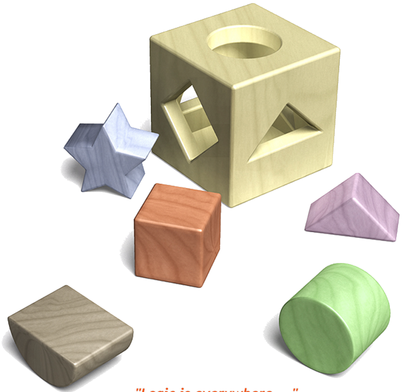
SAT Solving – Simplification

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

- **Types of Redundancy**
- **Simpification Algorithms**



"Logic is everywhere ..."

Simplification – Warm Up

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

Simplification – Warm Up

- Given a formiula F, when preserves removing a clause $C \in F$ equivalence?
- How is the above check performed?

Simplification – Warm Up

- **Given a formiula 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 formiula 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?

Simplification – Warm Up

- **Which of the following hold:**

Simplification – Warm Up

- **Which of the following hold:**

- $F \wedge x \equiv F|_x$

Simplification – Warm Up

- **Which of the following hold:**
- $F \land x \equiv F|_x$
- $F \land x \equiv_{SAT} F|_x$

Simplification – Warm Up

- **Which of the following hold:**
- $F \wedge x \equiv F|_x$
- $F \wedge x \equiv_{SAT} F|_x$
- $F \wedge x \models F|_x$

Simplification – Warm Up

- **Which of the following hold:**
- $F \wedge x \equiv F|_x$
- $F \wedge x \equiv_{SAT} F|_x$
- $F \wedge x \models F|_x$
- **Let C and D be clauses with $D \subset C : F \wedge D \models F \wedge C$**

Simplification – Warm Up

- **Which of the following hold:**
- $F \wedge x \equiv F|_x$
- $F \wedge x \equiv_{SAT} F|_x$
- $F \wedge x \models F|_x$
- **Let C and D be clauses with $D \subset C$: $F \wedge D \models F \wedge C$**
- **Let $D \subset C$: $F \wedge C \models F \wedge D$**

Simplification – Warm Up

- **How many (relevant partial) models has the formula $F = (a \vee \neg b) \wedge (\neg a \vee b)$?**

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 $F = (a \vee \neg b) \wedge (\neg a \vee b)$?**
- **Enumerate the models!**
- **How many models has the formula $F = (a \vee \neg a) \wedge (\neg a \vee a)$?**

Simplification – Warm Up

- **How many (relevant partial) models has the formula $F = (a \vee \neg b) \wedge (\neg a \vee b)$?**
- **Enumerate the models!**
- **How many models has the formula $F = (a \vee \neg a) \wedge (\neg a \vee a)$?**
- **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 $\boldsymbol{C}$ is a tautology iff it contains a complementary pair of literals.

Example

The clause $(\boldsymbol{a} \vee \boldsymbol{b} \vee \bar{\boldsymbol{b}})$ *is a tautology.*

Definition (Subsumption)

Clause $\boldsymbol{C}$ subsumes clause $\boldsymbol{D}$ iff $\boldsymbol{C} \subseteq \boldsymbol{D}$.

Example

The clause $(\boldsymbol{a} \vee \boldsymbol{b})$ *subsumes clause* $(\boldsymbol{a} \vee \boldsymbol{b} \vee \bar{\boldsymbol{c}})$*.*

Self-Subsuming Resolution

Self-Subsuming Resolution

$$\frac{C \vee l \qquad D \vee \bar{l}}{D} \quad C \subseteq D \qquad\qquad \frac{(a \vee b \vee l) \qquad (a \vee b \vee c \vee \bar{l})}{(a \vee b \vee c)}$$

resolvent D subsumes $D \vee \bar{l}$

Self-Subsuming Resolution

Self-Subsuming Resolution

$$\frac{C \vee l \qquad D \vee \bar{l}}{D} \quad C \subseteq D \qquad\qquad \frac{(a \vee b \vee l) \qquad (a \vee b \vee c \vee \bar{l})}{(a \vee b \vee c)}$$

resolvent D subsumes $D \vee \bar{l}$

Example

Assume a CNF contains both antecedents
$\ldots (a \vee b \vee l)(a \vee b \vee c \vee \bar{l}) \ldots$

If D is added, then $D \vee \bar{l}$ can be removed

which in essence removes $\bar{l}$ from $D \vee \bar{l}$
$\ldots (a \vee b \vee l)(a \vee b \vee c) \ldots$

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

Self-Subsuming Example

Self-Subsuming Resolution

$$\frac{C \vee l \qquad D \vee \bar{l}}{D} \quad C \subseteq D \qquad\qquad \frac{(a \vee b \vee l) \qquad (a \vee b \vee c \vee \bar{l})}{(a \vee b \vee c)}$$

resolvent D subsumes $D \vee \bar{l}$

Example: Remove literals using self-subsumption

$$\begin{array}{lcl}
(a \vee b \vee c) & \wedge & (\bar{a} \vee b \vee c) \wedge \\
(\bar{a} \vee b \vee \bar{c}) & \wedge & (a \vee \bar{b} \vee c) \wedge \\
(\bar{a} \vee \bar{b} \vee d) & \wedge & (\bar{a} \vee \bar{b} \vee \bar{d}) \wedge \\
(a \vee \bar{c} \vee d) & \wedge & (a \vee \bar{c} \vee \bar{d})
\end{array}$$

