Contributions to the Theory of Practical Quantified Boolean Formula Solving

Allen Van Gelder
http://www.cse.ucsc.edu/~avg

University of California, Santa Cruz

Abstract. Recent solvers for quantified boolean formulas (QBFs) use a clause learning method based on a procedure proposed by Giunchiglia et al. (JAIR 2006), which avoids creating tautological clauses. The underlying proof system is Q-resolution. This paper shows an exponential worst case for the clause-learning procedure. This finding confirms empirical observations that some formulas take mysteriously long times to solve, compared to other apparently similar formulas.
Q-resolution is known to be refutation complete for QBF, but not all logically implied clauses can be derived with it. A stronger proof system called QU-resolution is introduced, and shown to be complete in this stronger sense. A new procedure called QPUP for clause learning without tautologies is also described.
A generalization of pure literals is introduced, called effectively depth-monotonic literals. In general, the variable-elimination resolution operation, as used by Quantor, sQueezeBF, and Bloqqer is unsound if the existential variable being eliminated is not at innermost scope. It is shown that variable-elimination resolution is sound for effectively depth-monotonic literals even when they are not at innermost scope.

1 Introduction

Solvers for quantified boolean formulas (QBFs) are rapidly increasing in strength, partly due to increased understanding of how to incorporate conflict-driven clause learning (CDCL), which found great practical success in propositional satisfiability. Several current solvers are patterned after the Q-resolution method described by Giunchiglia *et al.* [13]. A thorough survey of the field through 2005 may be found in this paper.

It is starting to be recognized that simply delivering a 0 or 1 answer is not good enough, for a solver. There has to be some formal proof system to back up claimed answers. More than just verifying a claimed answer, users want additional information relevant to their applications. To accommodate these needs, solvers need to be implemented in an organized way. This paper addresses a few issues found in current solvers and suggests improvements.[1]

[1] See http://www.cse.ucsc.edu/~avg/QPUP/qpup-cp12-long.pdf for a longer version of this paper.

First, in Section 3 we motivate the study by showing that current clause-learning methods based on [13] might take exponential time to learn one clause. We describe a family of QBF formulas such that the first clause to be learned takes exponential time, although it has a linear-length Q-resolution derivation. Exponential time to learn one clause does not occur with a properly implemented propositional CDCL procedure.

In Section 4 we describe a clause-learning system that runs in polynomial time per clause learned. This avoids the exponential-time worst cases mentioned above. Although it may be impractical in its current form, it might provide the basis for a practical adaptation into the QDPLL framework.

In Section 5 we introduce QU-resolution, a resolution system for QBF that is capable of deriving any logically implied clause (Definition 2.5), a characteristic that is absent from all presently known QBF solvers that utilize the QDPLL framework of [13]. We show that QU-resolution can produce exponentially shorter refutations than Q-resolution on some QBF families. In related work, Egly recently showed that certain sequent systems can produce exponentially shorter refutations than Q-resolution on some QBF families [7]. These sequent systems, in which introduction of new variables is central, can produce exponentially shorter refutations than QU-resolution on the same formulas.

Like Q-resolution, QU-resolution preserves tree models. Tree models represent winning strategies for true formulas. These strategies, also called Skolem functions, contain the extra information needed by many practical applications to achieve the goal that is encoded in the QBF.

Effectively depth-monotonic literals (Section 6) allow QBF variable-elimination resolution in more cases. The paper concludes with Section 7.

2 Preliminaries

In general, ***quantified boolean formulas*** (QBFs) generalize propositional formulas by adding operations consisting of universal and existential quantification of boolean variables. A ***closed QBF*** is one in which every variable is quantified. See [15, 5, 17] for thorough introductions. This paper uses standard notation as much as possible, but some terms have no standard version. We consider resolution and universal reduction separately, although some papers combine them. Also, we use tree models, which are not found in all QBF papers, to define *super-sound* and *safe* operations, and to distinguish between them. Also, we define ordered assignments.

We say that a QBF is in *prenex conjunctive normal form* if all the quantifiers are outermost operators (the prenex, or quantifier prefix), and the quantifier-free portion (also called the matrix) is in CNF; i.e., $\Psi = \overrightarrow{Q}\,.\,\mathcal{F}$ consists of prenex $\overrightarrow{Q}$ and matrix $\mathcal{F}$. For this paper QBFs are in prenex conjunctive normal form. If p precedes q in the quantifier prefix, we say p ***is outer to*** q and q ***is inner to*** p. Clauses in $\mathcal{F}$ are called *input clauses*.

A *closed* QBF evaluates to either *invalid* (false) or *valid* (true), as defined by induction on its principal operator. We use 0 and 1 for truth values of literals and use true and false for semantic values of formulas.

1. $(\exists\, x\, \Phi(x))$ is true if and only if ($\Phi(0)$ is true or $\Phi(1)$ is true).
2. $(\forall\, x\, \Phi(x))$ is false if and only if ($\Phi(0)$ is false or $\Phi(1)$ is false).
3. Other operators have the same semantics as in propositional logic.

This definition emphasizes the connection of QBF to two-person games, in which player E (Existential) tries to set existential variables to make the QBF evaluate to true, and player A (Universal) tries to set universal variables to make the QBF evaluate to false. Players set their variable when it is outermost, or for non-prenex, when it is the root of a subformula (see [16] for more details). Only one player has a winning strategy.

For this paper a ***clause*** is a *disjunctively* connected set of literals. If the term ***cube*** is used, it refers to a *conjunctively* connected set of literals. Literals are variables or negated variables, with overbar denoting negation. We also use $\bot$ as a literal representing false when it makes more uniform notation. Clauses may be written as literals enclosed in square brackets (e.g., $[p, q, \overline{r}]$), and [] denotes the empty clause. Where the context permits, letters e and others near the beginning of the alphabet denote existential literals, while letters u and others near the end of the alphabet denote universal literals. Letters like p, q, r denote literals of unspecified quantifier type. The variable underlying a literal p is denoted by $|p|$ where necessary. Free variables or free literals e and u may be indicated by $\Psi(e, u)$.

The quantifier prefix is partitioned into maximal contiguous subsequences of variables of the same quantifier type, called ***quantifier blocks***. Each quantifier block has a unique ***qdepth***, with the outermost block having qdepth = 1. The ***scope*** of a quantified variable is the qdepth of its quantifier block. We say scopes are *outer* or *inner* to another scope to avoid any confusion about the direction, since there are varying conventions in the literature for numbering scopes.

