On QBF Proofs and Preprocessing

Mikoláš Janota[1], Radu Grigore[3], and Joao Marques-Silva[1,2]

[1] INESC-ID, Lisbon, Portugal
[2] University College Dublin, Ireland
[3] University of Oxford, UK

Abstract. QBFs (quantified boolean formulas), which are a superset of propositional formulas, provide a canonical representation for PSPACE problems. To overcome the inherent complexity of QBF, significant effort has been invested in developing QBF solvers as well as the underlying proof systems. At the same time, formula preprocessing is crucial for the application of QBF solvers. This paper focuses on a missing link in currently-available technology: How to obtain a certificate (e.g. proof) for a formula that had been preprocessed before it was given to a solver? The paper targets a suite of commonly-used preprocessing techniques and shows how to reconstruct certificates for them. On the negative side, the paper discusses certain limitations of the currently-used proof systems in the light of preprocessing. The presented techniques were implemented and evaluated in the state-of-the-art QBF preprocessor bloqqer.

1 Introduction

Preprocessing [24,47,46,9] and certificate generation [5,6,35,23,39,40] are both active areas of research related to QBF solving. Preprocessing makes it possible to solve many more problem instances. Certification ensures results are correct, and certificates are themselves useful in applications. In this paper we show how to generate certificates while preprocessing is used. Hence, it is now possible to certify the answers for many more problem instances than before.

QBF solvers are practical tools that address the standard PSPACE-complete problem: given a closed QBF, decide whether it is true. In principle, such solvers can be applied to any PSPACE problem, of which there are many; for example, model checking in first-order logic [50], satisfiability of word equations [43], the decision problem of the existential theory of the reals [18], satisfiability for many rank-1 modal logics [48], and so on [23,7,34]. Unlike SAT solvers (for NP problems), QBF solvers are not yet routinely used in practice to solve PSPACE problems: they need to improve.

Fortunately, QBF solvers do improve rapidly [44]. One of the main findings is that a two-phase approach increases considerably the number of instances that can be solved in practice: in the first phase, *preprocessing*, a range of fast techniques is used to simplify the formula; in the second phase, actual solving, a complete search is performed. Another recent improvement is that QBF solvers now produce *certificates*, which include the true/false answer together with a

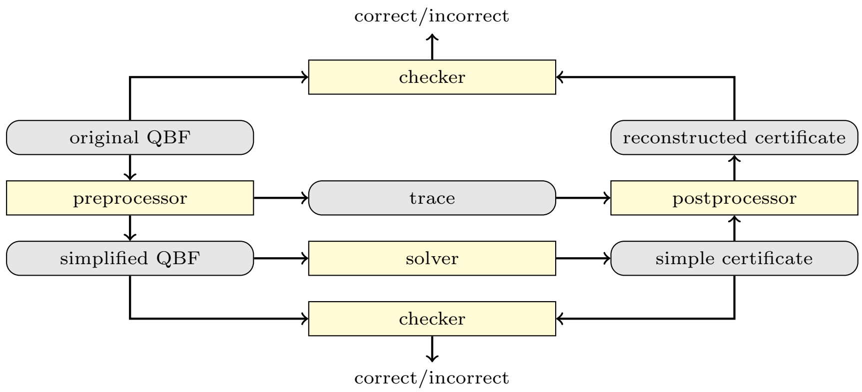


Fig. 1: Architecture

justification for it. Such a justification can be for example in the form of a proof of the given formula. Certificates ensure that answers are correct, and are sometimes necessary for other reasons. For example, certificates are used to suggest repairs in QBF-based diagnosis [25,49,45].

Clearly, both preprocessing and certificate generation are desirable. Alas, no tool-chain supports both preprocessing and certificate generation at the same time. This paper shows how to reconstruct certificates in the presence of a wide range of preprocessing techniques. In our setup (Figure 1), the preprocessor produces a simplified formula together with a *trace*. After solving, we add a postprocessing step, which uses the trace to reconstruct a certificate for the original formula out of a certificate for the simplified formula.

The contributions of this paper are the following:

- a review of many preprocessing techniques used in practice (Section 3)
- a negative result about the reconstruction of term resolution-based certificates (Section 4)
- certificate reconstruction techniques, corresponding to a wide range of formula preprocessing techniques (Section 5)
- an efficient implementation, and its experimental evaluation (Section 7)

2 Preliminaries

A *literal* is a Boolean variable or its negation. For a literal l, we write $\bar{l}$ to denote the literal *complementary* to l, i.e. $\bar{x} = \neg x$ and $\overline{\neg x} = x$; we write $\mathsf{var}(l)$ for x. A *clause* is a disjunction of literals. A formula in *conjunctive normal form* (CNF) is a conjunction of clauses. Whenever convenient, a clause is treated as a set of literals, and a CNF formula as a set of sets of clauses. Dually to a clause, a *term* is a conjunction of literals. A formula in *disjunctive normal form* (DNF) is a conjunction of terms.

For a set of variables X, an *assignment* τ is a function from X to the constants 0 and 1. We say that τ is *complete* for X if the function is total.

Substitutions are denoted as $\psi_1/x_1, \ldots, \psi_n/x_n$, with $x_i \neq x_j$ for $i \neq j$. An application of a substitution is denoted as $\phi[\psi_1/x_1, \ldots, \psi_n/x_n]$ meaning that variables x_i are simultaneously substituted with corresponding formula ψ_i in ϕ.

Quantified Boolean Formulas (QBFs) [14] extend propositional logic with quantifiers that have the standard semantics: $\forall x.\,\Psi$ is satisfied by the same truth assignments as $\Psi[0/x] \wedge \Psi[1/x]$, and $\exists x.\,\Psi$ as $\Psi[0/x] \vee \Psi[1/x]$. Unless specified otherwise, QBFs are in *closed prenex* form, i.e. in the form $\mathcal{Q}_1 x_1 \ldots \mathcal{Q}_k x_k.\,\phi$, where x_i form a nonrepeating sequence of variables and $\mathcal{Q}_i \in \{\exists, \forall\}$; the formula ϕ is over the variables $\{x_1, \ldots, x_k\}$. The propositional part ϕ is called the *matrix* and the rest the *prefix*. If additionally the matrix is in CNF, we say that the formula is in QCNF. A prefix $\mathcal{P}$ induces *ordering on literals* [13]: for literals l_1, l_2 we write $l_1 < l_2$ and say that l_1 is *less than* l_2 if $\mathsf{var}(l_1)$ appears before $\mathsf{var}(l_2)$ in $\mathcal{P}$.

A closed QBF is *false* (resp. *true*), iff it is semantically equivalent to the constant 0 (resp. 1). If a variable is universally quantified, we say that the variable is *universal*. For a literal l and a universal variable x such that $\mathsf{var}(l) = x$, we say that l is *universal*. Existential variable and literal are defined analogously.

