Blocked Clause Elimination for QBF

Armin Biere*, Florian Lonsing, and Martina Seidl

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

Abstract. Quantified Boolean formulas (QBF) provide a powerful framework for encoding problems from various application domains, not least because efficient QBF solvers are available. Despite sophisticated evaluation techniques, the performance of such a solver usually depends on the way a problem is represented. However, the translation to processable QBF encodings is in general not unique and may either introduce variables and clauses not relevant for the solving process or blur information which could be beneficial for the solving process. To deal with both of these issues, preprocessors have been introduced which rewrite a given QBF before it is passed to a solver.
In this paper, we present novel preprocessing methods for QBF based on blocked clause elimination (BCE), a technique successfully applied in SAT. Quantified blocked clause elimination (QBCE) allows to simulate various structural preprocessing techniques as BCE in SAT. We have implemented QBCE and extensions of QBCE in the preprocessor bloqqer. In our experiments we show that preprocessing with QBCE reduces formulas substantially and allows us to solve considerable more instances than the previous state-of-the-art.

1 Introduction

Preprocessing in the context of SAT solving refers to techniques applied on a propositional formula before the actual solving process is started [1,2,6,11]. The intention behind preprocessing is to simplify the formula in such a way that solving time spent on the preprocessed formula together with the time spent on the preprocessing is less than the solving time of the original formula. The term "simplification" denotes the reduction of the formula in size as well as modifications extending the formula. On the one hand, state-of-the-art preprocessors apply various techniques which result in safe substitution and removal of single variable occurrences or even of complete clauses. On the other hand, in some situations preprocessors are able to identify information which can be used within the solving process and, consequently, might actually increase formula size. Especially if only formulas in conjunctive normal form (CNF) are considered, the goal is to reconstruct structural information which has been blurred

* The 1st author is financially supported by the Austrian Science Foundation (FWF) NFN Grant S11408-N23 (RiSE).

by the normal form transformation. The extra effort spent on the preprocessing step is justified by the assumption that the costs of one single application of the implemented techniques is negligible, whereas a dynamic application within the solving process would be too expensive.

Preprocessing has been successfully applied to propositional formulas [1,2,6,11]. As SAT is the prototypical problem for NP, the problem of evaluating QBF (QSAT) is prototypical for PSPACE, offering a powerful framework for important application in artificial intelligence, knowledge representation, verification, and synthesis.

During the last decade, much effort has been spent in the development of efficient QBF solvers, but despite several success stories, the achievements are far from the progress which has been made in SAT solving, particularly when it comes to real applications. Motivated by the impact of preprocessing in SAT, some preprocessors for QBF have recently been presented which implement various kinds of preprocessing techniques and which proved to be advantageous for the evaluation of representative QBF benchmark sets [4,8,15,17,19]. Based on these experiments, we present the preprocessor bloqqer which incorporates several well established preprocessing techniques like (self-subsuming) resolution and expansion-based variable elimination as well as preprocessing techniques novel to QBF based on blocked clause elimination.

This paper is structured as follows. First, we introduce the basic terminology and the concepts used within this paper in Section 2 and we shortly review current preprocessing techniques for QBF in Section 3. In Section 4, we present novel preprocessing techniques based on blocked clause elimination for QBF and discuss the interrelationship with other approaches. We present our implementation bloqqer in Section 5 and compare and discuss the results obtained from various experiments. Finally, we conclude with an outlook to our future work.

2 Preliminaries

A QBF ϕ in prenex conjunctive normal form (PCNF) defined over the set of propositional variables V is an expression of the form $\phi = S_1 \dots S_k \psi$ where ψ is called the *matrix* and $S_1 \dots S_k$ is called the *quantifier prefix*.

The *matrix* is a propositional formula in conjunctive normal form, i.e., $\psi = C_1 \wedge \dots \wedge C_n$ where $C_1, \dots, C_n$ are clauses. A *clause* is a disjunction of literals, i.e., $C_i = l_1 \vee \dots \vee l_m$. A *literal* l is either a variable x or a negated variable $\neg x$ with $x \in V$. The function $\mathsf{var}(l)$ returns x if l is of the form x or $\neg x$. If $l = x$ then $\bar{l} = \neg x$ else $\bar{l} = x$. If convenient, we consider the matrix as a set of clauses and a clause as a set of literals. Consequently, we also write $\{C_1, \dots, C_n\}$ for ψ and $\{l_1, \dots, l_m\}$ for a clause C. A clause C is tautological if $l, \bar{l} \in C$. If not stated otherwise, we assume clauses to be non-tautological in the following.

The *quantifier prefix* $S_1 \dots S_k$ is an ordered partition of the variables V into *scopes* S_i. The size of a quantifier prefix $|S_1 \dots S_k|$ is given by $|S_1| + \dots + |S_k|$. The function $\mathsf{quant}(S)$ associates either an *existential quantifier* ($\mathsf{quant}(S) = \exists$) or a *universal quantifier* ($\mathsf{quant}(S) = \forall$) with each scope S. For scopes S_i and S_{i+1},

$\mathsf{quant}(S_i) \neq \mathsf{quant}(S_{i+1})$. Alternatively, we also write $Qx_1, \ldots, x_n$ for a scope $S = \{x_1, \ldots, x_n\}$ with $\mathsf{quant}(S) = Q, Q \in \{\forall, \exists\}$. For a clause C, its existential and its universal literals are defined by $L_Q(C) = \{l \in C \mid \mathsf{quant}(l) = Q\}$ with $Q \in \{\forall, \exists\}$. For a literal l with $\mathsf{var}(\mathrm{l}) \in S$, $\mathsf{quant}(l) = \mathsf{quant}(S)$ denotes the type of l. For literals l, k with $\mathsf{var}(\mathrm{l}) \in S_i$ and $\mathsf{var}(\mathrm{k}) \in S_j$, $l \leq k$ if $i \leq j$. The indices i and, resp., j are called the *level* of l and, resp., of k.

