SAT Solving – Simplification

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

- Types of Redundancy
- Simplification Algorithms

"Logic is everywhere ..."
Given a formula $F$, when preserves removing a clause $C \in F$ equivalence?
Given a formula $F$, when preserves removing a clause $C \in F$ equivalence?

How is the above check performed?
Simplification – Warm Up

- Given a formula $F$, when preserves removing a clause $C \in F$ equivalence?

- How is the above check performed?

- How complex is this check?
Simplification – Warm Up

- Given a formula $F$, when preserves removing a clause $C \in F$ equivalence?

- How is the above check performed?

- How complex is this check?

- Are there other redundancies to preserve satisfiability?
Which of the following hold:

\[ F \land x \equiv F \mid x \]

\[ F \land x \equiv \text{SAT} \]

\[ F \land x \equiv F \mid x \]

Let \( C \) and \( D \) be clauses with \( D \subset C \):

\[ F \land D \mid = F \land C \]

Let \( D \subset C \):

\[ F \land C \mid = F \land D \]
Simplification – Warm Up

Which of the following hold:

\[ F \land x \equiv F_{|x} \]
Simplification – Warm Up

Which of the following hold:

\[ F \land x \equiv F|_x \]

\[ F \land x \equiv_{\text{SAT}} F|_x \]
Simplification – Warm Up

Which of the following hold:

- \( F \land x \equiv F\mid x \)
- \( F \land x \equiv_{\text{SAT}} F\mid x \)
- \( F \land x \models F\mid x \)
Simplification – Warm Up

► Which of the following hold:

► $F \land x \equiv F\rvert_x$

► $F \land x \equiv_{\text{SAT}} F\rvert_x$

► $F \land x \models F\rvert_x$

► Let $C$ and $D$ be clauses with $D \subset C : F \land D \models F \land C$
Simplification – Warm Up

► Which of the following hold:

► $F \land x \equiv F|_x$

► $F \land x \equiv_{SAT} F|_x$

► $F \land x \models F|_x$

► Let $C$ and $D$ be clauses with $D \subset C : F \land D \models F \land C$

► Let $D \subset C : F \land C \models F \land D$
Simplification – Warm Up

► How many (relevant partial) models has the formula \( F = (a \lor \neg b) \land (\neg a \lor b) \)?

► Enumerate the models!
Simplification – Warm Up

► How many (relevant partial) models has the formula \( F = (a \lor \neg b) \land (\neg a \lor b) \)?

► Enumerate the models!
Simplification – Warm Up

► How many (relevant partial) models has the formula \( F = (a \lor \neg b) \land (\neg a \lor b) \)?

► Enumerate the models!

► How many models has the formula \( F = (a \lor \neg a) \land (\neg a \lor a) \)?
Simplification – Warm Up

▶ How many (relevant partial) models has the formula $F = (a \lor \neg b) \land (\neg a \lor b)$?

▶ Enumerate the models!

▶ How many models has the formula $F = (a \lor \neg a) \land (\neg a \lor a)$?

▶ Enumerate the models!
Simplification – Warm Up

▶ How many (relevant partial) models has the formula $F = (a \lor \neg b) \land (\neg a \lor b)$?

▶ Enumerate the models!

▶ How many models has the formula $F = (a \lor \neg a) \land (\neg a \lor a)$?

▶ Enumerate the models!

▶ Do you see a connection?
Revision – Notation

- Given a formula $F$ in CNF and a literal $x$, then $F_x = \{ C \in F \mid x \in C \}$. 
Acknowledgement

► Some slides are based on slides from

► Marijn Heule,
The University of Texas
Austin
Equivalence Preserving Techniques
Tautologies and Subsumption

Definition (Tautology)
A clause $C$ is a tautology iff it contains a complementary pair of literals.

Example
The clause $(a \lor b \lor \overline{b})$ is a tautology.

Definition (Subsumption)
Clause $C$ subsumes clause $D$ iff $C \subseteq D$.

Example
The clause $(a \lor b)$ subsumes clause $(a \lor b \lor \overline{c})$. 
Self-Subsuming Resolution

\[
\frac{C \lor l}{D} \quad \frac{D \lor \overline{l}}{C \subseteq D}
\]

The resolvent \( D \) subsumes \( D \lor \overline{l} \).

\[
\frac{(a \lor b \lor l)}{D} \quad \frac{(a \lor b \lor c \lor \overline{l})}{(a \lor b \lor c)}
\]

Example: Assume a CNF contains both antecedents \((a \lor b \lor l) (a \lor b \lor c \lor \overline{l})\). If \( D \) is added, then \( D \lor \overline{l} \) can be removed, which in essence removes \( \overline{l} \) from \( D \lor \overline{l} \). Initially in the SATeLite preprocessor, now common in most solvers (i.e., as pre- and in-processing).
Self-Subsuming Resolution

\[ \frac{C \lor l}{D} D \lor \bar{l} \]
\[ C \subseteq D \]

resolvent \( D \) subsumes \( D \lor \bar{l} \)

**Example**

Assume a CNF contains both antecedents
\[ \ldots (a \lor b \lor l)(a \lor b \lor c \lor \bar{l}) \ldots \]

If \( D \) is added, then \( D \lor \bar{l} \) can be removed

which in essence removes \( \bar{l} \) from \( D \lor \bar{l} \)
\[ \ldots (a \lor b \lor l)(a \lor b \lor c) \ldots \]

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

Self-Subsuming Resolution

\[ \frac{C \lor l}{D} \quad \frac{D \lor \bar{l}}{C \subseteq D} \quad (a \lor b \lor l) \quad (a \lor b \lor c \lor \bar{l}) \]

resolvent \( D \) subsumes \( D \lor \bar{l} \)

