CS257: Introduction to Automated Reasoning
DPLL and CDCL
Plan

• DPLL
  - Abstract DPLL
• CDCL (DP Chapter 2)
  - Abstract CDCL
  - Implication graphs

* Some of the slides today are contributed by Clark Barrett, Cesare Tinelli, and Emina Torlak.
The Original DPLL Procedure

- Modern SAT solvers are based on the DPLL procedure
- DPLL tries to build incrementally a satisfying truth assignment \( M \) for a CNF formula \( F \)
- \( M \) is grown by
  - deducing the truth value of a literal from \( M \) and \( F \), or
  - guessing a truth value
- If a wrong guess for a literal leads to an inconsistency, the procedure backtracks and tries the opposite value
DPLL as a Proof System

To facilitate a deeper look at DPLL, we present DPLL as a proof system called Abstract DPLL.

The procedure described next is a re-elaboration of those in [1,2].

Abstract DPLL: A Proof System for DPLL

States:

\[ \text{Fail} \quad \text{or} \quad \langle M, \Delta \rangle \]

where

- \( M \) is a **sequence of literals** and **decision points** denoting a **partial truth assignment**
- \( \Delta \) is a **set of clauses** denoting a CNF formula

**Def.** If \( M = M_0 \bullet M_1 \bullet \cdots \bullet M_n \) where each \( M_i \) contains no decision points

- \( M_i \) is **decision level** \( i \) of \( M \)
- \( M[i] \overset{\text{def}}{=} M_0 \bullet \cdots \bullet M_i \)
Abstract DPLL: A Proof System for DPLL

States:

Fail or \( \langle M, \Delta \rangle \)

Initial state:

- \( \langle (), \Delta_0 \rangle \), where \( \Delta_0 \) is to be checked for satisfiability

Expected final states:

- Fail if \( \Delta_0 \) is unsatisfiable
- \( \langle M, \Delta' \rangle \) otherwise, where
  - \( \Delta' \) is equivalent to \( \Delta_0 \) and
  - \( M \) satisfies \( \Delta' \)
Some clause terminology

Given a partial assignment: \(\{p_1 \mapsto 1, p_2 \mapsto 0, p_4 \mapsto 1\}\)

- \(\{p_1, p_3, \neg p_4\}\) is **satisfied**
- \(\{\neg p_1, p_2\}\) is **conflicting**
- \(\{\neg p_1, p_3, \neg p_4\}\) is **unit**
- \(\{\neg p_1, p_3, p_5\}\) is **unresolved**
- \(p_1\) is **assigned**
- \(p_3\) is **unassigned**
Some clause terminology

Given a partial assignment: \{p_1 \mapsto 1, p_2 \mapsto 0, p_4 \mapsto 1\}

- \{p_1, p_3, \neg p_4\} is satisfied
- \{\neg p_1, p_2\} is conflicting
- \{\neg p_1, p_3, \neg p_4\} is unit
- \{\neg p_1, p_3, p_5\} is unresolved
- \(p_1\) is assigned
- \(p_3\) is unassigned

One characteristic of DPLL-style SAT solvers is that given a partial assignment under which a clause becomes unit, it must be extended so that it satisfies the unassigned literal of this clause. Following this requirement is necessary but not sufficient for satisfying the formula.
Abstract DPLL: Proof Rules for the Original DPLL

Extending the assignment

\[
\begin{array}{c}
\{l_1, \ldots, l_n, l\} \in \Delta \\
\neg l_1, \ldots, \neg l_n \in M \\
l, \neg l \notin M \\
\end{array}
\]

\[M := M \cup \{l\}\]  \hspace{1cm} \text{(Propagate)}

Deduce the values of unassigned literals in unit clauses.
The clause \(\{l_1, \ldots, l_n, l\}\) is called the \textbf{antecedent clause} of \(l\). Denoted by \text{Antecedent}(l).

\textbf{Note:} When convenient, treat \(M\) as a set
Extending the assignment

\[
\begin{array}{c}
\{l_1, \ldots, l_n, l\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad l, \neg l \notin M \\
M := M \cup \{l\} \\
\end{array}
\] (Propagate)

Deduce the values of unassigned literals in unit clauses. The clause \(\{l_1, \ldots, l_n, l\}\) is called the **antecedent clause** of \(l\). Denoted by \text{Antecedent}(l).

**Note:** When convenient, treat \(M\) as a set

\[
\begin{array}{c}
l \text{ literal of } \Delta \quad \neg l \text{ not literal of } \Delta \quad l, \neg l \notin M \\
M := M \cup \{l\} \\
\end{array}
\] (Pure)

Make a pure literal true.
Abstract DPLL: Proof Rules for the Original DPLL

Extending the assignment

\[
\frac{\{l_1, \ldots, l_n, l\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad l, \neg l \notin M}{M := M \cup \{l\}} \quad \text{(Propagate)}
\]

Deduce the values of unassigned literals in unit clauses. The clause \(\{l_1, \ldots, l_n, l\}\) is called the **antecedent clause** of \(l\). Denoted by \text{Antecedent}(l).

Note: When convenient, treat \(M\) as a set

\[
\frac{l \text{ literal of } \Delta \quad \neg l \text{ not literal of } \Delta \quad l, \neg l \notin M}{M := M \cup \{l\}} \quad \text{(Pure)}
\]

Make a pure literal true.

\[
\frac{l \in \text{Lits}(\Delta) \quad l, \neg l \notin M}{M := M \cup \{l\}} \quad \text{(Decide)}
\]

Guess a truth value for an unassigned literal.

Note: \(\text{Lits}(\Delta) \overset{\text{def}}{=} \{l \mid l \text{ literal of } \Delta\} \cup \{-l \mid l \text{ literal of } \Delta\}\)
Proof Rules for the Original DPLL

Repairing the assignment

\[
\begin{array}{c}
\{l_1, \ldots, l_n\} \in \Delta \\
\neg l_1, \ldots, \neg l_n \in M \\
M = M' \cdot I \cdot N \\
\bullet \notin N
\end{array}
\]

(Backtrack)

\[M := M' \cdot \neg I\]

There is a conflicting clause and there is a decision point that we can backtrack to. Backtrack to the last decision point and try the opposite value for the literal than last time.

Note: Last premise of Backtrack enforces chronological backtracking
Proof Rules for the Original DPLL

**Repairing the assignment**

\[
\begin{align*}
\{l_1, \ldots, l_n\} & \in \Delta \\
\neg l_1, \ldots, \neg l_n & \in M \\
\neg l & \notin N \\

M & := M' \land N \\

\text{(Backtrack)}
\end{align*}
\]

There is a **conflicting clause** and there is a decision point that we can backtrack to.
Backtrack to the last decision point and try the **opposite value** for the literal than last time.

**Note:** Last premise of Backtrack enforces **chronological** backtracking

\[
\begin{align*}
\{l_1, \ldots, l_n\} & \in \Delta \\
\neg l_1, \ldots, \neg l_n & \in M \\
\neg l & \notin M \\

\text{(Fail)}
\end{align*}
\]

There is a **conflicting clause** and there are no decision points to backtrack to.
So the formula is unsatisfiable.
Proof Rules for the Original DPLL

\[
\frac{\{l_1, \ldots, l_n, l\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad l, \neg l \notin M}{M := M \cdot l} \quad \text{(Propagate)}
\]

\[
\frac{l \text{ literal of } \Delta \quad \neg l \text{ not literal of } \Delta \quad l, \neg l \notin M}{M := M \cdot l} \quad \text{(Pure)}
\]

\[
\frac{l \in \text{Lits}(\Delta) \quad l, \neg l \notin M}{M := M \cdot l} \quad \text{(Decide)}
\]

\[
\frac{\{l_1, \ldots, l_n\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad M = M' \cdot l \quad N \quad \bullet \notin N}{M := M' \cdot \neg l} \quad \text{(Backtrack)}
\]

\[
\frac{\{l_1, \ldots, l_n\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad \bullet \notin M}{\text{Fail}} \quad \text{(Fail)}
\]

Note: In DPLL, there are no rules to update \( \Delta \), the set of clauses. Such rules are present in CDCL as we will see.
DPLL execution example

\[ \Delta_0 := \{\{1, \neg 2\}, \{\neg 1, \neg 2\}, \{2, 3\}, \{\neg 3, 2\}, \{1, 4\}\} \]

note: We abbreviate \( p_n \) as \( n \).

\[
\begin{array}{ccc}
 M & \Delta & \text{rule} \\
 \{1, \neg 2\}, \{\neg 1, \neg 2\}, \{2, 3\}, \{\neg 3, 2\}, \{1, 4\} \\
\end{array}
\]

\[
\begin{array}{c}
\{l_1, \ldots, l_n\} \in \Delta \\
\neg l_1, \ldots, \neg l_n \in M \\
\end{array} \quad \begin{array}{c}
l, \neg l \notin M \\
\end{array} \quad \begin{array}{c}
M := M \perp l \\
\end{array} \quad \text{(Propagate)}
\]

\[
\begin{array}{c}
l \text{ literal of } \Delta \\
\neg l \text{ not literal of } \Delta \\
\end{array} \quad \begin{array}{c}
l, \neg l \notin M \\
\end{array} \quad \begin{array}{c}
M := M \perp l \\
\end{array} \quad \text{(Pure)}
\]

\[
\begin{array}{c}
\{l_1, \ldots, l_n\} \in \Delta \\
\neg l_1, \ldots, \neg l_n \in M \\
\end{array} \quad \begin{array}{c}
M = M' \perp l \in N \\
\bullet \notin N \\
\end{array} \quad \begin{array}{c}
M := M' \neg l \\
\end{array} \quad \text{(Backtrack)}
\]

\[
\begin{array}{c}
l \in \text{Lits}(\Delta) \\
\end{array} \quad \begin{array}{c}
l, \neg l \notin M \\
\end{array} \quad \begin{array}{c}
M := M \perp l \\
\end{array} \quad \text{(Decide)}
\]

\[
\begin{array}{c}
\{l_1, \ldots, l_n\} \in \Delta \\
\neg l_1, \ldots, \neg l_n \in M \\
\end{array} \quad \begin{array}{c}
\bullet \notin M \\
\end{array} \quad \text{Fail}
\]

\[
\begin{array}{c}
\{l_1, \ldots, l_n\} \in \Delta \\
\neg l_1, \ldots, \neg l_n \in M \\
\end{array} \quad \begin{array}{c}
\bullet \notin M \\
\end{array} \quad \text{Fail}
\]

October 11, 2023
DPLL execution example

\[
\Delta_0 := \{\{1, -2\}, \{-1, -2\}, \{2, 3\}, \{-3, 2\}, \{1, 4\}\}
\]

note: We abbreviate \(p_n\) as \(n\).

