Blocked Clause Elimination for QBF

Armin Biere, Florian Lonsing, and Martina Seidl

Institute for Formal Models and Verification (FMV)
Johannes Kepler University, Linz, Austria
`http://fmv.jku.at`

CADE'11
Wrocław, Poland
31 July - 5 August, 2011

SAT/QBF in Formal Verification:

- Systems and their properties: equivalence checking, BMC,. . .
- Propositional logic (SAT): NP-completeness.
- Quantified boolean formulae (QBF): PSPACE-completeness.
  - our focus!

Restricted Input Format of Solvers:

- E.g.: conjunctive normal form (CNF).
- But: formulae ϕ in practice are often non-CNF.
  - E.g. bounded model checking [BCCZ99]: transition relation of HW circuit.
- Original non-CNF encoding ϕ is optimized: ϕ_o.
- Optimized encoding ϕ_o is converted to CNF (loss of structure): ϕ_{oCNF}.
- Decision procedures (DP) operate on ϕ_{oCNF}.

$$\boxed{\text{System } S} \rightarrow \boxed{\text{verification problem } \phi} \rightarrow \boxed{\phi_o} \rightarrow \boxed{\phi_{oCNF}} \rightarrow \boxed{DP(\phi_{oCNF})}$$

SAT/QBF in Formal Verification:

- Systems and their properties: equivalence checking, BMC,. . .
- Propositional logic (SAT): NP-completeness.
- Quantified boolean formulae (QBF): PSPACE-completeness.
    - our focus!

Restricted Input Format of Solvers:

- E.g.: conjunctive normal form (CNF).
- But: formulae ϕ in practice are often non-CNF.
    - E.g. bounded model checking [BCCZ99]: transition relation of HW circuit.
- Original non-CNF encoding ϕ is optimized: ϕ_o.
- Optimized encoding ϕ_o is converted to CNF (loss of structure): ϕ_{oCNF}.
- Decision procedures (DP) operate on ϕ_{oCNF}.

$$\boxed{\text{System } S} \rightarrow \boxed{\text{verification problem } \phi} \rightarrow \boxed{\phi_o} \rightarrow \boxed{\phi_{oCNF}} \rightarrow \boxed{DP(\phi_{oCNF})}$$

How to Get Optimized Encoding ϕ_{oCNF}?

- Non-CNF/circuit-level preprocessing techniques.
- Optimized CNF conversions.
- See examples later.

$$\boxed{\text{System } S} \rightarrow \boxed{\text{verification problem } \phi} \rightarrow \boxed{\phi_o} \rightarrow \boxed{\phi_{oCNF}} \rightarrow \boxed{DP(\phi_{oCNF})}$$

Our Work:

- Blocked clause elimination (BCE) for QBF: QBCE.
- Original BCE comes from SAT-domain [JBH10].
- QBCE allows to remove clauses from CNF.
- QBCE on CNF is at least as effective as certain preprocessing techniques and optimized CNF conversion.
- *No* need to optimize non-CNF/circuit or CNF encoding separately.
- Using QBCE to preprocess input of CNF-based QBF solvers.

How to Get Optimized Encoding ϕ_{oCNF}?

- Non-CNF/circuit-level preprocessing techniques.
- Optimized CNF conversions.
- See examples later.

$$\boxed{\text{System } S} \rightarrow \boxed{\text{verification problem } \phi} \rightarrow \boxed{\phi_o} \rightarrow \boxed{\phi_{oCNF}} \rightarrow \boxed{DP(\phi_{oCNF})}$$

$$\searrow \boxed{\phi_{CNF}} \rightarrow \boxed{DP(QBCE(\phi_{CNF}))}$$

Our Work:

- Blocked clause elimination (BCE) for QBF: QBCE.
- Original BCE comes from SAT-domain [JBH10].
- QBCE allows to remove clauses from CNF.
- QBCE on CNF is at least as effective as certain preprocessing techniques and optimized CNF conversion.
- *No* need to optimize non-CNF/circuit or CNF encoding separately.
- Using QBCE to preprocess input of CNF-based QBF solvers.

Overview

- Quantified boolean formulae (QBF): syntax, semantics.
- Examples: circuit preprocessing, CNF conversion.
- Quantified blocked clause elimination (QBCE).
- Observations: QBCE subsumes approaches of circuit preprocessing and CNF conversion.
- Experiments: QBCE and related preprocessing techniques vs. state-of-the-art QBF preprocessing.
- Up to 30% more solved formulae by QBCE preprocessing + related techniques.

Propositional Logic:

- Boolean variables $x_1, \ldots, x_m$.
- Non-CNF/circuit formula $\phi(x_1, \ldots, x_m)$ with operators $\neg, \vee, \wedge$.