2.1 QU-resolution

QU-resolution [52] is a calculus for showing that a QCNF is false. It comprises two operations, *resolution* and $\forall$*-reduction*. Resolution is defined for two clauses $C_1 \vee x$ and $C_2 \vee \bar{x}$ such that $C_1 \cup C_2$ does not contain complementary literals nor any of the literals x, $\bar{x}$. The *QU-resolvent* (or simply resolvent) of such clauses is the clause $C_1 \vee C_2$. The $\forall$*-reduction* operation removes from a clause C all universal literals l for which there is *no* existential literal $k \in C$ s.t. $l < k$.

For a QCNF $\mathcal{P}\,.\,\phi$, a *QU-resolution proof* of a clause C is a finite sequence of clauses $C_1, \ldots, C_n$ where $C_n = C$ and any C_i in the sequence is part of the given matrix ϕ; or it is a QU-resolvent for some pair of the preceding clauses; or it was obtained from one of the preceding clauses by $\forall$-reduction. A QU-resolution proof is called a *refutation* iff C is the empty clause.

QU-resolution is a slight extension of Q-resolution [15]. Unlike QU-resolution, Q-resolution does *not* enable resolving on universal literals. While Q-resolution is on its own refutationally complete for QCNF, resolutions on universal literals are useful in certain situations (see also [21]).

2.2 Term-Resolution and Model Generation

Term-resolution is analogous to Q-resolution with the difference that it operates on terms and its purpose is to prove that a QBF is true [22]. Resolution is defined for two terms $T_1 \wedge x$ and $T_1 \wedge \bar{x}$ where $T_1 \cup T_2$ do not contain any complementary literals nor any of the literals x, $\bar{x}$; the resolvent is the term $T_1 \wedge T_2$. The $\exists$-*reduction* operation removes from a term T all existential literals l such that there is *no* universal literal $k \in T$ with $l < k$.

Since term-resolution is defined on terms, i.e. on DNF, the *model generation* rule is introduced in order to enable generation of terms from a CNF matrix. For a QCNF $\Phi = \mathcal{P} \,.\, \phi$, a term T is generated by the model generation rule if for each clause C there is a literal l s.t. $l \in C$ and $l \in T$. Then, a *term-resolution proof* of the term T_m from Φ a is a finite sequence $T_1, \ldots, T_m$ of terms such that each term T_i was generated by the model generation rule; or it was obtained from the previous terms by ∃-reduction or term-resolution. Such proof *proves* $\mathcal{P} \,.\, \phi$ iff T_m is the empty term. (Terms are often referred to as 'cubes', especially in the context of DPLL QBF solvers that apply cube learning.) In the remainder of the article, whenever we talk about term-resolution proofs for QCNF, we mean the application of the model generation and term-resolution rule. A QCNF formula is true iff it has a term-resolution proof [22].

In this paper, both term-resolution and QU-resolution proofs are treated as connected directed acyclic graphs so that the each clause/term in the proof corresponds to some node labeled with that clause/term.

2.3 QBF as Games

The semantics of QBF can be stated as a game between an *universal* and an *existential player* [1]. The universal player assigns values to universal variables and analogously the existential player assigns values to the existential variables. A player assigns a value to a variable if and only if all variables preceding it in the prefix were assigned a value. The universal player wins if under the complete resulting assignment the underlying matrix evaluates to false and the existential player wins if the underlying matrix evaluates to true. A formula is true iff there exists a winning strategy for the existential player. The notion of strategy was formalized into *models of QBF* [16].

Definition 1 (Strategy and Model). *Let $\Phi = \mathcal{P} \,.\, \phi$ be QBF with the universal variables $u_1, \ldots, u_n$ and with the existential variables $e_1, \ldots, e_m$. A* strategy *M is a sequence of propositional formulas $\psi_{e_1}, \ldots, \psi_{e_m}$ such that each ψ_{e_i} is over the universal variables preceding e_i in the quantification order. We refer to the formula ψ_x as the* definition *of x in M.*

A strategy M is a model *of Φ if and only if the following formula is true*

$$\forall u_1, \ldots, u_n.\, \phi[\psi_{e_1}/e_1, \ldots, \psi_{e_m}/e_m]$$

i.e., $\phi[\psi_{e_1}/e_1, \ldots, \psi_{e_m}/e_m]$ is a tautology.

Notation. *Let $\Phi = \mathcal{P} \,.\, \phi$ be a QBF as in Definition 1 and $M = (\psi_{e_1}, \ldots, \psi_{e_m})$ be a strategy. For a formula ξ we write $M(\xi)$ for the formula $\xi[\psi_{e_1}/e_1, \ldots, \psi_{e_m}/e_m]$. For a total assignment τ to the universal variables $U = u_1, \ldots, u_n$, we write $M(\xi, \tau)$ for $M(\xi)[\tau(u_1)/u_1, \ldots, \tau(u_n)/u_n]$. Intuitively, $M(\xi, \tau)$ is the result of the game under strategy M and the moves τ. Hence, if ξ is over the variables of Φ, then $M(\xi)$ is over U and $M(\xi, \tau)$ yields the constant which results from evaluating ξ under the strategy M and assignment τ. In particular, M is a model of Φ iff $M(\xi, \tau) = 1$ for any τ.*

Example 1. For a QCNF $\forall u \exists e.\, (\bar{u} \vee e) \wedge (u \vee \bar{e})$, the strategy $M = (\phi_e)$, where $\phi_e = u$ is a model. Observe that $M(\bar{u} \vee e) = M(u \vee \bar{e}) = u \vee \bar{u}$ are tautologies.

A formula QCNF is true if and only if it has a model [16, Lemma 1]; deciding whether a strategy is a model of a formula is coNP-complete [16, Lemma 3]. We should note that here we follow the definition of model by Büning *et. al.*, which has a syntactic nature. However, semantic-based definitions of the same concept appear in literature [35,4].

3 QBF Preprocessing Techniques

For the following overview of preprocessing techniques we consider a QCNF $\mathcal{P} \,.\, \phi$ for some quantifier prefix $\mathcal{P}$ and a CNF matrix ϕ. All the techniques are validity-preserving.

Let $C \in \phi$ be a clause comprising a single existential literal l. *Unit propagation* is the operation of removing from ϕ all clauses that contain l, and removing the literal $\bar{l}$ from clauses containing it.

A clause $C \in \phi$ is *subsumed* by a different clause $D \in \phi$ if $D \subseteq C$; *subsumption removal* consists in removing clause C.

Consider clauses $C, D \in \phi$ together with their resolvent R. If R subsumes C, then we say that C is *strengthened* by *self-subsumption* using D. *Self-subsumption strengthening* consists in replacing C with R [20].

