
Proceedings Paper:
Beyersdorff, O orcid.org/0000-0002-2870-1648, Chew, L and Sreenivasaiah, K (2015) A game characterisation of tree-like Q-resolution size. In: Dediou, A-H, Formenti, E, Martín-Vide, C and Truthe, B, (eds.) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics. LATA'15, 02-06 Mar 2015, Nice, France. Springer , pp. 486-498. ISBN 978-3-319-15579-1

https://doi.org/10.1007/978-3-319-15579-1_38

A Game Characterisation of Tree-like Q-Resolution Size*

Olaf Beyersdorff[1], Leroy Chew[1], and Karteek Sreenivasaiah[2]

[1] School of Computing, University of Leeds, UK
[2] The Institute of Mathematical Sciences, Chennai, India

Abstract. We provide a characterisation for the size of proofs in tree-like Q-Resolution by a Prover-Delayer game, which is inspired by a similar characterisation for the proof size in classical tree-like Resolution [10]. This gives the first successful transfer of one of the lower bound techniques for classical proof systems to QBF proof systems. We confirm our technique with two previously known hard examples. In particular, we give a proof of the hardness of the formulas of Kleine Büning et al. [20] for tree-like Q-Resolution.

1 Introduction

Proof complexity is a well established field that has rich connections to fundamental problems in computational complexity and logic [14, 21]. In addition to these foundational contributions, proof complexity provides the main theoretical approach towards an understanding of the performance of SAT solvers, which have gained a wide range of applications for the efficient solution of practical instances of NP-hard problems. As most modern SAT solvers employ CDCL-based methods, they correspond to Resolution. Lower bounds to the size and space of Resolution proofs therefore imply sharp bounds for running time and memory consumption of SAT algorithms. Consequently, Resolution has received key attention in proof complexity; and many ingenious techniques have been devised to understand the complexity of Resolution proofs (cf. [13, 26] for surveys).

There has been growing interest and research activity to extend the success of SAT solvers to the more expressive *quantified boolean formulas (QBF)*. Due to its PSPACE completeness, QBF is far more expressive than SAT and thus applies to further fields such as formal verification or planning [6, 25]. As for SAT solvers, runs of QBF solvers produce witnesses (respectively proofs) of unsatisfiability, and there has been great interest in trying to understand which formal system would correspond to the solvers. In particular, a number of Resolution-based proof systems have been developed for QBF, most notably Q-Resolution, introduced by Kleine Büning et al. [20], long-distance Q-Resolution [2], QU-Resolution [27], and ∀Exp+Res [19]. Designing two further calculi IR-calc and

* This work was supported by the EU Marie Curie IRSES grant CORCON, grant no. 48138 from the John Templeton Foundation, EPSRC grant EP/L024233/1, and a Doctoral Training Grant from EPSRC (2nd author).

IRM-calc, a unifying framework for most of these systems has recently been suggested in [7].

Understanding the sizes of proofs in these systems is important as lower bounds for proof size directly translate into lower bounds for running time of the corresponding QBF-solvers. However, in contrast to classical proof complexity we do not yet have many established methods that could be employed for this task. Very recently, the paper [8] introduces a general proof technique for QBF systems based on strategy extraction, that transfers circuit lower bounds to proof size lower bounds. However, no technique for classical Resolution is known to be effective for QBF systems. Except for recent results shown by the new strategy extraction method [8] all present lower bounds for QBF proof systems are either shown ad hoc (e.g. [18] or the lower bound for KBKF(t) in [8]) or are obtained by directly lifting known classical lower bounds to QBF (e.g. [15]).

Our contribution in this paper is to transfer one of the main game methods from classical proof complexity to QBF. Game techniques have a long tradition in proof complexity, as they provide intuitive and simplified methods for lower bounds in Resolution, e.g. for Haken's exponential bound for the pigeonhole principle in dag-like Resolution [23], or the optimal bound in tree-like Resolution [9] , and even work for strong systems [4] and other measures such as proof space [17] and width [1]. A unified game approach to hardness measures was recently established in [12]. Building on the classic game of Pudlák and Impagliazzo [24] for tree-like Resolution, the papers [9, 11] devise an asymmetric Prover-Delayer game, which was shown in [10] to even characterise tree-like Resolution size. Thus, in contrast to the classic symmetric Prover-Delayer game of [24], the asymmetric game in principle allows to always obtain the optimal lower bounds, which was demonstrated in [9] for the pigeonhole principle.

