Towards Understanding the Complexity of CAQE: A Proof-Theoretic Analysis of Its Core Procedure

Benjamin Böhm
Friedrich Schiller University Jena, Germany

Olaf Beyersdorff
Friedrich Schiller University Jena, Germany

Abstract

CAQE (Clausal Abstraction for Quantifier Elimination) is currently the most successful algorithmic paradigm for solving Quantified Boolean Formulas (QBF) practice-wise, clearly dominating recent QBF competitions. While apparently being a very strong solver, not much is known about CAQE theory-wise. We propose a framework for formalising runs in the basic CAQE algorithm (where failed assumptions are not considered) as proofs in a proof system $\mathsf{CAQE}^{\mathsf{CORE}}$, which we use to analyse the algorithm proof-theoretically and provide methods for future work to separate it from optimized versions of CAQE that are used in practice.

We show that one can perform strategy extraction on the $\mathsf{CAQE}^{\mathsf{CORE}}$ proof system in such a way that strategy size serves as a lower bound for basic CAQE runs. Furthermore, we introduce a measure, which we call CAQE width, which not only acts as a lower bound on $\mathsf{CAQE}^{\mathsf{CORE}}$ proofs, but – with the quantifier depth as exponent – as an upper bound as well. Using this analysis, we prove that on QBFs of bounded quantifier complexity, both QCDCL (Quantified Conflict Driven Clause Learning, formalised as a proof system) and Q-resolution p-simulate $\mathsf{CAQE}^{\mathsf{CORE}}$ and are indeed strictly stronger.

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases QBF, CAQE, CDCL, resolution, proof complexity, simulations, lower bounds

Digital Object Identifier 10.4230/LIPIcs.SAT.2026.7

Funding *Olaf Beyersdorff*: Carl-Zeiss Foundation and DFG grant BE 4209/3-2.

Acknowledgements We thank Leroy Chew for asking a question at SAT'25 on the proof-theoretic strength of CAQE, which led to this work.

1 Introduction

Quantified Boolean Formulas (QBF) extend the satisfiability problem (SAT) by quantification over Boolean variables. This increases the complexity of the problem (from NP to PSPACE), the versatility of expressions (many problems are more succinctly and more naturally encoded into QBF), and the range of applications [34].

Different from SAT, where conflict-driven clause learning (CDCL) is the ruling algorithmic paradigm, there are a number of competing approaches to QBF solving. Quantified conflict-driven clause learning lifts the CDCL paradigm [26] to QBF [38, 13]. QCDCL can be further enhanced by dependency schemes [28] and dependency learning [27] and is implemented in a number of QBF solvers (Qute [29], DepQBF [25]). A second approach comes through expansion of universal variables in the counter-example guided abstraction refinement (CEGAR) framework, implemented in RAReQS [23]. CAQE (Clausal Abstraction for Quantifier Elimination) is a third algorithmic paradigm, introduced in 2015 [32, 36] and implemented in the eponymous solver. CAQE also utilises CEGAR, but in a fundamentally different way compared to expansion.

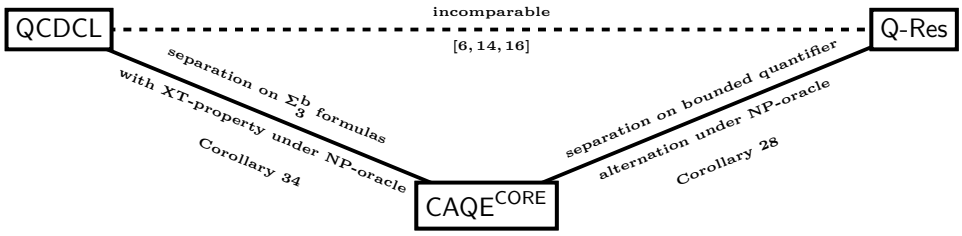

Figure 1 Hasse diagram of the simulation order of QCDCL, Q-Res and CAQE$^{\text{CORE}}$. Solid lines denote a simulation and exponential separation. The dashed line indicates mutual exponential separations.

With these various and seemingly orthogonal approaches, it is stunning that since 2017 versions of CAQE have won each QBF evaluation in the prenex CNF track, sometimes by quite a big margin [30, 31]. Experimental evaluations of different solvers as in QBF evaluations are an impactful driver for algorithmic progress. A systematic and theoretically concise comparison of different algorithmic solving techniques – in SAT, QBF and further fields – comes through proof complexity [18, 13, 12]. Proof complexity of QBF has been hugely advanced over the past decade (cf. [2] for a survey) and many theoretically clean and appealing QBF proof systems have been introduced together with proof methods to analyse them (e.g. [8, 5, 4, 10, 9, 1]).

When aiming to compare different solvers, we need proof systems tightly modelling solvers and such proof systems are typically quite different from rule-based calculi such as resolution [24, 37, 21], Frege or strong systems for proof logging [8, 22, 19]. A rigorous study of this kind has been performed on different versions of QCDCL and their relations to QBF resolution systems, unearthing surprising incomparability results – e.g. between standard QCDCL and Q-resolution [6] – as well as identifying potential improvements for solving (e.g. in out-of-order decisions [15, 16, 17]).

Such a tight theoretical model and comparison to other QBF solvers and proof systems is largely missing for the award-winning solver CAQE. An investigation into the proof-theoretic strength of CAQE was started in the paper [35], focussing on the comparison between CAQE and other CEGAR approaches such as RAReQS, and establishing their incomparability. However, the relation to QCDCL – arguably the other main QBF solving approach – has been open for the past decade.

Our results. We provide a theoretic proof-complexity analysis of the core procedure of CAQE and compare it to QCDCL and Q-Resolution, resulting in the situation depicted in Figure 1. We omit potential optimizations of CAQE such as failed assumptions as we plan to consider those in future work.

(a) As a first step, we tightly model the basic algorithmic paradigm of CAQE (without failed assumptions) as introduced in [32] as a proof system CAQE$^{\text{CORE}}$ such that runs of the basic CAQE algorithm can be translated into CAQE$^{\text{CORE}}$ proofs and conversely, each proof gives rise to a run of the basic algorithm (Theorem 6). We note that a proof system $\forall\mathsf{Red} + \mathsf{Res}$ was introduced in [35] that likewise allows to translate runs of the basic algorithm of CAQE into $\forall\mathsf{Red} + \mathsf{Res}$, but the converse direction fails (Theorem 8 and Theorem 24).

(b) In our main result we compare the core procedure of CAQE to QCDCL and show the possibly surprising result that QCDCL simulates this basic version of CAQE on bounded quantifier alternation (Corollary 35) and is exponentially stronger on selected formulas (Theorem 30). Our theoretical comparison utilises our new $\mathsf{CAQE}^{\mathsf{CORE}}$ proof system and the QCDCL proof systems developed in [16, 6].

As CAQE uses a massive number of SAT calls, we believe that conceptually the fairest comparison is by discounting SAT computations. This approach aligns with large parts of the QBF literature where propositional hardness is filtered out by allowing NP oracles in proofs [11] (whereby lower bounds measure "genuine" QBF hardness and become more challenging). For $\mathsf{CAQE}^{\mathsf{CORE}}$ this is fairly simple: we just count a SAT call as one step, whereas for QCDCL this is somewhat more intricate as QCDCL does not use SAT calls, but has all CDCL functionality "built in". We therefore define a QCDCL model where purely propositional parts of the traces are not counted (imagine, e.g. that QCDCL stumbles upon a hard propositional formula not containing any universal variables).

Further, we show that our proof system $\mathsf{CAQE}^{\mathsf{CORE}}$ is simulated by Q-Resolution (Corollary 29), again in the oracle model and on bounded quantifier alternation. While $\forall\mathsf{Red} + \mathsf{Res}$ is known to be simulated by Q-Resolution [35], this result does not discount SAT calls, which means that it does not imply our stronger simulation. Q-Resolution is a weak system and in particular known to be incomparable to QCDCL [16, 6, 14].

Conceptually, the simulation order in Figure 1 means that the core procedure of CAQE without failed assumptions is weaker than QCDCL and Q-Resolution from a proof-complexity perspective. Hence its practical outperformance appears to result from its clever and heavy SAT usage (cf. also the discussion in Section 9) as well as its use of optimisations such as failed assumptions, which we did not consider in this work.

c) Technically, we achieve our results by a characterisation of $\mathsf{CAQE}^{\mathsf{CORE}}$ proof size by a measure we call *CAQE width*, which yields lower as well as upper bounds on $\mathsf{CAQE}^{\mathsf{CORE}}$ proof size (Theorem 20, Proposition 22). This characterisation gives rise to exponential lower bounds and the separations as well as to efficient extraction of short decision lists (Theorem 27) and consequently the simulations.

Organisation. After setting up notation in Section 2 we explain the basic CAQE algorithm in Section 3 and formalise it as a proof system $\mathsf{CAQE}^{\mathsf{CORE}}$ in Section 4. In Section 5 we show that $\mathsf{CAQE}^{\mathsf{CORE}}$ is sound and complete and allows extraction of winning strategies. Section 6 introduces our main technical tool: $\mathsf{CAQE}^{\mathsf{CORE}}$ width that we use to obtain lower and upper bounds. In Section 7 we show that decision lists for winning strategies can be extracted from $\mathsf{CAQE}^{\mathsf{CORE}}$ proofs, yielding the simulation of $\mathsf{CAQE}^{\mathsf{CORE}}$ by Q-Res. Section 8 compares $\mathsf{CAQE}^{\mathsf{CORE}}$ and QCDCL. We conclude with a discussion in Section 9.

2 Preliminaries

Variables x and negated variables $\bar{x}$ are called *literals*. We denote the corresponding variable as $\mathrm{var}(x) := \mathrm{var}(\bar{x}) := x$. A *clause* is a disjunction of literals. A *unit clause* (ℓ) is a clause consisting of only one literal. The *empty clause* $(\bot)$ has zero literals. A *cube* is a conjunction of literals, sometimes viewed as a set of literals. We define a *unit cube* of a literal ℓ, denoted by $[\ell]$, and the *empty cube* $[\top]$ with "empty literal" $\top$. If C is a clause or a cube, we define $\mathrm{var}(C) := \{\mathrm{var}(\ell) : \ell \in C\}$.

An *assignment* σ of a set of variables V is a function $\sigma : V \to \{0,1\}$, which can also be interpreted as the set $\{\ell : \mathrm{var}(\ell) \in V \wedge \sigma(\ell) = 1\}$. We denote the set of all assignments of V as $\langle V \rangle$. A clause C is *satisfied* by σ if $C \cap \sigma \neq \varnothing$. A cube D is *falsified* by σ if $\neg D \cap \sigma \neq \varnothing$.

A clause C that is not satisfied by σ can be *restricted* by σ, defined as $C|_\sigma := \bigvee_{\ell \in C, \bar{\ell} \notin \sigma} \ell$. Similarly we can restrict a non-falsified cube D as $D|_\sigma := \bigwedge_{\ell \in D \setminus \sigma} \ell$. We can restrict a clause C to a set of variables V by $C|_V := \bigvee_{\ell \in C, \mathrm{var}(\ell) \in V} \ell$. We can restrict a cube D to a set of variables V by $D|_V := \bigwedge_{\ell \in D, \mathrm{var}(\ell) \in V} \ell$. A *CNF* (conjunctive normal form) is a conjunction of clauses and a *DNF* (disjunctive normal form) is a disjunction of cubes. A CNF (DNF) is *satisfied* (*falsified*) by σ if all its clauses (cubes) are satisfied (falsified) by σ.