Quantified Boolean Formulae (QBF):

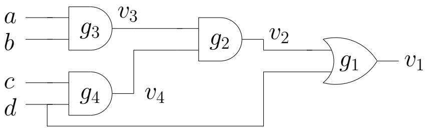
- Prenex CNF: quantifier-free CNF over quantified Boolean variables.
- PCNF $Q_1 S_1 \ldots Q_n S_n.\ \phi$, where $\phi := \bigwedge C_i$ a CNF with clauses C_i.
- Quantifiers $Q_i \in \{\exists, \forall\}$.
- Scope S_i: set of quantified variables.
- $Q_i S_i \leq Q_{i+1} S_{i+1}$: scopes are linearly ordered.
- Semantics recursively based on formula structure:
  - $\forall x \phi$ is sat. iff both $\phi[x/0]$ and $\phi[x/1]$ are sat.
  - $\exists x \phi$ is sat. iff either $\phi[x/0]$ or $\phi[x/1]$ is sat.

Example

A CNF: $(x \vee \neg y) \wedge (\neg x \vee y)$ and a PCNF: $\forall x \exists y.\ (x \vee \neg y) \wedge (\neg x \vee y)$.

Example: Non-Shared Input Elimination (NSI)

In all examples, assume only ∃-quantifiers for simplicity.



Gate g_3 is redundant by NSI.

Tseitin Encoding [Tse68]:

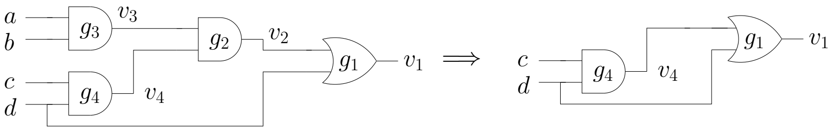
$v_1 \Leftrightarrow (v_2 \vee d)$:
$(\neg v_1 \vee v_2 \vee d) \wedge (v_1 \vee \neg v_2) \wedge (v_1 \vee \neg d)$

$v_2 \Leftrightarrow (v_3 \wedge v_4)$:
$(\neg v_2 \vee v_3) \wedge (\neg v_2 \vee v_4) \wedge (v_2 \vee \neg v_3 \vee \neg v_4)$

$v_3 \Leftrightarrow (a \wedge b)$:
$(\neg v_3 \vee a) \wedge (\neg v_3 \vee b) \wedge (v_3 \vee \neg a \vee \neg b)$

$v_4 \Leftrightarrow (c \wedge d)$:
$(\neg v_4 \vee c) \wedge (\neg v_4 \vee d) \wedge (v_4 \vee \neg c \vee \neg d)$

Example: Non-Shared Input Elimination (NSI)



Gate g_3 is redundant by NSI.

Tseitin Encoding [Tse68]:

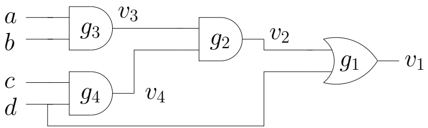
$v_1 \Leftrightarrow (v_2 \vee d)$:
$(\neg v_1 \vee v_2 \vee d) \wedge (v_1 \vee \neg v_2) \wedge (v_1 \vee \neg d)$

$v_2 \Leftrightarrow (v_3 \wedge v_4)$:
$(\neg v_2 \vee v_3) \wedge (\neg v_2 \vee v_4) \wedge (v_2 \vee \neg v_3 \vee \neg v_4)$

CNF-level QBCE subsumes circuit-level NSI.

$\cancel{v_3 \Leftrightarrow (a \wedge b) :}$
$\cancel{(\neg v_3 \vee a) \wedge (\neg v_3 \vee b) \wedge (v_3 \vee \neg a \vee \neg b)}$

$v_4 \Leftrightarrow (c \wedge d)$:
$(\neg v_4 \vee c) \wedge (\neg v_4 \vee d) \wedge (v_4 \vee \neg c \vee \neg d)$



Tseitin Encoding:

$v_1 \Leftrightarrow (v_2 \vee d)$:
$(\neg v_1 \vee v_2 \vee d) \wedge (v_1 \vee \neg v_2) \wedge (v_1 \vee \neg d)$

$v_2 \Leftrightarrow (v_3 \wedge v_4)$:
$(\neg v_2 \vee v_3) \wedge (\neg v_2 \vee v_4) \wedge (v_2 \vee \neg v_3 \vee \neg v_4)$

$v_3 \Leftrightarrow (a \wedge b)$:
$(\neg v_3 \vee a) \wedge (\neg v_3 \vee b) \wedge (v_3 \vee \neg a \vee \neg b)$

$v_4 \Leftrightarrow (c \wedge d)$:
$(\neg v_4 \vee c) \wedge (\neg v_4 \vee d) \wedge (v_4 \vee \neg c \vee \neg d)$

PG Encoding (Only "⇒") [PG86]:

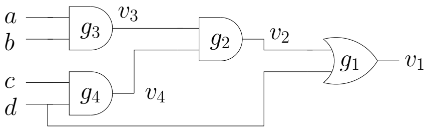
$v_1 \Rightarrow (v_2 \vee d)$:
$(\neg v_1 \vee v_2 \vee d) \cancel{\wedge (v_1 \vee \neg v_2) \wedge (v_1 \vee \neg d)}$

$v_2 \Rightarrow (v_3 \wedge v_4)$:
$(\neg v_2 \vee v_3) \wedge (\neg v_2 \vee v_4) \cancel{\wedge (v_2 \vee \neg v_3 \vee \neg v_4)}$

$v_3 \Rightarrow (a \wedge b)$:
$(\neg v_3 \vee a) \wedge (\neg v_3 \vee b) \cancel{\wedge (v_3 \vee \neg a \vee \neg b)}$

$v_4 \Rightarrow (c \wedge d)$:
$(\neg v_4 \vee c) \wedge (\neg v_4 \vee d) \cancel{\wedge (v_4 \vee \neg c \vee \neg d)}$

CNF-level QBCE subsumes Plaisted-Greenbaum encoding.



