Proof Complexity Modulo the Polynomial Hierarchy: Understanding Alternation as a Source of Hardness

Hubie Chen[*]

University of the Basque Country (UPV/EHU), San Sebastián, Spain; and IKERBASQUE, Basque Foundation for Science, Bilbao, Spain
`hubie.chen@ehu.eus`

Abstract

We present and study a framework in which one can present alternation-based lower bounds on proof length in proof systems for quantified Boolean formulas. A key notion in this framework is that of *proof system ensemble*, which is (essentially) a sequence of proof systems where, for each, proof checking can be performed in the polynomial hierarchy. We introduce a proof system ensemble called *relaxing QU-res* which is based on the established proof system *QU-resolution*. Our main results include an exponential separation of the tree-like and general versions of relaxing QU-res, and an exponential lower bound for relaxing QU-res; these are analogs of classical results in propositional proof complexity.

1998 ACM Subject Classification F.1.3 [Computation by Abstract Devices] Complexity measures and classes

Keywords and phrases proof complexity, polynomial hierarchy, quantified propositional logic

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.94

1 Introduction

Background. Traditionally, the area of *propositional proof complexity* studies proof length in propositional proof systems for certifying the unsatisfiability of instances of the *SAT problem*, which instances are quantifier-free propositional formulas [16, 5, 26]. This line of study is supported by multiple motivations; let us highlight a few. First, while satisfiable formulas can be easily certified by a satisfying assignment, it is also natural to desire efficiently verifiable proofs for unsatisfiable formulas (for instance, to check that a SAT algorithm judged unsatisfiability correctly); understanding whether and when proof systems have succinct proofs is a prime concern of this area. Relatedly, *SAT algorithms* for deciding the SAT problem can be typically shown to implicitly generate proofs in a proof system, and thus insight into proof length in the resulting proof system can be used to gain insight into the running-time behavior of SAT algorithms (see for example the discussions in [4, 1]). In addition, the question of whether or not there are proof systems admitting *polynomially bounded proofs* is (when formalized) equivalent to the question of whether or not NP is equal to coNP [16], and one can thus suggest that studying proof length in propositional proof systems sheds light on the relationship between these two complexity classes.

Over recent years, researchers have devoted increasing attention to methods for solving the *QBF problem*, a generalization of the SAT problem and a canonical PSPACE-complete

[*] This work was supported by the Spanish Project MINECO COMMAS TIN2013-46181-C2-R, Basque Project GIU15/30, and Basque Grant UFI11/45.

problem; an instance of this problem is a propositional formula where each variable is either existentially or universally quantified. (*QBF* is short for *quantified Boolean formula.*) It is often suggested that the move to studying this more general problem is based on advances in the efficacy of SAT algorithms (see for example [27]). As reinforces this suggestion, let us point out that one can find QBF solution techniques which use SAT algorithms as black-box, primitive components, and hence which arguably conceive of and treat the SAT problem as feasibly solvable. For instance, sKizzo, a QBF solver dating back to 2005, would convert the QBF being processed to a SAT instance and then call a SAT solver, whenever this was *affordable* [8]. As another example, a different QBF solver which extensively calls a SAT solver during a backtrack-style search was developed and studied [25].

The rise in the study of the QBF problem has resulted in the identification of a number of core algorithmic techniques and corresponding proof systems that aim to capture these (see for example [13, 18, 17, 23, 3, 21, 9, 10] and the references therein). We refer to these proof systems as *QBF proof systems*; they can be used as a basis for certifying a decision for a QBF instance. One can motivate the study of QBF proof systems in much the same way that the study of propositional proof systems has been motivated; hence, these QBF proof systems would seem to suggest a new chapter in the study of proof complexity, and a new domain for the existing lines of inquiry thereof.

However, one is immediately confronted with a dilemma upon inspecting the very basic question of whether or not a typical QBF proof system requires long (exponentially sized) proofs – again, a primary type of question in traditional proof complexity. As an example, let us discuss *Q-resolution* [13], a QBF proof system which is heavily studied and used, in both theory and practice (see for example [2, 19, 18, 22, 23, 3, 9] and the references therein). When applied to SAT instances (viewed as instances of QBF where all variables are existentially quantified), Q-resolution behaves identically to resolution (a heavily studied propositional proof system), and hence the known exponential lower bounds on resolution proof length [20, 7] transfer immediately to Q-resolution. This observation leaves one with a lingering sentiment – which is often expressed by members of the community – that there is something left to be said. After all, Q-resolution is defined on QBF instances, which are substantially more general than SAT instances; the observation does not yield any information about how Q-resolution handles this extra generality, that is, how it copes with alternation of quantifiers. Indeed, there is a sharp disconnect between observing a lower bound for a QBF proof system via a set of SAT instances, and the mentioned treatment of the SAT problem, by QBF algorithms, as a feasibly solvable primitive. These considerations naturally lead to the question of whether or not one can formulate and prove a lower bound which arises from alternation.

Contributions. In this article, we present and study a framework in which it is possible to present such alternation-based lower bounds on proof length in QBF proof systems.

We define a *proof system ensemble* to be an infinite collection of proof systems, where in each proof system, whether or not a given string π constitutes a proof of a given formula Φ can be checked in the polynomial hierarchy (Definition 2). A proof system ensemble is considered to have *polynomially bounded proofs* (for a language) if it contains a proof system which has polynomially bounded proofs in the usual sense (Definition 4). As a result, it is straightforward to define proof system ensembles that have succinct proofs for any set of QBFs with bounded alternation, such as a set of SAT instances (and the proof system ensembles studied herein all have this property); this in turn forces proof length lower bounds, by nature, to arise from a proof system's inability to cope with quantifier alternation.[1] In terms of complexity classes, the question of whether or not there exists a polynomially bounded proof system ensemble for the QBF problem (or any other PSPACE-complete problem) is equivalent to the question of whether or not PSPACE is contained in PH, the polynomial hierarchy (Proposition 5). Indeed, the relationship that traditional proof complexity bears to the NP equals coNP question is analogous to the relationship between the present framework and the PSPACE equals PH question. (Let us point out that no direct implication is known between these two open questions, and so, in a certain sense, progress in one framework may proceed orthogonally to progress in the other!)

