Q-Resolution with Generalized Axioms*

Florian Lonsing[1], Uwe Egly[1], and Martina Seidl[2]

[1] Knowledge-Based Systems Group, Vienna University of Technology, Vienna, Austria
[2] Institute for Formal Models and Verification, JKU, Linz, Austria

Abstract. Q-resolution is a proof system for quantified Boolean formulas (QBFs) in prenex conjunctive normal form (PCNF) which underlies search-based QBF solvers with clause and cube learning (QCDCL). With the aim to derive and learn stronger clauses and cubes earlier in the search, we generalize the axioms of the Q-resolution calculus resulting in an exponentially more powerful proof system. The generalized axioms introduce an interface of Q-resolution to any other QBF proof system allowing for the direct combination of orthogonal solving techniques. We implemented a variant of the Q-resolution calculus with generalized axioms in the QBF solver DepQBF. As two case studies, we apply integrated SAT solving and resource-bounded QBF preprocessing during the search to heuristically detect potential axiom applications. Experiments with application benchmarks indicate a substantial performance improvement.

1 Introduction

In the same way as SAT, the decision problem of propositional logic, is the archetypical problem complete for the complexity class NP, QSAT, the decision problem of *quantified Boolean formulas (QBF)*, is the archetypical problem complete for the complexity class PSPACE. The fact that many important practical reasoning, verification, and synthesis problems fall into the latter complexity class (cf. [3] for an overview) strongly motivates the quest for efficient QBF solvers.

As the languages of propositional logic and QBF only marginally differ from a syntactical point of view, namely the quantifiers, it is a natural approach to take inspiration from SAT solving and lift powerful SAT techniques to QSAT. Motivated by the success of *conflict-driven clause learning* (CDCL) in SAT solving [31], a generalized version of CDCL called *conflict/solution-driven clause/cube learning* (often abbreviated by QCDCL) is applied in QSAT solving [11]. Given a propositional formula in conjunctive normal form (CNF), a CDCL-based SAT solver enriches the original CNF with clauses—already found and justified conflicts—which force the solver into a different area of the search space until either a model, i.e., a satisfying variable assignment, is found or until the CNF is proven to be unsatisfiable. If a QBF in prenex conjunctive normal form (PCNF) is unsatisfiable then QCDCL works similar, apart from technical details. In the case of satisfiability, however, it is not sufficient to find one assignment satisfying the

* Supported by the Austrian Science Fund (FWF) under grants S11408-N23 and S11409-N23.

formula. To respect the semantics of universal quantification, QBF models have to be described either by assignment trees or by Skolem functions. Hence, a QBF solver may not abort the search if a satisfying assignment is found. Dual to clause learning, a cube (a conjunction of literals) is learned and the search is resumed. QCDCL is implemented in several state-of-the-art QBF solvers [12, 19, 21, 34].

Apart from QCDCL, orthogonal approaches to QBF solving have been developed. QBF competitions like the QBF Galleries 2013 [25] and 2014 [15] revealed the power of *expansion-based approaches* [1, 6, 18], which are based on a different proof system than search-based solving with QCDCL. We refer to related work [4, 5] for an overview of QBF proof systems. QCDCL relies on Q-resolution [19]. Traditionally, Q-resolution calculi[3] offer two kinds of axioms with limited deductive power: (i) the *clause axiom* stating that any clause in the CNF part of a QBF can be immediately derived and (ii) the *cube axiom* allowing to derive cubes which are propositional implicants of the CNF. In previous work [22], we generalized the cube axiom such that quantified blocked clause elimination (QBCE) [8], a clause elimination procedure for preprocessing, could be tightly integrated in QCDCL for learning smaller cubes earlier in the search.

To overcome the restrictions of the traditional axioms of Q-resolution, we extend previous work [22] on the cube axiom and present more powerful clause axioms. We generalize the traditional clause and cube axioms such that their application relies on checking the satisfiability of the PCNF under the current assignment in QCDCL. This way, the axioms can be applied earlier in the search. Further, they provide a framework to combine Q-resolution with any other (complete or incomplete) QBF proof system. We implemented the generalized axioms in the QCDCL solver DepQBF. As a case study, we integrated bounded expansion and SAT-based abstraction [9] in QCDCL as incomplete QBF solving techniques to detect potential axiom applications. Experimental results indicate a substantial performance increase, particularly on application benchmarks.

This paper is structured as follows. In Sections 2 and 3, we introduce preliminaries and recapitulate search-based QBF solving with QCDCL and traditional Q-resolution. Then we generalize the axioms of Q-resolution in Section 4 allowing for the integration of other proof systems. In Section 5 we integrate SAT-based abstraction into QCDCL. Implementation and evaluation are discussed in Section 6. We conclude with a summary and an outlook to future work in Section 7.

2 Preliminaries

We introduce the concepts and terminology used in the rest of the paper. A *literal* is a variable x or its negation $\bar{x}$. By $\bar{l}$ we denote the negation of literal l and $\mathsf{var}(l) := x$ if $l = x$ or $l = \bar{x}$. A disjunction, resp. conjunction, of literals is called *clause*, resp. *cube*. A propositional formula in *conjunctive normal form* (CNF) is a conjunction of clauses. If convenient, we interpret a CNF as a set of clauses, and clauses and cubes as sets of literals. A QBF in *prenex conjunctive*

[3] Note that there are different variants of Q-resolution, e.g., long-distance resolution [34], QU-resolution [33], etc. [2, 5].

normal form (PCNF) has the form $\Pi.\psi$ with prefix $\Pi := Q_1X_1 \dots Q_nX_n$ and matrix ψ, where ψ is a propositional CNF over the variables defined in Π. The variable sets X_i are pairwise disjoint and for $Q_i \in \{\forall, \exists\}$, $Q_i \neq Q_{i+1}$. We define $\mathsf{var}(\Pi) := X_1 \cup \dots \cup X_n$. The quantifier $\mathsf{Q}(\Pi, l)$ of a literal l is Q_i if $\mathsf{var}(l) \in X_i$. If $\mathsf{Q}(\Pi, l) = Q_i$ and $\mathsf{Q}(\Pi, k) = Q_j$, then $l \leq_\Pi k$ iff $i \leq j$. For a clause or cube C, $\mathsf{var}(C) := \{\mathsf{var}(l) \mid l \in C\}$ and for CNF ψ, $\mathsf{var}(\psi) := \{\mathsf{var}(l) \mid l \in C, C \in \psi\}$.

