Automated Reencoding of Boolean Formulas

Norbert Manthey[1] Marijn J. H. Heule[2,3] Armin Biere[3]

[1]Institute of Artificial Intelligence, Technische Universität Dresden, Germany

[2]Department of Computer Science, University of Texas, Austin, United States

[3]Institute for Formal Models and Verification, Johannes Kepler University, Austria

November 6, 2012
Haifa Verification Conference
Haifa, Israel

Motivation: Encoding into SAT

Applications:

- verification, model checking, scheduling, . . .
- SAT solvers usually perform well, but not always . . .
- . . . for instance if the *wrong encoding* is chosen

What is a *good encoding*:

- small number of variables
- small number of clauses
- search space should be pruned by unit propagation
  - as in the original domain (arc consistency)

Motivation: Encoding Matters

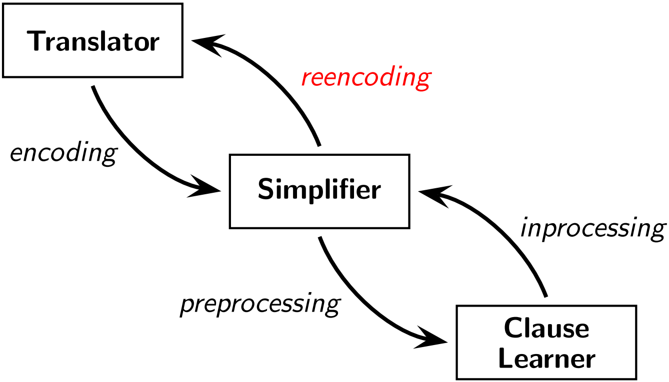
Context:

- The quality of the encoding has a huge impact on the performance of SAT solvers

Research Question:

- How can one *automatically* increase the quality of encodings?

Motivation: The Big Picture



Simplifiers

Variable Elimination [DavisPutnam'60]

Definition (Resolution)

Given two clauses $C = (x \vee a_1 \vee \dots \vee a_i)$ and $D = (\bar{x} \vee b_1 \vee \dots \vee b_j)$, the *resolvent* of C and D on variable x (denoted by $C \otimes_x D$) is $(a_1 \vee \dots \vee a_i \vee b_1 \vee \dots \vee b_j)$

Resolution on sets of clauses F_x and $F_{\bar{x}}$ (denoted by $F_x \otimes_x F_{\bar{x}}$) generates all (non-tautological) resolvents of $C \in F_x$ and $D \in F_{\bar{x}}$.

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

Proof procedure [DavisPutnam60]

VE is a complete proof procedure. Applying VE until fixpoint results in the empty formula (satisfiable) or empty clause (unsatisfiable)

Variable Elimination [DavisPutnam'60]

Definition (Resolution)

Given two clauses $C = (x \vee a_1 \vee \cdots \vee a_i)$ and $D = (\bar{x} \vee b_1 \vee \cdots \vee b_j)$, the *resolvent* of C and D on variable x (denoted by $C \otimes_x D$) is $(a_1 \vee \cdots \vee a_i \vee b_1 \vee \cdots \vee b_j)$

Resolution on sets of clauses F_x and $F_{\bar{x}}$ (denoted by $F_x \otimes_x F_{\bar{x}}$) generates all (non-tautological) resolvents of $C \in F_x$ and $D \in F_{\bar{x}}$.

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

Proof procedure [DavisPutnam60]

VE is a complete proof procedure. Applying VE until fixpoint results in the empty formula (satisfiable) or empty clause (unsatisfiable)

Variable Elimination [DavisPutnam'60]

Definition (Resolution)

Given two clauses $C = (x \vee a_1 \vee \cdots \vee a_i)$ and $D = (\bar{x} \vee b_1 \vee \cdots \vee b_j)$, the *resolvent* of C and D on variable x (denoted by $C \otimes_x D$) is $(a_1 \vee \cdots \vee a_i \vee b_1 \vee \cdots \vee b_j)$

Resolution on sets of clauses F_x and $F_{\bar{x}}$ (denoted by $F_x \otimes_x F_{\bar{x}}$) generates all (non-tautological) resolvents of $C \in F_x$ and $D \in F_{\bar{x}}$.