One of our main motivations in pursuing this work was to gain further insight into Q-resolution; here, we focus on a slight extension, *QU-resolution* [18], where from existing clauses one can derive new clauses in two ways: by a rule for eliminating literals on universally quantified variables and by resolving two clauses on any variable (in Q-resolution, one can only resolve on existentially quantified variables). Q-resolution, QU-resolution, and their relatives are typically defined only for clausal QBFs – QBFs that consist of a quantifier prefix followed by a conjunction of clauses. We show how to parameterize and lift QU-resolution to obtain a proof system ensemble which we call *relaxing QU-res* which is in fact defined on arbitrary QBFs (indeed, it is defined on what we call *quantified Boolean circuits*), and not just those in clausal form; relaxing QU-res is the main proof system ensemble that we study. Let us overview how we define it.

- We define an *axiom* of a QBF to be a clause which is, in a certain precise sense, entailed by the QBF (see Section 4.1).
- We then show that, given a QBF Φ and a partial assignment a to some of its variables, one can define a QBF $\Phi[a]$ derived naturally from Φ, where the variables on which a is defined have been instantiated (in a certain precise sense; see Section 4.2). This QBF $\Phi[a]$ has the key property that if it is false, then the clause corresponding to a is an axiom of the QBF Φ (see Proposition 8 for a precise statement). We view the notion of inferring clauses from the falsity of QBFs whose variables are partially instantiated as highly natural; indeed, in the case of SAT, performing such inferences is a basis of modern backtracking SAT solvers that perform *clause learning*.
- Recall that each proof system in our proof system ensemble may use, as an oracle, a level of the PH; in particular, the QBF problem restricted to a constant number of alternations may be used as an oracle. In order to infer clauses from a QBF Φ using the method just described, we need a way of detecting falsity of QBFs having the form $\Phi[a]$. But in general, this is difficult; such a QBF $\Phi[a]$ may have a high number of alternations, and thus might not be immediately decidable using an oracle of the described form. To the end of permitting the falsity detection of QBFs $\Phi[a]$ using such oracles, we define the notion of a *relaxation* of a QBF. A relaxation of a QBF Φ is obtained from Φ by changing the order of the quantifier/variable pairs in the quantifier prefix; roughly speaking, such a pair Qv may be moved to the left if Q is the universal quantifier ($\forall$), and may be moved to the right if Q is the existential quantifier ($\exists$). (See Section 4.2 for the precise definition.) A key property of this notion is that if a relaxation of a QBF Φ is false, then the QBF Φ is false (Proposition 9).

[1] Note that there is, a priori, a difference between allowing proof systems oracle access to the SAT problem – which would be natural for modelling QBF solvers that treat the SAT problem as feasibly solvable – and allowing oracle access to arbitrary levels of the PH. We focus on the latter for various reasons: the proof length lower bounds will arise from alternation; we believe that this results in a more robust model; and, this focus causes the proof length lower bounds, which are here of primary interest, to be stronger.

With this notion of relaxation in hand, we define, for each $k \geqslant 2$, the set $H(\Phi, \Pi_k)$ to contain the axioms that arise from QBFs $\Phi[a]$ having Π_k-relaxations (relaxations with a Π_k prefix) that are false. That is, in this set we collect the axioms obtainable by detecting falsity of QBFs $\Phi[a]$ via the consideration of Π_k-relaxations. (Hence, the detection is sound in that it is always correct, but it is not complete). Note that it holds that

$$H(\Phi, \Pi_2) \subseteq H(\Phi, \Pi_3) \subseteq H(\Phi, \Pi_4) \subseteq \cdots .$$

- This gives us a sequence of versions of QU-resolution: for each k, we obtain a version by defining a proof to be a sequence of clauses derived from the axioms $H(\Phi, \Pi_k)$ and the two aforementioned rules of QU-resolution. This sequence is the proof system ensemble *relaxing QU-res*. Let us remark that each of these versions is sound and complete, in a precise sense (see Definition 2 and Proposition 12).

A couple of remarks are in order. First, note that the empty clause is an axiom in $H(\Phi, \Pi_k)$ whenever Φ is a false QBF whose quantifier prefix is Π_k. Consequently, relaxing QU-res is polynomially bounded on any set of false QBFs having bounded alternation. Let us also note that although here we explicitly lift QU-resolution to a proof system ensemble, the approach that we take here can be applied to analogously lift any proof system which is based on deriving clauses from a set of axiom clauses.

Apart from the formulation of the framework, our main results are as follows. We prove an exponential separation between the tree-like and general versions of relaxing QU-res (Section 6), by exhibiting a set of formulas which have polynomial size QU-resolution proofs, but which require exponential size proofs in tree-like relaxing QU-res; this gives an alternation-based analog of the known separation between tree-like and general resolution [12, 6]. Tree-like QU-resolution proofs can be viewed as the traces of a natural backtrack-style QBF decision procedure (this is evident from the viewpoint in Section 4.3, and is also developed explicitly in [14, Section 4.3]), and so this separation formally differentiates the power of such backtracking and general QU-resolution. The lower bound of this separation is based on a prover-delayer game for tree-like QU-resolution proofs (Section 5), which can be viewed as a generalization of a known prover-delayer game for tree-like resolution [24]; note that independently of our work [15], a game similar to ours was presented for tree-like Q-resolution [11]. We also prove an exponential lower bound for relaxing QU-res (Section 7).

All in all, the ideas and techniques developed in this work draw upon and interface concepts from two-player game interaction, proof complexity, and quantified propositional logic. We believe that further progress could benefit from creative input from each of these areas, and certainly look forward to future research on the presented framework.

2 Preliminaries

For each integer k, we use $[k]$ to denote the set that is equal to $\{1, \ldots, k\}$ when $k \geqslant 1$, and that is equal to the empty set $\varnothing$ when $k < 1$. We use $\mathbb{N}$ to denote the natural numbers $\{0, 1, 2, \ldots\}$.

We use $\text{dom}(f)$ to indicate the domain of a function. A function f is a *restriction* of a function g if $\text{dom}(f) \subseteq \text{dom}(g)$ and, for each $a \in \text{dom}(f)$, it holds that $g(a) = f(a)$; when this holds, we also say that g is an *extension* of f. When f is a function, we use $f[a \to b]$ to denote the function on domain $\text{dom}(f) \cup \{a\}$ that maps a to b, and otherwise behaves like f. We write $f \upharpoonright S$ to denote the restriction of a function f to the set S. We say that two functions f and g *agree* if for each element $a \in \text{dom}(f) \cap \text{dom}(g)$, it holds that $f(a) = g(a)$.

When A and B are sets, we use $[A \to B]$ to denote the set of functions from A to B.