<table>
<thead>
<tr>
<th>(M)</th>
<th>(\Delta)</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Pure</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\{l_1, \ldots, l_n, l\} \in \Delta & \quad \neg l_1, \ldots, \neg l_n \in M & l, \neg l \notin M \\
M := M \mathbf{I} & & & (\text{Propagate}) \\
\text{\(l\) literal of} \ \Delta & \quad \neg l \text{ not literal of} \ \Delta & l, \neg l \notin M \\
M := M \mathbf{I} & & & (\text{Pure}) \\
\{l_1, \ldots, l_n\} \in \Delta & \quad \neg l_1, \ldots, \neg l_n \in M & M = M' \mathbf{\bullet} \ N & l \notin N \\
M := M' \mathbf{\bullet} \ l & & & (\text{Backtrack}) \\
\end{align*}
\]

\[
\begin{align*}
l \in \text{Lits}(\Delta) & \quad l, \neg l \notin M & M := M \mathbf{\bullet} \ l & (\text{Decide}) \\
\{l_1, \ldots, l_n\} \in \Delta & \quad \neg l_1, \ldots, \neg l_n \in M & l \notin M & \mathbf{\bullet} \notin M \\
\text{Fail} & & & (\text{Fail}) \\
\end{align*}
\]
DPLL execution example

\[ \Delta_0 := \{\{1, \neg 2\}, \{-1, \neg 2\}, \{2, 3\}, \{-3, 2\}, \{1, 4\}\} \]

note: We abbreviate \( p_n \) as \( n \).

<table>
<thead>
<tr>
<th>( M )</th>
<th>( \Delta )</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 ( \bullet 1 )</td>
<td>{1, \neg 2}, {-1, \neg 2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Pure</td>
</tr>
<tr>
<td>4 ( \bullet 1 )</td>
<td>{1, \neg 2}, {-1, \neg 2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Decide</td>
</tr>
</tbody>
</table>

\( \{l_1, \ldots, l_n, l\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad l, \neg l \notin M \)
\[ M := M \bullet l \] (Propagate)

\( l \) literal of \( \Delta \) \quad \neg l \) not literal of \( \Delta \) \quad l, \neg l \notin M \)
\[ M := M \bullet l \] (Pure)

\( \{l_1, \ldots, l_n\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad \neg \in M \)
\[ M := M \bullet \neg l \] (Backtrack)

\( l \in \text{Lits}(\Delta) \quad l, \neg l \notin M \)
\[ M := M \bullet l \] ( Decide)

\( \{l_1, \ldots, l_n\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad \bullet \notin M \)
\[ \text{Fail} \] (Fail)
DPLL execution example

\[ \Delta_0 := \{ \{1, -2\}, \{-1, -2\}, \{2, 3\}, \{-3, 2\}, \{1, 4\} \} \]

*note:* We abbreviate \( p_n \) as \( n \).

<table>
<thead>
<tr>
<th>( M )</th>
<th>( \Delta )</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 4 )</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Pure</td>
</tr>
<tr>
<td>( 4 \bullet 1 )</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Decide</td>
</tr>
<tr>
<td>( 4 \bullet 1 \rightarrow 2 )</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Propagate</td>
</tr>
</tbody>
</table>

**Propagate**

\[
\{l_1, \ldots, l_n, l\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad l, \neg l \notin M
\]

\[
M := M \bullet l
\]

**Pure**

\[
l \text{ literal of } \Delta \quad \neg l \text{ not literal of } \Delta \quad l, \neg l \notin M
\]

\[
M := M \bullet l
\]

**Decide**

\[
l \in \text{ Lits}(\Delta) \quad \neg l_1, \ldots, \neg l_n \in M \quad \bullet \notin M
\]

\[
M := M \bullet l
\]

**Fail**

\[
\{l_1, \ldots, l_n\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad \bullet \notin M
\]

\[
\text{Fail}
\]
DPLL execution example

\[ \Delta_0 := \{\{1, -2\}, \{-1, -2\}, \{2, 3\}, \{-3, 2\}, \{1, 4\}\} \]

**Note:** We abbreviate \( p_n \) as \( n \).

<table>
<thead>
<tr>
<th>( M )</th>
<th>( \Delta )</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 ( \bullet ) 1</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>4 ( \bullet ) 1</td>
</tr>
<tr>
<td>4 ( \bullet ) 1 -2</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Pure</td>
</tr>
<tr>
<td>4 ( \bullet ) 1 -2 3</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Propagate</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\{l_1, \ldots, l_n, l\} \in \Delta & \quad \neg l_1, \ldots, \neg l_n \in M & l, \neg l \notin M \quad \text{(Propagate)} \\
M := M \bullet l
\end{align*}
\]

\[
\begin{align*}
l \quad \text{literal of } \Delta & \quad \neg l \quad \text{not literal of } \Delta & l, \neg l \notin M \quad \text{(Pure)} \\
M := M \bullet l
\end{align*}
\]

\[
\begin{align*}
\{l_1, \ldots, l_n\} \in \Delta & \quad \neg l_1, \ldots, \neg l_n \in M & M = M' \bullet l \quad \bullet \notin N \quad \text{(Backtrack)} \\
M := M' \bullet \neg l
\end{align*}
\]

\[
\begin{align*}
l \in \text{Lits}(\Delta) & \quad l, \neg l \notin M \quad \text{(Decide)} \\
M := M \bullet l
\end{align*}
\]

\[
\begin{align*}
\{l_1, \ldots, l_n\} \in \Delta & \quad \neg l_1, \ldots, \neg l_n \in M & \bullet \notin M \quad \text{(Fail)} \\
\text{Fail}
\end{align*}
\]
DPLL execution example

\[ \Delta_0 := \{ \{1, -2\}, \{-1, -2\}, \{2, 3\}, \{-3, 2\}, \{1, 4\} \} \]

note: We abbreviate \( p_n \) as \( n \).

<table>
<thead>
<tr>
<th>( M )</th>
<th>( \Delta )</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td></td>
</tr>
<tr>
<td>4 \bullet 1</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Pure</td>
</tr>
<tr>
<td>4 \bullet 1 \neg 2</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Decide</td>
</tr>
<tr>
<td>4 \bullet 1 \neg 2</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Propagate</td>
</tr>
<tr>
<td>4 \neg 1 3</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Propagate</td>
</tr>
<tr>
<td>4 \neg 1</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Backtrack</td>
</tr>
</tbody>
</table>

\[
\frac{\{l_1, \ldots, l_n, l\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad \neg l \notin M}{M := M \bullet l} \quad \text{(Pure)}
\]

\[
\frac{l \text{ literal of } \Delta \quad \neg l \text{ not literal of } \Delta \quad l, \neg l \notin M}{M := M \bullet l} \quad \text{(Propagate)}
\]

\[
\frac{l \in \text{Lits}(\Delta) \quad \neg l, \neg l \notin M}{M := M \bullet l} \quad \text{(Decide)}
\]

\[
\frac{\{l_1, \ldots, l_n\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M}{\bullet \notin M \quad \text{Fail}}
\]

October 11, 2023
DPLL execution example

\[ \Delta_0 := \{\{1, \neg 2\}, \{-1, \neg 2\}, \{2, 3\}, \{-3, 2\}, \{1, 4\}\} \]

note: We abbreviate \( p_n \) as \( n \).

<table>
<thead>
<tr>
<th>( M )</th>
<th>( \Delta )</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>( {1, \neg 2}, {-1, \neg 2}, {2, 3}, {-3, 2}, {1, 4} )</td>
<td>Pure</td>
</tr>
<tr>
<td>4 ( \bullet ) 1</td>
<td>( {1, \neg 2}, {-1, \neg 2}, {2, 3}, {-3, 2}, {1, 4} )</td>
<td>Decide</td>
</tr>
<tr>
<td>4 ( \bullet ) 1 ( \neg ) 2</td>
<td>( {1, \neg 2}, {-1, \neg 2}, {2, 3}, {-3, 2}, {1, 4} )</td>
<td>Propagate</td>
</tr>
<tr>
<td>4 ( \bullet ) 1 ( \neg ) 2 \ 3</td>
<td>( {1, \neg 2}, {-1, \neg 2}, {2, 3}, {-3, 2}, {1, 4} )</td>
<td>Propagate</td>
</tr>
<tr>
<td>4 ( \neg ) 1</td>
<td>( {1, \neg 2}, {-1, \neg 2}, {2, 3}, {-3, 2}, {1, 4} )</td>
<td>Backtrack</td>
</tr>
<tr>
<td>4 ( \neg ) 1 ( \neg ) 2</td>
<td>( {1, \neg 2}, {-1, \neg 2}, {2, 3}, {-3, 2}, {1, 4} )</td>
<td>Propagate</td>
</tr>
<tr>
<td>4 ( \neg ) 1 ( \neg ) 2 \ 3</td>
<td>( {1, \neg 2}, {-1, \neg 2}, {2, 3}, {-3, 2}, {1, 4} )</td>
<td>Propagate</td>
</tr>
</tbody>
</table>
DPLL execution example

\[ \Delta_0 := \{ \{1, -2\}, \{-1, -2\}, \{2, 3\}, \{-3, 2\}, \{1, 4\} \} \]

note: We abbreviate \( p_n \) as \( n \).

