SAT Solving – Extensions

Steffen Hölldobler
International Center for Computational Logic
Technische Universität Dresden
Germany

- Heuristics
- Polynomial Sub-Classes
- Backdoors
- Simplification
- Implementation
- Combining Systematic and Stochastic Solvers
- Parallelization

"Logic is everywhere ..."
Variable Selection Heuristics – VSIDS

- **Variable State Independent Decay Sum**

- To each variable $A$ an *activity* $activity(A)$ is assigned

- **Initialization**
  - random, frequency of occurrence in given formula, or 1

- **Parameter** $decay \geq 1$ (often $decay := 1/0.95$)

- **Increment value** $inc$ (initially set to $inc := 1$)

- At each conflict do
  - $activity(A) := activity(A) \times inc$
    for each $A$ occurring in the derivation of the learned clause
  - $inc = inc \times decay$

- Pick the variable with the highest activity
Variable Selection Heuristics – VMTF

- **Variable Move to Front**

- Like VSIDS except the following
  - At each conflict do
    - $activity(A) := inc$
      for each $A$ occurring in the derivation of the learned clause
    - $inc := inc \times decay$
Variable Selection Heuristics – BerkMin

► Berkeley-Minsk SAT solver

► Store conflict clauses in a stack

► For each variable $A$
  ▶ $activity(A)$ counts the number of conflict clauses in which $A$ occurs
  ▶ $activity(A)$ is periodically divided by a small constant $\geq 1$

► Selection
  ▶ Select a variable with the highest activity occurring in the top-most unsatisfied clause of the stack
  ▶ If no such clause exists, select a variable with the highest activity
Variable Selection Heuristics – MOMS

- **Maximum number of Occurrences on clauses of Minimum Size**

- Let $m$ be the minimum clause length of formula $F$

- For each literal $L$
  let $h(L)$ be the number of occurrences of $L$ in clauses of length $m$

- Pick a literal with highest $h$-value
Polarity Selection Heuristics

- Random
  - Pick a random polarity

- Ratio Heuristics
  - Pick the polarity according to a predefined ratio between positive and negative literals

- Jeroslaw-Wang Heuristics
  - For any literal $L$ occurring in $F$ let $h(L) = \sum_{C \in F, L \in C} 2^{-|C|}$
  - Select polarity which leads to higher $h$-value

- Phase Saving / Progress Saving
  - Use the last polarity the variable had before it was backtracked
  - Use any other heuristics if the variable was not assigned before
Restart-Schedule Heuristics – Geometric Series

► Geometric series

▷ Let $decay \geq 1$
▷ Let $c$ be a counter (usually initialized with a value in $[100, 1000]$)
▷ Schedule a restart after the next $c$ conflicts
▷ When a restart is scheduled set $c := c \times decay$


► Nested Geometric Series

▷ Use two geometric series, where the outer is used as limit for the inner one
▷ Use the inner series as above until it exceeds the current value of the outer
▷ Reset the inner and increase the limit of the outer series

Restart-Schedule Heuristics – Luby Series

- Consider the Luby series
  
  \[1 1 2 1 1 2 4 1 1 2 4 8 \ldots\]

- Let \( f \) be a factor (usually set to a value in \([1, 512]\))

- Let \( c \) be a counter (initially set to 0)

- Let \( r \) be a counter (initially set to 1)

- At each conflict do
  - \( c := c + 1 \)
  - restart if \( c > f \times Luby[r] \)

- At each restart do
  - \( r := r + 1 \)

Huang: The Effect of Restarts on the Efficiency of Clause Learning
Remove Heuristics

► Usually, short clauses are not removed at all
  ▶ Let $C$ be a clause.
  ▶ $C$ is not removed if $|C| < n$, where $n \in \mathbb{N}$
  ▶ Often $n = 3$

Remove Heuristics – Activity Removal

Like VSIDS except the following

- to each clause an activity is assigned
- the activity is updated whenever the clause is used in a linear resolution derivation of a learnt clause

Let $C$ be a learned clause.

Let $L(C) = \{ \ell \mid \ell \text{ is the level of some literal occurring in } C \}$

Let $n \in \mathbb{N}$

Clause $C$ is removed if $|L(C)| \geq n$

Remove Heuristics – Progress Saving Measure

- Is based on the phase saving polarity heuristics
- Let $C$ be a clause
- Let $\mathcal{P}$ be the set of saved literal polarities, i.e. for each atom $A$ we find $A \in \mathcal{P}$ if the last used polarity of $A$ was positive and $\overline{A} \in \mathcal{P}$ otherwise
- Let $\text{psm}_\mathcal{P}(C) = |\mathcal{P} \cap C|$ 
- Remove clauses with a high psm-value
Heuristics – Parameter Selection


- **Idea** Use a stochastic local search algorithm on the parameter space of a SAT-solver to find a *good* parameter setting
Polynomial Sub-Classes

- **2SAT** \( F \) is in 2SAT iff each clause occurring in \( F \) has at most two literals

- **Horn** \( F \) is in Horn iff each clause has at most one positive literal

- **AHorn** \( F \) is in AHorn (anti Horn) iff each clause has at most one negative literal

- **RHorn** \( F \) is in RHorn (renamable Horn) iff there is a mapping \( \Phi \) from variables to literals such that after applying \( \Phi \) and modulo double negation \( F \) is in Horn

  ▶ Let \( F = \langle [1, 2], [\bar{3}, \bar{4}] \rangle \)

  ▶ Let \( \Phi = \{1 \mapsto \bar{5}, 3 \mapsto \bar{6}\} \)

  ▶ After applying \( \Phi \) and modulo double negation we obtain \( F = \langle [\bar{5}, 2], [6, \bar{4}] \rangle \)