A literal l is *pure* in Φ if $\bar{l}$ does not appear in ϕ. If l is pure and universal, then the *pure literal rule* (PRL) [17] consists in removing all occurrences of l. If l is pure and existential, then the PLR removes all the clauses containing the literal l.

The technique of *blocked clause elimination* (BCE) [36,9] hinges on the definition of a *blocked literal*. An existential literal l is blocked in a clause C if for any clause $D \in \phi$ s.t. $\bar{l} \in D$ there is a literal $k \in C$ with $k < l$ and $\bar{k} \in D$. A clause is blocked if it contains a blocked literal. BCE consists in removing blocked clauses from the matrix.

Variable elimination (VE) [42,24] replaces all clauses containing a certain variable with all their possible resolvents on that variable. In QBF, to ensure soundness, the technique is carried out only if a certain side-condition is satisfied. For an existential variable x, let us partition ϕ into $\phi_x \cup \phi_{\bar{x}} \cup \xi$ where ϕ_x has all clauses containing the literal x, and $\phi_{\bar{x}}$ has all clauses containing the literal $\bar{x}$. For any clause $C \in \phi_x$ that contains some literal k s.t. $x < k$ and any clause $D \in \phi_{\bar{x}}$, there is a literal $z < x$ s.t. $z \in C$ and $\bar{z} \in D$. Variable elimination consists in replacing $\phi_x \cup \phi_{\bar{x}}$ with the set of resolvents between the pairs of clauses of ϕ_x and $\phi_{\bar{x}}$ for which the resolution is defined.

The *binary implication graph* (e.g. [28]) G_ϕ is constructed by generating for each binary clause $l_1 \vee l_2 \in \phi$ two edges: $\bar{l}_1 \rightarrow l_2$ and $\bar{l}_2 \rightarrow l_1$. If two literals appear in the same strongly connected component of G_ϕ, then they must be equivalent. *Equivalent literal substitution* (ELS) consists in replacing literals appearing in the same strongly connected component S by one of the literals from S; this literal is called the *representative*. The representative is then substituted in place of the other literals of S. While in plain SAT preprocessing a representative can be chosen arbitrarily, in QBF it must be done with care. First, three conditions are checked: (1) S contains two distinct universal literals (also covers complementary universal literals); (2) S contains an existential literal l_e and a universal literal l_u such that $l_e < l_u$; (3) S contains two complementary existential literals. If either of the conditions (1), (2), or (3) is satisfied, then the whole formula is false (cf. [2]), and ELS stops. Otherwise, ELS picks as representative the literal that is the outermost with respect to the considered prefix. Observe that if the component contains exactly one universal literal, it will be chosen as the representative. All clauses that become tautologous due to the substitution, are removed from the matrix (this includes the binary clauses that were used to construct the strongly connected components).

4 Limitations

In this section we focus on the limitations of currently-available calculi from the perspective of preprocessing. In particular, we show that term-resolution+model-generation proofs cannot be tractably reconstructed for blocked clause elimination and variable elimination. For a given parameter $n \in \mathbb{N}^+$ construct the following true QCNF with $2n$ variables and $2n$ clauses.

$$\forall u_1 \exists e_1 \ldots \forall u_n \exists e_n . \bigwedge_{1 \leq i \leq n} (\bar{u}_i \vee e_i) \wedge (u_i \vee \bar{e}_i) \tag{1}$$

Proposition 1. *Any term-resolution proof of* (1) *has size exponential in* n.

Proof. Pick an arbitrary assignment τ to the universal variables $u_1, \ldots, u_n$. We say that a term T *agrees* with an assignment τ iff there is no literal l such that $\bar{l} \in T$ and $\tau(l) = 1$. Given a term-resolution proof π for (1), we show that π must have a leaf that agrees with τ by constructing a path from the root to some leaf such that each node on that path agrees with τ. The root of π agrees with τ because it does not contain any literals. If a term T agrees with τ, and T is obtained from T' by $\exists$-reduction, then T' also agrees with τ since τ assigns only to universal variables. If T agrees with τ and is obtained from T_0 and T_1 by term-resolution on some variable y, then $y \in T_k$ and $\bar{y} \in T_{1-k}$ for some $k \in \{0, 1\}$. Hence, at least one of the terms T_0 and T_1 agrees with τ.

Recall that each leaf T of π must be obtained by the model-generation rule; i.e., for each clause C of (1) there is a literal l s.t. $l \in C$ and $l \in T$. Hence, for each pair of clauses $(\bar{u}_i \vee e_i) \wedge (u_i \vee \bar{e}_i)$ either $\bar{u}_i, \bar{e}_i \in T$ or $u_i, e_i \in T$. Consequently, each leaf of π has n universal literals.

For each of the 2^n possible assignments τ, the proof π must contain a leaf T_τ that agrees with τ. Since T_τ contains n universal literals, for a different assignment τ' there must be another leaf $T_{\tau'}$ that agrees with it. Overall, π must must contain at least 2^n different terms.

Proposition 2. *Both blocked clause elimination and variable elimination reduce the matrix of* (1) *to the empty set of clauses in polynomial time.*

Proof. Immediate from definitions of blocked clause and variable elimination.

Corollary 1. *If blocked clause elimination or variable elimination are used for preprocessing, then reconstructing a term-resolution proof takes exponentially more time than preprocessing, in the worst case.*

In the remainder of the paper we do not consider term-resolution+model-generation proofs for certification since Corollary 1 shows that, in the context of preprocessing, this calculus is not appropriate. Rather than term-resolution, we will use models to certify true formulas. We should note, however, that for such we are paying a price of higher complexity for certificate verification. While term-resolution+model-generation proofs can be verified in polynomial time, verification of models is coNP-complete. (For false formulas, QU-resolution is used for certification, which is still verifiable in polynomial time.)

In a similar spirit, we do not consider the preprocessing technique of universal-expansion [12], which is based on the identity $\forall x. \Phi = \Phi[1/x] \wedge \Phi[0/x]$. While there is no hard evidence that there is no tractable algorithm for reconstructing QU-resolution proofs for universal-expansion, recent work hints in this direction [30]. Hence, only the techniques described in Section 3 are considered.

5 Certificate Reconstruction

This section shows how to produce certificates in the context of preprocessing. In particular, we focus on two types of certificates: QU-resolution refutations (Section 2.1) for false formulas and models (Definition 1) for true formulas. We consider each of the techniques presented in Section 3 and we show how a certificate is *reconstructed* from the certificate of the preprocessed formula. This means that reconstruction produces a model (resp. refutation) for a formula Φ from a model (resp. refutation) for a formula Φ', which resulted from Φ by the considered technique. For nontrivial reconstructions we also provide a proof of why the reconstruction is correct.