<table>
<thead>
<tr>
<th>( M )</th>
<th>( \Delta )</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Pure</td>
</tr>
<tr>
<td>4 \cdot 1</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Decide</td>
</tr>
<tr>
<td>4 \cdot 1 \cdot -2</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Propagate</td>
</tr>
<tr>
<td>4 \cdot 1 \cdot -2 \cdot 3</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Propagate</td>
</tr>
<tr>
<td>4 \cdot -1 \cdot -2</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Backtrack</td>
</tr>
<tr>
<td>4 \cdot -1 \cdot -2 \cdot -3</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Propagate</td>
</tr>
</tbody>
</table>

\[ \{l_1, \ldots, l_n, l\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad l, \neg l \notin M \]

(Propagate)

\[ I \in \text{Lits}(\Delta) \quad l, \neg l \notin M \]

(Decide)

\[ \{l_1, \ldots, l_n\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad \bullet \notin M \]

(Fail)

\[ \{l_1, \ldots, l_n\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \]

(Backtrack)
DPLL execution: exercise

\[ \Delta_0 := \{\{1, \neg 2\}, \{\neg 1, \neg 2\}, \{2, 3\}, \{\neg 3, 2\}, \{1, 4\}\} \]

<table>
<thead>
<tr>
<th>(M)</th>
<th>(\Delta)</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1, 2}, {\neg 1, \neg 2}, {2, 3}, {\neg 3, 2}, {1, 4}</td>
<td>{1, 2}, {\neg 1, \neg 2}, {2, 3}, {\neg 3, 2}, {1, 4}</td>
<td>Pure</td>
</tr>
<tr>
<td>{1, 2}, {\neg 1, \neg 2}, {2, 3}, {\neg 3, 2}, {1, 4}</td>
<td>{1, 2}, {\neg 1, \neg 2}, {2, 3}, {\neg 3, 2}, {1, 4}</td>
<td>Decide</td>
</tr>
</tbody>
</table>
| \{1, 2\}, \{\neg 1, \neg 2\}, \{2, 3\}, \{\neg 3, 2\}, \{1, 4\} | \{1, 2\}, \{\neg 1, \neg 2\}, \{2, 3\}, \{\neg 3, 2\}, \{1, 4\} | ...

How many steps (i.e., \# of rule applications) does it take to derive **Fail**?

Work with your neighbor. Submit your answer at

https://pollev.com/andreww095

| \{l_1, \ldots, l_n\} \in \Delta | \neg l_1, \ldots, \neg l_n \in M | l, \neg l \in M | (Propagate) |
| \hline | \hline | \hline | \hline |
| M := M \cup l | l \text{ literal of } \Delta | \neg l \text{ not literal of } \Delta | l, \neg l \notin M | (Pure) |
| M := M \cup l | l \notin \text{Lits}(\Delta) | l, \neg l \notin M | (Decide) |
| l \notin M | (Fail) |
| \{l_1, \ldots, l_n\} \in \Delta | \neg l_1, \ldots, \neg l_n \in M | \bullet \notin M | (Fail) |
| M := M \cup \{l\} | l \notin M | (Backtrack) |
DPLL execution: exercise

\[ \Delta_0 := \{\{1, -2\}, \{-1, -2\}, \{2, 3\}, \{-3, 2\}, \{1, 4\}\} \]

<table>
<thead>
<tr>
<th>(M)</th>
<th>(\Delta)</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}\</td>
<td>(Pure)</td>
<td></td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\{l_1, \ldots, l_n, l\} &\in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad l, \neg l \notin M \\
M &:= M \lnot l
\end{align*}
\]

(Propagate)

\[
\begin{align*}
\text{\(l\) literal of \(\Delta\)} \quad \neg l &\not\in \text{\(\Delta\)} \quad l, \neg l \notin M \\
M &:= M \lnot l
\end{align*}
\]

(Pure)

\[
\begin{align*}
\{l_1, \ldots, l_n\} &\in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad l \not\in M \\
M &:= M \bullet l
\end{align*}
\]

(Decide)

\[
\begin{align*}
l &\in \text{\(\text{Lits}(\Delta)\)} \quad l, \neg l \notin M \\
M &:= M \bullet l
\end{align*}
\]

(Fail)

\[
\begin{align*}
\{l_1, \ldots, l_n\} &\in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad \bullet \not\in M \\
\text{Fail}
\end{align*}
\]

(Backtrack)

\[
\begin{align*}
\{l_1, \ldots, l_n\} &\in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad M = M' \bullet l \bullet N \quad \bullet \notin N \\
M &:= M' \lnot l
\end{align*}
\]

October 11, 2023 CS257
### DPLL execution: exercise

\[ \Delta_0 := \{ \{1, -2\}, \{-1, -2\}, \{2, 3\}, \{-3, 2\}, \{1, 4\} \} \]

<table>
<thead>
<tr>
<th>(M)</th>
<th>(\Delta)</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Pure</td>
</tr>
</tbody>
</table>

---

\[ \{l_1, \ldots, l_n, l\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad l, \neg l \notin M \]

\[ M := M \cup l \quad \text{(Propagate)} \]

\[ l \text{ literal of } \Delta \quad \neg l \text{ not literal of } \Delta \quad l, \neg l \notin M \]

\[ M := M \cup l \quad \text{(Pure)} \]

\[ \{l_1, \ldots, l_n\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad \neg l \notin N \]

\[ M := M \cdot l \quad \text{(Fail)} \]

\[ l \in \text{Lits}(\Delta) \quad \neg l \notin M \]

\[ M := M \cup l \quad \text{(Decide)} \]

\[ \{l_1, \ldots, l_n\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad \bullet \notin M \]

\[ \text{Fail} \]

\[ \{l_1, \ldots, l_n\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad \bullet \notin M \]

\[ \text{Fail} \]
DPLL execution: exercise

\[ \Delta_0 := \{\{1, -2\}, \{-1, -2\}, \{2, 3\}, \{-3, 2\}, \{1, 4\}\} \]

<table>
<thead>
<tr>
<th>(M)</th>
<th>(\Delta)</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4)</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}\</td>
<td>Pure</td>
</tr>
<tr>
<td>(4 \bullet -3)</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}\</td>
<td>Decide</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\{l_1, \ldots, l_n\} \in \Delta &\quad \neg l_1, \ldots, \neg l_n \in M & l, \neg l \notin M \\
M &:= M \bullet l
\end{align*}
\]

(Propagate)

\[
\begin{align*}
l \text{ literal of } \Delta &\quad \neg l \text{ not literal of } \Delta & l, \neg l \notin M \\
M &:= M \bullet l
\end{align*}
\]

(Pure)

\[
\begin{align*}
\{l_1, \ldots, l_n\} \in \Delta &\quad \neg l_1, \ldots, \neg l_n \in M & M = M' \bullet l \quad \bullet \notin N \\
M &:= M' \bullet \neg l
\end{align*}
\]

(Backtrack)
DPLL execution: exercise

\( \Delta_0 := \{\{1, -2\}, \{-1, -2\}, \{2, 3\}, \{-3, 2\}, \{1, 4\}\} \)

<table>
<thead>
<tr>
<th>(M)</th>
<th>(\Delta)</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Pure</td>
</tr>
<tr>
<td>4 (\bullet) -3</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Decide</td>
</tr>
<tr>
<td>4 (\bullet) -3 2</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>Propagate</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\{l_1, \ldots, l_n, l\} &\in \Delta & \neg l_1, \ldots, \neg l_n &\in M & l, \neg l &\notin M & (\text{Propagate}) \\
M := M l &
\end{align*}
\]

\[
\begin{align*}
l &\text{ literal of } \Delta & \neg l &\text{ not literal of } \Delta & l, \neg l &\notin M & (\text{Pure}) \\
M := M l &
\end{align*}
\]

\[
\begin{align*}
\{l_1, \ldots, l_n\} &\in \Delta & \neg l_1, \ldots, \neg l_n &\in M & M = M' \bullet l & \bullet \notin N & (\text{Backtrack}) \\
\end{align*}
\]
DPLL execution: exercise

$$\Delta_0 := \{\{1, -2\}, \{-1, -2\}, \{2, 3\}, \{-3, 2\}, \{1, 4\}\}$$

<table>
<thead>
<tr>
<th>M</th>
<th>Δ</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>$4$</td>
<td>${1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}$</td>
<td>Pure</td>
</tr>
<tr>
<td>$4 \bullet -3$</td>
<td>${1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}$</td>
<td>Decide</td>
</tr>
<tr>
<td>$4 \bullet -3 2$</td>
<td>${1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}$</td>
<td>Propagate</td>
</tr>
<tr>
<td>$4 \bullet -3 2 1$</td>
<td>${1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}$</td>
<td>Propagate</td>
</tr>
</tbody>
</table>

$$\begin{align*}
\{l_1, \ldots, l_n\} & \in \Delta \\
\neg l_1, \ldots, \neg l_n & \in M \\
l, \neg l & \notin M
\end{align*}$$

(Propagate)

$$M := M \bullet l$$

$$\begin{align*}
l & \in \text{Lits}(\Delta) \\
l, \neg l & \notin M
\end{align*}$$

(Decide)

$$M := M \bullet l$$

$$\begin{align*}
\{l_1, \ldots, l_n\} & \in \Delta \\
\neg l_1, \ldots, \neg l_n & \in M \\
\bullet & \notin N
\end{align*}$$

(Fail)

$$\text{Fail}$$

$$\begin{align*}
\{l_1, \ldots, l_n\} & \in \Delta \\
\neg l_1, \ldots, \neg l_n & \in M \\
\bullet & \notin N
\end{align*}$$

(Backtrack)

$$M := M' \bullet l$$

$$M := M' \neg l$$
DPLL execution: exercise

$$\Delta_0 := \{\{1, -2\}, \{-1, -2\}, \{2, 3\}, \{-3, 2\}, \{1, 4\}\}$$

<table>
<thead>
<tr>
<th>M</th>
<th>(\Delta)</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}\</td>
<td>Pure</td>
</tr>
<tr>
<td>4 &amp; -3</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}\</td>
<td>Decide</td>
</tr>
<tr>
<td>4 &amp; -3 &amp; 2</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}\</td>
<td>Propagate</td>
</tr>
<tr>
<td>4 &amp; -3 &amp; 2 &amp; 1</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}\</td>
<td>Propagate</td>
</tr>
<tr>
<td>4 &amp; 3</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}\</td>
<td>Backtrack</td>
</tr>
</tbody>
</table>

