On Q-Resolution and CDCL QBF Solving

Mikoláš Janota

Microsoft Research, Cambridge, United Kingdom,
mikolas.janota@gmail.com

Abstract. The proof system Q-resolution and its variations provide the underlying proof systems for the DPLL-based QBF solvers. While (long-distance) Q-resolution models a conflict driven clause learning (CDCL) QBF solver, the inverse relation is not well understood. This paper shows that CDCL solving not only does not simulate Q-resolution, but also that this is deeply embedded in the workings of the solver. This contrasts with SAT solving, where CDCL solvers have been shown to simulate resolution.

1 Introduction

Conflict driven clause learning (CDCL) has been established as an efficient and practical method for SAT solving [26]. The relation between CDCL and propositional resolution has been extensively studied. It has been shown that, under various assumptions, modern SAT solvers simulate propositional resolution [4,27,2,5].

CDCL, with certain modifications, enables also solving quantified Boolean formulas (QBF) [30,11,22]. Analogously, propositional resolution also has its quantified counterpart *Q-resolution* [18] and a popular extension *long distance Q-resolution* [30,3]. As of today, the relation between CDCL solving and the underlying proof systems leaves a number of open questions.

The objective of this paper is to explore the relation between CDCL solving and (long-distance) Q-resolution. In particular, the paper gives a negative answer to the question whether a CDCL solver can simulate any Q-resolution proof. By this is meant that there is an infinite enumerable family of formulas $\Phi_1, \ldots, \Phi_n, \ldots$, for which there exist Q-resolution refutations polynomial in n but any CDCL solver requires computation time super-polynomial (or even exponential) in n. In fact, it is shown that not even tree-like Q-resolution can be simulated by CDCL.

QBF solving brings in the complication that propagation can also be performed on the universal variables, which is done with the aid of *solution driven cube learning (SDCL)*. When combined, CDCL and SDCL influence one another because order of propagation determines which clauses/cubes are learned. While the presented result focuses on CDCL solving, we show that it is still relevant in a restricted form for solving with the combined learning and some hints for future work are explored.

The paper is organized as follows. The paper reviews notation and basic concepts in Section 2. Section 3 reviews the family of formulas $\mathtt{CR}_n$, which are studied in the remainder of the paper. This section also builds on a previous result, which shows that level-ordered refutations of the said formula must be exponential [16]. Here we strengthen this result by showing that also any derivation of unit clauses must already be exponential (Section 3.1). This will let us study runs of the solver only until the first unit clause is derived. Section 4 studies properties of unit propagation on the studied family of formulas $\mathtt{CR}_n$. Section 5 uses these properties to show exponential behavior on $\mathtt{CR}_n$. Section 6 reviews a polynomial tree-like, ordered Q-resolution refutation of $\mathtt{CR}_n$. Finally, Section 7 discusses results of the paper and Section 8 concludes the paper.

Two distinctive proof techniques are used in the paper. The construction and motivation for the formula $\mathtt{CR}_n$ relies on the concept of two-player games perspective for QBF. This is somewhat similar to *Ehrenfeucht-Fraïssé games* [10] used in other domains (e.g. [21]). Second is to force the QBF solver to derive level-order proofs, which must be necessarily exponential on $\mathtt{CR}_n$. Further, we show that not only they are exponential for the whole formula but as early as a unit clause is derived. This greatly simplifies the lower-bound proof because propagation in a QBF solver is simpler with no unit clauses in its database.

2 Preliminaries

A *literal* is a Boolean variable or its negation. The literal complementary to a literal l is denoted as $\bar{l}$, i.e. $\bar{x} = \neg x$, $\overline{\neg x} = x$. A *clause* is a disjunction of zero or more non-complementary 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 literals. For a literal $l = x$ or $l = \neg x$, we write $\mathsf{var}(l)$ for x. For a clause C, we write $\mathsf{var}(C)$ to denote $\{\mathsf{var}(l) \mid l \in C\}$ and for a CNF ψ, $\mathsf{var}(\psi)$ denotes $\{l \mid l \in \mathsf{var}(\psi), C \in \psi\}$

A complementary concept to clause, is *cube*, which is a conjunction of zero or more non-complementary literals.

2.1 Quantified Boolean Formulas

Quantified Boolean Formulas (QBFs) [17] extend propositional logic by enabling quantification over Boolean variables. Any propositional formula ϕ is also a QBF with all variables *free*. If Φ is a QBF with a free variable x, the formulas $\exists x.\, \Phi$ and $\forall x.\, \Phi$ are QBFs with x *bound*, i.e. not free. Note that we disallow expressions such as $\exists x. \exists x.\, x$. Whenever possible, we write $\exists x_1 \dots x_k$ instead of $\exists x_1 \dots \exists x_k$; analogously for $\forall$. For a QBF $\Phi = \forall x.\, \Psi$ we say that x is *universal* in Φ and is existential in $\exists x.\, \Psi$. Analogously, a literal l is universal (resp. existential) if $\mathsf{var}(l)$ is universal (resp. existential).

The application of an assignment τ is defined for a QBF Φ if all variables of $\mathsf{dom}(\tau)$ are free in Φ, and, it is defined as $(\mathcal{Q}x.\, \Phi)\tau = \Phi\tau$ for $\mathcal{Q} \in \{\exists, \forall\}$. QBFs can be seen as compact representations of propositional formulas. In particular, the formula $\forall x.\,\Psi$ is satisfied by the same truth assignments as $\Psi[x{\to}0] \wedge \Psi[x{\to}1]$ and $\exists x.\,\Psi$ by $\Psi[x{\to}0] \vee \Psi[x{\to}1]$. Since $\forall x \forall y.\,\Phi$ and $\forall y \forall x.\,\Phi$ are semantically equivalent, we allow writing $\forall X$ for a set of variables X; analogously for $\exists$. A QBF with no free variables is *false* (resp. *true*), iff it is semantically equivalent to the constant 0 (resp. 1).