An *assignment* A is a mapping from the variables $\mathsf{var}(\Pi)$ of a QBF $\Pi.\psi$ to truth values *true* and *false*. We represent A as a set of literals $A = \{l_1, \dots, l_n\}$ with $\{\mathsf{var}(l_i) \mid l_i \in A\} \subseteq \mathsf{var}(\Pi)$ such that if a variable x is assigned *true* then $l_i \in A$ and $l_i = x$, and if x is assigned *false* then $l_i \in A$ and $l_i = \bar{x}$. Further, for any $l_i, l_j \in A$ with $i \neq j$, $\mathsf{var}(l_i) \neq \mathsf{var}(l_j)$. An assignment A is *partial* if it does not map every variable in $\mathsf{var}(\Pi)$ to a truth value, i.e., $\{\mathsf{var}(l_i) \mid l_i \in A\} \subset \mathsf{var}(\Pi)$. A QBF ϕ *under assignment* A, written as $\phi[A]$, is the QBF obtained from ϕ in which for all $l \in A$, all clauses containing l are removed, all occurrences of $\bar{l}$ are deleted, and $\mathsf{var}(l)$ is removed from the prefix. If the matrix of $\phi[A]$ is empty, then the matrix is satisfied by A and A is a *satisfying assignment* (written as $\phi[A] = \mathsf{T}$). If the matrix of $\phi[A]$ contains the empty clause, then the matrix is falsified by A and A is a *falsifying assignment* (written as $\phi[A] = \mathsf{F}$). A QBF $\Pi.\psi$ with $Q_1 = \exists$ (resp. $Q_1 = \forall$) is satisfiable iff $\Pi.\psi[\{x\}]$ or (resp. and) $\Pi.\psi[\{\bar{x}\}]$ is satisfiable where $x \in X_1$. Two QBFs ϕ and ϕ' are *satisfiability-equivalent*, written as $\phi \equiv_{sat} \phi'$, iff ϕ is satisfiable whenever ϕ' is satisfiable. Two propositional CNFs ψ and ψ' are *logically equivalent*, written as $\psi \equiv \psi'$, iff they have the same set of propositional models, i.e., satisfying assignments. Two simplification rules preserving satisfiability equivalence are *unit* and *pure literal detection*. If a QBF ϕ contains a unit clause $C = (l)$, where $\mathsf{Q}(\Pi, l) = \exists$, then $\phi \equiv_{sat} \phi[\{l\}]$. If a literal is pure in QBF ϕ, i.e., ϕ contains l but not $\bar{l}$, then $\phi \equiv_{sat} \phi[\{l\}]$ if $\mathsf{Q}(\Pi, l) = \exists$ and $\phi \equiv_{sat} \phi[\{\bar{l}\}]$ otherwise.

3 QCDCL-Based QBF Solving

Figure 1 shows an abstract workflow of traditional search-based QBF solving with QCDCL [12, 19, 21, 34]. Given a PCNF ϕ, assignments A are successively generated (box in top left corner of Fig. 1). In general, variables must be assigned in the ordering of the quantifier prefix. Variables may either be assigned tentatively as *decisions* or by a QBF-specific variant of *Boolean constraint propagation (QBCP)*. QBCP consists of unit and pure literal detection. Assignments of variables carried out in QBCP do not have to follow the prefix ordering. We formalize the assignments generated during a run of QCDCL as follows.

Definition 1 (QCDCL Assignment). *Given a QBF $\phi = \Pi.\psi$. Let assignment $A = A' \cup A''$ where A' are variables assigned as decisions and A'' are variables assigned by unit/pure literal detection. A is a* QCDCL assignment *if (1) for a maximal $l \in A'$ with $\forall l' \in A' : l' \leq_\Pi l$ it holds that $\forall x \in \mathsf{var}(\Pi) <_\Pi l : x \in \mathsf{var}(A)$ and (2) all $l \in A''$ are unit/pure in $\phi[A']$ after applying QBCP until completion.*

QCDCL generates only QCDCL assignments by Definition 1. Assignment generation by decisions and QBCP continues until the current assignment A is

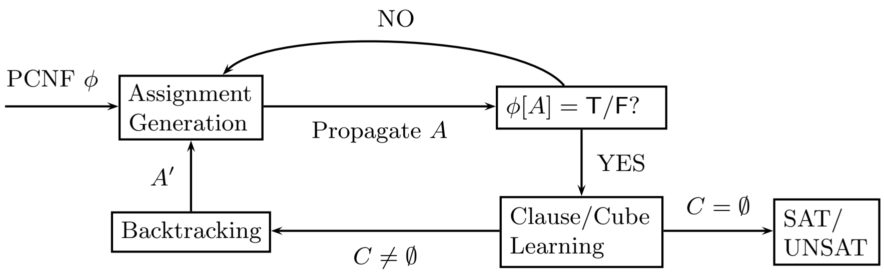

Fig. 1. Abstract workflow of QCDCL with traditional Q-resolution axioms.

either falsifying or satisfying by checking whether $\phi[A] = \mathsf{F}$ or $\phi[A] = \mathsf{T}$ (box in top right corner of Fig. 1). In these cases, a new *learned clause* or *learned cube* is derived in a *learning* phase, which is based on the *Q-resolution calculus*.

Definition 2 (Q-Resolution Calculus). *Let $\phi = \Pi.\psi$ be a PCNF. The rules of the* Q-resolution calculus (QRES) *are as follows.*

$$\frac{C_1 \cup \{p\} \qquad C_2 \cup \{\bar{p}\}}{C_1 \cup C_2} \quad \begin{array}{l} \textit{if for all } x \in \Pi\colon \{x, \bar{x}\} \not\subseteq (C_1 \cup C_2), \\ \bar{p} \notin C_1,\ p \notin C_2, \textit{ and either} \\ \textit{(1) } C_1, C_2 \textit{ are clauses and } \mathsf{Q}(\Pi, p) = \exists \textit{ or} \\ \textit{(2) } C_1, C_2 \textit{ are cubes and } \mathsf{Q}(\Pi, p) = \forall \end{array} \qquad (\textit{res})$$

$$\frac{C \cup \{l\}}{C} \quad \begin{array}{l} \textit{if for all } x \in \Pi\colon \{x, \bar{x}\} \not\subseteq (C \cup \{l\}) \textit{ and either} \\ \textit{(1) } C \textit{ is a clause, } \mathsf{Q}(\Pi, l) = \forall, \\ \qquad l' <_\Pi l \textit{ for all } l' \in C \textit{ with } \mathsf{Q}(\Pi, l') = \exists \textit{ or} \\ \textit{(2) } C \textit{ is a cube, } \mathsf{Q}(\Pi, l) = \exists, \\ \qquad l' <_\Pi l \textit{ for all } l' \in C \textit{ with } \mathsf{Q}(\Pi, l') = \forall \end{array} \qquad (\textit{red})$$

$$\frac{}{C} \quad \textit{if for all } x \in \Pi\colon \{x, \bar{x}\} \not\subseteq C,\ C \textit{ is a clause and } C \in \psi \qquad (\textit{cl-init})$$

$$\frac{}{C} \quad \begin{array}{l} A \textit{ is a QCDCL assignment,} \\ \phi[A] = \mathsf{T}, \\ \textit{and } C = (\bigwedge_{l \in A} l) \textit{ is a cube} \end{array} \qquad (\textit{cu-init})$$

QRES is a proof system which underlies QCDCL. Rule *cl-init* is an axiom to derive clauses which are already part of the given PCNF ϕ. In practice, the clause C selected by axiom *cl-init* is falsified under the current QCDCL assignment. Axiom *cu-init* allows to derive cubes based on a QCDCL assignment A which satisfies all the clauses of the matrix ψ of $\phi = \Pi.\psi$ (i.e., $\phi[A] = \mathsf{T}$). A cube C derived by axiom *cu-init* is an *implicant* of ψ, i.e., the implication $C \Rightarrow \psi$ is valid.

The *resolution* and *reduction* rules *res* and *red*, respectively, are applied either to clauses or cubes. Rule *red* is called *universal (existential) reduction* when applied to clauses (cubes). We write $UR(C)$ ($ER(C)$) to denote the clause (cube) resulting from universal (existential) reduction of clause (cube) C. The PCNF $UR(\phi)$ is obtained by universal reduction of all clauses in the PCNF ϕ.

Q-resolution of clauses [19] generalizes propositional resolution, which consists of rules *cl-init* and *res*, by the reduction rule *red*. Q-resolution of cubes was introduced for *cube learning* [12, 21, 34], the dual variant of clause learning.

QRES is sound and refutationally complete for PCNFs [12, 19, 21, 34]. The empty clause (cube) is derivable from a PCNF ϕ in QRES if and only if ϕ is unsatisfiable (satisfiable). A derivation of the empty clause (cube) from ϕ is a *clause (cube) resolution proof* of ϕ.