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

Proof procedure [DavisPutnam60]

VE is a complete proof procedure. Applying VE until fixpoint results in the empty formula (satisfiable) or empty clause (unsatisfiable)

Example (Bounded) VE [DavisPutnam'60] [EénBiere'05]

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 VE but not BVE

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

example: $|F_x \otimes F_{\bar{x}}| > |F_x| + |F_{\bar{x}}|$; in general: quadratic growth of clauses

Bounded VE (BVE): apply VE if the number of clauses does not increase.

Example (Bounded) VE [DavisPutnam'60] [EénBiere'05]

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 VE but not BVE

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

example: $|F_x \otimes F_{\bar{x}}| > |F_x| + |F_{\bar{x}}|$; in general: quadratic growth of clauses

Bounded VE (BVE): apply VE if the number of clauses does not increase.

Example (Bounded) VE [DavisPutnam'60] [EénBiere'05]

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 VE but not BVE

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

example: $|F_x \otimes F_{\bar{x}}| > |F_x| + |F_{\bar{x}}|$; in general: quadratic growth of clauses

Bounded VE (BVE): apply VE if the number of clauses does not increase.

Example (Bounded) VE [DavisPutnam'60] [EénBiere'05]

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 VE but not BVE

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

example: $|F_x \otimes F_{\bar{x}}| > |F_x| + |F_{\bar{x}}|$; in general: quadratic growth of clauses

Bounded VE (BVE): apply VE if the number of clauses does not increase.

Example (Bounded) VE [DavisPutnam'60] [EénBiere'05]

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 VE but not BVE

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

example: $|F_x \otimes F_{\bar{x}}| > |F_x| + |F_{\bar{x}}|$; in general: quadratic growth of clauses

Bounded VE (BVE): apply VE if the number of clauses does not increase.

Reencoding

Bounded Variable Addition: Main Idea

Main Idea

Given a CNF formula F, can we construct a 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?

Bounded Variable Addition: Main Idea

Main Idea

Given a CNF formula F, can we construct a 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?

Bounded Variable Addition: Main Idea

Main Idea

Given a CNF formula F, can we construct a 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 one clause *reduces* number of literals by 2

Not compatible with BVE, which would eliminate x immediately!

. . . so this does not work . . .

Bounded Variable Addition

Smallest Example

Replace

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

by

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

adds 1 variable

removes 1 clause

Bounded Variable Addition

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

Possible Patterns

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

- 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

How to find suitable patterns efficiently?

- Restrict the patterns to $|X_i| = 1$, i.e. single literals
- Test for each literal l whether it is part of a pattern

Bounded Variable Addition

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

Possible Patterns

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

- 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

How to find suitable patterns efficiently?

- Restrict the patterns to $|X_i| = 1$, i.e. single literals
- Test for each literal l whether it is part of a pattern

Bounded Variable Addition

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

Possible Patterns

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

- 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

How to find suitable patterns efficiently?

- Restrict the patterns to $|X_i| = 1$, i.e. single literals
- Test for each literal l whether it is part of a pattern

Bounded Variable Addition

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

Possible Patterns

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

- 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

How to find suitable patterns efficiently?

- Restrict the patterns to $|X_i| = 1$, i.e. single literals
- Test for each literal l whether it is part of a pattern

Bounded Variable Addition: Implementation

```
    SimpleBoundedVariableAddition (CNF formula F)
2   for l ∈ LIT(F) do
3         M_lit := {l}, M_cls := F_l
4         P := ∅
5      foreach C ∈ M_cls do
6         let l_min ∈ C \ {l} be least occurring in F
7            foreach D ∈ F_{l_min} do
8               if |C| = |D| and C \ D = l then
9                  l' := D \ C; P := P ∪ ⟨l', C⟩
11        let l_max be occurring most frequently in P
12        if adding l_max to M_lit further reduces |F| then
13           recalculate M_lit and M_cls; goto 4
17        if |M_lit| = 1 then continue
18        replace M_cls and M_lit with new clauses
26     return F
```

Impact on Cardinality Constraint Encodings

Cardinality Constraints / At-Most-One Constraints

Cardinality Constraints

Among the variables x_i, at most k are allowed to be assigned $\top$:

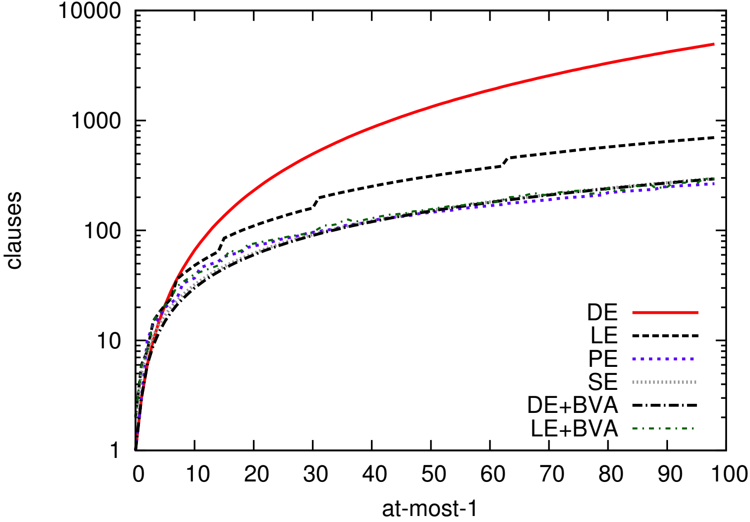
$$\sum x_i \leq k$$

- The talk focuses on At-Most-One Constraints ($k = 1$)
- Many Encodings have been proposed for At-Most-One Constraints

Good Encodings for Domains

Encoding	Clauses	
Direct Encoding (DE)	$\frac{n(n-1)}{2}$	many binary clauses
Log Encoding (LE)	$n \cdot \lceil \log n \rceil$	many duplicate patterns
Sequential Counter (SE)	$3n - 4$	represent as circuit first
Product Encoding (PE)	$2n + 4 \cdot \sqrt{n} + O(\sqrt[4]{n})$	best asymptotic bound

At-Most-One Constraints



BVA overcomes the drawback of DE and LE; for $k < 47$, DE+BVA is best

Evaluation on General SAT Instances

Results: FPGA Routing

- Try to route s inputs to t outputs (chnls_t)
- Uses many cardinality constraints with DE
- Results illustrate impact of BVA on cardinality constraints

	original			BVA preprocessed			
instance	#var	#cls	solve	#var	#cls	pre	solve
chnl10_11	220	1122	9372	302	562	0.00	**69.3**
chnl10_12	240	1344	7279	340	624	0.00	**15.0**
chnl10_13	260	1586	2682	380	686	0.00	**26.0**
chnl11_12	264	1476	TO	374	684	0.00	**41.6**
chnl11_13	286	1742	TO	418	752	0.00	**17.1**
chnl11_20	440	4220	TO	667	1228	0.00	**12.1**

Results: Bio-informatics

- Comparing gene evolutions by checking for same structure in trees
- No direct encoding inside the formulas
- BVA improves the encoding of the actual problem

	original			BVA preprocessed			
instance	#var	#cls	solve	#var	#cls	pre	solve
ndhf_09	1910	167476	TO	3098	14588	1.47	**187**
ndhf_10	2112	191333	TO	3418	16756	1.70	**1272**
rbcl_08	1278	67720	TO	1981	8669	0.29	**16**
rbcl_09	1430	79118	TO	2192	10157	0.39	**101**
rbcl_10	1584	91311	TO	2443	11811	0.43	**604**
rpoc_08	1278	74454	8628	2011	8494	0.39	**237**
rpoc_09	1430	86709	TO	2252	10063	0.47	**3590**
rpoc_10	1584	99781	TO	2474	11667	0.66	**11945**

Evaluation on General SAT Instances

Conclusions

- Bounded Variable Addition has been introduced
- Exchanges clauses for variables
- Adds a missing arc in the tool chain of SAT solving

- The quality of SAT encodings can be improved automatically
- Users of SAT solvers can rely on the solver to improve encoding
- Encoding and run time improvement on application instances

Automated Reencoding of Boolean Formulas

Norbert Manthey[1] Marijn J. H. Heule[2,3] Armin Biere[3]

[1]Institute of Artificial Intelligence, Technische Universität Dresden, Germany

[2]Department of Computer Science, University of Texas, Austin, United States

[3]Institute for Formal Models and Verification, Johannes Kepler University, Austria

November 6, 2012
Haifa Verification Conference
Haifa, Israel