- **UP+PL** \( F \) is in UP+PL iff it can be solved by applying only UNIT and PURE
2SAT – Backtrack Once


- Let $F$ be a 2SAT-formula and $J$ a partial interpretation

- Procedure $\text{unitPropagate}(F :: J)$
  - computes the closure of $F :: J$ under UNIT

- Procedure $\text{BTOSAT}(F :: J)$
  while $[] \not\in F|_J$ and $F|_J \neq \lambda$
  - choose an unassigned literal $L$
  - $F :: J' := \text{unitPropagate}(F :: J, \bar{L})$
  - if $[] \in F|_{J'}$ then $F :: J := \text{unitPropagate}(F :: J, \bar{L})$ else $F :: J := F :: J'$
  if $[] \in F|_J$ then return $\text{unsatisfiable}$ else return $J$
BTOSAT – Examples

- Suppose literals are assigned in their natural order
- Suppose positive literals are preferred
- What happens if BTOSAT is applied to
  \[ F = \langle [1, 2], [1, 3], [2, 4], [4, \overline{3}] \rangle \]
  \[ F = \langle [1, 2], [2, 3], \ldots, [8, 9], [8, \overline{9}], [8, 9] \rangle \]?
  \[ \Rightarrow \text{next slide} \]
  \[ F = \langle [1, 2], [2, 3], \ldots, [9, 10], [10, 11], \ldots, [18, 19],
     [1, 20], [1, 20], \ldots, [9, 20], [9, 20] \rangle \]?
  \[ \Rightarrow \text{Exercise} \]

- The worst-case complexity of BTOSAT is \( O(nm) \), where
  \[ n \text{ is the number of variables and} \]
  \[ m \text{ is the number of clauses in } F \]
BTOSAT – A Derivation

Let $F = \langle [1, 2], [2, 3], \ldots, [8, 9], [8, \bar{9}], [8, 9] \rangle$ and $J = ()$

We obtain $F :: ()$

$\sim \text{DECIDE} \quad F :: (\hat{1})$

$F|_{(\hat{1})} = \langle [2], [2, 3], \ldots, [8, 9], [8, \bar{9}], [8, 9] \rangle$

$F :: ([1, 2])$

$F|_{(1, 2)} = \langle [3], \ldots, [8, 9], [8, \bar{9}], [8, 9] \rangle$

$\sim \text{UNIT} \quad F :: ([1, 2, \ldots, 8])$

$F|_{(1,2,\ldots,8)} = \langle [9], \bar{9} \rangle$

$\sim \text{UNIT} \quad F :: ([1, 2, \ldots, 8, 9])$

$F|_{(1,2,\ldots,8,9)} = \langle [3] \rangle$

$\sim \text{UNIT} \quad F :: ([1, 2, \ldots, 8, 9])$

$F|_{(1,2,\ldots,8,9)} = \langle [3] \rangle$

$\sim \text{UNIT} \quad F :: ([1, 2, \ldots, 8])$

$F|_{(1,2,\ldots,8)} = \langle [9] \rangle$

$\sim \text{UNIT} \quad F :: ([1, 2, \ldots, 8, 9])$

$F|_{(1,2,\ldots,8,9)} = \langle [3] \rangle$

$\sim \text{UNIT} \quad F :: ([1, 2, \ldots, 8])$

$F|_{(1,2,\ldots,8)} = \langle [9] \rangle$

$\sim \text{UNIT} \quad F :: ([1, 2, \ldots, 8, 9])$

$F|_{(1,2,\ldots,8,9)} = \langle [3] \rangle$

$\sim \text{UNIT} \quad F :: ([1, 2, \ldots, 8, 9])$

$F|_{(1,2,\ldots,8,9)} = \langle [9] \rangle$

$\sim \text{UNIT} \quad F :: ([1, 2, \ldots, 8])$

$F|_{(1,2,\ldots,8)} = \langle [9] \rangle$

$\sim \text{UNIT} \quad F :: ([1, 2, \ldots, 8, 9])$

$F|_{(1,2,\ldots,8,9)} = \langle [9] \rangle$

$\sim \text{UNIT} \quad F :: ([1, 2, \ldots, 8])$

$F|_{(1,2,\ldots,8)} = \langle [9] \rangle$

$\sim \text{UNIT} \quad F :: ([1, 2, \ldots, 8, 9])$

$F|_{(1,2,\ldots,8,9)} = \langle [9] \rangle$

$\sim \text{UNIT} \quad F :: ([1, 2, \ldots, 8])$

$F|_{(1,2,\ldots,8)} = \langle [9] \rangle$

$\sim \text{UNIT} \quad F :: ([1, 2, \ldots, 8, 9])$

$F|_{(1,2,\ldots,8,9)} = \langle [9] \rangle$

$\sim \text{UNIT} \quad F :: ([1, 2, \ldots, 8])$

$F|_{(1,2,\ldots,8)} = \langle [9] \rangle$

$\sim \text{UNIT} \quad F :: ([1, 2, \ldots, 8, 9])$

$F|_{(1,2,\ldots,8,9)} = \langle [9] \rangle$

$\sim \text{UNIT} \quad F :: ([1, 2, \ldots, 8])$

$F|_{(1,2,\ldots,8)} = \langle [9] \rangle$

$\sim \text{UNIT} \quad F :: ([1, 2, \ldots, 8, 9])$

$F|_{(1,2,\ldots,8,9)} = \langle [9] \rangle$

$\sim \text{UNIT} \quad F :: ([1, 2, \ldots, 8])$

$F|_{(1,2,\ldots,8)} = \langle [9] \rangle$

$\sim \text{UNIT} \quad F :: ([1, 2, \ldots, 8, 9])$

$F|_{(1,2,\ldots,8,9)} = \langle [9] \rangle$
Let $F$ be a 2SAT-formula

We distinguish between permanent and temporary partial interpretations

$F\mid_{permVal}$
\begin{itemize}
  \item denotes the reduct of $F$ wrt a permanent partial interpretation
\end{itemize}