A QBF is *closed* if it does not contain any free variables. A QBF is in *prenex form* if it is of the form $\mathcal{Q}_1 X_1 \dots \mathcal{Q}_k X_k.\,\phi$, where $\mathcal{Q}_i \in \{\exists, \forall\}$, $\mathcal{Q}_i \neq \mathcal{Q}_{i+1}$, and ϕ is propositional. The propositional part ϕ is called the *matrix* and the rest the *prefix*. If a variable x is in the set X_i, we say that x is at *level* i and write $\mathsf{lv}(x) = i$; we write $\mathsf{lv}(l)$ for $\mathsf{lv}(\mathsf{var}(l))$.

We write QCNF for the class of QBFs in prenex form where the matrix is in CNF. Unless specified otherwise, QBFs are assumed to be closed and with CNF matrix.

2.2 Q-resolution

Q-resolution (Q-Res), by Kleine Büning et al. [18], is a resolution-like calculus that operates on QBFs in prenex form where the matrix is a CNF. The rules are given in Figure 1.

$$\frac{}{C}\ \text{(Axiom)} \qquad\qquad \frac{C_1 \cup \{x\} \qquad C_2 \cup \{\neg x\}}{C_1 \cup C_2}\ \text{(Res)}$$

C is a clause in the matrix. Variable x is existential. If $z \in C_1$, then $\neg z \notin C_2$.

$$\frac{C \cup \{u\}}{C}\ (\forall\text{-Red})$$

Variable u is universal. If $x \in C$ is existential, then $\mathsf{lv}(x) < \mathsf{lv}(u)$.

Fig. 1. The rules of Q-Res [18]

Long-distance resolution (LD-Q-Res) appears originally in the work of Zhang and Malik [30] and was formalized into a calculus by Balabanov and Jiang [3]. It merges complementary literals of a universal variable u into the special literal u^*. These special literals prohibit certain resolution steps. In particular, different literals of a universal variable u may be merged only if $\mathsf{lv}(x) < \mathsf{lv}(u)$, where x is the resolution variable. The rules are given in Figure 2. In practice, solvers do not maintain a literal of the form u^* but rather two complementary literals, e.g. $e \vee u \vee \neg u \vee z$. Such clauses arise naturally by learning and give propagation due to universal reduction (see below). An alternative formulation via no-goods is suggested by Klieber [20,19].

For a clause C, a universal literal $l \in C$ is *blocked* by an existential literal $k \in C$ iff $\mathsf{lv}(l) < \mathsf{lv}(k)$. *$\forall$-reduction* is the operation of removing from a clause C all universal literals that are *not* blocked by some literal.

For a QCNF $\mathcal{P}\,.\,\phi$, a *Q-resolution proof* of a clause C is a sequence of clauses $C_1, \dots, C_n$ where $C_n = C$ and any C_i in the sequence is part of the given

$$\frac{}{C}\ \text{(Axiom)} \qquad \frac{D \cup \{u\}}{D}\ (\forall\text{-Red}) \qquad \frac{D \cup \{u^*\}}{D}\ (\forall\text{-Red}^*)$$

C is a clause in the matrix. Literal u is universal and $\mathsf{lv}(u) \geq \mathsf{lv}(l)$ for all $l \in D$.

$$\frac{C_1 \cup U_1 \cup \{x\} \qquad C_2 \cup U_2 \cup \{\neg x\}}{C_1 \cup C_2 \cup U}\ \text{(Res)}$$

Variable x is existential. If for $l_1 \in C_1, l_2 \in C_2$, $\mathsf{var}(l_1) = \mathsf{var}(l_2) = z$ then $l_1 = l_2 \neq z^*$. U_1, U_2 contain only universal literals with $\mathsf{var}(U_1) = \mathsf{var}(U_2)$. For each $u \in \mathsf{var}(U_1)$ we require $\mathsf{lv}(x) < \mathsf{lv}(u)$. If for $w_1 \in U_1, w_2 \in U_2$, $\mathsf{var}(w_1) = \mathsf{var}(w_2) = u$ then $w_1 = \neg w_2$, $w_1 = u^*$ or $w_2 = u^*$. U is defined as $\{u^* \mid u \in \mathsf{var}(U_1)\}$.

Fig. 2. The rules of LD-Q-Res [3]

matrix ϕ; or was obtained from one of the preceding clauses by $\forall$-reduction; or it is a Q-resolvent of some pair of preceding clauses. A Q-resolution proof is called a *refutation* iff C is the empty clause, denoted $\bot$.

In this article Q-resolution and plain resolution proofs are treated as connected directed acyclic graphs (DAG). Any graph representing a resolution or Q-resolution proof has one and only one node with in-degree 0, which we call the *root* (the final clause in the proof). All the nodes with out-degree 0 are labeled with clauses from the original formula and we call them *leafs* of the proof or *axiom* clauses. Any non-axiom clause has two outgoing edges pointing to its respective antecedents.

A Q-resolution proof is called *tree-like* if the corresponding graph forms a tree (rooted in the final clause). It is known that at the propositional level, tree resolution does *not* p-simulate DAG resolution [7].

A Q-resolution proof is called *ordered*, if there exists a sequence of variables S such that for any path from a leaf to the root the sequence of variables being eliminated form a sub-sequence of S. Sometimes ordered propositional resolution is referred to as Davis-Putnam resolution as it corresponds to the Davis-Putnam algorithm. It is known that at the propositional level, ordered resolution does *not* p-simulate unrestricted resolution [13,1].

Definition 1 (level-ordered proof). *Let π be a Q-resolution proof of a QCNF formula Φ. We say that π is level-ordered iff the following holds. Consider an arbitrary path from the root to some leaf and some resolution steps on that path on literals x_1 and x_2 such that the resolution on x_1 is closer to the root. Then,* $\mathsf{lv}(x_1) \leq \mathsf{lv}(x_2)$.

