On Expansion and Resolution in CEGAR Based QBF Solving

Leander Tentrup(✉)

Saarland University, Saarbrücken, Germany
tentrup@react.uni-saarland.de

Abstract. A quantified Boolean formula (QBF) is a propositional formula extended with universal and existential quantification over propositions. There are two methodologies in CEGAR based QBF solving techniques, one that is based on a refinement loop that builds partial expansions and a more recent one that is based on the communication of satisfied clauses. Despite their algorithmic similarity, their performance characteristics in experimental evaluations are very different and in many cases orthogonal. We compare those CEGAR approaches using proof theory developed around QBF solving and present a unified calculus that combines the strength of both approaches. Lastly, we implement the new calculus and confirm experimentally that the theoretical improvements lead to improved performance.

1 Introduction

Efficient solving techniques for Boolean theories are an integral part of modern verification and synthesis methods. Especially in synthesis, the amount of choice in the solution space leads to propositional problems of enormous size. Quantified Boolean formulas (QBFs) have repeatedly been considered as a candidate theory for synthesis approaches [6,7,10–12,24] and recent advances in QBF solvers give rise to hope that QBF may help to increase the scalability of those approaches.

Solving quantified Boolean formulas (QBF) using partial expansions in a counterexample guided abstraction and refinement (CEGAR) loop [16] has proven to be very successful. From its introduction, the corresponding solver RAReQS won several QBF competitions. In recent work, a different kind of CEGAR algorithms have been proposed [18,25], implemented in the solvers Qesto and CAQE. All those CEGAR approaches share algorithmic similarities like working recursively over the structure of the quantifier prefix and using SAT solver to enumerate candidate solutions. However, instead of using partial expansions of the QBF as RAReQS does, newer approaches base their refinements on whether a set of clauses is satisfied or not. Despite those algorithmic similarities, the performance characteristics of the resulting solver in experimental

Supported by the European Research Council (ERC) Grant OSARES (No. 683300).

R. Majumdar and V. Kunčak (Eds.): CAV 2017, Part II, LNCS 10427, pp. 475–494, 2017.
DOI: 10.1007/978-3-319-63390-9_25

evaluations are very different and in many cases orthogonal: While RAReQS tends to perform best on instances with a low number of quantifier alternations, Qesto and CAQE have an advantage in instances with many alternations [25].

Proof theory has been repeatedly used to improve the understanding of different solving techniques. For example, the proof calculus ∀Exp+Res [17] has been developed to characterize aspects of expansion-based solving. In this paper, we introduce a new calculus ∀Red+Res that corresponds to the clausal-based CEGAR approaches [18,25]. The levelized nature of those algorithms are reflected by the rules of this calculus, universal reduction and propositional resolution, which are applied to blocks of quantifiers. We show that this calculus is inherently different to ∀Exp+Res explaining the empirical performance results. In detail, we show that ∀Red+Res polynomial simulates level-ordered Q-resolution. We also discuss an extension to ∀Red+Res that was already proposed as solving optimizations [25] and show that this extension makes the resulting calculus exponential more concise.

Further, we integrate the ∀Exp+Res calculus as a rule that can be used within the ∀Red+Res calculus, leading to a unified proof calculus for all current CEGAR approaches. We show that the unified calculus is exponential stronger than both ∀Exp+Res and ∀Red+Res, as well as just applying both simultaneously. This unified calculus serves as a base for implementing an expansion refinement in the QBF solver CAQE. On standard benchmark sets, the combined approach leads to a significant empirical improvement over the previous implementation.

2 Preliminaries

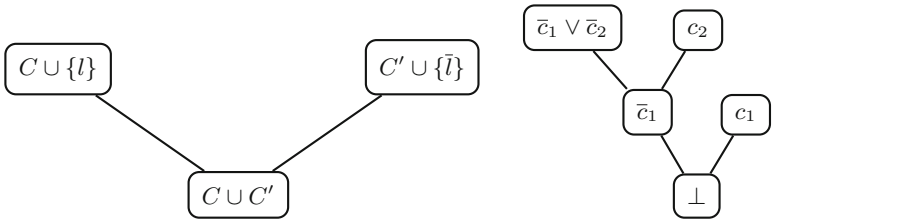
2.1 Quantified Boolean Formulas

We consider quantified Boolean formulas in prenex conjunctive normal form (PCNF), that is a formula consisting of a linear and consecutive quantifier prefix as well as a propositional matrix. A *matrix* is a set of clauses, and a clause is a disjunctive combination of *literals* l, that is either a variable or its negation.

Given a clause $C = (l_1 \vee l_2 \vee \ldots \vee l_n)$, we use set notation interchangeably, that is C is also represented by the set $\{l_1, l_2, \ldots, l_n\}$. Furthermore, we use standard set operations, such as union and intersection, to work with clauses.

For readability, we lift the quantification over variables to the quantification over sets of variables and denote a maximal consecutive block of quantifiers of the same type $\forall x_1.\forall x_2.\cdots\forall x_n.\varphi$ by $\forall X.\varphi$ and $\exists x_1.\exists x_2.\cdots\exists x_n.\varphi$ by $\exists X.\varphi$, accordingly, where $X = \{x_1, \ldots, x_n\}$.

Given a set of variables X, an *assignment* of X is a function $\alpha : X \to \mathbb{B}$ that maps each variable $x \in X$ to either true ($\top$) or false ($\bot$). When the domain of α is not clear from context, we write α_X. We use the instantiation of a QBF Φ by assignment α, written $\Phi[\alpha]$ which removes quantification over variables in $\mathrm{dom}(\alpha)$ and replaces occurrences of $x \in \mathrm{dom}(\alpha)$ by $\alpha(x)$. We write $\alpha \vDash \varphi$ if the assignment α satisfies a propositional formula φ, i.e., $\varphi[\alpha] \equiv \top$.



(a) Resolution rule

(b) Resolution proof for formula $(c_1)(c_2)(\bar{c}_1 \vee \bar{c}_2)$

Fig. 1. Visualization of the resolution rule as a graph.

2.2 Resolution

Propositional resolution is a well-known method for refuting propositional formulas in conjunctive normal form (CNF). The resolution rule allows to *merge* two clauses that contain the same variable, but in opposite signs.

$$\frac{C \cup \{l\} \quad C' \cup \{\bar{l}\}}{C \cup C'} \text{ res}$$

A resolution proof π is a series of applications of the resolution rule. A propositional formula is unsatisfiable if there is a resolution proof that derives the empty clause. We visualize resolution proofs by a graph where the nodes with indegree 0 are called the leaves and the unique node with outdegree 0 is called the root. We depict the graph representation of a resolution proof in Fig. 1(b). The *size* of a resolution proof is the number of nodes in the graph.

2.3 Proof Systems

We consider proof systems that are able to refute quantified Boolean formulas. To enable comparison between proof systems, one uses the concept of *polynomial simulation*. A proof system P polynomially simulates (p-simulates) P' if there is a polynomial p such that for every number n and every formula Φ it holds that if there is a proof of Φ in P' of size n, then there is a proof of Φ in P whose size is less than $p(n)$. We call P and P' polynomial equivalent, if P' additionally p-simulates P.

A refutation based calculus (such as resolution) is regarded as a proof system because it can refute the negation of a formula.

Figure 2 gives an overview over the proof systems introduced in this paper and their relation. An edge $P \to P'$ means that P p-simulates P' (transitive edges are omitted). A dashed line indicates incomparability results.