Definition 2.1 An ***assignment*** is a partial function from variables to truth values, usually represented as the set of literals mapped to 1. A ***total assignment*** is an assignment to all variables. Assignments are denoted by ρ, σ, τ. Application of an assignment σ to a logical expression is called a ***restriction*** and is denoted by $q\lceil_\sigma$, $C\lceil_\sigma$, $\mathcal{F}\lceil_\sigma$, etc. Quantifiers for assigned variables are deleted in $\Psi\lceil_\sigma$.

An ***ordered assignment*** is a special term that denotes a total assignment that is represented by a sequence of literals that are assigned 1 and are in the same order as their variables appear in the quantifier prefix. □

A ***winning strategy*** can be presented as an unordered directed tree. If it is a winning strategy for the E player, it is also called a ***tree model***, which we now describe. We shorten *unordered directed tree* to *tree* throughout this paper. The qualifier "unordered" means that the children of a node do not have a specified order; they are a set. Recall that a ***branch*** in a tree is a path from the root node to some leaf node. A tree can be represented as the set of its branches. We also

define a ***branch prefix*** to be a path from the root node that might terminate before reaching a leaf.

Definition 2.2 Let a QBF $\Phi = \overrightarrow{Q} \cdot \mathcal{F}$ be given. In this definition, σ denotes a (possibly empty) branch prefix of some ordered assignment for Φ. A ***tree model*** M for Φ is a nonempty set of ordered assignments for Φ that defines a tree, such that

1. Each ordered assignment makes $\mathcal{F}$ true, i.e., *satisfies* $\mathcal{F}$ in the usual propositional sense.
2. If e is an existential literal in Φ and some branch of M has the prefix (σ, e), then *no* branch has the prefix $(\sigma, \overline{e})$; that is, treating σ as a tree node in M, it has only one child and the edge to that child is labeled e.
3. If u is an universal literal in Φ and some branch of M has the prefix (σ, u), then *some* branch of M has the prefix $(\sigma, \overline{u})$; that is, treating σ as a tree node in M, it has two children and the edges to those children are labeled u and $\overline{u}$.

If τ is a partial assignment to all variables outer to existential variable e, then requirement (2) ensures that the "Skolem function" $e(\tau)$ is well defined as the unique literal following τ in a branch of M. If the formula evaluates to false, the set of tree models is empty. □

Definition 2.3 A ***tree countermodel*** R for Φ is essentially the dual of a tree model. That is, a tree node has two children with the edges labeled e and $\overline{e}$ when e is existential and has one child when the edge is labeled with universal literal u, and each branch *falsifies* some clause of $\mathcal{F}$.

If τ is a partial assignment to all variables outer to universal variable u, then (the dual of) requirement (3) ensures that the "Herbrand function" $u(\tau)$ is well defined as the unique literal following τ in a branch of R. If the formula evaluates to true, the set of tree countermodels is empty. □

Definition 2.4 For our purposes, an operation on a closed QBF is said to be ***safe*** if it does not change the truth value of the formula. An operation on a closed QBF is said to be ***super sound*** if it *preserves* the set of tree models (i.e., does not *add* or *delete* tree models). Clearly, preserving the set of tree models is a sufficient condition for safety. □

Definition 2.5 Let $\Psi = \overrightarrow{Q}.\mathcal{F}$ be a closed prenex QBF. Another quantifier-free formula $\mathcal{G}$ (usually a clause or a set of clauses) is said to be ***logically implied*** by Ψ if $\overrightarrow{Q}.(\mathcal{F} \wedge \mathcal{G})$ has the same set of tree models as Ψ; that is, the operation of adding $\mathcal{G}$ to $\mathcal{F}$ is super sound. In other words, $\mathcal{G}$ evaluates to true in every tree-model of Ψ. □

The proof system known as ***Q-resolution*** consists of two operations, *resolution* and *universal reduction*, defined next. Q-resolution is of central importance for QBFs because it is a *refutationally* complete proof system [14]. Unlike resolution for propositional logic, Q-resolution is not *inferentially* complete. That is, a new (non-tautological) clause C might be logically implied by a closed QBF Ψ, yet no subset of C is derivable by Q-resolution (see Example 5.5).

Definition 2.6 ***Resolution*** is defined as usual. Let clauses $C_1 = [q, \alpha]$ and $C_2 = [\overline{q}, \beta]$, where q is called the *clashing literal*. Let α be a literal sequence without conflicting literals and without q and $\overline{q}$. Let the same be true of β. Either or both of α and β may be empty. Then $\mathsf{res}_q(C_1, C_2) = [\alpha \cup \beta]$ is the *resolvent*. For Q-resolution, the clashing literal q is required to be existential, and the resolvent is not permitted to be tautologous.

Universal reduction is special to QBF. Let clauses $C_1 = [u, \alpha]$, where u is called the *reduction literal* and is universal. Let α be a literal sequence without conflicting literals and without u and $\overline{u}$. Further, let u be ***tailing for*** α, which means that the quantifier depth of u is greater than (u is inner to) that of any existential literal in α. Then $\mathsf{unrd}_u(C_1) = [\alpha]$. □

Lemma 2.7 Resolution and universal reduction are super-sound operations. Also, resolution preserves the set of tree countermodels.

Proof: Straightforward application of the definitions. ■

We are not aware of a prior definition of tree countermodel, which explicitly encodes a winning strategy for the universal player when the QBF is false. However, there has been work on extracting a winning strategy for the universal player from a Q-resolution refutation [2, 10].

It is worth observing that universal reduction might *add* tree countermodels, as shown by the following example. This is another reason to treat it as a separate operation from resolution.

Example 2.8 Consider $\Phi = \exists d \forall u \exists e \left\{[d, u], [\overline{u}, e], [\overline{d}, \overline{e}]\right\}$. The only tree countermodel is

$$\left\{(\overline{d}, \overline{u}, \overline{e}), (\overline{d}, \overline{u}, e), (d, u, \overline{e}), (d, u, e)\right\},$$

where parentheses enclose branches (ordered assignments).

