Electronic Colloquium on Computational Complexity, Report No. 102 (2018)

Short Proofs in QBF Expansion

Olaf Beyersdorff
School of Computing, University of Leeds, United Kingdom
o.beyersdorff@leeds.ac.uk

Leroy Chew
School of Computing, University of Leeds, United Kingdom
scslnc@leeds.ac.uk

Judith Clymo
School of Computing, University of Leeds, United Kingdom
scjc@leeds.ac.uk

Meena Mahajan
The Institute of Mathematical Sciences, HBNI, Chennai, India
meena@imsc.res.in

Abstract

For quantified Boolean formulas (QBF) there are two main different approaches to solving: QCDCL and expansion solving. In this paper we compare the underlying proof systems and show that expansion systems admit strictly shorter proofs than CDCL systems for formulas of bounded quantifier complexity, thus pointing towards potential advantages of expansion solving techniques over QCDCL solving.

Our first result shows that *tree-like* expansion systems allow short proofs of QBFs that are a source of hardness for QCDCL, i.e. tree-like ∀Exp+Res is strictly stronger than tree-like Q-Resolution.

In our second result we efficiently transform *dag-like* Q-Resolution proofs of QBFs with bounded quantifier complexity into ∀Exp+Res proofs. This is theoretical confirmation of experimental findings by Lonsing and Egly, who observed that expansion QBF solvers often outperform QCDCL solvers on instances with few quantifier alternations.

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases QBF, Proof Complexity, Resolution, SC, Polynomial Hierarchy

Digital Object Identifier 10.4230/LIPIcs...

Acknowledgements Some of this work was done at Dagstuhl Seminar 18051, Proof Complexity. Research supported by the John Templeton Foundation (1st author) and EPSRC (2nd author).

1 Introduction

Quantified Boolean formulas (QBFs) generalise propositional logic by adding Boolean quantification. While not more expressive than propositional formulas, QBFs allow more succinct encodings of many practical problems, including automated planning [10], verification [2], and ontologies [16]. From a complexity point of view, they capture all problems from PSPACE.

Following the enormous success of SAT solving [22], there has been increased attention on QBF solving in the past two decades. Currently, many QBF solvers such as DepQBF [18], RAReQS [11], GhostQ [15], and CAQE [21], to name but a few, compete on thousands of QBF instances. However, lifting the success of SAT to QBF presents significant additional challenges stemming from quantification, and solvers use quite different techniques to do this.

ISSN 1433-8092

We consider two popular paradigms for QBF solving. In the first, QBF solvers use *Conflict Driven Clause Learning (CDCL)* techniques from SAT solving, together with a 'reduction' rule to deal with universally quantified literals. In the second method, QBF solvers use quantifier expansion and remove the universally quantified literals in order to use SAT-based reasoning on the formulas.

These two paradigms can be modelled by different QBF proof systems. Modern SAT solvers correspond to the Resolution proof system [8, 20]; similarly QBF solvers correspond to different QBF Resolution calculi. CDCL-style QBF (QCDCL) systems correspond to the QBF resolution system Q-resolution ($\mathsf{Q\text{-}Res}$) [14], although algorithms implementing QCDCL, such as DepQBF, typically also implement additional reasoning techniques. In contrast, expansion solving builds on expansion QBF proof systems, with $\forall\mathsf{Exp{+}Res}$ as their base system [12]. In fact, $\forall\mathsf{Exp{+}Res}$ was originally developed to model RAReQS [12].

The proof systems $\forall\mathsf{Exp{+}Res}$ and $\mathsf{Q\text{-}Res}$ are known to be incomparable, i.e. there are families of QBFs that have polynomial-size refutations in one system, but require exponential size refutations in the other [4, 12]. As such we would not expect either QCDCL or expansion-based algorithms to be consistently stronger than the other, but would instead anticipate that solvers implementing the two systems would complement each other.

In this paper we closely re-examine the relations between $\mathsf{Q\text{-}Res}$ and $\forall\mathsf{Exp{+}Res}$ when imposing two natural and practically important restrictions.

In the first restriction, we require proofs to be *tree-like*, i.e. derived clauses may not be reused, a model corresponding to the classic DPLL algorithm [8]. While it is known that tree-like $\forall\mathsf{Exp{+}Res}$ p-simulates tree-like $\mathsf{Q\text{-}Res}$ [5, 12], we show that this simulation is strict by providing an exponential separation. This separation uses the QParity formulas [4], which are known to be hard in (even dag-like) $\mathsf{Q\text{-}Res}$ [4]. Here we construct short tree-like proofs in $\forall\mathsf{Exp{+}Res}$, using a Prover-Delayer game that characterises tree-like Resolution [6].

We generalise this technique for Q-C formulas based on descriptions of circuits C. For suitably chosen circuits, such QBFs yield lower bounds for quite strong QBF calculi, even including QBF Frege systems [3]. In contrast, we show that under certain width conditions on the circuit they have short proofs in tree-like $\forall\mathsf{Exp{+}Res}$. In particular, for bounded-width circuits we obtain polynomial-size tree-like $\forall\mathsf{Exp{+}Res}$ proofs, and for circuits from the class SC we get quasi-polynomial-size proofs in tree-like $\forall\mathsf{Exp{+}Res}$.

In our second restriction we consider families of QBFs of *bounded quantifier complexity*, which express exactly all problems from the polynomial hierarchy and thus cover most application scenarios. In this case, we show that (dag-like) $\mathsf{Q\text{-}Res}$ is p-simulated by $\forall\mathsf{Exp{+}Res}$. The technically involved simulation increases in complexity as the number of quantification levels grows, and indeed there is an exponential separation between the two systems on a family of QBFs with an unbounded number of quantification levels [12]. For practitioners, our result offers a partial explanation for the observation that "the performance of solvers based on different solving paradigms substantially varies on classes of PCNFs defined by their numbers of alternations" [19].

2 Preliminaries

A literal is a propositional variable or its negation. The literals x, $\neg x$ are complementary to each other. A *clause* is a disjunction of literals. A *CNF* formula is a conjunction of clauses.

Quantified Boolean formulas (QBFs). Propositional logic extended with the quantifiers $\forall, \exists$ gives rise to quantified Boolean logic [13]. Semantically, $\forall x.\, \Psi$ is equivalent to $\Psi[0/x] \wedge \Psi[1/x]$ and $\exists x.\, \Psi$ is equivalent to $\Psi[0/x] \vee \Psi[1/x]$.

$$\frac{}{C}\ (\mathrm{Ax}) \qquad\qquad \frac{C \vee x \qquad D \vee \neg x}{C \vee D}\ (\mathrm{Res})$$

Ax: C is a non-tautological clause in the propositional matrix.
Res: variable x is existential, and if literal $z \in C$, then $\neg z \notin D$.

$$\frac{C \vee u}{C} \qquad \frac{C \vee \neg u}{C}\ (\forall\text{-Red})$$

u is universal and all other variables $x \in C$ are left of u in the prefix.

