Algorithms and Complexity Group | Institute of Computer Graphics and Algorithms | TU Wien, Vienna, Austria

Technical Report AC-TR-21-011

July 2021

Proof Complexity of Symbolic QBF Reasoning

Stefan Mengel and Friedrich Slivovsky

Proof Complexity of Symbolic QBF Reasoning

Stefan Mengel[1]⋆ and Friedrich Slivovsky[2]⋆⋆

[1] CNRS, UMR 8188, Centre de Recherche en Informatique de Lens (CRIL), Lens, F-62300, France
Univ. Artois, UMR 8188, Lens, F-62300, France
[2] TU Wien, Vienna, Austria

Abstract. We introduce and investigate symbolic proof systems for Quantified Boolean Formulas (QBF) operating on Ordered Binary Decision Diagrams (OBDDs). These systems capture QBF solvers that perform symbolic quantifier elimination, and as such admit short proofs of formulas of bounded path-width and quantifier complexity. As a consequence, we obtain exponential separations from standard clausal proof systems, specifically (long-distance) QU-Resolution and IR-Calc.
We further develop a lower bound technique for symbolic QBF proof systems based on strategy extraction that lifts known lower bounds from communication complexity. This allows us to derive strong lower bounds against symbolic QBF proof systems that are independent of the variable ordering of the underlying OBDDs, and that hold even if the proof system is allowed access to an NP-oracle.

1 Introduction

Unlike in SAT solving, which is dominated by Conflict-Driven Clause Learning (CDCL), in QBF solving there is no single approach that is clearly dominant in practice. Instead, modern solvers are based on variety of techniques, such as (quantified) CDCL [40, 29, 32], expansion of universal variables [9, 25, 10], and abstraction [35, 26, 38].

In practice, these techniques turn out to be complementary, each having strengths and weaknesses on different classes of instances [34, 23, 30]. This complementarity of solvers can be analyzed theoretically by considering proof complexity. Essentially, the different paradigms used in solvers can be formalized as proof systems for QBF, which then can be analyzed with mathematical methods. Then, by separating the strength of different proof systems, one can show that the corresponding solvers are unable to solve problems efficiently that can be dealt with by other solvers. This motivation has led to a great interest in QBF proof complexity over the last few years and resulted in a good understanding

⋆ Partially supported by the PING/ACK project of the French National Agency for Research (ANR-18-CE40-0011).

⋆⋆ Supported by the Vienna Science and Technology Fund (WWTF) under grant ICT19-060.

of common QBF proof systems and how they relate to each other (see [8, 7] and the references therein).

In this paper, we focus on a *symbolic* approach to QBF solving that was originally implemented in the QBDD system [31]. Its underlying idea is to use OBDDs to represent constraints inside the solver, instead of clauses as used by most other SAT and QBF solvers. We formalize QBDD as a proof system in which the lines are OBDDs. More specifically, we consider QBF proof systems that are obtained from propositional OBDD-proof systems by adding $\forall$-reduction (cf. [7]). Propositional proof systems using OBDDs as lines have been studied intensively since the introduction of this model in [1], see e.g. [12]. We thus consider lifting these systems to QBF by adding $\forall$-reduction as very natural.

Analyzing the strength of OBDD-refutations, we first show that, even for a weak propositional system that allows only conjunction of lines and forgetting of variables, the resulting QBF proof system, which we refer to as OBDD$(\wedge, \exists, \forall)$ and which corresponds to traces of QBDD, *p*-simulates QU-resolution. We also show that OBDD$(\wedge, \exists, \forall)$, and in fact also QBDD, can make use of structural properties of QBF in the sense that instances of bounded pathwidth and bounded quantifier alternation can be solved efficiently. We do this by using a recent result on variable elimination for OBDDs from [13] to show that the intermediate OBDDs in QBDD are not too big in this setting. We then observe that other QBF proof systems from the literature have hard instances of bounded pathwidth and bounded quantifier alternation. This shows that OBDD$(\wedge, \exists, \forall)$ can efficiently refute QBFs that are out of reach for many other systems. In particular, it is exponentially separated from (long-distance) QU-resolution [3] and the expansion based IR-calc [8]. It follows that, at least in principle, QBDD can solve instances that other, more modern solvers cannot.

The main technical contribution of this work is a lower bound technique for OBDD-refutations. Here, we consider the strongest possible propositional system, which is semantic entailment of OBDDs. We first show that this system admits efficient strategy extraction of decision lists whose terms are OBDDs. Functions that can be succinctly encoded in this way have short protocols in a communication model from [24] for which it is known that lower bounds can be obtained by proving that a function does not have large monochromatic rectangles. To the best of our knowledge, such bounds are only known for fixed variable partitions. To prove lower bounds for OBDD-refutations that are independent of the variable order chosen for the OBDDs, we lift classical bounds on the inner product function to a graph-based generalization which we show has essentially the same properties as the inner product function, but for *all* variable partitions.

2 Preliminaries

2.1 Propositional Logic and Quantified Boolean Formulas

We assume an infinite set of propositional *variables* and consider *propositional formulas* built up from variables and the constants true (1) and false (0) using conjunction ($\wedge$), disjunction ($\vee$), and negation ($\neg$). We write $\mathsf{var}(\varphi)$ for the set of variables occurring in a formula φ. In particular, we are interested in formulas in *conjunctive normal form (CNF)*. A formula is in CNF if it is a conjunction of *clauses*. A clause is a disjunction of *literals*, and a literal is variable x or a negated variable $\neg x$. An *assignment* of a set X of variables is a mapping $\tau : X \to \{0, 1\}$ of variables to truth values. We write $[X]$ for the set of assignments of X. Given assignments $\tau : X \to \{0, 1\}$ and $\sigma : Y \to \{0, 1\}$ such that X and Y are disjoint, we let $\tau \cup \sigma$ denote the assignment of $X \cup Y$ such that $(\tau \cup \sigma)(x) = \tau(x)$ if $x \in X$ and $(\tau \cup \sigma)(x) = \sigma(x)$ if $x \in Y$. Furthermore, we write $\tau|_{X'}$ for the restriction of τ to $X' \subseteq X$. The result of applying an assignment τ to formula φ and propagating constants is denoted $\varphi[\tau]$. If $\varphi[\tau] = 1$ we say that τ *satisfies* φ, and if $\varphi[\tau] = 0$, the assignment τ *falsifies* φ. A *Quantified Boolean Formula (QBF)* is a pair $\Phi = \mathcal{Q}.\varphi$ consisting of a *quantifier prefix* $\mathcal{Q}$ and a propositional formula φ, called the *matrix* of Φ. If the matrix is in CNF, then Φ is in *Prenex Conjunctive Normal Form (PCNF)*. The quantifier prefix is a sequence $\mathcal{Q} = Q_1 x_1 \dots Q_n x_n$ where the $Q_i \in \{\forall, \exists\}$ are *quantifiers* and the x_i are propositional variables such that $\{x_1, \dots, x_n\} = \mathsf{var}(\varphi)$. We write $D_\Phi(x_i) = \{x_1, \dots, x_{i-1}\}$ for the set of variables that come before x_i in the quantifier prefix, and say x_i *left of* x_j and x_j *is right of* x_i if $i < j$. A variable x_i is *existential* if $Q_i = \exists$, and *universal* if $Q_i = \forall$. We write $\mathsf{var}_\exists(\Phi)$ for the set of existential variables, $\mathsf{var}_\forall(\Phi)$ for the set of universal variables, and $\mathsf{var}(\Phi)$ for the set of all variables occurring in Φ. Let Φ be a QBF. A *universal strategy* for Φ is a family $\boldsymbol{f} = \{f_u\}_{u \in \mathsf{var}_\forall(\Phi)}$ of functions $f_u : [\mathsf{var}(\Phi)] \to \{0, 1\}$ such that $f_u(\tau) = f_u(\sigma)$ for any assignments τ and σ that agree on $D_\Phi(u)$. If $\boldsymbol{f}$ is a universal strategy and $\tau : \mathsf{var}_\exists(\Phi) \to \{0, 1\}$ and assignment of existential variables, we write $\tau \cup \boldsymbol{f}(\tau)$ for the assignment of $\mathsf{var}(\Phi)$ such that $(\tau \cup \boldsymbol{f}(\tau))(x) = \tau(x)$ for existential variables $x \in \mathsf{var}_\exists(\Phi)$ and $(\tau \cup \boldsymbol{f}(\tau))(u) = f_u(\tau \cup \boldsymbol{f}(\tau))$ for universal variables $u \in \mathsf{var}_\forall(\Phi)$. A universal strategy $\boldsymbol{f}$ is a *universal winning strategy* for Φ if $\tau \cup \boldsymbol{f}(\tau)$ falsifies the matrix of Φ for every assignment τ of the existential variables. A QBF is *false* if it has a universal winning strategy, and *true* otherwise.

