Journal of Automated Reasoning (2024) 68:5
https://doi.org/10.1007/s10817-024-09694-6

Should Decisions in QCDCL Follow Prefix Order?

Benjamin Böhm[1] · Tomáš Peitl[2] · Olaf Beyersdorff[1]

Received: 8 July 2023 / Accepted: 8 January 2024 / Published online: 9 February 2024

Abstract
Quantified conflict-driven clause learning (QCDCL) is one of the main solving approaches for quantified Boolean formulas (QBF). One of the differences between QCDCL and propositional CDCL is that QCDCL typically follows the prefix order of the QBF for making decisions. We investigate an alternative model for QCDCL solving where decisions can be made in arbitrary order. The resulting system $\mathsf{QCDCL}^{\textsc{Any}}$ is still sound and terminating, but does not necessarily allow to always learn asserting clauses or cubes. To address this potential drawback, we additionally introduce two subsystems that guarantee to always learn asserting clauses ($\mathsf{QCDCL}^{\textsc{Uni-Any}}$) and asserting cubes ($\mathsf{QCDCL}^{\textsc{Exi-Any}}$), respectively. We model all four approaches by formal proof systems and show that $\mathsf{QCDCL}^{\textsc{Uni-Any}}$ is exponentially better than QCDCL on false formulas, whereas $\mathsf{QCDCL}^{\textsc{Exi-Any}}$ is exponentially better than QCDCL on true QBFs. Technically, this involves constructing specific QBF families and showing lower and upper bounds in the respective proof systems. We complement our theoretical study with some initial experiments that confirm our theoretical findings.

Keywords QBF · CDCL · Proof complexity · Lower bounds

1 Introduction

SAT solving was revolutionised in the late 1990 s by the advent of conflict-driven clause learning (CDCL), which has since been the dominating paradigm in propositional SAT solving [24, 25, 37]. A few years later, the CDCL approach was lifted to the computationally

---

An extended abstract of this paper appeared in the proceedings of SAT'22 [8].

T. Peitl: Supported by FWF grant J-4361 (Austrian Science Fund)
O. Beyersdorff: Supported by the Carl Zeiss Foundation and DFG grant BE 4209/3-1.

✉ Benjamin Böhm
benjamin.boehm@uni-jena.de

Tomáš Peitl
peitl@ac.tuwien.ac.at

Olaf Beyersdorff
olaf.beyersdorff@uni-jena.de

1 Friedrich Schiller University Jena, Jena, Germany

2 TU Wien, Vienna, Austria

even harder setting of quantified Boolean formulas (QBF) in the form of quantified CDCL (QCDCL) [38]. Though a number of competing approaches to QBF solving exist (cf. [7] for a recent overview), QCDCL is one of most competitive. State-of-the-art implementations include DepQBF [21] and Qute [29, 33].

In comparison to the propositional case, QCDCL poses additional technical challenges, stemming from partitioning the variables into existential and universal (SAT can be viewed as using only existential variables) and the dependencies between the variables imposed by the quantifier prefix. The presence of universal variables entails additional rules for unit propagation (universal reductions), while the variable dependencies imposed by the prefix are typically observed by decision heuristics in the sense that QCDCL follows the prefix order in decision making. The latter is arguably the most severe restriction when transitioning from CDCL to QCDCL. Another difference between CDCL and QCDCL arises from the fact that unlike in SAT, a satisfying assignment to the QBF matrix does not imply that the QBF is true. Instead, this is witnessed by additionally learning cubes (i.e., conjunctions of literals, also called terms) and producing a cube certificate for true QBFs.

Though CDCL and QCDCL are very efficient in practice and in particular on industrial instances (cf. [32] for an overview of QBF solving applications and [16, 22] for experimental studies of solver performance), their success and their inherent limitations are not at all well-understood from a theoretical perspective. The main theoretical approach is through proof complexity [10]. For SAT it is known that CDCL—viewed as a non-deterministic procedure—is equivalent to propositional resolution [1, 3, 30]. In particular, resolution refutations can be efficiently extracted from CDCL runs, whereby lower bounds for resolution proof size imply lower bounds for CDCL running time. However, when using CDCL with practical decision heuristics such as VSIDS [26], the model becomes exponentially weaker than resolution [36].

The situation is even more intricate in QBF. Again, from QCDCL runs, proofs can be efficiently extracted in the format of long-distance Q-Resolution [2, 38].[1] However, QCDCL—even as a non-deterministic procedure—is exponentially weaker than long-distance Q-Resolution and incomparable to the simpler system of Q-Resolution [5]. Thus it is very interesting, both from a theoretical and practical perspective, to gauge the precise power of QCDCL.

In this paper we introduce and investigate QCDCL models that drop the requirement of making variable decisions along the prefix order. Though it has been recently shown that following the prefix order in QCDCL is not needed for correctness,[2] existing prefix-relaxing techniques do not exploit this as much as they could. Dependency schemes [23, 28, 31, 34] work with the assumption that the prefix has to be observed, but notice that certain parts (often called *spurious dependencies*) can be relaxed in preprocessing. With dependency learning [29], a more recent, orthogonal technique, instead of calculating dependencies upfront the solver assumes independence until it runs into a problem, from which it learns a dependency on the fly (dependency learning can be combined with schemes [27]). These strategies are executed differently: with dependency schemes the solver can fully rely on the relaxed prefix and use it for decisions, propagation, and clause/cube learning alike; with dependency learning the solver can only use the relaxed prefix for decisions and propagation and must learn clauses and cubes with the original prefix in order to detect dependencies. However, both approaches share the restriction that once dependencies are found, decisions must respect them.