---

\(\{l_1, \ldots, l_n, l\} \in \Delta\)
\(\lnot l_1, \ldots, \lnot l_n \in M\)
\(l, \lnot l \notin M\)

(Propagate)

\(M := M \cup \{l\}\)

\(l\) literal of \(\Delta\)
\(\lnot l\) not literal of \(\Delta\)
\(l, \lnot l \notin M\)

(Pure)

\(M := M \cup \{l\}\)

\(\{l_1, \ldots, l_n\} \in \Delta\)
\(\lnot l_1, \ldots, \lnot l_n \in M\)
\(\bullet \notin N\)

(Decide)

\(M := M \cup \{l\}\)

\(l \in \text{Lits}(\Delta)\)
\(l, \lnot l \notin M\)

(Fail)

\(M := M' \cup \{l\}\)

\(\bullet \notin N\)

(Fail)

\(M := M' \cup \{l\}\)

\(\bullet \notin N\)

(Backtrack)

\(M := M' \cup \{l\}\)

\(\bullet \notin N\)
DPLL execution: exercise

\[ \Delta_0 := \{\{1, \neg 2\}, \{\neg 1, \neg 2\}, \{2, 3\}, \{\neg 3, 2\}, \{1, 4\}\} \]

<table>
<thead>
<tr>
<th>(M)</th>
<th>(\Delta)</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>{1, \neg 2}, {\neg 1, \neg 2}, {2, 3}, {\neg 3, 2}, {1, 4}\</td>
<td>Pure</td>
</tr>
<tr>
<td>4 \bullet \neg 3</td>
<td>{1, \neg 2}, {\neg 1, \neg 2}, {2, 3}, {\neg 3, 2}, {1, 4}\</td>
<td>Decide</td>
</tr>
<tr>
<td>4 \bullet \neg 3 \ 2</td>
<td>{1, \neg 2}, {\neg 1, \neg 2}, {2, 3}, {\neg 3, 2}, {1, 4}\</td>
<td>Propagate</td>
</tr>
<tr>
<td>4 \ 3</td>
<td>{1, \neg 2}, {\neg 1, \neg 2}, {2, 3}, {\neg 3, 2}, {1, 4}\</td>
<td>Backtrack</td>
</tr>
<tr>
<td>4 \ 3 \ 2</td>
<td>{1, \neg 2}, {\neg 1, \neg 2}, {2, 3}, {\neg 3, 2}, {1, 4}\</td>
<td>Propagate</td>
</tr>
<tr>
<td>4 \ 3 \ 2 \ 1</td>
<td>{1, \neg 2}, {\neg 1, \neg 2}, {2, 3}, {\neg 3, 2}, {1, 4}\</td>
<td>Propagate</td>
</tr>
</tbody>
</table>

\[
\{l_1, \ldots, l_n, l\} \in \Delta \\
\neg l_1, \ldots, \neg l_n \in M \\
M := M \bullet l \\
\text{(Propagate)}
\]

\[
\begin{align*}
&l \text{ literal of } \Delta \\
&\neg l \text{ not literal of } \Delta \\
&M := M \bullet l \\
&\text{(Pure)}
\end{align*}
\]

\[
\begin{align*}
&l \in \text{Lits}(\Delta) \\
&\neg l \notin M \\
&M := M \bullet l \\
&\text{(Decide)}
\end{align*}
\]

\[
\begin{align*}
&\{l_1, \ldots, l_n\} \in \Delta \\
&\neg l_1, \ldots, \neg l_n \in M \\
&\bullet \notin N \\
&\text{Fail}
\end{align*}
\]

\[
\begin{align*}
&\{l_1, \ldots, l_n\} \in \Delta \\
&\neg l_1, \ldots, \neg l_n \in M \\
&M = M' \bullet N \\
&\bullet \notin N \\
&M := M' \bullet l \\
&\text{(Backtrack)}
\end{align*}
\]
DPLL execution: exercise

\[ \Delta_0 := \{\{1, -2\}, \{-1, -2\}, \{2, 3\}, \{-3, 2\}, \{1, 4\}\} \]

<table>
<thead>
<tr>
<th>(M)</th>
<th>(\Delta)</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>\text{Pure}</td>
</tr>
<tr>
<td>4 • -3</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>\text{Decide}</td>
</tr>
<tr>
<td>4 • -3 2</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>\text{Propagate}</td>
</tr>
<tr>
<td>4 • -3 2 1</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>\text{Propagate}</td>
</tr>
<tr>
<td>4 3</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>\text{Backtrack}</td>
</tr>
<tr>
<td>4 3 2</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>\text{Propagate}</td>
</tr>
<tr>
<td>4 3 2 1</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>\text{Propagate}</td>
</tr>
<tr>
<td>4 3 2 1</td>
<td>{1, -2}, {-1, -2}, {2, 3}, {-3, 2}, {1, 4}</td>
<td>\text{Fail}</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\{l_1, \ldots, l_n, l\} \in \Delta & \quad \neg l_1, \ldots, \neg l_n \in M & l, \neg l \notin M \\
\text{(Propagate)} & & M := M \cdot l \\
\text{\(l\) literal of \(\Delta\)} & \neg l \text{ not literal of \(\Delta\)} & l, \neg l \notin M \\
\text{(Pure)} & & M := M \cdot l \\
\{l_1, \ldots, l_n\} \in \Delta & \neg l_1, \ldots, \neg l_n \in M & M = M' \cdot l \ N \bullet \notin N \\
\text{(Backtrack)} & & M := M' \cdot \neg l \\
\end{align*}
\]

\[
\begin{align*}
l \in \text{Lits}(\Delta) & \quad l, \neg l \notin M \\
\text{(Decide)} & & M := M \cdot l \\
\{l_1, \ldots, l_n\} \in \Delta & \neg l_1, \ldots, \neg l_n \in M \bullet \notin M \\
\text{(Fail)} & & \text{Fail} \\
\end{align*}
\]

October 11, 2023
Transforming DPLL to Resolution

The search procedure of DPLL can be in fact reduced to a resolution proof (a sequence of application of resolution rules).

For details, see Chapter 4.2 of “The Correctness of SAT Solvers and Related Issues” by Lintao Zhang.
DPLL Shortcomings

OK for randomly generated CNFs, but not for practical ones. Why?

- **No learning**: throws away all the work performed to conclude that the current partial assignment is bad. Revisits bad partial assignments that lead to the conflict due to the same root cause.

- **Chronological backtracking**: backtracks one level, even if it can be deduced that the current partial assignment became doomed at a lower level.

- **Naïve decisions**: picks an arbitrary variable to branch on. Fails to consider the state of the search to make heuristically better decisions.
Conflict-Driven Clause Learning (CDCL)

- **Learning**: $\Delta$ is augmented with a conflict clause that summarizes the root cause of the conflict.
Conflict-Driven Clause Learning (CDCL)

- **Learning**: $\Delta$ is augmented with a conflict clause that summarizes the root cause of the conflict.
- **Non-chronological backtracking**: backtracks $b$ levels, based on the cause of the conflict.
Conflict-Driven Clause Learning (CDCL)

- **Learning**: $\Delta$ is augmented with a conflict clause that summarizes the root cause of the conflict.
- **Non-chronological backtracking**: backtracks $b$ levels, based on the cause of the conflict.
- **Decision heuristics**: choose the next literal to add to the current partial assignment based on the state of the search.
From DPLL to CDCL Solvers

To model conflict-driven backjumping and learning, add to states a third component $C$ whose value is either no or a clause (often referred to as the conflict clause).
From DPLL to CDCL Solvers

To model conflict-driven backjumping and learning, add to states a third component $C$ whose value is either no or a clause (often referred to as the conflict clause).

States: Fail or $\langle M, \Delta, C \rangle$

Initial state:

- $\langle(), \Delta_0, \text{no}\rangle$, where $\Delta_0$ is to be checked for satisfiability

Expected final states:

- Fail if $\Delta_0$ is unsatisfiable
- $\langle M, G, \text{no}\rangle$ otherwise, where
  - $G$ is equivalent to $\Delta_0$ and
  - $M$ satisfies $G$
From DPLL to CDCL Solvers

Replace **Backtrack** with three rules:

1. **C** = ∅;
2. **C** = {l₁, ..., lₙ} ∪ D;
3. **C** = C ∪ {¬l₁, ..., ¬lₙ}.

The conflict clause is ∅, and there is a conflicting clause w.r.t. the current partial assignment M. So we set C to the conflicting clause.

**C** = {l₁} ∪ D

∆ contains a clause {l₁, ..., lₙ, ¬l₁, ..., ¬lₙ} such that 1) l is in the conflict clause; 2) ¬l is assigned true; 3) l₁, ..., lₙ are all assigned false and are assigned before l. We can derive a new conflict clause C by applying resolution.

**C** = {l₁, ..., lₙ, l}

lev(¬l₁), ..., lev(¬lₙ) ≤ i < lev(¬l),

C : = no

M : = M [i]

Compute the backtracking level: find the literals ¬l₁, ¬lₙ ∈ C that was assigned last and next to last. Backtrack to a level that is < lev(l) and ≥ lev(lₙ).

Maintain invariant: ∆ ⊧ C and M ⊧ ¬C when C ≠ no.
Replace **Backtrack** with three rules:

\[
\begin{align*}
\text{Conflict:} & \quad C = \text{no} \quad \{l_1, \ldots, l_n\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \\
C & := \{l_1, \ldots, l_n\}
\end{align*}
\]

The conflict clause is **no**, and there is a **conflicting clause** w.r.t. the current partial assignment \(M\). So we set \(C\) to the conflicting clause.
Replace **Backtrack** with three rules:

**Conflicting Clause**

\[
C = \text{no} \quad \{l_1, \ldots, l_n\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \\
C := \{l_1, \ldots, l_n\}
\]

The conflict clause is **no**, and there is a **conflicting clause** w.r.t. the current partial assignment \( M \). So we set \( C \) to the conflicting clause.

