Elimination Techniques
In Modern Propositional Logic Reasoning

Norbert Manthey
nmanthey@conp-solutions.com

December 7, 2017
Outline

- Satisfiability Testing
- Elimination in SAT
  - Solving Algorithms
  - Constraint Types
  - Model Reconstruction
  - Variable Addition
- Conclusion
Satisfiability Testing
Propositional Logic

- Variables: $v_1, v_2, \cdots \in \mathcal{V}$ of Boolean domain $\{\bot, \top\}$
  - often also seen as $\{0, 1\}$
- Connectives:
  - negation $\neg v_1$ (also written as $\overline{v_1}$)
  - disjunction $v_1 \lor v_2$
  - conjunction $v_1 \land v_2$
  - many more, can be defined over truth table
- Literals: $p, \neg q, x_1, \overline{x_2}, \ldots$ are variables, or negated variables
  - double negation is eliminated
- Function $\text{vars}(F)$ returns set of variables of formula $F$
- Function $\text{lits}(F)$ returns set of literals of formula $F$
Propositional Logic - Semantics

- Interpretation: function that maps variables to truth values
  - total: map all variables of the input language
  - partial: map variables of the input language
  - complete (wrt. formula): map all variables of the formula

- An interpretation $I$ satisfies a formula $F$, if the formula evaluates to $\top$ after mapping the variables to their truth values, i.e. $I \models F$. 

- Satisfiability Testing: Given a formula $F$, is it satisfiable?
  - Compute a model, an unsatisfiable subset or proof!
Propositional Logic - Semantics

- Interpretation: function that maps variables to truth values
  - total: map all variables of the input language
  - partial: map variables of the input language
  - complete (wrt. formula): map all variables of the formula

- An interpretation $I$ satisfies a formula $F$, if the formula evaluates to $\top$ after mapping the variables to their truth values, i.e. $I \models F$.

- A formula $F$ is satisfiable, if such an interpretation $I$ exists.

- Satisfiability Testing: Given a formula $F$, is it satisfiable?
  - Compute a model, an unsatisfiable subset or proof!
Propositional Logic - Conjunctive Normal Form (CNF)

- Proposition logic formulas can be complex
- Reasoners should be fast
- Pick reasonable subset

Clause: disjunction of literals

$$(x_1 \lor \cdots \lor x_k)$$

equal to a (multi)set of literals

\{x_1, \ldots, x_k\}

CNF Formula: conjunction of clauses

$$(C_1 \land \cdots \land C_n)$$

equal to a (multi)set of clauses

\{C_1, \ldots, C_k\}

Resolvent of clauses $C$ and $D$ with $x \in C$ and $x \in D$:

$$C \otimes D = (C \setminus x) \cup (D \setminus x)$$

Reduct $F$ wrt set of literals $x$, $F|_x$: map $x$ to $\top$, simplify

Subformula $F_x$ of $F$ wrt literal $x$:

$$F = \{\{x, y\}, \{x, y\}\}$$

$$F|_x = \{\{y\}\}$$

$$F_x = \{\{x, y\}\}$$
Propositional Logic - Conjunctive Normal Form (CNF)

- Proposition logic formulas can be complex
- Reasoners should be fast
- Pick reasonable subset

- Clause: disjunction of literals \((x_1 \lor \cdots \lor x_k)\)
  - equal to a (multi)set of literals \(\{x_1, \ldots, x_k\}\)
- CNF Formula: conjunction of clauses \((C_1 \land \cdots \land C_n)\)
  - equal to a (multi)set of clauses \(\{C_1, \ldots, C_k\}\)
- **Resolvent** of clauses \(C\) and \(D\) with \(x \in C\) and \(\overline{x} \in D\):
  - \(C \otimes D = (C \setminus x) \cup (D \setminus \overline{x})\)
Propositional Logic - Conjunctive Normal Form (CNF)

▶ Proposition logic formulas can be complex
▶ Reasoners should be fast
▶ Pick reasonable subset