1 For practical solving, more succinct proof checking formats are used, both for CDCL [14] and QCDCL [15].

2 It is needed for QDPLL [11, 13], but not for QCDCL (cf. also [5]).

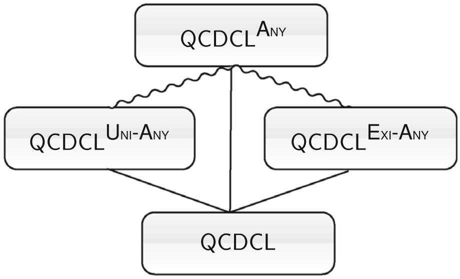

Fig. 1 Hasse diagram of the simulation order of QCDCL proof systems. Solid lines represent p-simulations and exponential separations. Waved lines represent p-simulations, for which separations are not known

Our contributions. We propose a new QCDCL model where decisions can ignore quantification entirely; only propagation and clause/cube learning use the prefix information.

When suggesting a new model for solving, there are at least two possible approaches: (1) to give a formal account of the model, prove its correctness, and theoretically quantify the gains on running time; or (2) provide an implementation and experimentally evaluate its practical performance. In this paper, our main focus is to contribute towards (1). While we also perform some initial proof-of-concept experiments, an extensive practical evaluation of the competitiveness of the approach is left for future work (cf. the conclusion).[3]

Specifically, our contributions are as follows:

1. Formal proof complexity models for QCDCL using arbitrary decisions. We provide a formal proof-complexity model for QCDCL with arbitrary decisions. This follows a recent line of research to formalise and rigorously analyse QCDCL from a proof complexity perspective [5, 9].

Our most general model $\mathsf{QCDCL}^{\mathsf{ANY}}$ allows arbitrary decisions. Care has to be taken to ensure that we can always learn new clauses and cubes, as otherwise termination of proof search is no longer guaranteed. We ensure this by adding a simple *new constraint condition* (NCC), which forbids making decisions that immediately falsify a clause or satisfy a cube (which is already trivially impossible in prefix-observing QCDCL).

A potential further drawback of not following prefix order is that we can no longer guarantee to learn *asserting* clauses or cubes.[4] In order to address this, we introduce two subsystems of $\mathsf{QCDCL}^{\mathsf{ANY}}$—termed $\mathsf{QCDCL}^{\mathsf{UNI\text{-}ANY}}$ and $\mathsf{QCDCL}^{\mathsf{EXI\text{-}ANY}}$—that allow to always learn asserting clauses and cubes, respectively. We prove that all three systems are sound, complete, and terminating.

2. Exponential separations between the QCDCL models. The main contribution of this paper lies in proving that both $\mathsf{QCDCL}^{\mathsf{UNI\text{-}ANY}}$ and $\mathsf{QCDCL}^{\mathsf{EXI\text{-}ANY}}$ allow for exponentially shorter proofs than the prefix-following QCDCL model. The resulting simulation order is depicted in Fig. 1.

To show this we construct two QBF families that exponentially separate the systems. Both employ general constructions—using a ‘twin’ and a ‘reverse’ construction—that could potentially be used for further formulas. Technically, we use the recently developed lower bound approach via the *gauge* of QBFs [9]. However, different from previous work [5, 9], which only considered clause learning, our lower bounds work against a more realistic QCDCL system that uses both clause and cube learning. Interestingly, the separation of $\mathsf{QCDCL}^{\mathsf{UNI\text{-}ANY}}$ from QCDCL works on false QBFs, while the separation of $\mathsf{QCDCL}^{\mathsf{EXI\text{-}ANY}}$ from

[3] It appears to us that in SAT solving, theoretical analysis has so far been mainly carried out in retrospect, after practical solving developments had already taken place. However, we also see a case for theoretical research providing a-priori *guidance* for practical developments.

[4] An asserting clause/cube becomes unit after backtracking. This concept is important in practical SAT and QBF solving where only asserting clauses/cubes are learned.

QCDCL uses true QBFs. The latter is the first dedicated QBF proof-complexity lower bound on true formulas.[5] In fact, we provide a general method how to transform hardness of false QBFs into hardness of true formulas.

3. Proof-of-concept experiments. Though this is not our main focus, we provide initial experiments that confirm our theoretical findings. These experiments are only meant to illustrate that our approach is in principle competitive with plain QCDCL, without considering the impact of other techniques like preprocessing or dependency learning, etc. (cf. the discussion of future work in the conclusion).

Organisation. The remainder of this paper is organised as follows. We start in Sect. 2 with reviewing QBF preliminaries. In Sects. 3 and 4 we introduce and formally model the new QCDCL versions. Their proof-complexity analysis and the separations are proven in Sect. 5. Section 6 describes our proof-of-concept experiments and Sect. 7 outlines further work.