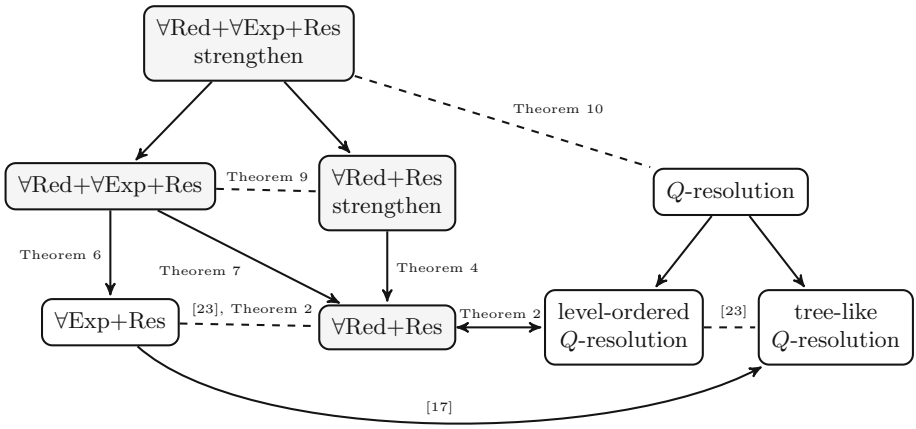


Fig. 2. Overview of the proof systems and their relations. Solid arrows indicate p-simulation relation. Dashed lines indicate incomparability results. The gray boxes are the ones introduced in this paper.

3 Proof Calculi

Given a PCNF formula $Q\,X_1 \ldots Q\,X_n.\, \bigwedge_{1 \leq i \leq m} C_i$. We define a function $\mathit{lit}(i,k)$ that returns the literals of clause C_i that are bound at quantifier level k ($1 \leq k \leq n$). Further, we generalize this definition to $\mathit{lit}(i, > k)$ and $\mathit{lit}(i, < k)$ that return the literals bound after (before) level k. We define $\mathit{lit}(i,0) = \mathit{lit}(i, n+1) = \emptyset$ for every $1 \leq i \leq m$. We use $\mathcal{C}$ to denote a set of clauses and $Q_k \in \{\exists, \forall\}$ to denote the quantification type of level k.

3.1 A Proof System for Clausal Abstractions

We start by defining the object on which our proof system ∀Red+Res is based on. A *proof object* $\mathcal{P}^k$ consists of a set of indices $\mathcal{P}$ where an index $i \in \mathcal{P}$ represents the i-th clause in the original matrix and k denotes the k-th level of the quantifier hierarchy. We define an operation $\mathit{lit}(\mathcal{P}^k) = \bigcup_{i \in \mathcal{P}} \mathit{lit}(i,k)$, that gives access to the literals of clauses contained in $\mathcal{P}^k$. The leaves in our proof system are singleton sets $\{i\}^z$ where z is the maximum quantification level of all literals in clause C_i. The root of a refutation proof is the proof object $\mathcal{P}^0$ that represents the empty set, i.e., $\mathit{lit}(\mathcal{P}^0) = \emptyset$.

The rules of the proof system is given in Fig. 3. It consists of three rules, an axiom rule (init) that generates leaves, a resolution rule (res), and a universal reduction rule (∀red). The latter two rules enable to transform a premise that is related to quantifier level k into a conclusion that is related to quantifier level $k-1$. The universal reduction rule and the resolution rule are used for universal and existential quantifier blocks, respectively.

$$\frac{\mathcal{P}_1^k \quad \cdots \quad \mathcal{P}_j^k \quad \pi}{\left(\bigcup_{i\in\{1,\ldots,j\}} \mathcal{P}_i\right)^{k-1}}\ \text{res} \qquad \begin{array}{l} Q_k = \exists \\ \pi \text{ is a resolution refutation proof for } \bigwedge_{1\le i\le j} lit(\mathcal{P}_i^k) \end{array}$$

$$\frac{\mathcal{P}^k}{\mathcal{P}^{k-1}}\ \forall\text{red} \qquad \begin{array}{l} Q_k = \forall \\ \forall l \in lit(\mathcal{P}^k).\, \bar{l} \notin lit(\mathcal{P}^k) \end{array}$$

$$\frac{}{\{i\}^k}\ \text{init} \qquad \begin{array}{l} 1 \le i \le m \\ lit(i, > k) = \emptyset \end{array}$$

Fig. 3. The rules of the ∀Red+Res calculus.

Resolution Rule. There is a close connection between (res) and the propositional resolution as (res) merges a number of proof objects $\mathcal{P}_i^k$ of level k into a single proof object of level $k-1$. It does so by using a resolution proof for a propositional formula that is constructed from the premises $\mathcal{P}_i^k$. This propositional formula $\bigwedge_{1\le i\le j} lit(\mathcal{P}_i^k)$ contains *only* literals of level k. Intuitively, this rule can be interpreted as follows: a resolution proof over those clauses rules out any possible existential assignment at quantifier level k, thus, one of those clauses has to be satisfied at an earlier level.

Universal Reduction Rule. In contrast to (res), (∀red) works on single proof objects. It can be applied if level k is universal and the premise does not encode a universal tautology, i.e., for every literal $l \in lit(\mathcal{P}^k)$, the negated literal $\bar{l}$ is not contained in $lit(\mathcal{P}^k)$.

Graph Representation. A proof in the ∀Red+Res calculus can be represented as a directed acyclic graph (DAG). The nodes in the DAG are proof objects $\mathcal{P}^k$ and the edges represent applications of (res) and (∀red). The rule (res) is represented by a hyper-edge that is labeled with the propositional resolution proof π. Edges representing the universal reduction can thus remain unlabeled without introducing ambiguity. The *size* of a ∀Red+Res proof is the number of nodes in the graph together with the number of inner (non-leaf, non-root) nodes of the containing propositional resolution proofs.

A *refutation* in the ∀Red+Res calculus is a proof that derives a proof object $\mathcal{P}^0$ at level 0. A proof for some $\mathcal{P}^k$ is a ∀Red+Res proof with root $\mathcal{P}^k$. Thus, a proof for $\mathcal{P}^k$ can be also viewed as a refutation for the formula $Q\,X_{k+1} \ldots Q\,X_n.\ \bigwedge_{i\in\mathcal{P}} lit(i, > k)$ starting with quantifier level $k+1$ and containing clauses represented by $\mathcal{P}$.

Example 1. Consider the following QBF

$$\underbrace{\exists e_1}_{1}.\ \underbrace{\forall u_1}_{2}.\ \underbrace{\exists c_1, c_2}_{3}.\ \underbrace{(\bar{e}_1 \vee c_1)}_{C_1}\underbrace{(\bar{u}_1 \vee c_1)}_{C_2}\underbrace{(e_1 \vee c_2)}_{C_3}\underbrace{(u_1 \vee c_2)}_{C_4}\underbrace{(\bar{c}_1 \vee \bar{c}_2)}_{C_5}. \tag{1}$$

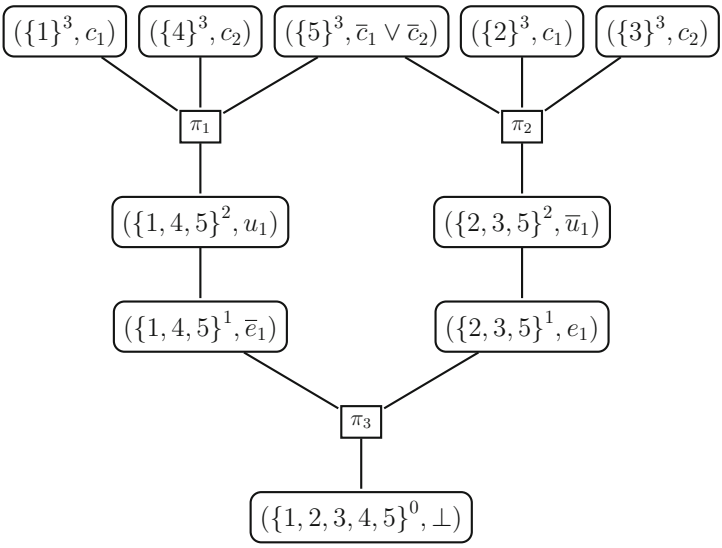


Fig. 4. A ∀Red+Res refutation for formula (1).