Figure 1 The rules of Q-Res [14]

In a closed QBF all variables are quantified, and we consider only closed QBFs in this paper. A closed QBF is either true or false, since if we semantically expand all the quantifiers we have a Boolean connective structure on $0, 1$. In a prenex QBF all quantification is done outside of the propositional connectives. A prenex QBF Φ thus has a propositional part ϕ called the matrix and a prefix of quantifiers Π and can be written as $\Phi = \Pi \cdot \phi$. When the propositional matrix of a prenex QBF is a CNF, then we have a PCNF.

Let $\mathcal{Q}_1 X_1 \ldots \mathcal{Q}_k X_k. \phi$ be a prenex QBF, where for $1 \leq i \leq k$, $\mathcal{Q}_i \in \{\exists, \forall\}$, $\mathcal{Q}_i \neq \mathcal{Q}_{i+1}$, X_i are pairwise disjoint sequences of variables. If $x \in X_i$, we say that x is at *level* i and write $\mathrm{lv}(x) = i$. We write $\mathrm{lv}(l)$ for $\mathrm{lv}(\mathrm{var}(l))$, where $\mathrm{var}(l)$ is the variable such that $l = \mathrm{var}(l)$ or $l = \neg\,\mathrm{var}(l)$. For fixed k, the class Σ_k^b contains all prenex QBFs of the above form with k quantifier blocks where the first block is existential.

Proof systems. Formally, a *proof system* [9] for a language L over alphabet Γ is a polynomial-time computable partial function $f : \Gamma^\star \rightharpoonup \Gamma^\star$ with $rng(f) = L$. For $x \in L$, a $y \in \Gamma^\star$ such that $f(y) = x$ is a proof of $x \in L$. If f and g are proof systems for the same language, then f *p-simulates* g if and only if there is a polynomially computable function mapping a proof in g to a proof of the same formula in f [9].

Q-Resolution. Introduced by Kleine Büning, Karpinski, and Flögel in [14], Q-Resolution (Q-Res) is a refutational system that operates on PCNFs (closed QBFs in prenex form where the matrix is a CNF). It uses the propositional resolution rule on existential variables with a side condition that prevents tautological clauses (cf. Figure 1). Tautologies are also forbidden from being introduced in the proof as axioms. In addition, Q-Res has a universal reduction rule ($\forall$-Red) to remove universal variables. A Q-Res proof of false QBF Φ is a sequence of clauses $C_1 \ldots C_m$ such that C_m is the empty clause (denoted $\bot$) and each C_i is either introduced by rule Ax, derived by rule Res from C_j and C_k on pivot x ($j, k < i$), or derived by $\forall$-Red from C_j ($j < i$). Clauses need not be unique in the proof. The parent(s) of a clause are the clause(s) used to derive it; the derived clause is called the child.

The size of a Q-Res proof π is the number of clauses it contains, denoted $|\pi|$. The size of refuting QBF Φ in Q-Res is minimum over the size of all possible Q-Res proofs.

A proof $C_1 \ldots C_m$ induces a directed acyclic graph (dag) with nodes corresponding to clauses, and edges to derivation steps. Clauses introduced by the rule Ax are source nodes and edges point from parent(s) of a proof step to the clause derived. If the induced dag is a tree then the proof is called tree-like. In tree-like Q-Res, only tree-like proofs are allowed.

QBF expansion. In addition to Q-Res we consider an alternative way of extending the resolution calculus based on *instantiation* of universal variables that was introduced to model *expansion-based QBF solving*. We operate on clauses that comprise only existential variables from the original QBF, which are additionally *annotated* by a substitution to universal variables, e.g. $\neg x^{0/u,1/v}$. For any annotated literal l^σ, the substitution σ must not make assignments to variables right of l, i.e. if $u \in \mathsf{dom}(\sigma)$, then u is universal and $\mathrm{lv}(u) < \mathrm{lv}(l)$. To preserve this invariant, we use the *auxiliary notation* $l^{[\sigma]}$, which for an existential literal l and an assignment σ to the universal variables filters out all assignments that are not permitted, i.e. $l^{[\sigma]} = l^{\{c/u \in \sigma \mid \mathrm{lv}(u) < \mathrm{lv}(l)\}}$. We say that an assignment is complete if its domain is all universal variables. Likewise, we say that a literal x^τ is fully annotated if all universal variables u with $\mathrm{lv}(u) < \mathrm{lv}(x)$ in the QBF are in $\mathsf{dom}(\tau)$, and a clause is fully annotated if all its literals are fully annotated.

The calculus $\forall$Exp+Res from [12] works with fully annotated clauses on which resolution is performed. For each clause C from the matrix and an assignment τ to all universal variables, falsifying all universal literals in C, $\forall$Exp+Res can use the axiom $\left\{ l^{[\tau]} \mid l \in C, l \text{ existential} \right\}$.

As its only rule it uses the resolution rule on annotated variables.

$$\frac{C \vee x^\tau \qquad D \vee \neg x^\tau}{C \cup D} \text{ (Res)}.$$

An $\forall$Exp+Res proof is a sequence of clauses where every clause is either a valid expansion of a clause in the input formula, or is derived by resolution from previous clauses. The size of a proof is the number of clauses it contains, and the size of refuting a formula is the smallest valid proof. A proof induces a dag, and we may restrict our attention to tree-like proofs.

Quantification Blocks. In $\forall$Exp+Res, annotations appear only on existential literals to the right of the annotation variables. This means that if the innermost quantifier block is universal, any literals of that level play no role in the proof, as these variables cannot appear in any annotation. Similarly, in Q-Res, any universal literals from the innermost quantifier block can be removed by $\forall$-Red immediately, without any significant cost to the proof size. For these reasons, we assume that the innermost block is existential.

The Prover-Delayer game. The Prover-Delayer game from [6] gives a characterisation of the size of tree-like resolution proofs of unsatisfiable formulas. The game has two players, who construct a partial assignment. The game terminates when the (partial) assignment constructed falsifies some clause. The Delayer tries to score as many points as possible before the inevitable termination, while the Prover tries to limit the Delayer's score. Starting with the empty assignment, the game proceeds in rounds:

- Prover chooses an unassigned variable x.
- Delayer assigns weights $0 \leq w_0, w_1 \leq 1$, such that $w_0 + w_1 = 1$.
- Prover now chooses to set the variable x either to 0 or to 1.
- If the variable is set to $i \in \{0, 1\}$ then Delayer scores $\log(\frac{1}{w_i})$ points.

Note that if the Delayer chooses $w_b = 1$ for some b, then the Prover can respond in this round by setting x to b, and the Delayer scores nothing in this round. For such a Delayer response ($w_0 = 1$ or $w_1 = 1$), we say that the Delayer chooses the value of the variable. Otherwise the actual choice is made by the Prover.