**Explain**

\[
C = \{l\} \cup D \quad \{l_1, \ldots, l_n, \neg l\} \in \Delta \quad \neg l_1, \ldots, \neg l_n, \neg l \in M \\
C := \{l_1, \ldots, l_n\} \cup D
\]

\( \Delta \) contains a clause \( \{l_1, \ldots, l_n, \neg l\} \) such that 1) \( l \) is in the conflict clause; 2) \( \neg l \) is assigned **true**; 3) \( l_1, \ldots, l_n \) are all assigned **false** and are assigned before \( l \). We can **derive a new conflict clause** \( C \) by applying resolution.

**Note:** \( l <_M l' \) if \( l \) occurs before \( l' \) in \( M \)
Replace **Backtrack** with three rules:

\[
C = \text{no} \quad \{l_1, \ldots, l_n\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M\]  \quad (\text{Conflict})

The conflict clause is **no**, and there is a conflicting clause w.r.t. the current partial assignment \( M \). So we set \( C \) to the conflicting clause.

\[
C = \{l\} \cup D \quad \{l, \ldots, l_n, \neg l\} \in \Delta \quad \neg l_1, \ldots, \neg l_n, \neg l \in M \quad \neg l_1, \ldots, \neg l_n <_M \neg l\]  \quad (\text{Explain})

\( \Delta \) contains a clause \( \{l_1, \ldots, l_n, \neg l\} \) such that 1) \( l \) is in the conflict clause; 2) \( \neg l \) is assigned true; 3) \( l_1, \ldots, l_n \) are all assigned false and are assigned before \( l \). We can derive a new conflict clause \( C \) by applying resolution.

\[
C = \{l_1, \ldots, l_n, l\} \quad \text{lev}(\neg l_1), \ldots, \text{lev}(\neg l_n) \leq i < \text{lev}(\neg l)\]  \quad (\text{Backjump})

Compute the backtracking level: find the literals \( \neg l, \neg l_n \in C \) that was assigned last and next to last. Backtrack to a level that is \(< \text{lev}(l) \) and \( \geq \text{lev}(l_n) \).

Note: \( \text{lev}(l) = i \) iff \( l \) occurs in decision level \( i \) of \( M \)
From DPLL to CDCL Solvers

Replace **Backtrack** with three rules:

1. **Conflict**
   \[
   C = \text{no} \quad \{l_1, \ldots, l_n\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M
   \]
   \[
   C := \{l_1, \ldots, l_n\}
   \]
   The conflict clause is **no**, and there is a **conflicting clause** w.r.t. the current partial assignment **M**. So we set **C** to the conflicting clause.

2. **Explain**
   \[
   C = \{l\} \cup D \quad \{l_1, \ldots, l_n, \neg l\} \in \Delta \quad \neg l_1, \ldots, \neg l_n, \neg l \in M \quad \neg l_1, \ldots, \neg l_n, \neg l <_M \neg l
   \]
   \[
   C := \{l_1, \ldots, l_n\} \cup D
   \]
   \(
   \Delta \) contains a clause \( \{l_1, \ldots, l_n, \neg l\} \) such that 1) \( l \) is in the conflict clause; 2) \( \neg l \) is assigned **true**; 3) \( l_1, \ldots, l_n \) are all assigned **false** and are assigned before \( l \). We can **derive a new conflict clause** **C** by applying resolution.

3. **Backjump**
   \[
   C = \{l_1, \ldots, l_n, l\} \quad \neg l_1, \ldots, \neg l_n \leq i \leq \neg l
   \]
   \[
   C := \text{no} \quad M := [i]
   \]
   **Compute the backtracking level:** find the literals \( \neg l, \neg l_n \in C \) that was assigned last and next to last. **Backtrack** to a level that is \( < \neg l \) and \( \geq \neg l_n \).

   **Maintain invariant:** \( \Delta \models C \) and \( M \models \neg C \) when \( C \neq \text{no} \)
Modify **Fail** to

\[ C \neq \text{no} \]

\[ C \in M \]

(Fail)

Fail

C contains a conflict clause and there are no decision points to backjump to. So the formula is unsatisfiable.
Modify \textbf{Fail} to

\[
\frac{C \neq \text{no} \quad \bullet \notin M}{\text{Fail}} \quad \text{(Fail)}
\]

$C$ contains a \textit{conflict clause} and there are no decision points to backjump to. So the formula is \textit{unsatisfiable}.
CDCL Execution Example

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\} \} \]

<table>
<thead>
<tr>
<th></th>
<th>M</th>
<th>$\Delta$</th>
<th>$\Delta$</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>
CDCL Execution Example

\[ \Delta := \{C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\}\} \]

<table>
<thead>
<tr>
<th>(M)</th>
<th>(\Delta)</th>
<th>(C)</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
</tbody>
</table>

\[\left\{l_1, \cdots, l_n, l\right\} \in \Delta \quad \neg l_1, \cdots, \neg l_n \in M \quad l, \neg l \notin M \quad \rightarrow \quad M := M / l \quad \text{(Propagate)}\]
CDCL Execution Example

$$\Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\}\}$$

<table>
<thead>
<tr>
<th></th>
<th>$M$</th>
<th>$\Delta$</th>
<th>$C$</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\Delta$</td>
<td>no</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>$\Delta$</td>
<td>no</td>
<td>no</td>
<td>Propagate</td>
</tr>
</tbody>
</table>

$$\begin{align*}
\{l_1, \ldots, l_n, l\} \in \Delta &\quad \neg l_1, \ldots, \neg l_n \in M & l, \neg l \notin M &\quad (\text{Propagate})
\end{align*}$$

\[M := M \upharpoonright l\]
CDCL Execution Example

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\} \} \]

<table>
<thead>
<tr>
<th>( M )</th>
<th>( \Delta )</th>
<th>( C )</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Delta )</td>
<td>no</td>
<td>( \text{Propagate} )</td>
<td></td>
</tr>
<tr>
<td>1 ( \Delta )</td>
<td>no</td>
<td>( \text{Propagate} )</td>
<td></td>
</tr>
<tr>
<td>1 2 ( \Delta )</td>
<td>no</td>
<td>( \text{Decide} )</td>
<td></td>
</tr>
<tr>
<td>1 2 ( \bullet ) 3 ( \Delta )</td>
<td>no</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( l \in \text{Lits}(\Delta) \quad l, \neg l \notin M \)

\[ M := M \bullet l \quad \text{(Decide)} \]
CDCL Execution Example

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\} \} \]

<table>
<thead>
<tr>
<th></th>
<th>(\Delta)</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>(M)</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 (\cdot) 3</td>
<td>(\Delta)</td>
<td>no</td>
<td>Decide</td>
</tr>
<tr>
<td>1 2 (\cdot) 3 4</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
</tbody>
</table>

\[ \{l_1, \ldots, l_n, l\} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad l, \neg l \notin M \]

\[ M := M I \quad \text{(Propagate)} \]
CDCL Execution Example

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\} \} \]

<table>
<thead>
<tr>
<th>( M )</th>
<th>( \Delta )</th>
<th>( C )</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 ( \bullet ) 3</td>
<td>( \Delta )</td>
<td>no</td>
<td>Decide</td>
</tr>
<tr>
<td>1 2 ( \bullet ) 3 4</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 ( \bullet ) 3 4 5</td>
<td>( \Delta )</td>
<td>no</td>
<td>Decide</td>
</tr>
</tbody>
</table>

\[ l \in \text{Lits}(\Delta) \quad l, \neg l \notin M \]

\[ M := M \bullet l \] (Decide)
CDCL Execution Example

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\} \} \]

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( M )</td>
<td>( \Delta )</td>
<td>( C )</td>
<td>rule</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3</td>
<td>( \Delta )</td>
<td>no</td>
<td>Decide</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td>( \Delta )</td>
<td>no</td>
<td>Decide</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4 5 6</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
\{ l_1, \ldots, l_n, \} \in \Delta \quad \rightarrow \quad l_1, \ldots, \neg l_n \in M \quad \neg l, \neg l \notin M \quad M := M l \\
\text{(Propagate)}
\]
## CDCL Execution Example

\[ \Delta := \{ C_1 : \{ 1 \}, C_2 : \{ -1, 2 \}, C_3 : \{ -3, 4 \}, C_4 : \{ -5, -6 \}, C_5 : \{ -1, -5, 7 \}, C_6 : \{ -2, -5, 6, -7 \} \} \]

<table>
<thead>
<tr>
<th>(M)</th>
<th>(\Delta)</th>
<th>(C)</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 • 3</td>
<td>(\Delta)</td>
<td>no</td>
<td>Decide</td>
</tr>
<tr>
<td>1 2 • 3 4</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 • 3 4 • 5</td>
<td>(\Delta)</td>
<td>no</td>
<td>Decide</td>
</tr>
<tr>
<td>1 2 • 3 4 • 5 • 6</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 • 3 4 • 5 • 6 • 7</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
</tbody>
</table>

\[
\{ l_1, \ldots, l_n, I \} \in \Delta \quad \neg l_1, \ldots, \neg l_n \in M \quad I, \neg I \notin M
\]

\[
M := M \cup \{ l \}
\]

(Propagate)
CDCL Execution Example

$$\Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\}\}$$

<table>
<thead>
<tr>
<th>$M$</th>
<th>$\Delta$</th>
<th>$C$</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta$</td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$\Delta$</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>$\Delta$</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 3</td>
<td>$\Delta$</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 3 4</td>
<td>$\Delta$</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td>$\Delta$</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 3 4 5 6</td>
<td>$\Delta$</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 3 4 5 6 7</td>
<td>$\Delta$</td>
<td>${-2, -5, 6, -7}$</td>
<td>Conflict</td>
</tr>
</tbody>
</table>

$$C = \text{no} \quad \{l_1, \ldots, l_n\} \in \Delta \quad -l_1, \ldots, -l_n \in M$$

$$C := \{l_1, \ldots, l_n\}$$

(Conflict)
CDCL Execution Example

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1,2\}, C_3 : \{-3,4\}, C_4 : \{-5,6\}, C_5 : \{-1,-5,7\}, C_6 : \{-2,-5,6,-7\}\} \]