2 Preliminaries

Propositional and quantified formulas. Variables x and negated variables $\bar{x}$ are called *literals*. We denote the corresponding variable as $\text{var}(x) := \text{var}(\bar{x}) := x$.

A *clause* is a disjunction of literals and a *cube* is a conjunction of literals. We will sometimes interpret clauses and cubes as sets of literals on which we can perform set-theoretic operations.

A *unit clause* (ℓ) is a clause that consists of only one literal. The *empty clause* consists of zero literals, denoted $(\bot)$. We sometimes paraphrase $(\bot)$ as a unit clause with the 'empty literal' $\bot$. A clause C is called *tautological* if $\{\ell, \bar{\ell}\} \subseteq C$ for some literal ℓ.

We define a *unit cube* of a literal ℓ, denoted by $[\ell]$, and the *empty cube* $[\top]$ with 'empty literal' $\top$. A cube D is *contradictory* if $\{\ell, \bar{\ell}\} \subseteq D$ for some literal ℓ. If C is a clause or a cube, we define $\text{var}(C) := \{\text{var}(\ell) : \ell \in C\}$. The negation of a clause $C = \ell_1 \vee \ldots \vee \ell_m$ is the cube $\neg C := \overline{C} := \bar{\ell}_1 \wedge \ldots \wedge \bar{\ell}_m$.

A *(total) assignment* σ of a set of variables V is a non-tautological set of literals such that for all $x \in V$ there is some $\ell \in \sigma$ with $\text{var}(\ell) = x$. A *partial assignment* σ of V is an assignment of a subset $W \subseteq V$. A clause C is *satisfied* by an assignment σ if $C \cap \sigma \neq \emptyset$. A cube D is *falsified* by σ if $\neg D \cap \sigma \neq \emptyset$. 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

$$D|_\sigma := \bigwedge_{\ell \in D \setminus \sigma} \ell.$$

Intuitively, an assignment sets all its literals to true.

A *CNF* (conjunctive normal form) is a conjunction of clauses and a *DNF* (disjunctive normal form) is a disjunction of cubes. We restrict a CNF (resp. DNF) ϕ by an assignment σ as

$$\phi|_\sigma := \bigwedge_{C \in \phi \text{ non-satisfied}} C|_\sigma \quad \left(\text{resp. } \phi|_\sigma := \bigvee_{D \in \phi \text{ non-falsified}} D|_\sigma\right).$$

[5] Also in SAT, basically all lower bounds are for unsatisfiable formulas.

For a CNF (DNF) ϕ and an assignment σ, if $\phi|_\sigma = \emptyset$, then ϕ is *satisfied* (*falsified*) by σ.

A *QBF* (quantified Boolean formula) $\Phi = \mathcal{Q} \cdot \phi$ consists of a propositional formula ϕ, called the *matrix*, and a *prefix* $\mathcal{Q}$. A *prefix* $\mathcal{Q} = \mathcal{Q}'_1 V_1 \ldots \mathcal{Q}'_s V_s$ consists of non-empty and pairwise disjoint sets of variables $V_1, \ldots, V_s$ and quantifiers $\mathcal{Q}'_1, \ldots, \mathcal{Q}'_s \in \{\exists, \forall\}$ with $\mathcal{Q}'_i \neq \mathcal{Q}'_{i+1}$ for $i \in [s-1]$. For a variable x in $\mathcal{Q}$, the *quantifier level* is $\text{lv}(x) := \text{lv}_\Phi(x) := i$, if $x \in V_i$. For $\text{lv}_\Phi(\ell_1) < \text{lv}_\Phi(\ell_2)$ we write $\ell_1 <_\Phi \ell_2$, while $\ell_1 \leq_\Phi \ell_2$ means $\text{lv}_\Phi(\ell_1) \leq \text{lv}_\Phi(\ell_2)$.

For a QBF $\Phi = \mathcal{Q} \cdot \phi$ with ϕ a CNF (DNF), we call Φ a *QCNF* (*QDNF*). We define $\mathfrak{C}(\Phi) := \phi$ (resp. $\mathfrak{D}(\Phi) := \phi$). Φ is an *AQBF* (augmented QBF), if $\phi = \psi \vee \chi$ with CNF ψ and DNF χ. Again we write $\mathfrak{C}(\Phi) := \psi$ and $\mathfrak{D}(\Phi) := \chi$.

We restrict a QCNF (QDNF) $\Phi = \mathcal{Q} \cdot \phi$ by an assignment σ as $\Phi|_\sigma := \mathcal{Q}|_\sigma \cdot \phi|_\sigma$, where $\mathcal{Q}|_\sigma$ is obtained by deleting all variables from $\mathcal{Q}$ that appear in σ. Analogously, we restrict an AQBF $\Phi = \mathcal{Q} \cdot (\psi \vee \chi)$ as $\Phi|_\sigma := \mathcal{Q}|_\sigma \cdot (\psi|_\sigma \vee \chi|_\sigma)$.

If L is a set of literals (e.g., an assignment), we can get the negation of L, which we define as $\neg L := \overline{L} := \{\bar{\ell} | \ \ell \in L\}$.