▶ **Theorem 1** (Beyersdorff, Galesi and Lauria; [6]). *Let ϕ be an unsatisfiable false CNF with shortest tree-like Resolution refutation of size S. Then, there is a Delayer strategy such that the Delayer scores at least $\log(\lceil \frac{S}{2} \rceil)$ points in any game on ϕ against any Prover. Furthermore, no Delayer strategy can guarantee more; there is a Prover strategy that limits the Delayer score to $\log(\lceil \frac{S}{2} \rceil)$ points.*

3 QBFs based on Parity

The false QBFs QParity, introduced in [4], express the contradiction that for some binary string x, for every z, the parity of the number of 1s in x is different from z. The propositional matrix is falsified whenever $z = \textsc{Parity}(x_1, \dots, x_n)$. The $\mathrm{Q}\textsc{Parity}_n$ QBFs are given as

$$\exists x_1 \dots \exists x_n \forall z \exists t_0 \dots \exists t_n (\neg t_0) \wedge (z \vee t_n) \wedge (\neg z \vee \neg t_n) \wedge \bigwedge_{i=1}^{n} (t_i = t_{i-1} \oplus x_i),$$

where the formulas $(t_i \equiv t_{i-1} \oplus x_i)$ are expressed by the four clauses $(\neg t_{i-1} \vee x_i \vee t_i)$, $(t_{i-1} \vee \neg x_i \vee t_i)$, $(t_{i-1} \vee x_i \vee \neg t_i)$, $(\neg t_{i-1} \vee \neg x_i \vee \neg t_i)$.

These formulas are known to be hard for both tree-like and dag-like Q-Res [4]. Here we show that they have short tree-like proofs in $\forall$Exp+Res.

▶ **Theorem 2.** $\mathrm{Q}\textsc{Parity}_n$ *have polynomial-size tree-like $\forall$Exp+Res proofs.*

Proof. In order to analyse $\forall$Exp+Res proofs, it suffices to simply expand all clauses in every universal assignment and look at all propositional resolution refutations from tree-like resolution. We expand out all the clauses of $\mathrm{Q}\textsc{Parity}_n$ based on the two settings to the single universal variable z. The formula now contains twice as many clauses. We now need to construct a tree-like resolution proof from these clauses. We use the asymmetric Prover-Delayer game. The Prover strategy is given in Algorithm 1.

Algorithm 1 Prover Strategy

```
$i = 0$, $j = n$.
The Prover queries the variables of the unit clauses $\neg t_0^{0/z}, \neg t_0^{1/z}, t_n^{0/z}, \neg t_n^{1/z}$. The Delayer
is forced to satisfy these clauses or get a constant score.
while $j - i > 1$ do
    $k \leftarrow \lfloor \frac{i+j}{2} \rfloor$.
    The Prover queries the variable $t_k^{1/z}$.
    The Prover chooses the value that gives the Delayer the least score.
    The Prover queries the variable $t_k^{0/z}$.
    The Prover chooses the value that gives the Delayer the least score.
    if $t_k^{0/z} = t_k^{1/z}$ then $i \leftarrow k$ else $j \leftarrow k$
The Prover now queries $x_j$, and chooses the value that gives the Delayer the least score.
▷ The game ends here because $t_i^{0/z} = t_i^{1/z}$, $t_j^{0/z} \neq t_j^{1/z}$, and for $c \in \{0,1\}$, there are clauses
expressing $t_j^{c/z} \equiv x_j \oplus t_i^{c/z}$.
```

The Delayer can score at most 2 points per loop iteration, hence a total score of at most $2\lceil \log(n) \rceil$. By Theorem 1, there are tree-like resolution refutations with $O(n^2)$ leaves.

One nice feature of the Prover-Delayer technique is that we can construct a tree-like Resolution proof from a Prover strategy. To simplify, let us suppose $n = 2^r$.

We proceed in rounds. The idea is that inductively, in the kth round, for each $0 \leq i < 2^{r-k}$, we produce 2^{r-k} copies of the clauses

$$t_{2^k i}^{0/z} \vee t_{2^k i}^{1/z} \vee \neg t_{2^k(i+1)}^{0/z} \vee t_{2^k(i+1)}^{1/z}, \qquad \neg t_{2^k i}^{0/z} \vee \neg t_{2^k i}^{1/z} \vee \neg t_{2^k(i+1)}^{0/z} \vee t_{2^k(i+1)}^{1/z},$$

$$t_{2^k i}^{0/z} \vee t_{2^k i}^{1/z} \vee t_{2^k(i+1)}^{0/z} \vee \neg t_{2^k(i+1)}^{1/z}, \qquad \neg t_{2^k i}^{0/z} \vee \neg t_{2^k i}^{1/z} \vee t_{2^k(i+1)}^{0/z} \vee \neg t_{2^k(i+1)}^{1/z}.$$

For the base case $k = 0$, these clauses are produced using the axioms. For instance, the clause $t_{i-1}^{0/z} \vee t_{i-1}^{1/z} \vee \neg t_i^{0/z} \vee t_i^{1/z}$ is produced by resolving $t_{i-1}^{0/z} \vee x_i \vee \neg t_i^{0/z}$ and $t_{i-1}^{1/z} \vee \neg x_i \vee t_i^{1/z}$.

Once we reach round r, we have the clause $t_0^{0/z} \vee t_0^{1/z} \vee \neg t_{2^r}^{0/z} \vee t_{2^r}^{1/z}$ which we resolve with $\neg t_0^{0/z}, \neg t_0^{1/z}, t_{2^r}^{0/z}$ and $\neg t_{2^r}^{1/z}$ to get a contradiction. ◀

Consequently, tree-like ∀Exp+Res is strictly stronger than tree-like Q-Res.

▶ **Corollary 3.** *Tree-like ∀Exp+Res p-simulates tree-like Q-Res, but tree-like Q-Res does not simulate ∀Exp+Res, and there are QBFs providing an exponential separation.*

Proof. From [12] we know that tree-like ∀Exp+Res p-simulates tree-like Q-Res. The QBFs QParity show that this is strict; QParity requires exponential size proofs for tree-like Q-Res (shown in [4]), and by Theorem 2, has short proofs in tree-like ∀Exp+Res. ◀

4 Short tree-like expansion proofs for QBFs based on thin circuits

We extend the method demonstrated above for other QBFs based on Boolean circuits with bounded fan-in. We assume without loss of generality that each gate has a fan-in of two. (The fan-in of a gate is the number of incoming wires.)

▶ **Definition 4.** Let C be a Boolean circuit over variables $x_1, \dots, x_n$, with gates computing binary Boolean functions. Let the gates of C be $g_1, \dots, g_m$ in topological order, with g_m being the output gate. The QBF Q-C expresses the contradiction $\exists x_1 \dots x_n \forall z\ [z \neq C(x_1, \dots, x_n)]$ and is constructed as follows.

The variable set $X = \{x_1, \dots, x_n\}$ contains all the input variables of C. The variable set T has one variable t_j corresponding to each gate g_j of C; $T = \{t_1, \dots, t_m\}$. The quantifier prefix is $\exists X \forall z \exists T$. The QBF is of the form

$$(z \vee t_m) \wedge (\neg z \vee \neg t_m) \wedge \bigwedge_{j=1}^{S} [t_j \text{ is consistent with the inputs to gate } g_j].$$

The predicate $[t_j$ is consistent$]$ depends on t_j and at most two other variables in $X \cup T$, depending on the connections in C. Hence it can be written as a short CNF formula; we include these clauses. (E.g. if $g_2 = g_1 \wedge x_1$ then we add clauses $(t_2 \vee \neg t_1 \vee \neg x_1), (\neg t_2 \vee t_1), (\neg t_2 \vee x_1)$.)