Example: Remove literals using self-subsumption

\[
(a \lor b \lor c) \land (\bar{a} \lor b \lor c) \land \\
(\bar{a} \lor b \lor \bar{c}) \land (a \lor \bar{b} \lor c) \land \\
(\bar{a} \lor \bar{b} \lor d) \land (\bar{a} \lor \bar{b} \lor \bar{d}) \land \\
(a \lor \bar{c} \lor d) \land (a \lor \bar{c} \lor \bar{d})
\]
Self-Subsuming Example

Self-Subsuming Resolution

\[ \frac{C \lor l}{D} \quad \frac{D \lor \overline{l}}{D} \quad C \subseteq D \]

resolvent \( D \) subsumes \( D \lor \overline{l} \)

Example: Remove literals using self-subsumption

\[
\begin{align*}
(a \lor b \lor l) & \quad (a \lor b \lor c \lor \overline{l}) \\
(a \lor b \lor c) & \quad (a \lor b \lor c) \\
(a \lor b \lor \overline{c}) & \quad (a \lor b \lor c) \\
(a \lor b \lor d) & \quad (a \lor b \lor \overline{d}) \\
(a \lor \overline{c} \lor d) & \quad (a \lor \overline{c} \lor \overline{d})
\end{align*}
\]
Self-Subsuming Example

Self-Subsuming Resolution

\[
\begin{align*}
C \vee l & \quad D \vee \bar{l} \\
\hline
D & \quad C \subseteq D
\end{align*}
\]

resolvent \( D \) subsumes \( D \vee \bar{l} \)

Example: Remove literals using self-subsumption

\[
\begin{align*}
(b \vee c) & \quad (\bar{a} \vee b \vee c) \\
(\bar{a} \vee b) & \quad (a \vee \bar{b} \vee c) \\
(\bar{a} \vee \bar{b} \vee d) & \quad (\bar{a} \vee \bar{b} \vee \bar{d}) \\
(a \vee \bar{c} \vee d) & \quad (a \vee \bar{c} \vee \bar{d})
\end{align*}
\]
Self-Subsuming Example

Self-Subsuming Resolution

\[
\frac{C \lor l}{D} \quad \frac{D \lor \bar{l}}{C \subseteq D}
\]

\[ (a \lor b \lor l) \rightarrow (a \lor b \lor c \lor \bar{l}) \]

resolvent \( D \) subsumes \( D \lor \bar{l} \)

Example: Remove literals using self-subsumption

\[
( b \lor c ) \land ( \bar{a} \lor b \lor c ) \land \\
( \bar{a} \lor b ) \land ( a \lor c ) \land \\
( \bar{a} \lor \bar{b} \lor d ) \land ( \bar{a} \lor \bar{b} \lor \bar{d} ) \land \\
( a \lor \bar{c} \lor d ) \land ( a \lor \bar{c} \lor \bar{d} )
\]
Self-Subsuming Example

Self-Subsuming Resolution

\[
\frac{C \lor l}{D} \quad \frac{D \lor \bar{l}}{D \subseteq D}\quad \frac{(a \lor b \lor l)(a \lor b \lor c \lor \bar{l})}{(a \lor b \lor c)}
\]

resolvent \( D \) subsumes \( D \lor \bar{l} \)

Example: Remove literals using self-subsumption

\[
(b \lor c) \land (\bar{a} \lor b \lor c) \land (\bar{a} \lor b) \land (a \lor c) \land (\bar{a} \lor \bar{b} \lor \bar{d}) \land (a \lor \bar{c} \lor d) \land (a \lor \bar{c} \lor \bar{d})
\]
Self-Subsuming Example

Self-Subsuming Resolution

\[ \frac{C \lor l}{D} \quad \frac{D \lor \bar{l}}{C \subseteq D} \]

resolvent \( D \) subsumes \( D \lor \bar{l} \)

Example: Remove literals using self-subsumption

\[(b \lor c) \land (\bar{a} \lor b \lor c) \land (\bar{a} \lor b) \land (a \lor c) \land (\bar{a} \lor \bar{b}) \land (a \lor \bar{c} \lor \bar{d})\]
Self-Subsuming Example

Self-Subsuming Resolution

\[ \frac{C \lor l}{D} \quad \frac{D \lor \overline{l}}{C \subseteq D} \]

resolvent \( D \) subsumes \( D \lor \overline{l} \)

Example: Remove literals using self-subsumption

\[
\begin{align*}
& (b \lor c) \land (\overline{a} \lor b \lor c) \land \\
& (\overline{a}) \land (a \lor c) \land \\
& (\overline{a} \lor \overline{b}) \land (\overline{a} \lor \overline{b} \lor \overline{d}) \land \\
& (a \lor \overline{c}) \land (a \lor \overline{c} \lor \overline{d})
\end{align*}
\]
Self-Subsuming Example

Self-Subsuming Resolution

\[
\frac{C \lor l}{D} \quad \frac{D \lor \bar{l}}{C \subseteq D}
\]

resolvent \(D\) subsumes \(D \lor \bar{l}\)

Example: Remove literals using self-subsumption

\[
(b \lor c) \land (\bar{a} \lor b \lor c) \land \\
(\bar{a}) \land (a) \land \\
(\bar{a} \lor \bar{b}) \land (\bar{a} \lor \bar{b} \lor \bar{d}) \land \\
(a \lor \bar{c}) \land (a \lor \bar{c} \lor \bar{d})
\]
Self-Subsuming Example