Having a reconstruction for each of the preprocessing techniques individually enables us to reconstruct a certificate for the whole preprocessing process. The preprocessing process produces a sequence of formulas $\Phi_0, \ldots, \Phi_n$ where Φ_0 is the input formula, Φ_n is the final result, and each formula Φ_{i+1} is obtained from Φ_i by *one* preprocessing technique. For the purpose of the reconstruction, we are given a certificate $\mathcal{C}_n$ for the formula Φ_n. This final certificate $\mathcal{C}_n$ is in practice obtained by a QBF solver. The reconstruction for the whole processing process works backwards through the sequence of formulas $\Phi_0, \ldots, \Phi_n$. Using $\mathcal{C}_n$, it reconstructs a certificate $\mathcal{C}_{n-1}$ for the formula Φ_{n-1}, then for Φ_{n-2} and so on until it produces a certificate $\mathcal{C}_0$ for the input formula. The reminder of the section describes these individual reconstructions for the considered techniques.

We begin by two simple observations. If a transformation removes a clause, then reconstruction of a QU-resolution proof does not need to do anything. Analogously, reconstruction of models is trivial for transformations adding new clauses.

Observation 1 *Consider a QCNF $\Phi = \mathcal{P} \,.\, \phi$ and a clause $C \in \phi$. Any QU-resolution proof of $\Phi' = \mathcal{P} \,.\, \phi \smallsetminus \{C\}$ is also a QU-resolution proof of Φ.*

Observation 2 *Consider a QCNF $\Phi = \mathcal{P} \,.\, \phi$ and a clause C over the variables of Φ. Any model of $\Phi' = \mathcal{P} \,.\, \phi \cup \{C\}$ is a model of Φ.*

5.1 Subsumption, Self-Subsumption, and Unit Propagation

In the case of subsumption, a QCNF $\Phi = \mathcal{P} \,.\, \phi$ is transformed into $\Phi' = \mathcal{P} \,.\, \phi \smallsetminus \{C\}$ for a clause C for which that there is another clause $D \in \phi$ such that $D \subseteq C$. For reconstructing QU-resolution nothing needs to be done due to Observation 1. For any model M' of Φ', the formula $M'(\phi \smallsetminus \{C\})$ is a tautology and in particular $M'(D)$ is a tautology and therefore necessarily $M'(C)$ is a tautology because C is weaker than D. Hence, $M'(\phi)$ is a tautology and M' is also a model of Φ.

In order to reconstruct unit propagation and self-subsumption we first show how to reconstruct resolution steps. For such, consider the transformation of a QCNF $\Phi = \mathcal{P} \,.\, \phi$ into the formula $\Phi' = \mathcal{P} \,.\, \phi \cup \{C\}$ where C is a resolvent of some clauses $D_1, D_2 \in \phi$. Any QU-resolution proof π' of Φ' where C appears as a leaf of π' is transformed into a QU-resolution proof of Φ by prepending this leaf with the resolution step of D_1 and D_2. Any M' model of Φ' is also a model of Φ due to Observation 2.

Each self-subsumption strengthening consists of two steps: resolution and subsumption. Unit propagation consists of resolution steps, subsumption, and the pure literal rule (see Section 5.3). Hence, certificates are reconstructed accordingly. Note that in self-subsumption strengthening, resolution steps may be carried out on universal literals while in unit propagation this would not be meaningful because the moment the matrix contains a unit clause where the literal is universal, the whole formula is trivially false due to universal reduction.

5.2 Variable Elimination (VE)

To eliminate a variable x from $\mathcal{P} \,.\, \phi$, VE partitions the matrix ϕ into the sets of clauses ϕ_x, $\phi_{\bar{x}}$, and ξ as described in Section 3. Subsequently, ϕ_x and $\phi_{\bar{x}}$ are replaced by the set $\phi_x \otimes \phi_{\bar{x}}$, which is defined as the set of all possible resolvents on x of clauses that do not contain another complementary literal. Recall that VE can be only carried out if the side-condition specified in Section 3 is fulfilled.

To reconstruct a QU-resolution proof we observe that VE can be split into operations already covered. The newly added clauses are results of resolution on existing clauses, which was already covered in Section 5.1. Clauses containing x are removed, which does not incur any reconstruction due to Observation 1.

To reconstruct models we observe that any given formula Φ can be written as $\Phi = \mathcal{P}_1 \, \exists x \, \mathcal{P}_2 \,.\, (x \vee \phi_1) \wedge (\bar{x} \vee \phi_2) \wedge \xi$ for CNF formulas ϕ_1, ϕ_2, and ξ that do not contain x. Then, VE consists in transforming Φ into the formula $\Phi' = \mathcal{P}_1 \, \mathcal{P}_2 \,.\, (\phi_1 \vee \phi_2) \wedge \xi$ (note that $\phi_1 \vee \phi_2$ corresponds to $(x \vee \phi_1) \otimes (\bar{x} \vee \phi_2)$). VE's side-condition specifies that any clause $C \in \phi_1$ that contains some literal k

such that $k > x$ and any clause $D \in \phi_2$, there is a literal $z < x$ such that $z \in C$ and $\bar{z} \in D$.

In order to construct a model for the original formula Φ from a model M' of Φ', we aim to add to M' a definition for x which sets x to 1 when ϕ_1 becomes 0 and it sets it to 1 when ϕ_2 becomes 0. Since M' is a model of Φ', the strategy M' satisfies one of the ϕ_1, ϕ_2 for any game. The difficulty lies in the fact that ϕ_1 and ϕ_2 may contain variables that are on the right from x in the quantifier prefix (those in $\mathcal{P}_2$) and these must not appear in the definition of x. Hence, we cannot use ϕ_1 and ϕ_2 to define x as they are. Instead, we construct a formula ϕ_2' by removing from ϕ_2 all unsuitable literals, i.e. literals k for which $x < k$. Then, we set the definition for x to $M'(\phi_2')$. Now whenever ϕ_2' evaluates to 1, so do ϕ_2 and $(x \vee \phi_1) \wedge (\bar{x} \vee \phi_2)$, because x is set to 1. If, however, ϕ_2' evaluates to 0, then ϕ_2 might not necessarily evaluate to 0, but x is set to 0 by our strategy regardless. Due to the side-condition, in such cases ϕ_1 must evaluate to 1 and therefore our strategy is safe. This is formalized by the following proposition.