Let l be a literal, then $\phi[l]$ denotes the QBF which is obtained by deleting the clauses C with $l \in C$, by removing each occurrence of $\bar{l}$, and by substituting the scope S_i with $\mathsf{var}(\mathrm{l}) \in S_i$ by $S_i \backslash \{\mathsf{var}(\mathrm{l})\}$. The truth value of a QBF ϕ is recursively defined as follows.

- If $\psi = \emptyset$ then ϕ is true, if $\emptyset \in \psi$ then ϕ is false.
- If $\mathsf{quant}(S_1) = \forall$ (resp., $\mathsf{quant}(S_1) = \exists$) and $x \in S_1$, then ϕ is true iff $\phi[x]$ and (resp., or) $\phi[\neg x]$ is true.

The *Q-resolvent* $C_1 \otimes C_2$ of two clauses C_1 and C_2 with $l \in C_1$, $\bar{l} \in C_2$, and $\mathsf{quant}(l) = \exists$ is defined as $C_1' \backslash \{l\} \cup C_2' \backslash \{\bar{l}\}$ where

$$C_i' = C_i \backslash \{k \mid k \in C_i, \mathsf{quant}(k) = \forall, \forall k' \in C_i \text{ with } \mathsf{quant}(k') = \exists \ : k > k'\}.$$

The literal l is called *pivot element*. The removal of a universally quantified literal k from a clause which does not contain any existentially quantified variables with a higher level than k is also referred to as *forall reduction*. The construction rule of Q-resolvents enhanced with the forall reduction rule form the quantified resolution calculus which is sound and complete for QBF [5].

A literal l is called *pure* in a QBF $\phi = S_1 \ldots S_n \psi$ if $l \in \bigcup_{C \in \psi}$ and $\bar{l} \notin \bigcup_{C \in \psi}$. Then ϕ is equivalent to $\phi[l]$ if $\mathsf{quant}(l) = \exists$ and equivalent to $\phi[\bar{l}]$ if $\mathsf{quant}(l) = \forall$. An existentially quantified literal l is called *unit* in ϕ if $\{l, k_1, \ldots, k_m\} \in \psi$ with $\mathsf{quant}(k_i) = \forall$ and $l < k_i$. If l is unit in ϕ then ϕ is equivalent to $\phi[l]$. If ϕ contains a non-tautological clause with universally quantified literals only, then ϕ is false.

3 Preprocessing Techniques for QBF

Recently, several preprocessors for QBF have been proposed which implement different techniques to prepare formulas in PCNF for the actual solving process. Only the preprocessor PReDom [15] operates on a circuit-based representation with the aim to identify structural dominators. A node n_1 of a circuit dominates a node n_2 if every path starting from n_1 contains n_2. PReDom reduces dominated subcircuits such that the truth value of the original QBF is preserved.

The preprocessor realized within the logic framework proverbox [4], the preprocessors sQueezeBF [8] and prequel [19], as well as the approach presented in [17], all process formulas in PCNF encoded in the QDIMACS format and implement—among other techniques—unit propagation, forall reduction, and some kind of equality detection and substitution. The former three systems additionally implement pure literal detection and subsumption as well. When these

basic simplification techniques are not applicable anymore, the preprocessor presented in [17] performs constant detection. If a literal is implied by the matrix of the given QBF, then it may be added as unit clause. For testing whether this implication holds, a SAT solver is applied. The preprocessor prequel [19] uses hyper binary resolution which—if applicable—resolves a clause of arbitrary length with binary clauses, until the resolvent is binary or even a unit clause. In the preprocessor built upon the framework proverbox [4], universally quantified variables are selectively expanded and consequently eliminated. Furthermore, Q-resolution is integrated in order to reduce duplications introduced by universal expansion. The preprocessor sQueezeBF [8] also implements Q-resolution (cf. [5]), but with the goal to remove existentially quantified variables. Similar to the approach implemented in the QBF solver Quantor [3], an existentially quantified variable may be eliminated from a formula, if all possible resolvents over this variable are added to the formula instead.

All of the previously presented techniques are applied with the intention to eliminate variables, literals, and/or clauses. Only hyper binary resolution is used to uncover hidden information beneficial for the solving process. In the preprocessor sQueezeBF [8], which we consider as the previous state-of-the-art, two rewriting rules have been introduced. These rewrite rules which are applied if the substitution of an equivalence would negatively affect the size of the QBF are defined as follows.

Lemma 1 (RW1: Removed Implication, see [8]).
Let $\phi = S_1 \dots S_n((l \vee \alpha) \wedge (l \Leftrightarrow \gamma) \wedge \psi)$ such that (i) $\mathsf{quant}(l) = \exists$, (ii) l does occur neither in ψ, α, nor γ, and (iii) $k \leq l$ forall literals k occurring in γ. Then ϕ is equivalent to the formula $S_1 \dots S_n((l \vee \alpha) \wedge (l \Rightarrow \gamma) \wedge \psi)$.

This rewriting may be beneficial with respect to two aspects: (i) the formula becomes smaller, and (ii) when α becomes true, l is pure. In the next section, we will argue that quantified blocked clause elimination is able to simulate RW1. In the case that l occurs in two polarities, the following rewrite rule may be applied.