Self-Subsuming Resolution

\[
\frac{C \lor l}{D} \quad \frac{D \lor \bar{l}}{C \subseteq D}
\]

resolvent \(D\) subsumes \(D \lor \bar{l}\)

Example: Remove literals using self-subsumption

\[
\begin{align*}
(a \lor b \lor l) & \quad (a \lor b \lor c \lor \bar{l}) \\
(a \lor b \lor c) & \quad (a \lor b \lor c)
\end{align*}
\]
Probing

- Idea: use unit propagation do derive extra information

- Vivification of a clause $C = (l_1 \lor \cdots \lor l_n)$, $C \in F$

  1. Unit propagation results in the empty clause:
     $F :: (\overline{l_1}, \ldots, \overline{l_i}) \sim_{UNIT}^* F :: J$, where $[] \in F|J$, $i < n$

Exploit:

$F|J = (\overline{l_1} \land \cdots \land \overline{l_i}) \rightarrow x$

Then, replace $C$ with

1. $C := (l_1 \lor \cdots \lor l_i)$
2. $C := (l_1 \lor \cdots \lor l_i \lor l_j)$
3. $C := C \{l_j\}$, by above statement, and self-subsuming
Probing

- **Idea:** use unit propagation do derive extra information

- **Vivification** of a clause $C = (l_1 \lor \cdots \lor l_n), C \in F$

  1. Unit propagation results in the empty clause:
     $$F :: (l_1, \ldots, \overline{l_i}) \Rightarrow^{\text{UNIT}} F :: J,$$
     where $[] \in F|J, i < n$

  2. Unit propagation implies another literal of the clause $C$
     $$F :: (l_1, \ldots, \overline{l_i}) \Rightarrow^{\text{UNIT}} F :: J,$$
     where $l_j \in J, i < j < n$:

  3. Unit propagation implies another negated literal of the clause $C$
     $$F :: (l_1, \ldots, \overline{l_i}) \Rightarrow^{\text{UNIT}} F :: J,$$
     where $l_j \in J, i < j < n$: 

Exploit:
$$F | \overline{l_1} \land \cdots \land \overline{l_i} \Rightarrow x,$$

Thus:
$$F \equiv F \land (l_1 \lor \cdots \lor l_i \lor x)$$

Then, replace $C$ with
$$C := (l_1 \lor \cdots \lor l_i)$$
Probing

- Idea: use unit propagation do derive extra information

- **Vivification** of a clause \( C = (l_1 \lor \cdots \lor l_n), C \in F \)

  1. Unit propagation results in the empty clause:
     \[ F :: (\overline{l_1}, \ldots, \overline{l_i}) \xrightarrow{\text{UNIT}}^* F :: J, \text{ where } [] \in F|_J, i < n \]
  2. Unit propagation implies another literal of the clause \( C \)
     \[ F :: (\overline{l_1}, \ldots, \overline{l_i}) \xrightarrow{\text{UNIT}}^* F :: J, \text{ where } l_j \in J, i < j < n: \]
  3. Unit propagation implies another negated literal of the clause \( C \)
     \[ F :: (\overline{l_1}, \ldots, \overline{l_i}) \xrightarrow{\text{UNIT}}^* F :: J, \text{ where } \overline{l_j} \in J, i < j < n: \]

- Exploit: \( F \models ((\overline{l_1} \land \cdots \land \overline{l_i}) \rightarrow x), \text{ hence } F \equiv F \land (l_1 \lor \cdots \lor l_i \lor x) \)
Probing

- Idea: use unit propagation do derive extra information

- **Vivification** of a clause $C = (l_1 \lor \cdots \lor l_n)$, $C \in F$
  1. Unit propagation results in the empty clause:
     $F ::= (\overline{l_1}, \ldots, \overline{l_i}) \leadsto^{\text{UNIT}} F :: J$, where $[] \in F|_J$, $i < n$
  2. Unit propagation implies another literal of the clause $C$
     $F ::= (\overline{l_1}, \ldots, \overline{l_i}) \leadsto^{\text{UNIT}} F :: J$, where $l_j \in J$, $i < j < n$:
  3. Unit propagation implies another negated literal of the clause $C$
     $F ::= (\overline{l_1}, \ldots, \overline{l_i}) \leadsto^{\text{UNIT}} F :: J$, where $\overline{l_j} \in J$, $i < j < n$:

- Exploit: $F \models ((\overline{l_1} \land \cdots \land \overline{l_i}) \rightarrow x)$, hence $F \equiv F \land (l_1 \lor \cdots \lor l_i \lor x)$

- Then, replace $C$ with
  1. $C := (l_1 \lor \cdots \lor l_i)$
Probing

▶ **Idea:** use unit propagation do derive extra information

▶ **Vivification** of a clause \( C = (l_1 \lor \cdots \lor l_n), \ C \in F \)

1. **Unit propagation results in the empty clause:**
   \[ F :: (\overline{l_1}, \ldots, \overline{l_i}) \sim^{\text{UNIT}} F :: J, \text{ where } [] \in F|J, \ i < n \]

2. **Unit propagation implies another literal of the clause** \( C \)
   \[ F :: (\overline{l_1}, \ldots, \overline{l_i}) \sim^{\text{UNIT}} F :: J, \text{ where } l_j \in J, \ i < j < n: \]

3. **Unit propagation implies another negated literal of the clause** \( C \)
   \[ F :: (\overline{l_1}, \ldots, \overline{l_i}) \sim^{\text{UNIT}} F :: J, \text{ where } \overline{l_j} \in J, \ i < j < n: \]