Proposition 3. *Let $\Phi = \mathcal{P}_1\, \exists x\, \mathcal{P}_2 \,.\, (x \vee \phi_1) \wedge (\bar{x} \vee \phi_2) \wedge \xi$ with ϕ_1 and ϕ_2 not containing x; let $\Phi' = \mathcal{P}_1\, \mathcal{P}_2 \,.\, (\phi_1 \vee \phi_2) \wedge \xi$, as above. Define ϕ_2' to be ϕ_2 with all the literals not less than x deleted; i.e., $\phi_2' = \{\{l \mid l \in C, l < x\} \mid C \in \phi_2\}$. If M' is a model for Φ', then $M = M' \cup \{\psi_x\}$ is a model for Φ, where $\psi_x = M'(\phi_2')$.*

Proof. The functions of M form a well-defined strategy since M' is a well-defined strategy and ψ_x does not contain any literals k with $k > x$. To show that M is a model of Φ, consider any complete assignment τ to the universal variables of Φ. Now we wish to show that the matrix of Φ evaluates to 1 under M and τ. Since M' is a model of Φ', and ξ does not contain x, it holds that $M(\xi, \tau) = M'(\xi, \tau) = 1$. So it is left to be shown that the subformula $(x \vee \phi_1) \wedge (\bar{x} \vee \phi_2)$ is true under M and τ.

Because ϕ_1, ϕ_2 do not contain x we have $M(\phi_1) = M'(\phi_1)$, $M(\phi_2) = M'(\phi_2)$, $M(\phi_2') = M'(\phi_2')$, and $M(\phi_1 \vee \phi_2, \tau) = M'(\phi_1 \vee \phi_2, \tau) = 1$. Split on the following cases (distinguishing between the values of x under τ and M).

If $M(x, \tau) = M'(\phi_2', \tau) = 1$. Because ϕ_2' is stronger than ϕ_2, i.e. $M(\phi_2') \rightarrow M(\phi_2)$, also $M(\phi_2, \tau) = 1$. Hence $M((x \vee \phi_1) \wedge (\bar{x} \vee \phi_2), \tau) = 1$.

If $M(x, \tau) = M'(\phi_2', \tau) = 0$. There must be a clause $C' \in \phi_2'$ s.t. $M'(C', \tau) = 0$, i.e. for all literals $l \in C'$, $M'(l, \tau) = 0$. Let $C \in \phi_2$ be a clause from which C' resulted by removing some literals (possibly none), i.e. $C' = \{l \mid l \in C, l < x\}$. Now consider two sub-cases depending on whether $C = C'$ or $C \neq C'$. If $C = C'$, $M'(C, \tau) = 0$ and $M'(\phi_2, \tau) = 0$, from which $M'(\phi_1, \tau) = 1$ because $M'(\phi_1 \vee \phi_2, \tau) = 1$. Hence $M((x \vee \phi_1) \wedge (\bar{x} \vee \phi_2)) = 1$. If $C \neq C'$, due to the side-condition, C contains for each clause $D \in \phi_1$ a literal l_D s.t. $\bar{l}_D \in D$ and $l_D < x$. Since each literal l_D is less than x, it is also in C'. Since $M(C', \tau) = 0$, each $M(l_D, \tau) = 0$ and $M(\bar{l}_D, \tau) = 1$. From which $M(\phi_1, \tau) = 1$ and $M((x \vee \phi_1) \wedge (\bar{x} \vee \phi_2), \tau) = 1$. □

5.3 Pure Literal Rule (PLR)

PLR for existential literals is a special case of both variable elimination and blocked clause elimination. (An existential pure literal is a blocked literal in any clause.) Hence, certificate reconstruction for existential PLR is done accordingly.

For a universal literal l with $\mathsf{var}(l) = y$, a QCNF $\Phi = \mathcal{P}_1 \forall y \mathcal{P}_2 \,.\, \phi$ is translated into the QCNF formula $\Phi' = \mathcal{P}_1 \mathcal{P}_2 \,.\, \phi'$ by removing l from all clauses where it appears. To obtain a QU-resolution proof π for Φ from a QU-resolution proof π' one inserts l in any of the leafs $C' \in \phi'$ of π' s.t. there exists $C \in \phi$ with $C' = C \smallsetminus \{l\}$. Then, $\forall$-reductions of l are added to π' whenever possible. Note that the addition of l cannot lead to tautologous resolvents since only l is inserted and never $\bar{l}$. The newly added universal literals must be necessarily $\forall$-reduced as π' eventually resolves away all existential literals. Since l is universal, any model of Φ' is also a model of Φ.

5.4 Blocked Clause Elimination (BCE)

For a QCNF $\Phi = \mathcal{P} \,.\, \phi$, BCE identifies a blocked clause $C \in \phi$ and a blocked existential literal $l \in C$, and removes C from ϕ. Recall that for a blocked literal it holds that for any $D \in \phi$ such that $\bar{l} \in D$ there exists a literal $k \in C$ such that $\bar{k} \in D$ and $k < l$.

To reconstruct QU-resolution proofs, nothing needs to be done due to Observation 1. To show how to reconstruct models, let M' be a model for $\Phi' = \mathcal{P} \,.\, \phi \smallsetminus \{C\}$. Let W be the set of literals that serve as witnesses for l being blocked, i.e. $W = \left\{k \in C \mid k \neq l \text{ and there exists a } D \in \phi \text{ s.t. } \bar{k}, \bar{l} \in D \text{ and } k < l\right\}$.

The intuition for constructing a model for $\mathcal{P} \,.\, \phi$ is to play the same as M' except for the case when the literals W are all 0, then make sure that l evaluates to 1. This is formalized by the following proposition.

Proposition 4. *Let Φ, Φ', M', and W be defined as above. Let $x = \mathsf{var}(l)$ and $\psi'_x \in M'$ be the definition for x. Define $\psi_x = \psi'_x \vee M'(\bigwedge_{k \in W} \bar{k})$ if $l = x$ and $\psi_x = \psi'_x \wedge M'(\bigvee_{k \in W} k)$ if $l = \bar{x}$. Finally, define $M = M' \smallsetminus \{\psi'_x\} \cup \{\psi_x\}$. Then M is a model of Φ. (Note that universal literals of W are untouched by M'.)*

Proof. Strategy M is well-defined because literals in W are all less than l and therefore definitions for those literals also contains literals less than l. Let us consider some total assignment τ to the universal variables of Φ under which all literals in W are 0 under M (for other assignments M behaves as M' and C is true). Now let us split the clauses of ϕ into 3 groups. Clauses that do not contain $\bar{l}$ nor l; clauses that contain l; and those that contain $\bar{l}$. For any clause $D \in \phi$ not containing l nor $\bar{l}$, $M(D, \tau) = 1$ since $M(D, \tau) = M'(D, \tau)$ and M' is a model of Φ'. For any clause $D \in \phi$ containing l, $M(D, \tau) = 1$ since $M(l, \tau) = 1$; this includes the clause C. Due to the sidecondition, any clause $D \in \phi$ that contains $\bar{l}$ also contains a literal k s.t. $\bar{k} \in W$. Since for $M(\bar{k}, \tau) = 0$, i.e. $M(k, \tau) = 1$, it holds that $M(D, \tau) = 1$. ☐

5.5 Equivalent Literal Substitution (ELS)

For a formula $\Phi = \mathcal{P} \,.\, \phi$, ELS constructs strongly connected components of the binary implication graph G of ϕ. Once a strongly connected component S of the graph is constructed, ELS checks whether S yields falsity. If it does, ELS produces a QU-resolution proof for such. The following discusses scenarios of falsity that may arise. First recall that if there is a path in G from a literal l_1 to l_k then there is a set of clauses $(\bar{l}_1 \vee l_2)$, $(\bar{l}_2 \vee l_3)$, $\dots$, $(\bar{l}_{k-1} \vee l_k)$, which through a series of QU-resolution steps enables us to derive the clause $\bar{l}_1 \vee l_k$. Also recall that whenever there is a path from l_1 to l_k in some component S_1, there is also a path from $\bar{l}_1$ to $\bar{l}_k$ in the component S_2, obtained from S_1 by negating all literals and reversing all edges. These observations are repeatedly used in the following text.