The refutation in the ∀Red+Res calculus is given in Fig. 4. In the nodes, we represent the proof objects $\mathcal{P}^k$ in the first component and the represented clause in the second component. The proof follows the structure of the quantifier prefix, i.e., it needs four levels to derive a refutation. The resolution proof π_1 for propositional formula

$$\mathit{lit}(\{1\}^3) \wedge \mathit{lit}(\{4\}^3) \wedge \mathit{lit}(\{5\}^3) \equiv (c_1)(c_2)(\overline{c}_1 \vee \overline{c}_2)$$

is depicted in Fig. 1(b).

In the following, we give a formal correctness argument and compare our calculus to established proof systems. A QBF proof system is *sound* if deriving a proof implies that the QBF is false and it is *refutational complete* if every false QBF has a proof.

Theorem 1. *∀Red+Res is sound and refutational complete for QBF.*

Proof. The completeness proof is carried out by induction over the quantifier prefix.

Induction Base. Let $\exists X.\, \varphi$ be a false QBF and φ be propositional. Then (res) derives some $\mathcal{P}^0$ because resolution is complete for propositional formulas. Let $\forall X.\, \varphi$ be a false QBF and φ be propositional. Picking an arbitrary (non-tautological) clause C_i and applying (∀red) leads to $\{i\}^0$.

Induction Step. Let $\exists X.\, \Phi$ be a false QBF, i.e., for all assignments α_X the QBF $\Phi[\alpha_X]$ is false. Hence, by induction hypothesis, there exists a ∀Red+Res proof

for every $\Phi[\alpha_X]$. We transform those proofs in a way that they can be used to build a proof for Φ. Let P be a proof of $\Phi[\alpha_X]$. P has a distinct root node (representing the empty set), that was derived using (∀red) as $\Phi[\alpha_X]$ starts with a universal quantifier. To embed P in Φ, we increment every level in P by one, as Φ has one additional (existential) quantifier level. Then, instead of deriving the empty set, the former root node derives a proof object of the form $\mathcal{P}^1$. Let N be the set of those former root nodes. By construction, there exists a resolution proof π such that the empty set can be derived by (res) using N (or a subset thereof). Assuming otherwise leads to the contradiction that some $\Phi[\alpha_X]$ is true.

Let $\forall X.\, \Phi$ be a false QBF, i.e., there is an assignment α_X such that the QBF $\Phi[\alpha_X]$ is false. Hence, by induction hypothesis, there exists a ∀Red+Res proof for $\Phi[\alpha_X]$. Applying (∀red) using α_X is a ∀Red+Res proof for Φ.

For soundness it is enough to show that one cannot derive a clause using this calculus that changes the satisfiability. Let $\Phi = Q\, X_1 \ldots Q\, X_n.\, \bigwedge_{1 \leq i \leq m} C_i$ be an arbitrary QBF. For every level k and every $\mathcal{P}^k$ generated by the application of the ∀Red+Res calculus, it holds that Φ and $Q\, X_1 \ldots Q\, X_n.\, \bigwedge_{1 \leq i \leq m} C_i \wedge (\bigvee_{i \in \mathcal{P}} \bigvee_{l \in lit(i, \leq k)} l)$ are equisatisfiable. Assume otherwise, then either (∀red) or (res) have derived a $\mathcal{P}^k$ that would make Φ false. Again, by induction, one can show that if (∀red) derived a $\mathcal{P}^k$ that makes Φ false, the original premise $\mathcal{P}^{k+1}$ would have made Φ false; likewise, if (res) derived a $\mathcal{P}^k$ that makes Φ false, the conjunction of the premises have made Φ false. □

Comparison to Q-resolution Calculus. Q-resolution [19] is an extension of the (propositional) resolution rule to handle universal quantification. The universal reduction rule allows the removal of universal literal u from a clause C if no existential literal $l \in C$ depends on u. There are also additional rules on when the resolution rule can be applied, i.e., it is not allowed to produce tautology clauses using the resolution rule. The definitions of Q-resolution proof and refutation are analogous to the propositional case.

There are two restricted classes of Q-resolution that are commonly considered, that is *level-ordered* and *tree-like* Q-resolution. A Q-resolution proof is level-ordered if resolution of an existential literal l at level k happens before every other existential literal with level $<k$. A Q-resolution proof is tree-like if the graph representing the proof has a tree shape.

As a first result, we show that ∀Red+Res is polynomially equivalent to level-ordered Q-resolution, i.e., a proof in our calculus can be polynomially simulated in level-ordered Q-resolution and vice versa. While this is straightforward from the definitions of both calculi, this is much less obvious if one looks at the underlying algorithms of the CEGAR approaches [18,25] and QCDCL [27].

Theorem 2. *∀Red+Res and level-ordered Q-resolution are p-sim. equivalent.*

Proof. A ∀Red+Res proof can be transformed into a Q-resolution proof by replacing every node $\mathcal{P}^k$ by the clause $(\bigvee_{i \in \mathcal{P}} \bigvee_{l \in lit(i, \leq k)} l)$ and by replacing the hyper-edge labeled with π by a graph representing the applications of the resolution rule. Similarly, a level-ordered Q-resolution proof can be transformed

into a ∀Red+Res proof by a step-wise transformation from leaves to the root. This way, one can track the clauses needed for constructing the proof objects $\mathcal{P}^k$ at every level k. □

Despite being equally powerful, the differences are important and enable the expansion based extension that we will introduce in the next section. One difference is that our calculus only reasons about literals of one quantifier level, which allows us to use plain resolution without any changes (as are needed in Q-resolution). Further, the proof rules capture the fact that only proof obligations are communicated between the quantifier levels of the QBF. An immediate consequence is that every refutation in the proof system is DAG-like and has exactly depth $k + 1$.

Since the level-ordering constraint imposes an order on the resolution, the size of the refutation proof may be exponentially larger for some formulas [14]. Hence, also ∀Red+Res is in general exponentially weaker than unrestricted Q-resolution. In practice, and already noted by Janota and Marques-Silva [17], solvers that are based on Q-resolution proofs produce level-ordered Q-resolution.

In the initial version of CAQE [25] an optimization that can generate new resolvents at level k without recursion into deeper levels was described. We model this optimization as a new rule extending the ∀Red+Res calculus and show that this rule leads to an exponential separation.

Strong UNSAT Rule. In the implementation of CAQE, we used an optimization which we called *strong UNSAT refinement* [25], that allowed the solver to strengthen a certain type of refinements. The basic idea behind this optimization is that if the solver determines that, at an existential level k, a certain set of clauses $\mathcal{C}$ cannot be satisfied at the same time, then every alternative set of clauses $\mathcal{C}'$, that is equivalent with respect to the literals in levels $>k$, cannot be satisfied as well. We introduce the following proof rule that formalizes this intuition. We extend proof objects $\mathcal{P}^k$ such that they can additionally contain fresh literals, i.e., literals that were not part of the original QBF. Those literals are treated as they were bound at level k, i.e., they are contained in $lit(\mathcal{P}^k)$ and can thus be used in the premise of the rule (res), but are not contained in the conclusion $\mathcal{P}^{k-1}$.

$$\frac{(\mathcal{P} \cup \{i\})^k}{(\{a\} \cup \mathcal{P})^k \quad \{\overline{a}, j_1\}^k \quad \cdots \quad \{\overline{a}, j_n\}^k}\ \text{strengthen}$$

$Q_k = \exists$,
$lit(j, > k) \subseteq lit(i, > k)$
for all $j \in \{j_1, \ldots, j_n\}$,
a fresh var.