$F\mid_{tempVal\_permVal}$
\begin{itemize}
  \item denotes the reduct of $F$ wrt a permanent and a temporary partial interpretation, where the permanent interpretation overrules the partial one
\end{itemize}
2SAT – BinSAT

► Procedure \textit{unitPropagate}(F) computes the closure of \( F \) under UNIT with respect to and setting \( \text{permVal}(A) \) and \( \text{permVal}(\overline{A}) \) accordingly

► Procedure \textit{BinSAT}(F)

for each atom \( A \) occurring in \( F \) do

\begin{itemize}
  \item \( \text{tempVal}(A), \text{tempVal}(\overline{A}), \text{permVal}(A), \text{permVal}(\overline{A}) := \text{nil} \)
\end{itemize}

\( F := \text{unitPropagate}(F) \)

while \( [] \not\in F|_{\text{permVal}} \) and \( (\exists L) \text{permVal}(L) = \text{tempVal}(L) = \text{nil} \) do

\begin{itemize}
  \item \( \text{tempUnitPropagate}(L) \)
\end{itemize}

If \( [] \in F|_{\text{permVal}}^{\text{tempVal}} \) then return \textit{unsatisfiable} else return \textit{satisfiable}
Let $F$ be a 2SAT-formula

Let $L$ be an unassigned or a temporarily assigned literal

Procedure $\text{tempUnitPropagate}(L)$

- if $\text{tempVal}(L) = \bot$ (a conflict has occurred) then do
  - $F := \text{unitPropagate}(F \land [L])$
  - return

- $\text{tempVal}(L) := \top; \text{tempVal}(\overline{L}) = \bot$

for each $[\overline{L}, L'] \in F$ do

- if $[] \in F|_{\text{permVal}}$ then return
- if $\text{tempVal}(L') \neq \top$ then $\text{tempUnitPropagate}(L')$
BinSAT returns only *satisfiable*

In this case, a model can be generated as follows

For each $A$ do

- if $permVal(A) \neq nil$ then $A$ is assigned $permVal(A)$
- otherwise, $A$ is assigned $tempVal(A)$
Suppose literals assigned in their natural order. 

Suppose positive literals are preferred. 

What happens if BinSAT is applied to 

\[ F = \langle [1, 2], [1, 3], [2, 4], [4, 3] \rangle? \]

\[ F = \langle [1, 2], [2, 3], \ldots, [8, 9], [8, \bar{9}], [\bar{8}, 9], [8, 9] \rangle? \]

\[ F = \langle [1, 2], [2, 3], \ldots, [9, 10], [10, 11], \ldots, [18, 19], [1, 20], [1, 20], \ldots, [9, 20], [9, 20] \rangle? \]

The worst-case complexity of BinSAT is \( O(m) \), where 

\( m \) is the number of clauses in \( F \)
2SAT – BinSAT – A Derivation

- **Notation**
  - $F :: permVal :: tempVal$
    - denotes a formula with a permanent and a temporary partial interpretation

- **Let** $F = \langle [1, 2], [1, 3], [2, \bar{4}], [4, \bar{3}] \rangle$.

- **We obtain**

  $F :: () :: ()$
  $\xrightarrow{\text{tempDECIDE}} F :: () :: (\bar{1}) \quad \{[1, 2]\}$
  $\xrightarrow{\text{tempUNIT}} F :: () :: (1, 2) \quad \{[2, \bar{4}]\}$
  $\xrightarrow{\text{tempUNIT}} F :: () :: (1, 2, \bar{4}) \quad \{[4, \bar{3}]\}$
  $\xrightarrow{\text{tempUNIT}} F :: () :: (1, 2, \bar{4}, \bar{3}) \quad \emptyset$
  $\xrightarrow{\text{SAT}} F :: \text{SAT}$
2SAT – BinSAT – Another Derivation

- Let \( F = \langle [\overline{1}, 2], [\overline{2}, 3], \ldots, [\overline{8}, 9], [\overline{8}, \overline{9}], [8, 9] \rangle \)

- We obtain \( F :: () :: () \)

\[ \sim_{\text{tempDECIDE}} F :: () :: (\overline{1}) \quad \{[\overline{1}, 2]\} \]

\[ \sim_{\text{tempUNIT}} F :: () :: (\overline{1}, 2) \quad \{[\overline{2}, 3]\} \]

\[ \ldots \]

\[ \sim_{\text{tempUNIT}} F :: () :: (\overline{1}, 2, \ldots, 8) \quad \{[\overline{8}, 9], [\overline{8}, \overline{9}]\} \]

\[ \sim_{\text{tempUNIT}} F :: () :: (\overline{1}, 2, \ldots, 8, 9) \quad \{[9, 8], [9, \overline{8}]\} \]

\[ \sim F :: (\overline{8}) :: (\overline{1}, 2, \ldots, 8, 9) \]

\[ \sim_{\text{UNIT}} F :: (\overline{8}, 9) :: (\overline{1}, 2, \ldots, 8, 9) \quad [\quad] \in F|_{(\overline{8}, 9)} \]

\[ \sim_{\text{UNSAT}} F :: \text{UNSAT} \]

- **Note**

\[ \sim \] denotes the call of \textit{unitPropagate} within \textit{tempUnitPropagate}.

- In general, temporary partial interpretations are kept but they may be overwritten by the permanent ones.
Backdoors


- Why show SAT solvers such a good scaling behavior although SAT is in NP?
  - Practical combinatorial problem instances have a substantial amount of (hidden) tractable sub-structure
  - New algorithmic techniques exploit such tractable structure
Let $F$ be a SAT-instance.