(1) If S contains two universal literals l_1 and l_2, derive the clause $\bar{l}_1 \vee l_2$, which is then $\forall$-reduced to the empty clause. (Note that this also covers $l_2 = \bar{l}_1$.)

(2) If S contains an existential literal l_e and an universal literal l_u such that $l_e < l_u$, derive the clause $\bar{l}_e \vee l_u$ from which $\forall$-reduction gives $\bar{l}_e$. Derive l_e analogously. Finally resolve $\bar{l}_e$ and l_e to obtain the empty clause.

(3) If S contains two literals e and $\bar{e}$ for some existential variable e, derive the unit clauses e and $\bar{e}$ and resolve them into the empty clause.

If none of the three conditions above are satisfied, all literals in S are substituted by a representative literal r, which is the smallest literal from S w.r.t. the literal ordering $<$. This yields a formula $\Phi' = \mathcal{P}' \,.\, \phi'$, where $\mathcal{P}'$ resulted from $\mathcal{P}$ by removing all variables that appear in S except for $\mathsf{var}(r)$. A certificate is reconstructed as follows.

If a QU-resolution proof π' for Φ' relies on a clause $C' \in \phi'$ that resulted from some cause $C \in \phi$ by replacing a $l \in S$ by r, construct the clause $\bar{l} \vee r$ and resolve it with C to obtain C'. Analogously, if C' resulted from C by replacing $\bar{l} \in S$ with $\bar{r}$, construct the clause $l \vee \bar{r}$ and resolve it with C to obtain C'.

If M' is a model of Φ' and r is existential, then S does not contain any universal literals and M' defines the value for r by some formula $\psi_r = M'(r)$. In such case ψ_r is over universal variables that are less than all the literals in S because r was chosen to be the outermost literal. If $x \in S$ for some existential variable x, set ψ_x as ψ_r; if $\bar{x} \in S$ for some existential variable x, set ψ_x as $\neg\psi_r$. If r is universal, all the other literals in S are existential and so for $x \in S \smallsetminus \{r\}$ we set $\psi_x = r$; for $\bar{x} \in S \smallsetminus \{r\}$, we set $\psi_x = \bar{r}$.

6 Related Work

Local simplifications based on identities such as $0x = 0$ appear in number of instances of automated reasoning (c.f. [27]). In SAT solving, it was early recognized that going beyond such local simplifications leads to significant performance gains. A notable technique is variable elimination (VE), which originates in the Davis&Putnam procedure (DP). While DP is itself complete, it suffers from unwieldy memory consumption. It has been shown that applying VE *only* if it does not lead to increase of the formula's size, gives an incomplete yet powerful technique [51]. The preprocessor SatELite [20] boosts VE by subsumption, self-subsumption, and unit propagation.

Nowadays, preprocessors (and SAT solvers themselves) contain a number of preprocessing techniques such as *blocked clauses elimination* [36,41,32], *hyper binary resolution* [3] and others (cf. [28]). Reconstructing solutions in SAT is generally easier than in QBF, but it has also been investigated [31].

Many SAT preprocessing techniques were generalized for QBF [8,47,24,11,9]; application thereof is crucial for QBF solving [44]. QBF leads to a number of specifics in the techniques. VE can be only performed under a certain side-condition (Section 3); Van Gelder [52] further generalizes this side-condition. A technique specific to QBF is *universal-variable expansion* [12,11] where a universal quantifier $\forall x.\Phi$ is expanded into $\Phi[0/x] \wedge \Phi[1/x]$ and then brought into the prenex form by variable renaming. (Expansion can be used to obtain a complete solver [5,8,37,29].) In his recent work, Van Gelder provides some initial insights into reconstruction of variable elimination and expansion [53]. There, however, he only shows how to reconstruct an individual leaf of a term-resolution proof, but does not show how to construct the proofs themselves.

A number of works focus on the certification of QBF solvers (e.g. [6,35,39,26]) motivated by error prevention [10], but also because the certificates themselves can be useful (e.g. [25,49,45,4,33]).

7 Experimental Evaluation

We test five scenarios, corresponding to different settings for preprocessing (**f**ull, **s**imple, or **n**one) and for solving (with a **q**dag dependency manager, or **s**imple). Table 1 defines and names the scenarios that we tested — the last letter indicates whether certificate generation was enabled (**y**es or **n**o). The scenario *nsy* represents the state-of-the-art in QBF solving *with* certificate generation, and is the scenario we set out to improve. The scenario *ssy* represents our contribution to QBF solving with certificate generation. We use the QBC format for certificates [35]: the size of models is the number of $\wedge$-gates used, the size of refutations is the number of resolution steps used. (See online[1] for the exact testing environment being used.)

Results and Discussion. Figure 2 shows the overall performance of five scenarios on the QBFEVAL 2012 benchmark. There is a clear gap between scenarios that use preprocessing (fqn, ssn, ssy) and scenarios that do not use preprocessing (nsn, nsy) — preprocessing is clearly beneficial. The gap nsy–nsn shows that enabling tracing in depqbf deteriorates its performance. The gap ssy–ssn is smaller than the gap nsy–nsn, indicating that enabling tracing in bloqqer+depqbf deteriorates performance *less* than it does for depqbf alone. The gap fqn–ssn should be reduced by future work. The most important observation to make on Figure 2 is that