Inspired by these games, we develop here a Prover-Delayer game which tightly characterises the proof size in tree-like Q-Resolution. The idea behind this game is that a Delayer claims to know a satisfying assignment to a false formula, while a Prover asks for values of variables until eventually finding a contradiction. In the course of the game the Delayer scores points proportional to the progress the Prover makes towards reaching a contradiction. By an information-theoretic argument we show that the optimal Delayer will score exactly logarithmically many points in the size of the smallest tree-like Q-Resolution proof of the formula. Thus exhibiting clever Delayer strategies gives lower bounds to the proof size, and in principle these bounds are guaranteed to be optimal. In comparison to the game of [9–11], our formulation here needs a somewhat more powerful Prover, who can forget information as well as freely set universal variables. This is necessary as the Prover needs to simulate more complex Q-Resolution proofs involving universal variables and $\forall$-reductions.

We illustrate this new technique with two examples. The first was used by Janota and Marques-Silva [18] to separate Q-Resolution from the system $\forall$Exp+Res defined in [19]. We use these separating formulas as an easy first illustration of our technique. Our Delayer strategy as well as the analysis here are quite straightforward; in fact, a simple symmetric game in the spirit of [24]

would suffice to get the lower bound. Our second example are the well-known KBKF(t)-formulas of Kleine Büning, Karpinski and Flögel [20]. In the same work [20], where Q-Resolution was introduced, these formulas were suggested as hard formulas for the system. Very recently, the formulas KBKF(t) were even shown to be hard for IR-calc, a system stronger than Q-Resolution [8]. In fact, a number of further separations of QBF proof systems builds on the hardness of KBKF(t) [3,16] (cf. also [8]). Here we use our new technique to show that these formulas require exponential-size proofs in tree-like Q-Resolution. In terms of the lower bound, this result is weaker than the result obtained in [8]. However, it provides an interesting example for our new game technique. In contrast to the first example, both the Delayer strategy as well as the scoring analysis is technically involved. Here we need the refined asymmetric game. The formulas KBKF(t) have very unbalanced proofs, so we cannot use a symmetric Delayer, as symmetric games only yield a lower bound according to the largest full binary tree embeddable into the proof tree (cf. [10]).

The remaining part of this paper is organised as follows. We start in Section 2 with setting up notation and reviewing Q-Resolution. Section 3 contains our characterisation of tree-like Q-Resolution in terms of the Prover-Delayer game. The two mentioned examples for this lower bound technique follow in Sections 4 and 5, the latter of which contains the hardness proof for KBKF(t). We conclude with some open directions for future research in Section 6.

2 Preliminaries

A *literal* is a Boolean variable or its negation; we say that the literal x is *complementary* to the literal $\neg x$ and vice versa. If l is a literal, $\neg l$ denotes the complementary literal, i.e. $\neg\neg x = x$. A *clause* is a disjunction of zero or more literals. The empty clause is denoted by $\bot$, which is semantically equivalent to false. A formula in *conjunctive normal form* (CNF) is a conjunction of clauses. Whenever convenient, a clause is treated as a set of literals and a CNF formula as a set of clauses. For a literal $l = x$ or $l = \neg x$, we write $\mathrm{var}(l)$ for x and extend this notation to $\mathrm{var}(C)$ for a clause C and $\mathrm{var}(\psi)$ for a CNF ψ.

Quantified Boolean Formulas (QBFs) extend propositional logic with quantifiers with the standard semantics that $\forall x.\Psi$ is satisfied by the same truth assignments as $\Psi[0/x] \wedge \Psi[1/x]$ and $\exists x.\Psi$ as $\Psi[0/x] \vee \Psi[1/x]$. Unless specified otherwise, we assume that QBFs are in *closed prenex* form with a CNF *matrix*, i.e., we consider the form $\mathcal{Q}_1 X_1 \dots \mathcal{Q}_k X_k.\ \phi$, where X_i are pairwise disjoint sets of variables; $\mathcal{Q}_i \in \{\exists, \forall\}$ and $\mathcal{Q}_i \neq \mathcal{Q}_{i+1}$. The formula ϕ is in CNF and is defined only on variables $X_1 \cup \dots \cup X_k$. The propositional part ϕ of a QBF is called the *matrix* and the rest the *prefix*. If a variable x is in the set X_i, we say that x is at *level* i and write $\mathrm{lev}(x) = i$; we write $\mathrm{lev}(l)$ for $\mathrm{lev}(\mathrm{var}(l))$. A closed QBF is *false* (resp. *true*), iff it is semantically equivalent to the constant 0 (resp. 1).

Often it is useful to think of a QBF $\mathcal{Q}_1 X_1 \dots \mathcal{Q}_k X_k.\ \phi$ as a *game* between the *universal* and the *existential player*. In the i-th step of the game, the player $\mathcal{Q}_i$ assigns values to the variables X_i. The existential player wins the game iff the

matrix ϕ evaluates to 1 under the assignment constructed in the game. The universal player wins iff ϕ evaluates to 0. A QBF is false iff there is a *winning strategy* for the universal player, i.e. if the universal player can win any game.