(Long-distance) Q-resolution and Q-consensus. Let C_1 and C_2 be two clauses (cubes) from a QCNF (QDNF) or AQBF Φ. Let ℓ be an existential (universal) literal with $\text{var}(\ell) \notin \text{var}(C_1) \cup \text{var}(C_2)$. The *resolvent* of $C_1 \vee \ell$ and $C_2 \vee \bar{\ell}$ over ℓ is defined as

$$(C_1 \vee \ell) \overset{\ell}{\bowtie} (C_2 \vee \bar{\ell}) := C_1 \vee C_2$$

(resp. $(C_1 \wedge \ell) \overset{\ell}{\bowtie} (C_2 \wedge \bar{\ell}) := C_1 \wedge C_2$).

Let $C := \ell_1 \vee \ldots \vee \ell_m$ be a clause from a QCNF or AQBF Φ such that $\ell_i \leq_\Phi \ell_j$ for all $i < j, i, j \in \{1, \ldots, m\}$. Let k be minimal such that $\ell_k, \ldots, \ell_m$ are universal. Then we can perform a *universal reduction* step and obtain

$$\text{red}^\forall_\Phi(C) := \ell_1 \vee \ldots \vee \ell_{k-1}.$$

Analogously, we perform *existential reduction* on cubes. Let $D := \ell_1 \wedge \ldots \wedge \ell_m$ be a cube of a QDNF or AQBF Φ with $\ell_i \leq_\Phi \ell_j$ for all $i < j, i, j \in \{1, \ldots, m\}$. Let k be minimal such that $\ell_k, \ldots, \ell_m$ are existential. Then $\text{red}^\exists_\Phi(D) := \ell_1 \wedge \ldots \wedge \ell_{k-1}$.

If it is clear that C is a clause or a cube, we can just write $\text{red}_\Phi(C)$ or even $\text{red}(C)$, if the QBF Φ is also obvious. We will write $\text{red}(\Phi) = \text{red}_\Phi(\Phi)$, if we reduce all clauses and cubes of the AQBF Φ according to its prefix.

As defined by Kleine Büning et al. [20], a Q-resolution (Q-consensus) proof π from a QCNF (QDNF) or AQBF Φ of a clause (cube) C is a sequence of clauses (cubes) $\pi = (C_i)_{i=1}^m$, such that $C_m = C$ and for each C_i one of the following holds:

- *Axiom:* $C_i \in \mathfrak{C}(\Phi)$ (resp. $C_i \in \mathfrak{D}(\Phi)$);
- *Resolution:* $C_i = C_j \overset{x}{\bowtie} C_k$ with x existential (universal), $j, k < i$, and C_i non-tautological (non-contradictory);
- *Reduction:* $C_i = \text{red}^\forall_\Phi(C_j)$ (resp. $C_i = \text{red}^\exists_\Phi(C_j)$) for some $j < i$.

We call C the *root* of π. In [2], an extension of Q-resolution (Q-consensus) proofs to long-distance Q-resolution (long-distance Q-consensus) proofs was introduced by replacing the resolution rule by

- *Resolution (long-distance):* $C_i = C_j \overset{x}{\bowtie} C_k$ with x existential (universal) and $j, k < i$. The resolvent C_i is allowed to contain tautologies such as $u \vee \bar{u}$ (resp. $u \wedge \bar{u}$), if u is universal (existential). If there is such a universal (existential) $u \in \text{var}(C_j) \cap \text{var}(C_k)$, then we require $x <_\Phi u$.

Furthermore, a Q-resolution (Q-consensus) or long-distance Q-resolution (long-distance Q-consensus) proof π from Φ of the empty clause $(\bot)$ (the empty cube $[\top]$) is called a *refutation* (*certificate*) of Φ. In that case, Φ is called *false* (*true*). We will sometimes interpret π as a set of clauses (or cubes).

A proof system S *p-simulates* a system S', if every S' proof can be transformed in polynomial time into an S proof of the same formula.

3 Our QCDCL Models

To analyse the complexity of QCDCL procedures, we need to fully formalise them as proof systems. This approach was initiated in [5] and [9], and we follow that framework. We will only sketch this formalization here.

We store all relevant information of a QCDCL run in *trails*. Since QCDCL uses several runs and potentially also restarts, a QCDCL proof will typically consist of many trails.

Definition 1 (trails) A *trail* $\mathcal{T}$ for a QCNF or AQBF Φ is a (finite) sequence of pairwise distinct literals from Φ, including the empty literals $\bot$ and $\top$. In general, a trail has the form

$$\mathcal{T} = (p_{(0,1)}, \ldots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \ldots, p_{(1,g_1)}; \ldots; \mathbf{d_r}, p_{(r,1)}, \ldots, p_{(r,g_r)}), \tag{1}$$

where the d_i are *decision literals* and $p_{(i,j)}$ are *propagated literals*. A trail $\mathcal{T}$ has *run into a conflict* if $\bot \in \mathcal{T}$ or $\top \in \mathcal{T}$.