In QCDCL, the rules of QRES are applied to derive new learned clauses or cubes. A learned clause (cube) C is added conjunctively (disjunctively) to the PCNF $\phi = \Pi.\psi$ to obtain $\Pi.(\psi \wedge C)$ $(\Pi.(\psi \vee C))$. After C has been added, certain assignments in the current assignment A are retracted during *backtracking*, resulting in assignment A' ($C \neq \emptyset$ in Fig. 1). Assignment generation based on A' continues, where learned clauses and cubes participate in QBCP. Typically, only *asserting* learned clauses and cubes are generated in QCDCL. A clause (cube) C is asserting if $UR(C)$ $(ER(C))$ is unit under A' after backtracking. QCDCL terminates if and only if the empty clause or cube is learned ($C = \emptyset$ in Fig. 1).

Example 1 ([22]). Given a PCNF ϕ with prefix $\exists z, z' \forall u \exists y$ and matrix ψ:

$$\begin{aligned} \psi := & (u \vee \bar{y}) \wedge (\bar{u} \vee y) \wedge \\ & (z \vee u \vee \bar{y}) \wedge (z' \vee \bar{u} \vee y) \wedge \\ & (\bar{z} \vee \bar{u} \vee \bar{y}) \wedge (\bar{z}' \vee u \vee y) \end{aligned}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(\bar{z} \wedge \bar{z}' \wedge \bar{u} \wedge \bar{y})}}{(\bar{z} \wedge \bar{z}' \wedge \bar{u})} \qquad \dfrac{\dfrac{}{(\bar{z} \wedge \bar{z}' \wedge u \wedge y)}}{(\bar{z} \wedge \bar{z}' \wedge u)}}{\bar{z} \wedge \bar{z}'}}{\emptyset}$$

Let $A_1 := \{\bar{z}, \bar{z}', \bar{u}, \bar{y}\}$ and $A_2 := \{\bar{z}, \bar{z}', u, y\}$ be satisfying QCDCL assignments to be used for applications of axiom *cu-init*. A derivation of the empty cube by rules *cu-init*, *red*, *res*, and *red* (from top to bottom) is shown on the right. $\diamond$

4 Generalizing the Axioms of QRES

The axioms *cl-init* and *cu-init* of QRES have limited deductive power. Any clause derived by *cl-init* already appears in the matrix ψ of the PCNF $\phi = \Pi.\psi$. Any cube derived by *cu-init* is an implicant of ψ.

To overcome these limitations, we equip QRES with two additional axioms—one to derive clauses and one to derive cubes—which generalize *cl-init* and *cu-init*. *Generalized model generation (GMG)* [22] was presented as a new axiom to derive learned cubes. The combination of QRES with GMG is stronger than QRES with *cu-init* in terms of the sizes of cube resolution proofs it is able to produce. In the following, we formulate a generalized clause axiom which we combine with QRES in addition to GMG. Thereby, we obtain a variant of QRES which is stronger than traditional QRES also in terms of sizes of clause resolution proofs.

Figure 2 shows an abstract workflow of search-based QBF solving with QCDCL relying on QRES *with* generalized axioms. This workflow is the same as in Fig. 1 except for applications of axioms (box in top right corner). The generalized axioms are applied if the PCNF $\phi[A]$ under a QCDCL assignment A is *(un)satisfiable*. This is in contrast to the more restricted conditions $\phi[A] = \mathsf{T}$

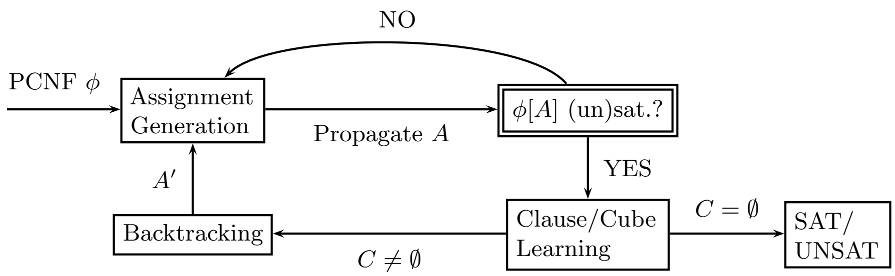

Fig. 2. Abstract workflow of QCDCL with generalized Q-resolution axioms.

or $\phi[A] = \mathsf{F}$ in Fig. 1. We show that the generalized axioms allow to combine *any* sound (but maybe incomplete) QBF solving technique with QCDCL based on QRES. First, we define *QCDCL clauses* and recapitulate *QCDCL cubes* [22].

Definition 3 (QCDCL Clause/Cube). *Given a QBF $\phi = \Pi.\psi$. The* QCDCL clause *C of QCDCL assignment A is defined by $C = (\bigvee_{l \in A} \bar{l})$. The* QCDCL cube *$C$ of QCDCL assignment A is defined by $C = (\bigwedge_{l \in A} l)$.*

By Definition 1, a QCDCL clause or cube cannot contain complementary literals x and $\bar{x}$ of some variable x. According to QCDCL assignments, we split QCDCL clauses and cubes into decision literals and literals assigned by unit and pure literal detection. Let C be a QCDCL clause or QCDCL cube. Then $C = C' \cup C''$ where C' is the maximal subset of C such that $X_1 \cup \ldots \cup X_{i-1} \subset \mathsf{var}(C')$ and $C' \cap X_i \neq \emptyset$. The literals in C' are the first $|C'|$ consecutive variables of Π which are assigned, i.e., C' contains all the variables in C assigned as decisions.[4] The literals in C'' are assigned due to pure and unit literal detection and may occur anywhere in Π starting from X_{i+1}. Further we define $\mathsf{dec}(C) = C'$ and $\mathsf{der}(C) = C''$. We review *generalized model generation* [22] as an axiom to derive cubes.

Definition 4 (Generalized Model Generation [22]). *Given a PCNF ϕ and a QCDCL assignment A according to Definition 1. If $\phi[A]$ is satisfiable, then the QCDCL cube $C = (\bigwedge_{l \in A} l)$ is obtained by* generalized model generation.

Theorem 1 ([22]). *Given PCNF $\phi = \Pi.\psi$ and a QCDCL cube C obtained from ϕ by generalized model generation. Then it holds that $\Pi.\psi \equiv_{sat} \Pi.(\psi \vee C)$.*

Corollary 1 ([22]). *By Theorem 1, a cube C obtained from PCNF $\Pi.\psi$ by generalized model generation can be used as a learned cube in QCDCL.*

Dual to generalized model generation, we define *generalized conflict generation* to derive clauses which can be added to a PCNF in a satisfiability-preserving way.

Definition 5 (Generalized Conflict Generation). *Given a PCNF ϕ and a QCDCL assignment A according to Definition 1. If $\phi[A]$ is unsatisfiable, then the QCDCL clause $C = (\bigvee_{l \in A} \bar{l})$ is obtained by* generalized conflict generation.

[4] C' can also contain literals assigned by pure/unit literal detection, but as they are left to the maximal decision variable in the prefix, we treat them like decision variables.