We use the notation $\sigma \models \psi$ for a propositional formula ψ and assignment σ if σ satisfies ψ. We write $\varphi \models \psi$ for propositional formulas φ and ψ if each satisfying assignment of φ satisfies ψ.

A *QBF* (quantified Boolean formula) $\Phi = \mathcal{Q} \cdot \varphi$ consists of a propositional formula φ, called the *matrix*, and a *prefix* $\mathcal{Q}$. A *prefix* $\mathcal{Q} = Q_1 V_1 \dots Q_m V_m$ consists of non-empty and pairwise disjoint sets of variables $V_1, \dots, V_s$ and quantifiers $Q_1, \dots, Q_s \in \{\exists, \forall\}$ with $Q_i \neq Q_{i+1}$ for $i < m$.

A *QCNF* is a QBF $\Phi = \mathcal{Q} \cdot \varphi$ with CNF φ. We define $\mathrm{var}(\Phi) := \bigcup_{C \in \Phi} \mathrm{var}(C)$. We restrict a CNF φ by an assignment σ as $\varphi|_\sigma := \bigwedge_{C \in \varphi \text{ non-satisfied}} C|_\sigma$ and a QCNF $\Phi = \mathcal{Q} \cdot \varphi$ as $\Phi|_\sigma := \mathcal{Q}|_\sigma \cdot \varphi|_\sigma$, where $\mathcal{Q}|_\sigma$ is obtained by deleting all variables from $\mathcal{Q}$ that appear in σ.

Let Φ be a QCNF with the prefix $\forall U_0 \exists X_1 \forall U_1 \dots \exists X_m \forall U_m \exists X_{m+1}$, where U_0 is allowed to be empty. A *(universal) strategy* is a tuple of functions $s = (s_{U_0}, \dots, s_{U_m})$ with $s_{U_j} : \langle X_1 \cup \dots \cup X_j \rangle \longrightarrow \langle U_j \rangle$ for $j = 0, \dots, m$. If for each $\gamma = (\gamma_1, \dots, \gamma_{m+1}) \in \langle X_1 \cup \dots \cup X_{m+1} \rangle$ the assignment $\gamma \cup \bigcup_{j=0}^{m} s_{U_j}\left(\bigcup_{i=1}^{j} \gamma_i\right)$ falsifies the matrix of Φ, then s is called a *universal winning strategy*. We define the size of a universal (winning) strategy as $|s| := \sum_{j=0}^{m} |\operatorname{Image}(s_{U_j})|$. Existential strategies are defined dually.

A *proof system* (cf. [20]) for a language L is a poly-time computable function $\mathsf{P} : \{0,1\}^* \to \{0,1\}^*$ with $\operatorname{Image}(\mathsf{P}) = L$. A P-*proof* π of some $\Phi \in L$ is an element of $\{0,1\}^*$ such that $\mathsf{P}(\pi) = \Phi$. The inclusion $\operatorname{Image}(\mathsf{P}) \subseteq L$ is called *soundness*, while $\operatorname{Image}(\mathsf{P}) \supseteq L$ is the *completeness* of P. Here we consider the languages of unsatisfiable CNFs, false QCNFs and true QCNFs. A proof system P_1 *simulates* a proof system P_2 if there exists a poly-time computable function f such that for each π we have $\mathsf{P}_1(f(\pi)) = \mathsf{P}_2(\pi)$. If P_1 and P_2 simulate each other, we write $\mathsf{P}_1 \equiv_p \mathsf{P}_2$.

We often add an *NP oracle* to a proof system. While the details depend on the proof system, the idea is that P proofs can evaluate SAT calls for free [11].

Resolution (Res) is a proof system for unsatisfiable CNFs. A Res proof (or *refutation*) π of a CNF φ is a sequence of clauses $\pi = (C_1, \dots, C_k)$, where each C_i is either an axiom $C_i \in \varphi$ or derived by the *Resolution rule* $C_i = (C_a \vee C_b) \setminus \{x, \bar{x}\}$ with $x \in C_i$, $\bar{x} \in C_b$ and $a, b < i$.

Res can be lifted to a proof system Q-Resolution (Q-Res) [24] for false QCNFs by restricting the Resolution rule to existentially quantified literals x, disallowing the derivation of tautological clauses, and adding the *Reduction rule* $C_i = C_j \setminus \{u_1, \dots, u_n\}$, where $u_1, \dots, u_n$ are universal literals such that there are no existential literals in C_j that are quantified right of each u_h.

We can add an NP-oracle to Q-Res and obtain $\mathsf{Q}^{\mathsf{NP}}\text{-}\mathsf{Res}$ [4] by replacing the Resolution rule with the following *Oracle rule*: For some $\mathcal{G} \subseteq \{C_1, \dots, C_{i-1}\}$: $\bigwedge_{C \in \mathcal{G}} C^\exists \models C_i^\exists$ and $C^\forall$ is a subclause of $C_i^\forall$ for each $C \in \mathcal{G}$. Here $C^\exists$ (resp. $C^\forall$) denotes the existential (resp. universal) subclause of C.

3 CAQE: the basic algorithmic paradigm

CAQE, which stands for *Clausal Abstraction for Quantifier Elimination*, was introduced in 2015 [32]. It is a recursive algorithm that works on the first quantifier block of a QCNF and eliminates the quantifiers one by one. The idea is simple, but elegant: Suppose we have

a QNF $\Phi = \exists V \mathcal{Q} \cdot \varphi$, such that $\exists V$ is the first quantifier block and $\mathcal{Q}$ are the remaining quantifiers. Our goal is to find out if Φ is true or false. Obviously, Φ is true if and only if there exists an assignment $\alpha \in \langle V \rangle$ such that $\Phi|_\alpha = \mathcal{Q} \cdot \varphi|_\alpha$ is true, whereas the truth value of $\Phi|_\alpha$ can be checked recursively. If we can find at least one such α, we are done and we have proven the truth of Φ.

However, if Φ is a false formula, then we need to check that for all assignments $\alpha \in \langle V \rangle$ the formula $\Phi|_\alpha$ is indeed false. If we go through all of these assignments, we end up with a runtime exponential in $|V|$. The trick is now to pick specific assignments α that suffice to show the falsity of Φ. More precisely, we want to pick the "optimal" or "best" assignments α which satisfy the "highest number of" clauses from Φ. If these optimal assignments do not succeed to make Φ true, then neither will the assignments that are "worse". Let us demonstrate this on an example:

▶ **Example 1.** Let Φ be the QCNF with prefix $\exists x_1, x_2 \forall u_1, u_2 \exists y_1, y_2$ and the matrix

$$\begin{array}{lll} x_1 \vee u_1 \vee y_1 & x_1 \vee x_2 \vee u_2 \vee y_2 & \bar{y}_1 \vee \bar{y}_2 \\ \bar{x}_1 \vee \bar{u}_1 \vee y_1 & \bar{x}_1 \vee \bar{u}_2 \vee y_2 & \\ & \bar{x}_2 \vee \bar{u}_2 \vee y_2. & \end{array}$$

This formula is false since the strategy of the universal player is setting $u_1 \equiv x_1$ and $u_2 \equiv x_1 \vee x_2$. Now, there are four assignments for $\{x_1, x_2\}$ that would need to be tested: $\alpha_1 = (x_1 \mapsto 0, x_2 \mapsto 0)$, $\alpha_2 = (x_1 \mapsto 0, x_2 \mapsto 1)$, $\alpha_3 = (x_1 \mapsto 1, x_2 \mapsto 0)$ and $\alpha_4 = (x_1 \mapsto 1, x_2 \mapsto 1)$.

However, we can observe that α_3 already satisfies all the clauses that are satisfied by α_4. Even worse, α_4 satisfies less clauses than α_3. Hence, if we have shown that $\Phi|_{\alpha_3}$ is false, this also implicates that $\Phi|_{\alpha_4}$ must be false as well. Therefore, if we test α_3 before α_4, we can omit testing α_4 altogether. In other words, checking $\alpha_1, \alpha_2, \alpha_3$ suffices to prove the falsity of Φ.

An important aspect of CAQE is the direct inclusion of a SAT solver. Essentially, the task of the SAT solver, usually implemented as CDCL, is to return an assignment $\alpha_j \in \langle V \rangle$ such that α_j is not "worse" than or equal to one of the previously tested $\alpha_1, \ldots, \alpha_{j-1}$. Or more precisely, the SAT solver will return an α_j such that for each $i = 1, \ldots, j-1$ the assignment α_j satisfies at least one clause that is not satisfied by α_i. That also means that if the SAT solver cannot find such an α_j we can conclude that the formula is false.

The procedure for formulas $\Phi = \forall V \mathcal{Q} \cdot \varphi$ that start with a universal quantifier $\forall V$ is similar. The main difference is that the roles of universal and existential blocks are swapped and "best" or optimal assignments $\alpha \in \langle V \rangle$ are assignments that satisfy as few clauses as possible.

In the next section we will explain in further detail how CAQE compares the α in terms of optimality. A high-level overview of the functionality of CAQE is given in Figure 2.

4 Formalising basic CAQE as a proof system

Intuitively, each clause C_r of the given QCNF Φ is split into two parts: the top part and the bottom part. The top part, which we denote as $C_{r,1}$, consists of variables from the first quantifier block and contains all the information that we need to find all optimal assignments of that block. The bottom part, denoted as $C_{r,>1}$, contains the rest of the clause and is needed for the recursive call. Furthermore, $C_{r,1}$ and $C_{r,>1}$ are connected by an auxiliary

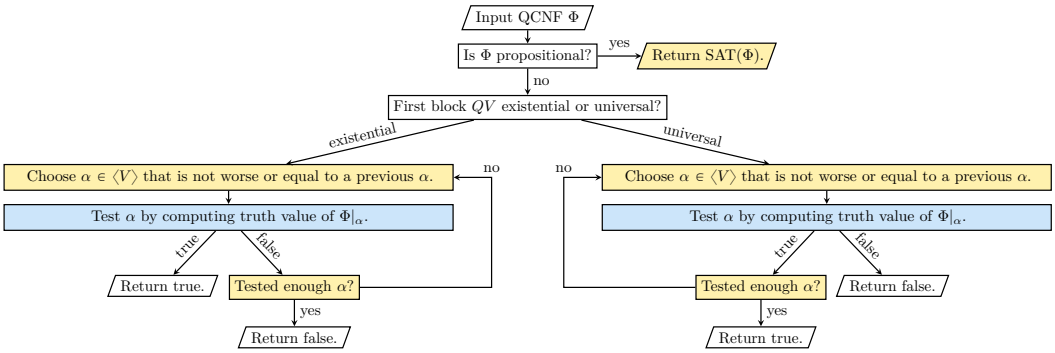

Figure 2 High-level flowchart of the basic CAQE algorithm. Yellow nodes are performed by the SAT solver. Blue nodes are performed recursively.

variable b_r (sometimes also called a *bottom variable*) which we need to encode whether or not an assignment has satisfied C_r on the first block. For each clause C_r we have a separate variable b_r, and we define B as the set of all auxiliary variables.