Decision literals are written in **boldface**. We use a semicolon before each decision to mark the end of a decision level. If one of the empty literals $\bot$ or $\top$ is contained in $\mathcal{T}$, then it has to be the last literal $p_{(r,g_r)}$. In this case, we say that $\mathcal{T}$ has *run into a conflict*.

Trails can be interpreted as non-tautological sets of literals, and therefore as (partial) assignments. We write $x <_{\mathcal{T}} y$ if $x, y \in \mathcal{T}$ and x is left of y in $\mathcal{T}$. Furthermore, we write $x \leq_{\mathcal{T}} y$ if $x <_{\mathcal{T}} y$ or $x = y$.

As trails are produced gradually from left to right in an algorithm, we define $\mathcal{T}[i, j]$ for $(i, j) \in (\{0, \ldots, r\} \times \{0, \ldots, g_i\}) \backslash \{(0, 0)\}$ as the subtrail that contains all literals from $\mathcal{T}$ up to (and excluding) $p_{(i,j)}$ (resp. d_i, if $j = 0$) in the same order. Intuitively, $\mathcal{T}[i, j]$ is the state of the trail before we assigned the literal at the point $[i, j]$ (which is $p_{(i,j)}$ or d_i).

Each propagated literal $p_{(i,j)} \in \mathcal{T}$ belongs to an *antecedent clause* (if $p_{(i,j)}$ is existential) or an *antecedent cube* (if $p_{(i,j)}$ is universal) from Φ, which we call $\text{ante}_{\mathcal{T}}(p_{(i,j)})$. At the point where $p_{(i,j)}$ was propagated in $\mathcal{T}$, we need that $\text{ante}_{\mathcal{T}}(p_{(i,j)})$ had become unit, hence $\text{red}_{\Phi}(\text{ante}_{\mathcal{T}}(p_{(i,j)})|_{\mathcal{T}[i,j]}) = (p_{(i,j)})$ if $p_{(i,j)}$ is existential, and $\text{red}_{\Phi}(\text{ante}_{\mathcal{T}}(p_{(i,j)})|_{\mathcal{T}[i,j]}) = [\bar{p}_{(i,j)}]$, if $p_{(i,j)}$ is universal.

Trails are not generated arbitrarily, as they follow some further rules in practice, such as propagations should not be skippable. We denote trails with these conditions as *natural trails*.

Definition 2 (natural trails) We call $\mathcal{T}$ a *natural trail* for the formula Φ, if for each $i \in \{1, \ldots, r\}$ the formula $\text{red}(\Phi|_{\mathcal{T}[i,0]})$ does not contain unit or empty constraints. Furthermore, the formula $\Phi|_{\mathcal{T}[i,j]}$ must not contain empty constraints for each $i \in \{1, \ldots, r\}$, $j \in \{1, \ldots, g_i\}$, except $[i, j] = [r, g_r]$. Intuitively, we require that decisions are only made if and only if there are no more propagations on the same decision level left. Also, conflicts must be detected immediately if there are any.

We state some general facts about trails and antecedent clauses/cubes one should keep in mind.

Remark 1 Let $\mathcal{T}$ be a trail, $\ell \in \mathcal{T}$ a propagated literal and $A := \text{ante}_{\mathcal{T}}(\ell)$.

- If ℓ is existential, then $\ell \in A$ and for each literal $x \in A$ with $x \neq \ell$ we need $\bar{x} <_{\mathcal{T}} \ell$.
- If ℓ is universal, then $\bar{\ell} \in A$ and for each literal $u \in A$ with $u \neq \bar{\ell}$ we need $u <_{\mathcal{T}} \ell$.

An essential element of QCDCL is clause and cube learning. This guarantees to make 'progress' after each trail (at least under some conditions that we will specify later).

Definition 3 (learnable constraints) Let $\mathcal{T}$ be a trail for Φ of the form (1) with $p_{(r,g_r)} \in \{\bot, \top\}$. Starting with $\text{ante}_{\mathcal{T}}(\bot)$ (resp. $\text{ante}_{\mathcal{T}}(\top)$) we reversely resolve over the antecedent clauses (cubes) that were used to propagate the existential (universal) variables, until we stop at some arbitrarily chosen point. The clause (cube) we so derive is a *learnable constraint*. Note that clause (cube) learning will skip propagations caused by cubes (clauses) and interpret the corresponding literal in the trail as a decision. Universal (existential) reduction will be performed whenever applicable (basically before and after each resolution step in the learning process). We denote the set of learnable constraints by $\mathfrak{L}(\mathcal{T})$.

We can also learn cubes from trails that did not run into conflict. If $\mathcal{T}$ is a total assignment of the variables from Φ, then we define the set of learnable constraints as the set of cubes $\mathfrak{L}(\mathcal{T}) := \{\text{red}^{\exists}_{\Phi}(D) |\ D \subseteq \mathcal{T} \text{ and } D \text{ satisfies } \mathfrak{C}(\Phi)\}$.

In QCDCL, our goal is to make 'progress' in each run/trail. Thus, we have to ensure that we can always learn new clauses or cubes from a constructed trail. Since we want to work with QCDCL models that do not necessarily follow the prefix order for decision making, it is not guaranteed that we can even learn new constraints from each trail. As we will show later, we need the following condition to prevent such a situation, which could easily lead to a loop in practical solving.

