SAT Solving – Algorithms

Steffen Hölldobler and Norbert Manthey
International Center for Computational Logic
Technische Universität Dresden
Germany

- DPLL
- CDCL
- A solving abstraction

"Logic is everywhere ..."
Warm Up

- Used programming languages
Warm Up

- Used programming languages
- Size of implemented projects
Warm Up

► Used programming languages

► Size of implemented projects

► Parallel computing (multi-core, GPGPU, cluster)
Warm Up

- Used programming languages
- Size of implemented projects
- Parallel computing (multi-core, GPGPU, cluster)
- Interest in computer architecture
Revision

- Used Data Types
- Semantics
Formulas and Interpretations

► Let $F$ be a formulas an $I$ be an interpretation
► $I$ can
  ▶ satisfy $F$, if $F|_I \equiv \top$
  ▶ falsify $F$, if $F|_I \equiv \bot$
Formulas and Interpretations

- Let $F$ be a formulas an $I$ be an interpretation
- $I$ can
  - satisfy $F$, if $F|_I \equiv \top$
  - falsify $F$, if $F|_I \equiv \bot$

- A formula can be
  - unsatisfiable, $F \equiv \bot$
  - satisfiable
  - tautologic, $F \equiv \top$
Formulas and Interpretations

- Let $F$ be a formulas and $I$ be an interpretation

- $I$ can
  - satisfy $F$, if $F|_I \equiv \top$
  - falsify $F$, if $F|_I \equiv \bot$

- A formula can be
  - unsatisfiable, $F \equiv \bot$
  - satisfiable
  - tautologic, $F \equiv \top$

- Property: $F \equiv \top$, then $\neg F \equiv \bot$. 
Clauses and Conjunctive Normal Forms

Definition

- A clause is a generalized disjunction \([L_1, \ldots, L_n]\), \(n \geq 0\), where every \(L_i, 1 \leq i \leq n\), is a literal
- A clause is a unit clause if it contains precisely one literal
- A clause is a binary clause if it contains precisely two literals
Clauses and Conjunctive Normal Forms

Definition

- A clause is a generalized disjunction \([L_1, \ldots, L_n]\), \(n \geq 0\), where every \(L_i, 1 \leq i \leq n\), is a literal
- A clause is a unit clause if it contains precisely one literal
- A clause is a binary clause if it contains precisely two literals

Definition

- A formula is in conjunctive normal form (clause form, CNF) iff it is of the form \(<C_1, \ldots, C_m>\), \(m \geq 0\), and every \(C_j, 1 \leq j \leq m\), is a clause

Implementation and working assumptions

- A clause is an array of literals
  - Maintained to be a set of literals (no duplicates)
  - Clauses are no tautologies (excluded during parsing)
- A formula is an array of (pointers/references to) clauses
  - Maintained to be a multi set
Propositional Resolution

- Remind: clauses are considered to be sets

- **Definition** Let $C_1$ be a clause containing $L$ and $C_2$ be a clause containing $\overline{L}$; the (propositional) resolvent of $C_1$ and $C_2$ with respect to $L$ is the clause

  $$(C_1 \setminus \{L\}) \cup (C_2 \setminus \{\overline{L}\})$$

  $C$ is said to be a resolvent of $C_1$ and $C_2$ iff there exists a literal $L$ such that $C$ is the resolvent of $C_1$ and $C_2$ wrt $L$

- **Examples when resolving on $a$**

  - $(a \lor \neg a) \otimes (\neg a \lor a) = (a \lor \neg a)$
  - $(a \lor \neg b) \otimes (\neg a \lor b) = (b \lor \neg b)$
  - $(a \lor b) \otimes (\neg a \lor b) = (b)$

- Resolvents can subsume antecedents
- Usually, resolvents have more literals than antecedents
Propositional Resolution