Remark 1. Janota and Silva define *level-order refutations* [16], which the above definition generalizes for an arbitrary proof.

2.3 CDCL and SDCL Solving

Basic understanding of CDCL+SDCL QCNF solving is assumed; for further details see [12,22]. We assume that a solver's state is uniquely determined by a sequence of *decisions* $\mathcal{D}$, a *trail* $\mathcal{T}$, and a *database of clauses and cubes*. The input formula is always in the database of clauses.

For the purpose of this paper we only assume CDCL, i.e. universal variables are handled by traditional chronological backtracking and there are no cubes in the database. SDCL is also briefly discussed for completeness.

The trail $\mathcal{T}$ and decisions $\mathcal{D}$ are modeled as sequences of literals, where $\mathcal{D}$ is a subsequence of $\mathcal{T}$. Any literal l in $\mathcal{T}$ but not in $\mathcal{D}$ is said to be *propagated*, and is obtained by unit propagation from $\mathcal{D}$. For a literal l we write $\mathcal{T}(l)$ to denote the value of l in $\mathcal{T}$, i.e. $\mathcal{T}(l) = 1$ if $l \in \mathcal{T}$ and $\mathcal{T}(l) = 0$ if $\bar{l} \in \mathcal{T}$.

CDCL A *conflict* is reached whenever unit propagation gives an existential literal two opposing values. This corresponds to violating a clause in the database. Upon a conflict, *clause learning* is invoked. Learning performs Q-resolution steps in the reverse chronological order of propagations. We assume that any $\forall$-reduction is carried out as soon as possible.

The learning process stops when the derived clause C fulfills the *unique implication point* property, which for QBF has the following conditions [30]. There exists a literal $l \in C$ such that all the following properties are fulfilled:

1. l is existential and it has the highest decision level in C.
2. l is at a decision level where the decision is an existential literal.
3. All universal literals with quantification level smaller than l are decided at a decision level smaller than l.

CDCL QBF solving is simulated by long-distance Q-resolution. More precisely, if a solver decides given formula as false, a long-distance Q-resolution refutation can be constructed for it in time polynomial to the running time of the solver.

Certain propagation can yield long-distance resolution steps, which can be avoided by modifying the UIP scheme [11], this however potentially leads to exponential blowup [28]. For the purpose of this paper, this difference is not important as we later see that long-distance steps do not occur in the considered formula.

SDCL Solution driven cube learning (SDCL) is symmetrical to CDCL with the difference that the initial cube database is empty and new cubes are also created when the current assignment gives a model to the initial formula. Hence, in SDCL initially the solver has an empty database of cubes. Once it reaches an assignment μ that satisfies the original matrix, it uses it to generate a cube. This is done by conjoining the literals that form the assignment μ. The cube is then existentially reduced and inserted into the database. Such cube is from now on used for propagation. If a cube is fully satisfied by the current assignment, this

a_1	• • •	a_1	• • •	a_N	• • •	a_N
b_1	• • •	b_N	• • •	b_1	• • •	b_N

Fig. 3. Completion principle

constitutes a losing situation for the universal player. Hence, unit cubes are used for propagation and fully satisfied cubes to jumpstart learning just as conflicting clauses do. See [22] for further details. We note that symmetrical approaches to SDCL+CDCL solving exist [29,14].

Model for the Paper For the results in the paper we are assuming a CDCL solver with *no* SDCL. This means that the solver makes decisions along the quantifier prefix until it either reaches a conflict and then it applies clause learning, or, it satisfies the matrix and then it backtracks to the last universal variable and flips its polarity (if such exists). We do not make any assumptions about restarts, i.e., restarts may be issued arbitrarily.

3 Formula

The formula used to show the main result is taken from Janota and Silva [16].[1] Its construction derives from a principle named *completion principle*. Two sets are considered: $A = \{a_1, \ldots, a_n\}$, $B = \{b_1, \ldots, b_n\}$ and their Cartesian-product $A \times B$. Let us visualize the cross-product as in Figure 3. The following game is played. In the first round, the $\exists$-player deletes one and only one cell from each column. In the second round, the $\forall$-player chooses one of the two rows. The $\forall$-player wins if the chosen row contains either the complete set A or the set B.

There is the following winning strategy for the $\forall$-player. If the $\exists$-player wants to make sure that the bottom row (the b_i row) does *not* contain the complete set B, it must delete at least one element from each of the n copies of B. Hence, for the j-th copy of B there is an element a_j that was not deleted and thus forming the complete set A. Hence, the winning strategy for the $\forall$-player is to look at the bottom row and see if it contains a complete copy of B. If it does, the $\forall$-player selects the bottom row and otherwise he selects the top row.

Let us construct a formula based on this principle. For each column (i, j) introduce a variable x_{ij} that determines which cell is deleted by the $\exists$-player in the first round. For the $\forall$-player introduce a single universal variable z, which determines the selected row. Further, add clauses that make sure that whenever one of the sets A or B is complete, the formula becomes false.

In the remainder of the paper we denote the set of variables x_{ij}, $i, j \in 1..n$ by $\mathcal{X}$, and a_i, b_i, $i \in 1..n$ by $\mathcal{L}$ (the letter $\mathcal{L}$ is chosen to evoke "last").

[1] See `http://sat.inesc-id.pt/~mikolas/cdcl16` for a formula generator.

The formula $\texttt{CR}_n$ is defined by the prefix $\exists\,\mathcal{X}\,\forall z \exists\,\mathcal{L}$ and the following matrix.

$$x_{ij} \lor z \lor a_i,\; i, j \in 1..n \tag{1}$$

$$\neg x_{ij} \lor \neg z \lor b_j,\; i, j \in 1..n \tag{2}$$

$$\bigvee_{i \in 1..n} \neg a_i \tag{3}$$

$$\bigvee_{i \in 1..n} \neg b_i \tag{4}$$

The first two types of clauses (1) and (2) represent the effects of the moves. The last two clauses (3) and (4) disable setting all a_i and b_i to true, respectively. Hence, the whole formula $\texttt{CR}_n$ is false because once z is set according to the strategy outlined above, the variables $\mathcal{L}$ must be set such that variables from one of the sets a_i and b_i will be all true. Consequently, one of the clauses (3) and (4) must be falsified.