Definition 4 A trail $\mathcal{T}$ for a formula Φ fulfils the *New Constraint Condition* (*NCC* for short), if for each decision d_i the formula $\text{red}(\Phi|_{\mathcal{T}[i,0]\cup\{d_i\}})$ does not contain the empty clause or cube.

Intuitively, this means that a decision must not lead to a conflict immediately. It will become clear later, why we can always find a decision that does not violate the NCC. In fact, classical QCDCL automatically fulfils this condition.

We will now formally define our four QCDCL proof systems, namely QCDCL, $\mathsf{QCDCL}^{\textsc{Any}}$, $\mathsf{QCDCL}^{\textsc{Uni-Any}}$, and $\mathsf{QCDCL}^{\textsc{Exi-Any}}$.

Definition 5 (QCDCL proof systems) Let S be one of QCDCL, $\mathsf{QCDCL}^{\textsc{Any}}$, $\mathsf{QCDCL}^{\textsc{Uni-Any}}$, $\mathsf{QCDCL}^{\textsc{Exi-Any}}$. An S proof ι from a QCNF $\Phi = \mathcal{Q} \cdot \phi$ of a clause or cube C is a (finite) sequence of triples

$$\iota := [(\mathcal{T}_i, C_i, \pi_i)]_{i=1}^{m},$$

where $C_m = C$, each $\mathcal{T}_i$ is a trail for Φ_i that fulfils the NCC, each $C_i \in \mathfrak{L}(\mathcal{T}_i)$ is one of the constraints we can learn from each trail and π_i is the long-distance Q-resolution or long-distance Q-consensus proof from Φ_i of C_i we obtain by performing the steps in Definition 3. If necessary, we set $\pi_i := \emptyset$. We will denote the set of trails in ι as $\mathfrak{T}(\iota)$.

The AQBFs Φ_i are defined as follows:

$$\Phi_1 := \mathcal{Q} \cdot (\mathfrak{C}(\Phi) \vee \emptyset)$$

and

$$\Phi_{j+1} := \begin{cases} \mathcal{Q} \cdot \big((\mathfrak{C}(\Phi_j) \wedge C_j) \vee \mathfrak{D}(\Phi_j)\big) & \text{if } C_j \text{ is a clause,} \\ \mathcal{Q} \cdot \big(\mathfrak{C}(\Phi_j) \vee (\mathfrak{D}(\Phi_j) \vee C_j)\big) & \text{if } C_j \text{ is a cube,} \end{cases}$$

for $j = 1, \ldots, m-1$.

The four systems differ from each other in the way decisions are made. We extend the definition of natural trails with decision rules that belong to the corresponding system S. A natural trail $\mathcal{T}$ for a formula Ψ that fulfils the following rules for S is called a *natural S trail*:

- QCDCL: For each decision d_i we have that $\text{lv}_{\Psi|_{\mathcal{T}[i,0]}}(d_i) = 1$. I.e., decisions are level-ordered.
- QCDCL$^{\text{ANY}}$: Decisions can be made arbitrarily as long as the NCC is fulfilled.
- QCDCL$^{\text{UNI-ANY}}$: An existential decision d_i can only be made if all universal variables that are quantified left of d_i were already assigned in $\mathcal{T}$. Universal decisions can be made in any order as long as the NCC is fulfilled.
- QCDCL$^{\text{EXI-ANY}}$: A universal decision d_i can only be made if all existential variables that are quantified left of d_i were already assigned in $\mathcal{T}$. Existential decisions can be made in any order as long as the NCC is fulfilled.

After each trail, we will backtrack to some arbitrary previous point in the trail and continue to decide or propagate from that point.

If $C = C_m = (\bot)$, then ι is called an *S refutation* of Φ. If $C = C_m = [\top]$, then ι is called an *S certificate* of Φ. The proof ends once we have learned $(\bot)$ or $[\top]$.

If C is a clause, we can stick together the long-distance Q-resolution derivations from $\{\pi_1, \ldots, \pi_m\}$ and obtain a long-distance Q-resolution proof from Φ of C, which we call $\mathfrak{R}(\iota)$. Similarly, if C is a cube, we can stick together the long-distance Q-consensus derivations and obtain a long-distance Q-consensus proof $\mathfrak{R}(\iota)$ from Φ of C.

The *size* of ι is defined as $|\iota| := \sum_{i=1}^{m} |\mathcal{T}_i|$. Obviously, we have $|\mathfrak{R}(\iota)| \in \mathcal{O}(|\iota|)$.

Our formalisation above is based on [5, 9]. However, since in the present paper cube learning is always included, our plain model QCDCL now includes clause *and* cube learning (while in [5, 9], QCDCL denotes a system with just clause learning, but without learning cubes).

The concept behind the two models QCDCL$^{\text{UNI-ANY}}$ and QCDCL$^{\text{ANY}}$ was already introduced in [5] (albeit defined slightly differently, they were called QCDCL$^{\text{ASS-R-ORD}}_{\text{RED}}$ and QCDCL$^{\text{ANY-ORD}}_{\text{NO-RED}}$ in those papers). However, since we include cube learning now, our models here match practical solving much better.

