overline{x}_4) \land \\
F_{\text{extra}}
\end{align*}
\]
CDCL: Search and Clause Learning

\[(x_1 \lor x_4) \land (x_3 \lor \overline{x}_4 \lor \overline{x}_5) \land (\overline{x}_3 \lor \overline{x}_2 \lor \overline{x}_4) \land \mathcal{F}_{\text{extra}}\]

\[
\begin{align*}
\text{0} & : x_5 = 1 \\
\text{1} & : x_2 = 1 \\
\text{2} & : \\
\text{6} & : x_1 = 0 \quad x_4 = 1 \quad x_3 = 1 \quad x_3 = 0 \\
\text{7} & : 
\end{align*}
\]
CDCL: Search and Clause Learning

\[ (x_1 \lor x_4) \land (x_3 \lor \overline{x_4} \lor \overline{x_5}) \land (\overline{x_3} \lor \overline{x_2} \lor \overline{x_4}) \land \mathcal{F}_{\text{extra}} \]

\begin{align*}
x_1 &= 0 \\
x_2 &= 1 \\
x_3 &= 0 \\
x_4 &= 1 \\
x_5 &= 1
\end{align*}
CDCL: Search and Clause Learning

\[(x_1 \lor x_4) \land (x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land (\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land \mathcal{F}_{\text{extra}}\]

\(x_5 = 1\)
\(x_2 = 1\)
\(x_1 = 0\)
\(x_4 = 1\)
\(x_3 = 1\)
\(x_3 = 0\)
CDCL: Search and Clause Learning

\[(x_1 \lor x_4) \land (x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land (\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land \mathcal{F}_{\text{extra}}\]

\[0 \quad 1 \quad 2 \quad 6 \quad 7\]

\[x_5 = 1\]
\[x_2 = 1\]
\[x_1 = 0\]
\[x_4 = 1\]
\[x_3 = 1\]
\[x_3 = 0\]

\[(\bar{x}_2 \lor \bar{x}_4 \lor \bar{x}_5)\]
CDCL: Search and Clause Learning

\[
(\bar{x}_1 \lor x_4) \land \\
(x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land \\
(\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land \\
F_{\text{extra}}
\]
CDCL: Search and Clause Learning

\[(x_1 \lor x_4) \land (x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land (\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land F_{\text{extra}}\]

\[(\bar{x}_2 \lor \bar{x}_4 \lor \bar{x}_5)\]
Implication Graphs

Implication graph

- Each node labelled with a variable assignment
  - Note: unit clauses in the input formula are \textit{fixed values before search}
- Decisions are source nodes (fan-in 0)
- Implied $x = 1$ propagated based on a clause $(x \lor x_1 \lor \cdots \lor x_k)$: the children of $x = 1$ are the nodes labelled with $x_1 = 0, \ldots, x_k = 0$. 
Conflict Graphs

Implication graph

- Each node labelled with a variable assignment
- Decisions are source nodes (fan-in 0)
- Implied $x = 1$ propagated based on a clause $(x \lor x_1 \lor \cdots \lor x_k)$: children of $x = 1$ are the nodes labelled with $x_1 = 0, \ldots, x_k = 0$.

Conflict graph

- An implication graph is *conflicting* if it contains both $x = 0$ and $x = 1$ for some variable $x$
  - $x$ is the *conflict variable*
- Conflict graph:
  - A sub-graph of a conflicting implication graph
  - Contains only those nodes from which $x = 0$ or $x = 1$ is reachable
Conflict Analysis: Learning Conflict Clauses

SAT (Lecture 3)
Conflict Analysis: Learning Conflict Clauses

\((\neg x_1 \lor \neg x_3 \lor x_5 \lor x_{17} \lor \neg x_{19})\)

tri-asserting clause
Conflict Analysis: Learning Conflict Clauses

\[(x_{10} \lor \neg x_8 \lor x_{17} \lor \neg x_{19})\]

first unique implication point
Conflict Analysis: Learning Conflict Clauses

\[(x_2 \lor \neg x_4 \lor \neg x_8 \lor x_{17} \lor \neg x_{19})\]

second unique implication point
**Conflict Clauses and Unique Implication Points**

**Conflict Clauses**

- **Conflict cut**: a cut in a conflict graph with
  - at least one of \( x = 0, x = 1 \) on the right-hand \((\text{conflict})\) side of the cut
  - all decisions on the left-hand \((\text{reason})\) side
- **Conflict clause (for a given conflict cut):**
  - Let \( x_1 = 1, \ldots, x_k = 1 \) wlog be the source nodes of the edges cut.
  - Conflict clause associated with the cut: \((\neg x_1 \lor \cdots \lor \neg x_k)\).