Theorem 2. *Given PCNF $\phi = \Pi.\psi$ and a QCDCL clause C obtained from ϕ by generalized conflict generation using QCDCL assignment A. Then it holds that $\Pi.\psi \equiv_{sat} \Pi.(\psi \wedge C)$.*

Proof (Sketch). We argue that if $\Pi.\psi$ is satisfiable, so is $\Pi.(\psi \wedge C)$. The case for unsatisfiability is trivial. Let $C = C' \cup C''$ with $C' = \mathsf{dec}(C)$ and $C'' = \mathsf{der}(C)$. Further, let $A = A' \cup A''$ such that $\mathsf{var}(A') = \mathsf{var}(C')$ and $\mathsf{var}(A'') = \mathsf{var}(C'')$. Now assume that $\Pi.\psi$ is satisfiable, but $\Pi.(\psi \wedge C)$ is not. In order to falsify C, its subclause C' has to be falsified, i.e., the first $|C'|$ variables of Π have to be set according to A'. Then, due to pure and unit, also C'' is falsified, and therefore, each assignment falsifying C has to contain A. But $\Pi.\psi[A]$ is unsatisfiable. Since $\Pi.\psi$ is satisfiable, there have to be other decisions than the decisions of A to show its satisfiability, but these also satisfy $\Pi.(\psi \wedge C)$. $\square$

Corollary 2. *By Theorem 2, a clause C obtained from PCNF $\Pi.\psi$ by generalized conflict generation can be used as a learned clause in QCDCL.*

Based on Corollaries 1 and 2, we formulate axioms to derive learned clauses (cubes) from QCDCL assignments A under which the PCNF ϕ is (un)satisfiable.

Definition 6 (Generalized Axioms). *Let $\phi = \Pi.\psi$ be a PCNF. The* generalized clause *and* cube axioms *are as follows.*

$$\frac{}{C} \quad \begin{array}{l} A \textit{ is a QCDCL assignment,} \\ \phi[A] \textit{ is unsatisfiable,} \\ \textit{and } C = (\bigvee_{l \in A} \bar{l}) \textit{ is a QCDCL clause} \end{array} \qquad (\textit{gen-cl-init})$$

$$\frac{}{C} \quad \begin{array}{l} A \textit{ is a QCDCL assignment,} \\ \phi[A] \textit{ is satisfiable,} \\ \textit{and } C = (\bigwedge_{l \in A} l) \textit{ is a QCDCL cube} \end{array} \qquad (\textit{gen-cu-init})$$

The generalized axioms *gen-cl-init* and *gen-cu-init* are added to QRES in addition to the traditional axioms *cl-init* and *cu-init* from Definition 2.

Example 2. Consider the PCNF from Example 1. Let $A := \{\bar{z}, \bar{z}'\}$ be a QCDCL assignment where z and z' are assigned as decisions. The PCNF $\phi[A] = \forall u \exists y.(u \vee \bar{y}) \wedge (\bar{u} \vee y)$ is satisfiable. We apply axiom *gen-cu-init* to derive the cube $C := (\bar{z} \wedge \bar{z}')$ and finally the empty cube $ER(C) = \emptyset$ (proof shown on the right).

$$\frac{\overline{\bar{z} \wedge \bar{z}'}}{\emptyset}$$

$\diamond$

In contrast to axioms *cl-init* and *cu-init* (the latter corresponds to *model generation* [12]), the generalized axioms allow to derive clauses that are not part of the given PCNF ϕ and cubes that are not implicants of the matrix of ϕ.

Given the empty assignment $A = \{\}$ and a PCNF ϕ, the empty clause or cube can be derived using A by axioms *gen-cl-init* or *gen-cu-init* right away if $\phi[A]$ is unsatisfiable or satisfiable, respectively. However, checking the satisfiability of the PCNF $\phi[A]$ as required in the side conditions of the generalized axioms is PSPACE-complete. Therefore, *in practice* it is necessary to consider non-empty QCDCL assignments A and apply either complete approaches in a bounded way, like the successful expansion-based approaches [1, 6, 13, 18], or incomplete polynomial-time procedures, e.g., as used in preprocessing [13], to check the satisfiability of $\phi[A]$. *Sign abstraction* [21] can be regarded as a first approach towards more powerful cube learning as formalized by axiom *gen-cu-init*.

Axioms *gen-cl-init* and *gen-cu-init* provide a formal framework for combining Q-resolution in QRES with *any* QBF decision procedure $\mathcal{D}$ by using $\mathcal{D}$ to check $\phi[A]$. This framework also applies to related combinations of search-based QBF solving with variable elimination [27]. Regarding proof complexity, decision procedures like expansion and Q-resolution are incomparable as the lengths of proofs they are able to produce for certain PCNFs differ by an exponential factor [2, 5, 16]. Due to this property, the combination of incomparable procedures in QRES via the generalized axioms allows to benefit from their individual strengths. For example, the use of expansion to check the satisfiability of $\phi[A]$ in axioms *gen-cl-init* and *gen-cu-init* results in a variant of QRES which is exponentially stronger than traditional QRES. For satisfiable PCNFs, QBCE, originally a preprocessing technique to eliminate redundant clauses in a PCNF, was shown to be effective to solve $\phi[A]$ for applications of axiom *gen-cu-init* [22], resulting in an exponentially stronger cube proof system.

If a decision procedure $\mathcal{D}$ is applied as a black box to check $\phi[A]$, then QRES extended by *gen-cl-init* and *gen-cu-init* is not a proof system as defined by Cook and Reckhow [10] because the final proof P of ϕ cannot be checked in polynomial time. However, $\mathcal{D}$ can be augmented to return a proof P' of $\phi[A]$ for every application of *gen-cl-init* and *gen-cu-init*. Such proof P' may be formulated, e.g., in the QRAT proof system [14]. Finally, the proof P of ϕ contains subproofs P', all of which can be checked in polynomial time, like P itself (the size of P may blow up exponentially in the worst case depending on the decision procedures that are used to produce the subproofs P').

The QCDCL framework (Fig. 2) readily supports applications of the generalized axioms *gen-cl-init* and *gen-cu-init*. A clause (resp. cube) C derived by these axioms is first reduced by universal (resp. existential) reduction to obtain a reduced clause (cube) $C' \subseteq C$. Then C' is used to derive an asserting learned clause (cube) *in the same way* as in clause learning by traditional QRES (Definition 2).

5 An Abstraction-Based Clause Axiom

Axioms *gen-cl-init* and *gen-cu-init* by Definition 6 are based on QCDCL assignments, where decision variables have to be assigned in prefix ordering. To overcome the order restriction, we introduce a clause axiom which allows to derive clauses based on an abstraction of a PCNF and *arbitrary* assignments.

Definition 7 (Existential Abstraction). *Let $\phi = \Pi.\psi$ be a PCNF with prefix $\Pi := Q_1X_1Q_2X_2 \ldots Q_nX_n$ and matrix ψ. The* existential abstraction $Abs_\exists(\phi) := \Pi'.\psi$ *of ϕ has prefix $\Pi' := \exists(X_1 \cup X_2 \cup \ldots \cup X_n)$.*

Lemma 1. *Let $\phi = \Pi.\psi$ be a PCNF, $Abs_\exists(\phi)$ its existential abstraction, and A a partial assignment of the variables in $Abs_\exists(\phi)$. If $Abs_\exists(\phi)[A]$ is unsatisfiable then $\psi \equiv \psi \wedge (\bigvee_{l\in A} \bar{l})$.*

Proof. Obviously, every model M of $\psi \wedge (\bigvee_{l \in A} \overline{l})$ is also a model of ψ. To show the other direction, let M be a model of ψ, but $(\psi \wedge (\bigvee_{l \in A} \overline{l}))[M] = \mathsf{F}$. Then $A \subseteq M$. Since $Abs_{\exists}(\phi)[A]$ is unsatisfiable, also $\psi[A]$ is unsatisfiable. Then M cannot be a model of ψ. □

Theorem 3 (cf. [29, 30]). *For a PCNF $\phi = \Pi.\psi$, $Abs_{\exists}(\phi)$ its existential abstraction, and a partial assignment A of the variables in $Abs_{\exists}(\phi)$ such that $Abs_{\exists}(\phi)[A]$ is unsatisfiable, it holds that $\Pi.\psi \equiv_{sat} \Pi.(\psi \wedge (\bigvee_{l \in A} \overline{l}))$.*

Proof. By Lemma 1, ψ and $\psi \wedge (\bigvee_{l \in A} \overline{l})$ have the same sets of propositional models. As argued in the context of SAT-based QBF solving [29] and QBF preprocessing [30], model-preserving manipulations of the matrix of a PCNF result in a satisfiability-equivalent PCNF.[5] □

Definition 8 (Abstraction-Based Conflict Generation). *Given a PCNF ϕ, its existential abstraction $Abs_{\exists}(\phi)$ and an assignment A (not necessarily being a QCDCL assignment). If $Abs_{\exists}(\phi)[A]$ is unsatisfiable, then the clause $C = (\bigvee_{l \in A} \overline{l})$ is obtained by* abstraction-based conflict generation.