▶ Clause: disjunction of literals \((x_1 \lor \cdots \lor x_k)\)
  ▶ equal to a (multi)set of literals \(\{x_1, \ldots, x_k\}\)
▶ CNF Formula: conjunction of clauses \((C_1 \land \cdots \land C_n)\)
  ▶ equal to a (multi)set of clauses \(\{C_1, \ldots, C_k\}\)
▶ Resolvent of clauses \(C\) and \(D\) with \(x \in C\) and \(\overline{x} \in D\):
  ▶ \(C \otimes D = (C \setminus x) \cup (D \setminus \overline{x})\)
▶ Reduct \(F\) wrt set of literals \(x\), \(F\mid_x\): map \(x\) to \(\top\), simplify
▶ Subformula \(F_x\) of \(F\) wrt literal \(x\): clauses with \(x\)
Propositional Logic - Conjunctive Normal Form (CNF)

- Proposition logic formulas can be complex
- Reasoners should be fast
- Pick reasonable subset

- Clause: disjunction of literals \((x_1 \lor \cdots \lor x_k)\)
  - equal to a (multi)set of literals \(\{x_1, \ldots, x_k\}\)
- CNF Formula: conjunction of clauses \((C_1 \land \cdots \land C_n)\)
  - equal to a (multi)set of clauses \(\{C_1, \ldots, C_k\}\)
- \textbf{Resolvent} of clauses \(C\) and \(D\) with \(x \in C\) and \(\bar{x} \in D\):
  - \(C \otimes D = (C \setminus x) \cup (D \setminus \bar{x})\)
- \textbf{Reduct} \(F\) wrt set of literals \(x\), \(F|_x\): map \(x\) to \(\top\), simplify
- \textbf{Subformula} \(F_x\) of \(F\) wrt literal \(x\): clauses with \(x\)

\[
F = \{\{x, y\}, \{\bar{x}, y\}\} \quad F|_x = \{\{y\}\} \quad F_x = \{\{x, y\}\}
\]
Propositional Logic - Formula Relations

► Given, formulas $F$ and $G$
► $F \models G$, if all (total) interpretations $I$ with $I \models F$ also satisfy $G$, $I \models G$

► Equivalence $F \equiv G$: $F \models G$ and $G \models F$
► Equi-Satisfiability $F \equiv_{SAT} G$: $F$ and $G$ are both satisfiable, or $F$ and $G$ are both unsatisfiable

► **Unsatisfiability-Preserving** $F \models_{UNSAT} G$: if $F \models G$ and $F \equiv_{SAT} G$
Propositional Logic - Formula Relations

- Given, formulas $F$ and $G$
- $F \models G$, if all (total) interpretations $I$ with $I \models F$ also satisfy $G$, $I \models G$

- Equivalence $F \equiv G$: $F \models G$ and $G \models F$
- Equi-Satisfiability $F \equiv_{SAT} G$: $F$ and $G$ are both satisfiable, or $F$ and $G$ are both unsatisfiable

- **Unsatisfiability-Preserving** $F \models_{UNSAT} G$: if $F \models G$ and $F \equiv_{SAT} G$

  
  \[
  \begin{align*}
  x \models (x \lor y) & \quad x \equiv_{SAT} y \\
  (x \land \overline{x}) \models y & \quad (x \land \overline{x}) \models_{UNSAT} (y \land \overline{y}) \\
  (x \land \overline{x}) \models_{UNSAT} y \text{ does not hold!}
  \end{align*}
  \]
**Definition (Model Constructibility)**

A formula $G$ is model constructible with respect to a formula $F$ and to a set of variables $S$, in symbols $F \leadsto_{mc}^S G$, if for each total model $I$ of $F$ there exists a total model $I'$ of $G$ such that $I(x) = I'(x)$ for all $x \in (\mathcal{V} \setminus S)$.

**Definition (Constructibility)**

A formula $G$ is constructible from a formula $F$, in symbols $F \leadsto \cap G$, if for each model $I$ of $F$ there exists a model $I'$ of $G$ such that $I(x) = I'(x)$ for all $x \in \text{vars}(F)$.