If universal reduction is applied on the clause $[d, u]$ giving $[d]$, there is a second tree countermodel in which u is positive on all four branches. □

3 QDPLL Exponential Case

QDPLL is the name commonly used for a family of QBF solving procedures with clause learning similar to those described by Giunchiglia *et al.* [11, 12], [13]. Letz describes similar ideas that are used in `SemProp`, but with a different learning procedure [21]. For QDPLL, it is assumed that the input clauses are non-tautological and that any tailing universal literals have been reduced away (Definition 2.6).

When a conflict occurs, QDPLL derives a *learned clause* using Q-resolution on clauses in the conflict graph. We begin by showing a case in which QDPLL spends time that is exponential in the size of the conflict graph to derive its first learned clause. This is remarkable because the learned clause has a linear Q-resolution derivation.

At a high level, the clause learning procedure is similar to that found in CDCL SAT solvers. We assume the reader is generally familiar with the SAT version.

1. Begin with the working clause W_0 being the clause that became falsified.
2. At each derivation step i, choose a literal $\overline{q}$ in the current working clause W_{i-1} and Q-resolve with the antecedent of q in the conflict graph to produce the next working clause W_i. The ***antecedent*** clause is the clause that became unit to imply q.
3. Stop when W_i is an asserting clause that satisfies some additional conditions specific to QBF. Learn W_i. A clause is ***asserting*** when it has a unique literal at the highest decision level among the decision levels represented in the clause.

The procedure is called `Rec-C-Resolve` in [13]. The algorithm in which it is used is called *Q-DLL-LN*.

The central point of the above procedure is the choice of literal $\overline{q}$ in W_{i-1}, which will be used as the clashing literal. The policy in the cited paper is this: Choose the existential literal that was implied *most recently* unless that that choice would produce a tautologous resolvent (in which case Q-resolution is not defined). Otherwise choose an existential literal in W_{i-1} whose quantifier block has innermost scope. This policy is implemented in early versions of `QuBE`. Essentially the same policy is used in `depQBF` [18, 19]. A slight variant is used in `CirQit` [9, 8].

Example 3.1 We now describe a family named *qdpllexp* whose run time increases exponentially with instance length for the three solvers just mentioned.[2] The essence of `qdpllexp_06` is shown in Figure 1. The generation pattern simply varies the number of middle sections and should be self-evident.

The computation begins by assuming outermost existential b_1 is true, implying e_{99} and f_{99} at innermost scope. Now u_{10} is tailing, allowing e_9 and f_9 to be implied. This pattern continues until $\left[\overline{d_1}, \overline{c_1}\right]$ is falsified. In each four-literal clause the two negative existential literals "block" the universal literal. After they are falsified by unit-clause propagation, the universal literal can be reduced, yielding a new implied existential literal.

After $\left[\overline{d_1}, \overline{c_1}\right]$ is falsified, it becomes W_0 for the learning procedure outlined above. This is resolved with $\left[d_1, \overline{f_3}, \overline{e_3}, \overline{u_2}\right]$ (either choice gives similar results) to give $W_1 = \left[\overline{c_1}, \overline{f_3}, \overline{e_3}, \overline{u_2}\right]$. Now the most recently implied literal is c_1, but resolving W_1 with the antecedent of c_1 would be tautologous, so an innermost existential, say f_3, is used instead. Thus W_2 contains $\overline{u_4}$, preventing resolution on $\overline{e_3}$. Also, $\overline{e_3}$ still prevents reduction of $\overline{u_2}$. This pattern continues down the conflict graph. Eventually, e_{99} and f_{99} are introduced in W_5 by resolution on the antecedent of f_9. After e_{99} and f_{99} are resolved out for the first time (allowing $\overline{u_{10}}$ to be reduced), they are *re-introduced* by resolution on the antecedent of e_9. Then they get resolved out for the second time. By the conclusion of the procedure e_{99} and f_{99} are resolved out 32 times.

Further study shows that all paths in the graph are traversed, and it is well known that this family of graphs has exponentially many paths.

[2] See `www.cse.ucsc.edu/~avg/ProofChecker/QdpllexpSimple.tar` for some instances.

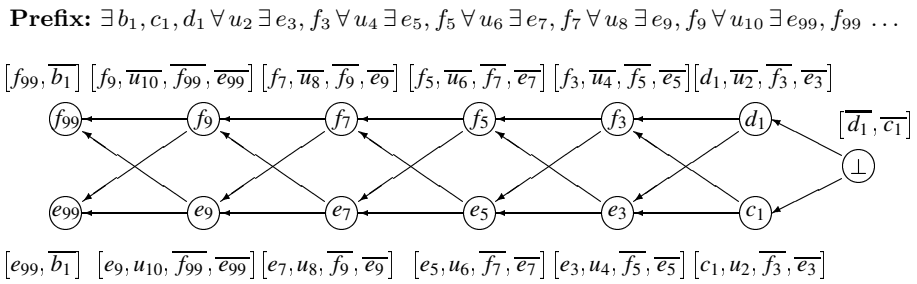

Fig. 1. Excerpt of exponential family of QBF formulas; see Example 3.1. The implied literal is shown inside each circle, which represents a vertex in the conflict graph. Antecedent clauses are shown above or below each circle. Arrows point to reason literals. E. g., f_9 is implied because $\overline{f_{99}}$ and $\overline{e_{99}}$ are falsified, then $\overline{u_{10}}$ is reduced away.

Table 1. Running times in seconds on *qdpllexp* family. See Example 3.1.

family index	18	19	20	21	22	23
`QuBE 1.3`	10	22	47	105	segv	segv
`depQBF 0.1`	8	16	32	69	140	298
`CirQit3.15`	1	1	3	5	11	21

"segv" denotes "segmentation violation".

Table 1 provides empirical confirmation that the running time doubles for each increase of one level in the family. A one-level increase produces 10 additional variables and four more four-literal clauses. The clauses seen in Figure 1 are essentially cloned with fresh variables, except that b_1 appears instead of $\overline{b_1}$.