2.2 Graphs and Pathwidth of Formulas

Let $G = (V, E)$ a graph and for every set $V' \subseteq V$ let $N[V']$ denote the open neighborhood of V, i.e., the set of all vertices in $V \setminus V'$ that have a neighbor in V'. The *expansion* of G is then defined as $\min_{V' \subseteq V, |V'| \leq |V|/2} \frac{|N(V')|}{|V'|}$.

A *path decomposition* of a graph $G = (V, E)$ is a pair (P, λ) where $P = p_1, \dots, p_n$ is a sequence of *nodes* p_i, and $\lambda : \{p_1, \dots, p_n\} \to 2^V$ maps nodes p_i to subsets $\lambda(p_i) \subseteq V$ of vertices called *bags*, subject to the following constraints:

1. Each vertex appears in some bag, that is, $V \subseteq \bigcup_{i=1}^n \lambda(p_i)$,
2. For each edge $vw \in E$ there is a node p_i such that $\{v, w\} \subseteq \lambda(p_i)$.
3. If $v \in \lambda(p_i)$ and $v \in \lambda(p_j)$ for $1 \leq i < j \leq n$, then $v \in \bigcap_{k=i}^j \lambda(p_k)$.

The *width* of a path decomposition is $\max_{i=1}^n |\lambda(p_i)| - 1$, and the *pathwidth* of a graph G is the minimum width of any path decomposition of G. The pathwidth of a CNF formula φ is the pathwidth of its *primal graph*, which is the graph with vertex set $\mathsf{var}(\varphi)$ and edge set $\{xy \mid \exists C \in \varphi \text{ s.t. } x, y \in \mathsf{var}(C)\}$, and the pathwidth of a PCNF formula is the pathwidth of its matrix.

2.3 OBDD

We only give a short introduction into ordered binary decision diagrams (short OBDDs), a classical representation of Boolean functions [11]; see [39] for a text-book treatment.

Let X be a set of variables and π an ordering of X. A π-OBDD on variables X is defined to be a directed acyclic graph B with one source s and two sinks labeled 0 and 1, called the 0- and 1-sink respectively. All non-sink nodes are labeled with variables from X such that on every path P in B the variables appear in the order π. Moreover, all non-sink nodes have two outgoing edges, one labeled with 0, the other with 1. The size of B, denoted by $|B|$, is defined as the number of nodes in B. Given an assignment $a \in \{0, 1\}$, the OBDD B computes a value $B(a)$ as follows: starting in the root, we construct a path by taking for every node v labeled be a variable x the edge labeled with $a(x)$. We continue until we end up in a sink, and the label of the sink is the value of B on a denoted by $B(a)$. This way B computes a Boolean function and every Boolean function can be computed by an OBDD. The OBDD B is called complete if on every source-sink path P all variables in X appear as node labels. The *width* of a complete OBDD B is defined as the maximal number of nodes that are labeled with the same variable.

Observation 1. *There is a polynomial time algorithm that, given an OBDD B, computes an equivalent complete OBDD B'. Moreover, $|B'| \leq (|X| + 1)|B|$.*

Binary Boolean functions can be efficiently applied to OBDDs as stated in the following result.

Lemma 2. *Let $f : \{0, 1\}^2 \to \{0, 1\}$ be a binary Boolean function. Then there is an algorithm that, given two π-OBDDs B_1 and B_2, computes in time polynomial in $|B_1|+|B_2|$ a π-OBDD B such that B computes on input $a \in \{0, 1\}^X$ the value $B(a) := f(B_1(a), B_2(a))$. In particular, the size of B is polynomial in that of B_1 and B_2.*

OBDDs are well-known to be *canonical* in the sense that, for fixed variable order π, there is a unique representation of any Boolean function f by a π-OBDD.

Lemma 3. *Let f be a Boolean function on variables X and let π be a variable order of X. Then there is a unique π-OBDD of minimal size (up to isomorphism) computing f. Moreover, given a π-OBDD representing f, this unique OBDD can be computed in polynomial time. The same is true for complete OBDDs.*

Throughout this paper, we always assume that OBDDs are minimized with the help of the algorithm of Lemma 3.

2.4 Combinatorial Rectangles

Let X be a set of variables and $\Pi = (X_1, X_2)$ a partition of X. We call Π *balanced* if $\min(|X_1|, |X_2|) \geq \lfloor |X|/2 \rfloor$. More generally, for $0 < b \leq 1/2$ we say that Π is b-balanced if $\min(|X_1|, |X_2|) \geq \lfloor b|X| \rfloor$. A combinatorial rectangle with partition Π is a Boolean function $R(X) = R_1(X_1) \wedge R_2(X_2)$. A dual way of seeing R is defining A to be the models of R_1 and B those of R_2. Then the models of R are exactly $A \times B$ and in a slight abuse of notation we then also write $R = A \times B$. The *size* of R is $|A| \cdot |B|$. A function R is called a balanced rectangle if and only if R is a combinatorial rectangle with a partition Π that is balanced.

Let f be a Boolean function and let R be a combinatorial rectangle. We say that R is monochromatic with respect to f if either all models of R are models of f or no model of R is a model of f. When f is clear from the context, we simply call R a monochromatic rectangle without remarking f explicitly. We also say that f *has* the monochromatic rectangle R. The *color* of a monochromatic rectangle R with respect to f is the value $f(x)$ taken by the function on $x \in R$. We will use the following well-known connection between OBDD and rectangles [28].