Self-Subsuming Example

Self-Subsuming Resolution

$$\frac{C \vee l \qquad D \vee \bar{l}}{D} \quad C \subseteq D \qquad\qquad \frac{(a \vee b \vee l) \qquad (a \vee b \vee c \vee \bar{l})}{(a \vee b \vee c)}$$

resolvent D subsumes $D \vee \bar{l}$

Example: Remove literals using self-subsumption

$$\begin{array}{l}
(\phantom{\bar{a} \vee{}} b \vee c) \wedge (\bar{a} \vee b \vee c) \wedge \\
(\bar{a} \vee b \vee \bar{c}) \wedge (a \vee \bar{b} \vee c) \wedge \\
(\bar{a} \vee \bar{b} \vee d) \wedge (\bar{a} \vee \bar{b} \vee \bar{d}) \wedge \\
(a \vee \bar{c} \vee d) \wedge (a \vee \bar{c} \vee \bar{d})
\end{array}$$

Self-Subsuming Example

Self-Subsuming Resolution

$$\frac{C \vee l \qquad D \vee \bar{l}}{D} \quad C \subseteq D \qquad\qquad \frac{(a \vee b \vee l) \qquad (a \vee b \vee c \vee \bar{l})}{(a \vee b \vee c)}$$

resolvent D subsumes $D \vee \bar{l}$

Example: Remove literals using self-subsumption

$$\begin{array}{llll}
(\quad b \vee c) & \wedge & (\bar{a} \vee b \vee c) & \wedge \\
(\bar{a} \vee b \quad) & \wedge & (a \vee \bar{b} \vee c) & \wedge \\
(\bar{a} \vee \bar{b} \vee d) & \wedge & (\bar{a} \vee \bar{b} \vee \bar{d}) & \wedge \\
(a \vee \bar{c} \vee d) & \wedge & (a \vee \bar{c} \vee \bar{d}) &
\end{array}$$

Self-Subsuming Example

Self-Subsuming Resolution

$$\frac{C \vee l \qquad D \vee \bar{l}}{D} \quad C \subseteq D \qquad\qquad \frac{(a \vee b \vee l) \qquad (a \vee b \vee c \vee \bar{l})}{(a \vee b \vee c)}$$

resolvent D subsumes $D \vee \bar{l}$

Example: Remove literals using self-subsumption

$$\begin{array}{llllll}
(& b \vee c) & \wedge & (\bar{a} \vee b \vee c) & \wedge \\
(\bar{a} \vee b &) & \wedge & (a \vee \quad c) & \wedge \\
(\bar{a} \vee \bar{b} \vee d) & & \wedge & (\bar{a} \vee \bar{b} \vee \bar{d}) & \wedge \\
(a \vee \bar{c} \vee d) & & \wedge & (a \vee \bar{c} \vee \bar{d}) &
\end{array}$$

Self-Subsuming Example

Self-Subsuming Resolution

$$\frac{C \vee l \qquad D \vee \bar{l}}{D} \quad C \subseteq D \qquad\qquad \frac{(a \vee b \vee l) \qquad (a \vee b \vee c \vee \bar{l})}{(a \vee b \vee c)}$$

resolvent D subsumes $D \vee \bar{l}$

Example: Remove literals using self-subsumption

$$\begin{array}{lll}
(\phantom{\bar{a} \vee{}} b \vee c) & \wedge & (\bar{a} \vee b \vee c) \wedge \\
(\bar{a} \vee b\phantom{{}\vee c}) & \wedge & (a \vee \phantom{b \vee{}} c) \wedge \\
(\bar{a} \vee \bar{b}\phantom{{}\vee c}) & \wedge & (\bar{a} \vee \bar{b} \vee \bar{d}) \wedge \\
(a \vee \bar{c} \vee d) & \wedge & (a \vee \bar{c} \vee \bar{d})
\end{array}$$

Self-Subsuming Example

Self-Subsuming Resolution

$$\frac{C \vee l \qquad D \vee \bar{l}}{D} \quad C \subseteq D \qquad\qquad \frac{(a \vee b \vee l) \qquad (a \vee b \vee c \vee \bar{l})}{(a \vee b \vee c)}$$

resolvent D subsumes $D \vee \bar{l}$

Example: Remove literals using self-subsumption

$$\begin{array}{llll}
(\phantom{\bar{a} \vee{}} b \vee c\,) & \wedge & (\,\bar{a} \vee b \vee c\,) & \wedge \\
(\bar{a} \vee b \phantom{{}\vee c}\,) & \wedge & (\,a \vee \phantom{b \vee{}} c\,) & \wedge \\
(\bar{a} \vee \bar{b} \phantom{{}\vee c}\,) & \wedge & (\,\bar{a} \vee \bar{b} \vee \bar{d}\,) & \wedge \\
(a \vee \bar{c} \phantom{{}\vee c}\,) & \wedge & (\,a \vee \bar{c} \vee \bar{d}\,) &
\end{array}$$