3.1 Lower Bounds for Level-ordered Q-Resolution

An exponential lower-bound for level-ordered resolution for $\texttt{CR}_n$ is known [16].

Proposition 1. **[16, Prop. 5]** *Any level-ordered Q-resolution refutation of* $\texttt{CR}_n$ *is exponential in* n.

For the purpose of this paper a stronger result is needed. Here we show that any level-ordered proof of any *unit clause* is already exponential.

Lemma 1. *Let* π *be a level-ordered Q-resolution proof from* $\texttt{CR}_n$*, where* π *is a proof of a unit clause* $\{l\}$ *with* $\mathsf{var}(l) \in \mathcal{L}$*. Let* P *be a path from the root of* π *to some clause* C *where* P *does not contain any resolutions on* $\mathcal{L}$*. Let* $\pi_C \subseteq \pi$ *be the proof of* C*.*

Then, π_C *contains one of the clauses* (3)*,* (4)*.*

Proof. We consider the following cases:

1. π_C contains at least one resolution step on $\mathcal{L}$. Then one of (3), (4) must appear in π_C since they are the only clauses containing negative occurrences of the $\mathcal{L}$ variables.
2. π_C does not contain any resolution step on $\mathcal{L}$. Then, if there are no clauses (3), (4), than all axiom clauses are of type (1) or (2). This would be a contradiction with the requirement that π proves a unit clause since the variable z would always remain blocked as there are no more $\mathcal{L}$ resolutions on the path from the root of π to C.

$\square$

Lemma 2. *Let* π *be a level-ordered Q-resolution proof from* $\texttt{CR}_n$*, where* π *is a proof of a unit clause* $\{l\}$ *with* $\mathsf{var}(l) \in \mathcal{L}$*. Let* P *be a path from the root of* π *to some clause* C *where* P *does not contain any resolutions on* $\mathcal{L}$ *and* P *is maximal in that respect.*

Then, C *contains at least* $n - 1$ *different* $\mathcal{X}$ *variables.*

Proof. Observe that all $\mathcal{L}$ variables are treated symmetrically in the formula so without loss of generality, consider $l \in \{a_k, \neg a_k\}$ for some $k \in 1..n$.

Let $\pi_C \subseteq \pi$ be the proof of C. From Lemma 1, one of the clauses (3), (4) must appear in π_C.

Let us assume that (3) is in π_C. Since π is level-ordered and it derives the clause $\{l\}$, all a_i with $i \in 1..n$, $i \neq k$ must be resolved away in π_C. This means that π_C contains the clause $x_{ij} \vee z \vee a_i$ for each $i \in 1..n$, $i \neq k$. Since π_C has no resolutions on $\mathcal{X}$, the corresponding $\mathcal{X}$ variables also appear in C.

If (4) is in π_C, the reasoning is analogous to the above. $\square$

Proposition 2. *Let π be a level-ordered Q-resolution proof from $\texttt{CR}_n$, where π is a proof of a unit clause $\{l\}$ with $\mathsf{var}(l) \in \mathcal{L}$. Then π is exponential in n.*

Proof. Pick an assignment τ to all the $\mathcal{X}$ variables. Construct a path P from the root of π that contains $\forall$-reductions and $\mathcal{X}$ resolutions only and is maximal in that respect such that it respects the assignment τ.

Due to Lemma 2, P ends in a clause C containing at least $n-1$ $\mathcal{X}$ variables. There are 2^{n^2} assignments to $\mathcal{X}$ variables and C covers at most $2^{n^2-(n-1)}$ of those. Hence, there are at least $2^{n^2}/2^{n^2-(n-1)} = 2^{n-1}$ clauses in π. $\square$

Proposition 3. *Let π be a level-ordered Q-resolution proof from $\texttt{CR}_n$, where π is a proof of a unit clause $\{l\}$ with $\mathsf{var}(l) \in \mathcal{X}$. Then π is exponential in n.*

Proof. Since x_{ij} and $\neg x_{ij}$ are treated symmetrically in $\texttt{CR}_n$, any level-ordered sub-exponential proof of x_{ij} could be rewritten to a level-ordered sub-exponential proof of $\neg x_{ij}$. Resolving the two would give a level-ordered sub-exponential refutation of $\texttt{CR}_n$, which would be a contradiction with Proposition 1. $\square$

Corollary 1. *Let C be a unit or empty clause with a level-ordered Q-resolution proof π from $\texttt{CR}_n$. Then π is exponential with respect to n.*

4 Properties of Propagation on $\texttt{CR}_n$

To be able to reason about the clauses that are learned during solving of $\texttt{CR}_n$, several properties of unit propagation are needed. A crucial property of the input formula is that there are no clauses enabling propagation "across quantification levels". Namely, while decisions are being made on the $\mathcal{X}$ variables, no propagation happens in the $\mathcal{L}$ variables. These get value only once z gets a value. For this purpose we introduce the concept of mixed clauses.

Definition 2 (mixed clause). *We say that a clause is mixed if it contains both $\mathcal{X}$ variables and $\mathcal{L}$ variables, i.e., if $\mathsf{var}(C) \cap \mathcal{X} \neq \emptyset$ and $\mathsf{var}(C) \cap \mathcal{L} \neq \emptyset$.*

The following lemma shows that Q-resolution does not enable us to derive mixed clauses without the variable z.

Lemma 3. *Let π be an arbitrary Q-resolution proof from $\texttt{CR}_n$. For any mixed clause $C \in \pi$ it holds that $z \in \mathsf{var}(C)$.*

Proof (By induction on the derivation depth.). The hypothesis holds for all the axiom clauses of $\mathtt{CR}_n$.