**Conflict clauses are entailed by the current clauses**

\[ F \models C \]

for any current set of clauses \( F \) and conflict clause \( C \).

**Unique Implication Points (UIP)**

- A **dominator** node at the decision level of the conflict
  - Every path from the decision at the level of the conflict goes through the node.
Conflict Analysis and Learning Clauses: Key Techniques

1-UIP conflict learning

Learn clauses with *first unique implication points*
- A clause with the UIP *closest to* the conflict

Backjumping (a.k.a non-chronological backtracking)

Backjump over the 1-UIP upto the maximum decision depth of literals in the clause
- UIP conflict clauses are *asserting*:
  After the backjump, *unit propagation* assigns the negation of the UIP

VSIDS

Focused decision heuristics *based on conflict analysis*
- Variable State Independent Decaying Sum
Decision Heuristics: VSIDS [MoskewiczMZZM’01]

- **VSIDS:** Variable State Independent Decaying Sum
- **Intuition:** focus search on recent conflicts
- **Original idea (zChaff):** [MoskewiczMZZM’01]
  - At each conflict, increment scores of involved variables by 1
  - Half all score every 256 conflicts
- **Improved in Minisat:** [EénS’03]
  - At each conflict, increase scores of involved variables by $\delta$, and
  - Let $\delta := 1.05\delta$
Additional Beneficial Techniques in CDCL Solvers

- *Search restarts*
- *Conflict clause minimization*
- *Phase saving*: cheap component caching
- *Watched literals*: efficient implementation of unit propagation
- *Forgetting*: Which learned clauses should be remembered?
- *Preprocessing / inprocessing*: additional inference techniques
Restart

- **Restart**: After a conflict, erase *the whole current assignment*, i.e. backjump to decision level 0

- (Traditional) intuition:
  A means of avoiding heavy-tail behavior  \[ \text{[GomesSC'97; GomesSK'98]} \]
i.e. *getting stuck*

---

Scheduling restarts within CDCL solvers

- Geometric restarts: \( s_{i+1} = \delta s_i \), where \( \delta > 1 \)
e.g. 100, 150, 225, 333, 500, 750, ...

- Luby restarts: \[ \text{[AlistairSZ'93]} \]
e.g. 100, 100, 200, 100, 100, 200, 400, ...

- Adaptive restarts \[ \text{[Biere'08]} \]

- Problem-specific \[ \text{[SinzI'09]} \]

- ...

- Empirical study: \[ \text{[Huang'07]} \]
Restarting

- **Restart:**
  - After a conflict, erase the whole current assignment,
  - i.e. backjump to decision level 0

- (Traditional) intuition:
  A means of avoiding heavy-tail behavior \[GomesSC'97; GomesSK'98\]
  i.e. getting stuck

### Scheduling restarts within CDCL solvers

- **Geometric restarts:** \( s_{i+1} = \delta s_i \), where \( \delta > 1 \)
  - e.g. 100, 150, 225, 333, 500, 750, . . .

- **Luby restarts:** \[AlistairSZ'93\]
  - e.g. 100, 100, 200, 100, 100, 200, 400, . . .

- **Adaptive restarts** \[Biere’08\]

- **Problem-specific** \[SinzI’09\]

- . . .

- **Empirical study:** \[Huang’07\]
Phase Saving

How to choose the value of a decision literal?

Problem

Non-chronological backtracking may erase solutions to unrelated search space components

Solution

*remember how unit propagation has assigned values*

Phase saving:

- Remember the *phase* (value) assigned last time to each variable by *unit propagation*
- Assign decision variables according to the saved phases
- “light-weight component caching”
 Phase Saving [Pipatsrisawat D’07]  

- "light-weight component caching"