Tseitin Encoding:

$v_1 \Leftrightarrow (v_2 \vee d)$:
$(\neg v_1 \vee v_2 \vee d) \wedge (v_1 \vee \neg v_2) \wedge (v_1 \vee \neg d)$

$v_2 \Leftrightarrow (v_3 \wedge v_4)$:
$(\neg v_2 \vee v_3) \wedge (\neg v_2 \vee v_4) \wedge (v_2 \vee \neg v_3 \vee \neg v_4)$

$v_3 \Leftrightarrow (a \wedge b)$:
$(\neg v_3 \vee a) \wedge (\neg v_3 \vee b) \wedge (v_3 \vee \neg a \vee \neg b)$

$v_4 \Leftrightarrow (c \wedge d)$:
$(\neg v_4 \vee c) \wedge (\neg v_4 \vee d) \wedge (v_4 \vee \neg c \vee \neg d)$

PG Encoding (Only "$\Rightarrow$") [PG86]:

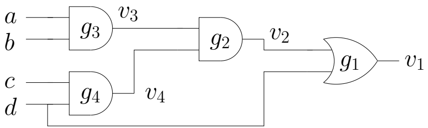
$v_1 \Rightarrow (v_2 \vee d)$:
$(\neg v_1 \vee v_2 \vee d) \cancel{\wedge (v_1 \vee \neg v_2) \wedge (v_1 \vee \neg d)}$

$v_2 \Rightarrow (v_3 \wedge v_4)$:
$(\neg v_2 \vee v_3) \wedge (\neg v_2 \vee v_4) \cancel{\wedge (v_2 \vee \neg v_3 \vee \neg v_4)}$

$v_3 \Rightarrow (a \wedge b)$:
$(\neg v_3 \vee a) \wedge (\neg v_3 \vee b) \cancel{\wedge (v_3 \vee \neg a \vee \neg b)}$

$v_4 \Rightarrow (c \wedge d)$:
$(\neg v_4 \vee c) \wedge (\neg v_4 \vee d) \cancel{\wedge (v_4 \vee \neg c \vee \neg d)}$

CNF-level QBCE subsumes Plaisted-Greenbaum encoding.



Tseitin Encoding:

$v_1 \Leftrightarrow (v_2 \vee d)$:
$(\neg v_1 \vee v_2 \vee d) \wedge (v_1 \vee \neg v_2) \wedge (v_1 \vee \neg d)$

$v_2 \Leftrightarrow (v_3 \wedge v_4)$:
$(\neg v_2 \vee v_3) \wedge (\neg v_2 \vee v_4) \wedge (v_2 \vee \neg v_3 \vee \neg v_4)$

$v_3 \Leftrightarrow (a \wedge b)$:
$(\neg v_3 \vee a) \wedge (\neg v_3 \vee b) \wedge (v_3 \vee \neg a \vee \neg b)$

$v_4 \Leftrightarrow (c \wedge d)$:
$(\neg v_4 \vee c) \wedge (\neg v_4 \vee d) \wedge (v_4 \vee \neg c \vee \neg d)$

PG Encoding (Only "$\Rightarrow$") [PG86]:

$v_1 \Rightarrow (v_2 \vee d)$:
$(\neg v_1 \vee v_2 \vee d) \cancel{\wedge (v_1 \vee \neg v_2) \wedge (v_1 \vee \neg d)}$

$v_2 \Rightarrow (v_3 \wedge v_4)$:
$(\neg v_2 \vee v_3) \wedge (\neg v_2 \vee v_4) \cancel{\wedge (v_2 \vee \neg v_3 \vee \neg v_4)}$

$v_3 \Rightarrow (a \wedge b)$:
$(\neg v_3 \vee a) \wedge (\neg v_3 \vee b) \cancel{\wedge (v_3 \vee \neg a \vee \neg b)}$

$v_4 \Rightarrow (c \wedge d)$:
$(\neg v_4 \vee c) \wedge (\neg v_4 \vee d) \cancel{\wedge (v_4 \vee \neg c \vee \neg d)}$

CNF-level QBCE subsumes Plaisted-Greenbaum encoding.

Blocked Clause Elimination (BCE) for SAT [JBH10]

- Allows to remove *blocked* clauses from CNF.
- At least as effective as many circuit-level preprocessing techniques.
- Subsumes NSI, pure literal rule, Plaisted-Greenbaum encoding.
- Based on checking all possible resolvents of a variable.