Cases of unexpectedly bad performance on application instances might be due to getting trapped in a similar structure during some derivations. Next, Section 4 proposes a modified clause learning procedure that avoids both exponential worst cases and tautologous resolvents. Example 4.2 shows that $[b_1]$ can be learned with a linear-length derivation. Hence, there is an opportunity for significant improvement in the efficiency of QBF clause learning. □

Performance problems of this kind can be avoided by accepting clauses with contradictory universal literals, a scheme dubbed *long-distance resolution* by Zhang and Malik [28]. Although they argued that the practice was sound *in the context of their particular solver*, there is no proof theory for it, in the sense of Cook and Reckhow [6]. According to Narizzano *et al.* [22], the version of `QuBE` used as the basis for `QuBE-cert` uses (some version of) long-distance resolution. [3]

[3] See `www.cse.ucsc.edu/~avg/ProofChecker/CheqTaut.tar.gz` for small instances on which `QuBE-cert` produces tautologous clauses in certificates.

Algorithm 1: Compute `qpup[`f`]`. Upon conflict, call `computeQpup(`$\bot$`)`.

```
   /* Precondition: f is implied (hence is existential or ⊥).          */
   /* Postcondition: qpup [f] is computed.                             */
 1 computeQpup (f)
 2 workCl = copy(antecedent[f])
 3 for (each implied q such that ¬q ∈ antecedent[f], in order of implication) do
 4     if (¬q == f) then
 5         continue
 6     if (qpup [q] == NULL) then
 7         computeQpup (q)
 8     workCl = res (¬q, workCl, qpup [q])
 9     for each tailing universal u_i ∈ workCl do
10         workCl = unrd (u_i, workCl)
11 qpup [f] = workCl
12 return
```

4 QBF Pseudo-Unit Propagation

Pseudo-unit propagation (PUP) was introduced for propositional clause learning [26] and found empirically to produce longer proofs than the 2005 version of `zchaffSE`, which uses the more common first UIP technique.

This section introduces ***QBF pseudo-unit propagation*** **(QPUP)**. QPUP can be used to derive a learned clause from a conflict graph. To focus on the main ideas, this section assumes that the pure literal rule is not in use. The combination of the pure literal rule with clause learning for QBF solving raises issues that are discussed in [13].

For QDPLL, it is assumed that the input clauses are non-tautological and that any tailing universal literals have been reduced away (Definition 2.6).

Definition 4.1 Let a QDPLL search be given, comprising a sequence of assumptions and unit-clause propagations. Each assumption is made upon a variable in the outermost quantifier block that contains a currently unassigned variable.

An ***implied literal*** is an existential literal that is assigned as a result of unit-clause propagation, including universal reductions. We call $\bot$ the implied literal when the search ends with a conflict. Each implied literal has an associated ***antecedent*** clause, which implied it. At the time e is implied, let `qpup[`e`]` be initialized to `NULL`.

In conjunction with the QDPLL search, suppose f is an implied literal with antecedent clause C. That is, all literals in C other than f are either assigned false, or they are universal literals that are tailing with respect to f (all universal literals are tailing with respect to $\bot$). Then the single-parameter partial function $qpup(f)$ is defined inductively for this search, as shown in Algorithm 1.

1. If C has no negations of implied literals, then $qpup(f) = C$. Note that C contains no unassigned universal literals tailing to f in this case.

2. If C contains $\overline{e_i}$ for $i = 1, \ldots, k$ and $k \geq 1$, where e_i are earlier implied literals in the same order as they were implied, then $qpup(f)$ is the result of successively resolving C with $qpup(e_i)$ for $i = 1, \ldots, k$. Applicable universal reductions are performed after each resolution step.

Note that f may be $\bot$. Then $qpup(\bot)$ contains only negated assumptions and non-tailing universal literals that have been assigned false by the search procedure. This is a conflict clause in the usual sense: this set of literals is inconsistent with the given formula. □

The *qpup* clauses may be computed lazily when a conflict occurs, or eagerly, as soon as a literal is implied. If computed eagerly, no recursive calls occur at lines 6 and 7 in `computeQpup`.

Example 4.2 The idea of QPUP is illustrated by an example, referring to the clauses in Figure 1. For simplicity, we assume *qpup* is computed eagerly. As in Example 3.1, b_1 is assumed. Next, b_1 implies f_{99}, but we define $qpup(f_{99}) = \left[f_{99}, \overline{b_1}\right]$ because b_1 is only an assumption. Similarly, b_1 implies e_{99} and we define $qpup(e_{99}) = \left[e_{99}, \overline{b_1}\right]$. Now f_9 is implied by unit-clause propagations on f_{99} and e_{99}, to start.

Unit-clause propagation can also be thought of as unit-clause resolution. If we had really derived f_{99} and e_{99} as unit clauses, we could resolve them with $\left[f_9, \overline{f_{99}}, \overline{e_{99}}, \overline{u_{10}}\right]$ to shorten that clause. Instead, we resolve $qpup(f_{99})$ and $qpup(e_{99})$ with this clause, then apply universal reduction, to get $qpup(f_9) = \left[f_9, \overline{b_1}\right]$. Notice that the introduced literal $\overline{b_1}$ cannot block the universal reduction because b_1 was an assumption, so it must have outer scope to any unassigned universal literals.

Continuing in this way, $qpup(p)$ and $qpup(\overline{p})$ are derived successively for $p = f_9, e_9, \ldots, d_1, c_1$. Finally, $\left[\overline{b_1}\right]$ is learned with a linear number of steps. □

Lemma 4.3 With $qpup(f)$ as in Definition 4.1, let q be the most recent assumption before f was implied. Then every existential literal in $qpup(f)$ *other than* f has scope outer to or equal to the scope of q.

Proof: The existential literals of $qpup(f)$ other than f are negations of assumptions, and assumptions are made in outer to inner order. ■

If cube processing is intermixed with unit-clause propagation, $qpup(f)$ can contain universal literals whose scopes are inner to q (the most recent assumption) and outer to f. These literals were assigned false by the search procedure. However, if $f = \bot$, there are no such literals in $qpup(f)$. When $f \neq \bot$, these universal literals cannot cause tautologous resolvents, because unit-clause propagation would satisfy any clause that might produce a tautologous resolvent. The results of QPUP are soundly derived clauses, each containing an implied literal plus negations of some earlier-assigned literals.

For a formula with n variables, there are at most n *qpup* clauses at any point in the search, and each *qpup* clause is derived with at most n steps. Only *qpup* clauses are used to derive $qpup(\bot)$, so the procedure to learn one clause is polynomial in n. The problem with Example 3.1 is that most derived clauses are not *qpup* clauses.