Lemma 2 (RW2: Splitted Equivalence, see [8]).
Let $\phi = S_1 \dots S_i \dots S_n((l \vee \alpha) \wedge (\overline{l} \vee \beta) \wedge (l \Leftrightarrow \gamma) \wedge \psi)$ such that (i) $\mathsf{quant}(l) = \exists$, $\mathsf{var}(l) \in S_i$, (ii) l does occur neither in ψ, α, β, nor γ, and (iii) $k \leq l$ forall literals k occurring in γ. Then ϕ is equivalent to the formula

$$S_1 \dots S'_i \dots S_n((l \vee \alpha) \wedge (l' \vee \beta) \wedge (l \Rightarrow \gamma) \wedge (l' \Rightarrow \overline{\gamma}) \wedge \psi)$$

where l' is a fresh variable with $\mathsf{quant}(l') = \exists$ and the same polarity as l and $S'_i = S_i \cup \{\mathsf{var}(l')\}$.

The application of this rewrite rule does not reduce formula size, but it may trigger pure literal elimination. If α (resp., β) becomes true, then $(l \Rightarrow \gamma)$ (resp., $(\gamma \Rightarrow \overline{l'})$) may be removed. At first sight, this rewrite rule affects unit literal propagation adversely, because the original QBF ϕ reduces to $\beta \wedge \gamma \wedge \psi$ if α is

false and to $\alpha \wedge \gamma \wedge \psi$ if β is false. Having the rewrite rule applied, we only obtain $(l' \vee \beta) \wedge \gamma \wedge (l' \Rightarrow \overline{\gamma}) \wedge \psi$ in the one case and $(l \vee \alpha) \wedge \neg\gamma \wedge (l \Rightarrow \gamma) \wedge \psi$ in the other case. To overcome this limitation, the *efficiency clause* $(\bar{l} \vee \overline{l'})$ which is obviously entailed by $(l \Rightarrow \gamma) \wedge (l' \Rightarrow \overline{\gamma})$ has to be added to the formula when the rewrite rule is applied.

4 Quantified Blocked Clause Elimination

Originally introduced for restricting worst-case upper bounds for SAT-algorithms [12], *blocked clauses* have proven to be effective for preprocessing in SAT [11,16], because blocked clauses may be eliminated while preserving satisfiability. In the following, we generalize the notion of blocked clauses and blocked clause elimination (BCE) for QBF. We prove that also in the case of QBF under certain restrictions blocked clauses may be omitted. Subsequently, we shortly discuss the integration of quantified BCE (QBCE) with other preprocessing techniques, before we propose extensions for QBCE.

4.1 Definition

Within a resolution proof, blocked clauses of a formula only generate clauses which are tautological. Consequently, blocked clauses may be removed without changing the truth value of a formula. In the following we describe the characteristics which allow the syntactical identification of blocked clauses in QBF.

Definition 1 (Quantified Blocking Literal). *A literal l with $\mathsf{quant}(l) = \exists$ in a clause $C \in \psi$ of a QBF $\phi = S_1 \dots S_n \psi$ is called a* quantified blocking literal *if forall $C' \in \psi$ with $\bar{l} \in C'$, a literal k with $k \leq l$ exists such that $k, \bar{k} \in C \otimes C'$.*

Definition 2 (Quantified Blocked Clause). *A clause is* quantified blocked *if it contains a quantified blocking literal.*

Example 1. Both clauses in $\forall x \exists y((x \vee \neg y) \wedge (\neg x \vee y))$ are quantified blocked clauses, whereas none of the clauses in $\exists x \forall y((x \vee \neg y) \wedge (\neg x \vee y))$ is a quantified blocked clause.

As the following theorem shows, quantified blocked clauses contain redundant information only, and may therefore be removed from the formula.

Theorem 1 (Quantified Blocked Clause Elimination (QBCE)). *Let $\phi = S_1 \dots S_n(\psi \cup C)$ be a QBF and let C be a quantified blocked clause in ϕ with blocking literal l. It holds that $\phi \Leftrightarrow S_1 \dots S_n \psi$.*

Proof. Let C be a quantified blocked clause with the quantified blocking literal l with $\mathsf{var}(l) \in S_i$, $i \leq n$. The direction $\phi \Rightarrow S_1 \dots S_n \psi$ trivially holds. We show $S_1 \dots S_n \psi \Rightarrow \phi$ by induction over $q = |S_1 \dots S_{i-1}|$. W.l.o.g. assume $i = n$.

In the base case, we have $q = 0$, i.e., $\mathsf{var}(l) \in S_1$ with $\mathsf{quant}(S_1) = \exists$. The same argument as in SAT applies: Let σ be a satisfying assignment for ψ, i.e.,

for each $C' \in \psi$ there exists a literal l' such that $\sigma(l') = \top$. If σ satisfies C, the implication $S_1\psi \Rightarrow \phi$ holds, otherwise we construct a satisfying assignment σ' for $\psi \cup C$ as follows. Let $\sigma'(l') = \sigma(l')$ for $l' \neq l$ and $\sigma'(l) = \top$. σ' satisfies not only C but also all other clauses $C' \in \psi$. If $\bar{l} \in C'$, there exists a literal $k \neq l$ such that $k \in C$ and $\bar{k} \in C'$, with $\sigma(k) = \sigma(C) = \sigma'(k) = \bot$ and thus $\sigma'(C') = \sigma'(\bar{k}) = \top$ [12]. Note that $k \in S_1$ due to the restriction $k \leq l$.

For the induction step, assume $q > 0$. Let h be a literal with $\mathsf{var}(h) = y$ and $y \in S_1$. Note that $\mathsf{var}(l) \neq y$. We show that $S_1 \backslash \{y\} \ldots S_n\psi[h] \Rightarrow \phi[h]$. The rest follows from lifting the implication over the conjunction that defines the semantics of universal quantification if $\mathsf{quant}(S_1) = \forall$ and respectively over the disjunction that defines the semantics of the existential quantification if $\mathsf{quant}(S_1) = \exists$. Three cases have to be considered for showing that $C[h]$ is a blocking clause or removed in $\phi[h]$.