Clauses. In this article, we employ the following terminology to discuss clauses. A *literal* is a propositional variable v or the negation $\overline{v}$ thereof. Two literals are *complementary* if one is a variable v and the other is $\overline{v}$; each is said to be the *complement* of the other. A *clause* is a disjunction of literals that contains, for each variable, at most one literal on the variable. A clause is sometimes viewed as the set of the literals that it contains; two clauses are considered equal if they are equal as sets. A clause is *empty* if it does not contain any literals. The variables of a clause are simply the variables that underlie the clause's literals, and the set of variables of a clause α is denoted by $\text{vars}(\alpha)$. When α is a clause, we use $\text{assign}(\alpha)$ to denote the unique propositional assignment f with $\text{dom}(f) = \text{vars}(\alpha)$ such that α evaluates to false under f. In the other direction, when f is a propositional assignment, we use $\text{clause}(f)$ to denote the unique clause α with $\text{vars}(\alpha) = \text{dom}(f)$ that evaluates to false under f. *We will freely and tacitly interchange between a clause α and its corresponding assignment* $\text{assign}(\alpha)$. A clause γ is a *resolvent* of two propositional clauses α and β on variable v if there exists a literal $L \in \alpha$ such that its complement M is in β, $\gamma = (\alpha\backslash\{L\}) \cup (\beta\backslash\{M\})$, and v is the variable underlying L and M.

Quantified Boolean circuits and formulas. We assume basic familiarity with quantified propositional logic. A *QBC* (short for *quantified Boolean circuit*) consists of a quantifier prefix $\vec{P} = Q_1v_1 \dots Q_nv_n$, where each Q_i is a quantifier in $\{\forall, \exists\}$ and each v_i is a propositional variable; and, a Boolean circuit ϕ built from the constants 0 and 1, propositional variables among $\{v_1, \dots, v_n\}$, and the gates AND ($\wedge$), OR ($\vee$), and NOT ($\neg$). We refer to the computational problem of deciding whether or not a QBC is false as the *QBC problem.* For brevity, we sometimes refer to existentially quantified variables as $\exists$-variables, and universally quantified variables as $\forall$-variables. While it is typical to notate a QBC by simply specifying the prefix $\vec{P}$ immediately followed by the circuit ϕ, we will typically separate these two parts by a colon for the sake of readability, using for example $\vec{P} : \phi$. We assume that each quantifier prefix does not contain repeated variables. When $\Phi = \vec{P} : \phi$ is a QBC, by a *partial assignment of* Φ, we refer to a propositional assignment $f : S \to \{0, 1\}$ defined on a subset S of the variables appearing in $\vec{P}$. A *QBF* is a QBC $\vec{P} : \phi$ where ϕ is a Boolean formula. A *clausal QBF* is a QBF $\vec{P} : \phi$ where ϕ is the conjunction of clauses.

Quantifier prefixes. Let $i \geqslant 1$. A quantifier prefix $\vec{P} = Q_1v_1 \dots Q_nv_n$ is Π_i if $Q_1 \dots Q_n$, viewed as a string over the alphabet $\{\forall, \exists\}$, is contained in the language denoted by the regular expression $\forall^*\exists^*\forall^*\exists^* \dots$, which contains i starred quantifiers, beginning with $\forall^*$ and alternating; Σ_i is defined similarly, but with respect to the regular expression $\exists^*\forall^*\exists^*\forall^* \dots$.

The following notation is relative to a quantifier prefix $\vec{P} = Q_1v_1 \dots Q_nv_n$; when we use it, the prefix will be clear from context. We write $v_i \preceq v_j$ if $i \leqslant j$ or if $j < i$ and $Q_j = Q_{j+1} = \dots = Q_i$. We extend this binary relation (and others) to sets in the following natural way: when U and V are sets of variables, we write $U \preceq V$ if for each $u \in U$ and each $v \in V$, it holds that $u \preceq v$. We also write, for example, that $U \preceq v$ for a single variable v when $U \preceq \{v\}$. We write $v_i \equiv v_j$ if $v_i \preceq v_j$ and $v_j \preceq v_i$. It is straightforward to verify that $\equiv$ is an equivalence relation; we refer to each equivalence class of $\equiv$ as a *quantifier block.* We write $v_i \precneqq v_j$ if $v_i \preceq v_j$ and $v_i \not\equiv v_j$. When S is a set of variables, we use $\text{last}(S)$ to denote the variable of S appearing last in the quantifier prefix, that is, the variable v_m, where $m = \max\{i \mid v_i \in S\}$. Typically, when we use the function $\text{last}(S)$, it is in conjunction with the just-defined binary relations, and hence what is most relevant will be the relative location of the quantifier block of $\text{last}(S)$.

Strategies. Let $\Phi = \vec{P} : \phi$ be a QBC; let X denote the $\exists$-variables of Φ, and let Y denote the $\forall$-variables of Φ. When $x \in X$, define $Y_{<x}$ to be the set of variables $\{y \in Y \mid y \lneq x\}$; dually, when $y \in Y$, define $X_{<y}$ to be the set of variables $\{x \in X \mid x \lneq y\}$.

An *$\exists$-strategy* is a sequence of mappings $\sigma = (\sigma_x)_{x \in X}$ where each σ_x is a mapping from $[Y_{<x} \to \{0,1\}]$ to $\{0,1\}$. When $\tau : Y \to \{0,1\}$ is an assignment to the universally quantified variables, we use $\langle \sigma, \tau \rangle$ to denote the assignment f defined by $f(y) = \tau(y)$ for each $y \in Y$ and $f(x) = \sigma_x(\tau \upharpoonright Y_{<x})$ for each $x \in X$. We say that $(\sigma_x)_{x \in X}$ is a *winning $\exists$-strategy* if for every assignment $\tau : Y \to \{0,1\}$, it holds that the assignment $\langle \sigma, \tau \rangle$ satisfies ϕ. A *model* of Φ is defined to be a winning $\exists$-strategy of Φ.

Dually, we define a *$\forall$-strategy* to be a sequence of mappings $\tau = (\tau_y)_{y \in Y}$ where each τ_y is a mapping from $[X_{<y} \to \{0,1\}]$ to $\{0,1\}$. When $\sigma : X \to \{0,1\}$ is an assignment to the existentially quantified variables, we use $\langle \tau, \sigma \rangle$ to denote the assignment f defined by $f(x) = \sigma(x)$ for each $x \in X$ and $f(y) = \tau_y(\sigma \upharpoonright X_{<y})$ for each $y \in Y$. We say that $(\sigma_y)_{y \in Y}$ is a *winning $\forall$-strategy* if for every assignment $\sigma : X \to \{0,1\}$, it holds that the assignment $\langle \tau, \sigma \rangle$ falsifies ϕ.