A sub-solver $S$ given $F$ as input satisfies the following:

- **Trichotomy** $S$ either rejects $F$ or determines $F$ correctly.
- **Efficiency** $S$ runs in polynomial time.
- **Trivial solvability** $S$ can determine if $F$ is trivially true (i.e. is empty) or trivially false (i.e. contains the empty clause).
- **Self-reducibility** If $S$ determines $F$, then it determines $F|_J$ for any (partial) interpretation $J$. 
Sub-Solver – Example

2SAT

- **Trichotomy**  S must reject a formula $F$ if it is not 2SAT, i.e. if $F$ contains a clause with more than 2 literals
- **Efficiency**  S must solve 2SAT in polynomial time like eg. BTOSAT or BinSAT
- **Trivial Solvability**  obvious
- **Self-reducibility**  If $F$ is in 2SAT, then $F|_J$ is also in 2SAT
Backdoors

- Let $S$ be a sub-solver and $F$ a SAT-instance.
- A non-empty subset $\mathcal{B} \subseteq \text{atoms}(F)$ is a weak backdoor in $F$ for $S$ if for some $J : \mathcal{B} \rightarrow \{\top, \bot\}$, $S$ returns a satisfying assignment for $F|_J$.
- A non-empty subset $\mathcal{B} \subseteq \text{atoms}(F)$ is a strong backdoor in $F$ for $S$ if for all $J : \mathcal{B} \rightarrow \{\top, \bot\}$, $S$ returns a satisfying assignment for $F|_J$. 
Backdoors – Example 1

- Consider $F = \langle [2, 3], [1, 3, 4], [\overline{1}, 6], [\overline{1}, 5, \overline{6}], [4, \overline{5}], [2, \overline{5}, \overline{6}] \rangle$

- $\mathcal{B} = \{1\}$ is a weak backdoor in $F$ for UP+PL
  - Let $F' = F|_1 = \langle [\overline{2}, 3], [6], [5, \overline{6}], [4, \overline{5}], [2, \overline{5}, \overline{6}] \rangle$
  - We obtain $F' :: ()$
  - $F' :: (6)$
  - $F' :: (6, 5)$
  - $F' :: (6, 5, 2)$
  - $F' :: (6, 5, 2, 3)$
  - $F' :: (6, 5, 2, 3, 4)$
  - $\leadsto \text{SAT} F' :: \text{SAT}$
  - $F'|_{(6)} = \langle [\overline{2}, 3], [5], [4, \overline{5}], [2, \overline{5}] \rangle$
  - $F'|_{(6,5)} = \langle [\overline{2}, 3], [4], [2] \rangle$
  - $F'|_{(6,5,2)} = \langle [3], [4] \rangle$
  - $F'|_{(6,5,2,3)} = \langle [4] \rangle$
  - $F'|_{(6,5,2,3,4)} = \langle \rangle$
Backdoors – Example 2

- **Reconsider** $F = \langle [2, 3], [1, 3, 4], [\overline{1}, 6], [\overline{1}, 5, \overline{6}], [4, \overline{5}], [2, \overline{5}, \overline{6}] \rangle$

- $\mathcal{B} = \{1, 2\}$ is a strong backdoor in $F$ for 2SAT because
  - $F_1 = F|_{1,2} = \langle [3], [6], [5, \overline{6}], [4, \overline{5}] \rangle$ and $(4, 5, 6, 3) \models F_1$
  - $F_2 = F|_{\overline{1},2} = \langle [3], [\overline{3}, 4], [4, \overline{5}] \rangle$ and $(\overline{5}, 4, 3) \models F_2$
  - $F_3 = F|_{1,\overline{2}} = \langle [6], [5, \overline{6}], [4, \overline{5}] \rangle$ and $(4, 5, 6) \models F_3$
  - $F_4 = F|_{\overline{1},\overline{2}} = \langle [\overline{3}, 4], [4, \overline{5}], [\overline{5}, \overline{6}] \rangle$ and $(\overline{5}, 4, 3) \models F_4$
Minimal and Smallest Backdoors

Let $S$ be a sub-solver and $F$ a SAT-instance.

A (weak/strong) backdoor $B$ in $F$ for $S$ is said to be **minimal** iff no proper subset of $B$ is a (weak/strong) backdoor in $F$ for $S$.

A (weak/strong) backdoor $B$ in $F$ for $S$ is said to be **smallest** iff it is minimal and $|B| \leq |B'|$ for any minimal (weak/strong) backdoor $B'$ in $F$ for $S$. 

Minimal and Smallest Backdoors – Examples

Let $G = \langle [1, 2, 4], [4, 6], [4, 6, 7], [5, 6], [5, 6, 7] \rangle$

$F = \langle [1, 3], [2, 3], [3, 4, 5] \rangle \land G$

- $G$ can be solved by a Horn sub-solver
- $(1, 4, 5) \models G$

- $B_1 = \{3, 4\}$ is a strong backdoor in $F$ for Horn \(\leadsto\) Exercise
  - Neither $\{3\}$ nor $\{4\}$ are strong backdoors in $F$ for Horn
  - $B_1 = \{3, 4\}$ is a minimal strong backdoor in $F$ for Horn

- $B_2 = \{1, 3, 4\}$ is another strong backdoor in $F$ for Horn
  - $B_2$ is not a minimal strong backdoor in $F$ for Horn

- $B_3 = \{1, 2, 4\}$ is yet another strong backdoor in $F$ for Horn. \(\leadsto\) Exercise
  - $B_3$ is not a smallest strong backdoor in $F$ for Horn
  - $B_1$ is a smallest strong backdoor in $F$ for Horn \(\leadsto\) Exercise

- $B_4 = \{3, 5\}$ is another smallest strong backdoor in $F$ for Horn \(\leadsto\) Exercise
Backdoors – Remarks