1. $h \in C$. Then C is removed from $\phi[h]$.
2. $h \notin C$ and $\bar{h} \notin C$. Consequently, $C[h] = C$. Furthermore, C is still a quantified blocked clause in $\phi[h]$, since h was not used to make a resolvent on l tautological. Then the induction hypothesis is applicable.
3. $\bar{h} \in C$. Consequently, $C[h] = C \backslash \{\bar{h}\}$ which is a quantified blocked clause in $\phi[h]$, because each clause C' with $h, \bar{h} \in C \otimes C'$ is removed from $\phi[h]$ and other clauses C' with $k, \bar{k} \in C \otimes C'$ and $y \neq \mathsf{var}(k)$ still produce tautological resolvents with C on l. Note $l \in C[h]$ since $l \neq \bar{h}$.

For a QBF ϕ in PCNF, quantified blocked clause elimination repeats the removal of quantified blocked clauses from ϕ until fixpoint. The resulting QBF is denoted by $\mathsf{QBCE}(\phi)$.

Theorem 2. *The application of $\mathsf{QBCE}(\phi)$ on a QBF ϕ is confluent.*

Proof. The argument is similar as for propositional logic (cf. [11]).

Note that for the soundness of quantified blocked clause elimination for QBF as stated in Theorem 1, the level of the blocking literal must be equal or higher than the level of the literal making the resolvent tautological as the following example illustrates.

Example 2. As we have seen in Example 1, the QBF

$$\exists x \forall y((x \vee \neg y) \wedge (\neg x \vee y))$$

contains no blocking clause. If we loosen the criterion and do not consider the quantifier levels of the variables, then all clauses become blocking clauses and according to Theorem 1, they may be removed immediately. Consequently, the formula evaluates to true, what is in contrast to the formula's original truth value. In this formula, a contradiction is directly derivable if forall reduction is applied. An extended example, where forall reduction is not applicable, is given by the formula

$$\exists x \forall y \exists z((x \vee \neg z) \wedge (\neg x \vee z) \wedge (y \vee \neg z) \wedge (\neg y \vee z))$$

which contains an additional existential variable z which is equivalent to y. The variable z prohibits the application of the forall reduction rule. Furthermore, the first two clauses are not quantified blocked on x and $\neg x$, respectively, because $z < x$ does not hold. If they are removed, the formula evaluates to true.

4.2 Discussion

Quantified blocked clauses as defined above may be eliminated from a formula without changing its truth value, because they contain redundant information only. Hence, quantified blocked clause elimination is applied in order to remove clauses from a QBF which may result in a reduction in the number of variables occurring in the formula too. The following properties established for SAT [11,12], also hold for QBF. For the sake of compactness, we omit the prefix "quantified" if no confusion arises.

1. Formulas which are smaller with respect to their number of clauses potentially contain more blocked clauses. If the matrix of a QBF ϕ_1 is a subset of the matrix of the QBF ϕ_2 then there might be clauses which are blocked in ϕ_1, but not in ϕ_2. If there is a clause C which is blocked in ϕ_2, but not in ϕ_1, then $C \notin \phi_1$.
2. From the statement above, it follows immediately that QBCE has a unique fixpoint. If a clause C is blocked in a QBF ϕ, then any clause C' with $C \neq C'$ blocked in ϕ is also blocked in $\phi \backslash \{C\}$.
3. If a clause C is subsumed by a blocked clause C', i.e., $C' \subseteq C$, then C is also a blocked clause. Obviously, the other direction does not hold.
4. Clauses containing a pure literal are blocked. The pure literal is the blocking literal. In fact, QBCE may be considered as a generalization of pure literal elimination rule.
5. If the clauses $C_1 \dots C_n$ are the only clauses of a QBF ϕ which contain the literal l, then a clause C with $\bar{l} \in C$ is blocked if for each clause C_i, the clause C contains a literal k_i with $\bar{k}_i \in C_i$ and $k_i < l$. In particular, if a QBF ϕ contains an equivalence of the form $(l, \bar{k}_1, \dots, \bar{k}_n), (\bar{l}, k_1), \dots, (\bar{l}, k_n)$ and l occurs in no other than these clauses, then the equivalence may be removed due to QBCE.

The fifth property indicates that QBCE may be used to eliminate equivalences under certain conditions. In fact, like BCE in SAT [11], QBCE is able to achieve similar simplifications on a formula in PCNF as other techniques directly applied on the original encoding with more structural information (e.g., a circuit-based representation) before the transformation to normal form is performed. Therefore, we show the close connection between QBCE and the rewriting rules RW1 and RW2 given in Definition 1 and Definition 2. As argued by [8], the application of equivalence rewriting together with the application of the pure literal elimination rule show a similar effect than *don't care propagation* performed on the original, non-CNF formula.

Theorem 3. *QBCE subsumes RW1 given in Definition 1.*

Proof. First we argue, that with QBCE the same effect may be obtained as with RW1, and then we provide a QBF which may be simplified by the application of QBCE, but not by the application of RW1.

1. Whenever RW1 is applicable, also QBCE is applicable. Recall that RW1 substitutes the matrix of a QBF of the form $(l \vee \alpha) \wedge (l \Leftrightarrow \gamma) \wedge \psi$ by $(l \vee \alpha) \wedge (l \Rightarrow \gamma) \wedge \psi$ with the restriction that l does occur neither in α, γ, nor ψ and that all literals of γ have a lower level than l. The rule RW1 therefore removes clauses of the form $\bar{\gamma} \vee l$. Since $\bar{l}$ occurs only in the clauses representing the equivalence, the clauses of $\bar{\gamma} \vee l$ are blocked and may be omitted.
2. In some situations, QBCE is applicable, but not RW1. For example, the QBF

$$\forall y \exists x \exists z((x \vee z) \wedge (\bar{x} \vee \bar{z}) \wedge (z \vee \bar{y} \vee x) \wedge (\bar{z} \vee y \vee \bar{x}))$$

   is reducible by QBCE, but not by RW1. Only if we had applied subsumption first, also RW1 would have been applicable.