Q-Resolution, by Kleine Büning et al. [20], is a resolution-like calculus that operates on QBFs in prenex form where the matrix is a CNF. The rules are given in Figure 1. All proofs in Q-Resolution are refutations, deriving $\emptyset$. Q-Resolution derivations can be associated with a graph where vertices are the clauses of the proof and each resolution inference $\frac{C \quad D}{E}$ gives rise to two directed edges (C, E) and (D, E). Likewise a universal reduction $\frac{C}{D}$ yields an edge (C, D). We speak of *tree-like Q-Resolution* if we only allow Q-Resolution proofs which have trees as its associated graphs. This means that intermediate clauses cannot be used more than once and have to be rederived otherwise. There are exponential separations known between tree-like and dag-like Resolution in the classical case (cf. [26]), that carry over between tree-like and dag-like Q-Resolution.

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

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

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

Literal u is universal. For existential x in clause D, $\mathrm{lev}(x) < \mathrm{lev}(u)$.

Fig. 1. The rules of Q-Res [20]

3 Prover-Delayer Game

In this section, we present a two player game along with a scoring system. The players will be called Prover and Delayer (referred by pronouns 'she' and 'he' respectively). The game is played on a QBF F. The Delayer tries to maximise the score. The Prover tries to win the game by falsifying the formula (which ends the game) and giving the Delayer as small a score as possible. The game proceeds in rounds. Each round of the game has the following phases:

1. *Setting universal variables:* The Prover can assign values to any number of universal variables of her choice that are not blocked, i.e., a universal variable u can be assigned a value by the Prover if all the existential variables with higher quantification level than u are currently unassigned.
2. *Declare Phase:* The Delayer can choose to assign values to any unassigned existential variables of his choice. The Delayer does not score from this.
3. *Query Phase:* This phase has three stages:
   (a) Prover queries any one existential variable x that is currently unassigned.
   (b) Delayer replies with positive weights w_0 and w_1 such that $w_0 + w_1 = 1$.

(c) Prover assigns a value for x. If she assigns $x = b$ for some $b \in \{0, 1\}$, the Delayer scores $\lg(\frac{1}{p_b})$ points.

4. *Forget Phase:* The Prover can forget values of any of the assigned variables of her choice. Any variable chosen in this phase will become unassigned.

The Prover wins the game if any clause in F is falsified. In every round, we check if the Prover has won the game after each phase.

We will now show that our game characterizes tree like Q-Resolution.

Theorem 1. *If ϕ has a tree-like Q-Resolution proof of size at most s, then there exists a Prover strategy such that any Delayer scores at most $\lg\lceil \frac{s}{2} \rceil$ points.*

Proof. We take a similar approach as in [10]. Let Π be a tree-like Q-Resolution refutation of ϕ. Informally, the Prover plays according to Π, starting at the empty clause and following a path in the tree to one of the axioms. At a Resolution inference the Prover will query the resolved variable and at a universal reduction she will set the universal variable. The Prover will keep the invariant that at each moment in the game, the current assignment α assigns exactly all literals from the current clause C on the path in Π, and moreover α falsifies C. This invariant holds in the beginning at the empty clause, and in the end, Prover wins by falsifying an axiom.

We will now elaborate and describe a randomized Prover strategy. Let the Prover be at a node in Π labelled with clause C. We describe what she does in the four stages.

Setting universal variables: If the current clause C was derived in the proof Π by a $\forall$-reduction $\frac{C \vee z}{C}$, then Prover sets $z = 0$. This is possible as the current assignment contains only variables from C and therefore z is not blocked. Prover then moves to the clause $C \vee z$. The Prover repeats this till arriving at a clause derived by the Resolution rule (or winning the game).

Query phase: Prover is now at a clause in Π that was derived by a Resolution step $\frac{C_1 \vee x \;\; C_2 \vee \neg x}{C_1 \vee C_2}$. If the Delayer already set the value of x in his Declare phase, then Prover follows this choice and moves on in the proof tree, possibly setting further universal variables. She does this until she reaches a clause derived by Resolution, where resolved variable x is unassigned. She queries x. On Delayer replying with weights w_0 and w_1, she chooses $x = i$ with probability w_i.

If $x = 0$, then Prover defines S to be the set of all variables not in $C_1 \vee x$ and proceeds down to the subtree rooted at that clause. Else, she defines S to be all variables not in $C_2 \vee \neg x$ and proceeds down to the corresponding subtree.

Forget Phase: The Prover forgets all variables in the set S.