Note that the QBF Q-C is of size $O(m)$, where m is the number of gates in the circuit C. In particular, if C is of size poly(n), then so is Q-C.

The following result appears in [4]. We include a brief proof sketch to provide context for the subsequent discussion.

▶ **Proposition 5** (Proposition 28 in [4])**.** *For every family of polynomial-size circuits $\{C_n\}$, the QBF family $\{Q\text{-}C_n\}$ has poly(n)-size proofs in dag-like ∀Exp+Res.*

Proof (sketch). Proceeding by induction on $i \in [m]$, we prove the statement $t_i^{0/z} = t_i^{1/z}$ in clausal form.

Since the input X variables have no annotation, we can show the base case for the first T variable t_1, $t_1^{0/z} = t_1^{1/z}$, by resolving on X variables.

For the inductive step, we have already derived clauses expressing that $t_j^{0/z} = t_j^{1/z}$ for $j < i$. Since the inputs to gate g_i come from $\{g_j \mid j < i\} \cup X$, and since we have axiom clauses expressing the consistency predicates for $t_i^{0/z}$ and $t_i^{1/z}$, we can derive $t_i^{0/z} = t_i^{1/z}$ in a short proof fragment. (There are at most 6 variables involved in this fragment.)

Finally, having derived $t_m^{0/z} = t_m^{1/z}$, we simply perform resolution with the unit clauses $t_m^{0/z}$ and $\neg t_m^{1/z}$ to obtain the empty clause. ◀

For tree-like ∀Exp+Res, we would like something similar. However the above proof sketch reuses the derivations of $t_j^{0/z} = t_j^{1/z}$ at all stages i where g_j is an input to g_i. For tree-like

$\forall$Exp+Res, we cannot reuse them. Instead we generalise the binary search Prover strategy idea from Theorem 2. Note that this technique works precisely because the circuits underlying QParity formulas have very small "width". This could work in any similarly structured circuit C for Q-C. As long as C has polynomial length $p(n)$ and constant "width" bounded by b, the Prover can use binary search to devise a strategy where the Delayer scores at most $c\log(p(n))$ (for constant c); hence the proof size is at most $p(n)^c$ which is polynomial.

If we relax our desire for polynomial-size proofs to just quasi-polynomial size we can allow circuits with non-constant width. If we have the restriction of poly-logarithmic width, then we get quasi-polynomial size proofs.

To make this intuition formal, we recall the notion of width in circuits.

▶ **Definition 6** (Layered Circuits, and Circuit Width). A circuit is said to be *layered* if its gates can be partitioned into disjoint sets S_i for $1 \leq i \leq \ell$, such that for each i, for each gate in layer S_i, all its inputs are either input variables or the outputs of gates in S_{i-1}. The output gate is in the final layer S_ℓ. The *width* of a layered circuit is the maximum number of gates in any layer; $\mathrm{width}(C) = \max\{|S_i| \mid i \in \ell]\}$.

▶ **Theorem 7.** *Let C be a layered circuit of size m and width w, and let Q-C be the corresponding QBF. Then Q-C has a proof, in tree-like $\forall$Exp+Res, of size $m^{O(w)}$.*

Proof Sketch. Consider the expanded CNF obtained from Q-C. Let X be the set of x variables, U be the set of $t^{0/z}$ variables and V be the set of $t^{1/z}$ variables. Let the clauses expressing that the t variables correspond to the computation of C on x be denoted $F(X,T)$. A proof in (tree-like) $\forall$Exp+Res that Q-C is false is a proof in (tree-like) Resolution that the CNF $G(X,U,V)$, defined below, is unsatisfiable.

$$G(X,U,V) = F(X,U) \wedge F(X,V) \wedge t_m^{0/z} \wedge \neg t_m^{1/z}$$

Let the number of layers be ℓ; note that $\ell \leq m$. We will describe a Prover strategy that limits the Delayer's score to $O(w\log\ell)$ and then invoke Theorem 1.

Let W_ℓ be the variables corresponding to the output gate; $W_\ell = \{t_m^{0/z}, t_m^{1/z}\}$. For $i \in [\ell]$, let $W_{i-1} \subseteq X \cup U \cup V$ be the variables feeding into gates at layer i of C.

The Prover uses binary search to identify a layer j where all corresponding variables of W_{j-1} from U and V are in agreement, whereas some corresponding variables of W_j from U and V disagree. The strategy is given in Algorithm 2.

There are variables $t_b^{0/z} \neq t_b^{1/z}$ in W_j, but for all t_a variables in W_{j-1}, $t_a^{0/z} = t_a^{1/z}$. Furthermore all X variables in W_{j-1} are set. So $t_b^{0/z} \neq t_b^{1/z}$ must cause a contradiction with the copies of the clauses expressing consistency of gate g_b with its inputs.

For every layer i where the Prover queries variables from W_i, there are at most $4w$ variables to query, and this is the maximum score for the Delayer on W_i. The Prover looks at no more than $\lceil\log\ell\rceil$ sets W_i. Hence the score for the Delayer is at most $4w\log\ell + 1$. ◀

▶ **Corollary 8.** *Suppose $\{C_n\}$ is a family of layered circuits with width bounded by a constant. Then the family of QBFs Q-C_n has polynomial-size proofs in tree-like $\forall$Exp+Res.*

A language $L \subseteq \{0,1\}^*$ is in P/poly if there is a family of circuits $\{C_n\}_{n\geq 1}$ where the size of each C_n is $n^{O(1)}$, such that C_n computes the characteristic function of $L \cap \{0,1\}^n$. If, furthermore, the depth of C_n is $(\log n)^{O(1)}$, then the language is in NC. If, instead, the width of C_n is $(\log n)^{O(1)}$, then the language is in SC. Note that P/poly and SC so defined are the non-uniform analogues of the complexity classes P and TIME, SPACE$(n^{O(1)}, (\log n)^{O(1)})$.

Algorithm 2 Prover Strategy in Theorem 7

```
$i = 0$, $j = \ell$.
The Prover queries the variables $t_m^{0/z}$ and $t_m^{1/z}$ of $W_l$. The Delayer is forced to satisfy the
unit clauses or get a constant score.
The Prover queries the variables of $W_0$, and chooses the values that give the Delayer the
least score.
while $j - i > 1$ do
    $k \leftarrow \lfloor (i+j)/2 \rfloor$.
    while some $t$ variables in $W_k$ are unassigned do
        The Prover queries an unassigned $t_a^{0/z}$ in $W_k$.
        The Prover chooses the value that gives the Delayer the least score.
        The Prover queries the variable $t_a^{1/z}$.
        The Prover chooses the value that gives the Delayer the least score.
        Prover queries all remaining $X$ variables in $W_k$.
        Prover chooses the values that give the Delayer the least score.
    if $t_b^{0/z} \neq t_b^{1/z}$ for some pair of variables in $W_k$ then  $j \leftarrow k$ else  $i \leftarrow k$.
            ▷ Once $j - i = 1$, the game ends in a contradiction witnessed by the variables
corresponding to some gate at layer $j$.
```

▶ **Corollary 9.** *Suppose $\{C_n\}$ is an* SC *family of circuits. Then the family of QBFs Q-C_n has quasi-polynomial size proofs in tree-like* $\forall$Exp+Res*.*

In essence, we show that these Q-C formulas, which can give lower bounds in QCDCL style systems [3], are easy even for tree-like $\forall$Exp+Res. Notice that the false Q-C formulas are all Σ_3^b; this is the minimum quantifier complexity needed to separate Q-Res and $\forall$Exp+Res.