In summary, QPUP shows that conflict clauses can be learned by QDPLL with polynomially long derivations, although current implementations of QDPLL sometimes carry out exponentially long derivations. We hope this motivates the search for a better system of clause learning in QBF.

5 QU-Resolution

This section introduces ***QU-Resolution***. This is a natural extension of Q-resolution. Although Q-Resolution is refutationally complete, it cannot derive all logically implied clauses (see Definition 2.5). Since clause learning involves deriving logically implied clauses, and we saw in Section 3 that such derivations might be exponentially long, in terms of the overall number of clauses needed to derive the learned clause, it makes sense to look at variants of Q-resolution.

This section shows that QU-resolution is ***inferentially complete***, in the sense that if some (non-tautological) clause C is a super-sound addition to a closed QBF Φ, then some subset of C is derivable by QU-resolution. We also show that it provides a theoretical underpinning for previously reported failed-literal preprocessing in QBF [20, 27].

For propositional conflict-driven clause learning, it is known that the learned clause has a quadratically long derivation, in terms of the overall number of clauses needed to derive it, even when clause minimization and "volunteers" are included [23, 24]. (See cited papers; minimization and volunteers are not important to this paper.)

Definition 5.1 A ***QU-derivation*** is the same as a Q-resolution derivation (see Section 2), except that it includes resolutions in which the clashing literal is universal. Tautologous resolvents are still prohibited. A ***QU-refutation*** is a QU-derivation of the empty clause.

A ***regular*** QU-derivation is one such that no variable appears twice as a clashing literal or reduction literal on any directed path through the derivation DAG. An ***ordered*** QU-derivation is one such that variables appear in the same order on every directed path through the derivation DAG. (Not all variables need appear on a path; ordered derivations are necessarily regular). A ***prefix-ordered*** QU-derivation is an ordered QU-derivation such that variables appear in outer to inner order of the quantifier prefix on paths directed away from the root of the derivation DAG. □

Definition 5.2 If D is a clause, and clause $D' \subseteq D$ as a set of literals, then we say D' ***subsumes*** D. The notation $D^{(-)}$ means "*some* clause that subsumes D". That is, a statement "$D^{(-)}$ ..." should be read as "For *some* clause D' that subsumes D, D' ...". In general, the statement will not hold for all subsets of D. Recall that the empty clause and D itself are subsets of D. □

Lemma 5.3 If clause D is derived from $\Psi = \overrightarrow{Q}.\mathcal{F}$ by QU-Resolution, then D is logically implied by Ψ.

The proof of the next theorem employs the framework first published by Anderson and Bledsoe [1]. The original proof that Q-resolution is refutationally

complete uses the same idea [14], but that paper does not mention that the refutation may be required to have the additional properties of being prefix-ordered and regular.

Theorem 5.4 If (non-tautological) clause D is logically implied by $\Psi = \overrightarrow{Q}.\mathcal{F}$, then $D^{(-)}$ can be derived from Ψ by prefix-ordered QU-Resolution. If D is the empty clause then the QU-resolution can also be a Q-resolution.

Proof: Although many prefix orders may be equivalent, fix one for the proof.The proof is by induction on n, the number of variables in the quantifier prefix. The base case is $n = 0$. The conclusion is immediate, as D must be the empty clause and $\mathcal{F}$ must contain the empty clause to by hypotheses of the theorem. For $n > 0$, assume the claim holds for $m < n$ variables. There are several cases.

If literal $p \in D$ and $|p|$ is outermost, consider $\Psi_1 = \Psi\lceil_{\overline{p}}$ and $D_1 = D - \{p\}$. By the inductive hypothesis, $D_1^{(-)}$ has a QU-derivation from Ψ_1; call it π_1. Let π do the same proof operations as π_1, except starting with clauses in $\mathcal{F}$ instead of $\mathcal{F}\lceil_{\overline{p}}$. All operations remain correct because p is outermost. Each clause in π has at most an extra literal p, compared to the corresponding clause in π_1. Therefore π derives $D^{(-)}$.

If literals p and $\overline{p}$ are *not* in D and $|p|$ is outermost, consider $\Psi_0 = \Psi\lceil_p$ and $\Psi_1 = \Psi\lceil_{\overline{p}}$. If M_0 is any tree model of Ψ_0, then prepending $p = 1$ to every ordered assignment in M_0 gives a subset of some model of Ψ; call it M. By hypothesis of the theorem, D is true on every branch of M, so D is true on every branch of M_0. By the inductive hypothesis, $D^{(-)}$ has a QU-derivation from Ψ_0; call it π_0. (If Ψ_0 has no tree model, let π_0 be a Q-refutation of Ψ_0.) Let π do the same proof operations as π_0, except starting with clauses in $\mathcal{F}$ instead of $\mathcal{F}\lceil_p$. All operations remain correct because p is outermost. Each clause in π has at most an extra literal $\overline{p}$, compared to the corresponding clause in π_0. Therefore π derives $D_0 = (D \cup \{\overline{p}\})^{(-)}$ from Ψ. (If Ψ_0 has no tree model, $D_0 = (\{\overline{p}\})^{(-)}$.) Similarly, $D^{(-)}$ has a QU-derivation from Ψ_1; call it π_1; and $D_1 = (D \cup \{p\})^{(-)}$ can be derived from Ψ. If D_0 lacks $\overline{p}$, it serves as $D^{(-)}$ derived from Ψ. If D_1 lacks p, it serves as $D^{(-)}$ derived from Ψ. Otherwise, resolving D_0 and D_1 on p derives $D^{(-)}$.

If D is the empty clause and universal variable u is outermost, the refutation can be achieved with prefix-ordered Q-resolution. Let Ψ_0 and Ψ_1 be as defined in the previous paragraph with u in the place of p. By hypothesis of the theorem and the semantics of QBF (see Definition 2), either Ψ_0 is false or Ψ_1 is false. By the inductive hypothesis, either π_0 or π_1 derives the empty clause with Q-resolution. Then either the empty clause or $[u]$ or $[\overline{u}]$ can be derived from Ψ with Q-resolution. Universal reduction, if needed, completes the refutation. ■

The proof shows that the part of the derivation inner to the variables that occur in the target clause D can always use universal reduction in preference to resolution with a universal clashing literal (if preserving tree countermodels is not required).