For a fixed Delayer D, let $q_{D,\ell}$ denote the probability (over all random choices made within the game) that the game ends at leaf ℓ. Let π_D be the corresponding distribution induced on the leaves.

For the Prover strategy described above, we have the following claim:

Claim. If the game ends at a leaf ℓ, then the Delayer scores exactly $\alpha_\ell = \lg\left(\frac{1}{q_{D,\ell}}\right)$ points.

Proof. Note that since Π is a tree-like Q-Resolution proof, there is exactly one path from the root of Π to ℓ. Let p be the unique path that leads to the leaf ℓ and let the number of random choices made along p be m. Then, we have $q_{D,\ell} = \prod_{i=1}^{m} q_i$ where q_i is the probability for the ith random choice made along p. Since p is the unique path that leads to ℓ, the number of points α_ℓ scored by the Delayer when the game ends at ℓ is exactly the number of points scored when the game proceeds along the path p. The number of points scored by the Delayer along p is given by: $\alpha_\ell = \sum_{i=1}^{m} \lg\left(\frac{1}{q_i}\right) = \lg\left(\prod_i \frac{1}{q_i}\right) = \lg\left(\frac{1}{q_{D,\ell}}\right)$ □

The Prover strategy we described is randomized. The expected score over all leaves ℓ is the following expression: $\sum_{\text{leaves } \ell \in \Pi} q_{D,\ell}\alpha_\ell = \sum_{\text{leaves } \ell \in \Pi} q_{D,\ell} \lg \frac{1}{q_{D,\ell}}$.

But this quantity is exactly the Shannon entropy $\mathcal{H}(\pi_D)$. Since D is fixed, this entropy will be maximum when π_D is the uniform distribution; i.e., $\mathcal{H}(\pi_D)$ is maximum when, for all leaves ℓ, the probability that the game ends at ℓ is the same. A tree like Q-Resolution proof of size s has at most $\lceil s/2 \rceil$ leaves. So the support of the distribution π_D has size at most $\lceil s/2 \rceil$ and hence $\mathcal{H}(q_{D,\ell}) \leq \lg\lceil s/2 \rceil$.

If the expected score with the randomised Prover is $\leq \lg\lceil s/2 \rceil$, then there is a deterministic Prover who restricts the scores to at most $\lg\lceil s/2 \rceil$. Now we derandomise the Prover by just fixing her random choices accordingly. If the Delayer is optimal she can pick arbitrarily if not she can pick to exploit this. □

To obtain the characterisation of Q-Resolution we also need to show the opposite direction, exhibiting an optimal Delayer:

Theorem 2. *Let ϕ be an unsatisfiable QBF formula and let s be the size of a shortest tree-like Q-Resolution proof for ϕ. Then there exists a Delayer who scores at least $\lg\lceil s/2 \rceil$ points against any Prover.*

Proof. For any unsatisfiable QBF formula ϕ, let $L(\phi)$ denote the number of leaves in the shortest tree-like Q-Resolution proof of ϕ. For a partial assignment α to variables in ϕ, let $\phi|_\alpha$ denote ϕ restricted to the partial assignment α.

The Delayer starts with the empty assignment α and changes α throughout the game. On receiving a query for an existential variable x, the Delayer does the following:

1. Updates α to reflect any changes made by the Prover to any of the variables. These changes include assignments made to both universal variables as well as existential variables.
2. Computes the quantities $\ell_0 = L(\phi|_{\alpha,x=0})$ and $\ell_1 = L(\phi|_{\alpha,x=1})$.
3. Replies with weights $w_0 = \frac{\ell_0}{\ell_0+\ell_1}$ and $w_1 = \frac{\ell_1}{\ell_0+\ell_1}$.

We show by induction on the number of existential variables n in ϕ that the Delayer always scores at least $\lg L(\phi)$ points: Base case $n = 0$, $L(\phi) = 0$ and the Delayer scores at least 0 points. Assume the statement is true for all $n < k$. Now for $n = k$, consider the first query by the Prover, after she possibly made some universal choices according to the partial assignment α. Let the queried

variable be x. If the Prover chose $x = b$ where $b \in \{0, 1\}$, then the Delayer scores $\lg \frac{1}{w_b}$ for this step alone. After assigning $x = b$, the formula $\phi|_{\alpha,x=b}$ has $k-1$ existential variables and hence we use induction hypothesis to conclude that the remaining rounds in the game give the Delayer at least $\lg L(\phi|_{\alpha,x=b})$. Hence the total score is evaluated as: $\lg\left(L(\phi|_{\alpha,x=0}) + L(\phi|_{\alpha,x=1})\right) \geq \lg L(\phi|_\alpha) \geq \lg L(\phi)$.