- Left: Always assign variable to 0 (as in the Minisat solver)
- Right: Assign variable according to saved phase
Two Watched Literals: Implementing Unit Propagation

\[ x_1 = \ast, \ x_2 = \ast, \ x_3 = \ast, \ x_4 = \ast, \ x_5 = \ast, \ x_6 = \ast \]
Two Watched Literals: Implementing Unit Propagation

\[ x_1 = *, x_2 = *, x_3 = *, x_4 = *, x_5 = 1, x_6 = * \]

\[ \neg x_1 \quad x_2 \quad \neg x_3 \quad \neg x_5 \quad x_6 \]

\[ x_1 \quad \neg x_3 \quad x_4 \quad \neg x_5 \quad \neg x_6 \]
Two Watched Literals: Implementing Unit Propagation

\[ x_1 = \ast, x_2 = \ast, x_3 = 1, x_4 = \ast, x_5 = 1, x_6 = \ast \]
Two Watched Literals: Implementing Unit Propagation

\[ x_1 = *, x_2 = *, x_3 = 1, x_4 = *, x_5 = 1, x_6 = * \]

\[ \neg \neg x_1 \quad x_2 \quad \neg \neg x_3 \quad \neg \neg x_5 \quad x_6 \]

\[ x_1 \quad x_4 \quad \neg \neg x_3 \quad \neg \neg x_5 \quad \neg \neg x_6 \]
Two Watched Literals: Implementing Unit Propagation

\[ x_1 = 1, \ x_2 = \ast, \ x_3 = 1, \ x_4 = \ast, \ x_5 = 1, \ x_6 = \ast \]
Two Watched Literals: Implementing Unit Propagation

\[ x_1 = 1, x_2 = \ast, x_3 = 1, x_4 = \ast, x_5 = 1, x_6 = \ast \]
Two Watched Literals: Implementing Unit Propagation

\[ x_1 = 1, x_2 = \ast, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = \ast \]
Two Watched Literals: Implementing Unit Propagation

\[ x_1 = 1, \ x_2 = 0, \ x_3 = 1, \ x_4 = 0, \ x_5 = 1, \ x_6 = * \]
Two Watched Literals: Implementing Unit Propagation

\[ x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 1 \]
Two Watched Literals: Implementing Unit Propagation

\[ x_1 = 1, \ x_2 = 0, \ x_3 = 1, \ x_4 = 0, \ x_5 = 1, \ x_6 = 1 \]
Two Watched Literals

- Only examine (get in the cache) a clause when both a watch pointer gets falsified & the other one is not satisfied
- While backjumping, *just unassign variables* — no changes to the pointers
- Conflict clauses are instrumented with watch pointers
Forgetting Conflict Clauses

Problem

Learning more and more conflict clause will suffocate the solver

Solution: *forget* conflict clauses heuristically

- Very important in practice
- Idea:
  - Start with storage space for a small number of conflict clauses
  - Increase storage space gradually (as long as search does not terminate)

Which conflict clauses to forget?
Forgetting Conflict Clauses: Heuristics

Clause-length based
Keep shortest clauses

Activity-based
Keep clauses that are frequently involved in conflict analysis
- Maintain heuristic score for each learned clause
- Increment score by 1 if variable in the clause occurs in a new conflict clause/in the derivation of a conflict clause

Literal blocks distance
Glucose [AudemardS'09]

LBD value of a learned clause
= the number of unique decision levels among literals in the clause
- Keep clauses with small LBD
- Intuition: small LBD
  ⇨ used more for propagation ⇨ used more for conflicts
LBD v Clause-Length Based Forgetting

![Graph showing the cumulative distribution function (CDF) of measure against measure (length or dependencies level). The graph compares LBD, used in propagations, Length, used in propagations, LBD, used in conflict analysis, and Length, used in conflict analysis. The x-axis represents the measure (length or dependencies level), and the y-axis represents the CDF of measure. The graph indicates the percentage of measures up to a certain level.](image-url)
What is Important in Practice?

- Principled experimental study on the importance of different techniques [KatebiSM’11]
- DPLL: the CDCL solver Minisat without clause learning, VSIDS, restarts, watched literals, ...
What is Important in Practice?

- Principled experimental study on the importance of different techniques [KatebiSM’11]
- 2WL: two watched literal scheme
What is Important in Practice?

- Principled experimental study on the importance of different techniques [KatebiSM’11]
- PHS: phase saving
Principled experimental study on the importance of different techniques [KatebiSM’11]

RST: restarts
What is Important in Practice?

- Principled experimental study on the importance of different techniques [KatebiSM’11]
- VSIDS decision heuristics
What is Important in Practice?