Consequently, the application of QBCE has at least the same effects as the application of RW1:

- The number of clauses is reduced. If γ is a disjunction of n literals then n binary clauses are blocked, if γ is a conjunction of n literals then a clause of size $n + 1$ is blocked.
- The application of QBCE may directly trigger the application of other pruning techniques. For example, if a clause $(l \vee \bar{x})$ is removed then x might become pure and, depending on the quantification type of x, either all clauses containing x or all occurrences of x may be removed immediately.
- Other pruning techniques may become applicable during preprocessing or even during the solving process. For example, if QBCE has been applied and α later becomes true, then the literal l becomes pure.

The application of RW1 enables similar optimizations achieved by using the Plaisted-Greenbaum transformation [18] instead of the Tseitin transformation [20]. In this case, the subformula which shall be abbreviated by a freshly introduced variable occurs in one polarity only, therefore one direction of the implication may be omitted. When the subformula occurs in both polarities, then it is possible to treat positive and negative occurrences independently and to introduce two new variables.

The retrospective application of this approach is covered by RW2. The rule RW2 provides no direct simplifications itself and even introduces an extra variable, but after its application, it becomes more likely that (i) more variables become pure and (ii) RW1 or QBCE become applicable. In fact, if RW2 is used during preprocessing, then there exist situations, where the pure literal elimination rule, which is implemented by most state-of-the-art QBF solvers, performs reductions. The same effect can be achieved, if QBCE is applied dynamically during the solving process without applying RW2. Hence, no new variables have to be introduced to achieve the same effects. Consider the following formula.

Let $\psi = ((l \vee \alpha) \wedge (\bar{l} \vee \beta) \wedge (l \Leftrightarrow \gamma) \wedge \delta)$ be the matrix of a QBF and let $\psi' = ((l \vee \alpha) \wedge (l' \vee \beta) \wedge (l \Rightarrow \gamma) \wedge (\gamma \Rightarrow \bar{l'}) \wedge \delta)$ be the formula obtained after the application of RW2. Then the benefits of RW2 in combination with the pure literal elimination rule identified by [11], can also be obtained by QBCE.

1. If α (resp. β) becomes true in ψ', then l (resp. l') becomes pure and ψ' may be simplified to $((l' \vee \beta) \wedge (\gamma \Rightarrow \bar{l'}) \wedge \delta)$ (resp. $((l \vee \alpha \wedge (l \Rightarrow \gamma)) \wedge \delta)$). If α (resp. β) becomes true in ψ then the same reductions may be obtained by the application of QBCE on ψ.
2. If both α and β become true, then ψ' may be reduced to δ because l and l' are pure. Also ψ may be reduced to γ by the application of QBCE if α and β become true.

When combining RW2 and QBCE, the two techniques potentially influence each other as follows:

- The application of RW2 preserves existing blocked clauses and might even uncover new blocked clauses.
- The application of QBCE might limit or even inhibit the application of RW2, namely when one of the clauses forming the equivalence is removed.

These observations indicate that it might be advantageous to apply RW2 before QBCE. As we will see in the next section, our experiments confirm this conjecture.

4.3 Extensions

For SAT, several extensions of BCE and related clause elimination procedures have been proposed. Based on the adoption of BCE for QSAT, also these extensions may be applied for preprocessing QBF. The goal is to add literals to clauses for making them either tautological or for triggering the application of blocked clause elimination. For SAT, covered clauses have been introduced in [9,10]. In the following, we leverage covered clauses from SAT to QBF.

Quantified Blocked Covered Clause Elimination. Covered clauses are clauses which are blocked or tautological when they are enriched with literals contained in any resolvent with pivot element l, the covering literal.

Definition 3 (Quantified Covered Literal). *Let the set of* resolution candidates $R_\phi(C, l) = \{C' \backslash \{\bar{l}\} \mid C' \in \psi, \bar{l} \in C', \nexists k : \{k, \bar{k}\} \subseteq C \otimes C'\}$ *where* ϕ *is a QBF with matrix* ψ, $C \in \psi$, $l \in C$. *The set of* quantified covered literals $\mathcal{C}_\phi(C, l)$ *with respect to a clause* C *and a literal* l *is given by the intersection of the resolution candidates*

$$\mathcal{C}_\phi(C, l) = \bigcap \{C'' \mid C' \in R_\phi(C, l), C'' \subseteq C', \forall k \in C'' : k \leq l\}.$$

A literal l is called *covering literal* if $\mathcal{C}_\phi(C, l) \neq \emptyset$, i.e., l covers the literals in $\mathcal{C}_\phi(C, l)$.

Lemma 3. *The replacement of a clause C in a QBF ϕ by $C \cup \mathcal{C}_\phi(C, l)$ preserves unsatisfiability.*

Proof. Analogous to the proof of Theorem 1.

In the following, $\mathcal{C}_\phi(C)$ denotes the clause C extended with all quantified covered literals, i.e., for all $l \in \mathcal{C}_\phi(C)$ it holds that $\mathcal{C}_\phi(C, l) \subseteq \mathcal{C}_\phi(C)$.

Lemma 4 (Quantified Covered Literal Addition). *The replacement of a clause C in a QBF ϕ by $\mathcal{C}_\phi(C)$ preserves unsatisfiability.*

Proof. Iterative application of Lemma 3.