5 $\forall$Exp+Res simulates dag-like Q-Res for bounded alternations

We now move from the tree-like systems to the stronger dag-like model. While it is known that $\forall$Exp+Res and Q-Res are in general incomparable [4], we will show here a simulation of dag-like Q-Res by $\forall$Exp+Res for QBFs of *bounded* quantifier complexity.

The natural way to transform a clause in a Q-Res refutation of Φ into a clause in a $\forall$Exp+Res refutation is similar to how axioms are instantiated in $\forall$Exp+Res. We define some complete assignment α to the universal variables of Φ that does not satisfy the clause. All universal literals are removed (since they are falsified under α), and each existential literal x is replaced by $x^{[\alpha]}$ (recall that $[\alpha]$ indicates the restriction of α to those variables that appear before the annotated literal in the quantifier prefix). The difficulty is to ensure that the resolution steps in the attempted $\forall$Exp+Res proof are valid. In particular, in every resolution step the pivot literals must have the same annotation. It may be impossible to find suitable annotations for each clause in the given Q-Res proof that respect this restriction.

Consider the simple example of Figure 2. The clause $(y \vee x)$ is resolved once with a clause containing universal literal u, and separately with another clause containing $\neg u$. It is not possible to define a single annotation for x in $(y \vee x)$ so that both these steps are valid in $\forall$Exp+Res. Instead, the clause could be duplicated to allow for both annotations. If the clause $(y \vee x)$ had occurred part way through a Q-Res proof then its entire derivation would also need to be copied – once with $0/u$ in the annotation, and again with $1/u$. Of course, such duplication of clauses may result in an exponential increase in the proof size if there are many different possible annotations to consider. Notice that the duplication of

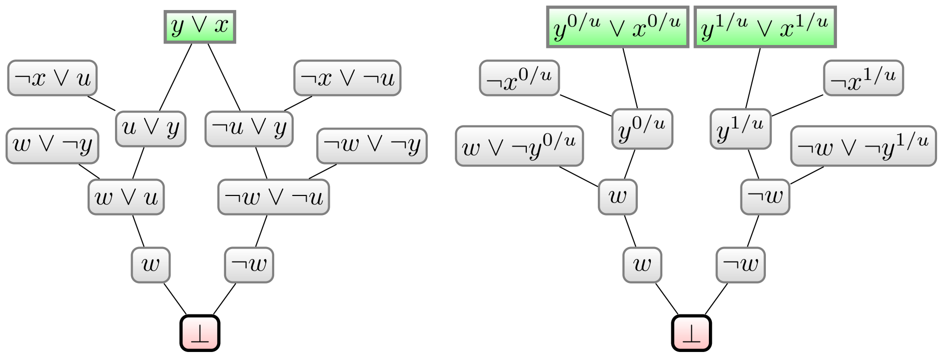

Figure 2 Duplicating clauses to create an expansion refutation of QBF with prefix $\exists w \forall u \exists xy$

clauses is only necessary in dag-like proofs, and this intuitively demonstrates why we have a simulation in the tree-like systems. We show that if attention is restricted to QBFs that have $O(1)$ quantifier blocks, then we can support all the required annotations without an exponential increase in the proof size.

Recall that a Q-Res proof is a sequence of clauses, not necessarily unique, and induces a dag by the inference relationship. Our aim is to construct from a Q-Res proof of PCNF Φ a sequence of new Q-Res proofs of Φ, the last of which can be readily turned into a valid $\forall$Exp+Res proof.

5.1 Expanding a Q-Resolution proof

We start with a few useful definitions and observations regarding Q-Res proofs.

In a Q-Res refutation there is no benefit to delaying a $\forall$-Red step since it cannot cause a later step to be blocked, and the result of $\forall$-Red is stronger than its parent, so it is safe to assume that $\forall$-Red is carried out as soon as possible. If a clause is used in a $\forall$-Red step then it is not used in any other proof step, since this would delay a possible $\forall$-Red on that path, so all possible $\forall$-Red steps for a clause are carried out consecutively, without branching.

Where the proof contains consecutive $\forall$-Red steps, they may be re-ordered so that innermost literals are removed first. This allows consecutive $\forall$-Red which remove literals from the same level in the quantifier prefix to be treated as a single step in the construction below. Literals removed in consecutive $\forall$-Red steps cannot be complementary, since they must all appear together in the clause prior to the sequence.

For the remainder of this subsection let $\pi = C_1 \dots C_m$ be a Q-Res proof of PCNF $\Phi = \mathcal{Q}_1 X_1 \dots \mathcal{Q}_k X_k . \phi$ and let $i \in [k]$ with $\mathcal{Q}_i = \forall$.

▶ **Definition 10.** A $\forall$-Red step in π is *at level i* if the universal literal(s) removed by the step belong to X_i.

▶ **Definition 11.** Let A and B be two clauses in π. Then A is *i-connected* to B if there is a subsequence $C_{a_1} \dots C_{a_n}$ of π such that $C_{a_1} = A$, $C_{a_n} = B$, $\forall l \in \{1 \dots n-1\}$ C_{a_l} is a parent of $C_{a_{l+1}}$ in π, and no member of the sequence is derived by $\forall$-Red at any level $j \leq i$ in π.

▶ **Definition 12.** The *level i derivation* of a clause C in π, denoted $\pi(C, i)$, is the subsequence of π ending at C and containing exactly those clauses which are i-connected to C.

▶ **Definition 13.** $\mathcal{A}^i_\pi$ is the set of clauses that are parents of a $\forall$-Red step at level i in π.

▶ **Lemma 14.** *Let C be a clause in π. If C does not belong to the level i derivation of any $P \in \mathcal{A}^i_\pi$ then C contains no universal literals at level i.*

Proof. Suppose $u \in X_i \cap C$. Consider some path p between C and the root of π. Since u does not appear in the empty clause, this path must include a $\forall$-Red step with parent P at which u is removed, i.e. there is a sub-path p' between C and P along which every clause also contains u. Since $\forall$-Red is carried out as soon as possible and in reverse level order, it follows that C is i-connected to P and so $C \in \pi(P, i)$ with $P \in \mathcal{A}^i_\pi$. ◀

The level i derivation of a clause C is the section of π that could possibly contribute universal literals from X_i to that clause. Every universal literal from X_i that appears in the level i derivation of C must also appear in C, since there are no $\forall$-Red steps at level i to remove them, so any two clauses that both appear in the same level i derivation cannot contain complementary universal literals at level i. The main idea is to use the level i derivation of $\forall$-Red steps at level i to find and isolate sections of a proof that contain no complementary universal literals from X_i and which therefore could be given the same (level i) annotation in a $\forall$Exp+Res proof.