Theorem 4. *Let g be a function in variables X computed by a π-OBDD of width w. Let X_1 be a prefix of the variable order π and let $X_2 := X \setminus X_1$. Then $g(X) = \bigvee_{i=1}^{w} R_i(X)$, where every R_i is rectangle with partition (X_1, X_2).*

3 Symbolic QBF Proof Systems

We consider line-based QBF proof systems for PCNF formulas where each line is an OBDD. Each derivation begins with a sequence of OBDDs corresponding to the clauses in the matrix of the PCNF formula. New OBDDs are derived by propositional reasoning or universal reduction (cf. Frege systems with universal reduction [7]). Formally, let $\Phi = Q_1 x_1 \dots Q_n x_n . C_1 \wedge \dots \wedge C_m$ be a PCNF formula. An *OBDD derivation* of L_k from Φ is a sequence $L_1, \dots, L_k$ of OBDDs—all with the same variable order π—such that each L_i represents clause C_i for $1 \leq i \leq m$, or is derived using one of the following rules:

1. **conjunction** ($\wedge$): L_i represents $L_j \wedge L_k$ for $j, k < i$.
2. **projection** ($\exists$): L_i represents $\exists x . L_j$ for some $x \in \mathsf{var}(L_j)$ and $j < i$.
3. **entailment** ($\models$): L_i is entailed by $L_{i_1}, \dots, L_{i_k}$, for $i_1, \dots i_k < i$.[3]
4. **universal reduction** ($\forall$): L_i represents $L_j[u/c]$, where $j < i$, u is a universally quantified variable that is rightmost among variables in L_j and $c \in \{0, 1\}$.

Here, $L_j[u/c]$ denotes the OBDD obtained from L by removing each node labeled with variable u and rerouting all incoming edges to its neighbor along the c-labeled edge (effectively substituting c for u). The *size* of an OBDD derivation

[3] Note that OBDD derivations using the entailment rule do not lead to proof systems in the sense of Cook and Reckhow [15], since checking entailment is coNP-hard.

is the sum of the sizes of the OBDDs in the derivation, and the *width* of an OBDD derivation is the maximum width of any OBDD in the derivation.

With the exception of entailment, each of these proof rules can be checked in polynomial time by applying an operation or transformation to the OBDDs in the premises, and verifying that the result—which is unique due to canonicity—matches the OBDD in the conclusion. Moreover, the entailment rule does not trivialize OBDD proofs since it only considers *propositional* entailment, and a QBF can be false without its matrix being unsatisfiable. Finally, it is not difficult to see that OBDD derivations are sound.

Proposition 5. *Let $L_1, \dots, L_k$ be an OBDD derivation from Φ. If Φ is true then $Q_1 x_1 \dots Q_n x_n . L_1 \wedge \dots \wedge L_k$ is true.*

An *OBDD-refutation* of Φ is an OBDD derivation of an OBDD representing 0. A π-*OBDD derivation* is an OBDD derivation where all OBDDs use variable order π. We sometimes explicitly mention the derivation rules used in a proof. For instance, an OBDD$(\wedge, \exists, \forall)$ derivation is OBDD derivation using only conjunction, projection, and universal reduction.

A Proof System for Symbolic Quantifier Elimination

We can use symbolic QBF proof systems to study the QBF solver QBDD proposed by Pan and Vardi [31]. Given a PCNF formula $\Phi = Q_1 x_1 \dots Q_n x_n . \varphi$, QBDD maintains *buckets* $S_1, \dots, S_n$ of OBDDs such that x_i is the rightmost variable (with respect to the quantifier prefix) occurring in the OBDDs of S_i. Initially, the S_i are the sets of clauses in φ that have x_i as their rightmost variable, represented as OBDDs. QBDD proceeds by eliminating variables from the inside out, starting with the variable x_n. To eliminate the variable x_i, it computes the conjunction of OBDDs in bucket S_i, then removes x_i from the result by quantifying either existentially or universally, depending on the quantifier Q_i. The resulting OBDD is then added to the correct bucket. The procedure terminates with a constant 1 or constant 0 OBDD, depending on whether the QBF Φ is true or false. Since any universal variable is innermost upon elimination, a run of QBDD corresponds to an OBDD$(\wedge, \exists, \forall)$-derivation.

Let $\mathsf{tower}(k, 0) := 2^k$ and $\mathsf{tower}(k, q+1) := \mathsf{tower}(2^k, q)$. In this subsection, we prove the following result:

Proposition 6. QBDD *solves PCNF formulas Φ with q quantifier blocks and pathwidth k in time* $\mathsf{tower}(k, q+1)\ \mathsf{poly}(|\Phi|)$.

Since, as stated above, the runs of QBDD are proofs in OBDD$(\wedge, \exists, \forall)$, we directly get the following result on the strength of OBDD$(\wedge, \exists, \forall)$.

Corollary 7. *Every false PCNF Φ with q quantifier width and pathwidth k has an* OBDD$(\wedge, \exists, \forall)$*-refutation of size* $\mathsf{tower}(k, q+1)\mathsf{poly}(|\Phi|)$.

As the basic tool, we use the following variable elimination result for OBDDs.

Lemma 8 ([13]). *Let B be an OBDD of width w and let X be a subset of the variables in B. Then there is an OBDD B' of width 2^w that encodes $\exists X.B$ with the same variable order as B. Moreover B' can be computed in time $2^w\mathsf{poly}(|B|)$.*

Note that since OBDD can be negated without increase of the representation size, we get that the same result is true for $\forall$-elimination. Iterating this result directly yields the following corollary.

Corollary 9. *Let B be an OBDD of width w and let $\mathcal{Q} = Q_1X_1 \dots Q_qX_q$ a quantifier prefix with q blocks. Then the QBF $Q_1X_1 \dots Q_qX_q.B$ has an OBDD representation B' of width $\mathsf{tower}(w, q)$. Moreover, B' can be computed in time $\mathsf{tower}(w, q)\mathsf{poly}(|B|)$.*

An analogous construction for the more general representation of structured DNNF [33] is at the heart of the treewidth based QBF-algorithm in [13].

We can now proceed with the proof of Proposition 6.

Proof (of Proposition 6). Let $\Phi = Q_1X_1 \dots Q_qX_q.\varphi$, and let (P, λ) be a path decomposition of width k of the primal graph of φ. In [20] it is shown that there is a variable order π depending only on (P, λ) such that there is a complete OBDD B of width 2^k computing φ. Let $P_i := \bigwedge_{j\in[i]} S_i$. Then P_i is the conjunction of some clauses of φ, so (P, λ) yields a path decomposition of P_i of width at most k. It follows that for every $i \in [q]$, there is a complete OBDD representation of P_i with order π and width at most 2^k.

We claim that all OBDDs that are computed by QBDD have width at most $\mathsf{tower}(k, q+1)$. Note first that all S_i have pathwidth at most k as above, so we can compute all of them by only conjoining OBDDs with order π and of width at most 2^k. Now whenever we eliminate a variable, the result is a a function that we get from P_i by eliminating some variables. But since these variables are only in q quantifier blocks and we eliminate from the inside out, we have by Corollary 9 that the width of the result is at most $\mathsf{tower}(2^k, q) = \mathsf{tower}(k, q+1)$. Noting that a complete OBDD of width w in n variables has size at most wn and using canonicity and Lemma 2 in all steps completes the proof. $\square$

4 Relation to Other Proof Systems

In this section, we show that OBDD$(\wedge, \exists, \forall)$ is separated from several clausal QBF proof systems. These results are obtained by identifying classes of QBFs that are hard for these proof systems but having bounded pathwidth and a fixed number of quantifier blocks.