[1] http://sat.inesc-id.pt/~mikolas/lpar13-prepro/

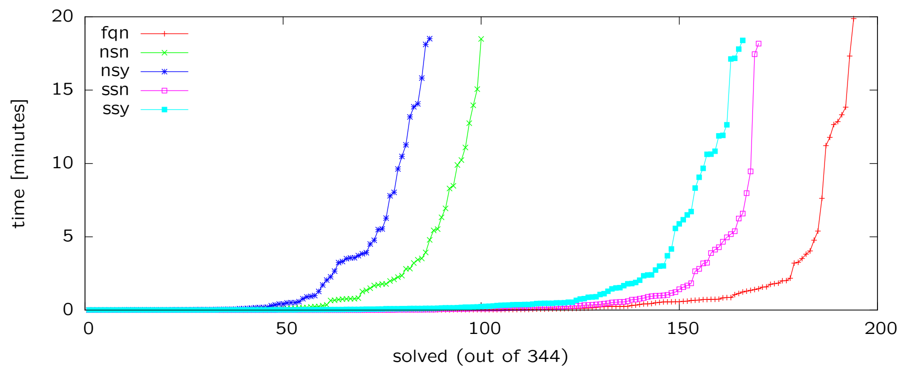


Fig. 2: Overall performance on the QBFEVAL 2012 benchmark

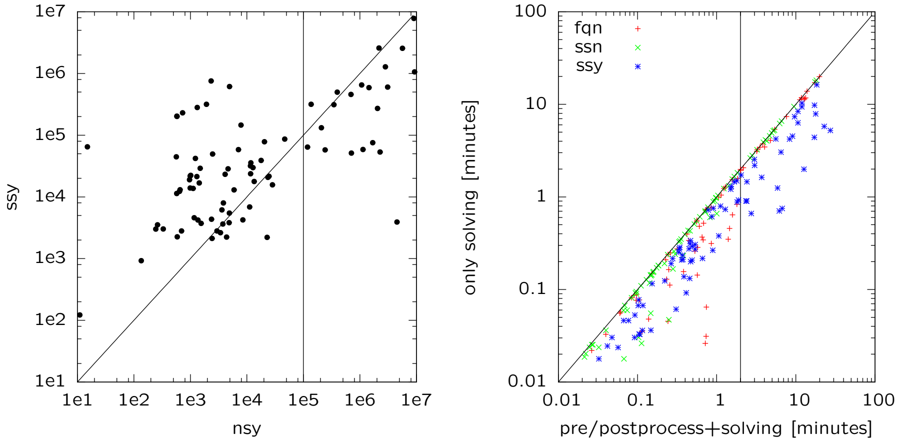
Table 1: Number of solved instances out of 344, for several scenarios

Scenario				True/SAT		False/UNSAT		
Name	Preprocessing	Solving	Tracing	Unchecked	Checked	Unchecked	Checked	Total
fqn	full	qdag	no	99	n/a	94	n/a	194
nsn	none	simple	no	42	n/a	58	n/a	100
nsy	none	simple	yes	7	25	0	55	87
ssn	simple	simple	no	80	n/a	90	n/a	170
ssy	simple	simple	yes	8	69	0	89	166

our proposed scenario (ssy) significantly improves the state-of-the-art in QBF solving with certificate generation (nsy). Table 1 gives the total number of solved instances for each scenario, thus it corresponds to the rightmost points in Figure 2. The generated certificates (in scenarios nsy, ssy) were not all checked: Those instances on which the certificate checker timed out are listed in the unchecked column. (Recall that checking strategies is coNP-complete.) The 7 unchecked certificates in the nsy scenario are largely disjoint from the 8 unchecked certificates in the ssy scenario — the overlap is exactly one instance.

Figure 3 shows that preprocessing is beneficial mostly for hard instances. Figure 3a depicts certificate size with preprocessing (ssy) versus certificate size without preprocessing (nsy). There is a clear threshold around 10^5: above it preprocessing helps, below it preprocessing is detrimental. Figure 3b depicts time spent in the solver versus total solving time (which includes preprocessing and postprocessing) for the three scenarios that use preprocessing. There is a clear threshold around 2 minutes: above it, scenarios that do not generate certificates (fqn, ssn) have negligible overhead.

The correlation between certificate size and total running time is only moderate (≈ 0.6). As an example of the high variance, for the 10 instances that were solved in 64-to-128 seconds, the average certificate size was 4.7×10^5, with a standard deviation of 4.8×10^5.



(a) Certificate size

(b) Solving time

Fig. 3: The effect of pre/postprocessing on certificate size and on solving time

Table 2: Time spent in solver as a percent of the total solving time.

Scenario	min [%]	med [%]	geom avg [%]	max [%]
fqn	4	91	66	100
ssn	19	98	86	100
ssy	11	57	50	92

8 Conclusions and Future Work

This paper brings together two different facets of QBF solving: preprocessing and certification. Certification is important for practical applications of QBF and preprocessing is crucial for performance of nowadays QBF solvers. Both of the facets were extensively investigated [22,52,40,8,47,9,24] but there is no available toolchain combining the two. However, the need for such technology has been recognized by others [44]. This paper addresses exactly this deficiency. For a number of representative preprocessing techniques, the paper shows how certificates can be reconstructed from a certificate of a preprocessed formula. Experimental evaluation of the implemented prototype demonstrates that the proposed techniques enable QBF solving *with* certification that is performance-wise very close to a state-of-the-art QBF solving *without* certification. Hence, the contribution of the paper is not only theoretical but also practical since the implemented tool will be useful to the QBF community.

On the negative side, the paper demonstrates that current methods of QBF certification are insufficient for full-fledged preprocessing in the case of true formulas. Namely, term-resolution+model-generation proofs incur worst-case exponential blowup in blocked clause elimination and variable elimination. This is an important drawback because term-resolution proofs can be checked in polynomial time, which is not the case for model-based certification (used in the paper). This drawback delimits one direction for future work: Can we produce polynomially-verifiable certificates for true QBFs in the context of preprocessing? Another item of future work is narrowing the performance gap between solving with and without certificate generation. In this regard, methods for certifying universal-variable expansion should be developed [12] and other techniques, such as hyper-binary resolution, must be certified.

Last but not least, methods for solving QBF were generalized to domains such as SMT or verification [19,38]. We may expect that the contributions made by this paper will also be helpful for these works.

Acknowledgments

We thank Armin Biere and Allen Van Gelder for helpful conversations on QBF and preprocessing. This work is partially supported by SFI PI grant BEACON (09/IN.1/I2618), FCT grants ATTEST (CMU-PT/ELE/0009/2009), POLARIS (PTDC/EIA-CCO/123051/2010), and INESC-ID's multiannual PIDDAC funding PEst-OE/EEI/LA0021/2011. Grigore was supported by the EPSRC Programme Grant 'Resource Reasoning' (EP/H008373/2).