Let C be a new mixed clause derived by resolution from some clauses D_1 and D_2, for which the hypothesis holds. Since C is mixed, at least one of D_1, D_2 must be mixed. Therefore C also contains z.

Let C be derived by $\forall$-reduction from an existing clause D. Since $\mathcal{L}$ variables block $\forall$-reduction of z, the clauses C and D do not contain $\mathcal{L}$ variables and therefore are not mixed. □

Remark 2. The above-lemma can be easily generalized. Indeed, if the input formula only contains mixed clauses that have a universal variable in the middle, that variable cannot be reduced unless all the variables at the higher quantification level are resolved away.

The following lemma is crucial for our result. As long as there are no unit clauses, there cannot be propagation across quantification levels since all the mixed clauses need z to have a value to give propagation.

Lemma 4. *Let $\mathcal{T}$ be the trail for a CDCL solver in a state before any unit clauses are learned. If $\mathsf{var}(\mathcal{T}) \subseteq \mathcal{X}$ then there is no propagation on $\mathcal{L}$ variables.*

Proof. Since $\mathtt{CR}_n$ does not contain any unit clauses and no unit clauses have been learned so far, any propagation on $\mathcal{L}$ must come from a clause in the database that has at least two literals. Since $\mathsf{var}(\mathcal{T}) \subseteq \mathcal{X}$, such clause would have to contain an $\mathcal{X}$ variable. However, due to Lemma 3, any mixed clauses contain also the z variable, which is unassigned and therefore cannot give a unit $\mathcal{L}$ clause. There is no propagation on z since we're assuming only CDCL (not SDCL). □

5 Exponential Lower Bound for CDCL QBF Learning

This section shows that a run of a CDCL solver on the formula $\mathtt{CR}_n$ is exponential. This is done by showing that the proofs of all learned clauses are level-ordered. Due to Corollary 1, it is sufficient to show that the proofs of learned clauses are level-ordered until a unit clause is learned. Indeed, even if the solver derives some non-level-ordered proofs after the first unit clause has been learned, the proof of the unit clause is already exponential. Therefore, the solver must perform an exponential number of steps to derive the unit clause. Note also that the input formula has no unit clauses as long as $n \geq 2$.

Note that the reasoning below needs to account for all the clauses derived during learning, not just the learned clauses. We start by a couple of technical lemmas characterizing the learning process.

Lemma 5. *Let $\mathcal{T}$ be the trail for a CDCL solver in a state before any unit clauses are learned. Let $\mathcal{T}$ be such that it leads to a conflict and let C be some clause that is derived during the learning of the pertaining learned clause. If $l \in C$ with $\mathsf{var}(l) = z$, then $\mathcal{T}(l) = 0$.*

Proof. By construction, C is derived by resolution steps on clauses that participated in the propagation leading to the conflict.

While z is not assigned, propagation is only on clauses that contain $\mathcal{X}$ variables only because any clauses that contain also some $\mathcal{L}$ variables also contain z due to Lemma 3.

Once z is assigned, all clauses that contain a literal l with $var(l) = z$ and $\mathcal{T}(l) = 1$ cannot be used during propagation (they are effectively deleted from the propagation process). □

The above lemma immediately gives us the consequence that there are no long-distance resolution steps in learned clauses.

Corollary 2. *For the formula* CR_n*, a CDCL solver never learns clauses derived by long-distance Q-resolution steps.*

Lemma 6. *Let $\mathcal{T}$ be a trail for a CDCL solver in a state before any unit clauses are learned. Let $\mathcal{T}$ be such that it leads to a conflict and let C be some clause that is derived during the learning of the pertaining learned clause.*

Let C be such that it does not contain any propagated *$\mathcal{L}$ literals, but it contains some $\mathcal{L}$ literals. Then C is a UIP.*

Proof. Due to the precondition, all $\mathcal{L}$ are decisions in C therefore there must be one literal k with $\mathsf{var}(k) \in \mathcal{L}$ with the highest decision level as all $\mathcal{L}$ variables are decided after $\mathcal{X}$ variables due to Lemma 4. From Lemma 5, if C contains the variable z, then the corresponding literal is set to 0. This fulfills the conditions for C be a UIP. □

Proposition 4. *Let $\mathcal{T}$ be the trail for a CDCL solver in a state before any unit clauses are learned. Let $\mathcal{T}$ be such that it leads to a conflict and let C be some clause that is derived during the learning of the pertaining learned clause.*

If C contains any $\mathcal{L}$ variables, then it is derived by resolution steps only on the $\mathcal{L}$ variables.

Proof. The hypothesis is trivially true for all the axiom clauses.

If $\mathsf{var}(\mathcal{T}) \subseteq \mathcal{X}$ then the derivation of the learned clause only contains $\mathcal{X}$ variables due to Lemma 4.

If $\mathsf{var}(\mathcal{T}) \cap (\mathcal{L} \cup \{z\}) \neq \emptyset$ then consider the following cases:

1. C contains some $\mathcal{L}$ literal, forced to 0 by propagation. Then, resolution is performed on one such literal as propagation on $\mathcal{L}$ takes place later chronologically than propagation on $\mathcal{X}$ variables due to Lemma 4.
2. C does not contain any $\mathcal{L}$ literal, then the hypothesis is trivially satisfied.
3. C does not contain any *propagated* $\mathcal{L}$ literal, then C is a UIP due to Lemma 6.

□

Finally we need to show that as long as clauses containing $\mathcal{L}$ variables are only derived by resolution steps on $\mathcal{L}$, the corresponding proofs are level-ordered. For such we utilize the following two observations.

Observation 1 *Let C be derived by a resolution step over an $\mathcal{X}$ variable from clauses with level-ordered proofs. Then the proof of C is also level-ordered.*

Observation 2 *A proof that contains resolution steps only on $\mathcal{L}$ variables is level-ordered.*

Proposition 5. *Let $\mathcal{T}$ be the trail that leads to a conflict for a CDCL solver in a state before any unit clauses are learned. The proof of the corresponding learned clauses is level-ordered.*

Proof. Prove from Proposition 4 by induction on derivation depth.