- Principled experimental study on the importance of different techniques [KatebiSM’11]
- CL: *only* learnt clauses (no VSIDS, 2WL, PHS,...) + minimization
What is Important in Practice?

- Principled experimental study on the importance of different techniques [KatebiSM’11]
- CDCL: Minisat (all of CL, VSIDS, 2WL, PHS, ...)

---

**Scalability**

- DPLL/2WL/PHS/RST
- VSIDS
- CL
- CDCL
What is Important in Practice?

- Principled experimental study on the importance of different techniques [KatebiSM’11]
- -CL: All but learned clauses
What is Important in Practice?

- Principled experimental study on the importance of different techniques [KatebiSM’11]
- VSIDS: a more classical counting/occurrence based decision heuristic (DLIS)
What is Important in Practice?

- Principled experimental study on the importance of different techniques [KatebiSM’11]
- -2WL: no two watched literals (using counter-based unit propagation)
What is Important in Practice?

- Principled experimental study on the importance of different techniques [KatebiSM’11]
- RST: no restarts
Principled experimental study on the importance of different techniques [KatebiSM’11]:

- PHS: no phase saving
A (Partial) History of CDCL SAT Solvers

Grasp  ~1996
- Clause learning, non-chronological backtracking, UIPs

zChaff  ~2001
- VSIDS, (watched literals), 1-UIPs most efficient

MiniSAT http://minisat.se/  ~2003–
- The most applied CDCL SAT solver
- Clean code, easy to modify, described well on paper.
- Conflict clause minimization, (clause forgetting), SatElite 2016

Glucose (based on MiniSAT) ~2009–
http://www.labri.fr/perso/lsimon/glucose/
- Influential clause forgetting heuristics

- inprocessing, ...

http://fmv.jku.at/lingeling/

Several others left unmentioned here!
Resolution rule:
\[ (A \lor B) \text{ from } (x \lor A) \text{ and } (\neg x \lor B) \]
Yields a complete proof system for unsatisfiability of CNFs

DPLL and *Tree-like Resolution* are polynomially equivalent
  - Well-known & easy to see
  - DPLL search trees \( \approx \) regular tree-like resolution proofs
The following are polynomially equivalent:

- CDCL with
  - (unlimited) restarts,
  - standard restart schemes (polynomial-spaced restarts),
  - asserting clause learning schemes, e.g., using the 1-UIP learning scheme, and
  - *optimal* decision heuristics.

- (unrestricted) Resolution.

CDCL can derive each of the *needed* resolvents in $O(n^4)$.

[Agent Pipatsrisawat, Darwiche, AIJ'11]

Open question:
Does CDCL require restarts to polynomially simulate Resolution?
The following are polynomially equivalent:

- CDCL with
  - (unlimited) restarts,
  - standard restart schemes (polynomial-spaced restarts),
  - asserting clause learning schemes, e.g., using the 1-UIP learning scheme, and
  - optimal decision heuristics.

- (unrestricted) Resolution.

CDCL can derive each of the needed resolvents in $O(n^4)$.
[Chai et al., AIJ'11]

Open question:
Does CDCL require restarts to polynomially simulate Resolution?
Conflict Learning and Resolution

Conflict clauses can be derived with *Trivial Resolution* from the current clauses $F$.

**Trivial Resolution**

A Resolution derivation $(C_1, C_2, \ldots, C_l)$ is *trivial* iff:

- all variables resolved upon are distinct; and
- for each $i = 3..k$, the clause $C_i$ is
  - an initial clause (in $F$), or
  - derived from $C_{i-1}$ and an initial clause
Conflict Learning and Resolution

\((x_1 \lor x_4) \land (x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land (\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land \mathcal{F}_{\text{extra}}\)

\(x_1 = 0\)
\(x_4 = 1\)
\(x_2 = 1\)
\(x_3 = 0\)
\(x_5 = 1\)