Definition 4 (Quantified Covered Clause). *A clause C in a QBF ϕ is* covered *if $\mathcal{C}_\phi(C)$ is tautological or blocked w.r.t. ϕ.*

Theorem 4 (Quantified Covered Clause Elimination). *The removal of a covered clause preserves unsatisfiability.*

Proof. According to Lemma 4, each clause may be replaced by the clause $\mathcal{C}_\phi(C)$. If this clause is blocked, it may be removed according to Theorem 1. If it is tautological, it may be removed due to standard rewriting rules.

Example 3. In the QBF $\forall a, b, c\, \exists x, y((x \vee \neg a) \wedge (\neg x \vee y \vee b) \wedge (\neg x \vee y \vee c) \wedge (\neg y \vee a))$ the literal x of the clause $(x \vee \neg a)$ covers the literal y. We therefore may replace this clause by $(x \vee \neg a \vee y)$ which is blocked, and, consequently, can be eliminated.

As discussed in [10] covered clause elimination is confluent and more effective than QBCE.

Quantified Hidden Blocked Clause/Tautology Elimination. Quantified hidden blocked clauses and quantified hidden tautologies are uncovered by the addition of literals which are derivable from implications contained within a QBF. For SAT, these techniques have been presented in [9].

Definition 5 (Quantified Hidden Literal). *Let ϕ be a QBF with matrix ψ. A literal l is called* quantified hidden literal *w.r.t. a clause $C \in \psi$ if ψ contains a clause $(l_1, \ldots, l_n, \bar{l})$ with $l_i \leq l$ and $l_1, \ldots, l_n \in C$.*

Lemma 5. *The replacement of a clause C in a QBF ϕ with $C \cup \{l\}$ preserves unsatisfiability if l is a quantified hidden literal with respect to C.*

Proof. Analogous to the proof of Theorem 1.

Theorem 5. *Let C' be a clause obtained from a clause $C \in \phi$ by adding hidden literals. If C' is blocked or tautological, the removal of C from ϕ preserves unsatisfiability.*

Proof. Due to the (iterative) application of Lemma 5, C may be replaced by C' in ψ. If C' is blocked, it may be removed according to Theorem 1. If C' is tautological, it may be replaced due to standard rewriting rules.

Table 1. Impact of preprocessors

Family	no preprocessing			bloqqer			sQueezeBF		
	V	C	A	%V	%C	*A*	%V	%C	*A*
Abduction	1474	3435	2	-38	-22	-2	-7	-20	-1
Adder	3527	4405	3	-67	266	-1	-26	-37	0
blackbox*	11437	27819	153	-95	-77	-145	4	-81	-145
Blocks	518	6756	2	-44	-47	-1	7	-48	0
BMC	265932	680732	2	-98	-92	-1	-78	-95	0
Chain	3290	19663	2	-100	-100	-2	-100	-100	-2
circuits	1400	1920	2	-61	137	0	0	-40	0
confplan	1285	47890	2	-56	-6	-1	49	-70	0
Connect4	218810	93504	46	-99	-82	-32	-89	-45	-5
Counter	1951	5169	28	-80	-61	-22	1	-1	0
Debug	159502	1036810	2	-3	-15	0	-63	-52	0
evadepursue	7666	74014	9	-40	-54	0	-2	-51	0
FPGA*	65	433	2	332	828	0	1	-5	0
Impl	74	146	36	-100	-100	-36	-100	-100	-36
jmc_quant	508	995	4	25	321	0	0	-68	0
mqm	1724	5060	18	-50	10	-2	0	-14	0
pan	1847	10999	32	-91	-87	-31	38	-40	-11
Rintanen	1871	178750	2	-8	-1	0	7	0	0
Sakallah	44526	29282	2	-81	-50	-1	-79	-76	-1
Scholl-Becker	2758	7712	5	-83	-30	1	34	-43	-1
SortNet	1491	4972	2	-70	-10	0	21	-30	0
SzymanskiP	148973	168917	2	-100	-100	-2	-7	-70	0
tipdiam	5397	15428	2	-91	-79	-1	4	-78	0
tipfixpoint	9103	26407	2	-95	-88	-1	7	-71	0
Toilet	365	3129	2	-52	-100	-2	30	-44	-2
VonNeumann	1040116	1523169	2	-100	-100	-2	-100	-100	-2

5 Experimental Evaluation

Together with variable expansion, equivalence replacement, pure and unit literal elimination as well as with subsumption (cf. Section 3), the previously presented techniques are implemented in the preprocessor bloqqer[1]. To test our implementation, we applied bloqqer on the benchmark set used at the QBF Competition 2010[2] which consists of 568 formulas. For the sake of compactness, we aggregated the 36 families to 26 sets. All experiments were performed on 2.83 GHz Intel Core 2 Quad machines each equipped with 8 GB memory and running

[1] available at `http://fmv.jku.at/bloqqer`
[2] available at `http://www.qbflib.org`

Table 2. Experiments with various solvers

		# formulas				runtime (sec)		
	preprocessor	SOLVED	SAT	UNSAT	UNKN	$\sum$ (10^3)	AVG	MEDIAN
DepQBF	sQueezeBF/bloqqer	482	234	248	86	102	180	5
	bloqqer	467	224	243	101	112	198	5
	bloqqer/ sQueezeBF	452	213	239	116	147	258	19
	sQueezeBF	435	201	234	133	131	231	6
	no preprocessing	373	167	206	195	189	332	26
QuBE	sQueezeBF/bloqqer	454	207	247	114	129	227	7
	bloqqer	444	200	244	124	139	246	5
	bloqqer/sQueezeBF	421	183	238	147	174	307	27
	sQueezeBF	406	181	225	162	177	313	31
	no preprocessing	332	135	197	236	242	426	258
Nenofex	bloqqer/sQueezeBF	271	134	137	297	273	482	76
	sQueezeBF/bloqqer	270	136	134	298	277	488	31
	bloqqer	268	132	136	300	276	487	23
	sQueezeBF	246	122	124	322	297	524	88
	no preprocessing	221	107	114	347	319	561	113
Quantor	bloqqer	288	145	143	280	266	468	34
	sQueezeBF/bloqqer	285	147	138	283	268	472	39
	bloqqer/sQueezeBF	270	131	139	298	276	486	34
	sQueezeBF	222	106	116	346	318	561	49
	no preprocessing	206	100	106	362	333	587	38