Self-Subsuming Example

Self-Subsuming Resolution

$$\frac{C \vee l \qquad D \vee \bar{l}}{D} \quad C \subseteq D \qquad\qquad \frac{(a \vee b \vee l) \qquad (a \vee b \vee c \vee \bar{l})}{(a \vee b \vee c)}$$

resolvent D subsumes $D \vee \bar{l}$

Example: Remove literals using self-subsumption

$$\begin{array}{lllll}
(\quad b \vee c) & \wedge & (\bar{a} \vee b \vee c) & \wedge \\
(\bar{a} \quad) & \wedge & (a \vee \quad c) & \wedge \\
(\bar{a} \vee \bar{b} \quad) & \wedge & (\bar{a} \vee \bar{b} \vee \bar{d}) & \wedge \\
(a \vee \bar{c} \quad) & \wedge & (a \vee \bar{c} \vee \bar{d}) &
\end{array}$$

Self-Subsuming Example

Self-Subsuming Resolution

$$\frac{C \vee l \qquad D \vee \bar{l}}{D} \quad C \subseteq D \qquad\qquad \frac{(a \vee b \vee l) \qquad (a \vee b \vee c \vee \bar{l})}{(a \vee b \vee c)}$$

resolvent D subsumes $D \vee \bar{l}$

Example: Remove literals using self-subsumption

$$\begin{array}{llll}
(\phantom{\bar{a} \vee {}} b \vee c) & \wedge & (\bar{a} \vee b \vee c) & \wedge \\
(\bar{a}) & \wedge & (a) & \wedge \\
(\bar{a} \vee \bar{b}) & \wedge & (\bar{a} \vee \bar{b} \vee \bar{d}) & \wedge \\
(a \vee \bar{c}) & \wedge & (a \vee \bar{c} \vee \bar{d}) &
\end{array}$$

Self-Subsuming Example

Self-Subsuming Resolution

$$\frac{C \vee l \qquad D \vee \bar{l}}{D} \quad C \subseteq D \qquad\qquad \frac{(a \vee b \vee l) \qquad (a \vee b \vee c \vee \bar{l})}{(a \vee b \vee c)}$$

resolvent D subsumes $D \vee \bar{l}$

Example: Remove literals using self-subsumption

$$\begin{array}{l}
(\phantom{\bar{a} \vee{}} b \vee c) \wedge (\bar{a} \vee b \vee c) \wedge \\
(\phantom{\bar{a} \vee b \vee c}) \wedge (a) \wedge \\
(\bar{a} \vee \bar{b}) \wedge (\bar{a} \vee \bar{b} \vee \bar{d}) \wedge \\
(a \vee \bar{c}) \wedge (a \vee \bar{c} \vee \bar{d})
\end{array}$$

Probing

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

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

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

Probing

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

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

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

Probing

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

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

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

- **Exploit: $F \models ((\bar{l}_1 \wedge \dots \wedge \bar{l}_i) \rightarrow x)$, hence $F \equiv F \wedge (l_1 \vee \dots \vee l_i \vee x)$**

Probing

- Idea: use unit propagation do derive extra information

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

- Exploit: $F \models ((\bar{l}_1 \wedge \cdots \wedge \bar{l}_i) \rightarrow x)$, hence $F \equiv F \wedge (l_1 \vee \cdots \vee l_i \vee x)$

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

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

- Exploit: $F \models ((\bar{l}_1 \land \cdots \land \bar{l}_i) \rightarrow x)$, 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. $C := (l_1 \lor \cdots \lor l_i \lor l_j)$

Probing

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

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

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

- **Exploit:** $F \models ((\bar{l}_1 \land \dots \land \bar{l}_i) \to x)$, **hence** $F \equiv F \land (l_1 \lor \dots \lor l_i \lor x)$

- **Then, replace C with**

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

Probing

- **Failed Literal** test for some literal l
  - $F :: (l) \leadsto^*_{UNIT} F :: J$, 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
  - $F :: (l) \leadsto^*_{UNIT} F :: J_l$, J_l is the set of all implied literals of l
  - $F :: (\neg l) \leadsto^*_{UNIT} F :: J_{\neg l}$, $J_{\neg l}$ is the set of all implied literals of $\neg l$

Probing

- **Failed Literal** test for some literal l
  - $F :: (l) \leadsto^*_{UNIT} F :: J$, 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
  - $F :: (l) \leadsto^*_{UNIT} F :: J_l$, J_l is the set of all implied literals of l
  - $F :: (\neg l) \leadsto^*_{UNIT} F :: J_{\neg l}$, $J_{\neg l}$ 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
  - $F :: (l) \leadsto^*_{UNIT} F :: J$, 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
  - $F :: (l) \leadsto^*_{UNIT} F :: J_l$, J_l is the set of all implied literals of l
  - $F :: (\neg l) \leadsto^*_{UNIT} F :: J_{\neg l}$, $J_{\neg l}$ 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:** $F \rightsquigarrow_{bad} F' \rightsquigarrow_{bad} F'' \rightsquigarrow_{bad} F''' \dots$
- **Reconstruction processes this chain in the opposite direction**
- $\dots J''' \rightarrow J'' \rightarrow J' \rightarrow J$
- **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 $\overline{l_1}$ 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 $F \models a \leftrightarrow b \leftrightarrow c$.