▶ Remind: clauses are considered to be sets

▶ Definition  Let $C_1$ be a clause containing $L$ and $C_2$ be a clause containing $\bar{L}$; The (propositional) resolvent of $C_1$ and $C_2$ with respect to $L$ is the clause

$$(C_1 \setminus \{L\}) \cup (C_2 \setminus \{\bar{L}\})$$

$C$ is said to be a resolvent of $C_1$ and $C_2$ iff there exists a literal $L$ such that $C$ is the resolvent of $C_1$ and $C_2$ wrt $L$

▶ Examples when resolving on $a$

▶ $(a \lor \neg a) \otimes (\neg a \lor a) = (a \lor \neg a)$

▶ $(a \lor \neg b) \otimes (\neg a \lor b) = (b \lor \neg b)$

▶ $(a \lor b) \otimes (\neg a \lor b) = (b)$

▶ Resolvents can subsume antecedents
Propositional Resolution

- **Remind:** clauses are considered to be sets

- **Definition** Let $C_1$ be a clause containing $L$ and $C_2$ be a clause containing $\overline{L}$; The (propositional) resolvent of $C_1$ and $C_2$ with respect to $L$ is the clause

\[
(C_1 \setminus \{L\}) \cup (C_2 \setminus \{\overline{L}\})
\]

$C$ is said to be a resolvent of $C_1$ and $C_2$ iff there exists a literal $L$ such that $C$ is the resolvent of $C_1$ and $C_2$ wrt $L$

- **Examples when resolving on $a$**

  - $(a \lor \neg a) \otimes (\neg a \lor a) = (a \lor \neg a)$
  - $(a \lor \neg b) \otimes (\neg a \lor b) = (b \lor \neg b)$
  - $(a \lor b) \otimes (\neg a \lor b) = (b)$

- **Resolvents can subsume antecedents**

- **Usually, resolvents have more literals than antecedents**
SAT Solving
SAT Solving - Example

- Given: Conjunction of clauses
- Task: Find satisfying interpretation for variables if possible!

\[ F = (a \lor c) \land (\bar{b} \lor \bar{e} \lor \bar{f}) \land (\bar{a} \lor \bar{d} \lor f) \land (\bar{a} \lor \bar{b} \lor \bar{d} \lor e) \land (\bar{a} \lor b) \]

- How to find a solution?
- Some questions:
SAT Solving - Example

- Given: Conjunction of clauses
- Task: Find satisfying interpretation for variables if possible!

\[ F = (a \lor c) \land (\bar{b} \lor \bar{e} \lor \bar{f}) \land (\bar{a} \lor \bar{d} \lor f) \land (\bar{a} \lor \bar{b} \lor \bar{d} \lor \bar{e}) \land (\bar{a} \lor b) \]

- How to find a solution?
- Some questions:

1. How many combinations (solution candidates) exist for 6 Boolean variables?
SAT Solving - Example

- Given: Conjunction of clauses
- Task: Find satisfying interpretation for variables if possible!

\[ F = (a \lor c) \land (\bar{b} \lor \bar{e} \lor \bar{f}) \land (\bar{a} \lor \bar{d} \lor f) \land (\bar{a} \lor \bar{b} \lor \bar{d} \lor e) \land (\bar{a} \lor b) \]

- How to find a solution?
- Some questions:

1. How many combinations (solution candidates) exist for 6 Boolean variables?
2. How many percent of the candidates are cut by a unit clause?
SAT Solving - Example

► Given: Conjunction of clauses

► Task: Find satisfying interpretation for variables if possible!

\[ F = (a \lor c) \land (\bar{b} \lor \bar{e} \lor \bar{f}) \land (\bar{a} \lor \bar{d} \lor f) \land (\bar{a} \lor \bar{b} \lor \bar{d} \lor e) \land (\bar{a} \lor b) \]

► How to find a solution?