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(M)</td>
<td>(\Delta)</td>
<td>(C)</td>
<td>rule</td>
</tr>
<tr>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 (\bullet) 3</td>
<td>(\Delta)</td>
<td>no</td>
<td>Decide</td>
</tr>
<tr>
<td>1 2 (\bullet) 3 4</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 (\bullet) 3 4 5</td>
<td>(\Delta)</td>
<td>no</td>
<td>Decide</td>
</tr>
<tr>
<td>1 2 (\bullet) 3 4 5 (-6)</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 (\bullet) 3 4 5 (-6) 7</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 (\bullet) 3 4 5 (-6) 7</td>
<td>({\neg-2, \neg5,6,\neg7})</td>
<td>Conflict</td>
<td></td>
</tr>
<tr>
<td>({\neg-1,\neg2,\neg5,6})</td>
<td>Explain w. (C_5)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
C = \{l\} \cup D \quad \{l_1, \ldots, l_n, \neg l\} \in \Delta \quad \neg l_1, \ldots, \neg l_n, \neg l \in M \quad \neg l_1, \ldots, \neg l_n <_M \neg l
\]

\[
C := \{l_1, \ldots, l_n\} \cup D
\]

\[
C = \{\neg7\} \cup \{\neg2, \neg5,6\} \quad \{\neg1, \neg5,7\} \in \Delta \quad 1,5 <_M 7
\]

\[
\Rightarrow C = \{\neg1, \neg5\} \cup \{\neg2, \neg5,6\} = \{\neg1, \neg2, \neg5,6\}
\]
CDCL Execution Example

$$\Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\} \}$$

<table>
<thead>
<tr>
<th>$M$</th>
<th>$\Delta$</th>
<th>$C$</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta$</td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$1$</td>
<td>$\Delta$</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>$1 2$</td>
<td>$\Delta$</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>$1 2 \bullet 3$</td>
<td>$\Delta$</td>
<td>no</td>
<td>Decide</td>
</tr>
<tr>
<td>$1 2 \bullet 3 4$</td>
<td>$\Delta$</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>$1 2 \bullet 3 4 \bullet 5$</td>
<td>$\Delta$</td>
<td>no</td>
<td>Decide</td>
</tr>
<tr>
<td>$1 2 \bullet 3 4 \bullet 5 \bullet 6 7$</td>
<td>$\Delta$</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>$1 2 \bullet 3 4 \bullet 5 \bullet 6 7$</td>
<td>${ -2, -5, 6, -7 }$</td>
<td>Conflict</td>
<td></td>
</tr>
<tr>
<td>$1 2 \bullet 3 4 \bullet 5 \bullet 6 7$</td>
<td>${ -1, -2, -5, 6 }$</td>
<td>Explain w. $C_5$</td>
<td></td>
</tr>
<tr>
<td>$1 2 \bullet 3 4 \bullet 5 \bullet 6 7$</td>
<td>${ -1, -2, -5 }$</td>
<td>Explain w. $C_4$</td>
<td></td>
</tr>
</tbody>
</table>

\[
C = \{ l \} \cup D \quad \{ l_1, \ldots, l_n, \neg l \} \in \Delta \quad \neg l_1, \ldots, \neg l_n, \neg l \in M \quad \neg l_1, \ldots, \neg l_n <_M \neg l
\]

(Explain)

\[
C := \{ l_1, \ldots, l_n \} \cup D
\]

\[
C = \{ 6 \} \cup \{ \neg 1, \neg 2, \neg 5 \} \quad \{ \neg 5, \neg 6 \} \in \Delta \quad 5 <_M \neg 6
\]

\[
\Rightarrow C = \{ \neg 1, \neg 2, \neg 5 \} \cup \{ \neg 5 \} = \{ \neg 1, \neg 2, \neg 5 \}
\]
CDCL Execution Example

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\}\} \]

\[
\begin{array}{|c|c|c|}
\hline
M & \Delta & C & \text{rule} \\
\hline
\Delta & \text{no} & & \\
1 & \Delta & \text{no} & \text{Propagate} \\
1 2 & \Delta & \text{no} & \text{Propagate} \\
1 2 3 & \Delta & \text{no} & \text{Decide} \\
1 2 3 4 & \Delta & \text{no} & \text{Propagate} \\
1 2 3 4 5 & \Delta & \text{no} & \text{Decide} \\
1 2 3 4 5 6 & \Delta & \text{no} & \text{Propagate} \\
1 2 3 4 5 6 7 & \Delta & \{-2, -5, 6, -7\} & \text{Conflict} \\
1 2 3 4 5 6 7 & \Delta & \{-1, -2, -5, 6\} & \text{Explain w. } C_5 \\
1 2 3 4 5 6 7 & \Delta & \{-1, -2, -5\} & \text{Explain w. } C_4 \\
1 2 3 4 5 6 7 & \Delta & \text{no} & \text{Backjump} \\
\hline
\end{array}
\]

\[ C = \{l_1, \ldots, l_n, l\} \quad \text{lev}(\neg l_1), \ldots, \text{lev}(\neg l_n) < i < \text{lev}(\neg l) \quad \text{(Backjump)} \]

\[ C := \text{no} \quad M := M[i] \]

\[ \text{lev}(1) = 0 \quad \text{lev}(2) = 0 \quad \text{lev}(5) = 2 \]

\[ \Rightarrow \text{backtrack to } M[0]_{\neg 5} \]

Note: could backtrack to \(M[1]_{\neg 5}\) as well.
### CDCL Execution Example

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\}\} \]

<table>
<thead>
<tr>
<th>(M)</th>
<th>(\Delta)</th>
<th>(C)</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\Delta)</td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 (\bullet) 3</td>
<td>(\Delta)</td>
<td>no</td>
<td>Decide</td>
</tr>
<tr>
<td>1 2 (\bullet) 3 4</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 (\bullet) 3 4 (\bullet) 5</td>
<td>(\Delta)</td>
<td>no</td>
<td>Decide</td>
</tr>
<tr>
<td>1 2 (\bullet) 3 4 (\bullet) 5 (\bullet) 6 (\bullet) 7</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 (\bullet) 3 4 (\bullet) 5 (\bullet) 6 (\bullet) 7 (\bullet) 8</td>
<td>(\Delta)</td>
<td>{-2, -5, 6, -7}</td>
<td>Conflict</td>
</tr>
<tr>
<td>1 2 (\bullet) 3 4 (\bullet) 5 (\bullet) 6 (\bullet) 7 (\bullet) 8 (\bullet) 9</td>
<td>(\Delta)</td>
<td>{-1, -2, -5, 6}</td>
<td>Explain w. (C_5)</td>
</tr>
<tr>
<td>1 2 (\bullet) 3 4 (\bullet) 5 (\bullet) 6 (\bullet) 7 (\bullet) 8 (\bullet) 9 (\bullet) 10</td>
<td>(\Delta)</td>
<td>{-1, -2, -5}</td>
<td>Explain w. (C_4)</td>
</tr>
<tr>
<td>1 2 (\bullet) 3 4 (\bullet) 5 (\bullet) 6 (\bullet) 7 (\bullet) 8 (\bullet) 9 (\bullet) 10 (\bullet) 11</td>
<td>(\Delta)</td>
<td>no</td>
<td>Backjump</td>
</tr>
<tr>
<td>1 2 (\bullet) 3 4 (\bullet) 5 (\bullet) 6 (\bullet) 7 (\bullet) 8 (\bullet) 9 (\bullet) 10 (\bullet) 11 (\bullet) 12</td>
<td>(\Delta)</td>
<td>no</td>
<td>Decide</td>
</tr>
</tbody>
</table>
CDCL Execution Example

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\} \} \]

<table>
<thead>
<tr>
<th>(M)</th>
<th>(\Delta)</th>
<th>(C)</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 3</td>
<td>(\Delta)</td>
<td>no</td>
<td>Decide</td>
</tr>
<tr>
<td>1 2 3 4</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td>(\Delta)</td>
<td>no</td>
<td>Decide</td>
</tr>
<tr>
<td>1 2 3 4 5 6</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 3 4 5 6 7</td>
<td>({\text{Conflict}})</td>
<td>no</td>
<td>Explain w. (C_5)</td>
</tr>
<tr>
<td>1 2 3 4 5 6 7</td>
<td>({\text{Conflict}})</td>
<td>no</td>
<td>Explain w. (C_4)</td>
</tr>
<tr>
<td>1 2 3 4 5 6 7</td>
<td>({-1, -2, -5})</td>
<td>no</td>
<td>Backjump</td>
</tr>
<tr>
<td>1 2 5</td>
<td>(\Delta)</td>
<td>no</td>
<td>Decide</td>
</tr>
<tr>
<td>1 2 5 3</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 5 3 4</td>
<td>(\Delta)</td>
<td>no</td>
<td>SAT!</td>
</tr>
</tbody>
</table>
From DPLL to CDCL Solvers

Also add

\[
\Delta \models C \quad C \notin \Delta \\
\Delta := \Delta \cup \{C\} \quad \text{(Learn)}
\]

Learn can be applied to any clause stored in \( C \) when \( C \neq \text{no} \).

Memory can become quickly filled with millions of (conflict) clauses, so it would be nice to be able to delete clauses.

If the solver got stuck in a hopeless branch, it would be nice to be able to restart altogether.

The progress is not completely lost due to Learn.
From DPLL to CDCL Solvers

Also add

<table>
<thead>
<tr>
<th>[ \Delta \models C \quad C \notin \Delta ]</th>
<th>[ \Delta := \Delta \cup {C} ]</th>
<th>(Learn)</th>
</tr>
</thead>
</table>

Learn can be applied to any clause stored in \( C \) when \( C \neq \text{no} \).