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)
    - $F \models (a \rightarrow b) \wedge (b \rightarrow c) \wedge (c \rightarrow a)$, then $F \models a \leftrightarrow b \leftrightarrow c$.
  - By structural hashing
    - $F \models (x \leftrightarrow (a \wedge b)) \wedge (y \leftrightarrow (a \wedge b)$, 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)
    - $F \models (a \rightarrow b) \wedge (b \rightarrow c) \wedge (c \rightarrow a)$, then $F \models a \leftrightarrow b \leftrightarrow c$.
  - By structural hashing
    - $F \models (x \leftrightarrow (a \wedge b)) \wedge (y \leftrightarrow (a \wedge b)$, 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, *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 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

$$
\begin{array}{r|ccc}
 & \multicolumn{3}{c}{\overbrace{(x \lor c) \qquad (x \lor \bar{d}) \qquad (x \lor \bar{a} \lor \bar{b})}^{F_x}} \\
\hline
F_{\bar{x}} \left\{ \begin{array}{r} (\bar{x} \lor a) \\ (\bar{x} \lor b) \\ (\bar{x} \lor \bar{e} \lor f) \end{array} \right. & \begin{array}{c} (a \lor c) \\ (b \lor c) \\ (c \lor \bar{e} \lor f) \end{array} & \begin{array}{c} (a \lor d) \\ (b \lor d) \\ (d \lor \bar{e} \lor f) \end{array} & \begin{array}{c} (a \lor \bar{a} \lor \bar{b}) \\ (b \lor \bar{a} \lor \bar{b}) \\ (\bar{a} \lor \bar{b} \lor \bar{e} \lor f) \end{array}
\end{array}
$$

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 $F_{\bar{x}}$ by $F_x \otimes_x F_{\bar{x}}$

Example of clause distribution

		$(x \vee c)$	$(x \vee \bar{d})$	$(x \vee \bar{a} \vee \bar{b})$
$F_{\bar{x}}$	$(\bar{x} \vee a)$	$(a \vee c)$	$(a \vee d)$	$\cancel{(a \vee \bar{a} \vee \bar{b})}$
	$(\bar{x} \vee b)$	$(b \vee c)$	$(b \vee d)$	$\cancel{(b \vee \bar{a} \vee \bar{b})}$
	$(\bar{x} \vee \bar{e} \vee f)$	$(c \vee \bar{e} \vee f)$	$(d \vee \bar{e} \vee f)$	$(\bar{a} \vee \bar{b} \vee \bar{e} \vee f)$

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

	$\overbrace{(x \lor c)}^{F_x}$	$(x \lor \bar{d})$	$(x \lor \bar{a} \lor \bar{b})$
$F_{\bar{x}}$: $(\bar{x} \lor a)$	$(a \lor c)$	$(a \lor d)$	$\cancel{(a \lor \bar{a} \lor \bar{b})}$
$(\bar{x} \lor b)$	$(b \lor c)$	$(b \lor d)$	$\cancel{(b \lor \bar{a} \lor \bar{b})}$
$(\bar{x} \lor \bar{e} \lor f)$	$(c \lor \bar{e} \lor f)$	$(d \lor \bar{e} \lor f)$	$(\bar{a} \lor \bar{b} \lor \bar{e} \lor f)$

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

Exponential growth of clauses in general

VE by substitution

General idea

Detect gates (or definitions) $x = \mathrm{GATE}(a_1, \ldots, a_n)$ in the formula and use them to reduce the number of added clauses

VE by substitution

General idea

Detect gates (or definitions) $x = \mathrm{GATE}(a_1, \ldots, a_n)$ in the formula and use them to reduce the number of added clauses

Possible gates

gate	G_x	$G_{\bar{x}}$
$\mathrm{AND}(a_1, \ldots, a_n)$	$(x \lor \bar{a}_1 \lor \cdots \lor \bar{a}_n)$	$(\bar{x} \lor a_1), \ldots, (\bar{x} \lor a_n)$
$\mathrm{OR}(a_1, \ldots, a_n)$	$(x \lor \bar{a}_1), \ldots, (x \lor \bar{a}_n)$	$(\bar{x} \lor a_1 \lor \cdots \lor a_n)$
$\mathrm{ITE}(c, t, f)$	$(x \lor \bar{c} \lor \bar{t}), (x \lor c \lor \bar{f})$	$(\bar{x} \lor \bar{c} \lor t), (\bar{x} \lor c \lor f)$

VE by substitution

General idea

Detect gates (or definitions) $x = \mathrm{GATE}(a_1, \ldots, a_n)$ in the formula and use them to reduce the number of added clauses

Possible gates

gate	G_x	$G_{\bar{x}}$
$\mathrm{AND}(a_1, \ldots, a_n)$	$(x \vee \bar{a}_1 \vee \cdots \vee \bar{a}_n)$	$(\bar{x} \vee a_1), \ldots, (\bar{x} \vee a_n)$
$\mathrm{OR}(a_1, \ldots, a_n)$	$(x \vee \bar{a}_1), \ldots, (x \vee \bar{a}_n)$	$(\bar{x} \vee a_1 \vee \cdots \vee a_n)$
$\mathrm{ITE}(c, t, f)$	$(x \vee \bar{c} \vee \bar{t}), (x \vee c \vee \bar{f})$	$(\bar{x} \vee \bar{c} \vee t), (\bar{x} \vee c \vee f)$

Variable elimination by substitution

Let $R_x = F_x \setminus G_x$; $R_{\bar{x}} = F_{\bar{x}} \setminus G_{\bar{x}}$.