We first consider *Q-Resolution* [27], *QU-Resolution* [21], and *Long-Distance Q-Resolution* [2, 19], which can be further generalized and combined into *Long-Distance QU-Resolution* [3].[4] QU-Resolution allows resolution on universal pivots, Long-Distance Q-Resolution can derive tautological clauses in certain cases,

[4] This system is typically referred to as *LQU$^+$-Resolution*.

and Long-Distance QU-Resolution additionally permits the derivation of tautological clauses by resolution on universal pivots.

For all the proof systems above, we define the size of a refutation to be the number of clauses in it. As usual, we say a proof system P *p*-simulates another proof system P' if for every proof Π' in P' there is a proof Π in P such that the length of Π is polynomial in that of Π'.

Proposition 10. $\mathrm{OBDD}(\wedge, \exists, \forall)$ *p-simulates QU-Resolution.*

Proof. We simulate QU-resolution line by line, using the fact that all clauses have small OBDD representations. An application of universal reduction in QU-resolution that removes literal l corresponds to an application of universal reduction in an OBDD derivation that replaces l by 0. Resolution of clauses $C_1 \vee x$ and $\neg x \vee C_2$ can be simulated by first computing an OBDD L' representing $(C_1 \vee x) \wedge (\neg x \vee C_2)$. Each clause C can be represented by an OBDD of size $O(|C|)$, for any variable ordering, so by Lemma 2, the OBDD L' can be computed in time polynomial in the size of the premises. To obtain an OBDD L representing the resolvent $C_1 \vee C_2$, we simply project out the pivot x, that is, $L = \exists x.L'$. □

Lower bounds against QU-Resolution can be obtained by lifting lower bounds against bounded-depth circuits and decision lists [8, 6]. This is because a decision list [36] encoding a universal winning strategy can be efficiently extracted from QU-Resolution refutations [2], and decision lists can be succinctly represented by bounded-depth circuits. For instance, the class QParity of formulas with the parity function as a unique universal winning strategy is hard for QU-Resolution [8]. This class was modified so as to also demonstrate hardness for Long-Distance QU-Resolution, resulting in the class of formulas defined below.

$$\begin{aligned}\mathrm{QUP{\scriptsize ARITY}}_n := {} & \exists x_1 \ldots \exists x_n \forall z_1 \forall z_2 \exists t_2 \ldots \exists t_n . \\ & \mathrm{xor}_u(x_1, x_2, t_2, z_1, z_2) \wedge \mathrm{xor}_u(x_1, x_2, t_2, \neg z_1, \neg z_2) \wedge \\ & \bigwedge_{i=3}^{n} (\mathrm{xor}_u(t_{i-1}, x_i, t_i, z_1, z_2) \wedge \mathrm{xor}_u(t_{i-1}, x_i, t_i, \neg z_1, \neg z_2)) \wedge \\ & (z_1 \vee z_2 \vee t_n) \wedge (\neg z_1 \vee \neg z_2 \vee \neg t_n),\end{aligned}$$

where

$$\begin{aligned}\mathrm{xor}_u(o_1, o_2, o, l_1, l_2) := {} & (l_1 \vee l_2 \vee \neg o_1 \vee o_2 \vee o) \wedge (l_1 \vee l_2 \vee o_1 \vee \neg o_2 \vee o) \wedge \\ & (l_1 \vee l_2 \vee \neg o_1 \vee \neg o_2 \vee \neg o) \wedge (l_1 \vee l_2 \vee o_1 \vee o_2 \vee \neg o).\end{aligned}$$

We restate the following result without a proof.

Theorem 11 ([8]). $\mathrm{QUP{\scriptsize ARITY}}_n$ *requires exponential-size refutations in Long-Distance QU-Resolution.*

At the same time, the QUParity formulas have a very simple structure that can be exploited by symbolic proof systems.

Lemma 12. *The class* $\{\text{QUP}\textsc{arity}_n\}_{n\in\mathbb{N}}$ *has bounded pathwidth.*

Proof. Let $n \in \mathbb{N}$ and consider the path $P = p_1, \ldots, p_n$ and node labeling λ such that $\lambda(p_1) = \{x_1, x_2, t_2, z_1, z_2\}$, $\lambda(p_i) = \{t_i, x_{i+1}, t_{i+1}, z_1, z_2\}$ for $2 \leq i < n$, as well as $\lambda(p_n) = \{z_1, z_2, t_n\}$. It is straightforward to verify that (P, λ) is a path decomposition of $\text{QUP}\textsc{arity}_n$, and its width is 4. □

Since $\text{QUP}\textsc{arity}_n$ only has three quantifier blocks, we obtain the following results by Proposition 6 and Theorem 11.

Corollary 13. *The formulas* $\text{QP}\textsc{arity}_n$ *have polynomial-size* $\text{OBDD}(\wedge, \exists, \forall)$ *refutations.*

Theorem 14. *Long-Distance QU-Resolution does not p-simulate* $\text{OBDD}(\wedge, \exists, \forall)$.

Next, we look at the expansion-based proof system *IR-calc* [8]. For classes of formulas with a bounded number of quantifier blocks, lower bounds against IR-calc can be obtained by considering the *strategy size*, which is the minimum range of any universal winning strategy (as a function mapping assignments of existential variables to assignments of universal variables) [4].

Definition 15 (Strategy Size [4]). *The* strategy size $S(\Phi)$ *of a false QBF* Φ *is the minimum cardinality of the range of a universal winning strategy for* Φ.

Theorem 16 ([4]). *A false PCNF formula* Φ *with at most* k *universal quantifier blocks requires IR-calc proofs of size* $\sqrt[k]{S(\Phi)}$.

We use this correspondence to establish a proof size lower bound for the following class of formulas, which is a variant of the *equality formulas* [5] obtained by splitting the "long" clause $(t_1 \vee \ldots \vee t_n)$ into smaller clauses using auxiliary variables e_i:

$$\begin{aligned}\text{EQ}'_n :=& \exists x_1 \ldots \exists x_n \forall u_1 \ldots \forall u_n \exists t_1 \ldots \exists t_n \exists e_1 \ldots \exists e_n . \\ & \bigwedge_{i=1}^{n} ((x_i \vee u_i \vee \neg t_i) \wedge (\neg x_i \vee \neg u_i \vee \neg t_i)) \wedge \\ & (t_1 \vee e_1) \wedge \bigwedge_{i=2}^{n-1} (\neg e_{i-1} \vee t_i \vee e_i) \wedge (\neg e_{n-1} \vee t_n)\end{aligned}$$

Lemma 17. EQ'_n *is false and the function* $\boldsymbol{f} : \sigma \mapsto \boldsymbol{f}(\sigma)$ *with* $\boldsymbol{f}(\sigma)(u_i) = \sigma(x_i)$ *for* $1 \leq i \leq n$ *is the unique universal winning strategy.*

Proof. Given any assignment σ of the existential variables x_i, applying the joint assignment $\sigma \cup \boldsymbol{f}(\sigma)$ results in unit clauses $\bigwedge_{i=1}^{n} (\neg t_i)$, and unit propagation derives a contradiction. Thus $\boldsymbol{f}$ is a universal winning strategy and EQ'_n is false. Consider an assignment σ of the x_i together with an assignment τ of the u_i such that $\sigma(x_i) \neq \tau(u_i)$ for some i. It is not difficult to see that the formula obtained by applying $\sigma \cup \tau$ can be satisfied by assigning the t_i and e_i appropriately, so the universal player can only win the evaluation game if they play according to $\boldsymbol{f}$. □

Proposition 18. *Any IR-calc refutation of* EQ'_n *has size* $\Omega(2^n)$.