▶ **Definition 2.** *Let $\Phi := QV_1\mathcal{Q} \cdot C_1 \wedge \ldots \wedge C_k$ with clauses $C_1, \ldots, C_k$. We define auxiliary variables $B := \{b_1, \ldots, b_k\}$ with $B \cap \mathrm{var}(\Phi) = \varnothing$. If $Q = \exists$, we define*

$$C_{r,1} := \bigvee_{\substack{\ell \in C_r \\ \mathrm{var}(\ell) \in V_1}} \ell \vee b_r \qquad \text{and} \qquad C_{r,>1} := \bigvee_{\substack{\ell \in C_r \\ \mathrm{var}(\ell) \notin V_1}} \ell \vee \bar{b}_r.$$

We set

$$\Phi_{\exists V_1} := \bigwedge_{r=1}^{k} C_{r,1} \qquad \text{and} \qquad \Phi_{>1} := \mathcal{Q} \cdot \bigwedge_{r=1}^{k} C_{r,>1}.$$

Dually, if $Q = \forall$, we define $C_{r,1} := \bigwedge_{\substack{\ell \in C_r \\ \mathrm{var}(\ell) \in V_1}} \bar{\ell} \vee \bar{b}_r$ and $C_{r,>1} := \bigvee_{\substack{\ell \in C_r \\ \mathrm{var}(\ell) \notin V_1}} \ell \vee \bar{b}_r$ and set $\Phi_{\forall V_1} := \bigwedge_{r=1}^{k} C_{r,1}$ and $\Phi_{>1} := \mathcal{Q} \cdot \bigwedge_{r=1}^{k} C_{r,>1}$.

For an assignment $\beta \in \langle B \rangle$, we define

$$\Phi_{>1}|_\beta := \mathcal{Q} \cdot \bigwedge_{1 \leqslant r \leqslant k} C_{r,>1}|_\beta = \mathcal{Q} \cdot \bigwedge_{\substack{1 \leqslant r \leqslant k \\ \beta(b_r) = 1}} C_{r,>1}|_\beta.$$

Note that $\Phi_{\exists V_1}$ and $\Phi_{\forall V_1}$ are propositional formulas, while both $\Phi_{>1}|_\beta$ and $\Phi_{>1}$ are QCNFs.[1]

The auxiliary B-variables that are assigned by β indicate whether a clause is already satisfied in the first quantifier block, or if it might need to be satisfied using the remaining blocks. The assignment $b_r \mapsto 1$ encodes the information "C_r might need to be satisfied in the bottom part", while $b_r \mapsto 0$ means "C_r is already satisfied and does not need to be satisfied by the bottom part". More precisely, the propositional formula $\Phi_{\exists V_1}$ (resp. $\Phi_{\forall V_1}$) is given to a SAT solver which returns a satisfying assignment $\alpha \cup \beta \in \langle V_1 \cup B \rangle$. Such an assignment always exists. In the case $\exists V_1$, the variable b_r must be set to 1 if α does not satisfy C_r, otherwise it is allowed (but not forced) to be set to 0.

[1] In fact, we will almost never use $\Phi_{>1}$ without a β. Hence it is unimportant whether or not B is quantified.

Those clauses C_r for which b_r is set to 1 are precisely the clauses that remain in $\Phi_{>1}$, hence $\Phi_{>1}|_\beta$ contains at least all the clauses that are not satisfied by the corresponding α. That means if $\Phi_{>1}|_\beta$ is true, Φ turns out to be true as well.

In the case $\forall V_1$, the b_r are interpreted a bit differently. Here b_r needs to be set to 0 if α satisfies C_r, and b_r is allowed (but not forced) to be set to 1 if α does not satisfy C_r. Therefore all clauses that remain in $\Phi_{>1}|_\beta$ are clauses that are not satisfied by α. If $\Phi_{>1}|_\beta$ turns out to be false, then this also shows the falsity of Φ itself.

▶ **Example 3.** Let Φ be the formula from Example 1. Then $\Phi_{\exists x_1,x_2}$ consists of the clauses

$$x_1 \vee b_1 \quad x_1 \vee x_2 \vee b_2 \quad (b_3) \quad \bar{x}_1 \vee b_4 \quad \bar{x}_1 \vee b_5 \quad \bar{x}_2 \vee b_6$$

and $\Phi_{>1}$ is the QCNF with the prefix $\forall u_1, u_2 \exists y_1, y_2$ and the matrix

$$\bar{b}_1 \vee u_1 \vee y_1 \quad \bar{b}_2 \vee u_2 \vee y_2 \quad \bar{b}_3 \vee \bar{y}_1 \vee \bar{y}_2 \quad \bar{b}_4 \vee \bar{u}_1 \vee y_1 \quad \bar{b}_5 \vee \bar{u}_2 \vee y_2 \quad \bar{b}_6 \vee \bar{u}_2 \vee y_2.$$

So far we have only explained how to obtain a solution $\alpha \cup \beta$ for the first iteration when no other α, β are tested yet. Remember that we only want to test α that are not worse than previous ones. Because this concept of optimality does not primarily depend on α syntactically, we will instead use the β for that as β describes the effect of α on the clauses. In the $\exists V_1$ case, we want to make sure that for each previous α the new α satisfies at least one clause that the previous α did not. In other words, the new β has to flip at least one 1 from each previous β to 0. If $\beta_1, \ldots, \beta_{j-1}$ are the previously tested assignments and β_j is the new one, then β_j has to satisfy the formula $\bigwedge_{a=1}^{j-1} \left(\bigvee_{b \in \beta_a^{-1}(1)} \bar{b} \right)$ which we will simply add to $\Phi_{\exists V_1}$ by conjunction.

Analogously, in the case $\forall V_1$ we require that the new β flips at least one 0 from each previous β to 1. This can be enforced by $\bigwedge_{a=1}^{j-1} \left(\bigvee_{b \in \beta_a^{-1}(0)} b \right)$ which we add to $\Phi_{\forall V_1}$ by conjunction.[2]

Our goal to design a proof system that characterises the basic CAQE algorithm amounts to defining a structure that stores all the relevant information for each recursive call of the algorithm. We call these structures *slices*. A slice contains the following information: The current formula for that call, all α that were tested, all β that correspond to α, and references to further recursive calls for $\Phi_{>1}|_\beta$ (i.e. references to further slices) and a Res refutation ρ in case the propositional formula given to the SAT solver turns out to be unsatisfiable at some point. There are two exceptions: If Φ itself is an unsatisfiable propositional formula, a slice simply consists of a Res refutation[3], while for a satisfiable propositional formula a slice contains a satisfying assignment only. This is due to the fact that CAQE will merely call the SAT solver if only a propositional formula remains (i.e. when we reach the last quantifier block). We summarise this formally as follows:

[2] Note that we consider a potentially weaker system of CAQE in this paper. In practical CAQE, the algorithm benefits from a feature of modern SAT solvers called *failed assumptions*. To put it simply, such a SAT solver not only returns "unsat" in case the formula is unsatisfiable, but also returns a subset of clauses that suffice to achieve unsatisfiability. CAQE is then able to utilize this information when generating the constraints $\bigvee_{b \in \beta_a^{-1}(1)} \bar{b}$ (resp. $\bigvee_{b \in \beta_a^{-1}(0)} b$) by omitting the B-variables that were not necessary for the refutation. This obviously shortens those clauses in some cases and can lead to a significant improvement in runtime. We plan to add this optimization to our framework in future work and separate it from the core procedure, but for now we will assume that the SAT solver always returns the full set of clauses as failed assumptions.

[3] One could use any refutationally complete propositional proof system.

▶ **Definition 4.** *Let Φ be a QCNF with prefix $Q_1V_1Q_2V_2\dots Q_mV_m$, where V_i are disjoint sets of variables, $Q_i \in \{\exists, \forall\}$ and $Q_i \neq Q_{i+1}$ for all i. A* slice *S for Φ is either a tuple $S = (\Phi, (\alpha_1, \dots, \alpha_g), (\beta_1, \dots, \beta_g), (S_1, \dots, S_g), \rho)$ with $\alpha_j \in \langle V_1 \rangle$, $\beta_j \in \langle B \rangle$, ρ being a* Res *refutation (further specified in Definition 5) and S_j being further slices, or it is a pair $S = (\Phi, \pi)$ with π being a* Res *refutation of Φ, or it is a pair $S = (\Phi, \sigma)$ with σ being a satisfying assignment of Φ (the latter two cases only for propositional Φ).*

In the standard case $S = (\Phi, (\alpha_1, \dots, \alpha_g), (\beta_1, \dots, \beta_g), (S_1, \dots, S_g), \rho)$ we say that S refers *to the slices $S_1, \dots, S_g$. We set $|S| := g$ and* $\mathrm{Block}(S) := Q_1V_1$.

Note that we do not include computations of the SAT solver (i.e. ρ) in the size of our slices. This is because first we want to concentrate on genuine QBF-hardness where SAT calls come for free [11] and second we do not specify the SAT solver that is used.

Next we define $\mathsf{CAQE}^{\mathsf{CORE}}$ *proofs* and distinguish between refutations (for false formulas) and verifications (for true formulas). Intuitively, a $\mathsf{CAQE}^{\mathsf{CORE}}$ proof for a QCNF Φ is a sequence of slices, starting with the trivial cases of propositional formulas (one can consider these slices as axioms) and ending with a slice for Φ itself. From an algorithmic perspective, a proof is generated from back to front. The last slices of a proof correspond to the first quantifier blocks, while the first slices belong to the last quantifier block.

▶ **Definition 5.** *A* $\mathsf{CAQE}^{\mathsf{CORE}}$ proof *κ for a QCNF Φ is a sequence of slices $S_1, \dots, S_e$ such that S_e is a slice for Φ and each slice can only refer to slices that occur earlier in the sequence.*

$\kappa = (S_1, \dots, S_e)$ is called a $\mathsf{CAQE}^{\mathsf{CORE}}$ refutation *if one of the following is true:*

- *$S_e = (\Phi, \pi)$ and π is a* Res *refutation of Φ. We set $|S_e| := 1$ in this case.*
- *Φ is not propositional. $S_e = (\Phi, (\alpha_1, \dots, \alpha_g), (\beta_1, \dots, \beta_g), (S_{j_1}, \dots, S_{j_g}), \rho)$ with $|S_e| := g$ and ρ only exists if Φ starts with an existential quantifier.*[4] *Let Q_1V_1 be the first quantifier block. Then:*
  - *If $Q_1 = \exists$, then $\Phi_{\exists V_1} \wedge \bigwedge_{a=1}^{g} \left(\bigvee_{b \in \beta_a^{-1}(1)} \bar{b} \right)$ is unsatisfiable with a* Res *refutation ρ.*
    - *Each $\alpha_j \cup \beta_j$ satisfies $\Phi_{\exists V_1} \wedge \bigwedge_{a=1}^{j-1} \left(\bigvee_{b \in \beta_a^{-1}(1)} \bar{b} \right)$ for $j = 1, \dots, g$.*
    - *$\Phi_{>1}|_{\beta_1}, \dots, \Phi_{>1}|_{\beta_g}$ are false with $S_{j_1}, \dots, S_{j_g}$ being the slices for them.*
  - *If $Q_1 = \forall$, then*
    - *Each $\alpha_j \cup \beta_j$ satisfies $\Phi_{\forall V_1} \wedge \bigwedge_{a=1}^{j-1} \left(\bigvee_{b \in \beta_a^{-1}(0)} b \right)$ for $j = 1, \dots, g$.*
    - *$\Phi_{>1}|_{\beta_1}, \dots, \Phi_{>1}|_{\beta_{g-1}}$ are true with $S_1, \dots, S_{j_{g-1}}$ being the slices for them. Hence, each $\kappa' := (S_{j_1}, \dots, S_{j_h})$ is a* $\mathsf{CAQE}^{\mathsf{CORE}}$ *verification of $\Phi_{>1}|_{\beta_h}$ for $h = 1, \dots, g-1$, and*
    - *$\Phi_{>1}|_{\beta_g}$ is false, with S_{j_g} being the slice for it. Hence, $\kappa' := (S_1, \dots, S_{j_g})$ is a* $\mathsf{CAQE}^{\mathsf{CORE}}$ *refutation of $\Phi_{>1}|_{\beta_g}$.*

We set $|\kappa| := \sum_{i=1}^{e} |S_i|$. $\mathsf{CAQE}^{\mathsf{CORE}}$ *verifications are defined analogously.*

It is straightforward to check that $\mathsf{CAQE}^{\mathsf{CORE}}$ proofs tightly model the basic algorithmic CAQE paradigm without failed assumptions.