Replace $F_x \wedge F_{\bar{x}}$ by $G_x \otimes_x R_{\bar{x}} \wedge G_{\bar{x}} \otimes_x R_x$.

Always less than $F_x \otimes_x F_{\bar{x}}$!

VE by substitution

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

$$F_x = (x \vee c) \wedge (x \vee \bar{d}) \wedge (x \vee \bar{a} \vee \bar{b})$$
$$F_{\bar{x}} = (\bar{x} \vee a) \wedge (\bar{x} \vee b) \wedge (\bar{x} \vee \bar{e} \vee f)$$

VE by substitution

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

$$
\begin{aligned}
F_x &= (x \lor c) \land (x \lor \bar{d}) \land (x \lor \bar{a} \lor \bar{b}) \\
F_{\bar{x}} &= (\bar{x} \lor a) \land (\bar{x} \lor b) \land (\bar{x} \lor \bar{e} \lor f)
\end{aligned}
$$

Example of substitution

		R_x		G_x
		$(x \lor c)$	$(x \lor \bar{d})$	$(x \lor \bar{a} \lor \bar{b})$
$G_{\bar{x}}$	$(\bar{x} \lor a)$	$(a \lor c)$	$(a \lor d)$	
	$(\bar{x} \lor b)$	$(b \lor c)$	$(b \lor d)$	
$R_{\bar{x}}$	$(\bar{x} \lor \bar{e} \lor f)$			$(\bar{a} \lor \bar{b} \lor \bar{e} \lor f)$

VE by substitution

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

$$F_x = (x \vee c) \wedge (x \vee \bar{d}) \wedge (x \vee \bar{a} \vee \bar{b})$$
$$F_{\bar{x}} = (\bar{x} \vee a) \wedge (\bar{x} \vee b) \wedge (\bar{x} \vee \bar{e} \vee f)$$

Example of substitution

$$\begin{array}{ll|ccc}
 & & \multicolumn{2}{c}{\overbrace{\hspace{12em}}^{R_x}} & \overbrace{\hspace{8em}}^{G_x} \\
 & & (x \vee c) & (x \vee \bar{d}) & (x \vee \bar{a} \vee \bar{b}) \\
\hline
G_{\bar{x}} & (\bar{x} \vee a) & (a \vee c) & (a \vee d) & \\
 & (\bar{x} \vee b) & (b \vee c) & (b \vee d) & \\
R_{\bar{x}} & (\bar{x} \vee \bar{e} \vee f) & & & (\bar{a} \vee \bar{b} \vee \bar{e} \vee f)
\end{array}$$

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

Bounded Variable Addition

Main Idea

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

Reverse of Variable Elimination

For example, replace the clauses

$$\begin{array}{ccc} (a \lor c) & (a \lor d) & \\ (b \lor c) & (b \lor d) & \\ (c \lor \bar{e} \lor f) & (d \lor \bar{e} \lor f) & (\bar{a} \lor \bar{b} \lor \bar{e} \lor f) \end{array}$$

by

$$\begin{array}{ccc} (\bar{x} \lor a) & (\bar{x} \lor b) & (\bar{x} \lor \bar{e} \lor f) \\ (x \lor c) & (x \lor d) & (x \lor \bar{a} \lor \bar{b}) \end{array}$$

Bounded Variable Addition

Main Idea

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

Reverse of Variable Elimination

For example, replace the clauses

$$\begin{array}{ccc} (a \vee c) & (a \vee d) & \\ (b \vee c) & (b \vee d) & \\ (c \vee \bar{e} \vee f) & (d \vee \bar{e} \vee f) & (\bar{a} \vee \bar{b} \vee \bar{e} \vee f) \end{array}$$

by

$$\begin{array}{ccc} (\bar{x} \vee a) & (\bar{x} \vee b) & (\bar{x} \vee \bar{e} \vee f) \\ (x \vee c) & (x \vee d) & (x \vee \bar{a} \vee \bar{b}) \end{array}$$

Challenge: how to find suitable patterns for replacement?

Factoring Out Subclauses

Example

Replace

$$(a \vee b \vee c \vee d) \quad (a \vee b \vee c \vee e) \quad (a \vee b \vee c \vee f)$$

by

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

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

$$\begin{array}{cc} (a \lor d) & (a \lor e) \\ (b \lor d) & (b \lor e) \\ (c \lor d) & (c \lor e) \end{array}$$

by

$$\begin{array}{ccc} (\bar{x} \lor a) & (\bar{x} \lor b) & (\bar{x} \lor c) \\ (x \lor d) & (x \lor e) & \end{array}$$

adds 1 variable *removes* 1 clause

Bounded Variable Addition

Possible Patterns

$$\begin{array}{ccc} (X_1 \vee L_1) & \dots & (X_1 \vee L_k) \\ \vdots & & \vdots \\ (X_n \vee L_1) & \dots & (X_n \vee L_k) \end{array} \quad \equiv \quad \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{k} (X_i \vee L_j)$$

$$\textit{replaced by} \quad \bigwedge_{i=1}^{n} (y \vee X_i) \;\wedge\; \bigwedge_{j=1}^{k} (\bar{y} \vee L_j)$$