We now start to define the crucial elements of the simulation construction.

▶ **Definition 15** (Level i expansion of π)**.** For π a Q-Res refutation of PCNF where $\Phi = \mathcal{Q}_1 X_1 \dots \mathcal{Q}_k X_k. \phi$ with $\mathcal{Q}_i = \forall$, the level i expansion of π is defined by the following construction.

Let $P \in \mathcal{A}^i_\pi$ and C the unique child of P. We have assumed that consecutive $\forall$-Red steps are carried out in reverse level order, and with steps at the same level collapsed together so P was not derived by $\forall$-Red at level i or at any level $j < i$ in π.

Find $\pi(P, i)$, the level i derivation of P, and copy this section of the proof. The original is not discarded until later. Each identified clause $D \in \pi(P, i)$ generates a new (identical) clause D'. For $C_a \in \pi(P, i)$, and C_b a parent of C_a in π, the parent of C'_a is C'_b if it exists, and otherwise C_b. Crucially, update C to be derived from the copy P' of its parent P.

Repeat this process for each member of $\mathcal{A}^i_\pi$. Then the clauses are ordered so that clause C_a comes before any copies of C_a, and if $a > b$ then every copy of C_b comes before C_a or any of its copies. This ensures that the parent(s) of a proof step always appear before the clause they derive. Among copies of the same clause, assume an ordering based on the order in which $\forall$-Red steps appear in π.

Clauses from the original refutation which no longer have any children (i.e. if copies of that clause are used for every derivation it was involved in) are removed. The result is the level i expansion of π, written $e_i(\pi)$.

▶ **Lemma 16.** *Let π be a valid Q-Res refutation of Φ. For every universal level i in Φ, $e_i(\pi)$ is a valid Q-Res refutation of Φ.*

Proof. Every derivation is an exact copy of a derivation in π, and the imposed ordering respects the order of derivations. ◀

▶ **Lemma 17.** *Let π be a Q-Res proof of Φ and i a universal level in Φ, then $e_i(\pi)$ and π have the same number of $\forall$-Red steps at levels $j \leq i$, for j a universal level.*

Proof. A clause which is the result of a $\forall$-Red step at level $j \leq i$ does not belong to any level i derivation of a member of $\mathcal{A}^i_{e_i(\pi)}$, by definition, so will never be copied. ◀

▶ **Lemma 18.** *Let π be a Q-Res proof of Φ and i a universal level in Φ. The level i derivations of clauses in $\mathcal{A}^i_{e_i(\pi)}$ are disjoint.*

Proof. The clauses copied for $P \in \mathcal{A}^i_\pi$ are exactly the level i derivation of $P' \in \mathcal{A}^i_{e_i(\pi)}$. ◀

▶ **Lemma 19.** *Let π be a Q-Res proof of Φ and i a universal level in Φ. The parent and child of any proof step in $e_i(\pi)$ cannot belong to level i derivations of different members of $\mathcal{A}^i_{e_i(\pi)}$. Similarly, the two parents of a resolution step in $e_i(\pi)$ cannot belong to level i derivations of different members of $\mathcal{A}^i_{e_i(\pi)}$*

Proof. Let B be a parent of A in $e_i(\pi)$. By construction, if A is in the level i derivation of $P \in \mathcal{A}^i_{e_i(\pi)}$ and B is not then B is the result of a $\forall$-Red step at some level $j \leq i$ and so cannot be included in the level i derivation of any clause. If A is derived by resolution and does not belong to any level i derivation of $P \in \mathcal{A}^i_{e_i(\pi)}$ then neither can its parents. These observations combined with Lemma 18 give the result. ◀

▶ **Lemma 20.** *Let π be a Q-Res proof of Φ and i a universal level in Φ. The size of the level i expansion of Q-Res refutation π is at most $|\pi|^2$.*

Proof. If there are S distinct $\forall$-Red steps at level i, then each clause in π may be copied up to S times, so the size of the level i expansion of π is at most $|\pi| \cdot (S+1)$. Clearly $S < |\pi|$. ◀

Since the level i expansion of a Q-Res refutation is itself a Q-Res refutation, we can apply the process iteratively for different values of i. We will expand the proof for each universal level, starting from the innermost.

▶ **Definition 21** (Complete expansion of π)**.** Let π be a Q-Res proof of PCNF Φ. The complete expansion of π, denoted $E(\pi)$ is $E(\pi) = e_1(e_3 \ldots (e_{k-1}(\pi)))$ if $\mathcal{Q}_1 = \forall$ in Φ, and $E(\pi) = e_2(e_4 \ldots (e_{k-1}(\pi)))$ if $\mathcal{Q}_1 = \exists$. Intermediate stages are labelled π_i (where $\mathcal{Q}_i = \forall$), so that $\pi_i = e_i(\pi_{i+2}) = e_i(e_{i+2} \ldots (e_{k-1}(\pi)))$.

▶ **Lemma 22.** *Let π be a Q-Res proof of PCNF Φ. Then $E(\pi)$ is a Q-Res refutation of Φ.*

Proof. This follows from repeated application of Lemma 16. ◀

▶ **Lemma 23.** *Let π be a Q-Res proof of PCNF Φ and $\mathcal{Q}_i = \mathcal{Q}_{i+2} = \forall$ in Φ. Then the number of $\forall$-Red steps at level i in π_{i+2} equals the number of $\forall$-Red steps at level i in π.*

Proof. By repeated application of Lemma 17. ◀

▶ **Lemma 24.** *Given Q-Res proof π of PCNF $\Phi = \mathcal{Q}_1 X_1 \ldots \mathcal{Q}_k X_k.\, \phi$, $|E(\pi)| \leq |\pi|^{1+k/2}$*

Proof. The argument proceeds by a simple induction on the number of universal levels that have been expanded, showing that for every level i with $\mathcal{Q}_i = \forall$, $\pi_i \leq |\pi|^{1+(k+1-i)/2}$.

Let S be the maximum number of $\forall$-Red steps on any single level in π. Then, following Lemma 20, $|\pi_{k-1}| \leq |\pi| \cdot (S+1) \leq |\pi|^2 \leq |\pi|^{1+(k+1-(k-1))/2}$. Assume the hypothesis for $k-1, \ldots, i+2$. Since π_{i+2} has the same number of level i $\forall$-Red steps as π (Lemma 23), the method of Lemma 20 applied to π_{i+2} gives $|\pi_i| \leq |\pi_{i+2}| \cdot (S+1) \leq |\pi|^{1+(k+1-(i+2))/2} \cdot |\pi| \leq |\pi|^{1+(k+1-i)/2}$.