▶ **Exploit:** \( F \models ((\overline{l_1} \land \cdots \land \overline{l_i}) \rightarrow x), \text{ hence } F \equiv F \land (l_1 \lor \cdots \lor l_i \lor x) \)

▶ **Then, replace** \( C \) **with**

1. \( C := (l_1 \lor \cdots \lor l_i) \)
2. \( C := (l_1 \lor \cdots \lor l_i \lor l_j) \)
Probing

▶ Idea: use unit propagation do derive extra information

▶ Vivification of a clause $C = (l_1 \lor \cdots \lor l_n)$, $C \in F$

1. Unit propagation results in the empty clause:
   $F :: (\overline{l}_1, \ldots, \overline{l}_i) \rightsquigarrow_{\text{UNIT}}^* F :: J$, where $[] \in F|J$, $i < n$
2. Unit propagation implies another literal of the clause $C$
   $F :: (\overline{l}_1, \ldots, \overline{l}_i) \rightsquigarrow_{\text{UNIT}}^* F :: J$, where $l_j \in J$, $i < j < n$:
3. Unit propagation implies another negated literal of the clause $C$
   $F :: (\overline{l}_1, \ldots, \overline{l}_i) \rightsquigarrow_{\text{UNIT}}^* F :: J$, where $\overline{l}_j \in J$, $i < j < n$:

▶ Exploit: $F \models ((\overline{l}_1 \land \cdots \land \overline{l}_i) \rightarrow x)$, hence $F \equiv F \land (l_1 \lor \cdots \lor l_i \lor x)$

▶ Then, replace $C$ with

1. $C := (l_1 \lor \cdots \lor l_i)$
2. $C := (l_1 \lor \cdots \lor l_i \lor l_j)$
3. $C := C \setminus \{l_j\}$, by above statement, and self-subsuming
Failed Literal  test for some literal $l$

- $F :: (l) \leadsto^{*\text{UNIT}}_F J$, where $[] \in F|_J$, then add the unit clause $\neg l$
- 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 $F$

- $F :: (l) \leadsto^{*\text{UNIT}}_F J_l$, $J_l$ is the set of all implied literals of $l$
- $F :: (\neg l) \leadsto^{*\text{UNIT}}_F J_{\neg l}$, $J_{\neg l}$ is the set of all implied literals of $\neg l$
Probing

- **Failed Literal** test for some literal $l$
  - $F :: (l) \leadsto_{\text{UNIT}}^* F :: J$, where $[] \in F|_J$, then add the unit clause $\neg l$
  - 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 $F$
  - $F :: (l) \leadsto_{\text{UNIT}}^* F :: J_l$, $J_l$ is the set of all implied literals of $l$
  - $F :: (\neg l) \leadsto_{\text{UNIT}}^* F :: J_{\neg l}$, $J_{\neg l}$ is the set of all implied literals of $\neg l$

- $l'$ is an entailed literal if $l' \in J_l \cap J_{\neg l}$,
Probing

- **Failed Literal** test for some literal $l$
  - $F :: (l) \sim^{UNIT} F :: J$, where $[] \in F|J$, then add the unit clause $\neg l$
  - 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 $F$
  - $F :: (l) \sim^{UNIT} F :: J_l$, $J_l$ is the set of all implied literals of $l$
  - $F :: (\neg l) \sim^{UNIT} F :: J_{\neg l}$, $J_{\neg l}$ is the set of all implied literals of $\neg l$

- $l'$ is an entailed literal if $l' \in J_l \cap J_{\neg l}$

- $l'$ and $l$ are equivalent if $l' \in J_l$ and $\neg l' \in J_{\neg l}$
Simplification Techniques – Equivalence Preserving

▶ Equivalence Preserving Techniques:

▷ Unit Propagation
▷ Subsumption
▷ Resolution, (lazy) Hyper Binary Resolution
▷ Self-Subsuming Resolution (or Strengthening)
▷ Hidden Tautology Elimination
▷ Asymmetric Tautology Elimination
    both based on hidden or asymmetric literal addition
▷ 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
Equisatisfiability Preserving Techniques
Model Reconstruction

- Techniques preserve equisatisfiability, thus, model needs to be constructed
- Information required for model construction can be stored on a stack

- Reason: \( F \sim_{bad} F' \sim_{bad} F'' \sim_{bad} F''' \ldots \)

- 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 $F$, and $F \models (l_1 \leftrightarrow l_2)$,
then replace each occurrence of $l_1$ and $\overline{l_1}$ in $F$ by $l_2$ and $\overline{l_2}$, respectively,
and remove double negation.
Equivalent Literal Substitution

- Given a formula $F$, and $F \models (l_1 \leftrightarrow l_2)$,
  
  then replace each occurrence of $l_1$ and $\bar{l}_1$ in $F$ by $l_2$ and $\bar{l}_2$, respectively,
  
  and remove double negation

- How to find equivalences
  
  ▶ By probing
  
  ▶ By analyzing the binary implication graph (each SCC is an equivalence)
    
    $\Rightarrow F \models (a \rightarrow b) \land (b \rightarrow c) \land (c \rightarrow a)$, then $F \models a \leftrightarrow b \leftrightarrow c$. 

Steffen Hölldobler and Norbert Manthey
SAT Solving – Simplification
Equivalent Literal Substitution

- Given a formula $F$, and $F \models (l_1 \leftrightarrow l_2)$,

then replace each occurrence of $l_1$ and $\overline{l_1}$ in $F$ by $l_2$ and $\overline{l_2}$, respectively, and remove double negation.