► Some questions:

1. How many combinations (solution candidates) exist for 6 Boolean variables?
2. How many percent of the candidates are cut by a unit clause?
3. How many percent of the candidates are cut by a binary, ternary, ... clause?
Power of Modern SAT Solvers

- **RISS 4.27, SAT Competition 2014, application track**
- **Formulas with several million clauses and variables can be solved**
SAT Solving – With Search

- Assume a literal
- Propagate immediate consequences
- If a conflict, backtrack

\[ F \vee \overline{b} \overline{a} = (a \vee c) \land (\overline{b} \vee \overline{e} \vee \overline{f}) \land (\overline{a} \vee \overline{d} \vee f) \land (\overline{a} \vee \overline{b} \vee \overline{d} \vee e) \land (\overline{a} \vee b) \]

- Assume \( \overline{b} = \top \), then we have
  \[ J = ( \overline{b} ) \]

Are there variables with a forced assignment? \( \triangledown \overline{a} \) and \( \overline{c} \)
SAT Solving – With Search

- Assume a literal
- Propagate immediate consequences
- If a conflict, backtrack
- Known as DPLL (Davis Putnam Logemann Loveland)
SAT Solving – With Search

- Assume a literal
- Propagate immediate consequences
- If a conflict, backtrack
- Known as DPLL (Davis Putnam Logemann Loveland)
- What are immediate consequences?
SAT Solving – With Search

- Assume a literal
- Propagate immediate consequences
- If a conflict, backtrack
- Known as DPLL (Davis Putnam Logemann Loveland)

- What are immediate consequences?

\[ F = (a \lor c) \land (\bar{b} \lor \bar{e} \lor f) \land (\bar{a} \lor \bar{d} \lor f) \land (\bar{a} \lor \bar{b} \lor \bar{d} \lor e) \land (\bar{a} \lor b) \]

- Assume \( \bar{b} = \top \), then we have \( J = (\bar{b}) \)
- Are there variables with a forced assignment?
SAT Solving – With Search

- Assume a literal
- Propagate immediate consequences
- If a conflict, backtrack
- Known as DPLL (Davis Putnam Logemann Loveland)

What are immediate consequences?

$$F|_{\bar{b}} = (a \lor c) \land (\bar{a} \lor \bar{d} \lor f) \land (\bar{a})$$

- Assume $\bar{b} = \top$, then we have $J = (\bar{b})$
- Are there variables with a forced assignment?
SAT Solving – With Search

- Assume a literal
- Propagate immediate consequences
- If a conflict, backtrack
- Known as DPLL (Davis Putnam Logemann Loveland)

What are immediate consequences?

\[ F|\bar{b} = (a \lor c) \land (\bar{a} \lor \bar{d} \lor f) \land (\bar{a}) \]

- Assume \( \bar{b} = \top \), then we have \( J = (\bar{b}) \)
- Are there variables with a forced assignment?
  - \( \bar{a} \)
SAT Solving – With Search

- Assume a literal
- Propagate immediate consequences
- If a conflict, backtrack
- Known as DPLL (Davis Putnam Logemann Loveland)

What are immediate consequences?

\[ F|_{\bar{b}\bar{a}} = (a \lor c) \land (\bar{a} \lor \bar{d} \lor f) \land (\bar{a} ) \]

- Assume \( \bar{b} = \top \), then we have \( J = (\bar{b}) \)
- Are there variables with a forced assignment?
  - \( \bar{a} \)
SAT Solving – With Search

- Assume a literal
- Propagate immediate consequences
- If a conflict, backtrack
- Known as DPLL (Davis Putnam Logemann Loveland)
- What are immediate consequences?

\[ F|_{\bar{b} \bar{a}} = (c) \land \]

- Assume \( \bar{b} = \top \), then we have \( J = (\bar{b}) \)
- Are there variables with a forced assignment?
  - \( \bar{a} \) and \( \bar{c} \)
Davis Putnam Logemann Loveland (DPLL) in a Nutshell

\[(\overline{a} \lor b) \land (\overline{b} \lor \overline{d} \lor e) \land (c \lor d) \land (\overline{a} \lor \overline{b} \lor \overline{e} \lor f) \land (\overline{a} \lor \overline{e} \lor \overline{f}) \land (d \lor \overline{f}) \land (\overline{c} \lor e \lor f)\]