The following are well-known facts that we will treat as basic.

▶ **Proposition 1.** *Let Φ be a QBC.*

- *There exists a winning $\exists$-strategy for Φ (that is, a model of Φ) if and only if Φ is true.*
- *There exists a winning $\forall$-strategy for Φ if and only if Φ is false.*

3 Proof system ensembles

In this section, we formalize the notion of *proof system ensemble* and present some basic associated notions.

For each $m \geqslant 1$, fix $S(m)$ to be the QBC problem restricted to QBCs having a Σ_m prefix, which is a Σ_m^p-complete problem; for $m = 0$, fix $S(m)$ to be a polynomial-time decidable problem.

Let O be a language; when discussing an algorithm A that makes oracle calls, we use A^O to denote the instantiation of A where oracle calls are answered according to O.

▶ **Definition 2.** A *proof system ensemble* (A, r) *for a language* L consists of an algorithm A which may make oracle calls and receives inputs of the form $(k, (x, \pi))$ where $k \in \mathbb{N}$ and x and π are strings; and, a computable function $r : \mathbb{N} \to \mathbb{N}$ such that:

- For each $k \in \mathbb{N}$, there exists a polynomial p_k such that (for each pair (x, π)) the algorithm $A^{S(r(k))}$ halts on an input $(k, (x, \pi))$ within time $p_k(|(x, \pi)|)$.
- For each $k \in \mathbb{N}$, when L_k is set to $\{(x, \pi) \mid (k, (x, \pi)) \text{ is accepted by } A^{S(r(k))}\}$, it holds that the language $\{x \mid \exists \pi \text{ such that } (x, \pi) \in L_k\}$ is equal to L.

Let us provide an intuitive explanation of Definition 2. For each fixed value of k, the algorithm A provides a proof system for the language L; on inputs of the form $(k, (x, \pi))$, the algorithm is provided oracle access to $S(r(k))$, and needs to accept or reject within polynomial time (in $|(x, \pi)|$). Acceptance indicates that π is judged to be a proof that $x \in L$. The second condition in the definition states that each such proof system is sound and complete, that is, for each fixed k, an arbitrary string x is in L iff there exists a string π such that $(k, (x, \pi))$ is accepted by A.

We use the following terminology to present lower bounds on proof size in proof system ensembles.

▶ **Definition 3.** Let Z be a set of functions from $\mathbb{N}$ to $\mathbb{N}$. A proof system ensemble (A, r) *requires proofs of size* Z on a sequence $\{\Phi_1, \Phi_2, \ldots\}$ of instances if for each k, there exists $z \in Z$ where (for all $n \geqslant 1$ and all strings π) it holds that $(\Phi_n, \pi) \in L_k$ implies $|\pi| \geqslant z(n)$. Here, $|\pi|$ denotes the size of π. We also apply this terminology to other measures defined on proofs.

We say that a function f mapping strings to strings is a *polynomial-length function* if there exists a polynomial q such that, for each string x, it holds that $|f(x)| \leqslant q(|x|)$.

▶ **Definition 4.** A proof system ensemble (A, r) is *polynomially bounded* on a language L if there exists $k \in \mathbb{N}$ and there exists a polynomial-length function f (mapping strings to strings) such that the following holds: if $x \in L$, then it holds that $(x, f(x)) \in L_k$, where L_k is defined as in Definition 2.

▶ **Proposition 5.** *There exists a polynomially bounded proof system ensemble for a language L if and only if L is in the polynomial hierarchy.*

4 Relaxing QU-resolution

4.1 QU-resolution

Let $\Phi = \vec{P} : \phi$ be a QBC. We define an *axiom set of* Φ to be a set H of clauses on variables of $\vec{P}$ such that, for each $C \in H$, C is an *axiom* of Φ in the following sense: each model of $\vec{P} : \phi$ is a model of $\vec{P} : C$. Let us give examples. First, if the QBC Φ is false, then the empty clause is an axiom of Φ. Second, if C is any clause which is entailed by ϕ, then C is an axiom of Φ. A case of this is when a is an assignment to all variables of Φ that falsifies ϕ; then, clause(a) is entailed by ϕ and is an axiom of Φ.

Relative to a QBC $\Phi = \vec{P} : \phi$, we say that a clause C is obtainable from a second clause D by $\forall$-*elimination* if there exists a literal $L \in D$ such that $C = D \backslash \{L\}$ and the variable y underlying L is a $\forall$-variable and has vars$(C) \preceq y$.

With these notions, we define QU-resolution for quantified Boolean circuits in the following way.

▶ **Definition 6.** A *QU-resolution proof* of a QBC $\Phi = \vec{P} : \phi$ from an axiom set H (of Φ) is a finite sequence of clauses where each clause is either in H, is obtainable from a previous clause by $\forall$-elimination, or is obtainable from two previous clauses as a resolvent; in the last two cases, we assume that the clause is annotated with the previous clause(s) from which it is derived (this is to provide a clean correspondence between proofs and certain graphs to be defined, see Section 4.3). The *size* of such a proof is defined as the number of clauses. Such a proof is said to be a *falsity proof* if it ends with the empty clause.

It is a folklore and readily verified fact that when one has a clausal QBF $\Phi = \vec{P} : \phi$ with clause set H, and C appears in a QU-resolution proof of Φ from H, then any model of Φ is a model of $\vec{P} : C$. From this fact and the definition of axiom set, we immediately obtain the following proposition.

▶ **Proposition 7.** *Let C be a clause appearing in a QU-resolution proof of a QBC $\Phi = \vec{P} : \phi$ from axiom set H. Each model of $\vec{P} : \phi$ is a model of $\vec{P} : C$. Consequently, if C is the empty clause, then the QBC Φ is false.*

4.2 Relaxing

In order to define a proof system ensemble based on QU-resolution proofs, we now describe how to obtain a sequence of axiom sets for a given QBC. We start by exhibiting a way to infer that a partial assignment is an axiom of a QBC.

Let a be a partial assignment of a QBC $\Phi = \vec{P} : \phi$. Define $\vec{P}[a]$ to be the quantifier prefix which is equal to $\vec{P}$ but where the variables in $\mathrm{dom}(a)$ and their corresponding quantifiers are removed, and where each quantifier of a variable v with $v \lneq \mathrm{last}(a)$ is changed (if necessary) to an existential quantifier. Define $\phi[a]$ to be the circuit obtained from ϕ by replacing each variable $v \in \mathrm{dom}(a)$ with the constant $a(v)$. Define $\Phi[a]$ to be $\vec{P}[a] : \phi[a]$.