Since $E(\pi)$ is either π_1 (if $\mathcal{Q}_1 = \forall$) or π_2 (if $\mathcal{Q}_1 = \exists$), $|E(\pi)| \leq \max\{|\pi|^{1+k/2}, |\pi|^{1+(k-1)/2}\}$. ◀

Our claim is that the refutation $E(\pi)$ can easily be translated into a valid $\forall$Exp+Res refutation of Φ. We introduce a system of labelling clauses in the proofs π_i with partial assignments to the universal variables in the formula being refuted. Eventually, every clause in $E(\pi)$ will be associated with a complete assignment to universal variables which is then used to define annotations for the existential literals in that clause.

5.2 Annotating the expanded proof

▶ **Definition 25.** Let π be a Q-Res of PCNF $\Phi = \mathcal{Q}_1 X_1 \dots \mathcal{Q}_k X_k.\, \phi$, $\mathcal{Q}_i = \forall$. For a clause $D \in \mathcal{A}^i_{\pi_i}$ let α^D_i be the assignment to X_i that sets variables appearing in D so that D is not satisfied, and all other variables in X_i to 0.

▶ **Lemma 26.** *Let $D \in \mathcal{A}^i_{\pi_i}$. Then α^D_i does not satisfy any $C \in \pi_i(D, i)$.*

Proof. $C \in \pi_i(D, i)$, so by construction there is a path between C and D that does not include any ∀-Red step at level i. Therefore any level i literal $u \in C$ also appears in D, and any assignment to variables at level i that does not satisfy D also will not satisfy C. ◀

Immediately after generating π_i from π_{i+2} add the following labels: For each $D \in \mathcal{A}^i_{\pi_i}$, label all clauses in $\pi_i(D, i)$ with α^D_i. Any clause in π_i that is not in a level i derivation of some $D \in \mathcal{A}^i_{\pi_i}$ does not contain any literal at level i by Lemma 14, so is not satisfied by any assignment to level i variables. Label such clauses with the assignment setting all level i variables to 0. In subsequent expansions, clauses are copied with their labels. This means that all clauses in π_i will be labelled with complete assignments to all levels $\geq i$, and that $E(\pi)$ will have all clauses labelled with a complete assignment to all universal variables in Φ. No clause is labelled twice (Lemma 18).

▶ **Lemma 27.** *In π_i the parent and child of any proof step are labelled with the same assignment to universals from all levels $j \geq i$ for which both clauses contain some existential literal at a level greater than j. Similarly, if the proof step is resolution then the two parents are labelled with the same assignment to universals from all levels $j \geq i$ for which they both contain an existential literal at a level greater than j.*

Proof. For universal level $j > i$ assume that the result holds for refutation π_{i+2} and recall that every derivation in π_i is an exact copy of a derivation in π_{i+2}. Labels are copied with clauses, so the result also holds for π_i. In the base case where $i = k - 1$ there are no universal levels $j > i$.

For level i we consider whether the parent and child of a proof step belong to some $\pi_i(P, i)$ for $P \in \mathcal{A}^i_{\pi_i}$. If neither parent nor child belong to any such $\pi_i(P, i)$ then both have been labelled with the level i assignment setting all variables to 0. If the parent is in $\pi_i(P, i)$ for some $P \in \mathcal{A}^i_{\pi_i}$ but the child is not, then the child must be the result of ∀-Red at level i and so contains no existential literals at a level $> i$. If the child is in $\pi_i(P, i)$ for some $P \in \mathcal{A}^i_{\pi_i}$ but the parent is not, then the parent is the result of ∀-Red at some level $\leq i$ and so contains no existential literals at a level $> i$. If both parent and child are in $\pi_i(P, i)$ for some $P \in \mathcal{A}^i_{\pi_i}$ then they are both labelled with α^P_i. It is not possible for the parent and child of a proof step to belong to level i derivation of different clauses in $\mathcal{A}^i_{\pi_i}$ (Lemma 19).

The second statement follows by exactly the same argument. If neither parent belongs to any such $\pi_i(P, i)$, they are labelled identically at level i. If both parents belong to the same $\pi_i(P, i)$ then they are in the same section and are labelled identically at level i. If only one parent belongs to some $\pi_i(P, i)$, then the other is the result of ∀-Red at some level $\leq i$ and so contains no existential literals at a level $> i$. They cannot belong to level i derivation of different clauses in $\mathcal{A}^i_{\pi_i}$ (Lemma 19). ◀

5.3 Putting everything together for the simulation

To create a ∀Exp+Res proof from $E(\pi)$, we simply use the clause labels to generate annotations for the existential literals in each clause and our main result now follows easily:

▶ **Theorem 28.** *Let Φ be a QBF with k quantification blocks and π a* Q-Res *proof of Φ. Then there is an* $\forall$Exp+Res *proof of Φ of size at most $|\pi|^{1+k/2}$.*

Proof. From π, generate the Q-Res refutation $E(\pi)$ of Φ and label the clauses of $E(\pi)$ as described in Section 5.2. The assignment given in a clause label does not satisfy that clause. Remove all universal literals from clauses of $E(\pi)$ and replace all existential literals with annotated literals. An existential literal x in a clause C with label α is replaced by the annotated literal $x^{[\alpha]}$. $\forall$-Red steps are now meaningless and can be removed. Resolution steps remain, acting on annotated literals with matching annotations (Lemma 27). The leaves of $E(\pi)$ were all copies of leaves in π, i.e. clauses from Φ, so the leaves of the constructed $\forall$Exp+Res refutation are annotated versions of those same clauses from Φ. This gives a valid $\forall$Exp+Res proof of Φ constructed from the Q-Res proof. By Lemma 24, the new refutation has size at most $|\pi|^{1+k/2}$. ◀

The cost of transforming a dag-like Q-Res proof into a $\forall$Exp+Res proof by this construction depends on the quantifier complexity. We note that for tree-like proofs we can do the transformation while keeping the size the same, no matter what the number of quantifier alternations. This provides an alternative proof for the simulation of tree-like Q-Res by tree-like $\forall$Exp+Res, shown in [5, 12].

As for the tree-like model, we obtain that $\forall$Exp+Res is strictly stronger than Q-Res also for dag-like proofs, when restricted to QBFs of bounded quantifier complexity. The simulation follows from Theorem 28 and the exponential separation is given by QParity [4], which is a family of Σ_3^b formulas. Thus

▶ **Corollary 29.** *For each $k \geq 3$,* $\forall$Exp+Res *p-simulates* Q-Res *on Σ_k^b formulas, but the reverse simulation does not hold, and there are Σ_3^b formulas providing an exponential separation.*

Proof. The simulation is stated as Theorem 28 and the exponential separation is given by QParity [4], which is a family of Σ_3^b formulas. ◀

To show that this result is tight we combine the result that relaxing the requirement for bounded quantifier complexity allows formulas with polynomial sized dag-like Q-Res proofs but only exponential sized $\forall$Exp+Res proofs [12] with the observation that for QBFs with only one or two quantifier levels Q-Res and $\forall$Exp+Res are p-equivalent. Since we always assume the innermost block to be existential, we only need to check the case of Π_2^b formulas.