<table>
<thead>
<tr>
<th>Variable</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Davis Putnam Logemann Loveland (DPLL) in a Nutshell

\[ \overline{a} \lor b \land (\overline{b} \lor \overline{d} \lor e) \land (c \lor d) \land (\overline{a} \lor \overline{b} \lor \overline{e} \lor f) \land (\overline{a} \lor \overline{e} \lor f) \land (d \lor f) \land (\overline{c} \lor e \lor f) \]

Variable | a | b | c | d | e | f |
--- | --- | --- | --- | --- | --- | --- |
Reason | - | - | - | - | - | - | -

▶ add a search decision
Davis Putnam Logemann Loveland (DPLL) in a Nutshell

\[(\overline{a} \lor b) \land (b \lor d \lor e) \land (c \lor d) \land (\overline{a} \lor \overline{b} \lor \overline{e} \lor f) \land (\overline{a} \lor \overline{e} \lor \overline{f}) \land (d \lor f) \land (\overline{c} \lor e \lor f)\]

![DPLL Algorithm](image)

<table>
<thead>
<tr>
<th>Variable</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
<th>(e)</th>
<th>(f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason</td>
<td>-</td>
<td>(C_1)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

▶ propagate consequences
Davis Putnam Logemann Loveland (DPLL) in a Nutshell

\[(\overline{a} \lor b) \land (\overline{b} \lor d \lor e) \land (c \lor d) \land (\overline{a} \lor \overline{b} \lor e \lor f) \land (\overline{a} \lor \overline{e} \lor f) \land (d \lor \overline{f}) \land (\overline{c} \lor e \lor f)\]

<table>
<thead>
<tr>
<th>Variable</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason</td>
<td>-</td>
<td>C₁</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

▶ add a search decision
Davis Putnam Logemann Loveland (DPLL) in a Nutshell

\[(\overline{a} \lor b) \land (\overline{b} \lor \overline{d} \lor e) \land (c \lor d) \land (\overline{a} \lor \overline{b} \lor \overline{e} \lor f) \land (\overline{a} \lor \overline{e} \lor f) \land (d \lor f) \land (\overline{c} \lor e \lor f)\]

- **add a search decision**
Davis Putnam Logemann Loveland (DPLL) in a Nutshell

\[(\overline{a} \lor b) \land (\overline{b} \lor \overline{d} \lor e) \land (c \lor d) \land (\overline{a} \lor \overline{b} \lor \overline{e} \lor f) \land (\overline{a} \lor \overline{e} \lor \overline{f}) \land (d \lor \overline{f}) \land (c \lor e \lor f)\]

Variable

<table>
<thead>
<tr>
<th>Variable</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>C₁</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-</td>
<td></td>
<td>-</td>
<td></td>
<td>C₂</td>
<td></td>
</tr>
</tbody>
</table>

▶ propagate consequences
Davis Putnam Logemann Loveland (DPLL) in a Nutshell

\[(\overline{a} \lor b) \land (\overline{b} \lor d \lor e) \land (c \lor d) \land (\overline{a} \lor b \lor \overline{e} \lor f) \land (\overline{a} \lor \overline{e} \lor \overline{f}) \land (d \lor \overline{f}) \land (\overline{c} \lor e \lor f)\]

DPLL found conflict

\[C_5 = (\overline{a} \lor \overline{e} \lor \overline{f})\]

<table>
<thead>
<tr>
<th>Variable</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
<th>(e)</th>
<th>(f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason</td>
<td>-</td>
<td>(C_1)</td>
<td>-</td>
<td>-</td>
<td>(C_2)</td>
<td>(C_4)</td>
</tr>
</tbody>
</table>

▶ propagate consequences
Davis Putnam Logemann Loveland (DPLL) in a Nutshell

\[(\overline{a} \lor b) \land (\overline{b} \lor d \lor e) \land (c \lor d) \land (\overline{a} \lor \overline{b} \lor \overline{e} \lor f) \land (\overline{a} \lor \overline{e} \lor \overline{f}) \land (d \lor f) \land (\overline{c} \lor e \lor f)\]