Since QU-resolution is able to eliminate universal literals in some cases without using universal reduction and Example 2.8 showed that universal reduction

can change the set of tree countermodels, QU-resolution has a greater capability than Q-resolution to preserve tree countermodels, which might be important for certain applications that want to extract strategies for the A player.

Example 5.5 The following QBF family is given by Kleine Büning and Lettman [15] in the proof of their Theorem 7.4.8. The k-th QBF is:

$$\exists\, d_0\, d_1\, e_1 \forall\, x_1 \exists\, d_2\, e_2 \forall\, x_2 \cdots \exists\, d_k\, e_k \forall\, x_k \exists\, f_1 \cdots \exists\, f_k.$$

$$\begin{array}{rll}
\overline{[d_0]} & [d_j, \overline{x_j}, \overline{d_{j+1}}, \overline{e_{j+1}}] & \text{for } 1 \le j < k \\
[d_0, \overline{d_1}, \overline{e_1}] & [e_j, x_j, \overline{d_{j+1}}, \overline{e_{j+1}}] & \text{for } 1 \le j < k \\
[d_k, \overline{x_k}, \overline{f_1}, \ldots, \overline{f_k}] & [\overline{x_j}, f_j] & \text{for } 1 \le j \le k \\
[e_k, x_k, \overline{f_1}, \ldots, \overline{f_k}] & [x_j, f_j] & \text{for } 1 \le j \le k
\end{array}$$

That proof states that every Q-refutation of the k-th formula has at least 2^k steps.

With QU-resolution, begin by resolving the binary clauses on universal literals to produce unit clauses $[f_j]$ for $1 \le j \le k$. Then derive unit clauses $[d_k]$ and $[e_k]$ (using universal reduction on x_k and $\overline{x_k}$). The remainder is straightforward with Q-resolution. The overall number of steps is linear. □

Lonsing and Biere introduced *abstractions* of QBF for preprocessing purposes [20]. The idea is developed further in SAT 2012 [27]. Essentially the "abstraction" with respect to a variable p treats all universal variables outer to p as though they were existential. For inferential purposes, this amounts to using QU-resolution on these variables.

6 Depth-Monotonic Literals

This section addresses the question of when ***QBF variable-elimination resolution*** **(QVER)** is safe. QVER is the "resolve" operation in `Quantor`, described by Biere [3]. Variable-elimination resolution is the basic DP operation for CNF. For Q-resolution, add operation (4):

1. Find all resolvents with existential clashing literal q and add them to $\mathcal{F}$.
2. Discard tautologous resolvents.
3. Delete all clauses with q or $\overline{q}$.
4. Perform all universal reductions that are possible.

When is QVER safe? `Quantor` uses QVER for variables at innermost scope, but Biere did not prove that it is safe [3]. However, a proof may be found in [5, prop. 2.5.2]. In the cases that q is *not* in the innermost scope, the problem is step 3 above, deletion of clauses. Removal of constraints might change the value of a formula from false to true. We take a detour and return to this question.

A complementary question was considered by Bubeck and Kleine Büning. They considered universal expansion (the "expand" operation in `Quantor`, which eliminates a *universal* variable). They showed in their Theorem 1 [4] and Theorem 5.6.1 [5] that the operation can be simplified by not expanding existential variables that meet a certain criterion called a one-sided dependency.

Nondecisive clauses have been defined for propositional formulas. We extend this idea to QBF.

Definition 6.1 Let $\Phi = \overrightarrow{Q}.\mathcal{F}$ be a closed QBF with matrix $\mathcal{F}$, and let C be a clause in $\mathcal{F}$ that contains the literal q. C is ***depth-nondecisive*** on q with respect to $\mathcal{F}$ if for all clauses $D \in \mathcal{F}$ that contain $\overline{q}$ it is the case that at least one of the following conditions holds:

1. $\mathsf{res}_q(C, D)$ contains *no* literal with scope inner to q, even if the resolvent is tautologous; **or**
2. $\mathsf{res}_q(C, D)$ is tautologous due to the presence of literals $r \in C$ and $\overline{r} \in D$, where r is *not* inner to the scope of q; **or**
3. $\mathsf{res}_q(C, D)$ is subsumed by some clause C_1, where C_1 contains *no* literal with scope inner to q and either $C_1 \in \mathcal{F}$ or C_1 is a resolvent created under case (1) above.

The idea is that for any resolvent $R = \mathsf{res}_q(C, D)$, either R has no literals inner to q or R has some literals inner to q but even if those literals were deleted from R giving R', then R' could still be safely discarded because it is tautologous or subsumable. □

Example 6.2 In case (3) the subsumption by another resolvent using C does not imply that $\mathcal{F}$ already contains a subsumable clause. For example, let $C = [c, e]$, $D_1 = [c, d, \overline{e}]$, and $D_2 = [d, \overline{e}, f]$. Assume each variable has a different scope and literals are listed from outer to inner scope, which is also alphabetical order. No clauses are subsumed, and $\mathsf{res}_e(C, D_2)$ contains f, yet C is depth-nondecisive on e. □

Definition 6.3 Let $\Phi = \overrightarrow{Q}.\mathcal{F}$ be a closed QBF with matrix $\mathcal{F}$, which partitions into $\mathcal{G}_1 + \mathcal{G}_2 + \mathcal{G}_3$ ("+" denotes ***disjoint union*** in this section), and let q be an existential literal in Φ. Assume that:

1. $\mathcal{G}_1$ consists of clauses with q or $\overline{q}$ and *no* deeper variables. Deeper means strictly greater qdepth.
2. $\mathcal{G}_2$ consists of clauses with q or $\overline{q}$ and *some* deeper variables.
3. $\mathcal{G}_3$ consists of clauses without q or $\overline{q}$.

If every clause C in $\mathcal{G}_2$ that contains $\overline{q}$ is depth-nondecisive on $\overline{q}$ with respect to $\mathcal{F}$, then we say that q is ***effectively depth-monotonic*** in Φ. We say that the variable $|q|$ is effectively depth-monotonic in Φ if it holds for q or $\overline{q}$.

Clearly, if $|q|$ is pure in $\mathcal{G}_2$, then $|q|$ is effectively depth-monotonic in Φ. In this case we say that $|q|$ is ***depth-monotonic*** in Φ. □

Theorem 6.4 With q, Φ, $\mathcal{F}$, $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$ defined as in Definition 6.3, let $|q|$ be the clashing existential variable being contemplated for QVER on Φ, and let $|q|$ be effectively depth-monotonic in Φ. Then QVER on $|q|$ is safe.