\((\bar{x}_2 \lor \bar{x}_4 \lor \bar{x}_5)\)
Conflict Learning and Resolution

\[ (x_1 \lor x_4) \land (x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land (\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land F_{\text{extra}} \]

\[ \begin{align*}
    x_1 &= 0 \\
    x_2 &= 1 \\
    x_3 &= 1 \\
    x_4 &= 1 \\
    x_5 &= 1
\end{align*} \]

\[ \frac{(\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land (x_3 \lor \bar{x}_4 \lor \bar{x}_5)}{(\bar{x}_2 \lor \bar{x}_4 \lor \bar{x}_5)} \]
Conflict Learning and Resolution

\[(x_1 \lor x_4) \land (x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land (\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land \mathcal{F}_{\text{extra}}\]

\[
\begin{align*}
(\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) &\land (x_3 \lor \bar{x}_4 \lor \bar{x}_5) \\
(x_2 \lor \bar{x}_4 \lor \bar{x}_5) &\land (x_1 \lor x_4) \\
(x_1 \lor \bar{x}_2 \lor \bar{x}_5) &\land (x_1 \lor \bar{x}_2 \lor \bar{x}_5)
\end{align*}
\]
Conflict Learning and Resolution

\[(x_1 \lor x_4) \land (x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land (\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land \mathcal{F}_{\text{extra}}\]

\[F_{\text{extra}}\]

\begin{align*}
(\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) & \quad (x_3 \lor \bar{x}_4 \lor \bar{x}_5) \\
(\bar{x}_2 \lor \bar{x}_4 \lor \bar{x}_5) & \quad (x_1 \lor \bar{x}_4) \\
(\bar{x}_1 \lor \bar{x}_2 \lor \bar{x}_5) & \quad (x_1 \lor x_4) \\
\end{align*}
Conflict Learning and Resolution

\[(x_1 \lor x_4) \land (x_3 \lor \overline{x}_4 \lor \overline{x}_5) \land (\overline{x}_3 \lor \overline{x}_2 \lor \overline{x}_4) \land \mathcal{F}_{\text{extra}}\]

\[\begin{align*}
&x_1 = 0, \\
x_2 = 1, \\
x_3 = 0, \\
x_4 = 1, \\
x_5 = 1, \\
\end{align*}\]

\[\begin{align*}
&(x_1 \lor \overline{x}_2 \lor \overline{x}_5) \\
&(\overline{x}_2 \lor \overline{x}_4 \lor \overline{x}_5) \\
&(x_1 \lor \overline{x}_2 \lor \overline{x}_5) \\
&(x_3 \lor \overline{x}_4 \lor \overline{x}_5) \\
&(x_3 \lor \overline{x}_4 \lor \overline{x}_5) \\
\end{align*}\]
Conflict Learning and Resolution

\[
(x_1 \lor x_4) \land \\
(x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land \\
(\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land \\
\mathcal{F}_{\text{extra}}
\]

\[
(x_1 \lor \bar{x}_2 \lor \bar{x}_5) \\
(x_3 \lor \bar{x}_4 \lor \bar{x}_5) \\
\bar{x}_2 \lor \bar{x}_4 \lor \bar{x}_5 \\
\bar{x}_1 \lor \bar{x}_2 \lor \bar{x}_5
\]
Conflict Learning and Resolution

\[(x_1 \lor x_4) \land (x_3 \lor \overline{x}_4 \lor \overline{x}_5) \land (\overline{x}_3 \lor \overline{x}_2 \lor \overline{x}_4) \land F_{\text{extra}}\]
The Practical SAT-based Approach: Next Time

1. Encoding: how to represent your problem in CNF
   - Possibly through many steps via more high-level representations
2. Preprocessing: automated simplification (rewriting) of the CNF
   - reformulating the encoding
3. SAT solving: how to determine satisfiability fast in practice
The Practical SAT-based Approach: Next Time

1. Encoding: how to represent your problem in CNF
   - Possibly through many steps via more high-level representations
2. Preprocessing: automated simplification (rewriting) of the CNF
   - reformulating the encoding
3. SAT solving: how to determine satisfiability fast in practice
The Practical SAT-based Approach: Next Time

1. Encoding: how to represent your problem in CNF
   - Possibly through many steps via more high-level representations
2. Preprocessing: automated simplification (rewriting) of the CNF
   - reformulating the encoding
3. SAT solving: how to determine satisfiability fast in practice
Summary

Take-home message

CDCL: conflict-driven clause learning
- Basis of the currently most successful SAT solvers for various real-world domains
- CDCL ≠ DPLL
- Key components: 1-UIP learning, forgetting, restarts, VSIDS, . . .
- Connections between SAT solvers and Resolution refinements

Study goals

- CDCL search
  - Clause learning, learning schemes (1-UIP)
  - VSIDS, clause forgetting schemes, restarts

Next time: preprocessing

Speeding up (CDCL) SAT solvers by preprocessing CNF formulas before core search