The hypothesis is trivially true for axiom clauses and is trivially preserved by $\forall$-reduction. Split on the following two cases.

1. If a clause C is derived by resolution on a $\mathcal{L}$ variable, then both antecedents must contain at least one $\mathcal{L}$ variable, From Proposition 4, the antecedents are derived only by $\mathcal{L}$ resolutions only. Therefore, the proof is level-ordered (Observation 2).
2. If a clause C is derived by resolution on an $\mathcal{X}$ variable, then the derivation of C is level-ordered because the antecedents are level-ordered (IH) and due to Observation 1.

□

Remark 3. Proving that learned clauses are themselves level-ordered is not itself inductive. The solver could learn a clause containing an $\mathcal{L}$ variable while using $\mathcal{X}$ resolutions and then use this learned clause in a level-ordered manner.

Theorem 1. *Solving $\texttt{CR}_n$ by a CDCL QBF solver requires time exponential in n.*

Proof. Due to Proposition 5, proof of any learned clause is level-ordered while no unit clauses are learned. Consider the first unit clause learned by the solver. Due to Corollary 1, the Q-resolution proof of the clause is exponential in n. Therefore, the solver must have carried out an exponential number of steps to learn this clause.

Observe that restarting the solver does not affect in any way the above results. Indeed, the whole proof hinges on the fact that learned clauses are always level-ordered independently of the content of the current trail—as long as the trail observes the condition that a variable is decided only if all the variables that precede it in the prefix are valued. More broadly speaking, restarts in QBF do not change the fact that the solver has to assign variables according to quantification levels; unlike in SAT, where restarts can come up with an arbitrary order.

Note that the proof relies on the UIP learning scheme (Lemma 6). we conjecture, however, that this precondition can be weakened. Other schemas, like QPUP [28,24] could be considered.

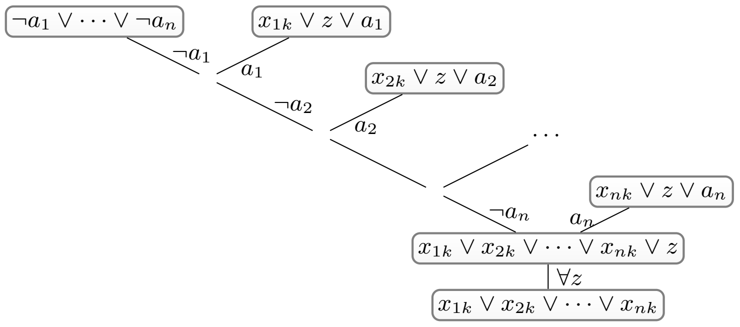
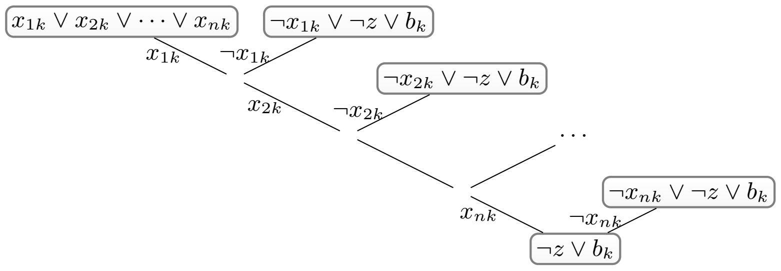


Fig. 4. Derivation of an "all x" clause for $k \in 1..n$ in n resolution steps.



$x_{1k} \vee x_{2k} \vee \cdots \vee x_{nk}$
$\neg x_{1k} \vee \neg z \vee b_k$
x_{1k}
$\neg x_{1k}$
$\neg x_{2k} \vee \neg z \vee b_k$
x_{2k}
$\neg x_{2k}$
$\ldots$
$\neg x_{nk} \vee \neg z \vee b_k$
x_{nk}
$\neg x_{nk}$
$\neg z \vee b_k$

Fig. 5. Derivation of an $\neg z \vee b_k$ clause for $k \in 1..n$ in n steps, using Figure 4.

6 Short Tree-like Q-resolution Refutation of $\mathtt{CR}_n$

This section shows that $\mathtt{CR}_n$ has a polynomial tree-like Q-resolution refutation. We do so in a constructive manner and the proof is conceptually divided into three parts.

First, derive clauses $\bigvee_{i\in 1..n} x_{ik}$ for $k \in 1..n$ (Figure 4). Each of these clauses lets us derive a clause $\neg z \vee b_k$ for $k \in 1..n$ (Figure 5). Finally, using the clause $\neg b_1 \vee \cdots \vee \neg b_n$, the empty clause is derived (Figure 6).

The first phase requires n^2 resolution steps and n $\forall$-reduction steps. The second phase requires n^2 resolution steps. Finally, the last phase requires n resolution steps and one $\forall$-reduction. Since the size of the input formula has $2n^2 + 2$ clauses, the proof size is linear in the formula's size.

Observe that each clause appears exactly once in the proof—the proof forms a tree. Also note that the proof is *not* level-ordered because it starts with resolutions on a_i variables, continues with $\mathcal{X}$ resolutions, and finishes with resolutions on b_i variables. However, the proof is *ordered* with the following order.

$$a_1, \ldots, a_n, x_{(1,1)}, \ldots, x_{(n,n)}, b_1, \ldots, b_n$$

Theorem 2. *$\mathtt{CR}_n$ has a polynomial refutation in ordered tree-like Q-resolution.*

Corollary 3. *Level-ordered resolution does not simulate ordered or tree-like Q-resolution.*

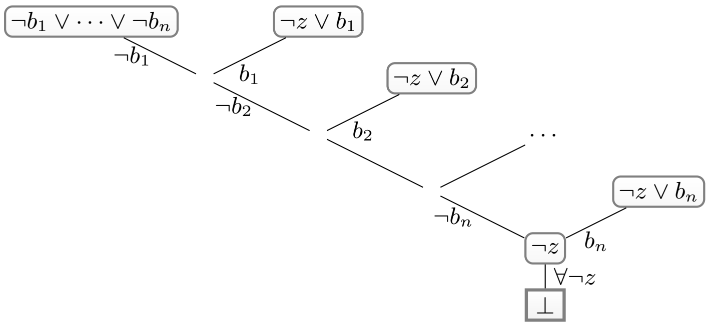


Fig. 6. Derivation of $\bot$, using Figure 5.