Remark 2 In QCDCL, decision making can never violate the NCC if we create the trails 'naturally' (i.e., decisions are only made if and only if there are no more propagations on the same decision level left, and conflicts must be detected immediately if there are any).

Proof If we made a level-ordered decision d_i and get a conflict immediately afterwards on a clause (w.l.o.g.) $C = \text{ante}_{\mathcal{T}}(\bot)$, then d_i must have been existential (otherwise we could have reduced $\bar{d}_i$ in order to get a conflict before) and we would need $\bar{d}_i \in C$. Furthermore, there must exist at least one universal literal $u \in C$ that was reduced while propagating $\bot$, otherwise C would have been a unit clause before we made the decision d_i. However, the reduction must have been blocked before deciding d_i, otherwise we could have used this reduction to propagate $\bar{d}_i$. That means u was quantified left of d_i, but this is a contradiction since the decision d_i was level-ordered. Hence, deciding the leftmost unassigned literal according to the prefix order, we will never violate the NCC. ◻

We still have to make sure to fulfil NCC when backtracking, though. We will explain later how this is achieved.

The next result states simulations between systems, cf. Fig. 1. They all follow by definition.

Proposition 1 *Each* QCDCL *proof is also a* QCDCL$^{\textit{UNI-ANY}}$ *and* QCDCL$^{\textit{EXI-ANY}}$ *proof, and each* QCDCL$^{\textit{UNI-ANY}}$ *or* QCDCL$^{\textit{EXI-ANY}}$ *proof is also a* QCDCL$^{\textit{ANY}}$ *proof.*

4 Learning Asserting Constraints

We recall the notion of an asserting clause (or cube). The concept originates from SAT solving [25], but directly lifts to QBF [13, 38]. Intuitively, asserting constraints are learnable constraints that become unit after backtracking. We give a more liberal definition as we do not refer to specific asserting constraints (such as UIP clauses).

Definition 6 (asserting constraints) Let $\mathcal{T}$ be a trail for a QCNF Φ that contains r decision literals. A clause (cube) $C \in \mathfrak{L}(\mathcal{T})$ is called *asserting*, if there exists some point $[i, j]$ such that $\text{red}_\Phi^\forall(C|_{\mathcal{T}[i,j]})$ is a unit clause (resp. $\text{red}_\Phi^\exists(C|_{\mathcal{T}[i,j]})$ is a unit cube). Furthermore, we require that we backtrack by at least one decision level, i.e., $i < r$ or $j = 0$.

Learning asserting clauses might be advantageous as it guarantees new unit propagations after backtracking to a suitable point. In addition, asserting clauses are always new.

Proposition 2 *If $\mathcal{T}$ is a trail in a* QCDCL$^{\text{UNI-ANY}}$ *(resp.* QCDCL$^{\text{EXI-ANY}}$*) proof of a formula Φ, and if $\bot \in \mathcal{T}$ (resp. $\top \in \mathcal{T}$), then there exists a new asserting or empty clause (cube) $C \in \mathfrak{L}(\mathcal{T})$.*

Furthermore, if C is non-empty, there exists a point $[i, j]$ in the trail to which we can backtrack after learning C such that the NCC continues to hold.

Proof We will show the case for conflicts on clauses for QCDCL$^{\text{UNI-ANY}}$ proofs, the QCDCL$^{\text{EXI-ANY}}$ case is completely dual.

Let the trail $\mathcal{T}$ look like

$$\mathcal{T} = (p_{(0,1)}, \ldots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \ldots, p_{(1,g_1)}; \ldots; \mathbf{d_r}, p_{(r,1)}, \ldots, p_{(r,g_r)}).$$

Then the sequence of learnable clauses is

$$\mathfrak{L}(\mathcal{T}) = (C_{(r,g_r)}, \ldots, C_{(r,1)}, \ldots, C_{(1,g_1)}, \ldots, C_{(1,1)}, C_{(0,g_0)}, \ldots, C_{(0,1)}).$$

We can assume that there exists at least one existential decision literal d_i such that $\bar{d}_i$ is contained in some $C \in \mathfrak{L}(\mathcal{T})$. Otherwise, the rightmost clause in $\mathfrak{L}(\mathcal{T})$ is empty since it contains negated decisions or universal literals only, which will be reduced to the empty clause ($\bot$).

Let $k \in \{1, \ldots, r\}$ be maximal such that an existential $\bar{d}_k$ is contained in some clause from $\mathfrak{L}(\mathcal{T})$. Let $p_{(\ell,m)} \in \mathcal{T}$ be the propagated (non-empty) literal directly right of d_k in $\mathcal{T}$ and set $D := C_{(\ell,m)}$. Note that $p_{(\ell,m)}$ does not need to be on the same decision level as d_k. Such a $p_{(\ell,m)}$ must exist by the NCC. We will show that D is asserting.

We consider the trail $\mathcal{T}$ at the point $[k, 0]$, that means right before d_k was decided. We will prove that $E := \text{red}_\Phi^\forall(D|_{\mathcal{T}[k,0]}) = (\bar{d}_k)$.