Theorem 3. *The strengthening rule is sound.*

Proof. In a resolution proof at level k, one can derive the proof objects $(\mathcal{P} \cup \{j\})^k$ for $j \in \{j_1, \ldots, j_n\}$ using the conclusion of the strengthening rule. Assume we have a proof for $(\mathcal{P} \cup \{i\})^k$ (premise), then the quantified formula $\forall X_{k+1} \ldots Q\, X_n. \bigwedge_{i^* \in \mathcal{P}} lit(i^*, > k) \wedge lit(i, > k)$ is false. Thus, the QBF with the same quantifier prefix and matrix, extended by some clause $lit(j, > k)$ for

$j \in \{j_1, \ldots, j_n\}$, is still false. Since every C_j subsumes C_i with respect to quantifier level greater than k ($lit(j, > k) \subseteq lit(i, > k)$), the clause $lit(i, > k)$ can be eliminated without changing satisfiability. Thus, the resulting quantified formula $\forall X_{k+1} \ldots Q\, X_n . \bigwedge_{i^* \in \mathcal{P}} lit(i^*, > k) \wedge lit(j, > k)$ is false and there exists a $\forall$Red+Res proof for $(\mathcal{P} \cup \{j\})^k$. □

Theorem 4. *The proof system without strengthening rule does not p-simulate the proof system with strengthening rule.*

Proof. We use the family of formulas CR_n that was used to show that level-ordered Q-resolution cannot p-simulate $\forall$Exp+Res [17]. We show that CR_n has a polynomial refutation in the $\forall$Red+Res calculus with strengthening rule, but has only exponential refutations without it. The latter follows from Theorem 2 and the results by Janota and Marques-Silva [17].

The formula CR_n has the quantifier prefix $\exists x_{11} \ldots x_{nn} \forall z \exists a_1 \ldots a_n b_1 \ldots b_n$ and the matrix is given by

$$\left(\bigvee_{i \in 1..n} \overline{a}_i\right) \wedge \left(\bigvee_{i \in 1..n} \overline{b}_i\right) \wedge \bigwedge_{i,j \in 1..n} \underbrace{(x_{ij} \vee z \vee a_i)}_{C_{ij}} \wedge \underbrace{(\overline{x}_{ij} \vee \overline{z} \vee b_j)}_{C_{\overline{ij}}}. \tag{2}$$

One can interpret the constraints as selecting rows and columns in a matrix where i selects the row and j selects the column, e.g., for $n = 3$ it can be visualized as follows:

$x_{11} \vee z \vee a_1$	$\overline{x}_{11} \vee \overline{z} \vee b_1$	$x_{12} \vee z \vee a_1$	$\overline{x}_{12} \vee \overline{z} \vee b_2$	$x_{13} \vee z \vee a_1$	$\overline{x}_{13} \vee \overline{z} \vee b_3$
$x_{21} \vee z \vee a_2$	$\overline{x}_{21} \vee \overline{z} \vee b_1$	$x_{22} \vee z \vee a_2$	$\overline{x}_{22} \vee \overline{z} \vee b_2$	$x_{23} \vee z \vee a_2$	$\overline{x}_{23} \vee \overline{z} \vee b_3$
$x_{31} \vee z \vee a_3$	$\overline{x}_{31} \vee \overline{z} \vee b_1$	$x_{32} \vee z \vee a_3$	$\overline{x}_{32} \vee \overline{z} \vee b_2$	$x_{33} \vee z \vee a_3$	$\overline{x}_{33} \vee \overline{z} \vee b_3$

Assume $z \to 0$, then we derive the proof object $\mathcal{P}^1 = \{i1 \mid i \in 1..n\}^1$ ($lit(\mathcal{P}^1) = \bigvee_{i \in 1..n} x_{i1}$) by applying the resolution and reduction rule. Likewise, for $z \to 1$, we derive the proof object $\mathcal{P}^1_0 = \{\overline{1j} \mid j \in 1..n\}^1$ ($lit(\mathcal{P}^1_0) = \bigvee_{j \in 1..n} \overline{x}_{1j}$). Applying the strengthening rule on $\mathcal{P}^1_0$ results in $\mathcal{P}^1_1 = (\{c_1\} \cup \{\overline{1j} \mid j \in 2..n\})^1$ and $\{\overline{c}_1, \overline{11}\}^1, \{\overline{c}_1, \overline{21}\}^1, \ldots, \{\overline{c}_1, \overline{n1}\}^1$ where c_1 is a fresh variable. Further $n-1$ applications of the strengthening rule starting on $\mathcal{P}^1_1$ lead to $\mathcal{P}^1_n = \{c_j \mid j \in 1..n\}^1$ and the proof objects $\{\overline{c}_j, \overline{ij} \mid i, j \in 1..n\}^1$, where c_j are fresh variables, as all clauses in a column are equivalent with respect to the inner quantifiers (contain $\overline{z} \vee b_j$).

Using $\mathcal{P}^1$ and $\{\overline{c}_1, \overline{11}\}^1, \{\overline{c}_1, \overline{21}\}^1, \ldots, \{\overline{c}_1, \overline{n1}\}^1$ from the first strengthening application, we derive the singleton set $\{\overline{c}_1\}$ using n resolution steps ($lit(\mathcal{P}^1) = \bigvee_{i \in 1..n} x_{i1}$ and $lit(\{\overline{c}_1, \overline{i1}\}^1) = \{\overline{c}_1, \overline{x}_{i1}\}$). Analogously, one derives the singletons $\{\overline{c}_2\} \ldots \{\overline{c}_n\}$ and together with $\mathcal{P}^1_n = \{c_j \mid j \in 1..n\}$ the empty set is derived. Thus, there exists a polynomial resolution proof leading to a proof object $\mathcal{P}^0$ and the size of the overall proof is polynomial, too. □

We note that despite being stronger than plain $\forall$Red+Res, the extended calculus is still incomparable to $\forall$Exp+Res.

Corollary 1. *$\forall$Red+Res with strengthening rule does not p-simulate $\forall$Exp+Res.*

Proof. We use a modification of formula CR_n (2), which we call CR'_n in the following. The single universal variable z is replaced by a number of variables z_{ij} for every pair $i, j \in 1..N$. It follows that the strengthening rule is never applicable and hence, the proof system is as strong as level-ordered Q-resolution which has an exponential refutation of CR_n while $\forall$Exp+Res has a polynomial refutation since the expansion tree has still only two branches [17]. □

When compared to Q-resolution, the strengthening rule can be interpreted as a step towards breaking the level-ordered constraint inherent to $\forall$Red+Res. The calculus, however, is not as strong as Q-resolution.

Corollary 2. *$\forall$Red+Res with strengthening rule does not p-sim. Q-resolution.*

Proof. The formula CR'_n from the previous proof has a polynomial (tree-like) Q-resolution proof. The proof for CR_n given by Mahajan and Shukla [23] can be modified for CR'_n. □

Both results follow from the fact that the strengthening rule as presented is not applicable to the formula CR'_n. Where in CR_n, the clauses $C_{\overline{ij}}$ are equal with respect to the inner quantifier when j is fixed ($\overline{z} \vee b_j$), in CR'_n they are all different ($\overline{z}_{ij} \vee b_j$). This difference is only due to the universal variables z_{ij}. Thus, we propose a stronger version of the strengthening rule that does the subset check only on the existential variables. For the universal literals, one additionally has to make sure that no resolvent produces a tautology (as it is the case in CR'_n). We leave the formalization to future work.

3.2 Expansion

The levelized nature of the proof system allows us to introduce additional rules that can reason about quantified subformulas. In the following, we introduce such a rule that allows us to use the $\forall$Exp+Res calculus [17] within a $\forall$Red+Res proof.