▶ **Proposition 8.** *Assume that a is a partial assignment of a QBC $\Phi = \vec{P} : \phi$ such that $\Phi[a]$ is false. Then* $\mathrm{clause}(a)$ *is an axiom of Φ, that is, each model of $\vec{P} : \phi$ is a model of $\vec{P} : \mathrm{clause}(a)$.*

We believe that Proposition 8 provides a natural way to derive axioms from a QBC. Consider the case where Φ is a SAT instance, that is, $\vec{P}$ is purely existential. In this case, if a is a partial assignment such that $\Phi[a]$ is false, then $\mathrm{clause}(a)$ is an axiom of Φ. Indeed, in this case $\Phi[a]$ is simply the QBC instance obtained by instantiating variables according to a, and then removing the instantiated variables from the quantifier prefix. Note that, in the context of backtrack search for SAT, it is typical that, when some variables have been set according to a partial assignment a, a solver attempts to detect falsity of $\Phi[a]$ by heuristics such as unit propagation and generalizations thereof.

In the case of general QBCs, it is natural to ask, when one has a partial assignment a and then instantiates its variables in ϕ to obtain $\phi[a]$, under what conditions $\mathrm{clause}(a)$ can be inferred as an axiom. Proposition 8 provides an answer to this question; let us explain intuitively why the quantifier prefix is adjusted to $\vec{P}[a]$. Consider the case where the first quantifier block of $\vec{P}$ is existential and a is a partial assignment to variables from this first block; then $\vec{P}[a]$ is simply $\vec{P}$ but with the variables of a removed, and so this case of the proposition generalizes the purely existential case just discussed. In the case where a is arbitrary, $\vec{P}[a]$ can be viewed as the prefix where the lowest number of quantifiers have been changed from universal to existential such that the first quantifier block is existential, and all variables of a fall into this first block.

Prima facie, Proposition 8 may appear to be of limited utility; even if one has oracle access to a level of the polynomial hierarchy, it may be that many partial assignments a give rise to a quantifier prefix $\vec{P}[a]$ which has too many alternations to be resolved by the oracle. In order to expand the class of axioms derivable by this proposition (relative to such an oracle), we introduce now the notion of a *relaxation* of a QBC.

A *relaxation* of a quantifier prefix $\vec{P} = Q_1 v_1 \ldots Q_n v_n$ is a quantifier prefix which has the form $\vec{P'} = Q_{\pi(1)} v_{\pi(1)} \ldots Q_{\pi(n)} v_{\pi(n)}$ where $\pi : [n] \to [n]$ is a permutation and where, for each $\forall$-variable y and for each $\exists$-variable x, it holds that $y \preceq x$ implies $y \preceq' x$; here, $\preceq$ and $\preceq'$ denote the binary relations of $\vec{P}$ and $\vec{P'}$, respectively. As an example, consider the quantifier prefix $\vec{P} = \exists x_1 \exists x_2 \forall y \forall y' \exists x_3$; relaxations thereof include $\forall y \forall y' \exists x_1 \exists x_2 \exists x_3$, $\exists x_1 \forall y' \exists x_2 \forall y \exists x_3$, and $\forall y' \exists x_2 \forall y \exists x_1 \exists x_3$. A *relaxation of a QBC* $\vec{P} : \phi$ is a QBC of the form $\vec{P'} : \phi$ where $\vec{P'}$ is a relaxation of $\vec{P}$; such a QBC is said to be a Π_i*-relaxation* if $\vec{P'}$ is Π_i.

The following is straightforward to verify.

▶ **Proposition 9.** *If a relaxation of a QBC Φ is false, then the QBC Φ is false.*

Note that for any quantifier prefix, a relaxation may be obtained by simply placing the universal quantifiers and their variables first, followed by the existential quantifiers and their variables. Hence, in this sense, each QBC has a canonical Π_2-relaxation, and in the sequel, we focus the discussion on relaxations that are Π_k-relaxations for values of k greater than or equal to 2.

Let Φ be a QBC; for $k \geqslant 2$, we define $H(\Phi, \Pi_k)$ to be the set that contains a clause C if there exists a Π_k-relaxation of $\Phi[\text{assign}(C)]$ that is false. The following fact follows immediately from Propositions 8 and 9.

▶ **Proposition 10.** *When Φ is a QBC and $k \geqslant 2$, it holds that $H(\Phi, \Pi_k)$ is an axiom set of Φ.*

▶ **Definition 11.** *Relaxing QU-res* is defined as the pair (A, r) where r is defined by $r(k) = k + 3$ and A is an algorithm defined to accept an input $(k, (\Phi, \pi))$ if Φ is a QBC and π is a QU-resolution falsity proof of Φ from axioms in $H(\Phi, \Pi_{k+2})$. In particular, the algorithm A examines each clause in π in order; when a clause C is not derived from previous ones by resolution or by $\forall$-elimination, membership of C in $H(\Phi, \Pi_{k+2})$ is checked by the Σ_{k+3} oracle. (Such an oracle can nondeterministically guess a Π_{k+2}-relaxation and then check this relaxation for falsity.)

▶ **Proposition 12.** Relaxing QU-res *is a proof system ensemble for the language of false QBCs.*

Let us now introduce some notions which will be used in our study of *tree-like relaxing QU-res* (defined below). Let f and g be partial assignments of a QBC Φ. We say that g is a *semicompletion* of f if g is an extension of f such that for each universally quantified variable y with $\text{dom}(f) \preceq y$ and $y \notin \text{dom}(f)$, it holds that $\text{dom}(g) \preceq y$ and $y \notin \text{dom}(g)$. A set H of partial assignments of Φ is *semicompletion-closed* if, whenever $f \in H$ and g is a semicompletion of f, it holds that $g \in H$.

4.3 A graph-based view

When $\pi = C_1, \ldots, C_n$ is a QU-resolution proof of a QBC $\vec{P} : \phi$ from axioms H, define $G(\pi)$ to be the directed acyclic graph where there is a vertex for each clause occurrence C_i, which vertex has label $\text{assign}(C_i)$; and, where (for all pairs of clauses C_i, C_j) there is a directed edge from the vertex of C_j to the vertex of C_i if C_j is derived from C_i.