**Definition (Mutual Constructibility)**

Two formulas $F$ and $G$ are mutually constructible, in symbols $F \leftrightarrow \cap G$, if $F \leadsto \cap G$ and $G \leadsto \cap F$. 
Mutual Constructibility

- Original formula
\[ F = (x \lor d) \land (\overline{a} \lor \overline{b} \lor x) \land (a \lor x) \land (b \lor x) \land (\overline{x} \lor c) \]

- Formula without \( x \), \( \text{vars}(F) \cap \text{vars}(G) = \{a, b, c, d\} \)
\[ G = (d \lor a) \land (d \lor b) \land (\overline{a} \lor \overline{b} \lor c) \]

- Both satisfiable: \( J_F = (abcdx) \quad J_G = (abcd\overline{x}) \)

- By changing the mapping of \( x \), \( J_F \) can be turned into \( J_G \), and vice versa. In this example, \( F \leftrightarrow \cap G \).
Formula Relations

More details in [Man14].
Elimination in SAT
Modern SAT Solving

- Successfully applied in different areas
  - hardware/software model checking, planning, optimization, verification, general purpose backend, . . .
- Many different input pattern
  - AND-gates, XOR-gates, cardinality constraints, clauses
- Combine different solving strategies
- Special purpose techniques
  - Gaussian Elimination, Cardinality Extraction, Variable Elimination, Clause Eliminations, Variable Addition, Failed Literal Probing
Solving Algorithms
**DavisPutnam (CNF formula $F$)**

**Input:** A formula $F$ in CNF

**Output:** The solution SAT or UNSAT of this formula

1. while true
2. if $F = \emptyset$ then return SAT  
   // satisfiability rule
3. if $\bot \in F$ then return UNSAT  
   // unsatisfiability rule
4. if $(x) \in F$ then  
   // unit rule
5. $F := F \mid_x$
6. continue
7. if $x \in \text{lits}(F)$ and $\bar{x} \notin \text{lits}(F)$ then  
   // pure literal rule
8. $F := F \mid_x$
9. continue
10. $G := F \setminus \{F_x \cup F_{\bar{x}}\}$  
    // clauses without $x$
11. $F := G \cup \{F_x \otimes F_{\bar{x}}\}$  
    // variable elimination
Using Elimination During Search

- **1960**: DP Algorithm [DP60]
- **1962**: search and backtracking instead of elimination (DLL) [DLL62]
- **1999**: backjumping and learning (CDCL) [MSS96]
- **200X**: improve heuristics, data structures [MMZ+01, SE02]
- **2005**: (partial) variable elimination as preprocessing
  - **MiniSAT with SatELite** [EB05]
- **2009**: simplification during search [Bie09]
- **2009**: (partial) Gaussian elimination [SNC09]
- **2012**: automated variable addition [MHB13]
- **2013**: (partial) cardinality reasoning [BLBLM14]

- Systems like **Lingeling**, **Riss** or **CryptoMiniSat** implement most of the above and schedule heuristically.
(Bounded) Variable Elimination

- Formula $F$ and variable $v$ to be eliminated
- $v$ might be **functionally dependent**, $v \leftrightarrow (a \land b)$
  - $G_v = \{ (v \lor \overline{a} \lor \overline{b}) \}$  $G_{\overline{v}} = \{ (\overline{v} \lor a), (\overline{v} \lor b) \}$
- before elimination, split:
  - $F_v = G_v \land R_v$  $F_{\overline{v}} = G_{\overline{v}} \land R_{\overline{v}}$
- new clauses $S := F_v \otimes F_{\overline{v}}$
- if functional dependent $S := R_v \otimes G_{\overline{v}} \land G_v \otimes R_{\overline{v}}$

$$F' := (F \setminus (F_v \cup F_{\overline{v}})) \cup S$$

- **Bounded (number of clauses matters):**
  - $|S| \leq |F_v| + |F_{\overline{v}}|$, ignoring tautologies
  - $|F_v| \leq 5 \land |F_{\overline{v}}| \leq 15$, or symmetric
Variable Elimination Example

- Original formula

\[ F = (x \lor d) \land (\overline{a} \lor \overline{b} \lor x) \land (a \lor \overline{x}) \land (b \lor \overline{x}) \land (\overline{x} \lor c) \]
Variable Elimination Example

- Original formula
  \[
  F = (x \lor d) \land (\overline{a} \lor \overline{b} \lor x) \land (a \lor \overline{x}) \land (b \lor \overline{x}) \land (\overline{x} \lor c)
  \]