We start by giving necessary notations used to define $\forall$Exp+Res. We refer the reader to [17] for further information.

Definition 1 (adapted from [17])

- *A $\forall$-expansion tree for QBF Φ with u universal quantifier blocks is a rooted tree $\mathcal{T}$ such that every path $p_0 \xrightarrow{\alpha_1} p_1 \cdots \xrightarrow{\alpha_u} p_u$ in $\mathcal{T}$ from the root p_0 to some leaf p_u has exactly u edges and each edge $p_{i-1} \xrightarrow{\alpha_i} p_i$ is labeled with a total assignment α_u to the universal variables at universal level u. Each path in $\mathcal{T}$ is uniquely defined by its labeling.*
- *Let $\mathcal{T}$ be a $\forall$-expansion tree and $P = p_0 \xrightarrow{\alpha_1} p_1 \cdots \xrightarrow{\alpha_u} p_u$ be a path from the root p_0 to some leaf p_u.*
  1. *For an existential variable x we define expand-var$(P, x) = x^\alpha$ where x^α is a fresh variable and α is the universal assignment of the dependencies of x.*

2. *For a propositional formula φ define $expand(P, \varphi)$ as instantiating φ with $\alpha_1, \ldots, \alpha_u$ and replacing every existential variable x by $expand\text{-}var(P, x)$.*
3. *Define $expand(\mathcal{T}, \Phi)$ as the conjunction of all $expand(P, \varphi)$ for each root-to-leaf P in $\mathcal{T}$.*

In difference to previous work, we allow to use the expansion rule on quantified subformulas of Φ additionally to applying it to Φ directly. By $\mathcal{C}^{\geq k}$ we denote a set of clauses that only contain literals bound at level $\geq k$.

$$\frac{\mathcal{T} \quad \mathcal{C}^{\geq k} \quad \pi}{\mathcal{P}^{k-1}} \ \forall\text{exp-res} \qquad \begin{array}{l} Q_k = \exists, \pi \text{ is a resolution refutation of the expansion} \\ \text{formula } expand(\mathcal{T}, \exists X_k. \forall X_{k+1} \ldots \exists X_m. \mathcal{C}^{\geq k}) \\ \mathcal{P}^{k-1} = \{i \mid C_i \in \mathcal{C}\}^{k-1} \end{array}$$

The rule states that if there is a universal expansion of the quantified Boolean formula $\exists X_k. \forall X_{k+1} \ldots \exists X_m. \mathcal{C}^{\geq k}$ and a resolution refutation π for this expansion, then there is no existential assignment that satisfies clauses $\mathcal{C}$ from level k. The size of the expansion rule is the sum of the size of the expansion tree and resolution proof [17].

Example 2. We demonstrate the interplay between (∀exp-res) and the ∀Red+Res calculus on the following formula

$$\overbrace{\exists e_1}^{1}. \overbrace{\forall u_1}^{2}. \overbrace{\exists c_1, c_2}^{3}. \overbrace{\forall a}^{4}. \overbrace{\exists b}^{5}. \overbrace{\exists x}^{5}. \overbrace{\forall z}^{6}. \overbrace{\exists t}^{7}.$$
$$\underbrace{(\bar{e}_1 \vee c_1)}_{1} \underbrace{(\bar{u}_1 \vee c_1)}_{2} \underbrace{(e_1 \vee c_2)}_{3} \underbrace{(u_1 \vee c_2)}_{4} \underbrace{(\bar{c}_1 \vee \bar{c}_2 \vee \bar{b} \vee \bar{a})}_{5} \underbrace{(z \vee t \vee b)}_{6} \underbrace{(\bar{z} \vee \bar{t})}_{7} \underbrace{(x \vee \bar{t})}_{8} \underbrace{(\bar{x} \vee t)}_{9}$$

To apply (∀exp-res), we use the clauses 5–9 from quantifier level 5, i.e., $\mathcal{C}^{\geq 5} = \{(\bar{b})(z \vee t \vee b)(\bar{z} \vee \bar{t})(x \vee \bar{t})(\bar{x} \vee t)\}$. The corresponding quantifier prefix is $\exists b \exists x \forall z \exists t$. Using the complete expansion of z ($\{z \to 0, z \to 1\}$) as the expansion tree $\mathcal{T}$, we get the following expansion formula

$$(\bar{b})(t^{\{z\to 0\}} \vee b)(x \vee \bar{t}^{\{z\to 0\}})(\bar{x} \vee t^{\{z\to 0\}})(\bar{t}^{\{z\to 1\}})(x \vee \bar{t}^{\{z\to 1\}})(\bar{x} \vee t^{\{z\to 1\}}),$$

which has a simple resolution proof π. The conclusion of (∀exp-res) leads to the proof object $\{5, 6, 7, 8, 9\}^4$, but only clause 5 contains literals bound before quantification level 5. After a universal reduction, the proof continues as described in Example 1.

Theorem 5. *The ∀exp-res rule is sound.*

Proof. Assume otherwise, then one would be able to derive a proof object $\mathcal{P}^{k-1}$ that is part of a ∀Red+Res refutation proof for true QBF Φ. Thus, the clause corresponding to $\mathcal{P}^{k-1}$ (cf. proof of Theorem 1) $(\bigvee_{i \in \mathcal{P}} \bigvee_{l \in lit(i, <k)} l)$ made Φ false. However, the same clause can be derived directly by applying the expansion $\mathcal{T}$ to the original QBF, i.e., expanding universal variables beginning with quantification level $k+1$, and propositional resolution on the resulting expansion formula. Thus, this clause can be conjunctively added to the matrix without changing satisfiability, leading to a contradiction. □

The resulting proof system can be viewed as a unification of the currently known CEGAR approaches for solving quantified Boolean formulas [16,18,25].

Theorem 6. *$\forall$Exp+Res does not p-simulate $\forall$Red+$\forall$Exp+Res.*

Proof. $\forall$Exp+Res does not p-simulate level-ordered Q-resolution [23]. □

The combination of both rules makes the proof system stronger than merely choosing between expansion and resolution proof upfront.

Theorem 7. *There is a QBF that has polynomial refutation in $\forall$Red+$\forall$Exp+Res, but has only exponential refutations in $\forall$Red+Res and $\forall$Exp+Res.*

Proof. For this proof, we take two formulas that are hard for Q-resolution and $\forall$Exp+Res, respectively. We build a new family of formulas that has a polynomial refutation in $\forall$Red+$\forall$Exp+Res, but only exponential refutations in $\forall$Red+Res and $\forall$Exp+Res.

The first formula we consider is formula (2) form [17], that we call DAG_n in the following:

$$\exists e_1 \forall u_1 \exists c_1 c_2 \dots \exists e_n \forall u_n \exists c_{2n-1} c_{2n}.$$
$$\Big(\bigvee_{i \in 1 \dots 2n} \overline{c}_i\Big) \wedge \bigwedge_{i \in 1 \dots n} (\overline{e}_i \vee c_{2i-1}) \wedge (\overline{u}_i \vee c_{2i-1}) \wedge (e_i \vee c_{2i}) \wedge (u_i \vee c_{2i})$$