<table>
<thead>
<tr>
<th>[ \Delta = \Delta' \cup {C} \quad \Delta' \models C ]</th>
<th>[ \Delta := \Delta' ]</th>
<th>(Forget)</th>
</tr>
</thead>
</table>

Memory can become quickly filled with millions of (conflict) clauses, so it would be nice to be able to delete clauses.
Also add

\[
\frac{\Delta \models C \quad C \notin \Delta}{\Delta := \Delta \cup \{C\}} \quad \text{(Learn)}
\]

Learn can be applied to any clause stored in $C$ when $C \neq \text{no}$.

\[
\frac{C = \text{no}}{\Delta := \Delta' \cup \{C\}} \quad \frac{\Delta' \models C}{\Delta := \Delta'} \quad \text{(Forget)}
\]

Memory can become quickly filled with millions of (conflict) clauses, so it would be nice to be able to delete clauses.

\[
\frac{M := M^0 \quad C := \text{no}}{} \quad \text{(Restart)}
\]

If the solver got stuck in a hopeless branch, it would be nice to be able to restart altogether. The progress is not completely lost due to Learn.
Modeling Modern SAT Solvers

At the core, current CDCL SAT solvers are implementations of the transition system with rules

Propagate, Decide,
Conflict, Explain, Backjump,
Learn, Forget, Restart
Modeling Modern SAT Solvers

At the core, current CDCL SAT solvers are implementations of the transition system with rules

Propagate, Decide,
Conflict, Explain, Backjump,
Learn, Forget, Restart

Basic CDCL \( \text{def} \)

\{ Propagate, Decide, Conflict, Explain, Backjump \}
Modeling Modern SAT Solvers

At the core, current CDCL SAT solvers are implementations of the transition system with rules

\(\text{Propagate, Decide, Conflict, Explain, Backjump, Learn, Forget, Restart}\)

Basic CDCL \(\overset{\text{def}}{=}\)

\(\{ \text{Propagate, Decide, Conflict, Explain, Backjump} \}\)

CDCL \(\overset{\text{def}}{=}\) Basic CDCL + \(\{ \text{Learn, Forget, Restart} \}\)
The Basic CDCL System – Correctness

Note the following terminology:

Irreducible state: state for which no Basic CDCL rules apply

Execution: sequence of transitions allowed by the rules and starting with $M = \emptyset$ and $C = \text{no}$

Exhausted execution: execution ending in an irreducible state

Proposition (Refutation Soundness) For every exhausted execution starting with $\Delta = \Delta_0$ and ending with Fail, the clause set $\Delta_0$ is unsatisfiable.

Proposition (Solution Soundness) For every exhausted execution starting with $\Delta = \Delta_0$ and ending with $C = \text{no}$, the clause set $\Delta_0$ is satisfied by $M$. 
Note the following terminology:

**Irreducible state**: state for which no Basic CDCL rules apply

**Execution**: sequence of transitions allowed by the rules and starting with $M = \emptyset$ and $C = \text{no}$

**Exhausted execution**: execution ending in an irreducible state

**Proposition (Strong Termination)** Every execution in Basic CDCL is finite.

**Note**: This is not so immediate, because of Backjump.
The Basic CDCL System – Correctness

Note the following terminology:

**Irreducible state:** state for which no Basic CDCL rules apply

**Execution:** sequence of transitions allowed by the rules and starting with $M = \emptyset$ and $C = \text{no}$

**Exhausted execution:** execution ending in an irreducible state

**Proposition (Strong Termination)** Every execution in Basic CDCL is finite.

**Lemma** Every exhausted execution ends with either $C = \text{no}$ or Fail.
Note the following terminology:

**Irreducible state**: state for which no Basic CDCL rules apply

**Execution**: sequence of transitions allowed by the rules and starting with $M = \emptyset$ and $C = \text{no}$

**Exhausted execution**: execution ending in an irreducible state

**Proposition** (Refutation Soundness) For every exhausted execution starting with $\Delta = \Delta_0$ and ending with $\text{Fail}$, the clause set $\Delta_0$ is unsatisfiable.

**Proposition** (Solution Soundness) For every exhausted execution starting with $\Delta = \Delta_0$ and ending with $C = \text{no}$, the clause set $\Delta_0$ is satisfied by $M$. 
The CDCL System – Strategies

To ensure termination, apply 1) at least one Basic CDCL rule between each two Learn applications; 2) Restart less and less often.

A common basic strategy applies the rules with the following priorities:

1. If $n > 0$ conflicts have been found so far, increase $n$ and apply Restart
2. If a clause is falsified by $M$, apply Conflict
3. Apply Explain repeatedly
4. Apply Learn
5. Apply Backjump
6. Apply Propagate to completion
7. Apply Decide

Step 3-5 is called conflict analysis and there are some heuristic choices in this process.

• When to stop applying Explain to a conflict?
• Which level to Backjump to?
The CDCL System – Strategies

To ensure termination, apply 1) at least one Basic CDCL rule between each two Learn applications; 2) Restart less and less often. A common basic strategy applies the rules with the following priorities:

1. If $n > 0$ conflicts have been found so far, increase $n$ and apply Restart
2. If a clause is falsified by M, apply Conflict
3. Apply Explain repeatedly
4. Apply Learn
5. Apply Backjump
6. Apply Propagate to completion
7. Apply Decide
The CDCL System – Strategies

To ensure termination, apply 1) at least one Basic CDCL rule between each two Learn applications; 2) Restart less and less often. A common basic strategy applies the rules with the following priorities:

1. If \( n > 0 \) conflicts have been found so far, increase \( n \) and apply Restart
2. If a clause is falsified by \( M \), apply Conflict
3. Apply Explain repeatedly
4. Apply Learn
5. Apply Backjump
6. Apply Propagate to completion
7. Apply Decide

Step 3-5 is called conflict analysis and there are some heuristic choices in this process.

- When to stop applying Explain to a conflict?
- Which level to Backjump to?
Conflict Analysis: Implication Graph

The goal of clause learning is to **blocks** partial assignments that lead to the current conflict.

A common strategy is to learn an **asserting clause**, a conflict clause that is **unit** after backtracking.

One way to illustrate different conflict analysis strategy is through **implication graphs**.
Conflict Analysis: Implication Graph

The goal of clause learning is to blocks partial assignments that lead to the current conflict.

A common strategy is to learn an asserting clause, a conflict clause that is unit after backtracking.

One way to illustrate different conflict analysis strategy is through implication graphs.

An implication graph is a labeled directed acyclic graph $G(V, E)$, where:

- $v \in V$ are literals of the current partial assignment. Each node is labeled with:
  - the literal that it represents
  - the decision level at which it entered the partial assignment

- $e \in E$ are directed labeled edges:
  - $E = \{(v_i, v_j) \mid v_i, v_j \in V, \neg v_i \in \text{Antecedent}(v_j)\}$
  - each edge $(v_i, v_j)$ is labeled with $\text{Antecedent}(v_j)$.

- $G$ can also contain a single conflict node labeled with $K$ and incoming edges $\{(v, K) \mid \neg v \in c\}$ labeled with $c$ for some conflicting clause $c$.

In this case, $G$ is called a conflict graph.
Conflict Analysis: Implication Graph

The goal of clause learning is to blocks partial assignments that lead to the current conflict.

A common strategy is to learn an asserting clause, a conflict clause that is unit after backtracking.

One way to illustrate different conflict analysis strategy is through implication graphs.

An implication graph is a labeled directed acyclic graph $G(V, E)$, where:

- $v \in V$ are literals of the current partial assignment. Each node is labeled with:
  - the literal that it represents
  - the decision level at which it entered the partial assignment

- $e \in E$ are directed labeled edges:
  - $E = \{(v_i, v_j) | v_i, v_j \in V, \neg v_i \in \text{Antecedent}(v_j)\}$
  - each edge $(v_i, v_j)$ is labeled with $\text{Antecedent}(v_j)$. 

$G$ can also contain a single conflict node labeled with $K$ and incoming edges $\{(v, K) | \neg v \in c\}$ labeled with $c$ for some conflicting clause $c$. In this case, $G$ is called a conflict graph.
Conflict Analysis: Implication Graph

The goal of clause learning is to blocks partial assignments that lead to the current conflict.

A common strategy is to learn an asserting clause, a conflict clause that is unit after backtracking.

One way to illustrate different conflict analysis strategy is through implication graphs.

An implication graph is a labeled directed acyclic graph $G(V, E)$, where:

- $v \in V$ are literals of the current partial assignment. Each node is labeled with:
  - the literal that it represents
  - the decision level at which it entered the partial assignment

- $e \in E$ are directed labeled edges:
  - $E = \{(v_i, v_j) | v_i, v_j \in V, \neg v_i \in \text{Antecedent}(v_j)\}$
  - each edge $(v_i, v_j)$ is labeled with $\text{Antecedent}(v_j)$.

- $G$ can also contain a single conflict node labeled with $\mathcal{K}$ and incoming edges $\{(v, \mathcal{K}) | \neg v \in c\}$ labeled with $c$ for some conflicting clause $c$.

In this case, $G$ is called a conflict graph.
Revisiting CDCL Execution Example with Implication Graph

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\}\} \]

<table>
<thead>
<tr>
<th>M</th>
<th>Δ</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Δ</td>
<td>no</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A Unique Implication Point (UIP) is any node other than \(K\) that is on all paths from the current decision node to \(K\). A first UIP is a UIP that is closest to the conflict node. In this case, \(5@2\) is the only UIP and thus also the first UIP.
Revisiting CDCL Execution Example with Implication Graph

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\} \} \]

<table>
<thead>
<tr>
<th>M</th>
<th>(\Delta)</th>
<th>C</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(\Delta)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
</tbody>
</table>

1@0

A **Unique Implication Point (UIP)** is any node other than \(K\) that is on all paths from the current decision node to \(K\). A **first UIP** is a UIP that is closest to the conflict node. In this case, 5@2 is the only UIP and thus also the first UIP.