▶ **Theorem 6.** *From each CAQE run without failed assumptions (i.e. the first* `solve` *procedure from [32]) we can efficiently extract a* $\mathsf{CAQE}^{\mathsf{CORE}}$ *proof. Conversely, each* $\mathsf{CAQE}^{\mathsf{CORE}}$ *proof can be efficiently transformed into a CAQE run.*

[4] We can omit ρ or treat it as a placeholder in case Φ starts with a universal quantifier.

$\mathsf{CAQE}^{\mathsf{CORE}}$ refutations only need to contain $\mathsf{CAQE}^{\mathsf{CORE}}$ verifications if the SAT solver did not succeed to immediately find a β such that $\Phi_{>1}|_\beta$ is false. In most of our results we only need to consider a shortest $\mathsf{CAQE}^{\mathsf{CORE}}$ refutation, for which we can assume that slices for universal blocks only contain one assignment (i.e. the correct assignment). That means for such results we can assume that $\mathsf{CAQE}^{\mathsf{CORE}}$ refutations do not contain verifications.

Next we recall the $\forall\mathsf{Red} + \mathsf{Res}$ proof system introduced in [35] to analyse CEGAR-type solving. We will show that $\forall\mathsf{Red} + \mathsf{Res}$, after adding an NP oracle, simulates $\mathsf{CAQE}^{\mathsf{CORE}}$ proofs, but is exponentially stronger (Theorem 24).

▶ **Definition 7** ($\forall\mathsf{Red} + \mathsf{Res}$, [35]). *Let $\Phi = Q_1V_1 \dots Q_mV_m \cdot C_1 \wedge \dots \wedge C_k$ with $Q_i \in \{\exists, \forall\}$ for each i and $Q_j \neq Q_{j+1}$ for each j. A $\forall\mathsf{Red} + \mathsf{Res}$ proof of Φ is a sequence of sets $\mathcal{P}^a$, where $\mathcal{P} \subseteq \{1, \dots, k\}$ and $a \in \{0, 1, \dots, m\}$, derived by the following three rules: the resolution rule $\frac{\mathcal{P}_1^a \dots \mathcal{P}_g^a \quad \pi}{(\mathcal{P}_1 \cup \dots \cup \mathcal{P}_g)^{a-1}}$ where $Q_a = \exists$ and π is a Res refutation of $\bigwedge_{h=1}^{g} \bigvee_{q \in \mathcal{P}_h} C_q|_{V_a}$, the reduction rule $\frac{\mathcal{P}^a}{\mathcal{P}^{a-1}}$ where $Q_a = \forall$ and there is no $u \in V_a$ with $u \in \bigvee_{q \in \mathcal{P}} C_q|_{V_a}$ and $\bar{u} \in \bigvee_{q \in \mathcal{P}} C_q|_{V_a}$, and the axiom rule $\frac{}{\{q\}^m}$ with $q \in \{1, \dots, k\}$.*

A $\forall\mathsf{Red} + \mathsf{Res}$ proof is called a $\forall\mathsf{Red} + \mathsf{Res}$ refutation if we derived $\mathcal{P}^0$ for some $\mathcal{P} \subseteq \{1, \dots, k\}$. The size of a $\forall\mathsf{Red} + \mathsf{Res}$ proof is the number of sets $\mathcal{P}^a$ and the sum of the sizes of all Res refutations π used in the proof. We call $\forall\mathsf{Red} + \mathsf{Res}^{\mathsf{NP}}$ the version of $\forall\mathsf{Red} + \mathsf{Res}$ in which the π are not counted.

▶ **Theorem 8.** *Let κ be a $\mathsf{CAQE}^{\mathsf{CORE}}$ refutation of Φ. Then there exists a $\forall\mathsf{Red} + \mathsf{Res}^{\mathsf{NP}}$ refutation ν with $|\nu| \leqslant |\kappa|$. Hence, $\forall\mathsf{Red} + \mathsf{Res}^{\mathsf{NP}}$ simulates $\mathsf{CAQE}^{\mathsf{CORE}}$.*

We would like to comment that it is unclear whether $\forall\mathsf{Red} + \mathsf{Res}$ simulates the non-oracle version of $\mathsf{CAQE}^{\mathsf{CORE}}$, or even the general algorithmic CAQE paradigm. While the Res refutations π in $\forall\mathsf{Red} + \mathsf{Res}$ proofs somehow correspond to the refutations ρ generated by a SAT solver during a $\mathsf{CAQE}^{\mathsf{CORE}}$ run, there might not be a polynomial-time transformation between those two. The reason is that the latter uses the auxiliary B-variables from $\mathsf{CAQE}^{\mathsf{CORE}}$, while the former does not. Resolving over those B-variables first would effectively eliminate them, but this might result in an exponential blow-up in the proof size. On the other hand, redefining $\forall\mathsf{Red} + \mathsf{Res}$ by allowing B-variables in the refutations π could cause other problems: In [35] it was proven that $\forall\mathsf{Red} + \mathsf{Res}$ is equivalent to $\mathsf{level\text{-}ordered\ Q\text{-}Res}$. For the first direction, one can argue that all π simply have to be stuck together in order to obtain a $\mathsf{level\text{-}ordered\ Q\text{-}Res}$ refutation. Allowing B-variables in π could render this simulation impossible.

We avoid these potential issues by adding an NP oracle and disregarding Res refutations both for $\forall\mathsf{Red} + \mathsf{Res}$ as well as for $\mathsf{CAQE}^{\mathsf{CORE}}$. If the simulation of $\mathsf{CAQE}^{\mathsf{CORE}}$ by $\forall\mathsf{Red} + \mathsf{Res}$ indeed holds without NP oracles, then the result in Theorem 8 would follow trivially (as well as the simulation of $\mathsf{CAQE}^{\mathsf{CORE}}$ by $\mathsf{Q}^{\mathsf{NP}}\text{-}\mathsf{Res}$). But whether or not this holds remains open.

5 Strategy extraction and soundness/completeness of CAQE

In this section we will show how to perform strategy extraction on $\mathsf{CAQE}^{\mathsf{CORE}}$ proofs. Strategy extraction is a common technique not only for obtaining lower bounds, but also for proving the soundness of a proof system. More precisely, from a given $\mathsf{CAQE}^{\mathsf{CORE}}$ refutation we extract a universal winning strategy, whereas from a $\mathsf{CAQE}^{\mathsf{CORE}}$ verification we extract an existential winning strategy. This results in strategy size as a lower bound for the size of $\mathsf{CAQE}^{\mathsf{CORE}}$ proofs.

From now on, if not otherwise specified, we will always assume that Φ is a QCNF with prefix $\forall U_0 \exists X_1 \forall U_1 \exists X_2 \forall U_2 \dots \exists X_m \forall U_m \exists X_{m+1}$, where U_0 is allowed to be empty. For the sake of readability, we will denote $X := X_1 \cup \dots \cup X_{m+1}$ and $U := U_0 \cup \dots \cup U_m$.

▶ **Proposition 9.** *From a* $\mathsf{CAQE}^{\mathsf{CORE}}$ *refutation* κ *of a QCNF* Φ *we can efficiently extract a universal winning strategy* s *such that* $|\operatorname{Image}(s_{U_j})| \leqslant |\{S \in \kappa : \operatorname{Block}(S) = U_j\}| \leqslant |\kappa|$ *for* $j = 0, \dots, m$*. By summing over all universal blocks, we even obtain* $|s| \leqslant |\kappa|$*.*

Proof Sketch. Let $\kappa = (S_1, \dots, S_e)$ and $\gamma \in \langle X \rangle$ be an assignment of the existential variables. We start with the last slice S_e, which is a slice for Φ, and construct a path upwards by always choosing a suitable slice which the current slice refers to. If the current slice $S = (\Phi, (\alpha_1, \dots, \alpha_g), (\beta_1, \dots, \beta_g), (S_{j_1}, \dots, S_{j_g}), \rho)$ is a slice for a formula starting with an existential quantifier block, we find an α_i that satisfies at least the clauses satisfied by γ on the first block and choose the corresponding slice S_{j_i} to move on. If S is a slice for a formula starting with a universal block, we choose α_g (the last assignment) as our universal response and proceed with the corresponding slice S_{j_g}. At some point we end with an unsatisfiable propositional formula in variables from the last block $\exists X_{m+1}$. This shows that we indeed constructed a winning strategy for the universal player because the matrix got falsified. ◀

For true QCNFs and $\mathsf{CAQE}^{\mathsf{CORE}}$ verifications, an analogous result holds. As a corollary we obtain soundness and completeness of the proof system.

▶ **Corollary 10.** $\mathsf{CAQE}^{\mathsf{CORE}}$ *is a sound proof system both on false and true QCNFs.*

6 Lower and upper bounds for proof size via CAQE width

Next we want to prove both lower and upper bounds for the size of $\mathsf{CAQE}^{\mathsf{CORE}}$ proofs for which we introduce a measure called *CAQE width*. The main idea is as follows: We already know that CAQE only tests assignments if they are not worse than previously tested ones. This means that we can minimise the number of assignments to be tested by only picking optimal ones. Hence, as an upper bound we can count the number of these optimal assignments. On the other hand, since we need to test at least all optimal assignments, this also yields a lower bound.

The intricate part is to generalise the notion of optimal assignments to formulas with more than one quantifier alternation, as we can only check optimality on the first block. To do this, we define *optimal moves* on the first variable block, and *optimal chains* as a sequence of optimal moves for several quantifier alternations.

▶ **Definition 11** (optimal moves). *Let* Φ *be a QCNF with matrix* $C_1 \wedge \dots \wedge C_k$ *such that its prefix starts with the quantifier block* QV*. In case* $Q = \exists$*, an assignment* $\alpha \in \langle V \rangle$ *is called an* optimal move *if and only if the following holds: For each* $\alpha' \in \langle V \rangle$ *and each clause* C_r *with* $\alpha \not\models C_r$ *and* $\alpha' \models C_r$ *there exists a clause* C_q *with* $\alpha \models C_q$ *and* $\alpha' \not\models C_q$*. In other words, there is no assignment* $\alpha' \in \langle V \rangle$ *such that* α' *satisfies all clauses satisfied by* α*, plus at least one more clause.*

Analogously, if $Q = \forall$ *an assignment* $\alpha \in \langle V \rangle$ *is called an* optimal move *if and only if the following holds: For each* $\alpha' \in \langle V \rangle$ *and each clause* C_r *with* $\alpha \models C_r$ *and* $\alpha' \not\models C_r$ *there is a clause* C_q *with* $\alpha \not\models C_q$ *and* $\alpha' \models C_q$*. In other words, there is no assignment* $\alpha' \in \langle V \rangle$ *such that* α *satisfies all clauses that are satisfied by* α'*, plus at least one more clause.*

Optimal moves can now be generalised to optimal move chains. An important aspect of optimal move chains is that they always rely on a given winning strategy s. This is because we need to consider quantifier alternations and we need to know how the variables are assigned that are quantified between the blocks on which we want to detect optimal moves.