- How to find equivalences
  - 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
    - $F \models (x \leftrightarrow (a \land b)) \land (y \leftrightarrow (a \land b))$, then $F \models (x \leftrightarrow y)$
    - Works for many other gate types, and variable definitions
    - Weakness: definitions have to be found (structural or semantically)
Equivalent Literal Substitution

Given a formula \( F \), and \( F \models (l_1 \leftrightarrow l_2) \),
then replace each occurrence of \( l_1 \) and \( \overline{l_1} \) in \( F \) by \( l_2 \) and \( \overline{l_2} \), respectively,
and remove double negation

How to find equivalences

- 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
  - \( F \models (x \leftrightarrow (a \land b)) \land (y \leftrightarrow (a \land b)) \), then \( F \models (x \leftrightarrow y) \)
  - Works for many other gate types, and variable definitions
  - Weakness: definitions have to be found (structural or semantically)

How to construct the model \( J \) from \( J' \)?:
Equivalent Literal Substitution

- Given a formula $F$, and $F \models (l_1 \leftrightarrow l_2)$,
  then replace each occurrence of $l_1$ and $\overline{l_1}$ in $F$ by $l_2$ and $\overline{l_2}$, respectively, and remove double negation

- How to find equivalences
  - 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
  - $F \models (x \leftrightarrow (a \land b)) \land (y \leftrightarrow (a \land b))$, then $F \models (x \leftrightarrow y)$
  - Works for many other gate types, and variable definitions
  - Weakness: definitions have to be found (structural or semantically)

- How to construct the model $J$ from $J'$?
  - If $l_2 \in J'$, then $J := (J' \setminus \{l_1, \neg l_1\}) \cup \{l_1\}$
  - If $\neg l_2 \in J'$, then $J := (J' \setminus \{l_1, \neg l_1\}) \cup \{\neg l_1\}$
Example VE by clause distribution

Definition (Variable elimination (VE))
Given a CNF formula $F$, \textit{variable elimination} (or DP resolution) removes a variable $x$ by replacing $F_x$ and $F_{\overline{x}}$ by $F_x \otimes_x F_{\overline{x}}$
Example VE by clause distribution

Definition (Variable elimination (VE))
Given a CNF formula $F$, variable elimination (or DP resolution) removes a variable $x$ by replacing $F_x$ and $F_{\bar{x}}$ by $F_x \otimes x F_{\bar{x}}$

Example of clause distribution

\[
\begin{array}{c|ccc}
 & (x \lor c) & (x \lor \bar{d}) & (x \lor \bar{a} \lor \bar{b}) \\
F_x & (a \lor c) & (a \lor d) & (a \lor \bar{a} \lor \bar{b}) \\
F_{\bar{x}} & (b \lor c) & (b \lor d) & (b \lor \bar{a} \lor \bar{b}) \\
& (c \lor \bar{e} \lor f) & (d \lor \bar{e} \lor f) & (\bar{a} \lor \bar{b} \lor \bar{e} \lor f) \\
\end{array}
\]
Example VE by clause distribution

Definition (Variable elimination (VE))
Given a CNF formula $F$, *variable elimination* (or DP resolution) removes a variable $x$ by replacing $F_x$ and $F_{\overline{x}}$ by $F_x \otimes_x F_{\overline{x}}$

Example of clause distribution

<table>
<thead>
<tr>
<th></th>
<th>$F_x$</th>
<th>$F_{\overline{x}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$(x \lor c)$</td>
<td>$(x \lor \overline{d})$</td>
</tr>
<tr>
<td>$(\overline{x} \lor a)$</td>
<td>$(a \lor c)$</td>
<td>$(a \lor d)$</td>
</tr>
<tr>
<td>$(\overline{x} \lor b)$</td>
<td>$(b \lor c)$</td>
<td>$(b \lor d)$</td>
</tr>
<tr>
<td>$(\overline{x} \lor \overline{e} \lor f)$</td>
<td>$(c \lor \overline{e} \lor f)$</td>
<td>$(d \lor \overline{e} \lor f)$</td>
</tr>
</tbody>
</table>
**Example VE by clause distribution**

**Definition (Variable elimination (VE))**
Given a CNF formula $F$, *variable elimination* (or DP resolution) removes a variable $x$ by replacing $F_x$ and $F_{\overline{x}}$ by $F_x \otimes x F_{\overline{x}}$

**Example of clause distribution**

<table>
<thead>
<tr>
<th>$F_x$</th>
<th>$F_{\overline{x}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(x \lor c)$</td>
<td>$(\overline{x} \lor a)$</td>
</tr>
<tr>
<td>$(x \lor \overline{d})$</td>
<td>$(\overline{x} \lor b)$</td>
</tr>
<tr>
<td>$(x \lor \overline{a} \lor \overline{b})$</td>
<td>$(\overline{x} \lor \overline{e} \lor f)$</td>
</tr>
</tbody>
</table>

In the example: $|F_x \otimes F_{\overline{x}}| > |F_x| + |F_{\overline{x}}|$

Exponential growth of clauses in general
VE by substitution

General idea
Detect gates (or definitions) \( x = \text{GATE}(a_1, \ldots, a_n) \) in the formula and use them to reduce the number of added clauses
VE by substitution

General idea
Detect gates (or definitions) \( x = \text{GATE}(a_1, \ldots, a_n) \) 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>((\bar{x} \lor a_1), \ldots,(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>
VE by substitution