Quantified Blocked Clause Elimination (QBCE) for QBF

- Generalizes BCE to QBF: minor but crucial adaption of BCE definition.
- Same benefits as BCE for SAT.
- Implementation: tool “bloqqer” combines QBCE and extensions with variable elimination, resolution, subsumption,. . .

Definition of QBCE: based checking possible Q-resolvents.

Resolution for QBF

Q-Resolution: propositional resolution + universal reduction (UR).

Definition (Universal Reduction)

Given a clause C, $UR(C) := C \setminus \{l_u \in L_\forall(C) \mid \nexists l_e \in L_\exists(C), l_u < l_e\}$, i.e. deleting universal literals which are "tailing" in C by quantifier ordering $<$.

Example (UR)

Given PCNF $\exists x \forall y \exists z. (x \vee y \vee z) \wedge (\neg x \vee \neg y)$. Then $UR((\neg x \vee \neg y)) = (\neg x)$.

Definition (Q-Resolution)

Let C_1, C_2 be clauses with $v \in C_1, \neg v \in C_2$ and $q(v) = \exists$ [BKF95].

1. Tentative Q-resolvent: $C_1 \otimes C_2 := (UR(C_1) \cup UR(C_2)) \setminus \{v, \neg v\}$.
2. If $\{x, \neg x\} \subseteq C_1 \otimes C_2$ then no Q-resolvent exists.
   $C_1 \otimes C_2$ is tautologous *with respect to* variable x.
3. Otherwise, Q-resolvent $C := UR(C_1 \otimes C_2)$.

Q-Resolution: propositional resolution + universal reduction (UR).

Definition (Universal Reduction)

Given a clause C, $UR(C) := C \setminus \{l_u \in L_\forall(C) \mid \nexists l_e \in L_\exists(C), l_u < l_e\}$, i.e. deleting universal literals which are "tailing" in C by quantifier ordering $<$.

Example (UR)

Given PCNF $\exists x \forall y \exists z. (x \vee y \vee z) \wedge (\neg x \vee \neg y)$. Then $UR((\neg x \vee \neg y)) = (\neg x)$.

Definition (Q-Resolution)

Let C_1, C_2 be clauses with $v \in C_1, \neg v \in C_2$ and $q(v) = \exists$ [BKF95].

1. Tentative Q-resolvent: $C_1 \otimes C_2 := (UR(C_1) \cup UR(C_2)) \setminus \{v, \neg v\}$.
2. If $\{x, \neg x\} \subseteq C_1 \otimes C_2$ then no Q-resolvent exists. $C_1 \otimes C_2$ is tautologous *with respect to* variable x.
3. Otherwise, Q-resolvent $C := UR(C_1 \otimes C_2)$.

Q-Resolution: propositional resolution + universal reduction (UR).

Definition (Universal Reduction)

Given a clause C, $UR(C) := C \setminus \{l_u \in L_\forall(C) \mid \nexists l_e \in L_\exists(C), l_u < l_e\}$, i.e. deleting universal literals which are "tailing" in C by quantifier ordering $<$.

Example (UR)

Given PCNF $\exists x \forall y \exists z.\, (x \vee y \vee z) \wedge (\neg x \vee \neg y)$. Then $UR((\neg x \vee \neg y)) = (\neg x)$.

Definition (Q-Resolution)

Let C_1, C_2 be clauses with $v \in C_1, \neg v \in C_2$ and $q(v) = \exists$ [BKF95].

1. Tentative Q-resolvent: $C_1 \otimes C_2 := (UR(C_1) \cup UR(C_2)) \setminus \{v, \neg v\}$.
2. If $\{x, \neg x\} \subseteq C_1 \otimes C_2$ then no Q-resolvent exists. $C_1 \otimes C_2$ is tautologous *with respect to* variable x.
3. Otherwise, Q-resolvent $C := UR(C_1 \otimes C_2)$.

Definition (Quantified Blocking Literal)

Given PCNF $\psi := Q_1 S_1 \ldots Q_n S_n.\ \phi$, a literal l in a clause $C \in \psi$ is a *quantified blocking literal* if for every clause C' with $\neg l \in C'$, $C \otimes C'$ is tautologous wrt. some variable k such that $k \leq l$ in prefix ordering.

$C_1 \in Occs(l)$ blocked?	$C_2 \in Occs(\neg l)$	$C_1 \otimes C_2$
$(x_1 \vee x_2 \vee \ldots \vee x_n \vee \ldots \vee l \vee \ldots)$	$(\ldots \neg x_1 \vee \ldots \vee \neg l \vee \ldots)$	$\{x_1, \neg x_1\} \in C_1 \otimes C_2$
	$(\ldots \neg x_2 \vee \ldots \vee \neg l \vee \ldots)$	$\{x_2, \neg x_2\} \in C_1 \otimes C_2$
	$\ldots$	
	$(\ldots \neg x_n \vee \ldots \vee \neg l \vee \ldots)$	$\{x_n, \neg x_n\} \in C_1 \otimes C_2$

Definition (Quantified Blocked Clause)

Given PCNF $Q_1 S_1 \ldots Q_n S_n.\ (\phi \wedge C)$. Clause C is *quantified blocked* if it contains a quantified blocking literal. Removing C preserves satisfiability.