We formulate a new axiom to derive clauses by abstraction-based conflict generation, which can be used as ordinary learned clauses in QCDCL (Theorem 3).

Definition 9 (Abstraction-Based Clause Axiom). *For a PCNF $\phi = \Pi.\psi$ and $Abs_{\exists}(\phi)$ by Definition 7, the* abstraction-based clause axiom *is as follows:*

$$\frac{}{C} \quad \begin{array}{l} A \textit{ is an assignment,} \\ Abs_{\exists}(\phi)[A] \textit{ is unsatisfiable,} \\ \textit{and } C = (\bigvee_{l \in A} \overline{l}) \textit{ is a clause} \end{array} \qquad (\textit{abs-cl-init})$$

Axiom *abs-cl-init* can be added to QRES in addition to all the other axioms. In the side condition of axiom *abs-cl-init*, the propositional CNF $Abs_{\exists}(\phi)[A]$ has to be solved, which naturally can be carried out by integrating a SAT solver in QCDCL. SAT solving has been applied in the context of QCDCL to derive learned clauses [29] and to overcome the ordering of the prefix of a PCNF. Further, many QBF solvers rely on SAT solving [17, 18, 26, 32]. Integrating axiom *abs-cl-init* in QRES by Definition 2 results in a variant of QRES which is exponentially stronger than traditional QRES, as illustrated by the following example.

Example 3. Consider the following family $(\phi_t)_{t \geq 1}$ of PCNFs defined by Kleine Büning et al. [19]. A formula ϕ_t in $(\phi_t)_{t \geq 1}$ has the quantifier prefix

$$\exists d_0 d_1 e_1 \forall x_1 \exists d_2 e_2 \forall x_2 \exists d_3 e_3 \ldots \forall x_{t-1} \exists d_t e_t \forall x_t \exists f_1 \ldots f_t$$

and a matrix consisting of the following clauses:

$$\begin{array}{llll} C_0 := \overline{d}_0 & C_1 := d_0 \vee \overline{d}_1 \vee \overline{e}_1 & \\ C_{2j} := d_j \vee \overline{x}_j \vee \overline{d}_{j+1} \vee \overline{e}_{j+1} & C_{2j+1} := e_j \vee x_j \vee \overline{d}_{j+1} \vee \overline{e}_{j+1} & \text{for } 1 \leq j < t \\ C_{2t} := d_t \vee \overline{x}_t \vee \overline{f}_1 \vee \ldots \vee \overline{f}_t & C_{2t+1} := e_t \vee x_t \vee \overline{f}_1 \vee \ldots \vee \overline{f}_t & \\ B_{2j-1} := x_j \vee f_j & B_{2j} := \overline{x}_j \vee f_j & \text{for } 1 \leq j \leq t \end{array}$$

[5] In fact, a stronger result is proved in [30]: model-preserving manipulations of the matrix of a PCNF result in a PCNF having the same set of *tree-like QBF models*.

The size of every clause resolution proof of ϕ_t in traditional QRES (Definition 2) is exponential in t [5, 19]. We show that QRES with axiom *abs-cl-init* allows to generate proofs of ϕ_t which are polynomial in t. To this end, we apply *abs-cl-init* to derive unit clauses (f_j) for all existential variables f_j in ϕ_t using assignments $A := \{\bar{f}_j\}$, respectively. Since $Abs_\exists(\phi_t)[A]$ contains complementary unit clauses (x_j) and $(\overline{x}_j)$ resulting from the clauses B_{2j-1} and B_{2j} in ϕ_t, the unsatisfiability of $Abs_\exists(\phi_t)[A]$ can be determined in polynomial time without invoking a SAT solver. The derived unit clauses (f_j) are resolved with clauses C_{2t} and C_{2t+1} to produce further unit clauses (d_t) and (e_t) after universal reduction. This process continues with C_{2j} and C_{2j+1} until the empty clause is derived using C_0 and C_1. $\diamond$

Abstraction-based failed literal detection [23], where certain universal quantifiers of a PCNF are treated as existential ones, implicitly relies on *QU-resolution*. QU-resolution allows universal variables as pivots in rule *res* and can generate the same proofs of $(\phi_t)_{t\geq 1}$ as in Example 3 [33]. Applying axiom *abs-cl-init* for clause learning in QCDCL harnesses the power of SAT solving. Furthermore, the combination of QRES (Definition 2) and *abs-cl-init* polynomially simulates[6] QU-resolution, which has not been applied systematically to learn clauses in QCDCL. Like with the axioms *gen-cl-init* and *gen-cu-init*, clauses derived by axiom *abs-cl-init* can readily be used to derive asserting learned clauses in QCDCL.

6 Case Study and Experiments

DepQBF[7] is a QCDCL-based QBF solver implementing the Q-resolution calculus as in Definition 2. Since version 5.0, DepQBF additionally applies the generalized cube axiom *gen-cu-init* based on dynamic blocked clause elimination (QBCE) [22]. The case where QBCE reduces the PCNF $\phi[A]$ under the current assignment A to the empty formula constitutes a successful application of axiom *gen-cu-init*. DepQBF comes with a sophisticated analysis of variable dependencies in a PCNF [28] to relax their linear prefix ordering. However, we disabled dependency analysis to focus the evaluation on axiom applications. In the following, we evaluate the impact of (combinations of) the generalized axioms *gen-cl-init* and *gen-cu-init* and the abstraction-based clause axiom *abs-cl-init* in practice.

6.1 Axiom Applications in Practice

In DepQBF, we attempt to apply the generalized axioms after QBCP has saturated in QCDCL, i.e., before assigning a variable as decision. We integrated the preprocessor Bloqqer [8] to detect applications of *gen-cl-init* and *gen-cu-init*. Bloqqer implements techniques such as equivalence reasoning, variable elimination, (variants of) QBCE, and expansion of universal variables. Since these techniques are applied in bounded fashion, Bloqqer can be regarded as an incomplete QBF solver. If the PCNF $\phi[A]$ is satisfiable (unsatisfiable) and Bloqqer solves it, then

[6] We refer to an appendix of this paper with additional results [24].
[7] DepQBF is free software: `http://lonsing.github.io/depqbf/`

a QCDCL cube (clause) is generated by axiom *gen-cu-init* (*gen-cl-init*), which is used to derive a learned cube (clause). Otherwise, QCDCL proceeds as usual with assigning a decision variable. Bloqqer is explicitly provided with the entire PCNF $\phi[A]$ before each call. To limit the resulting run time overhead in practice, Bloqqer is called in intervals of 2^n decisions, where $n := 11$ in our experiments. Further, Bloqqer is never called on PCNFs with more than 500,000 original clauses, and it is disabled at run time if the average time spent to complete a call exceeds 0.125 seconds.