The last inequality holds, because if $\phi|_\alpha$ is unsatisfiable, we can refute ϕ by deriving a clause with no existential literals, just containing all variables in the domain of α and then $\forall$-reduce. The theorem follows since for any binary tree of size s, the number of leaves is $\lceil s/2 \rceil$. □

4 A First Example

We consider the following formulas studied by Janota and Marques-Silva [18]:

$$F_n = \exists e_1 \forall u_1 \exists c_1^1 c_1^2 \cdots \exists e_i \forall u_i \exists c_i^1 c_i^2 \cdots \exists e_n \forall u_n \exists c_n^1 c_n^2 :$$
$$\bigwedge_{i=1}^{n} (e_i \to c_i^1) \wedge (u_i \to c_i^1) \wedge (\neg e_i \to c_i^2) \wedge (\neg u_i \to c_i^2) \wedge \bigvee_{i=1}^{n} (\neg c_i^1 \vee \neg c_i^2)$$

These formulas were used in [18] to show that $\forall$Exp+Res does not simulate Q-Resolution, i.e., F_n requires exponential-size proofs in $\forall$Exp+Res, but has polynomial-size Q-Resolution proofs. Janota and Marques-Silva [19] also show that $\forall$Exp+Res p-simulates tree-like Q-resolution, and hence it follows that F_n is also hard for the latter system. We reprove this result using our characterisation.

Let $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ be the set of all universal variables. In the following, we show a Delayer strategy that scores at least n points against any Prover. For the Declare phase, the Delayer executes Algorithm 1 till reaching a fixed point. For any variable queried by Prover, Delayer responds with weights $\left(\frac{1}{2}, \frac{1}{2}\right)$. For $i \in [n]$, let $T_i = \{e_i, c_i^1, c_i^2\}$. Let $\mathcal{C} = \bigvee_{i=1}^{n} (\neg c_i^1 \vee \neg c_i^2)$. Note that except for $\mathcal{C}$, all other clauses have only two literals.

Lemma 3. *Algorithm 1 never falsifies a clause that has only two literals.*

Lemma 4. *If the Delayer uses the strategy outlined above, then for any winning Prover strategy, the clause falsified is $\mathcal{C}$.*

Proof. Suppose the clause falsified was D. We will show that if $D \neq \mathcal{C}$, then the Delayer did not use our strategy. We consider the following cases:

Algorithm 1 Declare Routine

for all clauses $(\ell_1 \to \ell_2)$ in F_n **do**
  if $\ell_1 = 1$ **then** Declare $\ell_2 = 1$.
  if $\ell_2 = 0$ and $\text{var}(\ell_1) \notin \mathcal{U}$ **then** Declare $\ell_1 = 0$.
end for

1. D involves variable u_i for some $i \in [n]$:
   Note that u_i appears in clauses with either c_i^1 or c_i^2. Since both c_i^1 and c_i^2 block u_i, it has to be the case that when u_i was set by the Prover, the variables c_i^1 and c_i^2 were unassigned. Now it is straightforward to see that if the Delayer indeed used the declare routine described in Algorithm 1, then all clauses involving u_i become satisfied after u_i is set by the Prover.
2. D is $(e_i \to c_i^1)$ or $(\neg e_i \to c_i^2)$:
   Suppose w.l.o.g. that $D = (e_i \to c_i^1)$. As a consequence of Lemma 3, it must be the case that D was falsified because of the Prover choosing a value for either e_i or c_i^1. So we have two cases:
   - Prover chose a value for e_i to falsify D: So e_i was unassigned just before the query phase began. But if Algorithm 1 left e_i unassigned, then this means c_i is unassigned or $c_i^1 \neq 0$. Hence if the Delayer indeed used Algorithm 1, D could not have been falsified.
   - Prover chose a value for c_i^1 to falsify D: Following an argument just like the previous case, if the Delayer indeed used Algorithm 1, then c_i would be unassigned at the start of the query phase only if $e_i = 0$ or unassigned. In both these cases D cannot be falsified by choosing a value for c_i^1. □

Theorem 5. *Delayer scores at least n points against any Prover strategy.*

Proof. From Lemma 4, it is sufficient to show that any Prover strategy that falsifies $\mathcal{C}$ will give the Delayer a score of at least n. $\mathcal{C}$ can be falsified only if all variables c_i^1, c_i^2 have been assigned to 1. We observe that for any $i \in [n]$, the Prover can get at most one of c_i^1 or c_i^2 to be declared for free by setting u_i appropriately. To assign the other c_i to 1, the Prover can either query c_i directly and set it to 1 or query e_i and set it appropriately. Both these ways give the Delayer 1 point. Hence for every $i \in [n]$, the Delayer scores at least 1 point. □

With Theorem 1 this reproves the hardness of F_n for tree-like Q-Resolution, already implicitly established in [18, 19]:

Corollary 6. *Formulas F_n require tree-like Q-Resolution proofs of size $\Omega(2^n)$.*

This bound is tight as tree-like Q-Resolution refutations of size $O(2^n)$ exist.