▶ **Definition 12.** *Let s be a universal winning strategy for a QCNF with the prefix*

$$\forall U_0 \exists X_1 \forall U_1 \dots \exists X_m \forall U_m \exists X_{m+1},$$

where U_0 is allowed to be empty, and let $\gamma = (\gamma_1, \dots, \gamma_m) \in \langle W \rangle$ be some assignment with $X_1 \cup \dots \cup X_m \subseteq W$. For $j = 0, \dots, m$ we define $\mathrm{game}(s, \gamma, U_j)$ as the assignment that is created during the two-player game up to the block $\forall U_j$ while following the strategy s and assignment γ. Analogously, we define $\mathrm{game}(s, \gamma, X_j)$ as the assignment for the game up to block $\exists X_j$.

The case for existential winning strategies is defined analogously.

▶ **Definition 13** (optimal chains). *Let Φ be a false QCNF and s be a universal winning strategy s for Φ. An assignment $(\eta_1, \dots, \eta_j) \in \langle X_1 \cup \dots \cup X_j \rangle$ for $j \leqslant m$ is called an* optimal chain up to block j with respect to s *if and only if each $\eta_i \in \langle X_i \rangle$ is an optimal move for $\Phi|_{\mathrm{game}(s, \eta_1 \cup \dots \cup \eta_{i-1}, U_{i-1})}$ for each $i = 1, \dots, j$.*[5]

Analogously, in the case where Φ is true and s is an existential winning strategy, an assignment $(\eta_0, \dots, \eta_j) \in \langle U_0 \cup \dots \cup U_j \rangle$ for $j \leqslant m$ is called an optimal chain up to block j with respect to s *if and only if each $\eta_i \in \langle U_i \rangle$ is an optimal move for $\Phi|_{\mathrm{game}(s, \eta_0 \cup \dots \cup \eta_{i-1}, X_i)}$ for each $i = 0, \dots, j$.*[6]

We denote the set of all optimal chains up to block j with respect to s as $K_j(\Phi, s)$. Furthermore, we set $K(\Phi, s) := K_m(\Phi, s)$.

We recall the definition of maximal and minimal elements with respect to the subset relation. Let $\mathfrak{A}$ be a set of pairwise distinct sets. Then we define $\max_{\subseteq}(\mathfrak{A}) := \{M \in \mathfrak{A} : \nexists N \in \mathfrak{A} : M \subsetneq N\}$ and $\min_{\subseteq}(\mathfrak{A}) := \{M \in \mathfrak{A} : \nexists N \in \mathfrak{A} : N \subsetneq M\}$.

▶ **Definition 14** (CAQE width). *Let Φ be a QCNF. If Φ is false, we define*

$$\mathrm{cw}^-(\Phi) := \min_s \max_{1 \leqslant j \leqslant m} \max_{\tilde{\gamma} \in K_{j-1}(\Phi, s)} |\max_{\subseteq}\{\{r : \mathrm{game}(s, \gamma, X_j) \models C_r\} : \tilde{\gamma} \subseteq \gamma \in \langle X \rangle\}|$$

and if Φ is true we define

$$\mathrm{cw}^+(\Phi) := \min_s \max_{0 \leqslant j \leqslant m} \max_{\tilde{\gamma} \in K_{j-1}(\Phi, s)} |\min_{\subseteq}\{\{r : \mathrm{game}(s, \gamma, U_j) \models C_r\} : \tilde{\gamma} \subseteq \gamma \in \langle U \rangle\}|$$

as the CAQE width *of Φ.*[7]

Computing the CAQE width can be interpreted as a game between two players: the strategist and the optimizer. In the case where Φ is false, the strategist controls the universal variables and the optimizer controls the existential variables. The task of the strategist is to pick a universal winning strategy for Φ at the beginning and assign the variables of Φ along

[5] Note that it is not necessary to consider optimal moves on the block X_{m+1} because CAQE$^{\mathrm{CORE}}$ would simply use a SAT solver at this point. In general, we are only interested in optimal moves if at least one quantifier block with opposite quantification follows.

[6] For $i = 0$ we simply set $X_i = \varnothing$.

[7] In the case $j = 0$ we simply set $K_{j-1}(\Phi, s) := \varnothing$.

that strategy throughout the whole game. On the other hand, before assigning some block X_j the optimizer counts the number of optimal moves on that block under the side condition that different moves that have the same effect[8] on the formula only count once. Let p_j be that number. Then the optimizer chooses an optimal move to play it on the formula. The goal of the optimizer is to maximise $\max_j p_j$ by assigning X suitably, while the goal of the strategist is to minimise $\max_j p_j$ by choosing the best strategy s at the beginning. If both players play optimal, we end up with $\max_j p_j = \mathrm{cw}^-(\Phi)$.

The case where Φ is true is similar. There the strategist controls the existential variables and picks an existential winning strategy, whereas the optimizer assigns the universal variables. At the end we obtain $\max_j p_j = \mathrm{cw}^+(\Phi)$.

▶ **Example 15.** Let Φ be the QCNF from Example 1. Then we have $\mathrm{cw}^-(\Phi) = 3$.

There are formulas for which we need to count all assignments for CAQE width. One such example are the equality formulas, in which each assignment of the first block is optimal by symmetry.

▶ **Definition 16** (equality formulas [3, 17])**.** *The false QCNF* $\mathtt{Eq}_n$ *consists of the prefix* $\exists x_1, \ldots, x_n\ \forall u_1, \ldots, u_n \exists t_1, \ldots, t_n$ *and the matrix* $x_i \vee u_i \vee t_i,\ \bar{x}_i \vee \bar{u}_i \vee t_i,\ \bar{t}_1 \vee \ldots \vee \bar{t}_n$ *for* $i = 1, \ldots, n$.

▶ **Proposition 17.** *It holds* $\mathrm{cw}^-(\mathtt{Eq}_n) = 2^n$.

Proof. Let $\gamma_1, \gamma_2 \in \langle\{x_1, \ldots, x_n\}\rangle$ be two assignments with $\gamma_1 \neq \gamma_2$. Then there exists an i with $\gamma_1(x_i) \neq \gamma_2(x_i)$. Hence, $x_i \vee u_i \vee t_i$ is satisfied by γ_1 but not by γ_2, and $\bar{x}_i \vee \bar{u}_i \vee t_i$ is satisfied by γ_2 but not by γ_1, or vice versa. That means all assignments of the first block are optimal and we obtain $\mathrm{cw}^-(\mathtt{Eq}_n) = |\langle\{x_1, \ldots, x_n\}\rangle| = 2^n$. ◀

Next we define *fruitions*. These are projections of assignments that tell us which clauses of a formula Φ are satisfied by these assignments. As optimal chains, they are defined across several quantifier alternations and hence need a fixed winning strategy s as input. Intuitively, while assignments are purely syntactic, their fruitions encode the effect of their assignments on the formula.

▶ **Definition 18** (fruitions)**.** *Let* Φ *be a QCNF. If* Φ *is false and* s *is a universal winning strategy, then for each assignment* $\sigma = (\sigma_1, \ldots, \sigma_j) \in \langle X_1 \cup \ldots \cup X_j\rangle$ *with* $j \leqslant m$ *we call* $\mathrm{fru}(\sigma) := (I_1, \ldots, I_j)$ *the* fruition of σ *with* $I_i := \{r : \mathrm{game}(s, \sigma, X_i) \models C_r\}$*, for* $i = 1, \ldots, j$.

Analogously, if Φ *is true and* s *is an existential winning strategy, then for each assignment* $\sigma = (\sigma_0, \ldots, \sigma_j) \in \langle U_0 \cup \ldots \cup U_j\rangle$ *with* $j \leqslant m$ *we call* $\mathrm{fru}(\sigma) := (I_0, \ldots, I_j)$ *the* fruition of σ *with* $I_i := \{r : \mathrm{game}(s, \sigma, U_i) \models C_r\}$*, for* $i = 0, \ldots, j$.

We denote $F_j(\Phi, s) := \{\mathrm{fru}(\sigma) : \sigma \in K_j(\Phi, s)\}$ *and set* $F(\Phi, s) := F_m(\Phi, s)$*. We will sometimes refer to the elements of* $F_j(\Phi, s)$ *as* optimal fruitions (up to block j with respect to s).

It is possible to give an alternative definition of the CAQE width by using optimal fruitions instead of optimal chains. More importantly, this shows a direct connection between the CAQE width and the total number of optimal fruitions of a formula.

[8] We will denote this as the *fruition* later.

▶ **Lemma 19.** *For a false QCNF as in Definition 14 it holds*

$$\mathrm{cw}^-(\Phi) = \min_s \max_{1 \leqslant j \leqslant m} \max_{(I_1,\ldots,I_{j-1}) \in F_{j-1}(\Phi,s)} \left|\{I : (I_1,\ldots,I_{j-1},I) \in F_j(\Phi,s)\}\right|.$$

Analogously, for a true QCNF it holds

$$\mathrm{cw}^+(\Phi) = \min_s \max_{0 \leqslant j \leqslant m} \max_{(I_1,\ldots,I_{j-1}) \in F_{j-1}(\Phi,s)} \left|\{I : (I_1,\ldots,I_{j-1},I) \in F_j(\Phi,s)\}\right|.$$

In particular, we have $|F(\Phi,s)| \leqslant \mathrm{cw}^-(\Phi)^m$ *for false QCNFs,* $|F(\Phi,s)| \leqslant \mathrm{cw}^+(\Phi)^{m+1}$ *for true QCNFs, where* s *is the winning strategy that characterises the* CAQE$^{\mathsf{CORE}}$ *width.*[9]

We can now prove an upper bound for CAQE$^{\mathsf{CORE}}$ proof size using CAQE width. The main idea is to construct CAQE$^{\mathsf{CORE}}$ proofs by always choosing optimal moves for testing, and their number is bounded by the CAQE width. As a side result, we also prove the completeness of CAQE$^{\mathsf{CORE}}$ by showing how refutations or verifications can always be constructed.

▶ **Theorem 20.** *For each false QCNF* Φ *there exists a* CAQE$^{\mathsf{CORE}}$ *refutation of size* $\mathcal{O}(\mathrm{cw}^-(\Phi)^m)$.