Proof. By Lemma 17 the function $\boldsymbol{f}$ is the unique universal winning strategy for EQ'_n, and the cardinality of its range is 2^n. Thus $2^n = S(\mathrm{EQ}'_n)$ is a proof size lower bound for IR-calc by Theorem 16. $\square$

Lemma 19. *The class* $\{\mathrm{EQ}'_n\}_{n\in\mathbb{N}}$ *has bounded pathwidth.*

Proof. For $n \in \mathbb{N}$, we construct a path decomposition (P, λ) of EQ'_n as follows. We let $P = p_1, \dots, p_n$ and define the labeling λ as $\lambda(p_1) = \{x_1, u_1, t_1, e_1\}$, $\lambda(p_i) = \{e_{i-1}, x_i, u_i, t_i, e_i\}$ for $2 \le i \le n-1$, and $\lambda(p_n) = \{e_{n-1}, x_n, u_n, t_n\}$. $\square$

Corollary 20. *The formulas* EQ'_n *have polynomial-size* $\mathrm{OBDD}(\wedge, \exists, \forall)$ *refutations.*

Theorem 21. *IR-calc does not p-simulate* $\mathrm{OBDD}(\wedge, \exists, \forall)$.

5 A Lower Bound on OBDD Refutations

In this section, we present a technique for proving lower bounds on the size of OBDD-proofs even with the entailment ($\models$) rule. We first show that such proofs admit efficient extraction of universal winning strategies as *OBDD-decision lists*, a model which can in turn be efficiently transformed into *rectangle decision lists*. We then use a result by Impagliazzo and Williams [24] to show that lower bounds for such decision lists reduce to size bounds of rectangles for Boolean functions.

While the variable order must be the same for all OBDDs appearing in an OBDD-proof, it can be chosen arbitrarily so as to minimize proof size. To derive a lower bound using the method sketched above, we thus have to construct a function that does not have large monochromatic rectangles with respect to *any* balanced partition of its arguments. We obtain such a function as a generalization of the well-known *inner product* function.

5.1 From OBDD Proofs to Rectangle Decision Lists

Definition 22. *Let* $\mathcal{C}$ *be a class of Boolean functions. A* $\mathcal{C}$-decision list *of length* s *is a sequence* $(L_1, c_1), \dots, (L_s, c_s)$ *where the* $c_i \in \{0,1\}$ *are truth values and the* $L_i \in \mathcal{C}$ *are circuits, and* L_s *computes the constant function* 1*. Let* V *be the set of variables occurring in the circuits* L_i*. The decision list computes a function* $f : \{0,1\}^V \to \{0,1\}$ *as follows. Given an assignment* $\tau : V \to \{0,1\}$*, let* $i = \min\{1 \le j \le s \mid L_j(\tau) = 1\}$*. The we have* $f(\tau) = c_i$.

A (w, π)-*OBDD-decision list* is a $\mathcal{C}$-decision list where $\mathcal{C}$ is the class of Boolean functions computed by π-OBDDs of maximum width w. Similarly, for a partition (X, Y) of a set V of variables, an (X, Y)-*rectangle decision list* is a $\mathcal{C}$-decision list where $\mathcal{C}$ is the class of rectangles with respect to (X, Y).

The next result states that OBDD-decision lists can be efficiently extracted from OBDD-proofs. Due to space constraints, we omit the proof.

Theorem 23 (Strategy Extraction [2, 7]). *There is a linear-time algorithm that takes a π-OBDD-refutation of a PCNF formula Φ and outputs a family of (w, π)-OBDD-decision lists computing a universal winning strategy for Φ, where w is the width of the refutation.*

Lemma 24. *If there is a (w, π)-OBDD-decision list of length s computing a function $f : \{0,1\}^V \to \{0,1\}$, and (X, Y) is a bipartition of V such that X is the set of variables appearing in a prefix of π, then there is an (X, Y)-rectangle decision list of length $w(s-1)+1$ computing f.*

Proof. Let $(L_1, c_1), \dots, (L_s, c_s)$ be a (w, π)-OBDD-decision list computing function $f : \{0,1\}^V \to \{0,1\}$, and let (X, Y) be a bipartition of V such X corresponds to the variables in a prefix of π. By Theorem 4, each OBDD L_i for $1 \leq i < s$ is equivalent to a disjunction $\bigvee_{j=1}^{w} R_{ij}(V)$ of rectangles with respect to (X, Y). We construct an (X, Y)-rectangle decision list by replacing each pair (L_i, c_i) for $1 \leq i < s$ by the sequence $(R_{i1}, c_i), \dots, (R_{iw}, c_i)$. We can simply append (L_s, c_s) to this sequence since the constant L_s trivially is a rectangle. The resulting (X, Y)-rectangle decision list computes f and has length $w(s-1)+1$. ☐

5.2 From Rectangle Decision Lists to Communication Complexity

We next use a result of Impagliazzo and Williams [24] to prove lower bounds for rectangle decision lists. The following definition has been slightly simplified for our setting.

Definition 25. *Let f be a Boolean function on variables V and let $\Pi = (X, Y)$ be a partition of V. An* AND-protocol *for f with partition Π is the following: two players are given an assignment to X and Y, respectively, and want to compute f on the joint assignment. To this end, they play in several rounds. In each round, they deterministically compute one bit each and send it to a third party. The third party computes the conjunction of the two bits and sends it to the players. If the conjunction evaluates to 1, then the protocol ends and the players have to output the value of f on the given input.*

The length *of the AND-protocol is the maximal number of rounds the players have to play to compute f taken over all possible inputs for f.*

AND-protocols are interesting for us because of the following simple connection already observed without proof by Chattopadhyay et al. [14].

Proposition 26. *Let f be a function in variables V and let Π be a partition of V. If f is computed by a rectangle decision list of length s in which all rectangles have the partition Π, then there is an AND-protocol for φ with partition Π of length at most s.*

Proof. The players simply evaluate the rectangle decision list: for every line (R_i, c_i) where $R_i = R_{i,1}(X_1) \wedge R_{i,2}(X_2)$, the players evaluate $R_{i,1}$ and $R_{i,2}$ on their part of the input individually. Then the third party gives them the conjunction, so the value of the rectangle on the input. If it is 1, then the players know that f evaluates to c_i on their input. $\square$

Lower bounds on the length of AND-protocols can be shown thanks to the following result from [24].