- **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

$$\begin{matrix} (X_1 \vee L_1) & \dots & (X_1 \vee L_k) \\ \vdots & & \vdots \\ (X_n \vee L_1) & \dots & (X_n \vee L_k) \end{matrix} \quad \equiv \quad \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{k} (X_i \vee L_j)$$

replaced by $\bigwedge_{i=1}^{n} (y \vee X_i) \;\wedge\; \bigwedge_{j=1}^{k} (\bar{y} \vee L_j)$

- **Every k clauses share sets of literals L_j**
- **There are n sets of literals X_i that appear in clauses with L_j**
- **Reduction: $nk - n - k$ clauses are removed by replacement**

Bounded Variable Addition

Possible Patterns

$$\begin{array}{ccc} (X_1 \vee L_1) & \dots & (X_1 \vee L_k) \\ \vdots & & \vdots \\ (X_n \vee L_1) & \dots & (X_n \vee L_k) \end{array} \quad \equiv \quad \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{k} (X_i \vee L_j)$$

replaced by $\bigwedge_{i=1}^{n} (y \vee X_i) \;\wedge\; \bigwedge_{j=1}^{k} (\bar{y} \vee L_j)$

- **Every k clauses share sets of literals L_j**
- **There are n sets of literals X_i that appear in clauses with L_j**
- **Reduction: $nk - n - k$ clauses are removed by replacement**

Bounded Variable Addition on AtMostOneZero (1)

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

$$
\begin{array}{l}
(x_1 \vee x_2) \wedge (x_9 \vee x_{10}) \wedge (x_8 \vee x_{10}) \wedge (x_7 \vee x_{10}) \wedge (x_6 \vee x_{10}) \wedge \\
(x_1 \vee x_3) \wedge (x_2 \vee x_3) \wedge (x_8 \vee x_9) \wedge (x_7 \vee x_9) \wedge (x_6 \vee x_9) \wedge \\
(x_1 \vee x_4) \wedge (x_2 \vee x_4) \wedge (x_3 \vee x_4) \wedge (x_7 \vee x_8) \wedge (x_6 \vee x_8) \wedge \\
(x_1 \vee x_5) \wedge (x_2 \vee x_5) \wedge (x_3 \vee x_5) \wedge (x_4 \vee x_5) \wedge (x_6 \vee x_7) \wedge \\
(x_1 \vee x_6) \wedge (x_2 \vee x_6) \wedge (x_3 \vee x_6) \wedge (x_4 \vee x_6) \wedge (x_5 \vee x_6) \wedge \\
(x_1 \vee x_7) \wedge (x_2 \vee x_7) \wedge (x_3 \vee x_7) \wedge (x_4 \vee x_7) \wedge (x_5 \vee x_7) \wedge \\
(x_1 \vee x_8) \wedge (x_2 \vee x_8) \wedge (x_3 \vee x_8) \wedge (x_4 \vee x_8) \wedge (x_5 \vee x_8) \wedge \\
(x_1 \vee x_9) \wedge (x_2 \vee x_9) \wedge (x_3 \vee x_9) \wedge (x_4 \vee x_9) \wedge (x_5 \vee x_9) \wedge \\
(x_1 \vee x_{10}) \wedge (x_2 \vee x_{10}) \wedge (x_3 \vee x_{10}) \wedge (x_4 \vee x_{10}) \wedge (x_5 \vee x_{10})
\end{array}
$$

Bounded Variable Addition on AtMostOneZero (1)

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

$$
\begin{array}{l}
(x_1 \vee x_2) \wedge (x_9 \vee x_{10}) \wedge (x_8 \vee x_{10}) \wedge (x_7 \vee x_{10}) \wedge (x_6 \vee x_{10}) \wedge \\
(x_1 \vee x_3) \wedge (x_2 \vee x_3) \wedge (x_8 \vee x_9) \wedge (x_7 \vee x_9) \wedge (x_6 \vee x_9) \wedge \\
(x_1 \vee x_4) \wedge (x_2 \vee x_4) \wedge (x_3 \vee x_4) \wedge (x_7 \vee x_8) \wedge (x_6 \vee x_8) \wedge \\
(x_1 \vee x_5) \wedge (x_2 \vee x_5) \wedge (x_3 \vee x_5) \wedge (x_4 \vee x_5) \wedge (x_6 \vee x_7) \wedge \\
(x_1 \vee x_6) \wedge (x_2 \vee x_6) \wedge (x_3 \vee x_6) \wedge (x_4 \vee x_6) \wedge (x_5 \vee x_6) \wedge \\
(x_1 \vee x_7) \wedge (x_2 \vee x_7) \wedge (x_3 \vee x_7) \wedge (x_4 \vee x_7) \wedge (x_5 \vee x_7) \wedge \\
(x_1 \vee x_8) \wedge (x_2 \vee x_8) \wedge (x_3 \vee x_8) \wedge (x_4 \vee x_8) \wedge (x_5 \vee x_8) \wedge \\
(x_1 \vee x_9) \wedge (x_2 \vee x_9) \wedge (x_3 \vee x_9) \wedge (x_4 \vee x_9) \wedge (x_5 \vee x_9) \wedge \\
(x_1 \vee x_{10}) \wedge (x_2 \vee x_{10}) \wedge (x_3 \vee x_{10}) \wedge (x_4 \vee x_{10}) \wedge (x_5 \vee x_{10})
\end{array}
$$