October 11, 2023 CS257
Revisiting CDCL Execution Example with Implication Graph

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\} \} \]

<table>
<thead>
<tr>
<th>( M )</th>
<th>( \Delta )</th>
<th>( C )</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Delta )</td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 ( \Delta )</td>
<td>no</td>
<td></td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 ( \Delta )</td>
<td>no</td>
<td></td>
<td>Propagate</td>
</tr>
</tbody>
</table>

1@0 → \( C_2 \) → 2@0
Revisiting CDCL Execution Example with Implication Graph

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\}\} \]

<table>
<thead>
<tr>
<th>( M )</th>
<th>( \Delta )</th>
<th>( C )</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 ( \bullet ) 3</td>
<td>( \Delta )</td>
<td>no</td>
<td>Decide</td>
</tr>
</tbody>
</table>

A unique implication point (UIP) is any node other than \( K \) that is on all paths from the current decision node to \( K \). A first UIP is a UIP that is closest to the conflict node. In this case, 5@2 is the only UIP and thus also the first UIP.
Revisiting CDCL Execution Example with Implication Graph

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\} \} \]

<table>
<thead>
<tr>
<th>(M)</th>
<th>(\Delta)</th>
<th>(C)</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>(\Delta)</td>
<td>no</td>
<td>Decide</td>
</tr>
<tr>
<td>1 2 (\cdot) 3</td>
<td>(\Delta)</td>
<td>no</td>
<td>Propagate</td>
</tr>
</tbody>
</table>

October 11, 2023 CS257 24 / 26
Revisiting CDCL Execution Example with Implication Graph

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\} \} \]

<table>
<thead>
<tr>
<th>( M )</th>
<th>( \Delta )</th>
<th>( C )</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 ( \cdot ) 3</td>
<td>( \Delta )</td>
<td>no</td>
<td>Decide</td>
</tr>
<tr>
<td>1 2 ( \cdot ) 3 4</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 ( \cdot ) 3 4 ( \cdot ) 5</td>
<td>( \Delta )</td>
<td>no</td>
<td>Decide</td>
</tr>
</tbody>
</table>

October 11, 2023 CS257 24 / 26
Revisiting CDCL Execution Example with Implication Graph

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\}\} \]

<table>
<thead>
<tr>
<th>( M )</th>
<th>( \Delta )</th>
<th>( C )</th>
<th>( \text{rule} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 3</td>
<td>( \Delta )</td>
<td>no</td>
<td>Decide</td>
</tr>
<tr>
<td>1 2 3 4</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td>( \Delta )</td>
<td>no</td>
<td>Decide</td>
</tr>
<tr>
<td>1 2 3 4 5 6</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
</tbody>
</table>

A Unique Implication Point (UIP) is any node other than \( K \) that is on all paths from the current decision node to \( K \).

A first UIP is a UIP that is closest to the conflict node. In this case, \( 5@2 \) is the only UIP and thus also the first UIP.

October 11, 2023
Revisiting CDCL Execution Example with Implication Graph

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\}\} \]

<table>
<thead>
<tr>
<th>[M]</th>
<th>[\Delta]</th>
<th>[C]</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[\Delta]</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>12</td>
<td>[\Delta]</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>12 • 3</td>
<td>[\Delta]</td>
<td>no</td>
<td>Decide</td>
</tr>
<tr>
<td>12 • 3 4</td>
<td>[\Delta]</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>12 • 3 4 • 5</td>
<td>[\Delta]</td>
<td>no</td>
<td>Decide</td>
</tr>
<tr>
<td>12 • 3 4 • 5 • 6</td>
<td>[\Delta]</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>12 • 3 4 • 5 • 6 7</td>
<td>[\Delta]</td>
<td>no</td>
<td>Propagate</td>
</tr>
</tbody>
</table>

A Unique Implication Point (UIP) is any node other than \(K\) that is on all paths from the current decision node to \(K\). A first UIP is a UIP that is closest to the conflict node. In this case, 5@2 is the only UIP and thus also the first UIP.
Revisiting CDCL Execution Example with Implication Graph

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\}\} \]

<table>
<thead>
<tr>
<th>M</th>
<th>(\Delta)</th>
<th>(C)</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1\Delta</td>
<td>no</td>
<td>Propagate</td>
<td></td>
</tr>
<tr>
<td>1\Delta</td>
<td>no</td>
<td>Propagate</td>
<td></td>
</tr>
<tr>
<td>(1\Delta)</td>
<td>no</td>
<td>Decide</td>
<td></td>
</tr>
<tr>
<td>(1\Delta)</td>
<td>no</td>
<td>Decide</td>
<td></td>
</tr>
<tr>
<td>(1\Delta)</td>
<td>no</td>
<td>Propagate</td>
<td></td>
</tr>
<tr>
<td>(1\Delta)</td>
<td>no</td>
<td>Propagate</td>
<td></td>
</tr>
<tr>
<td>(1\Delta)</td>
<td>{-2, -5, 6, -7}</td>
<td>Conflict</td>
<td></td>
</tr>
</tbody>
</table>

A Unique Implication Point (UIP) is any node other than \(K\) that is on all paths from the current decision node to \(K\).

A first UIP is a UIP that is closest to the conflict node. In this case, \(5\) is the only UIP and thus also the first UIP.
Revisiting CDCL Execution Example with Implication Graph

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\} \} \]

<table>
<thead>
<tr>
<th>( M )</th>
<th>( \Delta )</th>
<th>( C )</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>12</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>12 • 3</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>12 • 3 4</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>12 • 3 4 • 5</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>12 • 3 4 • 5 • 6</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>12 • 3 4 • 5 • 6 7</td>
<td>( \Delta )</td>
<td>no</td>
<td>Conflict</td>
</tr>
</tbody>
</table>

Any separating cut that breaks all paths from root nodes to conflict node, with roots on the reason side and conflict node on the conflict side, defines a valid conflict clause.
Revisiting CDCL Execution Example with Implication Graph

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\} \} \]

<table>
<thead>
<tr>
<th>( M )</th>
<th>( \Delta )</th>
<th>( C )</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>12</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>123</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>12345</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>123456</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1234567</td>
<td>( { -2, -5, 6, -7} )</td>
<td>no</td>
<td>Conflict</td>
</tr>
<tr>
<td>1234567</td>
<td>( { -1, -2, -5, 6} )</td>
<td>no</td>
<td>Explain w. ( C_5 )</td>
</tr>
</tbody>
</table>

**Explain** can be viewed as picking a literal \( l \) in the conflict clause \( C \), and replace \( C \) with the \( l \)-resolvent of \( C \) and Antecedent(\( \neg l \)). In this case, we pick \( l := -7 \).
Revisiting CDCL Execution Example with Implication Graph

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\} \} \]

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 3</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 3 4</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 3 4 5 -6</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 3 4 5 -6 7</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2 3 4 5 -6 7</td>
<td>( \Delta )</td>
<td>( {-2, -5, 6, -7} )</td>
<td>Conflict</td>
</tr>
<tr>
<td>1 2 3 4 5 -6 7</td>
<td>( \Delta )</td>
<td>( {-1, -2, -5, 6} )</td>
<td>Explain w. ( C_5 )</td>
</tr>
<tr>
<td>1 2 3 4 5 -6 7</td>
<td>( \Delta )</td>
<td>( {-1, -2, -5} )</td>
<td>Explain w. ( C_4 )</td>
</tr>
</tbody>
</table>

**Explain** can be viewed as picking a literal \( l \) in the conflict clause \( C \), and replace \( C \) with the \( l \)-resolvant of \( C \) and **Antecedent**(\( \neg l \)).
In this case, we pick \( l := 6 \).
Revisiting CDCL Execution Example with Implication Graph

\[ \Delta := \{ C_1 : \{1\}, C_2 : \{-1, 2\}, C_3 : \{-3, 4\}, C_4 : \{-5, -6\}, C_5 : \{-1, -5, 7\}, C_6 : \{-2, -5, 6, -7\} \} \]

<table>
<thead>
<tr>
<th>( M )</th>
<th>( \Delta )</th>
<th>( C )</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>( \Delta )</td>
<td>no</td>
<td>Propagate</td>
</tr>
<tr>
<td>1 2</td>
<td>( \Delta )</td>
<td>no</td>
<td>Decide</td>
</tr>
<tr>
<td>1 2 3 4( \Delta )</td>
<td>no</td>
<td>Propagate</td>
<td></td>
</tr>
<tr>
<td>1 2 3 4 5( \Delta )</td>
<td>no</td>
<td>Decide</td>
<td></td>
</tr>
<tr>
<td>1 2 3 4 5 ( \Delta )</td>
<td>no</td>
<td>Propagate</td>
<td></td>
</tr>
<tr>
<td>1 2 3 4 5 ( \Delta )</td>
<td>no</td>
<td>Propagate</td>
<td></td>
</tr>
<tr>
<td>1 2 3 4 5 ( \Delta )</td>
<td>no</td>
<td>Conflict</td>
<td></td>
</tr>
<tr>
<td>1 2 3 4 5 ( \Delta )</td>
<td>no</td>
<td>Explain w. ( C_5 )</td>
<td></td>
</tr>
<tr>
<td>1 2 3 4 5 ( \Delta )</td>
<td>no</td>
<td>Explain w. ( C_4 )</td>
<td></td>
</tr>
<tr>
<td>1 2 3 4 5 ( \Delta )</td>
<td>no</td>
<td>Backjump</td>
<td></td>
</tr>
</tbody>
</table>

A **Unique Implication Point (UIP)** is any node other than \( \mathcal{K} \) that is on all paths from the current decision node to \( \mathcal{K} \).

A **first UIP** is a UIP that is closest to the conflict node.

In this case, 5@2 is the only UIP and thus also the first UIP.
Learning the First UIP

Empirical studies show that it is a good strategy to

- learn a conflict clause $C$ such that the first UIP is the only literal at the current decision level;
- backjump to the second lowest decision level among literals in $C$.

To compute such conflict clause, keep applying the Explain rule on the last assigned literal in $C$, until the first UIP is the only literal at the current decision level.
Learning the First UIP

Empirical studies show that it is a good strategy to

- learn a conflict clause $C$ such that the first UIP is the only literal at the current decision level;
- backjump to the second lowest decision level among literals in $C$.

To compute such conflict clause, keep applying the Explain rule on the last assigned literal in $C$, until the first UIP is the only literal at the current decision level.

The resulting conflict clause is an asserting clause.
Learning the First UIP

Empirical studies show that it is a good strategy to

- learn a conflict clause $C$ such that the first UIP is the only literal at the current decision level;
- backjump to the second lowest decision level among literals in $C$.

To compute such conflict clause, keep applying the Explain rule on the last assigned literal in $C$, until the first UIP is the only literal at the current decision level.

The resulting conflict clause is an asserting clause.

Possible explanations for the results of the empirical studies:

- The strategy has a low computational cost, compared with strategies that choose UIPs further away from the conflict.
- It backtracks to the lowest decision level.
Non-chronological Backtracking is not Necessarily Better

See “Chronological Backtracking” by Nadel and Ryvchin, SAT 2018.