▶ **Proposition 13.** *Let π be a QU-resolution proof of a QBC $\vec{P} : \phi$ from axioms H. The directed acyclic graph $G(\pi)$ has the following properties:*

(α) *If a node with label a has no out-edges, then* $\text{clause}(a)$ *is an element of H.*

(β) *If a node with label a has 1 out-edge to a node with label a', then a' is an extension of a with $\text{dom}(a') = \text{dom}(a) \cup \{y\}$ where y is a universally quantified variable with $\text{dom}(a) \preceq y$.*

(γ) *If a node with label a has 2 out-edges to nodes with labels a_1 and a_2, then there exists a variable v such that a_1 and a_2 are defined on v and $a_1(v) \neq a_2(v)$; $(\text{dom}(a_1) \cup \text{dom}(a_2))\backslash\{v\} = \text{dom}(a)$; a and a_1 are equal on the variables where they are both defined; and, a and a_2 are equal on the variables where they are both defined.*

Moreover, a labelled graph with these three properties naturally induces a QU-resolution proof: for each node, let a be its label, and associate to it $\text{clause}(a)$. □

▶ **Definition 14.** We say that a QU-resolution proof π is *tree-like* if the graph $G(\pi)$ is a tree. We define *tree-like relaxing QU-res* to be the proof system ensemble (A', r) described as follows. Let (A, r) denote relaxing QU-res. Then, the algorithm A' accepts an input $(k, (x, \pi))$ if A accepts it and π is tree-like.

5 A prover-delayer game for *tree-like relaxing QU-res*

In this section, we present a game that can be used to exhibit lower bounds on the size of tree-like QU-resolution proofs; this game can be viewed as a generalization of a game for studying tree-like resolution, which game was presented by Pudlák and Impagliazzo [24].

We first give an intuitive description of the game. Note, however, that this description is meant only to be suggestive. For a precise description, we urge the reader to consult the formal definition, which follows (Definition 15).

Relative to a QBC Φ and a set H of axioms, the game is played between two players, *Prover* and *Delayer*, which maintain a partial assignment. Prover's goal is to reach a partial assignment in H, while Delayer tries to slow down Prover, scoring points in the process. Prover starts by announcing the empty assignment, and Delayer responds with a semicompletion thereof. After this, the play proceeds in a sequence of rounds. In each round, Prover may perform one of three actions to the current assignment f: select a restriction of f; assign a value to a $\forall$-variable $y \notin \mathrm{dom}(f)$ having $\mathrm{dom}(f) \preceq y$; or, select a variable $v \notin \mathrm{dom}(f)$. In the first two cases, Delayer responds with a semicompletion of the resulting assignment. In the third case, Delayer may give a choice to the Prover. When a choice is given, the Prover sets the value of v, and Delayer may elect to claim a point which is then associated with v. When no choice is given, Delayer sets the value of v. After v is set, Delayer responds (as in the first two cases) with a semicompletion of the resulting assignment. Delayer is said to have a p-point strategy if, he has a strategy where, by the time that Prover achieves her goal, there are p variables on which the final assignment is defined such that Delayer has claimed points on these variables. In what follows, we assume $p \geqslant 1$.

▶ **Definition 15.** Let Φ be a QBC. Relative to a set H of axioms, a *p-point delayer strategy* consists of a set F of partial assignments of Φ and a function $s : F \to \mathbb{N}$ called the *score function* such that the following properties hold:

- *(semicompletion-of-empty)* There exists a semicompletion $g \in F$ of the empty assignment such that $s(g) = 0$.
- *(all-points)* If $f \in F \cap H$, then $s(f) \geqslant p$.
- *(monotonicity)* If $g \in F$, then each restriction of g has a semicompletion $f \in F$ such that $s(f) \leqslant s(g)$.
- *($\forall$-branching)* If $f \in F$ and $y \notin \mathrm{dom}(f)$ is a universally quantified variable with $\mathrm{dom}(f) \preceq y$, then, for each $b \in \{0, 1\}$, the assignment $f[y \to b]$ has a semicompletion $g \in F$ with $s(g) = s(f)$.
- *(double-branching)* If $f \in F$ and $v \notin \mathrm{dom}(f)$, there exists a value $b \in \{0, 1\}$ such that $f[v \to b]$ has a semicompletion $g \in F$ where (1) $s(g) \leqslant s(f) + 1$ and (2) if $s(g) = s(f) + 1$, the assignment $f[v \to \neg b]$ has a semicompletion $g' \in F$ with $s(g') \leqslant s(f) + 1$.

▶ **Theorem 16.** *Assume that there exists a p-point delayer strategy for a QBC Φ with respect to a semicompletion-closed axiom set H, and that π is a tree-like QU-resolution proof ending with the empty clause, from axioms H. Then, the tree $G(\pi)$ has at least 2^p leaves.*

6 Separation of the tree-like and general versions of *relaxing QU-res*

The family of sentences to be studied in this section is defined as follows. For each $i \in \{0\} \cup [n]$, define X_i to be the variable set $\{x_{i,j,k} \mid j, k \in \{0, 1\}\}$, and for each $i \in [n]$, define X'_i analogously to be the variable set $\{x'_{i,j,k} \mid j, k \in \{0, 1\}\}$. Define $\vec{P}_n$ to be the prefix $\exists X_0 \exists X'_1 \forall y_1 \exists X_1 \exists X'_2 \forall y_2 \exists X_2 \ldots \exists X'_n \forall y_n \exists X_n$. Note that, for a set of variables X, we use the notation $\exists X$ to represent the existential quantification of the variables in X, in any order (our discussion will always be independent of any particular order chosen). For $i \in [n]$, we refer to the variables in $X'_i \cup \{y_i\} \cup X_i$ as the *level i variables.*

- Define $B = \{\neg x_{0,j,k} \mid j,k \in \{0,1\}\} \cup \{x_{n,j,0} \vee x_{n,j,1} \mid j \in \{0,1\}\}$.
- For each $i \in [n]$ and each $j \in \{0,1\}$ define $H_{i,j} = \{\neg x'_{i,0,k} \vee \neg x'_{i,1,l} \vee x_{i-1,j,0} \vee x_{i-1,j,1} \mid k,l \in \{0,1\}\}$.
  Observe that the clause $\neg x'_{i,0,k} \vee \neg x'_{i,1,l} \vee x_{i-1,j,0} \vee x_{i-1,j,1}$ is logically equivalent to $(x'_{i,0,k} \wedge x'_{i,1,l}) \to (x_{i-1,j,0} \vee x_{i-1,j,1})$.
- For each $i \in [n]$, define $T_i = \{\neg x_{i,0,k} \vee y_i \vee x'_{i,0,k} \mid k \in \{0,1\}\} \cup \{\neg x_{i,1,k} \vee \neg y_i \vee x'_{i,1,k} \mid k \in \{0,1\}\}$.