To detect applications of the abstraction-based clause axiom *abs-cl-init*, we use the SAT solver PicoSAT [7] to check the satisfiability of the existential abstraction $Abs_\exists(\phi)[A]$ of the PCNF $\phi = \Pi.\psi$ under the current QCDCL assignment A. The matrix ψ is imported to PicoSAT once before the entire solving process starts. For each check of $Abs_\exists(\phi)[A]$, the QCDCL assignment A is passed to PicoSAT via assumptions, and PicoSAT is called incrementally. If $Abs_\exists(\phi)[A]$ is unsatisfiable, then we try to minimize the size of A by extracting the set $A' \subseteq A$ of *failed assumptions*. Failed assumptions are those assumptions that were relevant for the SAT solver to determine the unsatisfiability of $Abs_\exists(\phi)[A]$. Note that in general A' is not a QCDCL assignment. It holds that $Abs_\exists(\phi)[A']$ is unsatisfiable and hence we derive the clause $C = (\bigvee_{l \in A'} \bar{l})$ by axiom *abs-cl-init*.

In addition to Bloqqer and dynamic QBCE (which is part of DepQBF 5.0 [22]) used to detect applications of the generalized cube axiom *gen-cu-init*, we implemented a *trivial truth* [9] test based on the following abstraction.

Definition 10 (Universal Literal Abstraction, cf. *Trivial Truth* [9]). *Let $\phi = \Pi.\psi$ be a PCNF. The* universal literal abstraction *$Abs_\forall(\phi) := \Pi'.\psi'$ of ϕ is obtained by removing all universal literals from all the clauses in ψ and by removing all universal variables and universal quantifiers from Π.*

Lemma 2 ([9]). *For a PCNF $\phi = \Pi.\psi$, $Abs_\forall(\phi)$, and a QCDCL assignment A of variables in $Abs_\forall(\phi)$: if $Abs_\forall(\phi)[A]$ is satisfiable, then $\phi[A]$ is satisfiable.*

By Lemma 2, we can check the side condition of axiom *gen-cu-init* whether $\phi[A]$ is satisfiable under a QCDCL assignment A by checking whether $Abs_\forall(\phi)[A]$ is satisfiable. To this end, we use a second instance of PicoSAT. Note that while Definition 10 corresponds to trivial truth, the existential abstraction (Definition 7) corresponds to *trivial falsity* [9]. Hence by axiom applications, we apply trivial truth and falsity, which originate from purely search-based QBF solving without learning, to derive clauses and cubes in QCDCL.

Like Bloqqer, we call the two instances of PicoSAT to detect applications of *abs-cl-init* and *gen-cu-init* in QCDCL before assigning a decision variable. PicoSAT is called in intervals of 2^m decisions, where $m := 10$. PicoSAT is never called on PCNFs with more than 500,000 original clauses, and it is disabled at run time if the average time spent to complete a call exceeds five seconds.

6.2 Experimental Results

The integration of Bloqqer and SAT solving to detect axiom applications results in several variants of DepQBF. We use the letter code "DQ-{nQ|B|A|T}" to

Table 1. Preprocessing track. Solved instances ($\#T$), solved unsatisfiable ($\#U$) and satisfiable ones ($\#S$), and total wall clock time in seconds including time outs.

Solver	$\#T$	$\#U$	$\#S$	*Time*
RAReQS	107	44	63	255K
DQ-nQAT	105	46	59	266K
QESTO	104	46	58	267K
DQ-nQ	101	44	57	271K
DQ-AT	99	45	54	273K
DQ-BAT	98	43	55	276K
DQ	95	43	52	278K
DQ-A	95	44	51	280K
DQ-T	94	41	53	278K
DQ-B	94	42	52	284K
QELL-c	87	34	53	290K
CAQE	74	24	50	319K
GhostQ	61	18	43	338K

Table 2. QBFLIB track. Same column headers as Table 1.

Solver	$\#T$	$\#U$	$\#S$	*Time*
GhostQ	139	62	77	265K
DQ-AT	110	58	52	314K
DQ-BAT	109	56	53	314K
DQ-T	108	56	52	318K
QELL-c	106	48	58	320K
DQ-A	106	58	48	321K
DQ	105	57	48	326K
DQ-B	104	56	48	326K
DQ-nQAT	88	49	39	352K
DQ-nQ	82	44	38	362K
RAReQS	80	47	33	361K
QESTO	73	46	27	378K
CAQE	53	32	21	406K

label the variants, where "DQ" represents DepQBF 5.0 with dynamic QBCE used for axiom *gen-cu-init* [22]. Variant "nQ" indicates that dynamic QBCE is disabled. Letters, "B", "A", and "T" represent the additional application of Bloqqer for axioms *gen-cl-init* and *gen-cu-init*, SAT solving to check the existential abstraction for axiom *abs-cl-init*, and SAT solving to carry out the trivial truth test for *gen-cu-init*, respectively.

For the empirical evaluation, we used the original benchmark sets from the QBF Gallery 2014 [15][8] preprocessing track (243 instances), QBFLIB track (276 instances), and applications track (735 instances). We compare the variants of DepQBF to RAReQS [18] and GhostQ [20], which showed top performance in the QBF Gallery 2014, and to the recent solvers CAQE [26][9], QESTO [17], and QELL [32]. We tested QELL with (QELL-c) and without (QELL-nc) exploiting circuit information and show only the results of the better variant of the two in terms of solved instances. All experiments reported in the following were run on an AMD Opteron 6238 at 2.6 GHz under 64-bit Ubuntu Linux 12.04 with time and memory limits of 1800 seconds and 7 GB, respectively.

Tables 1 to 3 illustrate solver performance by solved instances and total wall clock time. For DepQBF, the variant where only dynamic QBCE is applied (DQ) is the baseline of the comparison. In the QBFLIB (Table 2) and applications track (Table 3), DepQBF with Bloqqer and SAT solving for axioms *gen-cl-init*, *gen-cu-init*, and *abs-cl-init* solves substantially more instances than DQ.

Disabling dynamic QBCE used for axiom *gen-cu-init* (variants with "nQ" in the tables) results in a considerable performance decrease, except in the preprocessing track (Table 1). There, dynamic QBCE is harmful to the performance.

[8] `http://qbf.satisfiability.org/gallery/`

[9] The authors [26] provided us with an updated version which we used in our tests.

Table 3. Applications track. Same column headers as Table 1.

Solver	*#T*	*#U*	*#S*	*Time*
DQ-BAT	466	236	230	553K
DQ-AT	461	234	227	555K
DQ-A	459	237	222	561K
DQ-B	449	222	227	563K
DQ-T	441	220	221	571K
DQ	441	224	217	575K
QELL-nc	434	302	132	563K
RAReQS	414	272	142	611K
CAQE	370	192	178	708K
GhostQ	347	166	181	752K
QESTO	331	188	143	767K
DQ-nQBAT	293	140	153	848K
DQ-nQ	279	127	152	880K

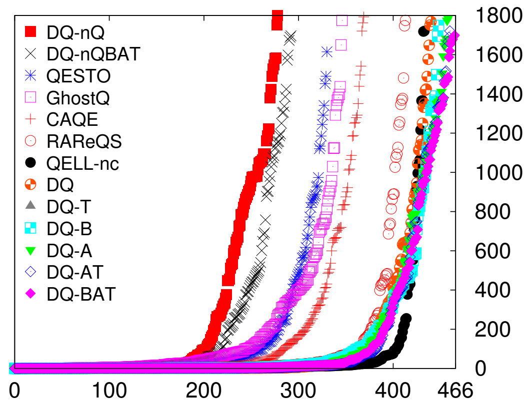

Fig. 3. Sorted run times (y-axis) of instances (x-axis) related to Table 3.