Ubuntu 9.04. The time limit and memory limit were set to 900 seconds and 7 GB, respectively. Time spent on the preprocessing is included in the time limit. If the preprocessor has not terminated after 900 seconds, the preprocessing is aborted and the formula is considered to be unsolved. In the following evaluation, sQueezeBF [8] serves as reference preprocessor, because sQueezeBF incorporates similar features as bloqqer except that bloqqer implements QBCE and sQueezeBF implements RW1 and RW2. Furthermore, sQueezeBF was shown to be the most effective state-of-the art preprocessor in [8].

First, we evaluated the impact of bloqqer on the formula size in terms of number of variables (V), number of clauses (C), and number of quantifier alternations in the prefix (A). Table 1 shows the concrete results for the different formula sets. The first column (no preprocessing) contains the average values for the number of variables, number of clauses, and quantifier alternations of the unpreprocessed formulas. The second column (bloqqer) indicates the effects of applying bloqqer on these formulas. For the subcolumns V and C the aver-

Table 3. # formulas (# satisfiable formulas) solved by DepQBF

Family (set size)	sQueezeBF/ bloqqer	bloqqer/ sQueezeBF	bloqqer	sQueezeBF	no preproc.
Abduction (52)	48 (29)	49 (30)	49 (30)	48 (29)	50 (31)
Adder (15)	3 (3)	4 (3)	4 (3)	1 (1)	0 (0)
blackbox∗ (61)	52 (2)	45 (2)	46 (2)	55 (2)	43 (0)
Blocks (5)	4 (2)	5 (2)	5 (2)	5 (2)	4 (1)
BMC (18)	14 (5)	16 (6)	15 (5)	13 (5)	12 (5)
Chain (1)	1 (1)	1 (1)	1 (1)	1 (1)	0 (0)
circuits (3)	2 (2)	2 (2)	2 (2)	2 (2)	2 (2)
conf_planning (15)	5 (4)	5 (4)	5 (4)	5 (4)	4 (3)
Connect4 (11)	8 (0)	8 (0)	8 (0)	8 (0)	8 (0)
Counter (4)	3 (3)	3 (3)	4 (4)	2 (2)	2 (2)
Debug (5)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
evader-pursuer (22)	17 (7)	11 (2)	11 (3)	10 (3)	10 (3)
FPGA∗ (3)	3 (1)	3 (1)	3 (1)	3 (1)	3 (1)
Impl (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)
jmc_quant (3)	3 (2)	3 (2)	3 (2)	3 (2)	0 (0)
mqm (136)	136 (66)	123 (58)	136 (66)	136 (66)	136 (66)
pan (80)	75 (41)	76 (41)	76 (41)	44 (22)	26 (15)
Rintanen (1)	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)
Sakallah (19)	11 (10)	13 (11)	15 (13)	10 (9)	0 (0)
Scholl-Becker (24)	15 (5)	14 (4)	14 (4)	13 (5)	11 (4)
Sorting_networks (6)	3 (1)	3 (1)	2 (1)	3 (1)	6 (4)
SzymanskiP (2)	2 (0)	2 (0)	2 (0)	2 (0)	0 (0)
tipdiam (14)	13 (12)	8 (8)	8 (8)	11 (10)	3 (3)
tipfixpoint (24)	19 (10)	13 (4)	13 (4)	16 (7)	9 (0)
Toilet (41)	41 (27)	41 (27)	41 (27)	40 (26)	40 (26)
VonNeumann (2)	2 (0)	2 (0)	2 (0)	2 (0)	2 (0)

age increase/decrease in percent w.r.t. the original formulas is shown, whereas for A the number of additional quantifier alternations is given. The third column shows how the application of sQueezeBF effects the formula size. bloqqer decreases the variable number of all but two formula sets by about 70 percent on average. For 21 formulas sets, we observe a decrease of the clause number of about 60 percent. For the majority of formula sets also a reduction of the quantifier prefix is achieved. This decrease of the formula size may be observed although bloqqer implements preprocessing techniques like variable expansion which adds new clauses and variables. QBCE is performed during the existential variable elimination through resolution. Hidden tautologies and hidden blocked clauses are found in the backward subsumption phase after variable elimination.

Overall, $116 * 10^4$ blocked clauses, $79 * 10^4$ hidden blocked clauses as well as $196 * 10^4$ hidden tautologies have been detected. The size reduction achieved by sQueezeBF is more moderate (cf. third column of Table 1). bloqqer is able to directly evaluate 148 formulas and has no timeouts, sQueezeBF solves only 39 formulas and does not terminate on 14 formulas.