Replace $(x_i \vee x_j)$ with $i \in \{1..5\}, j \in \{6..10\}$ by $(x_i \vee y)$, $(x_j \vee \bar{y})$

Bounded Variable Addition on AtMostOneZero (2)

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

$$
\begin{array}{l}
(x_1 \vee x_2) \wedge (x_9 \vee x_{10}) \wedge (x_8 \vee x_{10}) \wedge (x_7 \vee x_{10}) \wedge (x_6 \vee x_{10}) \wedge \\
(x_1 \vee x_3) \wedge (x_2 \vee x_3) \wedge (x_8 \vee x_9) \wedge (x_7 \vee x_9) \wedge (x_6 \vee x_9) \wedge \\
(x_1 \vee x_4) \wedge (x_2 \vee x_4) \wedge (x_3 \vee x_4) \wedge (x_7 \vee x_8) \wedge (x_6 \vee x_8) \wedge \\
(x_1 \vee x_5) \wedge (x_2 \vee x_5) \wedge (x_3 \vee x_5) \wedge (x_4 \vee x_5) \wedge (x_6 \vee x_7) \wedge \\
(x_1 \vee y) \wedge (x_2 \vee y) \wedge (x_3 \vee y) \wedge (x_4 \vee y) \wedge (x_5 \vee y) \wedge \\
(x_6 \vee \bar{y}) \wedge (x_7 \vee \bar{y}) \wedge (x_8 \vee \bar{y}) \wedge (x_9 \vee \bar{y}) \wedge (x_{10} \vee \bar{y})
\end{array}
$$

Bounded Variable Addition on AtMostOneZero (2)

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

$$
\begin{aligned}
&(x_1 \vee x_2) \wedge (x_9 \vee x_{10}) \wedge (x_8 \vee x_{10}) \wedge (x_7 \vee x_{10}) \wedge (x_6 \vee x_{10}) \wedge \\
&(x_1 \vee x_3) \wedge (x_2 \vee x_3) \wedge (x_8 \vee x_9) \wedge (x_7 \vee x_9) \wedge (x_6 \vee x_9) \wedge \\
&(x_1 \vee x_4) \wedge (x_2 \vee x_4) \wedge (x_3 \vee x_4) \wedge (x_7 \vee x_8) \wedge (x_6 \vee x_8) \wedge \\
&(x_1 \vee x_5) \wedge (x_2 \vee x_5) \wedge (x_3 \vee x_5) \wedge (x_4 \vee x_5) \wedge (x_6 \vee x_7) \wedge \\
&(x_1 \vee y) \wedge (x_2 \vee y) \wedge (x_3 \vee y) \wedge (x_4 \vee y) \wedge (x_5 \vee y) \wedge \\
&(x_6 \vee \bar{y}) \wedge (x_7 \vee \bar{y}) \wedge (x_8 \vee \bar{y}) \wedge (x_9 \vee \bar{y}) \wedge (x_{10} \vee \bar{y})
\end{aligned}
$$

Replace matched pattern

$$
\begin{aligned}
&(x_1 \vee z) \wedge (x_2 \vee z) \wedge (x_3 \vee z) \wedge \\
&(x_4 \vee \bar{z}) \wedge (x_5 \vee \bar{z}) \wedge (y \vee \bar{z})
\end{aligned}
$$

Bounded Variable Addition on AtMostOneZero (3)

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

$$
\begin{array}{l}
(x_1 \vee x_2) \wedge (x_9 \vee x_{10}) \wedge (x_8 \vee x_{10}) \wedge (x_7 \vee x_{10}) \wedge (x_6 \vee x_{10}) \wedge \\
(x_1 \vee x_3) \wedge (x_2 \vee x_3) \wedge (x_8 \vee x_9) \wedge (x_7 \vee x_9) \wedge (x_6 \vee x_9) \wedge \\
(x_1 \vee z) \wedge (x_2 \vee z) \wedge (x_3 \vee z) \wedge (x_7 \vee x_8) \wedge (x_6 \vee x_8) \wedge \\
(x_4 \vee \bar{z}) \wedge (x_5 \vee \bar{z}) \wedge (y \vee \bar{z}) \wedge (x_4 \vee x_5) \wedge (x_6 \vee x_7) \wedge \\
(x_4 \vee y) \wedge (x_5 \vee y) \wedge (x_6 \vee \bar{y}) \wedge (x_7 \vee \bar{y}) \wedge (x_8 \vee \bar{y}) \\
(x_9 \vee \bar{y}) \wedge (x_{10} \vee \bar{y})
\end{array}
$$

Bounded Variable Addition on AtMostOneZero (3)

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

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

Replace matched pattern

$$(x_6 \lor w) \land (x_7 \lor w) \land (x_8 \lor w) \land$$
$$(x_9 \lor \bar{w}) \land (x_{10} \lor \bar{w}) \land (\bar{y} \lor \bar{w})$$

Bounded Variable Addition

- **How 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 $D \in F_{\bar{l}}$, the resolvent $(C \setminus \{l\}) \cup (D \setminus \{\bar{l}\})$ 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.