- found conflict

\[C_5 = (\overline{a} \lor \overline{e} \lor \overline{f})\]

<table>
<thead>
<tr>
<th>Variable</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason</td>
<td>-</td>
<td>$C_1$</td>
<td>-</td>
<td>-</td>
<td>$C_2$</td>
<td>$C_4$</td>
</tr>
</tbody>
</table>

▶ backtrack from conflict and proceed with search
DPLL pseudo code

An iterative solving algorithm

\textbf{IDPLL} (CNF formula \( F \))

\begin{align*}
\textbf{Input:} & \text{ A formula } F \text{ in CNF} \\
\textbf{Output:} & \text{ The solution SAT or UNSAT of this formula}
\end{align*}

\begin{algorithmic}
\State \( J := () \) // start with empty interpretation
\While{true} // until we find a solution
\If{\( F|_J = \emptyset \)}
\Return SAT // satisfiability rule
\EndIf
\If{\( \subseteq \in F|_J \)} // there was a conflict
\If{\( J = J' \hat{x} J'' \) and \( \not\exists y \in J'' \)}
\State \( J := J' \bar{x} \) // backtrack and undo most recent decision
\State \text{continue}
\EndIf
\Else
\Return UNSAT // unsatisfiability rule
\EndIf
\EndWhile
\If{(\( x \)) \in F|_J} // unit rule
\State \( J := Jx \) // extend the interpretation
\State \text{continue}
\EndIf
\If{\( x \in \text{lits}(F|_J) \) and \( \bar{x} \not\in \text{lits}(F|_J) \)} // pure literal rule
\State \( J := Jx \)
\State \text{continue}
\EndIf
\State \( J := J\hat{x} \) for some \( x \in \text{lits}(F|_J) \) // decide rule
\end{algorithmic}
Conclusions of the DPLL Algorithm

- Chronological backtracking
- Heavily depends on the order of the decision variables
- How to perform unit propagation? How to find a unit in the formula efficiently?
Unit Propagation

- How to perform unit propagation?
- How to find a unit in the formula efficiently?

- Assumption: we use the presented pseudo code as algorithm.
Conflict Driven Clause Learning (CDCL) in a Nutshell
Conflict Driven Clause Learning (CDCL) in a Nutshell

\[ F = (\overline{a} \lor b) \land (\overline{b} \lor \overline{d} \lor e) \land (c \lor d) \land (\overline{a} \lor \overline{b} \lor \overline{e} \lor f) \land (\overline{a} \lor \overline{e} \lor f) \land (d \lor \overline{f}) \land (\overline{c} \lor e \lor f) \]

<table>
<thead>
<tr>
<th>Variable</th>
<th>( a )</th>
<th>( b )</th>
<th>( c )</th>
<th>( d )</th>
<th>( e )</th>
<th>( f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Conflict Driven Clause Learning (CDCL) in a Nutshell

\[ F = (\overline{a} \lor b) \land (\overline{b} \lor \overline{d} \lor e) \land (c \lor d) \land (\overline{a} \lor \overline{b} \lor \overline{e} \lor f) \land (\overline{a} \lor \overline{e} \lor f) \land (d \lor \overline{f}) \land (\overline{c} \lor e \lor f) \]

\[ 
\begin{array}{c|c|c|c|c|c|c|c}
\text{Variable} & a & b & c & d & e & f \\
\hline
\text{Reason} & - & - & - & - & - & - & -
\end{array}
\]

▶ add a search decision
Conflict Driven Clause Learning (CDCL) in a Nutshell

\[ F = (\overline{a} \lor b) \land (\overline{b} \lor d \lor e) \land (c \lor d) \land (\overline{a} \lor \overline{b} \lor e \lor f) \land (\overline{a} \lor \overline{e} \lor \overline{f}) \land (d \lor \overline{f}) \land (\overline{c} \lor e \lor f) \]