Second, we evaluated the impact of bloqqer on the runtimes of the four state-of-the-art QBF solvers DepQBF [14], QuBE [7], Nenofex [13], and Quantor [3]. For each solver we considered five preprocessing variants: (1) no preprocessing, (2) preprocessing with bloqqer only, (3) preprocessing with sQueezeBF only, (4) preprocessing with the combination bloqqer/sQueezeBF, and (5) preprocessing with the combination sQueezeBF/bloqqer. The results in Table 2 clearly show the positive impact of preprocessing on the number of solved formulas as well as on the runtime. The time values include a penalty of 900 for each unsolved formula. All solvers have in common that the omission of preprocessing negatively influences the solvers. The experiments also indicate that it might be advantageous not to use sQueezeBF alone, but in combination with bloqqer. Which variant is preferable seems to be solver dependent. The best results are obtained with sQueezeBF/bloqqer and the solver DepQBF. In the benchmark set, some families are represented very prominently compared to other families. Table 3 shows the detailed results when the solver DepQBF is applied. We clearly see that the accumulated results are also valid for the various sets.

Finally, we were interested in the impact of the different preprocessing techniques implemented in bloqqer and therefore we ran bloqqer with various options and passed the formals to DepQBF. Recall that with all options enabled, 467 formulas are solved and that with no preprocessing only 373 formulas are solved. If QBCE only is enabled, then still 403 formulas are solved, if all options except the extensions of QBCE are enabled, then 454 formulas are solved. Due to space limitations, we kindly refer to the web page of bloqqer for more details.

6 Conclusion and Future Work

As blocked clause elimination is an effective simplification technique for SAT, quantified blocked clause elimination is an effective simplification technique for QSAT. With QBCE similar effects can be achieved as with simplifications performed on formulas not in PCNF in combination with the polarity-based Plaisted-Greenbaum transformation. We provide an implementation of QBCE and extended variants together with well established simplification techniques in the preprocessor bloqqer. The application of QBCE results in a considerable reduction of formula size and improved solving time.

For future work, we consider to integrate QBCE and its extensions directly into a QBF solver. During the solving process clauses may become blocked which then may be removed immediately. Furthermore, we will investigate if it is possible to loosen the blocking criterion by taking variable dependencies into account.

References

1. F. Bacchus and J. Winter. Effective Preprocessing with Hyper-Resolution and Equality Reduction. In *Proc. of the 6th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT 2003)*, pages 341–355. Springer, 2004.
2. D. Le Berre. Exploiting the Real Power of Unit Propagation Lookahead. *Electronic Notes in Discrete Mathematics*, 9:59–80, 2001.
3. A. Biere. Resolve and Expand. In *Proc. of the 7th Int. Conf. on the Theory and Applications of Satisfiability Testing (SAT 2004)*, pages 59–70. Springer, 2005.
4. U. Bubeck and H. Kleine Büning. Bounded Universal Expansion for Preprocessing QBF. In *Proc. of the 10th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT 2007)*, volume 4501, pages 244–257, 2007.
5. H. Kleine Büning, M. Karpinski, and A. Flögel. Resolution for Quantified Boolean Formulas. *Information and Computation*, 117(1):12–18, 1995.
6. N. Eén and A. Biere. Effective Preprocessing in SAT through Variable and Clause Elimination. In *Proc. of the 8th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT 2005)*, pages 61–75. Springer, 2005.
7. E. Giunchiglia, P. Marin, and M. Narizzano. QuBE 7.0. *JSAT*, 7:83–88, 2010.
8. E. Giunchiglia, P. Marin, and M. Narizzano. sQueezeBF: An Effective Preprocessor for QBFs Based on Equivalence Reasoning. In *Proc. of the 13th Int. Conf. on Theory and Applications of Sat. Testing (SAT 2010)*, pages 85–98. Springer, 2010.
9. M. Heule, M. Järvisalo, and A. Biere. Clause elimination procedures for cnf formulas. In *Proc. of 17th Intl. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-17)*, volume 6397 of *LNCS*, pages 357–371. Springer, 2010.
10. M. Heule, M. Järvisalo, and A. Biere. Covered Clause Elimination. *CoRR*, abs/1011.5202, 2010. Short paper proceedings LPAR-17.
11. M. Järvisalo, A. Biere, and M. Heule. Blocked Clause Elimination. In *Proc. of the 16th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2010)*, pages 129–144. Springer, 2010.
12. O. Kullmann. On a Generalization of Extended Resolution. *Discrete Applied Mathematics*, 96:149–176, 1999.
13. F. Lonsing and A. Biere. Nenofex: Expanding NNF for QBF Solving. In *Proc. of the 11th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT 2008)*, pages 196–210. Springer, 2008.
14. F. Lonsing and A. Biere. DepQBF: A Dependency-Aware QBF Solver (System Description). *JSAT*, 7:71–76, 2010.
15. H. Mangassarian, B. Le, A. Goultiaeva, A.G. Veneris, and F. Bacchus. Leveraging Dominators for Preprocessing QBF. In *Design, Automation and Test in Europe (DATE 2010)*, pages 1695–1700. IEEE, 2010.
16. R. Ostrowski, E. Grgoire, B. Mazure, and L. Saïs. Recovering and Exploiting Structural Knowledge from CNF Formulas. In *Proc. of the 8th Int. Conf. on Princ. and Pract. of Constraint Prog. (CP 2002)*, pages 199–206. Springer, 2006.
17. F. Pigorsch and C. Scholl. An AIG-Based QBF-Solver Using SAT for Preprocessing. In *Proc. of the 47th Design Aut. Conf. (DAC 2010)*, pages 170–175, 2010.
18. David A. Plaisted and Steven Greenbaum. A Structure-Preserving Clause Form Translation. *Journal of Symbolic Computation*, 2(3):293–304, 1986.
19. H. Samulowitz, J. Davies, and F. Bacchus. Preprocessing QBF. In *Proc. of the 11th Int. Conf. on Principles and Practices of Constraint Programming (CP 2006)*, pages 514–529. Springer, 2006.
20. G.S. Tseitin. On the Complexity of Derivation in Propositional Calculus. *Studies in Constructive Mathematics and Mathematical Logic*, 2(115-125):10–13, 1968.