Proof Sketch. Let s be the strategy that characterises $\mathrm{cw}^-(\Phi)$. We create a CAQE$^{\mathsf{CORE}}$ proof by constructing the slices inductively, starting with a slice for Φ. If we construct a slice S for a formula Ψ beginning with an existential quantifier block, it suffices to collect all optimal moves $\alpha_1,\ldots,\alpha_g$ into S for that first block, leading to a slice

$$S = (\Psi, (\alpha_1,\ldots,\alpha_g), (\beta_1,\ldots,\beta_g), (S_{j_1},\ldots,S_{j_g}), \rho),$$

where slices S_{j_i} are constructed inductively. The number of optimal moves is bounded by $\mathrm{cw}^-(\Phi)$, hence $g \leqslant \mathrm{cw}^-(\Phi)$. If Ψ starts with a universal quantifier block, we choose α_1 for the first block according to our strategy s and obtain a slice $S = ((\alpha_1),(\beta_1),(S_{j_1}))$, where S_{j_1} is constructed inductively. At the end, following the strategy up to the last block, we obtain an unsatisfiable formula Ψ. Hence we can find a Res refutation π of Ψ and set the corresponding slice to $S = (\Psi,\pi)$. ◀

An analogous result holds for true QBFs, from which we conclude:

▶ **Corollary 21.** CAQE$^{\mathsf{CORE}}$ *is complete both on false and true QCNFs.*

Next we show that CAQE width can be used for lower bounds as well. The idea is simple: As the CAQE width somewhat counts the number of optimal assignments throughout the formula, and since CAQE only terminates once no more assignments can be tested that are not worse than the previously tested ones, we need to test at least the optimal assignments before returning the truth value.

▶ **Proposition 22.** *For each* CAQE$^{\mathsf{CORE}}$ *refutation* κ *of a false QCNF* Φ *we have* $\mathrm{cw}^-(\Phi) \leqslant |\kappa|$. *Analogously, for each true QCNF* Φ *and* CAQE$^{\mathsf{CORE}}$ *verification* κ *of* Φ *we have* $\mathrm{cw}^+(\Phi) \leqslant |\kappa|$.

Proof Sketch. For a false Φ, we consider the universal winning strategy s_κ that can be extracted from κ by Proposition 9. We compute the maximal number w of optimal moves by maximising over j and $\tilde{\gamma}$ as in the computation of $\mathrm{cw}^-(\Phi)$, but now we fix the strategy s_κ instead of minimising over s. We obtain $\mathrm{cw}^-(\Phi) \leqslant w \leqslant |\kappa|$. The case for true QCNFs works analogously. ◀

[9] If the first block for a true formula is existential, we even get $|F(\Phi,s)| \leqslant \mathrm{cw}^+(\Phi)^m$.

We now define new QBFs $\mathtt{Copy}_n$ consisting of n disjoint copies of $\mathtt{Eq}_1$:

▶ **Definition 23.** *The false QCNF* $\mathtt{Copy}_n$ *has prefix* $\exists X \forall U \exists T$ *with* $X = \{x_1, \ldots, x_n\}$, $U = \{u_1, \ldots, u_n\}$ *and* $T = \{t_1, \ldots, t_n\}$ *and clauses* $x_i \vee u_i \vee t_i$, $\bar{x}_i \vee \bar{u}_i \vee t_i$ *and* $(\bar{t}_i)$ *for* $i = 1, \ldots, n$.

These QBFs witness an exponential separation between $\mathsf{CAQE}^{\mathsf{CORE}}$ and $\forall\mathsf{Red} + \mathsf{Res}$:

▶ **Theorem 24.** $\mathtt{Copy}_n$ *needs exponential-size* $\mathsf{CAQE}^{\mathsf{CORE}}$ *refutations, but has constant-sizes* $\forall\mathsf{Red} + \mathsf{Res}$ *(and* $\forall\mathsf{Red} + \mathsf{Res}^{\mathsf{NP}}$*) refutations.*

Consequently, $\mathsf{CAQE}^{\mathsf{CORE}}$ is simulated by $\forall\mathsf{Red} + \mathsf{Res}^{\mathsf{NP}}$, but exponentially weaker. Hence the system $\forall\mathsf{Red} + \mathsf{Res}$ is too strong to capture the basic version of CAQE without failed assumptions. Whether $\forall\mathsf{Red} + \mathsf{Res}$ correctly captures the strength of full CAQE with failed assumptions remains open.

7 CAQE vs Q-Res: CAQE width and unified decision lists

In this section we want to show that $\mathsf{CAQE}^{\mathsf{CORE}}$ is simulated by $\mathsf{Q}^{\mathsf{NP}}_{-}\mathsf{Res}$. While versions of Q-Res (without the NP oracle) underly proof systems for QCDCL [15], the connection between $\mathsf{Q}^{\mathsf{NP}}_{-}\mathsf{Res}$ and $\mathsf{CAQE}^{\mathsf{CORE}}$ appears nontrivial. This is why we will make use of *decision lists*, which represent strategies and even characterise $\mathsf{Q}^{\mathsf{NP}}_{-}\mathsf{Res}$ on bounded quantifier alternation [4].

▶ **Definition 25** (decision lists, [33, 4])**.** *A* multi-output decision list $\mathcal{M}$ *from* X *to* U*, where* X *and* U *are disjoint sets of variables, is a sequence of pairs* $(D_1, \mu_1), \ldots, (D_z, \mu_z)$ *where*

- D_i *are cubes with* $\mathrm{var}(D_i) \subseteq X$ *and* $\bigvee_{i=1}^{z} D_i \equiv \top$,
- $\mu_i \in \langle U \rangle$

for $i = 1, \ldots, z$*. The size of* $\mathcal{M}$ *is defined as* $|\mathcal{M}| := z$.

$\mathcal{M}$ *computes the function from* $\langle X \rangle$ *to* $\langle U \rangle$ *that maps each* $\gamma \in \langle X \rangle$ *to* μ_{i_γ}*, where* $i_\gamma := \min\{i : \gamma \vDash D_i\}$*. If* $\mathcal{M}$ *computes a winning strategy for a QCNF* Φ*, then* $\mathcal{M}$ *is called a* unified decision list (UDL) *for* Φ*. In that case, we call* $\mathcal{M}$ *a* UDL *refutation of* Φ*. We will use the name of the UDL* $\mathcal{M}$ *to refer to the computed function.*

▶ **Theorem 26** ([4])**.** $\mathsf{Q}^{\mathsf{NP}}_{-}\mathsf{Res} \equiv_p \mathsf{UDL}$ *on bounded quantifier alternation.*

Instead of proving that $\mathsf{Q}^{\mathsf{NP}}_{-}\mathsf{Res}$ simulates $\mathsf{CAQE}^{\mathsf{CORE}}$ directly, we show that UDLs are bounded by the CAQE width.

▶ **Theorem 27.** *Let* Φ *be a false QCNF. Then there exists a UDL* $\mathcal{M}$ *for* Φ *with* $|\mathcal{M}| \leqslant \mathrm{cw}^-(\Phi)^m$.

Proof. Let s be a universal winning strategy for Φ such that

$$\mathrm{cw}^-(\Phi) := \max_{1 \leqslant j \leqslant m} \max_{\tilde{\gamma} \in K_{j-1}(\Phi, s)} |\max_{\subseteq}\{\{r : \mathrm{game}(s, \gamma, X_j) \vDash C_r\} : \tilde{\gamma} \subseteq \gamma \in \langle X \rangle\}|.$$

For each $\gamma = (\eta_1, \ldots, \eta_m) \in K(\Phi, s)$ we obtain its fruition $\mathrm{fru}(\gamma) = (I_1, \ldots, I_m)$, as well as the set of all fruitions $F(\Phi, s)$. Note that we have $|F(\Phi, s)| \leqslant \mathrm{cw}^-(\Phi)^m$ by Lemma 19. We will now construct another strategy $\tilde{s} = (\tilde{s}_{U_0}, \ldots, \tilde{s}_{U_m})$ as follows: Let $\sigma = (\sigma_1, \ldots, \sigma_m) \in \langle X \rangle$ be arbitrary. Let $\eta_1 \in \langle X_1 \rangle$ be a lexicographically minimal optimal move such that $\{r : \mathrm{game}(s, \sigma_1, X_1) \vDash C_r\} \subseteq \{r : \mathrm{game}(s, \eta_1, X_1) \vDash C_r\}$. This is possible as either σ_1 is already a minimal optimal move, or there exists an optimal move η_1 that satisfies all clauses satisfied by σ_1.

We continue inductively: Suppose we already constructed $\eta_1, \ldots, \eta_j$. Then let $\eta_{j+1} \in \langle X_{j+1} \rangle$ be a lexicographically minimal optimal move such that $\{r : \mathrm{game}(s, \eta_1 \cup \ldots \cup \eta_j \cup \sigma_{j+1}, X_{j+1}) \vDash C_r\} \subseteq \{r : \mathrm{game}(s, \eta_1 \cup \ldots \cup \eta_{j+1}, X_{j+1}) \vDash C_r\}$. By the same argument,

either σ_{j+1} is already optimal and minimal, or we find a better assignment η_{j+1} that satisfies all the clauses satisfied by σ_{j+1}. We have now constructed $\gamma = (\eta_1, \ldots, \eta_m) \in K(\Phi, s)$.

We set $\tilde{s}_{U_j}(\sigma_1 \cup \ldots \cup \sigma_j) := s_{U_j}(\eta_1 \cup \ldots \cup \eta_j)$. Obviously this is a strategy. It remains to show that it is in fact a winning strategy, meaning that for all $\sigma \in \langle X \rangle$ the assignment $\sigma \cup \tilde{s}(\sigma)$ has to falsify some clause C_q.

Let $\sigma = (\sigma_1, \ldots, \sigma_m) \in \langle X \rangle$ be arbitrary and let $\gamma = (\eta_1, \ldots, \eta_m) \in K(\Phi, s)$ be the optimal moves constructed as described above. We know that s is a winning strategy, hence some C_q is falsified by $\gamma \cup s(\gamma)$. Assume, for the sake of contradiction, that C_q is satisfied by $\sigma \cup \tilde{s}(\sigma) = \sigma \cup s(\gamma)$. That means there is some σ_j with $\sigma_j \models C_q$. But then we have $q \in \{r : \text{game}(s, \eta_1 \cup \ldots \cup \eta_{j-1} \cup \sigma_j, X_j) \models C_r\} \subseteq \{r : \text{game}(s, \eta_1 \cup \ldots \cup \eta_j, X_j) \models C_r\}$, and therefore $\text{game}(s, \eta_1 \cup \ldots \cup \eta_j, X_j)$ would also satisfy C_q. This is a contradiction to the fact that C_q is falsified by $\gamma \cup s(\gamma)$.

Hence $\tilde{s}$ is indeed a winning strategy for Φ. Intuitively, $\tilde{s}$ projects all input assignments to better optimal, lexicographically minimal assignments and applies s to these optimal assignments.

Let us write $F(\Phi, s) = \{\xi_1, \ldots, \xi_z\}$. For each ξ_i there exists a lexicographically smallest $\gamma_i \in K(\Phi, s)$ such that $\text{fru}(\gamma_i) = \xi_i$. We can assume that $F(\Phi, s) = \{\xi_1, \ldots, \xi_z\}$ is ordered along the lexicographic order of the corresponding γ_i.

We now define a UDL $\mathcal{M}$ as $(D_1, \mu_1), \ldots, (D_z, \mu_z)$ with $D_i := \bigwedge_{j=1}^{m} \bigwedge_{r \notin I_j \in \xi_i} \neg C_r|_{X_1 \cup \ldots \cup X_j}$ and $\mu_i := s(\gamma_i)$ for $i = 1, \ldots, z$, where the expression $r \notin I_j \in \xi_i$ means that we go through all indices r that do not appear in the j^{th} set I_j from ξ_i.

We now have to show that $\mathcal{M}$ is an UDL. We will do this by proving that $\mathcal{M}$ computes $\tilde{s}$.