<table>
<thead>
<tr>
<th>Variable</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason</td>
<td></td>
<td>C₁</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

▶ propagate consequences
Conflict Driven Clause Learning (CDCL) in a Nutshell

\[ F = (\bar{a} \lor b) \land (\bar{b} \lor \bar{d} \lor e) \land (c \lor d) \land (\bar{a} \lor \bar{b} \lor \bar{e} \lor f) \land (\bar{a} \lor \bar{e} \lor \bar{f}) \land (d \lor \bar{f}) \land (\bar{c} \lor e \lor f) \]

<table>
<thead>
<tr>
<th>Variable</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason</td>
<td>-</td>
<td>(C_1)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

▶ add a search decision
Conflict Driven Clause Learning (CDCL) in a Nutshell

\[ F = (\overline{a} \lor b) \land (\overline{b} \lor \overline{d} \lor \overline{e}) \land (c \lor d) \land (\overline{a} \lor \overline{b} \lor \overline{e} \lor f) \land (\overline{a} \lor \overline{e} \lor \overline{f}) \land (d \lor \overline{f}) \land (\overline{c} \lor e \lor f) \]

<table>
<thead>
<tr>
<th>Variable</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason</td>
<td>-</td>
<td>( C_1 )</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- add a search decision
Conflict Driven Clause Learning (CDCL) in a Nutshell

\[ F = (\overline{a} \lor b) \land (\overline{b} \lor d \lor e) \land (c \lor d) \land (\overline{a} \lor \overline{b} \lor \overline{e} \lor f) \land (\overline{a} \lor \overline{e} \lor f) \land (d \lor \overline{f}) \land (\overline{c} \lor e \lor f) \]

Variable | a | b | c | d | e | f
---|---|---|---|---|---|---
Reason  | - | \(C_1\) | - | - | \(C_2\) | -

▶ propagate consequences
Conflict Driven Clause Learning (CDCL) in a Nutshell

\[ F = (\overline{a} \lor b) \land (\overline{b} \lor \overline{d} \lor e) \land (c \lor d) \land (\overline{a} \lor \overline{b} \lor e \lor f) \land (\overline{a} \lor \overline{e} \lor f) \land (d \lor \overline{f}) \land (\overline{c} \lor e \lor f) \]

Variable | \(a\) | \(b\) | \(c\) | \(d\) | \(e\) | \(f\)
---|---|---|---|---|---|---
Reason | - | \(C_1\) | - | - | \(C_2\) | \(C_4\)

▶ propagate consequences
Conflict Driven Clause Learning (CDCL) in a Nutshell

\[ F = (\overline{a} \lor b) \land (\overline{b} \lor d \lor e) \land (c \lor d) \land (\overline{a} \lor \overline{b} \lor \overline{e} \lor f) \land (\overline{a} \lor \overline{e} \lor f) \land (d \lor f) \land (\overline{c} \lor e \lor f) \]

\[ C_5 = (\overline{a} \lor \overline{e} \lor f) \]

\[ C_7 = C_4 \otimes C_5 : (\overline{a} \lor b \lor \overline{e}) \]

create and add the learned clause to the formula
Conflict Driven Clause Learning (CDCL) in a Nutshell

\[ F = (\overline{a} \lor b) \land (\overline{b} \lor \overline{d} \lor e) \land (c \lor d) \land (\overline{a} \lor \overline{b} \lor \overline{e} \lor f) \land (\overline{a} \lor \overline{e} \lor f) \land (d \lor \overline{f}) \land (\overline{c} \lor \overline{e} \lor f) \]

\[ C_5 = (\overline{a} \lor \overline{e} \lor f) \]

\[ C_7 = C_4 \otimes C_5 : (\overline{a} \lor \overline{b} \lor \overline{e}) \]

\[ C_8 = C_7 \otimes C_1 : (\overline{a} \lor \overline{e}) \]