- Backdoors exist for each $F$
- Given a weak backdoor $\mathcal{B}$ in $F$
  - The search cost for solving $F$ is of order $2^{|\mathcal{B}|}$
- The size of backdoors in practical problem instances may be surprisingly small

<table>
<thead>
<tr>
<th>instance</th>
<th># vars</th>
<th># clauses</th>
<th>backdoor</th>
<th>fraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>logistics.c</td>
<td>6783</td>
<td>437431</td>
<td>12</td>
<td>0.0018</td>
</tr>
<tr>
<td>3bitadd_32</td>
<td>8704</td>
<td>32316</td>
<td>53</td>
<td>0.0061</td>
</tr>
<tr>
<td>pipe_01</td>
<td>7736</td>
<td>26087</td>
<td>23</td>
<td>0.0030</td>
</tr>
<tr>
<td>qg_30.1</td>
<td>1235</td>
<td>8523</td>
<td>14</td>
<td>0.0113</td>
</tr>
<tr>
<td>qg_35.1</td>
<td>1597</td>
<td>10658</td>
<td>15</td>
<td>0.0094</td>
</tr>
</tbody>
</table>

- Given $F$ with $|\text{var}(F)| = n$, $k \geq 0$, and sub-solver $S$. The problem whether there exists a (weak/strong) backdoor in $F$ for $S$ of size $k$ is $\mathcal{NP}$-hard
Backdoors and Parameterized Complexity

Simplification – Warm Up (1)

► When are two formulas $F$ and $G$ semantically equivalent ($F \equiv G$)?

► Let $G = F \setminus \{C\}$
  ▶ Is $G \equiv F$?
  ▶ Under which condition is $G \equiv F$?
  ▶ How is the check performed?
  ▶ How complex is this check?

► How is $F \equiv_{\text{SAT}} G$ defined?

► Are there other redundancies which can be eliminated?
Which of the following statements is true?

- \( F \land L \equiv F \mid_L \)
- \( F \land L \equiv_{\text{SAT}} F \mid_L \)
- \( F \land L \models F \mid_L \)
- Let \( C \) and \( D \) be clauses with \( C \subseteq D \) in \( F \land D \models F \land C \)
- Let \( C \) and \( D \) be clauses with \( C \subseteq D \) in \( F \land C \models F \land D \)
How many models has the formula $F = (a \lor \bar{b}) \land (\bar{a} \lor b)$?

Enumerate the models!

How many models has the formula $F = (a \lor \bar{a}) \land (\bar{a} \lor a)$?

Enumerate the models!

Do you see a connection?
Equivalence Preserving Techniques

A clause $C$ is a **tautology** iff it contains a complementary pair of literals

- $(a \lor b \lor \overline{b})$
- Tautologies can be removed

Clause $C$ **subsumes** clause $D$ iff $C \subseteq D$

- $(a \lor b)$ subsumes $(a \lor b \lor \overline{c})$
- Subsumed clauses can be removed
Resolution

Remember 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 $L$.

We will write $C_1 \otimes_L C_2$ to denote the resolvent of $C_1$ and $C_2$ wrt $L$.

Is the addition of resolvents an equivalence preserving technique?

Shall we apply it?
Self-Subsuming Resolution

- **Suppose** \( C \lor L \otimes_L D \lor \bar{L} = D \)
- **Example** \( (a \lor b \lor d) \otimes_d (a \lor b \lor c \lor \bar{d}) = (a \lor b \lor c) \)
- **Observe** the resolvent subsumes one of its parent clauses
- **Example (continued)** Suppose, a CNF contains both parent clauses

\[ \ldots (a \lor b \lor d), (a \lor b \lor c \lor \bar{d}) \ldots \]

- If \( D \) is added, then \( D \lor \bar{L} \) can be removed
- which in essence removes \( \bar{L} \) from \( D \lor \bar{L} \)

\[ \ldots (a \lor b \lor d), (a \lor b \lor c) \ldots \]

- **Initially in the SATeLite preprocessor**
- now common in most solvers (i.e., as pre- and inprocessing)
Self-Subsuming Resolution – Example

➢ Self-Subsuming Resolution \( C \vee L \otimes_L D \vee \bar{L} = D \)

➢ Example

\[
\begin{align*}
( b \vee c ) \land ( \bar{a} \vee b \vee c ) \land \\
( \bar{a} \vee \bar{b} ) \land ( \bar{a} \vee \bar{b} \vee \bar{d} ) \land \\
( a \vee \bar{c} ) \land ( a \vee \bar{c} \vee \bar{d} )
\end{align*}
\]
Probing (1)

► **Idea** use unit propagation do derive extra information

► **Vivification** of a clause $C = (L_1 \lor \cdots \lor L_n), C \in F$

► Unit propagation results in the empty clause

1. $F :: (\overline{L_1}, \ldots, \overline{L_i}) \sim_{UNIT}^* F :: J$, where $[] \in F|_J, i < n$

► Unit propagation implies another literal of the clause $C$

2. $F :: (\overline{L_1}, \ldots, \overline{L_i}) \sim_{UNIT}^* F :: J$, where $L_j \in J, i < j \leq n$

► Unit propagation implies another negated literal of the clause $C$

3. $F :: (\overline{L_1}, \ldots, \overline{L_i}) \sim_{UNIT}^* F :: J$, where $\overline{L_j} \in J, i < j \leq n$

► **Exploit** $F \models ((\overline{L_1} \land \cdots \land \overline{L_i}) \rightarrow L)$ and, hence, $F \equiv F \land (L_1 \lor \cdots \lor L_i \lor L)$