General idea
Detect gates (or definitions) $x = \text{GATE}(a_1, \ldots, a_n)$ 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>$(\bar{x} \lor a_1), \ldots, (\bar{x} \lor a_n)$</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 $R_x = F_x \setminus G_x; R_{\bar{x}} = F_{\bar{x}} \setminus G_{\bar{x}}$. Replace $F_x \land F_{\bar{x}}$ by $G_x \otimes_x R_{\bar{x}} \land G_{\bar{x}} \otimes_x R_x$. Always less than $F_x \otimes_x F_{\bar{x}}$!
VE by substitution

Example of gate extraction: \( x = \text{AND}(a, b) \)

\[
F_x = (x \lor c) \land (x \lor \bar{d}) \land (x \lor \bar{a} \lor \bar{b}) \\
F_{\bar{x}} = (\bar{x} \lor a) \land (\bar{x} \lor b) \land (\bar{x} \lor \bar{e} \lor f)
\]
VE by substitution

Example of gate extraction: \( x = \text{AND}(a, b) \)

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

Example of substitution

<table>
<thead>
<tr>
<th>( G_{\bar{x}} )</th>
<th>( R_x )</th>
<th>( G_x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (\bar{x} \lor a) )</td>
<td>( (x \lor c) )</td>
<td>( (x \lor \bar{d}) )</td>
</tr>
<tr>
<td>( (\bar{x} \lor b) )</td>
<td>( (a \lor c) )</td>
<td>( (a \lor d) )</td>
</tr>
<tr>
<td>( (\bar{x} \lor \bar{e} \lor f) )</td>
<td>( (b \lor c) )</td>
<td>( (b \lor d) )</td>
</tr>
<tr>
<td>( (\bar{a} \lor \bar{b} \lor \bar{e} \lor f) )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
VE by substitution

Example of gate extraction: \( x = \text{AND}(a, b) \)

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

Example of substitution

<table>
<thead>
<tr>
<th>( G_{\bar{x}} )</th>
<th>( R_x )</th>
<th>( G_x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(( \bar{x} \lor a ))</td>
<td>(x \lor c)</td>
<td>(x \lor \bar{d})</td>
</tr>
<tr>
<td>(( \bar{x} \lor b ))</td>
<td>(a \lor c)</td>
<td>(a \lor d)</td>
</tr>
<tr>
<td>(( \bar{x} \lor \bar{e} \lor f ))</td>
<td>(b \lor c)</td>
<td>(b \lor d)</td>
</tr>
</tbody>
</table>

using substitution: \(|F_x \otimes F_{\bar{x}}| < |F_x| + |F_{\bar{x}}|\)
Variable Elimination

► How to reconstruct the model?

► Given $F$, we picked literal $x$, removed $F_x$ and $F_{\bar{x}}$, and added $F_x \otimes F_{\bar{x}}$

► A model $J$ does not contain a value for $x$.

► How can it work?
Bounded Variable Addition
Bounded Variable Addition

Main Idea
Given a CNF formula $F$, can we construct a (semi)logically equivalent $F'$ by introducing a new variable $x \notin \text{VAR}(F)$ such that $|F'| < |F|$?
Bounded Variable Addition

Main Idea
Given a CNF formula $F$, can we construct a (semi)logically equivalent $F'$ by introducing a new variable $x \notin \text{VAR}(F)$ such that $|F'| < |F|$?

Reverse of Variable Elimination
For example, replace the clauses

$$
\begin{align*}
(a \lor c) & \quad (a \lor d) \\
(b \lor c) & \quad (b \lor d) \\
(c \lor \overline{e} \lor f) & \quad (d \lor \overline{e} \lor f) \quad (\overline{a} \lor \overline{b} \lor \overline{e} \lor f)
\end{align*}
$$

by

$$
\begin{align*}
(\overline{x} \lor a) & \quad (\overline{x} \lor b) \quad (\overline{x} \lor \overline{e} \lor f) \\
(x \lor c) & \quad (x \lor d) \quad (x \lor \overline{a} \lor \overline{b})
\end{align*}
$$
Bounded Variable Addition

Main Idea
Given a CNF formula $F$, can we construct a (semi)logically equivalent $F'$ by introducing a new variable $x \notin \text{VAR}(F)$ such that $|F'| < |F|$?

Reverse of Variable Elimination
For example, replace the clauses

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

by

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

Challenge: how to find suitable patterns for replacement?
Factoring Out Subclauses

Example

Replace

\((a \lor b \lor c \lor d) (a \lor b \lor c \lor e) (a \lor b \lor c \lor f)\)

by

\((x \lor d) (x \lor e) (x \lor f) (\bar{x} \lor a \lor b \lor c)\)

adds 1 variable and 1 clause  reduces number of literals by 2

Not compatible with VE, 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

\[(\bar{x} \lor a) (\bar{x} \lor b) (\bar{x} \lor c)\]
\[(x \lor d) (x \lor e)\]

*adds 1 variable*  
*removes 1 clause*
Bounded Variable Addition

Possible Patterns

\[(X_1 \vee L_1) \quad \ldots \quad (X_1 \vee L_k) \quad \ldots \quad (X_n \vee L_1) \quad \ldots \quad (X_n \vee L_k)\]

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

replaced by

\[\bigwedge_{i=1}^{n} (y \vee X_i) \quad \bigwedge_{j=1}^{k} (\bar{y} \vee L_j)\]

- Every \(k\) clauses share sets of literals \(L_j\)
- There are \(n\) sets of literals \(X_i\) that appear in clauses with \(L_j\)
Bounded Variable Addition

Possible Patterns

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

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

replaced by \[
\bigwedge_{i=1}^{n} (y \lor X_i) \land \bigwedge_{j=1}^{k} (\bar{y} \lor L_j)
\]