5 Hardness of the Formulas of Kleine Büning et al.

In our second example we look at a family of formulas first defined by Kleine Büning, Karpinski and Flögel [20]. The formulas are known to be hard for Q-Resolution and indeed for the stronger system IR-calc [8]. Here we use our technique to give an independent proof of their hardness in tree-like Q-Resolution.

Definition 7 (Kleine Büning, Karpinski and Flögel [20]). *Consider the clauses*

$$
\begin{array}{llll}
C_- = \{\neg y_0\} & C_0 = \{y_0, \neg y_1^0, \neg y_1^1\} & & \\
C_i^0 = \{y_i^0, x_i, \neg y_{i+1}^0, \neg y_{i+1}^1\} & C_i^1 = \{y_i^1, \neg x_i, \neg y_{i+1}^0, \neg y_{i+1}^1\} & & i \in [t-1] \\
C_t^0 = \{y_t^0, x_t, \neg y_{t+1}, \ldots, \neg y_{t+t}\} & C_t^1 = \{y_t^1, \neg x_t, \neg y_{t+1}, \ldots, \neg y_{t+t}\} & & \\
C_{t+i}^0 = \{x_t, y_{t+i}\} & C_{t+i}^1 = \{\neg x_i, y_{t+i}\} & & i \in [t]
\end{array}
$$

The KBKF(t) formulae are defined as the union of these clauses under the quantifier prefix $\exists y_0, y_1^0, y_1^1 \forall x_1 \exists y_2^0, y_2^1 \forall x_2, \ldots, \forall x_{t-1} \exists y_t^0, y_t^1 \forall x_t \exists y_{t+1} \ldots \exists y_{t+t}$.

We now want to show an exponential lower bound on proof size for the KBKF(t) formulas via our game. We will assume throughout that $t > 2$. We start with an informal description of the Delayer strategy.

Delayer strategy – informal description

At any point of time during a run of the game, there is a partial assignment that has been constructed by the Prover and Delayer. We define the following:

Definition 8. *For any partial assignment* α *to the variables, we define* z_α *to be the index of the highest subscript such that an* α *assigns a* 0 *to one or more existential variables with that subscript. If no such subscript exists, then* $z = 0$.

For convenience, we will drop the subscript and just say z when the partial assignment is clear from context. We usually mention the time during a run of the game by referring to z instead of explicitly mentioning the induced partial assignment. The idea behind the Delayer strategy is the following: We observe that for all $i < t-2$ and $j \in \{0,1\}$, to falsify the clause C_i^j, it is necessary that y_i^j is set to 0 and both y_{i+1}^0 and y_{i+1}^1 are set to 1. The strategy we design will not let the Prover win on clauses C_-, C_0, C_i^0, or C_i^1 for any $i < (t-2)$. We do this by declaring either y_{i+1}^0 or y_{i+1}^1 to 0 at a well chosen time. Furthermore, we will show : (1) When the game ends, $z \geq t$ and (2) After any round in the game, the Delayer has a score of at least $O(z)$ It is easy to see that the lower bound of $\Omega(t)$ for the score of the Delayer follows from statements (1) and (2).

Delayer strategy – details

Declare Phase: The Delayer sets y_0 to 0 in the declare phase of the first round.

Let F be the set of all existential variables that were chosen to be forgotten by the Prover in the forget phase of the previous round. The Delayer first does the following "Reset Step": For all variables y in F that had value 0 just before the forget phase of the previous round, the Delayer declares $y = 0$. After the reset step, the Delayer executes Algorithm 2 repeatedly until reaching a fixed point. The notation $y \leftarrow b$ means that the Delayer declares $y = b$ if and only if y is an unassigned variable. Also, we assume that z is updated automatically to be the highest subscript for existential literals set to 0. We observe the following about the reset step:

Observation 9 *The reset step ensures that* z *always increases monotonically (when* z *is measured at the beginning of each query phase).*

Line 14 of Algorithm 2 gives us the following observation:

Observation 10 *After the declare phase, for all* $i < z$*, the existential variables* y_i^0 *and* y_i^1 *has been assigned a value.*