It is known that DAG_n has a polynomial level-ordered Q-resolution proof and only exponential $\forall$Exp+Res proofs [17]. As a second formula, we use the $\mathrm{QParity}_n$ formula [2]

$$\exists x_1 \dots x_n \forall z \exists t_2 \dots t_n. \operatorname{xor}(x_1, x_2, t_2) \wedge \bigwedge_{i \in 3 \dots n} \operatorname{xor}(t_{i-1}, x_i, t_i) \wedge (z \vee t_n) \wedge (\overline{z} \vee \overline{t}_n)$$

where $\operatorname{xor}(o_1, o_2, o) = (\overline{o}_1 \vee \overline{o}_2 \vee \overline{o}) \wedge (o_1 \vee o_2 \vee \overline{o}) \wedge (\overline{o}_1 \vee o_2 \vee o) \wedge (o_1 \vee \overline{o}_2 \vee o)$ defines o to be equal to $o_1 \oplus o_2$. $\mathrm{QParity}_n$ has a polynomial $\forall$Exp+Res refutation but only exponential Q-resolution refutations [2]. We construct the following formula

$$\exists e_1 \forall u_1 \exists c_1 c_2 \dots \exists e_n \forall u_n \exists c_{2n-1} c_{2n}. \forall a \exists b. \exists x_1 \dots x_n \forall z \exists t_2 \dots t_n.$$
$$\bigwedge_{i \in 1 \dots n} (\overline{e}_i \vee c_{2i-1}) \wedge (\overline{u}_i \vee c_{2i-1}) \wedge (e_i \vee c_{2i}) \wedge (u_i \vee c_{2i}) \wedge$$
$$(\overline{a} \vee \overline{b} \vee \bigvee_{i \in 1 \dots 2n} \overline{c}_i) \wedge \operatorname{xor}(x_1, x_2, t_2) \wedge \bigwedge_{i \in 3 \dots n} \operatorname{xor}(t_{i-1}, x_i, t_i) \wedge (z \vee t_n \vee b) \wedge (\overline{z} \vee \overline{t}_n)$$

We argue in the following that this formula has a polynomial refutation in $\forall$Red+$\forall$Exp+Res. First, using ($\forall$exp-res) we can derive the proof object containing the clause $(\overline{a} \vee \bigvee_{i \in \{1 \dots 2n\}} \overline{c}_i)$ using the expansion tree $\mathcal{T} = \{z \to 0, z \to 1\}$ and the clauses from the last row (analogue to Example 2). After applying universal reduction, the proof object representing clause $(\bigvee_{i \in \{1 \dots 2n\}} \overline{c}_i)$ can be derived. For the remaining formula, there is a polynomial and level-ordered resolution proof [17], thus, the formula has a polynomial $\forall$Red+$\forall$Exp+Res proof.

There is no polynomial Q-resolution proof, because deriving $(\bigvee_{i\in\{1\ldots 2n\}} \overline{c}_i)$ is exponential in Q-resolution. Likewise, there is no polynomial ∀Exp+Res proof as the formula after deriving this clause has only exponential ∀Exp+Res refutations. □

One question that remains open, is how the new proof system compares to unrestricted Q-resolution. We already know that the new proof system polynomially simulates both tree-like Q-resolution as well as level-ordered Q-resolution.

Theorem 8. *∀Red+∀Exp+Res does not p-simulate Q-resolution.*

Proof (Sketch). We construct a formula that is hard for expansion and level-ordered Q-resolution, but easy for (unrestricted) Q-resolution. We have already seen in the proof of Theorem 7 that DAG_n is hard for ∀Exp+Res but easy for Q-resolution. However, the Q-resolution proof of DAG_n is level-ordered. Hence, we need an additional formula that is hard to refute for level-ordered Q-resolution. We use the modified pigeon hole formula from [14] where unrestricted resolution has polynomial proofs and resolution proofs that are restricted to a certain variable ordering are exponential. Using universal quantification, one can impose an arbitrary order on a level-ordered Q-resolution proof, thus, there is a quantified Boolean formula which has only exponential level-ordered Q-resolution but has a polynomial Q-resolution proof. The disjunction of those two formulas gives the required witness. This formula is easy to refute for Q-resolution, but the first one is hard for ∀Exp+Res and the second is hard for level-ordered Q-resolution. □

3.3 Comparison Between Extensions

We conclude this section by comparing the two extensions of the ∀Red+Res calculus introduced in this paper.

Theorem 9. *∀Red+∀Exp+Res and ∀Red+Res with strengthening rule are incomparable.*

Proof (Sketch). The family of formulas CR'_n from proof of Corollary 1 separates ∀Red+∀Exp+Res and ∀Red+Res with strengthening rule. Since the strengthening rule is not applicable, all ∀Red+Res proofs are exponential while there is a polynomial proof in ∀Red+∀Exp+Res.

The other direction is shown by using a similar construction as the one used in the proof of Theorem 7. We use a combination of CR_n and DAG_n to construct a formula that has only exponential refutations in ∀Red+∀Exp+Res, but a polynomial refutation using the strengthening rule. The formula DAG_n is used to generate the premise for the application of the strengthening rule to solve CR_n. To generate this premise using the rule (∀exp-res) one needs an exponential proof. There is a polynomial proof for DAG_n in ∀Red+Res, but there is none for CR_n, thus, ∀Red+∀Exp+Res has only exponential refutations. □

Algorithm 1. Modified CEGAR solving loop for existential quantifier

```
1: φ_k is the propositional abstraction for quantifier ∃X_k
2: procedure SOLVE∃(∃X_k.Ψ, P^k)
3:     while true do
4:         disable clauses C_i^k of φ_k where i ∉ P^k   ▷ those C_i are already satisfied < k
5:         generate candidate solution P_*^{k+1} using SAT solver and abstraction φ_k
6:         if no candidate exists then                  ▷ there is a resolution proof π
7:             return UNSAT, P^{k-1}
8:         else if Ψ is propositional then              ▷ base case for structural recursion
9:             return SAT, witness
10:        verify candidate recursively, call SOLVE∀(Ψ, P_*^{k+1})
11:        if candidate correct then
12:            return SAT, witness
13:        else
14:            counterexample consists of P_ce^k and expansion tree T
15:            refine φ_k such that one clause C_i^k in with i ∈ P_ce must be satisfied
16:            refine φ_k with abstraction of expansion of Φ with respect to T
```

Theorem 10. *∀Red+∀Exp+Res with strengthening rule and Q-resolution are incomparable.*

Proof. Follows from the proof of Theorem 8 as the witnessing formula can be constructed such that the strengthening rule is not applicable. The other direction follows from the separation of Q-resolution and ∀Exp+Res by Beyersdorff et al. [2]. □

4 Experimental Evaluation

4.1 Implementation

We extended the implementation of CAQE with the possibility to use the rule (∀exp-res) as introduced in Sect. 3.2[1]. While the rule is applicable at every level in the QBF in principle, the effectiveness decreases when applying it to deeply nested formulas where CAQE tends to perform better [25] than RAReQS. We aim to strike a balance between expansion and clausal-abstraction, i.e., keeping the best performance characteristics of both solving methods. Thus, in our implementation, we apply the expansion refinement (additional to the clausal-abstraction refinement) to the innermost universal quantifier.

An overview of the CEGAR algorithm is given in Algorithm 1. There is a close connection between the rules of the ∀Red+Res calculus and the presented algorithm. Especially, we use a SAT solver to prove the refutation needed in the rule (res). We refer to [25] for algorithmic details. Changes to the original algorithm are written in bold text.

[1] CAQE is available online at https://react.uni-saarland.de/tools/caqe/.

Abstraction. The abstraction for quantifier $\exists X_k$, written φ_k is the projection of the clauses of the matrix to variables in X_k, i.e., $\bigwedge_{1 \leq i \leq m} lit(i,k)$. We assume that there is a operation to "disable" clauses in φ_k which corresponds to the situation where a clause C_i is satisfied by some variable bound before k. Likewise, for every clause we allow the assumption that this clause will be satisfied by a some variable bound after k. This is used to generate candidate proof objects $\mathcal{P}_*^{k+1}$ for inner levels. In the refinement step, this assumption can be invalidated, i.e., there is a way to force satisfaction of a clause at level k. Those operations can be implemented by an incremental SAT solver and two additional literals controlling the satisfaction of clauses [25].