$Q_1 S_1 \ldots Q_n S_n.\ (\phi \wedge C) \overset{sat}{\equiv} Q_1 S_1 \ldots Q_n S_n.\ \phi.$

Example

All clauses blocked: $\forall x \exists y ((x \vee \neg y) \wedge (\neg x \vee y))$.
No clause blocked: $\exists x \forall y ((x \vee \neg y) \wedge (\neg x \vee y))$.

QBCE Definition

Definition (Quantified Blocking Literal)

Given PCNF $\psi := Q_1 S_1 \dots Q_n S_n.\ \phi$, a literal l in a clause $C \in \psi$ is a *quantified blocking literal* if for every clause C' with $\neg l \in C'$, $C \otimes C'$ is tautologous wrt. some variable k such that $k \leq l$ in prefix ordering.

$C_1 \in Occs(l)$ blocked?	$C_2 \in Occs(\neg l)$	$C_1 \otimes C_2$
$(x_1 \vee x_2 \vee \dots \vee x_n \vee \dots \vee l \vee \dots)$	$(\dots \neg x_1 \vee \dots \vee \neg l \vee \dots)$	$\{x_1, \neg x_1\} \in C_1 \otimes C_2$
	$(\dots \neg x_2 \vee \dots \vee \neg l \vee \dots)$	$\{x_2, \neg x_2\} \in C_1 \otimes C_2$
	$\dots$	
	$(\dots \neg x_n \vee \dots \vee \neg l \vee \dots)$	$\{x_n, \neg x_n\} \in C_1 \otimes C_2$

Definition (Quantified Blocked Clause)

Given PCNF $Q_1 S_1 \dots Q_n S_n.\ (\phi \wedge C)$. Clause C is *quantified blocked* if it contains a quantified blocking literal. Removing C preserves satisfiability.

$Q_1 S_1 \dots Q_n S_n.\ (\phi \wedge C) \stackrel{sat}{\equiv} Q_1 S_1 \dots Q_n S_n.\ \phi.$

Example

All clauses blocked: $\forall x \exists y((x \vee \neg y) \wedge (\neg x \vee y))$.
No clause blocked: $\exists x \forall y((x \vee \neg y) \wedge (\neg x \vee y))$.

QBCE Definition

Definition (Quantified Blocking Literal)

Given PCNF $\psi := Q_1 S_1 \dots Q_n S_n.\ \phi$, a literal l in a clause $C \in \psi$ is a *quantified blocking literal* if for every clause C' with $\neg l \in C'$, $C \otimes C'$ is tautologous wrt. some variable k such that $k \leq l$ in prefix ordering.

$C_1 \in Occs(l)$ blocked?	$C_2 \in Occs(\neg l)$	$C_1 \otimes C_2$
$(x_1 \vee x_2 \vee \dots \vee x_n \vee \dots \vee l \vee \dots)$	$(\dots \neg x_1 \vee \dots \vee \neg l \vee \dots)$	$\{x_1, \neg x_1\} \in C_1 \otimes C_2$
	$(\dots \neg x_2 \vee \dots \vee \neg l \vee \dots)$	$\{x_2, \neg x_2\} \in C_1 \otimes C_2$
	$\dots$	
	$(\dots \neg x_n \vee \dots \vee \neg l \vee \dots)$	$\{x_n, \neg x_n\} \in C_1 \otimes C_2$

Definition (Quantified Blocked Clause)

Given PCNF $Q_1 S_1 \dots Q_n S_n.\ (\phi \wedge C)$. Clause C is *quantified blocked* if it contains a quantified blocking literal. Removing C preserves satisfiability.
$Q_1 S_1 \dots Q_n S_n.\ (\phi \wedge C) \overset{sat}{\equiv} Q_1 S_1 \dots Q_n S_n.\ \phi$.

Example

All clauses blocked: $\forall x \exists y ((x \vee \neg y) \wedge (\neg x \vee y))$.
No clause blocked: $\exists x \forall y ((x \vee \neg y) \wedge (\neg x \vee y))$.

QBCE Definition

Definition (Quantified Blocking Literal)

Given PCNF $\psi := Q_1 S_1 \dots Q_n S_n.\ \phi$, a literal l in a clause $C \in \psi$ is a *quantified blocking literal* if for every clause C' with $\neg l \in C'$, $C \otimes C'$ is tautologous wrt. some variable k such that $k \leq l$ in prefix ordering.