Algorithm 2 Declare Routine

1: $y_z^0 \leftarrow 1$, $y_z^1 \leftarrow 1$, $z' := z$
2: **if** $y_z^{x_z} \neq 0$ or x_z unassigned **then**
3: **for all** $i > z$ **do** $y_i^0 \leftarrow 1$; $y_i^1 \leftarrow 1$
4: **end if**
5: **for** $i = t - 1$ to 1 **do**
6: **for** $j = 0$ to 1 **do**
7: **if** C_i^j is not satisfied with only one literal l that is unassigned **then** Satisfy C_i^j with that literal (if existential).
8: **end for**
9: **end for**
10: **if** $z \leq t - 2$ and either $y_{z+2}^0 = 1$ or $y_{z+2}^1 = 1$ **then** $y_{z+1}^{1-x_{z+1}} \leftarrow 0$
11: **if** $z \neq z'$, x_z assigned and $y_z^{x_z} = 0$ **then**
12: **if** x_{z+1} unassigned **then** $y_{z+1}^0 \leftarrow 0$ **else** $y_{z+1}^{1-x_z} \leftarrow 0$
13: **end if**
14: **for all** $i < z$ **do** $y_i^0 \leftarrow 0$, $y_i^1 \leftarrow 0$

Observation 11 *For all $i > z$, Algorithm 2 assigns all y_i^0 and y_i^1 to 1 before assigning any of them to 0.*

Query Phase:
Let the variable queried be y_i^b. From Observation 10, it is easy to see that $i \geq z$. We have the following cases:

- If $i > t$, then the Delayer replies with weights $w_0 = 2^{z-t-1}$ and $w_1 = 1 - w_0$.
- Else $z \leq i \leq t$. We have two cases:
  - If x_i is unassigned, then the Delayer replies with weights $w_0 = 2^{z-i}$ and $w_1 = 1 - w_0$.
  - Else x_i holds a value. Then we have the following cases:
    - If $b = \neg x_i$, then the Delayer replies with weights $w_0 = 2^{z-i}$ and $w_1 = 1 - w_0$.
    - Else $b = x_i$ and Delayer replies with weight $w_0 = 2^{z-j}$, where j is the largest index such that $\forall k : z < k \leq j, x_k$ is assigned and $y_k^{1-x_k} = 1$. Weight $w_1 = 1 - w_0$.

We now analyze the above strategy: We start with the following lemma:

Lemma 12. *If the Delayer uses the strategy outlined above, then against any Prover, at the end of the game on KBKF(t), $z \geq t$ (where z is defined as in Definition 8).*

Remark 13. If the Prover choses to assign 1 to a variable queried in the query phase on turn k, then by the query phase on turn $k+1$, the value of z (index of the rightmost zero) increments by at most 1. For the increase by 1 it is required that $y_z^{x_z} = 0$ and that for all $c \in \{0, 1\}$, y_{z+1}^c and y_{z+2}^c are unassigned before the query phase on turn k. If the Prover chose to assign 1 to the variable queried and it results in a change of z, then it must cause any of y_{z+1}^0, y_{z+1}^1, y_{z+2}^0 or y_{z+2}^1 to be set to 1, incrementing z be at most one.

For all $i \in [t]$, and $z < t-1$, let $s_z(y_i^c)$ denote the minimum (over all possible Prover strategies) Delayer score when y_i^c is assigned 1 by the Prover for the first time starting from a partial assignment where the right most zero is in column z and every variable to the right of column z is unassigned.

Combining Observation 9 with the fact that at the start of the game $z = 0$, Lemma 12 implies that the Prover increases z by at least t in the process of winning the game. We will now measure the scores that the Delayer accumulates.

Lemma 14. *For all $z < t-1$ and $i < t$, each of $s_z(y_i^0)$ and $s_z(y_i^1)$ is at least $2^{t-i} \lg \frac{2^{t-z}}{2^{t-z}-1}$.*

During a run of the game, z increases from 0 to t. Now we show that the Delayer scores $\Omega(z)$ points during any run of the game on KBKF(t) for large enough t:

Lemma 15. *There exists constants $t_0 > 0$ and $\alpha > 0$ such that for all $t > t_0$, at any point of time during a run of the game on KBKF(t), the Delayer has a score of at least αz.*

To show this lemma, we argue that the Delayer scores $\Omega(1)$ points for every increment of z during the game. This immediately gives us the required claim. Combining Lemma 12 and Lemma 15, we have:

Theorem 16. *There exists a Delayer strategy that scores $\Omega(t)$ against any Prover in the Prover-Delayer game on KBKF(t).*

Corollary 17. *KBKF(t) require tree-like Q-Resolution proofs of size $2^{\Omega(t)}$.*

6 Conclusion

In this paper we have shown that lower bound techniques from classical proof complexity can be transferred to the more complex setting of QBF. We have demonstrated this with respect to prover-delayer games, even obtaining a characterisation of tree-like size in Q-Resolution. Although tree-like (Q-)Resolution is a weak system, it is an important one as it corresponds to runs of the plain DLL algorithm, which serves as the basis of most SAT and QBF-solvers.