Theorem 27. *Let f be a function in variables V and let Π be a balanced partition of V. If f has an AND-protocol with partition Π of length s, then there is a monochromatic rectangle with respect to f with partition Π of size at least $\frac{1}{4es}2^{|V|}$.*

5.3 A Function with Only Small Monochromatic Rectangles

With Theorem 27, showing lower bounds for rectangle decision lists, and thus for OBDD-refutations, boils down to showing that functions do not have larger monochromatic rectangles. Such function are known in the literature, see e.g. [28], but all results that we are aware of are for a fixed partition of the variables. However, since we want to show lower bounds independent of the choice of the variable order used in the OBDD-refutation, we need functions that have no big monochromatic rectangles for *any* balanced partition of their variables. We will construct such functions in this section.

The following result will be a building block in our construction.

Proposition 28. *Let $F := \bigoplus_{i\in[n]} g_i(x_i, y_i)$ where every function g_i is either $g_i = x_i \wedge y_i$, $g_i = \neg x_i \wedge y_i$, $g_i = x_i \wedge \neg y_i$, or $g_i = x_i \vee y_i$. Then every monochromatic rectangle of F has size at most 2^n.*

To show Proposition 28, we will use the following well known result from communication complexity: Let $\mathsf{IP}(x_1, \ldots, x_n, y_1, \ldots, y_n)$ be the inner product function defined as $\mathsf{IP}(x_1, \ldots, x_n, y_1, \ldots, y_n) := \bigoplus_{i\in[n]} x_i \cdot y_i$ where $\cdot$ denotes the multiplication over $\{0, 1\}$ or equivalently conjunction. The following is well known, see e.g. [28].

Lemma 29. *All monochromatic rectangles of $\mathsf{IP}(x_1, \ldots, x_n, y_1, \ldots, y_n)$ have size at most 2^n.*

It is easy to see that the function F from Proposition 28 is a generalization of the inner product function. We will see that one can easily lift the bound on monochromatic rectangles.

Proof (of Proposition 28). First observe that $x_i \vee y_i = 1 \oplus (\neg x_i \wedge \neg y_i)$, so substituting every occurrence of $x_i \vee y_i$ by $\neg x_i \wedge \neg y_i$ will only change the color but not the size of any monochromatic rectangle. So in the remainder, we assume that there is no $g_i = x_i \vee y_i$ in F.

In a next step, we substitute all occurrences of negated variables by the respective variables without the negation. Call the resulting formula F'. This substitution is clearly a bijection σ between assignments that maintains the value, i.e., $F(X, Y) = F'(\sigma(X, Y))$. Since σ acts on the variables independently, we have that for every monochromatic rectangle $A \times B$ of F, the set $\sigma(A \times B)$

is a monochromatic rectangle as well and $A \times B$ and $\sigma(A \times B)$ have the same size. Now observing that F' is in fact the inner product function completes the proof using Lemma 29. □

We now introduce a generalization of IP with respect to an underlying graph structure. So let X be a set of Boolean variables and let G be a graph with vertex set X and edge set E. Then we define

$$\mathsf{IP}_G(X) = \bigoplus_{xy \in E} x \cdot y.$$

Note that with this definition $\mathsf{IP} = \mathsf{IP}_{M_n}$ where M_n is a matching with n edges. For the statement of the following lemma, recall that a matching is *induced* if it can be obtained as the subgraph induced by the endpoints of its edges.

Lemma 30. *Let $G = (X, E)$ be a graph with n variables. Let $\{e_1, \ldots, e_m\}$ be an induced matching of G and let (X_1, X_2) be a partition of X such that for every e_i one of the end points is in X_1 and one is in X_2. Then every monochromatic rectangle for IP_G respecting the partition (X_1, X_2) has size at most 2^{n-m}.*

Proof. Let X' be the variables that are no end point in any of the e_i. Fix an assignment $a : X' \to \{0, 1\}$. Let $e_i = x_i y_i$ and assume that $x_i \in X_1$ while $y_i \in X_2$. Let $\mathsf{IP}_{G,a}$ be the function in $X'' := \{x_i, y_i \mid i \in [m]\}$ that we get from IP_G by plugging a into the variables X'. Let g_i be the function that, given an assignment a_i to x_i and y_i, counts the number of edges e modulo 2 that are incident to at least one of x_i and y_i and such that $a_i \cup a$ assigns 1 to both end points of e. Clearly, $\mathsf{IP}_{G,a} = \bigoplus_{i \in [m]} g_i(x_i, y_i) \oplus c_a$ where $c_a \in \{0, 1\}$ is a constant depending only on a. We will show that, up to the constant c_a which does not change the size of monochromatic rectangles, the function $\mathsf{IP}_{G,a}$ has the form required by Proposition 28.

To this end, let us analyze g_i. Let $N'(x_i)$ be the neighbors of x_i different from y_i and let $N'(y_i)$ be the neighbors of y_i different from x_i. Let $p_a(x_i)$ be the parity of variables in $N'(x_i)$ that are assigned 1 by a and let $p_a(y_i)$ be defined analogously for y_i. Then $g_i = (p_a(x_i) \wedge x_i) \oplus (p_a(y_i) \wedge y_i) \oplus (x_i \wedge y_i)$. We analyze the different cases:

- If $p_a(x_i) = 0$ and $p_a(y_i) = 0$, then $g_i(x_i, y_i) = x_i \wedge y_i$.
- If $p_a(x_i) = 1$ and $p_a(y_i) = 0$, then $g_i(x_i, y_i) = x_i \oplus (x_i \wedge y_i)$. If $x_i = 0$, then this term is 0, so in all models we must have $x_i = 1$. But $g_i(1, y_i) = 1 \oplus y_i = \neg y_i$, so $g_i(x_i, y_i) = x_i \wedge \neg y_i$.
- If $p_a(x_i) = 0$ and $p_a(y_i) = 1$, then $g_i(x_i, y_i) = \neg x_i \wedge y_i$ is obtained by a symmetric argument.
- Finally, if $p_a(x_i) = 1$ and $p_a(y_i) = 1$ then $g_i(x_i, y_i) = x_i \oplus y_i \oplus (x_i \wedge y_i)$. Clearly, if $x_i = y_i = 0$, then g_i evaluates to 0. Moreover, all other assignments evaluate to 1. So $g_i(x_i, y_i) = x_i \vee y_i$.

Thus, in any case, g_i is of the form required by Proposition 28. It follows that every monochromatic rectangle of $\mathsf{IP}_{G,a}$ has size at most 2^m.

Now consider a monochromatic rectangle R in IP. Then, for every assignment $a : X' \to \{0,1\}$, restricting the variables X' according to a must give a monochromatic rectangle R_a as well. It follows that

$$|R| = \sum_{a:X' \to \{0,1\}} |R_a|.$$

But as we have seen, $|R_a| \leq 2^m$. Moreover, there are $2^{|X'|} = 2^{n-2m}$ assignments to X' and thus $|R| \leq 2^{n-2m}2^m = 2^{n-m}$ as claimed. □

Theorem 31. *Let $G = (X, E)$ be a graph with expansion d, degree Δ and n vertices. Let (X_1, X_2) be a b-balanced partition of X. Then all monochromatic (X_1, X_2)-rectangles have size at most $2^{n\left(1-\frac{nbd^2}{\Delta^2}\right)}$.*

Proof. We show that there is an induced matching of size $\frac{nbd^2}{\Delta^2}$ as in Lemma 30. Then the result follows directly.