Furthermore, let $\Phi_1 = \overrightarrow{Q}_1.(\mathcal{G}_3 \cup \mathcal{G}_4)$ be the result of QVER on $|q|$ and Φ. That is, $\overrightarrow{Q}_1$ is $\overrightarrow{Q}$ with $|q|$ deleted, and $\mathcal{G}_4$ consists of all non-tautologous resolvents with $|q|$ as the clashing variable using clauses in $\mathcal{G}_1 + \mathcal{G}_2$. Then every tree model M_1 for Φ_1 can be extended to a tree model M for Φ with a one-to-one correspondence between the branches of M_1 and the branches of M.

Proof: (Sketch[4]) By super-soundness of Q-resolution, it suffices to consider only the case that Φ_1 is `true`. For notation, let $A = \{\alpha\}$, $B = \{\beta\}$, $\Gamma = \{\gamma\}$, $\Delta = \{\delta\}$ denote the clause sets in $\overrightarrow{Q}_1$ corresponding to $\mathcal{G}_1$ and $\mathcal{G}_2$ as follows:

$$\mathcal{G}_1 = \{[q, \alpha] \mid \alpha \in A\} + \{[\overline{q}, \beta] \mid \beta \in B\}$$
$$\mathcal{G}_2 = \{[q, \gamma] \mid \gamma \in \Gamma\} + \{[\overline{q}, \delta] \mid \delta \in \Delta\}$$

Some α or β might be empty, but all γ and δ are nonempty. W.l.o.g., assume that q is effectively depth-monotonic in Φ.

Let M_1 be any tree model for Φ_1. Let (σ, τ) be any branch in M_1 where σ assigns values to all variables outer to $|q|$ and τ assigns values to all variables inner to $|q|$. Construct the corresponding branch in M as follows:

Condition on σ	**Corresponding Branch in M**
$\exists\, \beta \in B$ such that $\beta\lceil_\sigma \neq 1$, or $\exists\, \delta \in \Delta$ such that $\delta\lceil_\sigma \neq 1$	$(\sigma, \overline{q}, \tau) \in M$
$\forall\, \beta \in B$ $\beta\lceil_\sigma = 1$, and $\forall\, \delta \in \Delta$ $\delta\lceil_\sigma = 1$	$(\sigma, q, \tau) \in M$

(Note that $\delta\lceil_\sigma$ has some unassigned literals if it does not simplify to 1.)

By construction, M satisfies the requirement for tree models that, if (ρ, q) is a branch prefix of M, then there is no branch prefix of the form $(\rho, \overline{q})$, and *vice versa*. M satisfies $\mathcal{G}_3$, since M_1 satisfies $\mathcal{G}_3$. M satisfies all clauses in $\mathcal{G}_1 + \mathcal{G}_2$ that contain $\overline{q}$ by construction.

It remains to show that M satisfies all clauses in $\mathcal{G}_1 + \mathcal{G}_2$ that contain q. Assume $(\sigma, \overline{q}, \tau)$ is the branch in M corresponding to (σ, τ) in M_1, otherwise satisfaction is immediate.

First, consider an arbitrary clause $C = [q, \alpha] \in \mathcal{G}_1$. There is some $\beta \in B$ or $\delta \in \Delta$ that caused the assignment $q = 0$ in the construction of M.

(Subcase β) If $\beta\lceil_\sigma \neq 1$ and $[\alpha, \beta] \in \mathcal{G}_4$, then $\alpha\lceil_\sigma = 1$. If $\beta\lceil_\sigma \neq 1$ and $[\alpha, \beta] \notin \mathcal{G}_4$, it must be due to being tautologous. Then there is some literal r that is not inner to q such that $r \in \alpha$ and $\overline{r} \in \beta$. It follows that $\alpha\lceil_\sigma = 1$.

(Subcase δ) Suppose some δ causes the assignment $q = 0$. If $\delta\lceil_\sigma \neq 1$ and $[\alpha, \delta] \in \mathcal{G}_4$, then it must be subsumed by some $D_1 \in (\mathcal{G}_3 \cup \mathcal{G}_4)$ because $[\alpha, \delta]$ must have some literal inner to q. To qualify, D_1 must have no literals inner to q (case (3) in Definition 6.1). Some literal $p \in D_1$ is assigned 1 by σ. By the subcase hypothesis $p \notin \delta$, so again $\alpha\lceil_\sigma = 1$. If $\delta\lceil_\sigma \neq 1$ and $[\alpha, \delta] \notin \mathcal{G}_4$, the argument in subcase β applies with δ in the place of β.

Second, consider an arbitrary clause $C_2 = [q, \gamma] \in \mathcal{G}_2$. There is some $\beta \in B$ or $\delta \in \Delta$ that caused the assignment $q = 0$ in the construction of M. By arguments similar to the first case, $\gamma\lceil_\sigma = 1$. ■

Corollary 6.5 With q, Φ, $\mathcal{F}$, $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$ defined as of Theorem 6.3, if $|q|$ is depth-monotonic in Φ and the universal player has a winning strategy, then for each universal variable u_i that has inner scope to q, the winning strategy for u_i can be expressed as a function that is independent of q.

[4] See `http://www.cse.ucsc.edu/~avg/QPUP/qpup-cp12-long.pdf` for full proofs.

Proof: (Sketch) Place $|q|$ innermost in its quantifier block. W.l.o.g., let the pure literal be q for $\mathcal{G}_2$. All occurrences of u_i are in $\mathcal{G}_2$ or $\mathcal{G}_3$. Now consider the modified formula Φ_1 that differs from Φ only in that $|q|$ is outermost within the innermost quantifier block. It remains to show that Φ_1 is false.

Apply the quadrangle dependency theory of [25], Theorem 4.7. In Φ_1, $|q|$ does not have a quadrangle dependency with any u_i that is inner to $|q|$ in Φ, because paths from u_i or $\overline{u_i}$ to $\overline{q}$ require connections outer to u_i. Therefore, starting from Φ_1, $|q|$ can repeatedly be transposed with the next outer variable in the quantifier prefix without changing the value of the formula, until $|q|$ reaches the position it has in Φ, which is false by hypothesis. Since $|q|$ is inner to all u_i in Φ_1, if Φ_1 is false, then the universal player has a winning strategy such that the winning value for each u_i is independent of $|q|$. ■

Example 6.6 This example illustrates several topics from this section. Φ is shown in chart form.