Algorithm. The algorithm recurses on the structure of the quantifier prefix and communicates proof objects $\mathcal{P}$, which indicate the clauses of the matrix that are satisfied. At an existential quantifier, the abstraction generates a candidate solution (line 5) and checks recursively whether the candidate is correct (line 10). If not, the counterexample originally consists of a set of clauses (which could not be satisfied from the inner existential quantifiers). We extend this counterexample to also include an expansion tree $\mathcal{T}$ from the levels below. Additionally to the original refinement, we also build the expansion of the QBF with respect to the expansion tree $\mathcal{T}$, resulting in a QBF with the same quantifier prefix as the current level (with additional existential variables due to expansion). This QBF is then translated into a propositional formula in the same way as the original QBF. Lastly, the abstraction φ_k is then conjunctively combined with this propositional formula. Note that if the function returns UNSAT (line 7), the corresponding resolution proof from the SAT solver can be used to apply the rule (res) form the ∀Red+Res calculus.

As the underlying SAT solver in the implementation, we use PicoSAT [3], MiniSat [8], cryptominisat [26], or Lingeling [4].

4.2 Evaluation

In our evaluation, we show that the established theoretical separations shown in the last section translate to a significant empirical improvement. The evaluation is structured by the following three hypothesizes: First, the strengthen and expansion refinement give a significant improvement over the plain version of CAQE. Combining both refinements is overall better than only applying one of them. Second, we show that the improvement provided by the those refinements is independently of the underlying SAT solver. Third, when comparing on a per instance basis, the combined refinement effects the runtime mostly positively. We show that the improvement is up to three orders of magnitude.

We compare our implementation against RAReQS [16], Qesto [18], DepQBF in version 5.0 [21], and GhostQ [20]. For every solver except GhostQ, we use Bloqqer [5] in version 031 as preprocessor. For our experiments, we used a machine with a 3.6 GHz quad-core Intel Xeon processor and 32 GB of memory. The timeout and memout were set to 10 min and 8 GB, respectively.

Table 1. Number of solved instances of the QBFGallery 2014 and QBFEval 2016 benchmark sets.

Family	Total	CAQE-cryptominisat				RAReQS	Qesto	DepQBF	GhostQ
		Plain	Strengthen	Expansion	Both				
Eval2012r2	276	128	129	146	**149**	134	132	139	145
Bomb	132	94	**95**	94	94	82	78	80	82
Complexity	104	60	68	86	85	**90**	76	51	43
Dungeon	107	60	65	**70**	**70**	61	57	67	50
Hardness	114	108	102	**109**	101	69	106	80	51
Planning	147	45	93	65	95	**144**	55	38	13
Testing	131	91	86	93	91	95	90	99	**113**
Preprocessing	242	86	93	105	**110**	107	104	108	60
Gallery2014	1253	672	731	768	**795**	782	698	662	557
Eval2016	825	607	611	635	636	**644**	623	598	595
All	2078	1279	1342	1403	**1431**	1426	1321	1260	1152

Table 1 shows number of solved instances on the QBFGallery 2014 benchmark set, broken down by benchmark family, as well as the more recent QBFEval 2016 benchmark set. For CAQE, we only report on the best performing version, that is the one using cryptominisat as a backend solver.

The table shows that the strengthen and expansion refinement individually improve over the plain version of CAQE in the number of solved instances. Further, the combination of both refinements is the overall best solver, followed by RAReQS.

In the following, we refer to the combination of strengthen and expansion refinement as extended refinements. We want to detail the improvements due to the extended refinements and show their independence of the backend solver. The plot in Fig. 5 depicts the effect of the extended refinements with respect to the solved instances. The improvements in the number of solved instances are independent from the choice of the underlying SAT solver and range between 100 to 150 more instances solved compared to the plain version of CAQE.

The scatter plot depicted in Fig. 6 compares the running times of plain CAQE to the one using extended refinements (both using cryptominisat) on a per instance basis. Marks below the diagonal means that the variant using extended refinements is faster. It is remarkable that the extended refinements have mostly positive effect on the solving times. Only a few instances saw a significant increase in solving time and even less timed out with extended refinements while being solved before. On the other hand, we see improvements in solving time that exceed three orders of magnitude. This is an empirical confirmation of our goal stated before that our implementation of expansion-refinement adds performance characteristic of expansion-based solvers while keeping the characteristics of the clausal-abstraction algorithm.

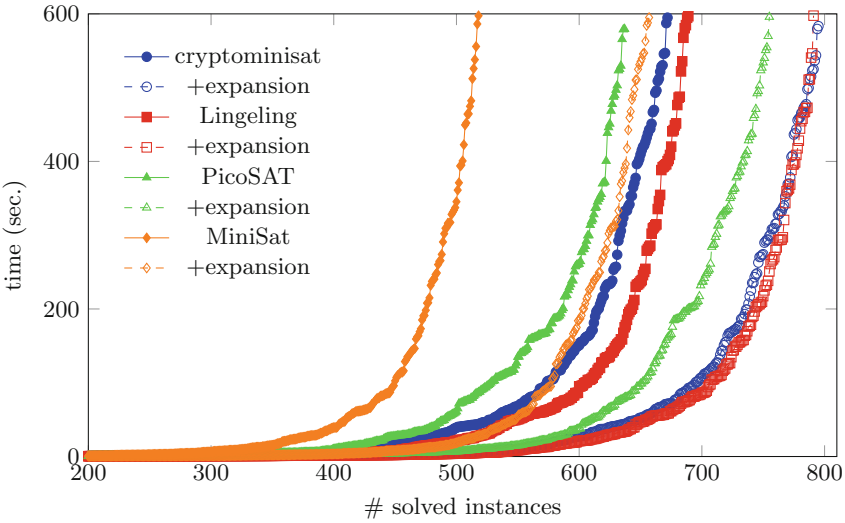


Fig. 5. Effect of the expansion refinement on the different configurations of CAQE on the GBFGallery 2014 benchmark sets.

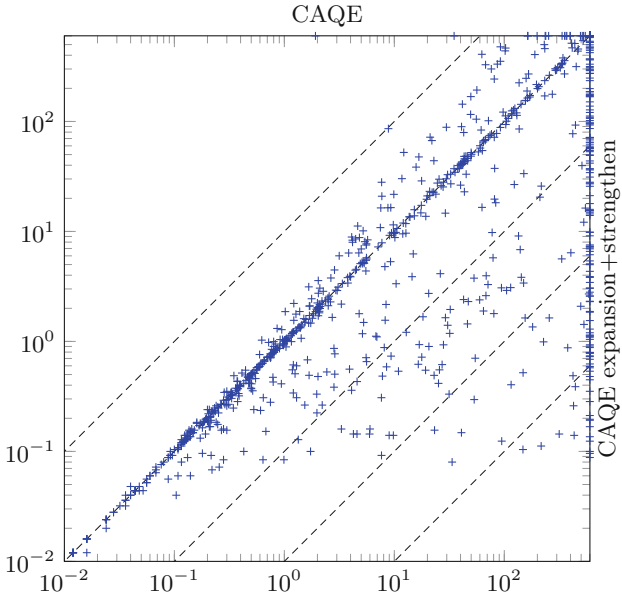


Fig. 6. Scatter plot comparing the solving time (in sec.) of CAQE with and without extended refinement.

5 Related Work

Q-resolution [19] is a variant of propositional refutation that is sound and refutation complete for QBF. There have been extensions proposed to Q-resolution, like long-distance resolution [27] and universal resolution [13], some which are implemented in the QCDCL solver DepQBF [21]. Recently, there has also been extensions proposed that extend Q-resolution by more generalized axioms [22]. In some sense, the ($\forall$exp-res) rule presented in this paper can be viewed as an new axiom rule for the $\forall$Red+Res calculus.