► By the above statements and self-subsuming resolution, replace $C$ with

1. $(L_1 \lor \cdots \lor L_i)$
2. $(L_1 \lor \cdots \lor L_i \lor L_j)$
3. $C \setminus \{L_j\}$
Failed Literal test for some literal $L$

- $F :: L \leadsto_{\text{UNIT}}^* F :: J$, where $[] \in F|_J$, then add the unit clause $\bar{L}$
- Could also apply conflict analysis
- Then: learn all UIP clauses (have to be units)

Test for entailed literals (also backbones, necessary assignments), and equivalent literals wrt $F$

- $F :: L \leadsto_{\text{UNIT}}^* F :: J_L$ where $J_L$ is the set of all implied literals of $L$
- $F :: \bar{L} \leadsto_{\text{UNIT}}^* F :: J_{\bar{L}}$ where $J_{\bar{L}}$ is the set of all implied literals of $\bar{L}$
- $L'$ is an entailed literal if $L' \in J_L \cap J_{\bar{L}}$
- $L'$ and $L$ are equivalent if $L' \in J_L$ and $\bar{L}' \in J_{\bar{L}}$
Simplification – Equivalence Preserving Techniques

- Unit propagation
- Subsumption
- Resolution, hyper binary resolution
- Self-subsuming resolution
- Hidden tautology elimination
- Asymmetric tautology elimination
- Probing
  - Clause vivification
  - Necessary assignments
  - Failed literals
- Adding and removing transitive implications (binary clauses)
- Higher reasoning: Gaussian elimination, Fourier-Motzkin method
- No need to construct a model, the found model can be used
Simplification – Equisatisfiability Preserving Techniques

- Model needs to be constructed
- Information required for model construction can be stored on a stack
- Reason: $F \sim_{\text{bad}} F' \sim_{\text{bad}} F'' \sim_{\text{bad}} F''' \ldots$
- Reconstruction processes this chain in the opposite direction
  - $\ldots J''' \rightarrow J'' \rightarrow J' \rightarrow J$
- Thus, techniques can be run in any order, and mixed with the good ones
- For all currently used techniques, this process is polynomial (linear in the stack)
Equivalent Literal Substitution

- Given a formula $F$ such that $F \models (L_1 \leftrightarrow L_2)$
  - then replace each occurrence of $L_1$ and $\overline{L_1}$ in $F$ by $L_2$ and $\overline{L_2}$, respectively
  - remove double negation

- How to find equivalent literals?
  - By probing
  - By analyzing the binary implication graph (each SCC is an equivalence)
    - $F \models (a \rightarrow b) \land (b \rightarrow c) \land (c \rightarrow a)$, then $F \models a \leftrightarrow b \leftrightarrow c$
  - By structural hashing
    - $F \models (L_1 \leftrightarrow (a \land b)) \land (L_2 \leftrightarrow (a \land b))$, then $F \models (L_1 \leftrightarrow L_2)$
    - Works for many other gate types and variable definitions
    - Weakness definitions have to be found (structurally or semantically)

- How to construct the model $J$ from $J'$?
  - If $L_2 \in J'$, then $J := (J' \setminus \{L_1, \overline{L_1}\}) \cup \{L_1\}$
  - If $\overline{L_2} \in J'$, then $J := (J' \setminus \{L_1, \overline{L_1}\}) \cup \{\overline{L_1}\}$
Variable Elimination by Clause Distribution

Given a formula $F$ in CNF and a literal $L$

$F_L = \{ C \in F \mid L \in C \}$

$F_L \otimes_L F_{\overline{L}} = \{ C \otimes_L D \mid C \in F_L \text{ and } D \in F_{\overline{L}} \}$

Given a formula $F$ in CNF, variable elimination (or DP resolution) removes a variable $X$ by replacing $F_X \cup F_{\overline{X}}$ by $F_X \otimes_X F_{\overline{X}}$

Example

\[
\begin{array}{c|ccc}
F_{\overline{X}} \setminus F_X & (X \lor c) & (X \lor d) & (X \lor \overline{a} \lor \overline{b}) \\
\hline
(X \lor a) & (a \lor c) & (a \lor d) & (a \lor \overline{a} \lor \overline{b}) \\
(\overline{X} \lor b) & (b \lor c) & (b \lor d) & (b \lor \overline{a} \lor \overline{b}) \\
(\overline{X} \lor \overline{e} \lor f) & (c \lor \overline{e} \lor f) & (d \lor \overline{e} \lor f) & (\overline{a} \lor \overline{b} \lor \overline{e} \lor f) \\
\end{array}
\]

Observe

$|F_X \otimes_X F_{\overline{X}}| > |F_X| + |F_{\overline{X}}|$

Exponential growth of clauses in general
Variable Elimination by Substitution

**Idea** Detect gates (or definitions) $X \leftrightarrow \text{GATE}(a_1, \ldots, a_n)$ in the formula and use them to reduce the number of added clauses.

**Possible gates**

<table>
<thead>
<tr>
<th>Gate</th>
<th>$G_X$</th>
<th>$G_{\bar{X}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND($a_1, \ldots, a_n$)</td>
<td>$(X \lor \bar{a}_1 \lor \cdots \lor \bar{a}_n)$</td>
<td>$(X \lor a_1), \ldots, (\bar{x} \lor a_n)$</td>
</tr>
<tr>
<td>OR($a_1, \ldots, a_n$)</td>
<td>$(X \lor \bar{a}_1), \ldots, (X \lor \bar{a}_n)$</td>
<td>$(\bar{X} \lor a_1 \lor \cdots \lor a_n)$</td>
</tr>
<tr>
<td>ITE($c$, $t$, $f$)</td>
<td>$(X \lor \bar{c} \lor \bar{t}), (X \lor c \lor \bar{f})$</td>
<td>$(\bar{X} \lor \bar{c} \lor t), (\bar{X} \lor c \lor f)$</td>
</tr>
</tbody>
</table>