▷ Claim 28. *i is minimal such that $\sigma \models D_i$ if and only if for each $j = 1, \ldots, m$ the assignment η_j from the optimal chain $\gamma_i = (\eta_1, \ldots, \eta_m)$ is the lexicographical minimal optimal move such that $\{r : \text{game}(s, \eta_1 \cup \ldots \cup \eta_{j-1} \cup \sigma_j, X_j) \models C_r\} \subseteq \{r : \text{game}(s, \eta_1 \cup \ldots \cup \eta_j, X_j) \models C_r\}$.*

Proof. For the first direction from left to right, let i be minimal such that $\sigma \models D_i$. That means for each $j = 1, \ldots, m$ we have that σ satisfies $\neg C_r|_{X_1 \cup \ldots \cup X_j}$ whenever $r \notin I_j$, where $(I_1, \ldots, I_m) = \xi_i$. In other words, if σ satisfies $C_r|_{X_1 \cup \ldots \cup X_j}$, then we have $r \in I_j$. Obviously, σ satisfies $C_r|_{X_1 \cup \ldots \cup X_j}$ if and only if $\sigma_1 \cup \ldots \cup \sigma_j$ satisifes C_r.

Let $q \in \{r : \text{game}(s, \eta_1 \cup \ldots \cup \eta_{j-1} \cup \sigma_j, X_j) \models C_r\}$. If $\text{game}(s, \eta_1 \cup \ldots \cup \eta_{j-1} \cup \sigma_j, X_j)|_U$ satisfies C_q, then $\text{game}(s, \eta_1 \cup \ldots \cup \eta_j, X_j)|_U$ satisfies C_q as well, since they are identical. If $\text{game}(s, \eta_1 \cup \ldots \cup \eta_{j-1} \cup \sigma_j, X_j)|_U$ does not satisfy C_q, then $\eta_1 \cup \ldots \cup \eta_{j-1} \cup \sigma_j$ satisfies C_q. If $\eta_1 \cup \ldots \cup \eta_{j-1}$ satisfies C_q, then so does $\eta_1 \cup \ldots \cup \eta_j$. If σ_j satisfies C_q, then σ satisfies $C_q|_{X_1 \cup \ldots \cup X_j}$ and we have $q \in I_j$. In any case, we obtain $q \in \{r : \text{game}(s, \eta_1 \cup \ldots \cup \eta_j, X_j) \models C_r\}$ and therefore $\{r : \text{game}(s, \eta_1 \cup \ldots \cup \eta_{j-1} \cup \sigma_j, X_j) \models C_r\} \subseteq \{r : \text{game}(s, \eta_1 \cup \ldots \cup \eta_j, X_j) \models C_r\}$. Due to the fact that the D_i are ordered lexicographically along the γ_i, each η_j is lexicographically minimal, which concludes the first direction.

For the other direction, suppose that for each $j = 1, \ldots, m$ the assignment η_j from the optimal chain $\gamma_i = (\eta_1, \ldots, \eta_m)$ is the lexicographically minimal optimal move such that $\{r : \text{game}(s, \eta_1 \cup \ldots \cup \eta_{j-1} \cup \sigma_j, X_j) \models C_r\} \subseteq \{r : \text{game}(s, \eta_1 \cup \ldots \cup \eta_j, X_j) \models C_r\}$. Let $q \notin I_j$ be arbitrary. We want to show that $\text{game}(s, \sigma_1 \cup \ldots \cup \sigma_j, X_j)$ does not satisfy C_q. Assume, for the sake of contradiction, $\text{game}(s, \sigma_1 \cup \ldots \cup \sigma_j, X_j)$ does indeed satisfy C_q.

If $\sigma_1 \cup \ldots \cup \sigma_j$ satisfies C_q, then there exists a minimal a such that σ_a satisfies C_q, which implies that $\text{game}(s, \eta_1 \cup \ldots \cup \eta_a, X_a)$ satisfies C_q as well. Then we would have $q \in I_a \subseteq I_j$, which is a contradiction to the choice of q.

Hence, $\sigma_1 \cup \ldots \cup \sigma_j$ does not satisfy C_q, and therefore σ falsifies $C_q|_{X_1 \cup \ldots \cup X_j}$. Because this is true for each $j = 1, \ldots, m$, we conclude that σ satisfies D_i. Because each η_j was defined to be lexicographically minimal, and the ordering of the D_i is consistent with this lexicographic order, we conclude that i is minimal, proving the claim. ◁

The claim proves that $\mathcal{M}$ computes exactly $\tilde{s}$. This not only shows $\bigvee_{i=1}^{z} D_i \equiv \top$, but also that $\mathcal{M}$ is a UDL with $|\mathcal{M}| = z = |F(\Phi, s)| \leqslant \mathrm{cw}^{-}(\Phi)^m$. ◀

Because of the lower bound of $\mathsf{CAQE}^{\mathsf{CORE}}$ proofs via CAQE width (Proposition 22) and the UDL characterization of $\mathsf{Q}^{\mathsf{NP}}_{=}\mathsf{Res}$ (Theorem 26), as well as fact that the QBFs $\mathtt{Copy}_n$ are obviously easy for $\mathsf{Q}^{\mathsf{NP}}_{=}\mathsf{Res}$ but hard for $\mathsf{CAQE}^{\mathsf{CORE}}$ (cf. Theorem 24) we conclude:

▶ **Corollary 29.** $\mathsf{Q}^{\mathsf{NP}}_{=}\mathsf{Res}$ *simulates* $\mathsf{CAQE}^{\mathsf{CORE}}$ *on bounded quantifier alternation, but* $\mathsf{CAQE}^{\mathsf{CORE}}$ *is exponentially weaker than* $\mathsf{Q}^{\mathsf{NP}}_{=}\mathsf{Res}$ *as witnessed by* $\mathtt{Copy}_n$.

8 CAQE vs QCDCL

Finally, we compare $\mathsf{CAQE}^{\mathsf{CORE}}$ and QCDCL. We will show that there is a separation between $\mathsf{CAQE}^{\mathsf{CORE}}$ and QCDCL in one direction, and for the other direction, at least on a specific but relevant class of QCNFs, QCDCL can simulate $\mathsf{CAQE}^{\mathsf{CORE}}$ proofs. We utilize the QCDCL proof system framework used in [17, 15, 7] that proof-theoretically tightly captures the algorithmic QCDCL paradigm. While we omit precise definitions (cf. [7]), a QCDCL proof is a sequence of tuples of the form $(\mathcal{T}_i, C_i, \pi_i)$. The $\mathcal{T}_i$ are called *trails*, which are (ordered) assignments containing all decided and propagated literals in the order in which they were assigned. Each C_i is the clause learned from the trail $\mathcal{T}_i$ and π_i is the derivation of the learned clause C_i from the axioms and previously learned clauses (e.g. via Q-Res).

We have already shown that the equality formulas have exponential CAQE width, hence they need exponential-size $\mathsf{CAQE}^{\mathsf{CORE}}$ proofs by Proposition 22. We recall that $\mathtt{Eq}_n$ have QCDCL refutations of quadratic size [16]. Hence we obtain:

▶ **Theorem 30.** $\mathtt{Eq}_n$ *require* $\mathsf{CAQE}^{\mathsf{CORE}}$ *proofs of size* $\Omega(2^n)$ *and have* QCDCL *proofs of size* $\mathcal{O}(n^2)$. *Hence,* $\mathsf{CAQE}^{\mathsf{CORE}}$ *cannot simulate* QCDCL.

One major difference between $\mathsf{CAQE}^{\mathsf{CORE}}$ and QCDCL is that QCDCL does not directly call a SAT solver. Instead, QCDCL itself is a natural extension of CDCL and therefore contains all CDCL functionality. This also means that it is not as trivial to discard propositional hardness from QCDCL as it is with $\mathsf{CAQE}^{\mathsf{CORE}}$ where computations by the SAT oracle are not counted towards $\mathsf{CAQE}^{\mathsf{CORE}}$ proof size. Hence we need to define explicitly which steps of QCDCL could be performed by CDCL as well.

▶ **Definition 31** (CDCL steps (informal))**.** *A* CDCL *step in a* QCDCL *proof* ι *is a subsequence of* ι *that can occur in a* CDCL *proof as well. A* CDCL *step in* QCDCL *occurs if the formula under the current assignment (i.e. the trail) contains a subformula that is propositional, unsatisfiable and only contains variables from the (currently) first existential quantifier block. In such a situation,* QCDCL *can run into a conflict by propagating and deciding variables from this subformula only without any reductions. The subsequent clause learning will therefore only perform operations that are also doable in* CDCL *if we do not overstep the point on which the unsatisfiable subformula appeared.*

For a QCDCL *refutation* ι *we denote by* $\iota|_{\mathsf{CDCL}}$ *the collection of* CDCL *steps in* ι. *Hence* $|\iota| - |\iota|_{\mathsf{CDCL}}|$ *measures "genuine" QCDCL size, discarding pure* CDCL *runs.*

To ensure that such CDCL steps are cleanly detectable, we require that trails follow a certain structure. A way to achieve this is by imposing the *XT-property* on the QCNF [14]. Informally, this XT-property causes the trails to be fully level-ordered (from X- to U- to T-variables), which is important for propagations to not getting mixed up too much. Originally, the XT-property was designed to obtain QCDCL lower bounds (cf. [14]) as level-ordered trails turn out to be disadvantageous for QCDCL runtime. Ironically, this XT-property will now help us to obtain upper bounds for QCDCL.

▶ **Definition 32** (XT-property, [14]). *A Σ_3^b QCNF with the prefix $\exists X \forall U \exists T$ fulfils the* XT-property *if no two clauses C_1, C_2 with $\mathrm{var}(C_1), \mathrm{var}(C_2) \subseteq T$ ("T-clauses") are resolvable and if it contains no unit clauses (ℓ) with $\ell \in T$ ("unit T-clauses") and no clauses C with $\mathrm{var}(C) \cap X \neq \varnothing$, $\mathrm{var}(C) \cap T \neq \varnothing$ and $\mathrm{var}(C) \cap U = \varnothing$ ("XT-clauses").*

Note that the XT-property still holds after learning a clause (cf. [14]). This means that a formula with the XT-property will continue to fulfil it for the entire QCDCL run. We can now show:

▶ **Theorem 33.** *Let Φ be a false Σ_3^b QCNF that fulfils the XT-property. Then there exists a* QCDCL *refutation ι with $|\iota| - |\iota|_{\mathsf{CDCL}} \leqslant \mathrm{cw}^-(\Phi) \cdot |\mathrm{var}(\Phi)|$.*

Proof Sketch. We make use of the UDL $\mathcal{M}$ which we have constructed in Theorem 27 from the (optimal) strategy s. This UDL consists of the pairs $(D_1, \mu_1), \ldots, (D_z, \mu_z)$ with $z \leqslant \mathrm{cw}^-(\Phi)$. The plan is to construct trails from which we learn each $\neg D_i$ for $i = 1, \ldots, z$. In order to learn $\neg D_i$, we need to choose D_i as decisions, then decide $\gamma_i|_X \setminus D_i$ and then $s(\gamma_i)$. We also need to take into account that some decisions might not be possible due to the propagations. In a nutshell, we show that we can essentially learn each $\neg D_i$ after a number of CDCL steps. Having learned all $\neg D_i$, we learn the empty clause with a CDCL step as $\bigwedge_{i=1}^{z} \neg D_i \equiv \bot$. Hence, by ignoring all CDCL steps, we only need at most z many (genuine) QCDCL trails in order to construct a refutation. ◀

▶ Remark 34. If $\Phi = \exists X \forall U \exists T \cdot C_1 \wedge \ldots \wedge C_k$ is a false Σ_3^b QCNF that does not fulfil the XT-property, then we can transform it into an equivalent Σ_3^b QCNF with the XT-property as follows: Let w be a new variable. We set $C'_r := w \vee C_r$ for all $r = 1, \ldots, k$ and define $\Phi' := \exists X \forall U \forall w \exists T \cdot C'_1 \wedge \ldots \wedge C'_k$. Since we do not alter the first block, we even get $\mathrm{cw}^-(\Phi) = \mathrm{cw}^-(\Phi')$. By allowing this preprocessing step, we can extend the upper bound from Theorem 33 to arbitrary false Σ_3^b QCNFs.