- Subformulas
Variable Elimination Example

- Original formula
  \[ F = (x \lor d) \land (\bar{a} \lor \bar{b} \lor x) \land (a \lor \bar{x}) \land (b \lor \bar{x}) \land (\bar{x} \lor c) \]

- Subformulas
  \[ G_x = (a \lor b \lor x) \quad G_{\bar{x}} = (a \lor \bar{x}) \land (b \lor \bar{x}) \]
  \[ R_x = (x \lor d) \quad R_{\bar{x}} = (\bar{x} \lor c) \]
Variable Elimination Example

- Original formula
  \[ F = (x \lor d) \land (\overline{a} \lor \overline{b} \lor x) \land (a \lor \overline{x}) \land (b \lor \overline{x}) \land (\overline{x} \lor c) \]

- Subformulas
  \[ G_x = (\overline{a} \lor \overline{b} \lor x) \quad G_{\overline{x}} = (a \lor \overline{x}) \land (b \lor \overline{x}) \]
  \[ R_x = (x \lor d) \quad R_{\overline{x}} = (\overline{x} \lor c) \]

- Formula without \( x \)
  \[ S := G_x \otimes R_{\overline{x}} \land R_x \otimes G_{\overline{x}} \]
  \[ S = (d \lor a) \land (d \lor b) \land (\overline{a} \lor \overline{b} \lor c) \]

- Redundant:
  \[ G_x \otimes G_{\overline{x}} = \top \]
  \[ R_x \otimes R_{\overline{x}} = (c \lor d) \]
BVE in 2005 won the competition significantly (267 solved, 242 next)
Elimination using Constraints

(http://www.pragmaticsofssat.org/2012/application-caactus-pos12.png)
Elimination using Constraints

- Problems do not come in CNF
- $F$ might contain cardinality constraints (CCs) or XORs
- Extract constraints, apply reasoning there
  - Boolean domain is $\{0, 1\}$ instead of $\{\bot, \top\}$
- Find new constraints to be encoded to CNF
  - or *efficiently* prove inconsistency
Elimination using Constraints

- Problems do not come in CNF
- $F$ might contain cardinality constraints (CCs) or XORs
- Extract constraints, apply reasoning there
  - Boolean domain is $\{0, 1\}$ instead of $\{\bot, \top\}$
- Find new constraints to be encoded to CNF
  - or *efficiently* prove inconsistency
- Cardinality Constraints: $\sum_i w_i x_i \leq k$, with $w_i, k \in \mathbb{Z}$
  - Instead of resolution, use addition, and multiplication
Elimination using Constraints

- Problems do not come in CNF
- $F$ might contain cardinality constraints (CCs) or XORs
- Extract constraints, apply reasoning there
  - Boolean domain is $\{0, 1\}$ instead of $\{\bot, \top\}$
- Find new constraints to be encoded to CNF
  - or efficiently prove inconsistency
- Cardinality Constraints: $\sum_i w_i x_i \leq k$, with $w_i, k \in \mathbb{Z}$
  - Instead of resolution, use addition, and multiplication
- XORs: $\sum_i x_i \mod 2 = 1$, with $w_i, k \in \mathbb{Z}$
  - Instead of resolution, use addition with modulo
  - Find new XOR constraints to be encoded to CNF
Model Reconstruction
Model Reconstruction

- $J' \models F'$ does not imply $J' \models F$, $v$ can be mapped arbitrarily
- solver only finds $J'$
- simplifier knows $F$
Model Reconstruction

- $J' \models F'$ does not imply $J' \models F$, $v$ can be mapped arbitrarily
- solver only finds $J'$
- simplifier knows $F$

$$J = \begin{cases} 
(J' \setminus \{v\}) \cup \overline{v}, & \text{if } J' \not\models F_v \\
(J' \setminus \overline{v}) \cup \{v\}, & \text{if } J' \not\models F_{\overline{v}} \\
J', & \text{otherwise} 
\end{cases}$$
Model Reconstruction