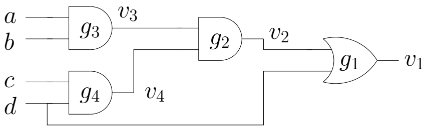
$C_1 \in Occs(l)$ blocked?	$C_2 \in Occs(\neg l)$	$C_1 \otimes C_2$
$(x_1 \vee x_2 \vee \dots \vee x_n \vee \dots \vee l \vee \dots)$	$(\dots \neg x_1 \vee \dots \vee \neg l \vee \dots)$	$\{x_1, \neg x_1\} \in C_1 \otimes C_2$
	$(\dots \neg x_2 \vee \dots \vee \neg l \vee \dots)$	$\{x_2, \neg x_2\} \in C_1 \otimes C_2$
	$\dots$	
	$(\dots \neg x_n \vee \dots \vee \neg l \vee \dots)$	$\{x_n, \neg x_n\} \in C_1 \otimes C_2$

Definition (Quantified Blocked Clause)

Given PCNF $Q_1 S_1 \dots Q_n S_n.\ (\phi \wedge C)$. Clause C is *quantified blocked* if it contains a quantified blocking literal. Removing C preserves satisfiability.
$Q_1 S_1 \dots Q_n S_n.\ (\phi \wedge C) \overset{sat}{\equiv} Q_1 S_1 \dots Q_n S_n.\ \phi$.

Example

All clauses blocked: $\forall x \exists y ((x \vee \neg y) \wedge (\neg x \vee y))$.
No clause blocked: $\exists x \forall y ((x \vee \neg y) \wedge (\neg x \vee y))$.



Gate g_3 is redundant by NSI.

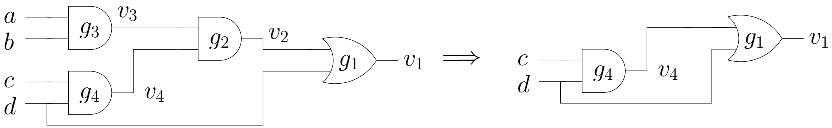
Tseitin Encoding:

$v_1 \Leftrightarrow (v_2 \vee d)$:
$(\neg v_1 \vee v_2 \vee d) \wedge (v_1 \vee \neg v_2) \wedge (v_1 \vee \neg d)$

$v_2 \Leftrightarrow (v_3 \wedge v_4)$:
$(\neg v_2 \vee v_3) \wedge (\neg v_2 \vee v_4) \wedge (v_2 \vee \neg v_3 \vee \neg v_4)$

$v_3 \Leftrightarrow (a \wedge b)$:
$(\neg v_3 \vee a) \wedge (\neg v_3 \vee b) \wedge (v_3 \vee \neg a \vee \neg b)$

$v_4 \Leftrightarrow (c \wedge d)$:
$(\neg v_4 \vee c) \wedge (\neg v_4 \vee d) \wedge (v_4 \vee \neg c \vee \neg d)$



Gate g_3 is redundant by NSI.

Tseitin Encoding:

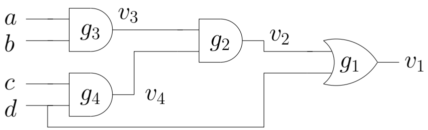
$v_1 \Leftrightarrow (v_2 \lor d)$:
$(\neg v_1 \lor v_2 \lor d) \land (v_1 \lor \neg v_2) \land (v_1 \lor \neg d)$

$v_2 \Leftrightarrow (v_3 \land v_4)$:
$(\neg v_2 \lor v_3) \land (\neg v_2 \lor v_4) \land (v_2 \lor \neg v_3 \lor \neg v_4)$

$v_3 \Leftrightarrow (a \land b)$:
$(\neg v_3 \lor a) \land (\neg v_3 \lor b) \land (v_3 \lor \neg a \lor \neg b)$

$v_4 \Leftrightarrow (c \land d)$:
$(\neg v_4 \lor c) \land (\neg v_4 \lor d) \land (v_4 \lor \neg c \lor \neg d)$

QBCE (blocking literals are blue) removes clauses resulting from redundant part $v_3 \Leftrightarrow (a \land b)$, and possibly more.



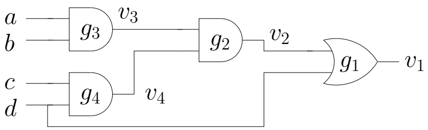
Tseitin Encoding:

$v_1 \Leftrightarrow (v_2 \vee d)$:
$(\neg v_1 \vee v_2 \vee d) \wedge (v_1 \vee \neg v_2) \wedge (v_1 \vee \neg d)$

$v_2 \Leftrightarrow (v_3 \wedge v_4)$:
$(\neg v_2 \vee v_3) \wedge (\neg v_2 \vee v_4) \wedge (v_2 \vee \neg v_3 \vee \neg v_4)$

$v_3 \Leftrightarrow (a \wedge b)$:
$(\neg v_3 \vee a) \wedge (\neg v_3 \vee b) \wedge (v_3 \vee \neg a \vee \neg b)$

$v_4 \Leftrightarrow (c \wedge d)$:
$(\neg v_4 \vee c) \wedge (\neg v_4 \vee d) \wedge (v_4 \vee \neg c \vee \neg d)$



Tseitin Encoding:

$v_1 \Leftrightarrow (v_2 \vee d)$:
$(\neg v_1 \vee v_2 \vee d) \wedge (v_1 \vee \neg v_2) \wedge (v_1 \vee \neg d)$

$v_2 \Leftrightarrow (v_3 \wedge v_4)$:
$(\neg v_2 \vee v_3) \wedge (\neg v_2 \vee v_4) \wedge (v_2 \vee \neg v_3 \vee \neg v_4)$