The $\forall$Exp+Res calculus [17] was introduced to allow reasoning over expansion-based QBF solving, exemplified by the QBF solver RAReQS [16]. The work on $\forall$Red+Res was motivated by the same desire, namely understanding the performance of the recently introduced QBF solvers CAQE [25] and Qesto [18]. The incomparability of $\forall$Exp+Res and $\forall$Red+Res [2,17] lead to the creation of stronger proof systems that unify those calculi, like IR-Calc [1]. Further separation results, between variants of IR-Calc and variants of Q-resolution, were given in [2]. Those extensions, however, do not have accompanying implementations. This also applies to recent work that is based on first-order resolution [9].

There are two well-known restrictions to Q-resolution, that is level-ordered and tree-like Q-resolution. Those restricted calculi were shown to be incomparable [23]. QCDCL based solver exhibit level-ordered proofs [15] and it was shown that $\forall$Exp+Res p-simulates tree-like Q-resolution [17]. We showed that $\forall$Red+Res is polynomial simulation equivalent to level-ordered Q-resolution, which explains similar performance characteristics of the underlying solvers. Further, the strengthening rule presented in this paper can be viewed as a first step towards breaking the level-ordered restriction. The $\forall$Red+$\forall$Exp+Res calculus p-simulates level-ordered and tree-like Q-resolution.

6 Conclusion

In this paper, we have introduced a new QBF proof calculus $\forall$Red+Res and showed that it is suitable for describing CEGAR based solving algorithms. We defined two extensions of the $\forall$Red+Res calculus and showed that there is a theoretical advantage over the basic calculus. Based on this foundation, we implemented an expansion refinement in the solver CAQE and evaluated it on standard QBF benchmark sets. Our experiments show that our new implementation significantly outperforms the previous one, with little to no negative impact, making it one of the most competitive QBF solver available. We have also shown that our theoretical considerations and the consequent algorithmic change explains those practical gains.

In future work, we want to improve the implementation by exploring heuristics for the application of the different refinements and we want to explore alternative versions of the strengthening rule presented in this paper.

Acknowledgments. I thank Christopher Hahn and the anonymous reviewers for their comments on earlier versions of this paper.

References

1. Beyersdorff, O., Chew, L., Janota, M.: On unification of QBF resolution-based calculi. In: Csuhaj-Varjú, E., Dietzfelbinger, M., Ésik, Z. (eds.) MFCS 2014. LNCS, vol. 8635, pp. 81–93. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44465-8_8
2. Beyersdorff, O., Chew, L., Janota, M.: Proof complexity of resolution-based QBF calculi. In: Proceedings of STACS. LIPIcs, vol. 30, pp. 76–89. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2015)
3. Biere, A.: PicoSAT essentials. JSAT **4**(2–4), 75–97 (2008)
4. Biere, A.: Lingeling essentials, a tutorial on design and implementation aspects of the the SAT solver lingeling. In: Proceedings of POS@SAT. EPiC Series in Computing, vol. 27, p. 88. EasyChair (2014)
5. Biere, A., Lonsing, F., Seidl, M.: Blocked clause elimination for QBF. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) CADE 2011. LNCS, vol. 6803, pp. 101–115. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22438-6_10
6. Bloem, R., Egly, U., Klampfl, P., Könighofer, R., Lonsing, F.: SAT-based methods for circuit synthesis. In: Proceedings of FMCAD, pp. 31–34. IEEE (2014)
7. Bloem, R., Könighofer, R., Seidl, M.: SAT-based synthesis methods for safety specs. In: McMillan, K.L., Rival, X. (eds.) VMCAI 2014. LNCS, vol. 8318, pp. 1–20. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54013-4_1
8. Eén, N., Sörensson, N.: An extensible SAT-solver. In: Giunchiglia, E., Tacchella, A. (eds.) SAT 2003. LNCS, vol. 2919, pp. 502–518. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24605-3_37
9. Egly, U.: On stronger calculi for QBFs. In: Creignou, N., Le Berre, D. (eds.) SAT 2016. LNCS, vol. 9710, pp. 419–434. Springer, Cham (2016). doi:10.1007/978-3-319-40970-2_26
10. Faymonville, P., Finkbeiner, B., Rabe, M.N., Tentrup, L.: Encodings of bounded synthesis. In: Legay, A., Margaria, T. (eds.) TACAS 2017. LNCS, vol. 10205, pp. 354–370. Springer, Heidelberg (2017). doi:10.1007/978-3-662-54577-5_20
11. Finkbeiner, B.: Bounded synthesis for petri games. In: Meyer, R., Platzer, A., Wehrheim, H. (eds.) Correct System Design. LNCS, vol. 9360, pp. 223–237. Springer, Cham (2015). doi:10.1007/978-3-319-23506-6_15
12. Finkbeiner, B., Tentrup, L.: Detecting unrealizability of distributed fault-tolerant systems. Logical Methods Comput. Sci. **11**(3) (2015)
13. Gelder, A.: Contributions to the theory of practical quantified boolean formula solving. In: Milano, M. (ed.) CP 2012. LNCS, pp. 647–663. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33558-7_47
14. Goerdt, A.: Davis-Putnam resolution versus unrestricted resolution. Ann. Math. Artif. Intell. **6**(1–3), 169–184 (1992)
15. Janota, M.: On Q-resolution and CDCL QBF solving. In: Creignou, N., Le Berre, D. (eds.) SAT 2016. LNCS, vol. 9710, pp. 402–418. Springer, Cham (2016). doi:10.1007/978-3-319-40970-2_25
16. Janota, M., Klieber, W., Marques-Silva, J., Clarke, E.M.: Solving QBF with counterexample guided refinement. Artif. Intell. **234**, 1–25 (2016)
17. Janota, M., Marques-Silva, J.: Expansion-based QBF solving versus Q-resolution. Theor. Comput. Sci. **577**, 25–42 (2015)
18. Janota, M., Marques-Silva, J.: Solving QBF by clause selection. In: Proceedings of IJCAI, pp. 325–331. AAAI Press (2015)
19. Büning, H.K., 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.: A non-prenex, non-clausal QBF solver with game-state learning. In: Strichman, O., Szeider, S. (eds.) SAT 2010. LNCS, vol. 6175, pp. 128–142. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14186-7_12
21. Lonsing, F., Biere, A.: DepQBF: a dependency-aware QBF solver. JSAT **7**(2–3), 71–76 (2010)
22. Lonsing, F., Egly, U., Seidl, M.: Q-resolution with generalized axioms. In: Creignou, N., Le Berre, D. (eds.) SAT 2016. LNCS, vol. 9710, pp. 435–452. Springer, Cham (2016). doi:10.1007/978-3-319-40970-2_27
23. Mahajan, M., Shukla, A.: Level-ordered Q-resolution and tree-like Q-resolution are incomparable. Inf. Process. Lett. **116**(3), 256–258 (2016)
24. Miller, C., Scholl, C., Becker, B.: Proving QBF-hardness in bounded model checking for incomplete designs. In: Proceedings of MTV, pp. 23–28. IEEE Computer Society (2013)
25. Rabe, M.N., Tentrup, L.: CAQE: a certifying QBF solver. In: Proceedings of FMCAD, pp. 136–143. IEEE (2015)
26. Soos, M., Nohl, K., Castelluccia, C.: Extending SAT solvers to cryptographic problems. In: Kullmann, O. (ed.) SAT 2009. LNCS, vol. 5584, pp. 244–257. Springer, Heidelberg (2009). doi:10.1007/978-3-642-02777-2_24
27. Zhang, L., Malik, S.: Conflict driven learning in a quantified Boolean satisfiability solver. In: Proceedings of ICCAD, pp. 442–449. ACM/IEEE Computer Society (2002)