- $J' \models F'$ does not imply $J' \models F$, $v$ can be mapped arbitrarily
- solver only finds $J'$
- simplifier knows $F$

$$J = \begin{cases} 
(J' \setminus \{v\}) \cup \{\overline{v}\}, & \text{if } J' \not\models F_{\overline{v}} \\
(J' \setminus \{\overline{v}\}) \cup \{v\}, & \text{if } J' \not\models F_v \\
J', & \text{otherwise}
\end{cases}$$

- Implementation
  - when eliminating $v$, store $F_v$ and $F_{\overline{v}}$
  - or, store only $F_v$ and set $J' := (J' \setminus \{v\}) \cup \{\overline{v}\}$
Variable Addition
Variable Addition

Definition (Extension)

A formula $F$ with two literals $l$ and $l'$ that occur in $F$ can be extended with a fresh variable $x$ to

$$F' = F \land (x \lor l) \land (x \lor l') \land (\overline{x} \lor \overline{l} \lor \overline{l'}).$$

- For any model $J'$ with $J' \models F'$, also $J' \models F$
- What would happen when using variable elimination next?
- Used for short theoretical proofs (extended resolution)
  - There exists clause based short proofs for e.g. pigeon hole
- Cannot be automated efficiently (as far as we know)
Variable Addition

Definition (Extension)

A formula $F$ with two literals $l$ and $l'$ that occur in $F$ can be extended with a fresh variable $x$ to $F' = F \land (x \lor l) \land (x \lor l') \land (\overline{x} \lor l \lor l')$.

- For any model $J'$ with $J' \models F'$, also $J' \models F$
- What would happen when using variable elimination next?
- Used for short theoretical proofs (extended resolution)
  - There exists clause based short proofs for e.g. pigeon hole
- Cannot be automated efficiently (as far as we know)
- Exploit **number of clauses matters**?
Can you reduce the number of clauses here?

\[ F := (a \lor c) \land (a \lor d) \land (a \lor e) \land (b \lor c) \land (b \lor d) \land (b \lor e) \]
Can you reduce the number of clauses here?

\[ F := (a \lor c) \land (a \lor d) \land (a \lor e) \land (b \lor c) \land (b \lor d) \land (b \lor e) \]

Simplified, with fresh variable \( x \)

\[ F' := (x \lor c) \land (x \lor d) \land (x \lor e) \land (a \lor \overline{x}) \land (b \lor \overline{x}) \]
Can you reduce the number of clauses here?

\[ F := (a \lor c) \land (a \lor d) \land (a \lor e) \land (b \lor c) \land (b \lor d) \land (b \lor e) \]

Simplified, with fresh variable \( x \)

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

How about variable elimination on \( x \)?
(Bounded) Variable Addition BVA

- Can you reduce the number of clauses here?
  \[ F := (a \lor c) \land (a \lor d) \land (a \lor e) \land (b \lor c) \land (b \lor d) \land (b \lor e) \]

- Simplified, with fresh variable \( x \)
  \[ F' := (x \lor c) \land (x \lor d) \land (x \lor e) \land (a \lor \overline{x}) \land (b \lor \overline{x}) \]

- How about variable elimination on \( x \)?
- BVA linearizes naive quadratic at-most-one encoding
Conclusion
Take Home Message

- Variable Elimination is an extremely powerful technique
- Produces mutual constructible formulas
- Similar techniques exist for higher level constraints
- The reverse – variable addition – is not that effective
- Elimination has to be applied limited
Elimination Techniques
In Modern Propositional Logic Reasoning

Norbert Manthey
nmanthey@conp-solutions.com

December 7, 2017

Thank you for your attention
Armin Biere.
PrecoSAT system description.

Armin Biere, Daniel Le Berre, Emmanuel Lonca, and Norbert Manthey.
Detecting cardinality constraints in CNF.

Martin Davis, George Logemann, and Donald Loveland.
A machine program for theorem-proving.

Martin Davis and Hilary Putnam.
A computing procedure for quantification theory.


João P. Marques-Silva and Karem A. Sakallah.
GRASP – a new search algorithm for satisfiability.

Niklas Sörensson and Niklas Eén.

Mate Soos, Karsten Nohl, and Claude Castelluccia.
Extending SAT solvers to cryptographic problems.
In SAT 2009, volume 5584 of LNCS, pages 244–257, 2009.