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 $D \in F_{\bar{l}}$, the resolvent $(C \setminus \{l\}) \cup (D \setminus \{\bar{l}\})$ 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 $(a \vee b) \wedge (a \vee \bar{b} \vee \bar{c}) \wedge (\bar{a} \vee c)$.
First clause is not blocked.
Second clause is blocked by both a *and* $\bar{c}$.
Third clause is blocked by c

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 $D \in F_{\bar{l}}$, the resolvent $(C \setminus \{l\}) \cup (D \setminus \{\bar{l}\})$ 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 $(a \vee b) \wedge (a \vee \bar{b} \vee \bar{c}) \wedge (\bar{a} \vee c)$.
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)

Definition (BCE)

While there is a blocked clause $\boldsymbol{C}$ in a CNF $\boldsymbol{F}$, remove $\boldsymbol{C}$ from $\boldsymbol{F}$.

Example

Consider $(\boldsymbol{a} \vee \boldsymbol{b}) \wedge (\boldsymbol{a} \vee \bar{\boldsymbol{b}} \vee \bar{\boldsymbol{c}}) \wedge (\bar{\boldsymbol{a}} \vee \boldsymbol{c})$.
After removing either $(\boldsymbol{a} \vee \bar{\boldsymbol{b}} \vee \bar{\boldsymbol{c}})$ *or* $(\bar{\boldsymbol{a}} \vee \boldsymbol{c})$, *the clause* $(\boldsymbol{a} \vee \boldsymbol{b})$ *becomes blocked (no clause with either* $\bar{\boldsymbol{b}}$ *or* $\bar{\boldsymbol{a}}$*).*

An extreme case in which BCE *removes all clauses!*

Blocked Clause Elimination (BCE)

Definition (BCE)

While there is a blocked clause $\boldsymbol{C}$ in a CNF $\boldsymbol{F}$, remove $\boldsymbol{C}$ from $\boldsymbol{F}$.

Example

Consider $(\boldsymbol{a} \vee \boldsymbol{b}) \wedge (\boldsymbol{a} \vee \bar{\boldsymbol{b}} \vee \bar{\boldsymbol{c}}) \wedge (\bar{\boldsymbol{a}} \vee \boldsymbol{c})$.
After removing either $(\boldsymbol{a} \vee \bar{\boldsymbol{b}} \vee \bar{\boldsymbol{c}})$ *or* $(\bar{\boldsymbol{a}} \vee \boldsymbol{c})$*, the clause* $(\boldsymbol{a} \vee \boldsymbol{b})$ *becomes blocked (no clause with either* $\bar{\boldsymbol{b}}$ *or* $\bar{\boldsymbol{a}}$*).*

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**

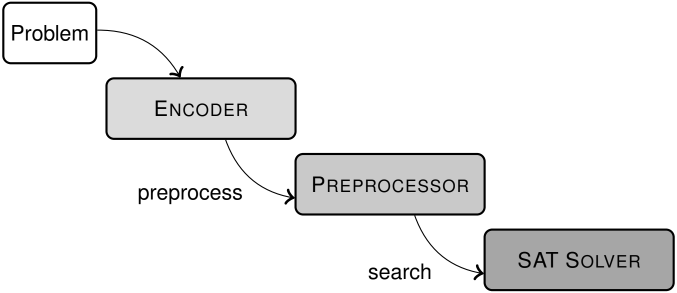
- **How can it work?**

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**

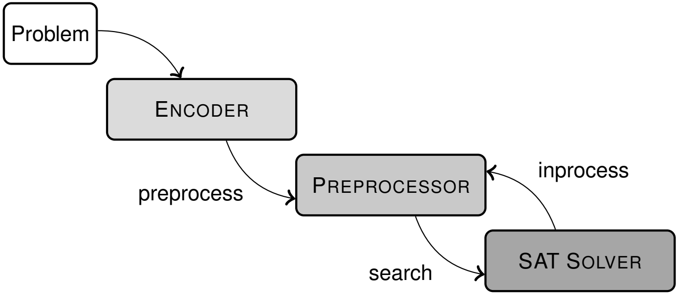
- **Not discussed here:**
  - **Adding redundant clauses**
  - **Minimizing redundant clauses**

Solving a Problem with SAT



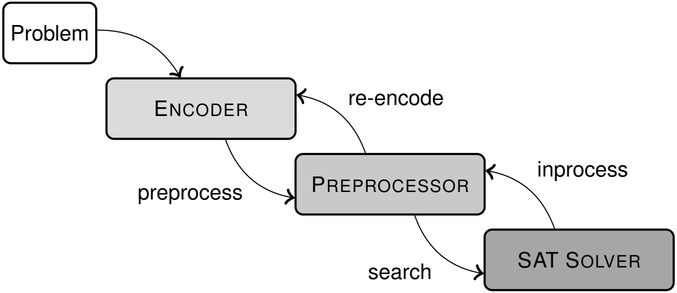
- **Research topics:**
  - **encode problems into CNF**
  - **simplify the problem**
  - **and search for a solution or prove there does not exist one**

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**

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**