▶ **Lemma 30.** Q-Res *and* $\forall$Exp+Res *are p-equivalent on Π_2^b formulas.*

Proof. Consider a QBF of the form $\forall Y \exists X \phi(X, Y)$, where X, Y are blocks of variables and ϕ is a propositional matrix. We first show that in an $\forall$Exp+Res refutation of $\forall Y \exists X \phi(X, Y)$, every literal appearing in that derivation of $\bot$ has the same annotation. This will be used to transform it into a Q-Res refutation. Our induction hypothesis (on the derivation depth) is that in each clause all the literals all have the same annotations. In every axiom clause, every literal is in block X and receives the same annotation in the variables of Y. In order to perform a resolution step, the annotations need to match exactly. This means, under our induction hypothesis, that the two parent clauses must have the same annotation and this is inherited by all the literals in the resolvent. We can therefore show via induction that each clause has the same annotation on all its literals; furthermore since child clauses inherit their parents' annotations, all annotations are the same.

We can then transform this directly into a Q-Res proof, keeping the resolution steps the same, however our axioms now have universal literals. The universal literals will never create a tautology, because they are all falsified by the same assignment.

For the converse, we observe that all universal reduction steps must happen at the end of a Q-Res refutation of $\forall Y \exists X \phi(X,Y)$. Before these reductions we have a non-tautological clause C of entirely universal literals. Since universal literals cannot be removed in any other way, these are the only universal literals that appear in this derivation. We take an assignment to Y that falsifies C. This can be the assignment that is used in a similar ∀Exp+Res refutation. ◀

However, from Theorem 28 it also follows that on QBFs with polylogarithmic quantifier blocks, ∀Exp+Res simulates Q-Res with only a quasi-polynomial blow-up in proof size. Thus the separating examples with short Q-Res proofs but requiring exponential size in ∀Exp+Res necessarily have super-polylogarithmic alternations.

6 Conclusion

Our results demonstrate proof-theoretic advantages of ∀Exp+Res over Q-Res, both for tree-like proofs and for low-quantifier complexity QBFs.

These advantages are not meant to suggest that QCDCL systems are inferior on all accounts. The simulation on bounded quantifier levels becomes less efficient as the number of alternations increase, and the existence of short proofs does not guarantee that proof search will find them. Also, the models of ∀Exp+Res and Q-Res greatly simplify the QBF solving algorithms. Because of unit-propagation in QCDCL solving, sometimes the proofs are better represented in the LD-Q-Res proof system from [1,23]. Also, QBF solvers regularly use dependency schemes [17] which we do not take into account here.

Nor should it be inferred that ∀-Red is an inherently weaker way of dealing with universally quantified variables than expansion. For example the systems Frege+∀-Red and eFrege+∀-Red [3] are very strong, and finding lower bounds for them is equivalent to solving major open problems in circuit complexity or propositional proof complexity [7].

References

1 Valeriy Balabanov and Jie-Hong R. Jiang. Unified QBF certification and its applications. *Formal Methods in System Design*, 41(1):45–65, 2012.

2 Marco Benedetti and Hratch Mangassarian. QBF-based formal verification: Experience and perspectives. *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)*, 5(1-4):133–191, 2008.

3 Olaf Beyersdorff, Ilario Bonacina, and Leroy Chew. Lower bounds: From circuits to QBF proof systems. In *Proc. ACM Conference on Innovations in Theoretical Computer Science (ITCS'16)*, pages 249–260. ACM, 2016.

4 Olaf Beyersdorff, Leroy Chew, and Mikoláš Janota. Proof complexity of resolution-based QBF calculi. In *Proc. Symposium on Theoretical Aspects of Computer Science*, pages 76–89. LIPIcs series, 2015.

5 Olaf Beyersdorff, Leroy Chew, Meena Mahajan, and Anil Shukla. Are short proofs narrow? QBF resolution is not so simple. *ACM Transactions on Computational Logic*, 19(1):1:1–1:26, 2018. (preliminary version in STACS 2016).

6 Olaf Beyersdorff, Nicola Galesi, and Massimo Lauria. A characterization of tree-like resolution size. *Information Processing Letters*, 113(18):666–671, 2013.

7 Olaf Beyersdorff and Ján Pich. Understanding Gentzen and Frege systems for QBF. In *Proc. ACM/IEEE Symposium on Logic in Computer Science (LICS'16)*, 2016.

8 Samuel R. Buss. Towards NP-P via proof complexity and search. *Ann. Pure Appl. Logic*, 163(7):906–917, 2012.

9 Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979.

10 Uwe Egly, Martin Kronegger, Florian Lonsing, and Andreas Pfandler. Conformant planning as a case study of incremental QBF solving. *Ann. Math. Artif. Intell.*, 80(1):21–45, 2017.

11 Mikoláš Janota, William Klieber, João Marques-Silva, and Edmund M. Clarke. Solving QBF with counterexample guided refinement. In Alessandro Cimatti and Roberto Sebastiani, editors, *Proc. 15th International Conference on Theory and Applications of Satisfiability Testing*, volume 7317, pages 114–128. Springer, 2012.

12 Mikoláš Janota and Joao Marques-Silva. Expansion-based QBF solving versus Q-resolution. *Theor. Comput. Sci.*, 577:25–42, 2015.

13 Hans Kleine Büning and Uwe Bubeck. Theory of quantified Boolean formulas. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 735–760. IOS Press, 2009.

14 Hans Kleine Büning, Marek Karpinski, and Andreas Flögel. Resolution for quantified Boolean formulas. *Inf. Comput.*, 117(1):12–18, 1995.

15 William Klieber, Samir Sapra, Sicun Gao, and Edmund M. Clarke. A non-prenex, non-clausal QBF solver with game-state learning. In Ofer Strichman and Stefan Szeider, editors, *Proc. 13th International Conference on Theory and Applications of Satisfiability Testing*, volume 6175, pages 128–142. Springer, 2010.

16 Roman Kontchakov, Luca Pulina, Ulrike Sattler, Thomas Schneider, Petra Selmer, Frank Wolter, and Michael Zakharyaschev. Minimal module extraction from DL-lite ontologies using QBF solvers. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, pages 836–841. AAAI Press, 2009.

17 Florian Lonsing. *Dependency Schemes and Search-Based QBF Solving: Theory and Practice.* PhD thesis, Johannes Kepler University, 2012.

18 Florian Lonsing and Armin Biere. DepQBF: A dependency-aware QBF solver. *JSAT*, 7(2-3):71–76, 2010.

19 Florian Lonsing and Uwe Egly. Evaluating QBF solvers: Quantifier alternations matter. *CoRR*, abs/1701.06612, 2017. URL: `http://arxiv.org/abs/1701.06612`, `arXiv:1701.06612`.

20 Jakob Nordström. On the interplay between proof complexity and SAT solving. *SIGLOG News*, 2(3):19–44, 2015.

21 Markus N Rabe and Leander Tentrup. CAQE: A certifying QBF solver. In *Proceedings of the 15th Conference on Formal Methods in Computer-Aided Design*, pages 136–143. FMCAD Inc, 2015.

22 Moshe Y. Vardi. Boolean satisfiability: theory and engineering. *Commun. ACM*, 57(3):5, 2014.

23 Lintao Zhang and Sharad Malik. Conflict driven learning in a quantified Boolean satisfiability solver. In *ICCAD*, pages 442–449, 2002.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il