Assume w.l.o.g. that $|X_1| \leq |X_2|$. Then, by the expansion property of G, there are at least $d|X_1|$ neighbors of X_1 in X_2. Call these neighbors X_2'. Note that X_2' has at least $d\min(\frac{|X|}{2}, |X_2'|) \geq d^2|X_1|$ neighbors in X_1 where the latter inequality is true because $d \leq 1$. Denote the set of vertices in X_1 that have a neighbor in X_2' by X_1'. Then $|X_1'| \geq d^2|X_1|$.

We now construct a matching between X_1' and X_2'. To this end, first delete all vertices not in $X_1' \cup X_2'$ from G. We then choose a matching iteratively as follows: pick a vertex $x_i \in X_1$ that has not been eliminated and that still has a neighbor y_i in X_2. We add x_iy_i to the matching and delete x_i and y_i and all their neighbors from G. If there are now any vertices in X_i that have no neighbors outside of X_i anymore, we delete those as well. We continue until G is empty.

We now analyze how many rounds we can make at least. First note that we delete at most $2\Delta - 2$ neighbors of x_i and y_i. Moreover, each of them can result in at most $\Delta - 1$ vertices that have no neighbor on the other side of the partition anymore. So overall we delete at most $2\Delta + (2\Delta - 2)(\Delta - 1) = 2\Delta^2 - 2\Delta + 2 \leq 2\Delta^2$ vertices. Since we start with at least $2d^2|X_1| \geq 2nbd^2$ vertices, we can make $\frac{2nbd^2}{2\Delta^2}$ iterations before running out of vertices. □

5.4 Putting It All Together

In this section, we will finally show the promised lower bound for OBDD-refutations by putting together the results of the last sections.

Theorem 32. *There is an infinite sequence (Φ_n) of false PCNF formulas such that $|\Phi_n| = O(n)$ and every OBDD-refutation of Φ_n has size $2^{\Omega(n)}$.*

Proof. Choose a family of graphs of degree at most Δ and expansion d for some constants Δ and d. Such families are well known to exist, see e.g. [22]. Out of this family, choose a sequence (G_n) such that G_n has n vertices X_n. Now let $\varphi_n' = \neg\mathsf{IP}_{G_n}$. Clearly, φ_n' can be computed by a Boolean circuit C_n of size $O(n)$.

We apply Tseitin-transformation on that circuit to get a CNF formula φ_n that has as satisfying assignments exactly the values of all gates in C_n under an assignment to inputs. Note that φ_n has variables for all non-inputs of C_n and thus in particular also for the output; let z be the variable corresponding to the output of C_n and let Y denote the remaining variables of φ_n introduced in the Tseitin-transformation. Then $\mathsf{var}(\varphi_n) = X_n \cup Y \cup \{z\}$. Moreover, φ_n has size $O(n)$. Now define

$$\Phi_n = \exists X_n \forall z \exists Y \varphi_n.$$

Then the only universal winning strategy f_z is to return for every assignment a to X_n the negation of the value that C_n evaluates to under a. But then, using Theorem 23 and Lemma 24, from every refutation of size s and width w of Φ_n, we get a rectangle decision list of length $s' = w(s-1)+1$ for $\neg C_n = \mathsf{IP}_{G_n}$. Using Proposition 26 and Theorem 27, we get that IP_{G_n} has a monochromatic rectangle of size $\frac{1}{4es'}2^{|X_n|} = \frac{1}{4es'}2^n$. But all monochromatic rectangles in IP_{G_n} have size at most $2^{n\left(1-\frac{nbd^2}{\Delta^2}\right)}$ by Theorem 31. Since d, b and Δ are positive constants, it follows that $s' = 2^{\Omega(n)}$. But then at least one of s and w are in $2^{\Omega(n)}$, which gives the desired size bound. □

6 Conclusion

We have introduced OBDD-refutations that model symbolic OBDD-based reasoning for QBF. We have shown that these systems, already in the form that was used (implicitly) in a symbolic QBF solver [31], are surprisingly strong as they allow solving instances that are hard for the proof systems underlying state-of-the-art QBF solvers. In view of this, it may be worthwhile to revisit these techniques in practice. There has been considerable progress in the computation of tree decompositions over the last few years (see e.g. [17]) that could benefit a symbolic approach. Moreover, it could be interesting to use progress in knowledge compilation on generalizations of OBDDs that have similar properties but can be exponentially more succinct [16]. While we consider it unlikely that such an approach would strictly beat current solvers, it might be sufficiently complementary to substantially improve the performance of a portfolio, much like the recently developed ADD-based symbolic model counter ADDMC has been shown to be highly complementary to DPLL-based state-of-the-art solvers [18].

We have also demonstrated limitations of OBDD-refutations by proving exponential lower bounds. Our results require that all OBDDs appearing in a proof have the same variable order, but practical OBDD libraries such as CUDD [37] allow for dynamic variable reordering. While it is not clear how to use this to give more efficient refutations in an implementation of a QBF solver, it would be interesting to see if we can still show lower bounds in this generalized setting. For refutations with variable reordering, the strategy extraction step and the transformation to rectangle decision lists go through unchanged, but there seems to be no equivalent of Theorem 27 for rectangle decision lists with varying partitions. It would be interesting to develop new techniques to show lower bounds in this setting.

References

1. Albert Atserias, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint propagation as a proof system. In Mark Wallace, editor, *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings*, volume 3258 of *Lecture Notes in Computer Science*, pages 77–91. Springer, 2004.
2. Valeriy Balabanov and Jie-Hong R. Jiang. Unified QBF certification and its applications. *Formal Methods Syst. Des.*, 41(1):45–65, 2012.
3. Valeriy Balabanov, Magdalena Widl, and Jie-Hong R. Jiang. QBF resolution systems and their proof complexities. In Carsten Sinz and Uwe Egly, editors, *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*, volume 8561 of *Lecture Notes in Computer Science*, pages 154–169. Springer, 2014.
4. Olaf Beyersdorff and Joshua Blinkhorn. Lower bound techniques for QBF expansion. *Theory Comput. Syst.*, 64(3):400–421, 2020.
5. Olaf Beyersdorff, Joshua Blinkhorn, and Luke Hinde. Size, cost, and capacity: A semantic technique for hard random qbfs. *Log. Methods Comput. Sci.*, 15(1), 2019.
6. Olaf Beyersdorff, Joshua Blinkhorn, and Meena Mahajan. Hardness characterisations and size-width lower bounds for QBF resolution. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages 209–223. ACM, 2020.
7. Olaf Beyersdorff, Ilario Bonacina, Leroy Chew, and Ján Pich. Frege systems for quantified boolean logic. *J. ACM*, 67(2):9:1–9:36, 2020.
8. Olaf Beyersdorff, Leroy Chew, and Mikolás Janota. New resolution-based QBF calculi and their proof complexity. *ACM Trans. Comput. Theory*, 11(4):26:1–26:42, 2019.
9. Armin Biere. Resolve and expand. In *SAT 2004 - The Seventh International Conference on Theory and Applications of Satisfiability Testing, 10-13 May 2004, Vancouver, BC, Canada, Online Proceedings*, 2004.
10. Roderick Bloem, Nicolas Braud-Santoni, Vedad Hadzic, Uwe Egly, Florian Lonsing, and Martina Seidl. Expansion-based QBF solving without recursion. In Nikolaj Bjørner and Arie Gurfinkel, editors, *2018 Formal Methods in Computer Aided Design, FMCAD 2018, Austin, TX, USA, October 30 - November 2, 2018*, pages 1–10. IEEE, 2018.
11. Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Computers*, 35(8):677–691, 1986.
12. Sam Buss, Dmitry Itsykson, Alexander Knop, and Dmitry Sokolov. Reordering rule makes OBDD proof systems stronger. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, volume 102 of *LIPIcs*, pages 16:1–16:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
13. Florent Capelli and Stefan Mengel. Tractable QBF by knowledge compilation. In Rolf Niedermeier and Christophe Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*, volume 126 of *LIPIcs*, pages 18:1–18:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