Define ϕ_n to be the conjunction of the clauses contained in the just-defined sets. Define Φ_n as $\vec{P}_n : \phi_n$. This definition of this family of sentences was inspired partially by the separating formulas of [12, 6].

Let us explain intuitively what the clauses mandate and why the sentences Φ_n are false. By the clauses in B, all of the variables $x_{0,j,k}$ must be set to 0. By the clauses in the sets $H_{1,j}$, either both variables $x'_{1,0,k}$ or both variables $x'_{1,1,k}$ must be set to 0. Once this occurs, the universal player can set the variable y_1 to 0 or 1 to force either both variables $x_{1,0,k}$ or both variables $x_{1,1,k}$ to 0 (respectively), via the clauses in T_1. This reasoning can then be repeated; for instance, at the next level, either both variables $x'_{2,0,k}$ or both variables $x'_{2,1,k}$ must be set to 0, and then after universal player assigning y_2 appropriately, either both variables $x_{2,0,k}$ or both variables $x_{2,1,k}$ are forced to 0. In the end, the existential player must violate one of the two clauses in B concerning level n.

▶ **Proposition 17.** *The sentences $\{\Phi_n\}_{n \geqslant 1}$ have QU-resolution proofs of size linear in n.*

Let $n \geqslant 1$; we will use the following terminology to discuss Φ_n.

We say that r is a *normal realization* of level $i \in [n]$ if it is an assignment defined on the level i variables such that, when b is set to $r(y_i)$, the following hold:

- $0 = r(x_{i,b,0}) = r(x'_{i,b,0}) = r(x_{i,b,1}) = r(x'_{i,b,1})$
- $r(x_{i,\neg b,0}) = r(x'_{i,\neg b,0}) \neq r(x_{i,\neg b,1}) = r(x'_{i,\neg b,1})$

We say that r is a *funny realization* of level $i \in [n]$ if it is an assignment defined on the level i variables such that, when b is set to $r(y_i)$, the following hold:

- $r(x_{i,b,0}) = r(x'_{i,b,0}) \neq r(x_{i,b,1}) = r(x'_{i,b,1})$
- $0 = r(x'_{i,\neg b,0}) = r(x'_{i,\neg b,1})$
- $r(x_{i,\neg b,0}) \neq r(x_{i,\neg b,1})$

We state two key and straightforwardly verified properties of realizations in the following proposition.

▶ **Proposition 18.** *No assignment defined on the level i variables is both a normal realization and a funny realization. Also, each normal realization and each funny realization (of level i) satisfies all clauses in T_i.*

We define the set of assignments F_n to be the set containing all *normal assignments* and all *funny assignments*, which we now turn to define. Let f be a partial assignment of Φ_n. Let $\ell \geqslant 0$ denote the maximum level ℓ such that f is defined on an $\exists$-variable in level ℓ.

We say that f is a *normal assignment* if the following hold:

- f is defined on the variables in $\{x_{0,j,k} \mid j,k \in \{0,1\}\}$ and equal to 0 on them.

- For each $i \in [\ell - 1]$, the restriction of f to the level i variables is a normal realization of level i.
- If $\ell \geqslant 1$, either the restriction of f to the level ℓ variables is a normal realization of level ℓ; or, f is *half-defined* on level ℓ, by which is meant that f is not defined on any variables in $\{x_{\ell,j,k} \mid j,k \in \{0,1\}\}$, but is defined on all variables in $\{x'_{\ell,j,k} \mid j,k \in \{0,1\}\}$ and has $\sum_{j,k\in\{0,1\}} x'_{\ell,j,k} = 1$.

For each normal assignment f, we define $s_n(f) = \ell$.

We say that f is a *funny assignment* if there exists $m \in [\ell]$ such that the following hold:

- f is defined on the variables in $\{x_{0,j,k} \mid j,k \in \{0,1\}\}$ and equal to 0 on them.
- For each $i \in [m - 1]$, the restriction of f to the level i variables is a normal realization of level i.
- The restriction of f to the level m variables is a funny realization of level m.
- For each i with $m < i \leqslant \ell$ and for each $j \in \{0,1\}$, if f is defined on one of the four variables in $\{x_{i,j,k}, x'_{i,j,k} \mid k \in \{0,1\}\}$, then it is defined on all of them and $f(x_{i,j,0}) = f(x'_{i,j,0}) \neq f(x_{i,j,1}) = f(x'_{i,j,1})$.

The following result is obtained by applying the main theorem of the previous section to the strategies (F_n, s_n).

▶ **Theorem 19.** *Tree-like relaxing QU-res requires proofs of size $\Omega(2^n)$ on the sentences $\{\Phi_n\}_{n\geqslant 1}$.*

7 Lower bound for *relaxing QU-res*

We define a family of QBCs, to be studied in this section, as follows. Let $n \geqslant 1$. Define $\vec{P}_n$ to be the quantifier prefix $\exists x_1 \forall y_1 \ldots \exists x_n \forall y_n$. Define $\phi_{n,j}$ to be true if and only if $j + \sum_{i=1}^{n}(x_i + y_i) \not\equiv n \pmod 3$. Define Φ_n to be the sentence $\vec{P}_n : \phi_{n,0}$; these are the sentences that will be used to prove the lower bound. It is straightforward to verify that ϕ_n can be represented as a circuit of size polynomial in n, and we assume that ϕ_n is so represented.

▶ **Proposition 20.** *For each $n \geqslant 1$, the sentence Φ_n is false.*

To obtain the lower bound, we show that for any proof π, the graph $G(\pi)$ must have exponentially many sinks. We begin by showing that any assignment to an initial segment of the $\exists$-variables can be mapped naturally to a sink.

▶ **Lemma 21.** *Let π be a relaxing QU-res proof of Φ_n from an axiom set, and suppose $t \geqslant 1$. Let $f : \{x_1, \ldots, x_{n-\lceil t/2 \rceil}\} \to \{0,1\}$ be an assignment. There exists a sink of $G(\pi)$ whose label agrees with f.*

We next show that each sink must be defined on a variable that occurs *towards the end* of the quantifier prefix, made precise as follows.