**Variable elimination by substitution**

- Let $R_X = F_X \setminus G_X$ and $R_{\bar{X}} = F_{\bar{X}} \setminus G_{\bar{X}}$
- Replace $F_X \cup F_{\bar{X}}$ by $G_X \otimes R_X \cup G_{\bar{X}} \otimes R_X$

**Always less than** $F_X \otimes F_{\bar{X}}$
Variable Elimination by Substitution

Example of gate extraction: $X \leftrightarrow \text{AND}(a, b)$

- $F_X = (X \lor c) \land (X \lor \bar{d}) \land (X \lor \bar{a} \lor \bar{b})$
- $F_{\bar{X}} = (\bar{X} \lor a) \land (\bar{X} \lor b) \land (\bar{X} \lor \bar{e} \lor f)$

Example of substitution

\[
\begin{array}{c|c|c|c}
& R_X & G_X \\
\hline
G_{\bar{X}} & (X \lor a) & (X \lor \bar{d}) & (X \lor \bar{a} \lor \bar{b}) \\
\hline
R_{\bar{X}} & (\bar{X} \lor b) & (a \lor c) & (a \lor d) \\
& (\bar{X} \lor \bar{e} \lor f) & (b \lor c) & (b \lor \bar{d}) \\
\end{array}
\]

- Observe $|F_X \otimes F_{\bar{X}}| < |F_X| + |F_{\bar{X}}|$
Variable Elimination

- How can the model be reconstructed?
- Given $F$, we picked literal $X$, removed $F_X$ and $F_{\bar{X}}$, and added $F_X \otimes_X F_{\bar{X}}$
- A model $J$ does not contain a value for $X$
- How does it work?
Bounded Variable Addition

- Given a CNF formula $F$
- **Idea** Can we construct a logically equivalent or equisatisfiable formula $F'$ by introducing a new variable $X \notin VAR(F)$ such that $|F'| < |F|?$
- **Challenge** How to find suitable patterns for replacement?

**Reverse of variable elimination**

**Example** Replace the clauses

$$(a \lor c) \quad (a \lor d)$$
$$(b \lor c) \quad (b \lor d)$$
$$(c \lor \overline{e} \lor f) \quad (d \lor \overline{e} \lor f) \quad (\overline{a} \lor \overline{b} \lor \overline{e} \lor f)$$

by

$$(\overline{X} \lor a) \quad (\overline{X} \lor b) \quad (\overline{X} \lor \overline{e} \lor f)$$
$$(X \lor c) \quad (X \lor d) \quad (X \lor \overline{a} \lor \overline{b})$$
Factoring Out Subclauses

Example Replace

\[(a \lor b \lor c \lor d) \quad (a \lor b \lor c \lor e) \quad (a \lor b \lor c \lor f)\]

by

\[(X \lor d) \quad (X \lor e) \quad (X \lor f) \quad (\bar{X} \lor a \lor b \lor c)\]

Add 1 variable and 1 clause

Reduces number of occurrences of literals by 2

Not compatible with variable elimination, which would eliminate \(X\) immediately

So this does not work . . .
Bounded Variable Addition

**Example**  Smallest pattern that is compatible: replace

\[(a \lor d) (a \lor e)\]
\[(b \lor d) (b \lor e)\]
\[(c \lor d) (c \lor e)\]

by

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

- Adds 1 variable
- Removes 1 clause
Bounded Variable Addition

- **Possible Patterns**

\[
\begin{align*}
(X_1 \lor L_1) & \quad \ldots \quad (X_1 \lor L_k) \\
\vdots & \quad \vdots \\
(X_n \lor L_1) & \quad \ldots \quad (X_n \lor L_k)
\end{align*}
\]

\[\equiv \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{k} (X_i \lor L_j)\]

- **Replaced by**

\[
\bigwedge_{i=1}^{n} (Y \lor X_i) \quad \land \quad \bigwedge_{j=1}^{k} (\overbar{Y} \lor L_j)
\]

where \(Y\) is a new variable

- Suppose, \(k\) clauses share a literal \(L_j\)
- Suppose, \(n\) literals \(X_i\) appear in the clauses
- Then, \(nk - n - k\) clauses are removed
Bounded Variable Addition on AtMostOneZero (1)

Example  Encoding of AtMostOneZero($x_1, \ldots, x_n$)

\[(x_1 \vee x_2) \land (x_9 \vee x_{10}) \land (x_8 \vee x_{10}) \land (x_7 \vee x_{10}) \land (x_6 \vee x_{10}) \land \]
\[(x_1 \vee x_3) \land (x_2 \vee x_3) \land (x_8 \vee x_9) \land (x_7 \vee x_9) \land (x_6 \vee x_9) \land \]
\[(x_1 \vee x_4) \land (x_2 \vee x_4) \land (x_3 \vee x_4) \land (x_7 \vee x_8) \land (x_6 \vee x_8) \land \]
\[(x_1 \vee x_5) \land (x_2 \vee x_5) \land (x_3 \vee x_5) \land (x_4 \vee x_5) \land (x_6 \vee x_7) \land \]
\[(x_1 \vee x_6) \land (x_2 \vee x_6) \land (x_3 \vee x_6) \land (x_4 \vee x_6) \land (x_5 \vee x_6) \land \]
\[(x_1 \vee x_7) \land (x_2 \vee x_7) \land (x_3 \vee x_7) \land (x_4 \vee x_7) \land (x_5 \vee x_7) \land \]
\[(x_1 \vee x_8) \land (x_2 \vee x_8) \land (x_3 \vee x_8) \land (x_4 \vee x_8) \land (x_5 \vee x_8) \land \]
\[(x_1 \vee x_9) \land (x_2 \vee x_9) \land (x_3 \vee x_9) \land (x_4 \vee x_9) \land (x_5 \vee x_9) \land \]
\[(x_1 \vee x_{10}) \land (x_2 \vee x_{10}) \land (x_3 \vee x_{10}) \land (x_4 \vee x_{10}) \land (x_5 \vee x_{10}) \land \]