A very interesting question for further research is to understand how far this transfer of techniques can be extended. In particular, it seems likely that the very general game-theoretic approaches of [23] can also be utilised for QBF systems. Two other seminal techniques that have found wide-spread applications for classical Resolution are feasible interpolation [22], which also applies to many further systems, and the size-width method of Ben-Sasson and Wigderson [5]. Is it possible to use analogous methods for Q-Resolution and its extensions?

References

1. Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. *Journal of Computer and System Sciences*, 74(3):323–334, 2008.

2. Valeriy Balabanov and Jie-Hong R. Jiang. Unified QBF certification and its applications. *Formal Methods in System Design*, 41(1):45–65, 2012.
3. Valeriy Balabanov, Magdalena Widl, and Jie-Hong R. Jiang. QBF resolution systems and their proof complexities. In *SAT*, pages 154–169, 2014.
4. Eli Ben-Sasson and Prahladh Harsha. Lower bounds for bounded depth Frege proofs via Buss-Pudlák games. *ACM Trans. on Computational Logic*, 11(3), 2010.
5. Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow - resolution made simple. *Journal of the ACM*, 48(2):149–169, 2001.
6. Marco Benedetti and Hratch Mangassarian. QBF-based formal verification: Experience and perspectives. *JSAT*, 5(1-4):133–191, 2008.
7. Olaf Beyersdorff, Leroy Chew, and Mikoláš Janota. On unification of QBF resolution-based calculi. In *MFCS*, pages 81–93, 2014.
8. Olaf Beyersdorff, Leroy Chew, and Mikoláš Janota. Proof complexity of resolution-based QBF calculi. *ECCC*, 21:120, 2014.
9. Olaf Beyersdorff, Nicola Galesi, and Massimo Lauria. A lower bound for the pigeonhole principle in tree-like resolution by asymmetric prover-delayer games. *Information Processing Letters*, 110(23):1074–1077, 2010.
10. Olaf Beyersdorff, Nicola Galesi, and Massimo Lauria. A characterization of tree-like resolution size. *Information Processing Letters*, 113(18):666–671, 2013.
11. Olaf Beyersdorff, Nicola Galesi, and Massimo Lauria. Parameterized complexity of DPLL search procedures. *ACM Trans. on Computational Logic*, 14(3), 2013.
12. Olaf Beyersdorff and Oliver Kullmann. Unified characterisations of resolution hardness measures. In *SAT*, pages 170–187, 2014.
13. Samuel R. Buss. Towards NP-P via proof complexity and search. *Ann. Pure Appl. Logic*, 163(7):906–917, 2012.
14. Stephen A. Cook and Phuong Nguyen. *Logical Foundations of Proof Complexity*. Cambridge University Press, 2010.
15. Uwe Egly. On sequent systems and resolution for QBFs. In *SAT*, p. 100–113, 2012.
16. Uwe Egly, Florian Lonsing, and Magdalena Widl. Long-distance resolution: Proof generation and strategy extraction in search-based QBF solving. In *LPAR*, 2013.
17. Juan Luis Esteban and Jacobo Torán. A combinatorial characterization of treelike resolution space. *Information Processing Letters*, 87(6):295–300, 2003.
18. Mikoláš Janota and Joao Marques-Silva. ∀Exp+Res does not p-simulate Q-resolution. International Workshop on Quantified Boolean Formulas, 2013.
19. Mikoláš Janota and Joao Marques-Silva. On propositional QBF expansions and Q-resolution. In *SAT*, pages 67–82, 2013.
20. Hans Kleine Büning, Marek Karpinski, and Andreas Flögel. Resolution for quantified Boolean formulas. *Inf. Comput.*, 117(1):12–18, 1995.
21. Jan Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*, Cambridge University Press, Cambridge, 1995.
22. Jan Krajíček. Interpolation theorems, lower bounds for proof systems and independence results for bounded arithmetic. *J. Symb. Log.*, 62(2):457–486, 1997.
23. Pavel Pudlák. Proofs as games. *American Math. Monthly*, pages 541–550, 2000.
24. Pavel Pudlák and Russell Impagliazzo. A lower bound for DLL algorithms for SAT. In *Proc. 11th Symposium on Discrete Algorithms*, pages 128–136, 2000.
25. Jussi Rintanen. Asymptotically optimal encodings of conformant planning in QBF. In *AAAI*, pages 1045–1050. AAAI Press, 2007.
26. Nathan Segerlind. The complexity of propositional proofs. *Bulletin of Symbolic Logic*, 13(4):417–481, 2007.
27. Allen Van Gelder. Contributions to the theory of practical quantified Boolean formula solving. In *CP*, pages 647–663. 2012.