- Every \( k \) clauses share sets of literals \( L_j \)
- There are \( n \) sets of literals \( X_i \) that appear in clauses with \( L_j \)
- Reduction: \( nk - n - k \) clauses are removed by replacement
Bounded Variable Addition

Possible Patterns

\[(X_1 \lor L_1) \; \ldots \; (X_1 \lor L_k)\]
\[
\vdots
\]
\[(X_n \lor L_1) \; \ldots \; (X_n \lor L_k)\]

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

replaced by

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

- Every \(k\) clauses share sets of literals \(L_j\)
- There are \(n\) sets of literals \(X_i\) that appear in clauses with \(L_j\)
- Reduction: \(nk - n - k\) clauses are removed by replacement
Bounded Variable Addition on AtMostOneZero (1)

Example encoding of AtMostOneZero ($x_1, x_2, \ldots, x_n$)

\[
\begin{align*}
(x_1 \lor x_2) &\land (x_9 \lor x_{10}) \land (x_8 \lor x_{10}) \land (x_7 \lor x_{10}) \land (x_6 \lor x_{10}) \\
(x_1 \lor x_3) &\land (x_2 \lor x_3) \land (x_8 \lor x_9) \land (x_7 \lor x_9) \land (x_6 \lor x_9) \\
(x_1 \lor x_4) &\land (x_2 \lor x_4) \land (x_3 \lor x_4) \land (x_7 \lor x_8) \land (x_6 \lor x_8) \\
(x_1 \lor x_5) &\land (x_2 \lor x_5) \land (x_3 \lor x_5) \land (x_4 \lor x_5) \land (x_6 \lor x_7) \\
(x_1 \lor x_6) &\land (x_2 \lor x_6) \land (x_3 \lor x_6) \land (x_4 \lor x_6) \land (x_5 \lor x_6) \\
(x_1 \lor x_7) &\land (x_2 \lor x_7) \land (x_3 \lor x_7) \land (x_4 \lor x_7) \land (x_5 \lor x_7) \\
(x_1 \lor x_8) &\land (x_2 \lor x_8) \land (x_3 \lor x_8) \land (x_4 \lor x_8) \land (x_5 \lor x_8) \\
(x_1 \lor x_9) &\land (x_2 \lor x_9) \land (x_3 \lor x_9) \land (x_4 \lor x_9) \land (x_5 \lor x_9) \\
(x_1 \lor x_{10}) &\land (x_2 \lor x_{10}) \land (x_3 \lor x_{10}) \land (x_4 \lor x_{10}) \land (x_5 \lor x_{10})
\end{align*}
\]
Bounded Variable Addition on AtMostOneZero (1)

Example encoding of AtMostOneZero ($x_1, x_2, \ldots, x_n$)

$$(x_1 \lor x_2) \land (x_9 \lor x_{10}) \land (x_8 \lor x_{10}) \land (x_7 \lor x_{10}) \land (x_6 \lor x_{10}) \land$$

$$(x_1 \lor x_3) \land (x_2 \lor x_3) \land (x_8 \lor x_9) \land (x_7 \lor x_9) \land (x_6 \lor x_9) \land$$

$$(x_1 \lor x_4) \land (x_2 \lor x_4) \land (x_3 \lor x_4) \land (x_7 \lor x_8) \land (x_6 \lor x_8) \land$$

$$(x_1 \lor x_5) \land (x_2 \lor x_5) \land (x_3 \lor x_5) \land (x_4 \lor x_5) \land (x_6 \lor x_7) \land$$

$$(x_1 \lor x_6) \land (x_2 \lor x_6) \land (x_3 \lor x_6) \land (x_4 \lor x_6) \land (x_5 \lor x_6) \land$$

$$(x_1 \lor x_7) \land (x_2 \lor x_7) \land (x_3 \lor x_7) \land (x_4 \lor x_7) \land (x_5 \lor x_7) \land$$

$$(x_1 \lor x_8) \land (x_2 \lor x_8) \land (x_3 \lor x_8) \land (x_4 \lor x_8) \land (x_5 \lor x_8) \land$$

$$(x_1 \lor x_9) \land (x_2 \lor x_9) \land (x_3 \lor x_9) \land (x_4 \lor x_9) \land (x_5 \lor x_9) \land$$

$$(x_1 \lor x_{10}) \land (x_2 \lor x_{10}) \land (x_3 \lor x_{10}) \land (x_4 \lor x_{10}) \land (x_5 \lor x_{10}) \land$$

Replace $(x_i \lor x_j)$ with $i \in \{1..5\}, j \in \{6..10\}$ by $(x_i \lor y), (x_j \lor \bar{y})$
Bounded Variable Addition on AtMostOneZero (2)

Example encoding of AtMostOneZero \((x_1, x_2, \ldots, x_n)\)

\[
(x_1 \lor x_2) \land (x_9 \lor x_{10}) \land (x_8 \lor x_{10}) \land (x_7 \lor x_{10}) \land (x_6 \lor x_{10}) \land \\
(x_1 \lor x_3) \land (x_2 \lor x_3) \land (x_8 \lor x_9) \land (x_7 \lor x_9) \land (x_6 \lor x_9) \land \\
(x_1 \lor x_4) \land (x_2 \lor x_4) \land (x_3 \lor x_4) \land (x_7 \lor x_8) \land (x_6 \lor x_8) \land \\
(x_1 \lor x_5) \land (x_2 \lor x_5) \land (x_3 \lor x_5) \land (x_4 \lor x_5) \land (x_6 \lor x_7) \land \\
(x_1 \lor y) \land (x_2 \lor y) \land (x_3 \lor y) \land (x_4 \lor y) \land (x_5 \lor y) \land \\
(x_6 \lor \overline{y}) \land (x_7 \lor \overline{y}) \land (x_8 \lor \overline{y}) \land (x_9 \lor \overline{y}) \land (x_{10} \lor \overline{y})
\]
Bounded Variable Addition on AtMostOneZero (2)