$v_3 \Leftrightarrow (a \wedge b)$:
$(\neg v_3 \vee a) \wedge (\neg v_3 \vee b) \wedge (v_3 \vee \neg a \vee \neg b)$

$v_4 \Leftrightarrow (c \wedge d)$:
$(\neg v_4 \vee c) \wedge (\neg v_4 \vee d) \wedge (v_4 \vee \neg c \vee \neg d)$

QBCE (Blocking Literals are Blue):

$v_1 \Rightarrow (v_2 \vee d)$:
$(\neg v_1 \vee v_2 \vee d) \cancel{\wedge (v_1 \vee \neg v_2) \wedge (v_1 \vee \neg d)}$

$v_2 \Rightarrow (v_3 \wedge v_4)$:
$(\neg v_2 \vee v_3) \wedge (\neg v_2 \vee v_4) \cancel{\wedge (v_2 \vee \neg v_3 \vee \neg v_4)}$

$v_3 \Rightarrow (a \wedge b)$:
$(\neg v_3 \vee a) \wedge (\neg v_3 \vee b) \cancel{\wedge (v_3 \vee \neg a \vee \neg b)}$

$v_4 \Rightarrow (c \wedge d)$:
$(\neg v_4 \vee c) \wedge (\neg v_4 \vee d) \cancel{\wedge (v_4 \vee \neg c \vee \neg d)}$

- QBCE removes clauses from direction "$\Leftarrow$" (same as PG: only "$\Rightarrow$").
- Apply Tseitin encoding to original circuit and optimize CNF by QBCE.

QBCE vs. State-Of-The-Art QBF Preprocessing

Equivalence Rewriting (ER): part of sQueezeBF [GMN10b] preprocessor.

Lemma

Let $\phi = Q_1 S_1 \ldots Q_n S_n.\ ((l \vee \alpha) \wedge (l \Leftrightarrow \gamma) \wedge \psi)$ *such that*

- *quant(l) = ∃,*
- *l neither occurs in CNFs* $\psi, \alpha,$ *nor* γ*, and*
- $k \leq l$ *for all literals k occurring in* γ*.*

Then $\phi \equiv Q_1 S_1 \ldots Q_n S_n.\ ((l \vee \alpha) \wedge (l \Rightarrow \gamma) \wedge \psi)$*.*
If α *is true, then l occurs only negatively and hence* $(l \Rightarrow \gamma)$ *can be removed.*

Observation: QBCE subsumes ER.

- Rewrite $(l \Leftrightarrow \gamma)$ into $(l \Rightarrow \gamma) \wedge (l \Leftarrow \gamma)$, i.e. $(\neg l \vee \gamma) \wedge (l \vee \neg\gamma)$.
- Then clauses $(l \vee \neg\gamma)$ of direction "$\Leftarrow$" are blocked and can be removed.

Observation: QBCE might be applicable when ER is not.

Example

$\forall y \exists x \exists z.\ ((x \vee z) \wedge (\neg x \vee \neg z) \wedge (z \vee \neg y \vee x) \wedge (\neg z \vee y \vee \neg x))$ is reducible by QBCE, but not by ER (applicable only after subsumption).

Experiments: QBCE for Preprocessing

Tool "bloqqer": implementation of QBCE.

- Combines (extensions of) QBCE with variable elimination (resolution, expansion), equivalence reasoning.

Reference Tool "sQueezeBF": part of QuBE solver [GMN10a, GMN10b].

- State-of-the-art PCNF preprocessing tool.
- Variable elimination, equivalence reasoning/rewriting/splitting,. . .

Setting: combination of bloqqer and sQueezeBF with various QBF solvers.

- Backtracking search: DepQBF 0.1 [LB10], QuBE 7.1 [GMN10a].
- Quantifier elimination: Quantor [Bie04].

Experiments (cont'd): Formula Sizes

Comparing/combining sQueezeBF and bloqqer:

- QBFEVAL 2010 benchmark set [PPT$^+$10], 568 formulae, 7 GB / 900 sec. limits.
- Statistics based on 372 formulae fully preprocessed but not solved by any of four combinations.
- Formulae solved by preprocessing, average preprocessing time (sec.).
- Last two columns: avg. ratios of variables ("v-ratio") and clauses ("c-ratio") in preprocessed/original formulae.
- Heuristic cutoff parameters to keep preprocessing times small.