Φ	$\exists q$	$\exists r$	$\forall u$	$\exists x$	$\exists y$
C_1	$\overline{q}$			$\overline{x}$	
C_2			$\overline{u}$	x	
C_3	q	$\overline{r}$			
C_4	q	r			y
C_5			u		$\overline{y}$

The winning strategy for u depends on q but not r; i.e., $u(q, r) = q$.

QVER(r) is sound. QVER(q) is not sound. Notice that q has no clauses in common with u, and neither does r. The difference lies in the paths from u or $\overline{u}$ to the various existential literals. Thus the two paths $C_2 - C_1$ and $C_5 - C_4$ establish a quadrangle dependency of q upon u. However, the only path from u or $\overline{u}$ to $\overline{r}$ involves a connection through q, but q is not inner to r, so this path does not qualify for establishing a quadrangle dependency.

By quadrangle dependency theory, r and u can be transposed without changing the truth value of Φ. After the transposition, r is innermost, so QVER(r) is sound and might be selected by `Quantor`. Since the formula that results after QVER(r) is the same, whether the transposition is carried out or not, an implementation need not actually perform the transposition. □

7 Conclusion

This paper presents theoretical analysis of several issues closely related to modern QBF solvers. Avenues for improvement in the state of the art are suggested. No QBF solvers with publicly available code were found to be very amenable to research experimentation, in contrast to MiniSat in the SAT domain. Therefore we leave it to various implementers to decide how to incorporate this paper's results into their own solvers.

Acknowledgment We thank Florian Lonsing, Mikolas Janota, Uwe Bubeck, and the reviewers for their careful reading of an earlier draft.

References

1. R. Anderson and W. W. Bledsoe. A linear format for resolution with merging and a new technique for establishing completeness. *JACM*, 17, 1970.
2. V. Balabanov and J. R. Jiang. Resolution proofs and Skolem functions in QBF evaluation and applications. In *Proc. CAV*, 2011.
3. A. Biere. Resolve and expand. In *SAT*, pages 238–246, 2004.
4. U. Bubeck and H. Kleine Büning. Bounded universal expansion for preprocessing QBF. In *Proc. SAT*, pages 244–257, 2007.
5. U. Bubeck. *Model-based Transformations for Quantified Boolean Formulas*. PhD thesis, University of Paderborn, 2010. (DISKI, 329, IOS Press; also at `http://www.ub-net.de/bubeck-qbf-transformations-2010.pdf`).
6. S. Cook and R. Reckhow. The relative efficiency of propositional proof systems. *J. Symbolic Logic*, 44:36–50, 1979.
7. U. Egly. On sequent systems and resolution for QBFs. In *Proc. SAT*, pages 100–113, 2012.
8. A. Goultiaeva and F. Bacchus. Exploiting QBF duality on a circuit representation. In *AAAI*, 2010.
9. A. Goultiaeva, Vicki Iverson, and Fahiem Bacchus. Beyond CNF: A circuit-based QBF solver. In *Proc. SAT*, pages 412–426, 2009.
10. A. Goultiaeva, A. Van Gelder, and F. Bacchus. A uniform approach for generating proofs and strategies for both true and false QBF formulas. In *Proc. IJCAI*, 2011.
11. E. Giunchiglia, M. Narizzano, and A. Tacchella. Learning for Quantified Boolean Logic Satisfiability. In *AAAI/IAAI*, pages 649–654, 2002.
12. E. Giunchiglia, M. Narizzano, and A. Tacchella. Monotone literals and learning in QBF reasoning. In *CP*, 2004.
13. E. Giunchiglia, M. Narizzano, and A. Tacchella. Clause/term resolution and learning in the evaluation of quantified boolean formulas. *JAIR*, 26, 2006.
14. H. Kleine Büning, M. Karpinski, and A. Flögel. Resolution for quantified boolean formulas. *Information and Computation*, 117:12–18, 1995.
15. H. Kleine Büning and T. Lettmann. *Propositional Logic: Deduction and Algorithms*. Cambridge University Press, 1999.
16. W. Klieber, S. Sapra, S. Gao, and E. Clarke. A non-prenex, non-clausal QBF solver with game-state learning. In *SAT, LNCS*, 2010.
17. F. Lonsing. *Dependency Schemes and Search-Based QBF Solving: Theory and Practice*. PhD thesis, Johannes Kepler University, 2012.
18. F. Lonsing and A. Biere. A compact representation for syntactic dependencies in QBFs. In *Proc. SAT*, pages 398–411. Springer, 2009.
19. F. Lonsing and A. Biere. Integrating dependency schemes in search-based QBF solvers. In *SAT*, pages 158–171. Springer, 2010.
20. F. Lonsing and A. Biere. Failed literal detection for QBF. In *Proc. SAT*, 2011.
21. R. Letz. Lemma and model caching in decision procedures for quantified boolean formulas. In *Proc. TABLEAUX (LNAI 2381)*, pages 160–175, 2002.
22. M. Narizzano, C. Peschiera, L. Pulina, and A. Tacchella. Evaluating and certifying QBFs: A comparison of state-of-the-art tools. *AI Commun.*, 22(4):191–210, 2009.
23. A. Van Gelder. Improved conflict-clause minimization leads to improved propositional proof traces. In *Proc. SAT*, pages 141–146, 2009.
24. A. Van Gelder. Generalized conflict-clause strengthening for satisfiability solvers. In *Proc. SAT (LNCS 6695)*, pages 329–342. Springer, 2011.

25. A. Van Gelder. Variable independence and resolution paths for quantified boolean formulas. In *Proc. CP*, pages 789–803, 2011.
26. A. Van Gelder. Producing and verifying extremely large propositional refutations: Have your cake and eat it too. *AMAI*, (accepted subject to revisions), 2012.
27. A. Van Gelder, S. B. Wood, and F. Lonsing. Extended failed literal detection for QBF. In *Proc. SAT*, 2012.
28. Lintao Zhang and Sharad Malik. Conflict driven learning in a quantified boolean satisfiability solver. In *Proc. ICCAD*, pages 442–449, 2002.