Corollary 4. *CDCL QBF solving does not simulate Q-resolution even under the restriction that the Q-resolution proofs are tree-like and ordered.*

Remark 4. Mahajan and Shukla in fact show that level-ordered Q-resolution and tree-like Q-resolution are incomparable [25]. In fact, the short resolution proof above also appears in their paper.

7 Discussion

QBF solving presents us with some subtleties and complications due to the two types of propagation and learning. Here we discuss how these relate to the presented results.

7.1 SDCL

The presented result is concerned only with CDCL and so we may ask whether SDCL can speed up the solving of $\mathtt{CR}_n$. A pivotal point in our proof is Lemma 4, which shows that there is no propagation from $\mathcal{X}$ variables to $\mathcal{L}$ variables, i.e., propagation across levels. The lemma relies on the fact that z will not be given a value by propagation from $\mathcal{X}$ variables. This can happen if SDCL is employed. More precisely, if the solver learns a cube containing only a subset of $\mathcal{X}$ variables and z, the variable z may be given a value *before* all $\mathcal{X}$ variable are assigned, which may subsequently give propagation on $\mathcal{L}$ variables. Such propagation could potentially lead to learned clauses with non-level-ordered proofs.

However, this can only happen when the universal player made a *wrong* choice for the value of z in the past. Since there is a winning strategy for the universal player, there always is a value for z that does not lead to a solution and consequently no cube learning takes place if the universal player follows the strategy.

Observation 3 *For a false QBF Φ, if universal variables are given values according to a winning strategy for the universal player, a SDCL+CDCL QBF*

n	CDCL	CDCL + SDCL	CDCL + SDCL − pure lits.
4	101	101	101
5	1081	1081	751
6	19611	19611	3531
7	370811	370811	36411
8	$>$ 9995451	$>$ 10000981	5464551
9	$>$ 10612011	$>$ 10619361	$>$ 931211
10	$>$ 10303551	$>$ 10313901	$>$ 8608251

Table 1. Number of backtracks for DepQBF on $\mathtt{CR}_n$ with a 1-hr. timeout. *Unsolved* instances are marked with $>$.

solver behaves identically to a CDCL QBF solver. Consequently, the Corollary 4 also holds for a SDCL+CDCL QBF solver under such restriction.

7.2 Pure Literals

Another relevant technique is *pure literals* [8,22]. Those enable assigning values to variables out of the quantification order. This can again influence what kind of clauses and cubes are learned. However, there is also a potential adversarial effect of pure literals. If the universal player makes better choices, it learns fewer cubes—which could have otherwise potentially speed up the proof.

While at this point we do not have a definite answer to the above questions, experimental evaluation might provide some hints. I have run the solver DepQBF [23] on $\mathtt{CR}_n$ and recorded the number of backtracking steps—these are presented for various configurations in Table 1. All configurations have the switches `--traditional-qcdcl --long-dist-res --dep-man=simple` to disable advanced features of DepQBF but also allow long-distance Q-resolution. The leftmost configuration only performs CDCL, the middle CDCL+SDCL, and the last one also combines CDCL and SDCL but switches off the pure-literal technique.

Instances that were not solved within 1 hour, are marked with $> B$ where B is the number of backtracking steps performed up to that point.

The configuration CDCL and CDCL+SDCL behave identically and can only solve $\mathtt{CR}_n$ for $n \in 1..7$. Interestingly enough, turning off pure literals leads to a significant improvement and also $n = 8$ is solved.

Further inspection reveals that the CDCL+SDCL configuration never learned any cubes. Because, as indicated above, it never makes a wrong decision for the universal variable z. Somewhat paradoxically, this is disadvantageous.

7.3 Other Work on Separation

In his thesis, Lonsing presents a formula that shows that Q-resolution is more powerful than standard CDCL solving—see Example 3.3.6 in [22]. The formula is of the form $\exists X \forall U \exists Z.\, \phi_1 \wedge \phi_2$, where ϕ_1 depends only on the X variables and it is a hard unsatisfiable propositional formula, e.g. pigeonhole principle [15]. The formula ϕ_2 depends on U and Z and is easy to refute. While a Q-resolution proof can simply use ϕ_2 to construct a refutation, a CDCL solver will try to construct an assignment satisfying ϕ_1, which is impossible and will take an exponential amount of time. As discussed in Lonsing's work, this can be overcome by analyzing variable dependencies such as in the DepQBF solver [23]. A solver could also refute this formula by deciding U out-of-order and soundly conclude that it is false. Hence, in some sense, the hardness of this formula really lies in the propositional formula ϕ_1 rather than the QBF structure. This contrasts with the $\texttt{CR}_n$ formulas where pulling the universal variable to the front makes the formula true. Nevertheless, this notion should be better understood and the recent work of Chen could possibly provide pointers in this direction [9].

8 Summary and Future Work

The paper compares the strength of QBF conflict driven clause learning (CDCL) to Q-resolution. In contrast to its propositional counterparts, CDCL QBF solving appears to be quite weak compared to general Q-resolution. Indeed, even if we impose the limit that the resolution should be tree-like and ordered, CDCL cannot simulate the refutation. Our result strengthens an existing separation between CDCL solving and Q-resolution, which can be overcome by variable dependencies (see Section 7.3). We further conjecture that the presented separation stems from the QBF hardness of the problem rather than propositional hardness.