▶ create and add the learned clause to the formula
Conflict Driven Clause Learning (CDCL) in a Nutshell

\[ F = (\overline{a} \lor b) \land (\overline{b} \lor d \lor e) \land (c \lor d) \land (\overline{a} \lor \overline{b} \lor \overline{e} \lor f) \land (\overline{a} \lor \overline{e} \lor \overline{f}) \land (d \lor \overline{f}) \land (\overline{c} \lor e \lor f) \]

CDCL

found conflict

\[ C_5 = (\overline{a} \lor \overline{e} \lor \overline{f}) \]

\[ C_{7} = C_4 \otimes C_5 : (\overline{a} \lor \overline{b} \lor \overline{e}) \]

minimization

\[ C_{8} = C_{7} \otimes C_1 : (\overline{a} \lor \overline{e}) \]

Variable \( \begin{array}{cccccc}
  a & b & c & d & e & f \\
  \hline
  - & C_1 & - & - & C_8 & -
\end{array} \)

▶ backtrack, add \( C_8 \), and proceed with unit propagation
The CDCL Algorithm

**CDCL** (CNF formula $F$)

**Input:** A formula $F$ in CNF  
**Output:** The solution SAT or UNSAT of this formula

1. $J := ()$  // start with empty interpretation
2. while true  
3. while $(x) \in F|_J$ do  // unit rule
4. $J := Jx$
5. if $[] \in F|_J$ then  // conflict
6. if $\exists y \in J$, such that $J = J' y J''$ then
7. $F := F \cup C$ with $F \models C$ and $C \notin F$  // learning
8. $J := J'$  // backjumping
9. else return UNSAT  // unsatisfiability rule
10. else  // no empty clause in $F|_J$
11. if $\text{atoms}(J) \supseteq \text{atoms}(F)$ then return SAT  // satisfiability rule
12. else $J := Jz$ with $\text{atoms}(z) \subseteq \text{atoms}(F)$  // decision rule
Conclusions of the CDCL Algorithm

- Heavily depends on the order of the decision variables
- Non-Chronological backtracking, backjumping
- Learning of new clauses
- No mentioned here: restarts, clause removal

Question: can the CDCL algorithm simulate the DPLL algorithm?
An Intuitive Abstraction of SAT Solver Techniques
Finding an Exit in a Maze

- Some rules
  - starting point is located in the left column
  - exit is on the right side (if there exists one)
  - search decisions can be done only when moving right
  - when moving left, use backtracking

- Property: satisfiable formulas correspond to mazes that can be solved by searching right only
The DPLL Algorithm

- Heuristics:
  - pick the highest possible column
  - then, pick the lowest column
The DPLL Algorithm

- Heuristics:
  - pick the highest possible column
  - then, pick the lowest column
The DPLL Algorithm

- Heuristics:
  - pick the highest possible column
  - then, pick the lowest column
The DPLL Algorithm

- No decision, hence *propagate*
The DPLL Algorithm

- No exit (conflict), hence backtrack
The DPLL Algorithm

- No exit in the upper search space, hence **backtrack**
The DPLL Algorithm

- Do next search decision
The DPLL Algorithm

- Enter the same search space as before
The CDCL Algorithm (conflict driven clause learning)

- Choose,
- Propagate,
- and Backtrack after Conflict
- ... enters the same search space again and again.
- Let’s go a few steps back ...
The CDCL Algorithm (conflict driven clause learning)

- No decision, hence propagate
The CDCL Algorithm (conflict driven clause learning)

- No exit (conflict), hence backtrack
The CDCL Algorithm (conflict driven clause learning)

▶ ... and learn a clause
The CDCL Algorithm (conflict driven clause learning)

- No exit in upper search space, backtrack and learn
The CDCL Algorithm (conflict driven clause learning)

Do next search decision
The CDCL Algorithm (conflict driven clause learning)

▶ Does not enter the same search space as before
Motivating Clause Removal
How to perform Unit Propagation

- When is a unit clause in the reduct?
- How to find a unit clause in the reduct, especially in the CDCL algorithm?
Coming Next

- Simplification
- Parallel Search