14. Arkadev Chattopadhyay, Meena Mahajan, Nikhil S. Mande, and Nitin Saurabh. Lower bounds for linear decision lists. *Chic. J. Theor. Comput. Sci.*, 2020, 2020.
15. Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *J. Symb. Log.*, 44(1):36–50, 1979.
16. Adnan Darwiche. SDD: A new canonical representation of propositional knowledge bases. In Toby Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 819–826. IJCAI/AAAI, 2011.
17. Holger Dell, Christian Komusiewicz, Nimrod Talmon, and Mathias Weller. The PACE 2017 parameterized algorithms and computational experiments challenge: The second iteration. In Daniel Lokshtanov and Naomi Nishimura, editors, *12th International Symposium on Parameterized and Exact Computation, IPEC 2017, September 6-8, 2017, Vienna, Austria*, volume 89 of *LIPIcs*, pages 30:1–30:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.
18. Jeffrey M. Dudek, Vu Phan, and Moshe Y. Vardi. ADDMC: weighted model counting with algebraic decision diagrams. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 1468–1476. AAAI Press, 2020.
19. Uwe Egly, Florian Lonsing, and Magdalena Widl. Long-distance resolution: Proof generation and strategy extraction in search-based QBF solving. In Kenneth L. McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning - 19th International Conference, LPAR-19, Stellenbosch, South Africa, December 14-19, 2013. Proceedings*, volume 8312 of *Lecture Notes in Computer Science*, pages 291–308. Springer, 2013.
20. Andrea Ferrara, Guoqiang Pan, and Moshe Y. Vardi. Treewidth in verification: Local vs. global. In Geoff Sutcliffe and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 12th International Conference, LPAR 2005, Montego Bay, Jamaica, December 2-6, 2005, Proceedings*, volume 3835 of *Lecture Notes in Computer Science*, pages 489–503. Springer, 2005.
21. Allen Van Gelder. Contributions to the theory of practical quantified boolean formula solving. In Michela Milano, editor, *Principles and Practice of Constraint Programming - 18th International Conference, CP 2012, Québec City, QC, Canada, October 8-12, 2012. Proceedings*, volume 7514 of *Lecture Notes in Computer Science*, pages 647–663. Springer, 2012.
22. Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.
23. Holger H. Hoos, Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Portfolio-based algorithm selection for circuit qbfs. In John N. Hooker, editor, *Principles and Practice of Constraint Programming - 24th International Conference, CP 2018, Lille, France, August 27-31, 2018, Proceedings*, volume 11008 of *Lecture Notes in Computer Science*, pages 195–209. Springer, 2018.
24. Russell Impagliazzo and Ryan Williams. Communication complexity with synchronized clocks. In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, USA, June 9-12, 2010*, pages 259–269. IEEE Computer Society, 2010.
25. Mikolás Janota, William Klieber, João Marques-Silva, and Edmund M. Clarke. Solving QBF with counterexample guided refinement. *Artif. Intell.*, 234:1–25, 2016.

26. Mikolás Janota and João Marques-Silva. Solving QBF by clause selection. In Qiang Yang and Michael J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 325–331. AAAI Press, 2015.
27. Hans Kleine Büning, Marek Karpinski, and Andreas Flögel. Resolution for quantified boolean formulas. *Inf. Comput.*, 117(1):12–18, 1995.
28. Eyal Kushilevitz and Noam Nisan. *Communication complexity.* Cambridge University Press, 1997.
29. Florian Lonsing and Armin Biere. Depqbf: A dependency-aware QBF solver. *J. Satisf. Boolean Model. Comput.*, 7(2-3):71–76, 2010.
30. Florian Lonsing and Uwe Egly. Evaluating QBF solvers: Quantifier alternations matter. In John N. Hooker, editor, *Principles and Practice of Constraint Programming - 24th International Conference, CP 2018, Lille, France, August 27-31, 2018, Proceedings*, volume 11008 of *Lecture Notes in Computer Science*, pages 276–294. Springer, 2018.
31. Guoqiang Pan and Moshe Y. Vardi. Symbolic decision procedures for QBF. In Mark Wallace, editor, *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings*, volume 3258 of *Lecture Notes in Computer Science*, pages 453–467. Springer, 2004.
32. Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Dependency learning for QBF. *J. Artif. Intell. Res.*, 65:180–208, 2019.
33. Knot Pipatsrisawat and Adnan Darwiche. New compilation languages based on structured decomposability. In Dieter Fox and Carla P. Gomes, editors, *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 517–522. AAAI Press, 2008.
34. Luca Pulina and Armando Tacchella. A self-adaptive multi-engine solver for quantified boolean formulas. *Constraints An Int. J.*, 14(1):80–116, 2009.
35. Markus N. Rabe and Leander Tentrup. CAQE: A certifying QBF solver. In Roope Kaivola and Thomas Wahl, editors, *Formal Methods in Computer-Aided Design, FMCAD 2015, Austin, Texas, USA, September 27-30, 2015*, pages 136–143. IEEE, 2015.
36. Ronald L. Rivest. Learning decision lists. *Mach. Learn.*, 2(3):229–246, 1987.
37. Fabio Somenzi. CUDD: CU decision diagram package-release 2.4. 0. *University of Colorado at Boulder*, 2009.
38. Leander Tentrup. Non-prenex QBF solving using abstraction. In Nadia Creignou and Daniel Le Berre, editors, *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings*, volume 9710 of *Lecture Notes in Computer Science*, pages 393–401. Springer, 2016.
39. Ingo Wegener. *Branching Programs and Binary Decision Diagrams.* SIAM, 2000.
40. Lintao Zhang and Sharad Malik. Conflict driven learning in a quantified boolean satisfiability solver. In Lawrence T. Pileggi and Andreas Kuehlmann, editors, *Proceedings of the 2002 IEEE/ACM International Conference on Computer-aided Design, ICCAD 2002, San Jose, California, USA, November 10-14, 2002*, pages 442–449. ACM / IEEE Computer Society, 2002.