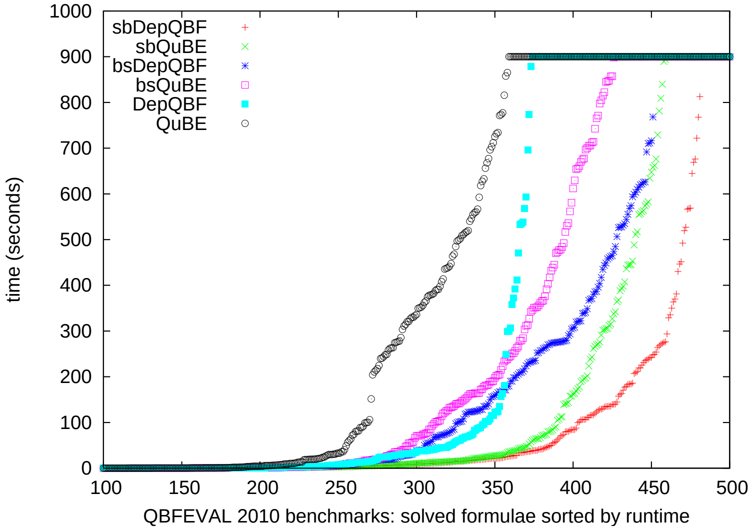
preprocessing	*solved*	*time*	*avg. v-ratio*	*avg. c-ratio*
sQueezeBF/bloqqer	161	37.9	0.489	1.097
bloqqer/sQueezeBF	151	92.7	0.501	1.086
bloqqer	149	7.07	0.490	1.112
sQueezeBF	39	36.25	0.958	0.610

QBFEVAL 2010 benchmark set [PPT+10], 568 formulae, 7 GB / 900 sec. limits.

		# formulas (total 568)			*run time (sec)*	
	preprocessing	*solved*	*sat*	*unsat*	*avg*	*med*
DepQBF	sQueezeBF/bloqqer	482 (+29%)	234	248	180	5
	bloqqer	467 (+25%)	224	243	198	5
	bloqqer/sQueezeBF	452 (+21%)	213	239	258	19
	sQueezeBF	435 (+16%)	201	234	231	6
	none	373	167	206	332	26
QuBE	sQueezeBF/bloqqer	454 (+36%)	207	247	227	7
	bloqqer	444 (+33%)	200	244	246	5
	bloqqer/sQueezeBF	421 (+26%)	183	238	307	27
	sQueezeBF	406 (+22%)	181	225	313	31
	none	332	135	197	426	258
Quantor	bloqqer	288 (+39%)	145	143	468	34
	sQueezeBF/bloqqer	285 (+38%)	147	138	472	39
	bloqqer/sQueezeBF	270 (+31%)	131	139	486	34
	sQueezeBF	222 (+7%)	106	116	561	49
	none	206	100	106	587	38

Experiments (cont'd): Times Plot

DepQBF and QuBE with "sQueezeBF/bloqqer" ("sb") and "bloqqer/sQueezeBF" ("bs").



Conclusions

Non-CNF/Circuit Formulae:

- Encodings of problems in practice typically not in CNF.
- Conversion to CNF might destroy structure: circuit-level preprocessing.

Quantified Blocked Clause Elimination (QBCE):

- Generalizes BCE for SAT to QBF.
- BCE on CNF subsumes circuit preprocessing and encoding techniques.
- In practice: QBCE (QBF) seems to be more important than BCE (SAT).
- Good performance when combined with other preprocessing techniques.

Future Work:

- Dynamic QBCE in search-based QBF solvers.

Bloqqer is open-source: `http://fmv.jku.at/bloqqer/`

DepQBF is open-source: `http://fmv.jku.at/depqbf/`

References

A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu.
Symbolic Model Checking without BDDs.
In R. Cleaveland, editor, *TACAS*, volume 1579 of *LNCS*, pages 193–207. Springer, 1999.

A. Biere.
Resolve and Expand.
In H. H. Hoos and D. G. Mitchell, editors, *SAT (Selected Papers)*, volume 3542 of *LNCS*, pages 59–70. Springer, 2004.

H. Kleine Büning, M. Karpinski, and A. Flögel.
Resolution for Quantified Boolean Formulas.
Inf. Comput., 117(1):12–18, 1995.

E. Giunchiglia, P. Marin, and M. Narizzano.
QuBE7.0 (System Description).
JSAT, 7:83–88, 2010.

E. Giunchiglia, P. Marin, and Massimo Narizzano.
sQueezeBF: An Effective Preprocessor for QBFs Based on Equivalence Reasoning.
In O. Strichman and S. Szeider, editors, *SAT*, volume 6175 of *LNCS*, pages 85–98. Springer, 2010.

M. Järvisalo, A. Biere, and M. Heule.

Blocked Clause Elimination.
In J. Esparza and R. Majumdar, editors, *TACAS*, volume 6015 of *LNCS*, pages 129–144. Springer, 2010.

F. Lonsing and A. Biere.
DepQBF: A Dependency-Aware QBF Solver.
JSAT, 7(2-3):71–76, 2010.

D. A. Plaisted and S. Greenbaum.
A Structure-Preserving Clause Form Translation.
J. Symb. Comput., 2(3):293–304, 1986.

C. Peschiera, L. Pulina, A. Tacchella, U. Bubeck, O. Kullmann, and I. Lynce.
The Seventh QBF Solvers Evaluation (QBFEVAL'10).
In O. Strichman and S. Szeider, editors, *SAT*, volume 6175 of *LNCS*, pages 237–250. Springer, 2010.

G. S. Tseitin.
On the Complexity of Derivation in Propositional Calculus.
Studies in Constructive Mathematics and Mathematical Logic, 1968.