The crux of our proof is that the investigated formula does not permit propagation across levels, which consequently leads to level-ordered derivations of the learned clauses. This observation suggests a number of interesting questions for future research. Can solution driven cube learning (SDCL) speed-up the proof? The experimental evaluation suggest that it might. However, only if the pure literal technique is turned *off*. This observation also has a practical consequence. Pure literals may lead to fewer learned cubes and consequently a decrease in the quality of the *clausal* proof. Can such adversarial effect be avoided?

References

1. Alekhnovich, M., Johannsen, J., Pitassi, T., Urquhart, A.: An exponential separation between regular and general resolution. Theory of Computing 3(5), 81–102 (2007)
2. Atserias, A., Fichte, J.K., Thurley, M.: Clause-learning algorithms with many restarts and bounded-width resolution. J. Artif. Intell. Res. (JAIR) 40, 353–373 (2011)
3. Balabanov, V., Jiang, J.H.R.: Unified QBF certification and its applications. Formal Methods in System Design 41(1), 45–65 (2012)
4. Beame, P., Kautz, H.A., Sabharwal, A.: Towards understanding and harnessing the potential of clause learning. J. Artif. Intell. Res. (JAIR) 22, 319–351 (2004)

5. Beame, P., Sabharwal, A.: Non-restarting SAT solvers with simple preprocessing can efficiently simulate resolution. In: Brodley, C.E., Stone, P. (eds.) Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence. pp. 2608–2615. AAAI Press (2014)
6. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press (2009)
7. Bonet, M.L., Esteban, J.L., Galesi, N., Johannsen, J.: Exponential separations between restricted resolution and cutting planes proof systems. In: 39th Annual Symposium on Foundations of Computer Science, FOCS. pp. 638–647. IEEE Computer Society (1998)
8. Cadoli, M., Schaerf, M., Giovanardi, A., Giovanardi, M.: An algorithm to evaluate quantified Boolean formulae and its experimental evaluation. J. Autom. Reasoning 28(2), 101–142 (2002)
9. Chen, H.: Proof complexity modulo the polynomial hierarchy: Understanding alternation as a source of hardness. In: 43rd International Colloquium on Automata, Languages, and Programming (ICALP) (2016)
10. Ehrenfeucht, A.: An application of games to the completeness problem for formalized theories. Fund. Math 49(129-141), 13 (1961)
11. Giunchiglia, E., Narizzano, M., Tacchella, A.: Clause/term resolution and learning in the evaluation of quantified Boolean formulas. Journal of Artificial Intelligence Research 26(1), 371–416 (2006)
12. Giunchiglia, E., Marin, P., Narizzano, M.: Reasoning with quantified boolean formulas. In: Biere et al. [6], pp. 761–780
13. Goerdt, A.: Davis-Putnam resolution versus unrestricted resolution. Ann. Math. Artif. Intell. 6(1–3), 169–184 (1992)
14. Goultiaeva, A., Seidl, M., Biere, A.: Bridging the gap between dual propagation and CNF-based QBF solving. In: DATE. pp. 811–814 (2013)
15. Haken, A.: The intractability of resolution. Theoretical Computer Science 39, 297–308 (1985)
16. Janota, M., Marques-Silva, J.: Expansion-based QBF solving versus Q-resolution. Theoretical Computer Science 577, 25–42 (April 2015)
17. Kleine Büning, H., Bubeck, U.: Theory of quantified boolean formulas. In: Biere et al. [6], pp. 735–760
18. Kleine Büning, H., Karpinski, M., Flögel, A.: Resolution for quantified Boolean formulas. Inf. Comput. 117(1), 12–18 (1995)
19. Klieber, W.: Formal Verification Using Quantified Boolean Formulas (QBF). Ph.D. thesis, Carnegie Mellon University (2014), http://www.cs.cmu.edu/~wklieber/thesis.pdf
20. Klieber, W., Sapra, S., Gao, S., Clarke, E.M.: A non-prenex, non-clausal QBF solver with game-state learning. In: Strichman, O., Szeider, S. (eds.) SAT. vol. 6175, pp. 128–142. Springer (2010)
21. Kolaitis, P.G.: The expressive power of stratified programs. Inf. Comput. 90(1), 50–66 (1991)
22. Lonsing, F.: Dependency Schemes and Search-Based QBF Solving: Theory and Practice. Ph.D. thesis, Johannes Kepler Universität (2012), http://www.kr.tuwien.ac.at/staff/lonsing/diss/
23. Lonsing, F., Biere, A.: DepQBF: A dependency-aware QBF solver. JSAT 7(2-3), 71–76 (2010)
24. Lonsing, F., Egly, U., Van Gelder, A.: Efficient clause learning for quantified boolean formulas via QBF pseudo unit propagation. In: Theory and Applications of Satisfiability Testing - SAT. pp. 100–115 (2013)

25. Mahajan, M., Shukla, A.: Level-ordered Q-resolution and tree-like Q-resolution are incomparable. Information Processing Letters 116(3), 256 – 258 (2016)
26. Marques Silva, J.P., Sakallah, K.A.: GRASP: A search algorithm for propositional satisfiability. IEEE Trans. Computers 48(5), 506–521 (1999)
27. Pipatsrisawat, K., Darwiche, A.: On the power of clause-learning SAT solvers as resolution engines. Artificial Intelligence 175(2), 512 – 525 (2011)
28. Van Gelder, A.: Contributions to the theory of practical quantified Boolean formula solving. In: Milano, M. (ed.) CP. vol. 7514, pp. 647–663. Springer (2012)
29. Zhang, L.: Solving QBF by combining conjunctive and disjunctive normal forms. In: AAAI. AAAI Press (2006)
30. Zhang, L., Malik, S.: Conflict driven learning in a quantified Boolean satisfiability solver. In: ICCAD. pp. 442–449 (2002)