References

1. Arora, S., Barak, B.: Computational Complexity — A Modern Approach. Cambridge University Press (2009)
2. Aspvall, B., Plass, M.F., Tarjan, R.E.: A linear-time algorithm for testing the truth of certain quantified Boolean formulas. Inf. Process. Lett. 8(3) (1979)
3. Bacchus, F., Winter, J.: Effective preprocessing with hyper-resolution and equality reduction. In: SAT (2003)
4. Balabanov, V., Jiang, J.H.R.: Unified QBF certification and its applications. Formal Methods in System Design 41(1) (2012)
5. Benedetti, M.: Evaluating QBFs via symbolic Skolemization. In: LPAR (2004)
6. Benedetti, M.: sKizzo: a suite to evaluate and certify QBFs. In: CADE (2005)
7. Benedetti, M., Mangassarian, H.: QBF-based formal verification: Experience and perspectives. JSAT 5(1-4) (2008)
8. Biere, A.: Resolve and expand. In: SAT (2004)
9. Biere, A., Lonsing, F., Seidl, M.: Blocked clause elimination for QBF. In: CADE (2011)
10. Brummayer, R., Lonsing, F., Biere, A.: Automated testing and debugging of SAT and QBF solvers. In: SAT (2010)
11. Bubeck, U.: Model-based transformations for quantified Boolean formulas. Ph.D. thesis, University of Paderborn (2010)
12. Bubeck, U., Kleine Büning, H.: Bounded universal expansion for preprocessing QBF. In: SAT 07 (2007)
13. Buning, H.K., Letterman, T.: Propositional Logic: Deduction and Algorithms. Cambridge University Press, New York, NY, USA (1999)

14. Büning, H.K., Bubeck, U.: Theory of quantified boolean formulas. In: Handbook of Satisfiability. IOS Press (2009)
15. Büning, H.K., Karpinski, M., Flögel, A.: Resolution for quantified Boolean formulas. Inf. Comput. 117(1) (1995)
16. Büning, H.K., Subramani, K., Zhao, X.: Boolean functions as models for quantified Boolean formulas. J. Autom. Reasoning 39(1) (2007)
17. Cadoli, M., Schaerf, M., Giovanardi, A., Giovanardi, M.: An algorithm to evaluate quantified Boolean formulae and its experimental evaluation. J. Autom. Reasoning 28(2) (2002)
18. Canny, J.F.: Some algebraic and geometric computations in PSPACE. In: STOC (1988)
19. Cheng, C.H., Shankar, N., Ruess, H., Bensalem, S.: EFSMT: A logical framework for cyber-physical systems (2013), http://arxiv.org/abs/1306.3456
20. Eén, N., Biere, A.: Effective preprocessing in SAT through variable and clause elimination. In: SAT (2005)
21. Egly, U., Widl, M.: Solution extraction from long-distance resolution proofs. http://fmv.jku.at/qbf2013/reportQBFWS13.pdf (Jul 2013)
22. Giunchiglia, E., Narizzano, M., Tacchella, A.: Clause/term resolution and learning in the evaluation of quantified Boolean formulas. JAIR 26(1) (2006)
23. Giunchiglia, E., Marin, P., Narizzano, M.: Reasoning with quantified boolean formulas. In: Handbook of Satisfiability. IOS Press (2009)
24. Giunchiglia, E., Marin, P., Narizzano, M.: sQueezeBF: An effective preprocessor for QBFs based on equivalence reasoning. In: SAT (2010)
25. Gorogiannis, N.: Computing Minimal Changes of Models of Systems. Ph.D. thesis, University of Birmingham (2003)
26. Goultiaeva, A., Van Gelder, A., Bacchus, F.: A uniform approach for generating proofs and strategies for both true and false QBF formulas. In: IJCAI. pp. 546–553. IJCAI/AAAI (2011)
27. Harrison, J.: Handbook of Practical Logic and Automated Reasoning. Cambridge University Press (2009)
28. Heule, M., Järvisalo, M., Biere, A.: Clause elimination procedures for CNF formulas. In: LPAR (2010)
29. Janota, M., Klieber, W., Marques-Silva, J., Clarke, E.M.: Solving QBF with counterexample guided refinement. In: SAT (2012)
30. Janota, M., Marques-Silva, J.: On propositional QBF expansions and Q-resolution. In: SAT (2013)
31. Järvisalo, M., Biere, A.: Reconstructing solutions after blocked clause elimination. In: SAT (2010)
32. Järvisalo, M., Biere, A., Heule, M.: Simulating circuit-level simplifications on CNF. J. Autom. Reasoning 49(4) (2012)
33. Jordan, C., Kaiser, L.: Experiments with reduction finding. In: SAT (2013)
34. Jussila, T., Biere, A.: Compressing BMC encodings with QBF. Electr. Notes Theor. Comput. Sci. 174(3) (2007)
35. Jussila, T., Biere, A., Sinz, C., Kröning, D., Wintersteiger, C.M.: A first step towards a unified proof checker for QBF. In: SAT (2007)
36. Kullmann, O.: New methods for 3-SAT decision and worst-case analysis. Theor. Comput. Sci. 223(1-2) (1999)
37. Lonsing, F., Biere, A.: Nenofex: Expanding NNF for QBF solving. In: SAT (2008)
38. Morgenstern, A., Gesell, M., Schneider, K.: Solving games using incremental induction. In: IFM (2013)

39. Narizzano, M., Peschiera, C., Pulina, L., Tacchella, A.: Evaluating and certifying QBFs: A comparison of state-of-the-art tools. AI Commun. 22(4) (2009)
40. Niemetz, A., Preiner, M., Lonsing, F., Seidl, M., Biere, A.: Resolution-based certificate extraction for QBF. In: SAT (2012)
41. Ostrowski, R., Grégoire, É., Mazure, B., Sais, L.: Recovering and exploiting structural knowledge from CNF formulas. In: CP (2002)
42. Pan, G., Vardi, M.Y.: Symbolic decision procedures for QBF. In: CP (2004)
43. Plandowski, W.: Satisfiability of word equations with constants is in PSPACE. J. ACM 51(3) (2004)
44. QBF gallery. http://www.kr.tuwien.ac.at/events/qbfgallery2013/ (2013)
45. Samanta, R., Deshmukh, J.V., Emerson, E.A.: Automatic generation of local repairs for Boolean programs. In: FMCAD (2008)
46. Samulowitz, H., Bacchus, F.: Binary clause reasoning in QBF. In: SAT (2006)
47. Samulowitz, H., Davies, J., Bacchus, F.: Preprocessing QBF. In: CP (2006)
48. Schröder, L., Pattinson, D.: PSPACE bounds for rank-1 modal logics. ACM Trans. Comput. Log. 10(2) (2009)
49. Staber, S., Bloem, R.: Fault localization and correction with QBF. In: SAT (2007)
50. Stockmeyer, L.J.: The complexity of decision problems in automata theory and logic. Ph.D. thesis, Massachusetts Institute of Technology (1974)
51. Subbarayan, S., Pradhan, D.K.: NiVER: Non increasing variable elimination resolution for preprocessing SAT instances. In: SAT (2004)
52. Van Gelder, A.: Contributions to the theory of practical quantified Boolean formula solving. In: CP (2012)
53. Van Gelder, A.: Certificate extraction from variable-elimination QBF preprocessors. http://fmv.jku.at/qbf2013/reportQBFWS13.pdf (Jul 2013)