Example encoding of AtMostOneZero \((x_1, x_2, \ldots, x_n)\)

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

Replace matched pattern

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

$$(x_1 \lor x_2) \land (x_9 \lor x_{10}) \land (x_8 \lor x_{10}) \land (x_7 \lor x_{10}) \land (x_6 \lor x_{10}) \land$$
$$\ldots$$

$$(x_1 \lor x_3) \land (x_2 \lor x_3) \land (x_8 \lor x_9) \land (x_7 \lor x_9) \land (x_6 \lor x_9) \land$$
$$\ldots$$

$$(x_1 \lor \overline{z}) \land (x_2 \lor \overline{z}) \land (x_3 \lor \overline{z}) \land (x_7 \lor x_8) \land (x_6 \lor x_8) \land$$
$$\ldots$$

$$(x_4 \lor \overline{\bar{z}}) \land (x_5 \lor \overline{\bar{z}}) \land (y \lor \overline{\bar{z}}) \land (x_4 \lor x_5) \land (x_6 \lor x_7) \land$$
$$\ldots$$

$$(x_4 \lor y) \land (x_5 \lor y) \land (x_6 \lor \overline{\bar{y}}) \land (x_7 \lor \overline{\bar{y}}) \land (x_8 \lor \overline{\bar{y}})$$

$$(x_9 \lor \overline{\bar{y}}) \land (x_{10} \lor \overline{\bar{y}})$$
Bounded Variable Addition on AtMostOneZero (3)

Example encoding of AtMostOneZero \((x_1, x_2, \ldots, x_n)\)

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

Replace matched pattern

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

► How to reconstruct the model?
Blocked Clause Elimination
Blocked Clauses

Definition (Blocking literal)
A literal $l$ in a clause $C$ of a CNF $F$ blocks $C$ w.r.t. $F$ if for every clause $D \in F$, the resolvent $(C \setminus \{l\}) \cup (D \setminus \{\overline{l}\})$ obtained from resolving $C$ and $D$ on $l$ is a tautology.

With respect to a fixed CNF and its clauses we have:

Definition (Blocked clause)
A clause is blocked if it contains a literal that blocks it.

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

Proposition
Removal of an arbitrary blocked clause preserves satisfiability.
Blocked Clauses

Definition (Blocking literal)
A literal $l$ in a clause $C$ of a CNF $F$ blocks $C$ w.r.t. $F$ if for every clause $D \in F$, the resolvent $(C \setminus \{l\}) \cup (D \setminus \{\overline{l}\})$ obtained from resolving $C$ and $D$ on $l$ is a tautology.

With respect to a fixed CNF and its clauses we have:

Definition (Blocked clause)
A clause is blocked if it contains a literal that blocks it.

Example
Consider the formula $(a \lor b) \land (a \lor \overline{b} \lor \overline{c}) \land (\overline{a} \lor c)$. 
First clause is not blocked. 
Second clause is blocked by both $a$ and $\overline{c}$. 
Third clause is blocked by $c$
Blocked Clauses

Definition (Blocking literal)
A literal \( I \) in a clause \( C \) of a CNF \( F \) blocks \( C \) w.r.t. \( F \) if for every clause \( D \in F \), the resolvent \((C \setminus \{I\}) \cup (D \setminus \{\overline{I}\})\) obtained from resolving \( C \) and \( D \) on \( I \) is a tautology.

With respect to a fixed CNF and its clauses we have:

Definition (Blocked clause)
A clause is blocked if it contains a literal that blocks it.

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

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

Definition (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!
Blocked Clause Elimination (BCE)

Definition (BCE)
While there is a blocked clause \( C \) in a CNF \( F \), remove \( C \) from \( F \).

Example
Consider \((a \lor b) \land (a \lor \overline{b} \lor \overline{c}) \land (\overline{a} \lor c)\).
After removing either \((a \lor \overline{b} \lor \overline{c})\) or \((\overline{a} \lor c)\), the clause \((a \lor b)\) becomes blocked (no clause with either \(\overline{b}\) or \(\overline{a}\)).
An extreme case in which BCE removes all clauses!

Proposition
BCE is confluent, i.e., has a unique fixpoint

- Blocked clauses stay blocked w.r.t. removal
BCE very effective on circuits

- BCE converts the Tseitin encoding to Plaisted Greenbaum encoding
  - Only one implication is needed in the translation

- BCE simulates Pure literal elimination
  - There are no resolvents

- BCE simulates Cone of influence
  - The used variable appears only as (unused) gate output
Blocked Clause Elimination

- How to reconstruct the model?

- Given $F$, we picked clause $C$ with blocking literal $x$
  
- $C$ was blocked with respect to $F_{\bar{x}}$

- 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 graph
  ▷ 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
Research topics:
- encode problems into CNF
- simplify the problem
- and search for a solution or prove there does not exist one
Solving a Problem with SAT

Problem

ENCODER

preprocess

PREPROCESSOR

search

inprocess

SAT SOLVER

Research topics:
- encode problems into CNF
- simplify the problem
- and search for a solution or prove there does not exist one
- simplification during search
Solving a Problem with SAT

Problem

ENCODER

re-encode

preprocess

PREPROCESSOR

inprocess

search

SAT SOLVER

▸ 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