We attribute this phenomenon to massive preprocessing, after which QBCE does not pay off. However, SAT solving for axioms *gen-cl-init* and *gen-cu-init* is crucial as the variant DQ-nQAT outperforms DQ-nQ without SAT solving.

In general, combinations of dynamic QBCE, Bloqqer, and SAT solving (for solving the existential abstraction and for testing trivial truth) outperform variants where only one of these techniques is applied. Examples are DQ-AT, DQ-A and DQ-T in Table 2 and DQ-BAT, DQ-B, DQ-A, and DQ-T in Table 3. The results in the applications track are most pronounced, where six out of eight variants of DepQBF outperform the other solvers (Fig. 3 shows a related cactus plot of the run times). In the following we focus on the applications track.

Consider the best performing variant DQ-BAT in Table 3. Table 4 shows statistics on the number of attempted and successful applications of axioms *gen-cl-init*, *gen-cu-init* and *abs-cl-init* by Bloqqer and SAT solving. On the 466 instances solved by DQ-BAT, Bloqqer was called on $\phi[A]$ at least once on 185 instances and successfully solved $\phi[A]$ at least once on 184 instances, thus allowing applications of axiom *gen-cl-init* or *gen-cu-init*. Bloqqer was disabled at run time on 143 instances due to the predefined limits. SAT solving for the trivial truth test for *gen-cu-init* (respectively, to solve the existential abstraction for *abs-cl-init*) was applied at least once on 364 (445) instances, was successful at least once on 177 (226) instances, and was disabled at run time on 21 (70) instances. While Bloqqer is applied less frequently than SAT solving by a factor of two, applications of Bloqqer have much higher success rates (97%) than SAT solving (8% and 22%).

In the following, we analyze applications of the abstraction-based clause axiom in more detail. The extraction of failed assumptions in SAT solving for *abs-cl-init* allows to reduce the size of the clauses learned by abstraction-based conflict generation. On 145 instances solved by DQ-BAT (Table 3), axiom *abs-cl-init* was applied more than once. Per instance, on average (median) 3,336K (70.7K) assumptions were passed to the SAT solver when solving $Abs_{\exists}(\phi)[A]$, 28.8K (2.3K)

Table 4. Related to variant DQ-BAT in Table 3: statistics on applications of Bloqqer (B), SAT solving for *abs-cl-init* (A), and SAT solving to test trivial truth for *gen-cu-init* (T) with respect to total solved instances ($\#T$) and solved satisfiable ($\#S$) and unsatisfiable ones ($\#U$).

	$\#T$	$\#S$	$\#U$
B tried:	18559	12052	6507
B success:	18150	11946	6204
B sat:	10917	10405	512
B unsat:	7233	1541	5692
T tried:	241,180	88,623	152,557
T success:	20,494	19,276	1,218
A tried:	301,652	122,929	178,723
A success:	67,129	34,306	32,823

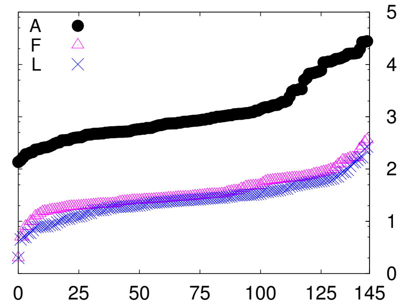

Fig. 4. Average SAT solver assumptions per successful application of *abs-cl-init* ("A") on 145 selected instances solved by DQ-BAT (Table 3), failed assumptions ("F"), and literals in the clauses learned by *abs-cl-init* ("L"), log_{10} scale on y-axis.

failed assumptions were extracted, and the clauses finally learned had 20.7K (1.5K) literals. The difference in the number of failed assumptions and the size of learned clauses is due to additional, heuristic minimization of the set of failed assumptions which we apply. Given that $Abs_{\exists}(\phi)[A]$ is unsatisfiable, it may be possible to remove assignments from A, thus resulting in a smaller assignment A', while preserving unsatisfiability of $Abs_{\exists}(\phi)[A']$. Additionally, universal reduction by rule *red* may remove literals from the clause learned by generalized conflict generation. Figure 4 shows related average statistics.

The abstraction-based clause axiom *abs-cl-init* is particularly effective on instances from the domain of conformant planning. With variant DQ-BAT (Table 3), 81 unsatisfiable instances from conformant planning were solved by a *single* application of axiom *abs-cl-init* where the empty clause was derived immediately. On 13 of these 81 instances, solving $Abs_{\exists}(\phi)$ was hard for the SAT solver, which took more than 900 seconds. In contrast to DQ-BAT, DQ does not use axiom *abs-cl-init* and failed to solve 15 of the 81 instances.

Additionally, we evaluated the variants of DepQBF and the other solvers on the benchmarks of the applications and QBFLIB tracks *with* preprocessing by Bloqqer before solving.[10] In the QBFLIB track, RAReQS and DQ-T solved the largest number of instances (134 in total each instead of 80 and 108 in Table 2). However, here it is important to remark that already the plain variant DepQBF solved 132 instances if Bloqqer is applied before solving. With partial preprocessing by Bloqqer (using only QBCE and universal expansion), on the applications track QELL-nc and DQ-AT each solved 483 instances, i.e., 49 and 22 more instances than without preprocessing (Table 3). Note that the best variant DQ-BAT of DepQBF in Table 3 solved 480 instances. Partial preprocessing increases the number of instances solved by the variants of DepQBF. In contrast

[10] We refer to an appendix of this paper with additional tables [24].

to that, with full preprocessing the performance of the variants of DepQBF on the applications track considerably decreases. If Bloqqer is applied to the full extent (enabling all techniques), then RAReQS, QELL-nc, and QESTO solve 547, 501, and 463 instances, respectively. The variant DQ-AT of DepQBF, however, which solved 483 instances with partial preprocessing, solves only 434 instances. The phenomenon that preprocessing is not always beneficial was also observed in the QBF Galleries [15, 25]. When applied without restrictions, Bloqqer rewrites a formula and thus destroys or blurs structural information. For some approaches structural information is essential to fully exploit their individual strengths.

7 Conclusion

The Q-resolution calculus QRES is a proof system which underlies clause and cube learning in QCDCL-based QBF solvers. In QCDCL, the traditional axioms of QRES either select clauses which already appear in the input PCNF ϕ or construct cubes which are implicants of the matrix of ϕ.

To overcome the limited deductive power of the traditional axioms, we presented two generalized axioms to derive clauses and cubes based on checking the satisfiability of ϕ under an assignment A generated in QCDCL. We also formulated a new axiom to derive clauses which relies on an existential abstraction of ϕ and on SAT solving. This abstraction-based axiom leverages QU-resolution and allows to overcome the prefix order restriction in QCDCL to some extent. The new axioms can be integrated in QRES and used for clause and cube learning in the QCDCL framework. They are compatible with any variant of Q-resolution, like long-distance resolution [35], QU-resolution [33], and combinations thereof [2].

For axiom applications in practice, *any* complete or incomplete QBF decision procedure can be applied to check the satisfiability of ϕ under assignment A. In this respect, the generalized axioms act as an interface to combining Q-resolution with other QBF decision procedures in QRES. The combination of orthogonal techniques like expansion via the generalized axioms results in variants of QRES which are stronger than traditional QRES with respect to proof complexity. A proof P produced by such variants of QRES can be checked in time which is polynomial in the size of P if subproofs of all clauses and cubes derived by the generalized axioms are provided by the QBF decision procedures.