▶ **Corollary 35.** QCDCL *simulates* $\mathsf{CAQE}^{\mathsf{CORE}}$ *on false Σ_3^b QCNFs that fulfil the XT-property, under the condition that* CDCL *steps in* QCDCL *are not counted. By adding the preprocessing step as described in Remark 34,* QCDCL *even simulates* $\mathsf{CAQE}^{\mathsf{CORE}}$ *on all false Σ_3^b QCNFs.*

9 Conclusion

We have proven the possibly surprising result that, in a theoretical setting, the basic version of CAQE (without failed assumptions) is weaker than QCDCL on a large class of formulas (cf. Figure 1), while in practice CAQE pretty much outperforms all competing QBF solvers. This indicates that optimisations such as failed assumptions might play an important role for the performance of the solver. In future work, our lower and upper bound techniques could be used to prove separations between different variants of CAQE formally, and hence justify the use of modifications on a proof complexity level. Furthermore, the fact that we discarded propositional hardness in both approaches may be a very relevant factor in

comparing these two solvers. It can be assumed that direct usage of a SAT solver is a huge advantage for CAQE, while QCDCL as an extension of CDCL utilises SAT solving in a rather intricate and hidden way. Given the relatively simple setup of CAQE$^{\text{CORE}}$ (Figure 2) this also means that CAQE$^{\text{CORE}}$ can at least in principle use the latest state-of-the-art SAT solver without any changes to its implementation, while QCDCL does not have access to the latest SAT technology without major implementation effort. Furthermore we have shown that, discounting propositional hardness, QBF solving can basically be reduced to computing all optimal moves (or rather all optimal fruitions). Further research into this direction may have the potential to not only improve the performance of CAQE itself, but also lead to decision heuristics for QCDCL.

References

1 Valeriy Balabanov, Magdalena Widl, and Jie-Hong R. Jiang. QBF resolution systems and their proof complexities. In *Proc. Theory and Applications of Satisfiability Testing (SAT)*, pages 154–169, 2014. doi:10.1007/978-3-319-09284-3_12.

2 Olaf Beyersdorff. Proof complexity of quantified Boolean logic – a survey. In Marco Benini, Olaf Beyersdorff, Michael Rathjen, and Peter Schuster, editors, *Mathematics for Computation (M4C)*, pages 353–391. World Scientific, Singapore, 2022.

3 Olaf Beyersdorff, Joshua Blinkhorn, and Luke Hinde. Size, cost, and capacity: A semantic technique for hard random QBFs. *Logical Methods in Computer Science*, 15(1), 2019. doi:10.23638/LMCS-15(1:13)2019.

4 Olaf Beyersdorff, Joshua Blinkhorn, Meena Mahajan, and Tomás Peitl. Hardness characterisations and size-width lower bounds for QBF resolution. *ACM Trans. Comput. Log.*, 24(2):10:1–10:30, 2023. doi:10.1145/3565286.

5 Olaf Beyersdorff, Joshua Blinkhorn, Meena Mahajan, Tomás Peitl, and Gaurav Sood. Hard QBFs for merge resolution. *ACM Trans. Comput. Theory*, 16(2):6:1–6:24, 2024. doi:10.1145/3638263.

6 Olaf Beyersdorff and Benjamin Böhm. Understanding the relative strength of QBF CDCL solvers and QBF resolution. *Log. Methods Comput. Sci.*, 19(2), 2023. doi:10.46298/LMCS-19(2:2)2023.

7 Olaf Beyersdorff, Benjamin Böhm, and Meena Mahajan. Runtime vs. extracted proof size: An exponential gap for CDCL on QBFs. *J. Autom. Reason.*, 69(4):30, 2025. doi:10.1007/S10817-025-09745-6.

8 Olaf Beyersdorff, Ilario Bonacina, Leroy Chew, and Jan Pich. Frege systems for quantified Boolean logic. *J. ACM*, 67(2):9:1–9:36, 2020. doi:10.1145/3381881.

9 Olaf Beyersdorff, Ilario Bonacina, Kaspar Kasche, Meena Mahajan, and Luc Nicolas Spachmann. Semi-algebraic proof systems for QBF. In Jeremias Berg and Jakob Nordström, editors, *28th International Conference on Theory and Applications of Satisfiability Testing (SAT)*, volume 341 of *LIPIcs*, pages 5:1–5:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.SAT.2025.5.

10 Olaf Beyersdorff, Leroy Chew, and Mikolás Janota. New resolution-based QBF calculi and their proof complexity. *ACM Transactions on Computation Theory*, 11(4):26:1–26:42, 2019. doi:10.1145/3352155.

11 Olaf Beyersdorff, Luke Hinde, and Ján Pich. Reasons for hardness in QBF proof systems. *ACM Transactions on Computation Theory*, 12(2):10:1–10:27, 2020. doi:10.1145/3378665.

12 Olaf Beyersdorff, Tim Hoffmann, and Luc Nicolas Spachmann. Proof complexity of propositional model counting. *J. Satisf. Boolean Model. Comput.*, 15(1):27–59, 2024. doi:10.3233/SAT-231507.

13 Olaf Beyersdorff, Mikolás Janota, Florian Lonsing, and Martina Seidl. Quantified Boolean formulas. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook*

of Satisfiability, Frontiers in Artificial Intelligence and Applications, pages 1177–1221. IOS Press, 2021. doi:10.3233/FAIA201015.

14 Benjamin Böhm and Olaf Beyersdorff. Lower bounds for QCDCL via formula gauge. *J. Autom. Reason.*, 67(4):35, 2023. doi:10.1007/S10817-023-09683-1.

15 Benjamin Böhm and Olaf Beyersdorff. QCDCL vs QBF resolution: Further insights. *J. Artif. Intell. Res.*, 81:741–769, 2024. doi:10.1613/JAIR.1.15522.

16 Benjamin Böhm, Tomás Peitl, and Olaf Beyersdorff. QCDCL with cube learning or pure literal elimination - what is best? *Artif. Intell.*, 336:104194, 2024. doi:10.1016/J.ARTINT.2024.104194.

17 Benjamin Böhm, Tomás Peitl, and Olaf Beyersdorff. Should decisions in QCDCL follow prefix order? *J. Autom. Reason.*, 68(1):5, 2024. doi:10.1007/S10817-024-09694-6.

18 Sam Buss and Jakob Nordström. Proof complexity and SAT solving. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, Frontiers in Artificial Intelligence and Applications, pages 233–350. IOS Press, 2021. doi:10.3233/FAIA200990.

19 Leroy Chew and Marijn J. H. Heule. Relating existing powerful proof systems for QBF. In Kuldeep S. Meel and Ofer Strichman, editors, *25th International Conference on Theory and Applications of Satisfiability Testing (SAT)*, volume 236 of *LIPIcs*, pages 10:1–10:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SAT.2022.10.

20 Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979. doi:10.2307/2273702.

21 Uwe Egly, Florian Lonsing, and Magdalena Widl. Long-distance resolution: Proof generation and strategy extraction in search-based QBF solving. In *Proc. Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, pages 291–308, 2013. doi:10.1007/978-3-642-45221-5_21.

22 Marijn J. H. Heule, Martina Seidl, and Armin Biere. Solution validation and extraction for QBF preprocessing. *J. Autom. Reason.*, 58(1):97–125, 2017. doi:10.1007/S10817-016-9390-4.

23 Mikolás Janota and Joao Marques-Silva. Expansion-based QBF solving versus Q-resolution. *Theor. Comput. Sci.*, 577:25–42, 2015. doi:10.1016/J.TCS.2015.01.048.

24 Hans Kleine Büning, Marek Karpinski, and Andreas Flögel. Resolution for quantified Boolean formulas. *Inf. Comput.*, 117(1):12–18, 1995. doi:10.1006/INCO.1995.1025.

25 Florian Lonsing and Uwe Egly. DepQBF 6.0: A search-based QBF solver beyond traditional QCDCL. In *Proc. International Conference on Automated Deduction (CADE)*, pages 371–384, 2017. doi:10.1007/978-3-319-63046-5_23.

26 João P. Marques Silva, Inês Lynce, and Sharad Malik. Conflict-driven clause learning SAT solvers. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, Frontiers in Artificial Intelligence and Applications. IOS Press, 2021. doi:10.3233/FAIA200987.

27 Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Dependency learning for QBF. *J. Artif. Intell. Res.*, 65:180–208, 2019. doi:10.1613/JAIR.1.11529.

28 Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Long-distance Q-resolution with dependency schemes. *J. Autom. Reasoning*, 63(1):127–155, 2019. doi:10.1007/S10817-018-9467-3.

29 Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Qute in the QBF evaluation 2018. *J. Satisf. Boolean Model. Comput.*, 11(1):261–272, 2019. doi:10.3233/SAT190124.

30 Luca Pulina and Martina Seidl. The 2016 and 2017 QBF solvers evaluations (QBFEVAL'16 and QBFEVAL'17). *Artif. Intell.*, 274:224–248, 2019. doi:10.1016/J.ARTINT.2019.04.002.

31 QBF solver evaluation portal. https://www.qbflib.org/index_eval.php.

32 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)*, pages 136–143. IEEE, 2015. doi:10.1109/FMCAD.2015.7542263.

33 Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987. doi:10.1007/BF00058680.

34 Ankit Shukla, Armin Biere, Luca Pulina, and Martina Seidl. A survey on applications of quantified Boolean formulas. In *Proc. IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 78–84, 2019. doi:10.1109/ICTAI.2019.00020.

35 Leander Tentrup. On expansion and resolution in CEGAR based QBF solving. In Rupak Majumdar and Viktor Kuncak, editors, *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II*, volume 10427 of *Lecture Notes in Computer Science*, pages 475–494. Springer, 2017. doi:10.1007/978-3-319-63390-9_25.

36 Leander Tentrup. CAQE and quabs: Abstraction based QBF solvers. *J. Satisf. Boolean Model. Comput.*, 11(1):155–210, 2019. doi:10.3233/SAT190121.

37 Allen Van Gelder. Contributions to the theory of practical quantified Boolean formula solving. In *Proc. Principles and Practice of Constraint Programming (CP)*, pages 647–663, 2012. doi:10.1007/978-3-642-33558-7_47.

38 Lintao Zhang and Sharad Malik. Conflict driven learning in a quantified Boolean satisfiability solver. In *Proc. IEEE/ACM International Conference on Computer-aided Design (ICCAD)*, pages 442–449, 2002. doi:10.1145/774572.774637.