▶ **Lemma 22.** *Let π be a relaxing QU-res proof of Φ_n from axiom set $H(\Phi, \Pi_t)$, where $t \geqslant 2$ and $n \geqslant \lceil t/2 \rceil$. Each sink of $G(\pi)$ has a label a that is defined on one of the following variables:*

$$x_{n-(\lceil t/2\rceil-1)}, y_{n-(\lceil t/2\rceil-1)}, \ldots, x_{n-1}, y_{n-1}, x_n, y_n\,.$$

When f is a partial assignment of Φ_n, we refer to the elements of $\{v \mid v \preceq \mathrm{last}(f)\} \backslash \mathrm{dom}(f)$ as *holes*.

▶ **Lemma 23.** *Let π be a relaxing QU-res proof of Φ_n from an axiom set of the form $H(\Phi_n, \Pi_t)$. Each sink of $G(\pi)$ has a label f having at most one hole.*

▶ **Theorem 24.** *Suppose that $t \geqslant 2$ and that $n \geqslant \lceil t/2 \rceil$. Let π be a QU resolution proof of Φ_n from the axiom set $H(\Phi_n, \Pi_t)$. The graph $G(\pi)$ has at least $2^{n-\lceil t/2 \rceil - 1}$ sinks.*

From the previous theorem, we immediately obtain the following.

▶ **Theorem 25.** *Relaxing QU-res requires proofs of size $\Omega(2^n)$ on the sentences $\{\Phi_n\}_{n\geqslant 1}$.*

Acknowledgements. The author thanks Stefan Mengel and Moritz Müller for useful comments on a draft of this article.

References

1 Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. Clause-learning algorithms with many restarts and bounded-width resolution. *J. Artif. Intell. Res. (JAIR)*, 40:353–373, 2011.

2 Valeriy Balabanov and Jie-Hong R. Jiang. Resolution proofs and skolem functions in QBF evaluation and applications. In *Computer Aided Verification – 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, pages 149–164, 2011.

3 Valeriy Balabanov, Magdalena Widl, and Jie-Hong R. Jiang. QBF resolution systems and their proof complexities. In *Theory and Applications of Satisfiability Testing – SAT 2014 – 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, pages 154–169, 2014.

4 Paul Beame, Henry A. Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *J. Artif. Intell. Res. (JAIR)*, 22:319–351, 2004.

5 Paul Beame and Toniann Pitassi. Propositional proof complexity: Past, present and future. *Bulletin of the EATCS*, 65:66–89, 1998.

6 Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. Near optimal separation of tree-like and general resolution. *Combinatorica*, 24(4):585–603, 2004.

7 Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow – resolution made simple. *J. ACM*, 48(2):149–169, 2001.

8 Marco Benedetti. skizzo: A suite to evaluate and certify QBFs. In *Automated Deduction – CADE-20, 20th International Conference on Automated Deduction, Tallinn, Estonia, July 22-27, 2005, Proceedings*, pages 369–376, 2005.

9 Olaf Beyersdorff, Leroy Chew, and Mikolas Janota. On unification of QBF resolution-based calculi. In *Mathematical Foundations of Computer Science 2014 – 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, pages 81–93, 2014.

10 Olaf Beyersdorff, Leroy Chew, and Mikolás Janota. Proof complexity of resolution-based QBF calculi. In *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, pages 76–89, 2015.

11 Olaf Beyersdorff, Leroy Chew, and Karteek Sreenivasaiah. A game characterisation of tree-like q-resolution size. *Electronic Colloquium on Computational Complexity (ECCC)*, 2014.

12 Maria Luisa Bonet, Juan Luis Esteban, Nicola Galesi, and Jan Johannsen. On the relative complexity of resolution refinements and cutting planes proof systems. *SIAM J. Comput.*, 30(5):1462–1484, 2000.

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

14 Hubie Chen. Beyond q-resolution and prenex form: A proof system for quantified constraint satisfaction. *CoRR*, abs/1403.0222, 2014.

15 Hubie Chen. Proof complexity modulo the polynomial hierarchy: Understanding alternation as a source of hardness. *CoRR*, abs/1410.5369, 2014.

16 Stephen A. Cook and Robert A. Reckhow. On the lengths of proofs in the propositional calculus (preliminary version). In *Proceedings of the 6th Annual ACM Symposium on Theory of Computing, April 30 – May 2, 1974, Seattle, Washington, USA*, pages 135–148, 1974.

17 Uwe Egly, Florian Lonsing, and Magdalena Widl. Long-distance resolution: Proof generation and strategy extraction in search-based QBF solving. In *Logic for Programming, Artificial Intelligence, and Reasoning – 19th International Conference, LPAR-19, Stellenbosch, South Africa, December 14-19, 2013. Proceedings*, pages 291–308, 2013.

18 Allen Van Gelder. Contributions to the theory of practical quantified boolean formula solving. In *Principles and Practice of Constraint Programming – 18th International Conference, CP 2012, Québec City, QC, Canada, October 8-12, 2012. Proceedings*, pages 647–663, 2012.

19 Alexandra Goultiaeva, Allen Van Gelder, and Fahiem Bacchus. A uniform approach for generating proofs and strategies for both true and false QBF formulas. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 546–553, 2011.

20 Armin Haken. The intractability of resolution. *Theor. Comput. Sci.*, 39:297–308, 1985.

21 Marijn Heule, Martina Seidl, and Armin Biere. A unified proof system for QBF preprocessing. In *Automated Reasoning – 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014. Proceedings*, pages 91–106, 2014.

22 Mikolás Janota, Radu Grigore, and João Marques-Silva. On QBF proofs and preprocessing. In *Logic for Programming, Artificial Intelligence, and Reasoning – 19th International Conference, LPAR-19, Stellenbosch, South Africa, December 14-19, 2013. Proceedings*, pages 473–489, 2013.

23 Mikolás Janota and Joao Marques-Silva. On propositional QBF expansions and Q-resolution. In *SAT*, pages 67–82, 2013.

24 Pavel Pudlák and Russell Impagliazzo. A lower bound for DLL algorithms for k-sat (preliminary version). In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA.*, pages 128–136, 2000.

25 Horst Samulowitz and Fahiem Bacchus. Using SAT in QBF. In *Principles and Practice of Constraint Programming – CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, pages 578–592, 2005.

26 N. Segerlind. The complexity of propositional proofs. *Bull. Symbolic Logic*, 13:417–626, 2007.

27 Yinlei Yu and Sharad Malik. Validating the result of a quantified boolean formula (QBF) solver: theory and practice. In *Proceedings of the 2005 Conference on Asia South Pacific Design Automation, ASP-DAC 2005, Shanghai, China, January 18-21, 2005*, pages 1047–1051, 2005.