In order to demonstrate the effectiveness of the newly introduced axioms, we made case studies using the QCDCL solver DepQBF. We applied the preprocessor Bloqqer and SAT solving as incomplete QBF decision procedures in DepQBF to detect axiom applications. Overall, our experiments showed a considerable performance improvement of QCDCL, particularly on application instances.

As future work, it would be interesting to integrate techniques like expansion-based QBF solving more tightly in QCDCL than what we achieved with Bloqqer in our case study. A tighter integration would allow to reduce the run time overhead we observed in practice. Further research directions include axiom applications based on different QBF solving techniques in parallel QCDCL, and potential relaxations of the prefix order in assignments used for axiom applications.

References

1. Ayari, A., Basin, D.A.: QUBOS: Deciding Quantified Boolean Logic Using Propositional Satisfiability Solvers. In: FMCAD. LNCS, vol. 2517, pp. 187–201. Springer (2002)
2. Balabanov, V., Widl, M., Jiang, J.R.: QBF Resolution Systems and Their Proof Complexities. In: SAT. LNCS, vol. 8561, pp. 154–169. Springer (2014)
3. Benedetti, M., Mangassarian, H.: QBF-Based Formal Verification: Experience and Perspectives. JSAT 5(1-4), 133–191 (2008)
4. Beyersdorff, O., Chew, L., Janota, M.: On Unification of QBF Resolution-Based Calculi. In: MFCS. LNCS, vol. 8635, pp. 81–93. Springer (2014)
5. Beyersdorff, O., Chew, L., Janota, M.: Proof Complexity of Resolution-based QBF Calculi. In: STACS. Leibniz International Proceedings in Informatics (LIPIcs), vol. 30, pp. 76–89. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2015)
6. Biere, A.: Resolve and Expand. In: SAT. LNCS, vol. 3542, pp. 59–70. Springer (2004)
7. Biere, A.: PicoSAT Essentials. JSAT 4(2-4), 75–97 (2008)
8. Biere, A., Lonsing, F., Seidl, M.: Blocked Clause Elimination for QBF. In: CADE. LNCS, vol. 6803, pp. 101–115. Springer (2011)
9. Cadoli, M., Giovanardi, A., Schaerf, M.: An Algorithm to Evaluate Quantified Boolean Formulae. In: AAAI. pp. 262–267. AAAI Press / The MIT Press (1998)
10. Cook, S.A., Reckhow, R.A.: The Relative Efficiency of Propositional Proof Systems. J. Symb. Log. 44(1), 36–50 (1979)
11. Giunchiglia, E., Marin, P., Narizzano, M.: Reasoning with Quantified Boolean Formulas. In: Handbook of Satisfiability, FAIA, vol. 185, pp. 761–780. IOS Press (2009)
12. Giunchiglia, E., Narizzano, M., Tacchella, A.: Clause/Term Resolution and Learning in the Evaluation of Quantified Boolean Formulas. JAIR 26, 371–416 (2006)
13. Heule, M., Järvisalo, M., Lonsing, F., Seidl, M., Biere, A.: Clause Elimination for SAT and QSAT. JAIR 53, 127–168 (2015)
14. Heule, M., Seidl, M., Biere, A.: A Unified Proof System for QBF Preprocessing. In: IJCAR. LNCS, vol. 8562, pp. 91–106. Springer (2014)
15. Janota, M., Jordan, C., Klieber, W., Lonsing, F., Seidl, M., Van Gelder, A.: The QBF Gallery 2014: The QBF Competition at the FLoC Olympic Games. JSAT 9, 187–206 (2015)
16. Janota, M., Marques-Silva, J.: Expansion-based QBF solving versus Q-resolution. Theor. Comput. Sci. 577, 25–42 (2015)
17. Janota, M., Marques-Silva, J.: Solving QBF by Clause Selection. In: IJCAI. pp. 325–331. AAAI Press (2015)
18. Janota, M., Klieber, W., Marques-Silva, J., Clarke, E.: Solving QBF with counterexample guided refinement. Artif. Intell. 234, 1–25 (2016)
19. Kleine Büning, H., Karpinski, M., Flögel, A.: Resolution for Quantified Boolean Formulas. Inf. Comput. 117(1), 12–18 (1995)
20. Klieber, W., Sapra, S., Gao, S., Clarke, E.M.: A Non-prenex, Non-clausal QBF Solver with Game-State Learning. In: SAT. LNCS, vol. 6175, pp. 128–142. Springer (2010)
21. Letz, R.: Lemma and Model Caching in Decision Procedures for Quantified Boolean Formulas. In: TABLEAUX. LNCS, vol. 2381, pp. 160–175. Springer (2002)
22. Lonsing, F., Bacchus, F., Biere, A., Egly, U., Seidl, M.: Enhancing Search-Based QBF Solving by Dynamic Blocked Clause Elimination. In: LPAR. LNCS, vol. 9450, pp. 418–433. Springer (2015)

23. Lonsing, F., Biere, A.: Failed Literal Detection for QBF. In: SAT. LNCS, vol. 6695, pp. 259–272. Springer (2011)
24. Lonsing, F., Egly, U., Seidl, M.: Q-Resolution with Generalized Axioms. CoRR abs/1604.05994 (2016), `http://arxiv.org/abs/1604.05994`, SAT 2016 proceedings version with appendix
25. Lonsing, F., Seidl, M., Van Gelder, A.: The QBF Gallery: Behind the scenes. Artif. Intell. 237, 92–114 (2016)
26. Rabe, M.N., Tentrup, L.: CAQE: A Certifying QBF Solver. In: FMCAD. pp. 136–143. IEEE (2015)
27. Reimer, S., Pigorsch, F., Scholl, C., Becker, B.: Enhanced Integration of QBF Solving Techniques. In: Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen (MBMV). pp. 133–143. Verlag Dr. Kovac (2012)
28. Samer, M., Szeider, S.: Backdoor Sets of Quantified Boolean Formulas. JAR 42(1), 77–97 (2009)
29. Samulowitz, H., Bacchus, F.: Using SAT in QBF. In: CP. LNCS, vol. 3709, pp. 578–592. Springer (2005)
30. Samulowitz, H., Davies, J., Bacchus, F.: Preprocessing QBF. In: CP. LNCS, vol. 4204, pp. 514–529. Springer (2006)
31. Silva, J.P.M., Lynce, I., Malik, S.: Conflict-Driven Clause Learning SAT Solvers. In: Handbook of Satisfiability, FAIA, vol. 185, pp. 131–153. IOS Press (2009)
32. Tu, K., Hsu, T., Jiang, J.R.: QELL: QBF Reasoning with Extended Clause Learning and Levelized SAT Solving. In: SAT. LNCS, vol. 9340, pp. 343–359. Springer (2015)
33. Van Gelder, A.: Contributions to the Theory of Practical Quantified Boolean Formula Solving. In: CP. LNCS, vol. 7514, pp. 647–663. Springer (2012)
34. Zhang, L., Malik, S.: Conflict Driven Learning in a Quantified Boolean Satisfiability Solver. In: ICCAD. pp. 442–449. ACM / IEEE Computer Society (2002)
35. Zhang, L., Malik, S.: Towards a Symmetric Treatment of Satisfaction and Conflicts in Quantified Boolean Formula Evaluation. In: CP. LNCS, vol. 2470, pp. 200–215. Springer (2002)