Replace \((x_i \vee x_j)\) with \(i \in \{1..5\}, j \in \{6..10\}\) by \((x_i \vee y), (x_j \vee \bar{y})\)
Example Encoding of AtMostOneZero($x_1, \ldots, x_n$)

\[
(x_1 \lor x_2) \land (x_9 \lor x_{10}) \land (x_8 \lor x_{10}) \land (x_7 \lor x_{10}) \land (x_6 \lor x_{10}) \land \\
(x_1 \lor x_3) \land (x_2 \lor x_3) \land (x_8 \lor x_9) \land (x_7 \lor x_9) \land (x_6 \lor x_9) \land \\
(x_1 \lor x_4) \land (x_2 \lor x_4) \land (x_3 \lor x_4) \land (x_7 \lor x_8) \land (x_6 \lor x_8) \land \\
(x_1 \lor x_5) \land (x_2 \lor x_5) \land (x_3 \lor x_5) \land (x_4 \lor x_5) \land (x_6 \lor x_7) \land \\
(x_1 \lor y) \land (x_2 \lor y) \land (x_3 \lor y) \land (x_4 \lor y) \land (x_5 \lor y) \land \\
(x_6 \lor \bar{y}) \land (x_7 \lor \bar{y}) \land (x_8 \lor \bar{y}) \land (x_9 \lor \bar{y}) \land (x_{10} \lor \bar{y})
\]

Replace matched pattern by

\[
(x_1 \lor z) \land (x_2 \lor z) \land (x_3 \lor z) \land (x_4 \lor \bar{z}) \land (x_5 \lor \bar{z}) \land (y \lor \bar{z})
\]
Example  Encoding of AtMostOneZero($x_1, \ldots, x_n$)

\[(x_1 \lor x_2) \land (x_9 \lor x_{10}) \land (x_8 \lor x_{10}) \land (x_7 \lor x_{10}) \land (x_6 \lor x_{10}) \land \]
\[(x_1 \lor x_3) \land (x_2 \lor x_3) \land (x_8 \lor x_9) \land (x_7 \lor x_9) \land (x_6 \lor x_9) \land \]
\[(x_1 \lor z) \land (x_2 \lor z) \land (x_3 \lor z) \land (x_7 \lor x_8) \land (x_6 \lor x_8) \land \]
\[(x_4 \lor \bar{z}) \land (x_5 \lor \bar{z}) \land (y \lor \bar{z}) \land (x_4 \lor x_5) \land (x_6 \lor x_7) \land \]
\[(x_4 \lor y) \land (x_5 \lor y) \land (x_6 \lor \bar{y}) \land (x_7 \lor \bar{y}) \land (x_8 \lor \bar{y}) \land \]
\[(x_9 \lor \bar{y}) \land (x_{10} \lor \bar{y}) \land \]

Replace matched pattern by

\[(x_6 \lor w) \land (x_7 \lor w) \land (x_8 \lor w) \land (x_9 \lor \bar{w}) \land (x_{10} \lor \bar{w}) \land (\bar{y} \lor \bar{w}) \]
Bounded Variable Addition

▶ How can the model be reconstructed?
Blocked Clauses

- A literal \( L \) in a clause \( C \) of a CNF \( F \) blocks \( C \) in \( F \) if for every clause \( D \in F_L \), the resolvent \( (C \setminus \{L\}) \cup (D \setminus \{\bar{L}\}) \) obtained from resolving \( C \) and \( D \) on \( L \) is a tautology.

- A clause is blocked if it contains a literal that blocks it.

- Example: Consider the formula \((a \lor b) \land (a \lor \bar{b} \lor \bar{c}) \land (\bar{a} \lor c)\)
  
  - First clause is not blocked.
  - Second clause is blocked by both \( a \) and \( \bar{c} \).
  - Third clause is blocked by \( c \).

- Proposition: Removal of an arbitrary blocked clause preserves satisfiability.
Blocked Clause Elimination (BCE)

- **BCE** While there is a blocked clause $C$ in a CNF $F$, remove $C$ from $F$

- **Example** Consider $(a \lor b) \land (a \lor \bar{b} \lor \bar{c}) \land (\bar{a} \lor c)$

  - After removing either $(a \lor \bar{b} \lor \bar{c})$ or $(\bar{a} \lor c)$, the clause $(a \lor b)$ becomes blocked

    - no clause with either $\bar{b}$ or $\bar{a}$

  - An extreme case in which BCE removes all clauses

- **Proposition** BCE is confluent
How can a model be reconstructed?

Given $F$, we picked clause $C$ with blocking literal $L$

$C$ was blocked with respect to $F_L$

A model $J$ might falsify $C$

How can it work?
Simplification Techniques – The Bad and Powerful

▶ Equisatisfiability Preserving Techniques
  ▶ (Bounded) variable elimination
  ▶ Bounded variable addition
  ▶ Blocked clause elimination
  ▶ Covered clause elimination
  ▶ Equivalent literal substitution
    ▶ based on SCCs in binary implication graphs
    ▶ based on structural hashing
    ▶ based on Probing
  ▶ Resolution asymmetric tautology elimination

▶ Need to store extra information to construct the model

▶ Not discussed here
  ▶ Adding redundant clauses
  ▶ Minimizing redundant clauses
Solving a Problem with SAT

- Research topics:
  - encode problems into CNF
  - simplify the problem
  - and search for a solution or prove there does not exist one
  - simplification during search
  - automatically translate naive encodings into sophisticated encodings