If there is an existential literal $\bar{d}_k \neq y \in E$, then $\bar{y}$ cannot have been assigned in $\mathcal{T}[k, 0]$, hence we have $d_k <_\mathcal{T} \bar{y}$. But that means $\bar{y}$ had to be a decision, otherwise it would have been resolved away during clause learning. But this is a contradiction to the maximality of k. We conclude that such a y cannot exist.

Let us now assume there is a universal literal $u \in E$. Then we need $u <_\Phi d_k$ since it was not reduced during clause learning. But $\mathcal{T}$ was a trail in a QCDCL$^{\text{UNI-ANY}}$ proof, hence $\text{lv}_{\Phi|_{\mathcal{T}[k,0]}}(d_k) = 1$ and therefore $\bar{u} \in \mathcal{T}[k, 0]$. Then we get $u \notin E$, contradiction. Thus such a u cannot exist, and E is in fact a unit clause.

We can backtrack to the point $[k, 0]$ (i.e., before we made the decision d_k) and will not hurt the NCC since the only new clause we have learned can only propagate the non-empty literal $\bar{d}_k$.

At the end, we have to show that D is new. In fact, if D was already known, we would get a conflict directly after deciding d_k, which would violate the NCC. Thus, D must be a new clause. □

A similar result holds for the any-order model, albeit with the difference that we might not be able to learn asserting constraints. But at least we can guarantee to learn a new clause/cube.

Proposition 3 *If $\mathcal{T}$ is a trail in a* QCDCL$^{\text{ANY}}$ *proof for a formula Φ, that has run into a conflict or in which we assigned all variables, then $\mathfrak{L}(\mathcal{T})$ contains a new clause or cube C that is not contained in Φ.*

Further, if C is non-empty, there exists a point $[i, j]$ in the trail to which we can backtrack after learning C such that the NCC continues to hold.

Proof Case 1: $\mathcal{T}$ runs into a conflict.

Let the trail $\mathcal{T}$ look like

$$\mathcal{T} = (p_{(0,1)}, \ldots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \ldots, p_{(1,g_1)}; \ldots; \mathbf{d_r}, p_{(r,1)}, \ldots, p_{(r,g_r)}).$$

Then the sequence of learnable clauses is

$$\mathfrak{L}(\mathcal{T}_i) = (C_{(r,g_r)}, \ldots, C_{(r,1)}, \ldots, C_{(1,g_1)}, \ldots, C_{(1,1)}, C_{(0,g_0)}, \ldots, C_{(0,1)}).$$

By the NCC, we have that $g_r > 1$. We will show that $C_{(r,1)}$ (which is the clause/cube we get after resolving over $p_{(r,g_r-1)}, \ldots, p_{(r,1)}$) is a new clause (cube).

Assume not. Consider the restricted clause (cube) $E := C_{(r,1)}|_{\mathcal{T}_i[r,1]}$. Suppose that there is an existential (universal) literal $x \in E \subseteq C_{(r,1)}$. That means that x is contained in at least one antecedent clause (cube) after (and including) $p_{(r,1)}$. In particular, we need $\bar{x} \in \mathcal{T}$ (resp. $x \in \mathcal{T}$). Because x is still contained in $C_{(r,1)}$, it cannot have been resolved away during learning, hence $\bar{x} \in \mathcal{T}[r, 1]$ (resp. $x \in \mathcal{T}[r, 1]$). This is a contradiction to the definition of E.

We conclude that E can only contain universal (existential) literals, hence $\text{red}_{\Phi}^{\forall}(E) = (\bot)$ (resp. $\text{red}_{\Phi}^{\exists}(E) = [\top]$). But then we would have got a conflict directly after d_r, which is impossible by the NCC. That means that $C_{(r,1)}$ must a new clause (cube).

We can backtrack to the point where we undo the rightmost existential (universal) literal in $\mathcal{T}$ that is contained in $C_{(r,1)}$. At this point, $C_{(r,1)}$ will not become unit since it still includes at least this one literal.

Case 2: $\mathcal{T}$ does not run into a conflict, but we assigned all variables in $\mathcal{T}$.

Assume that we cannot find such a C. Then there exists a $C \in \mathfrak{L}(\mathcal{T})$ such that $C \in \mathfrak{D}(\Phi_a)$, where Φ_a is the current formula for $\mathcal{T}$. That means there exists a cube $E \subseteq \mathcal{T}$ such that $\text{red}_{\Phi_a}^{\exists}(E) = C$ and E satisfies $\mathfrak{C}(\Phi_a)$. In particular, we have $\text{red}_{\Phi_a}^{\exists}(C|_{\mathcal{T}}) = [\top]$, which means that $\mathcal{T}$ should have run into a conflict. This is a contradiction.

We can backtrack to the point where we undo the rightmost universal literal in $\mathcal{T}$, that is contained in C. Then C will just propagate this universal literal and not an empty one. If this point is on the last decision level, we can alternatively restart. □

Remark 3 To illustrate the importance of the NCC, we give an example of a QCDCL$^{\text{ANY}}$ trail—violating the NCC—from which we cannot learn a new clause. Consider the trail $\mathcal{T} = (\mathbf{x}, \bot)$ for the false QCNF $\forall u \exists x \cdot (u \vee x) \wedge (u \vee \bar{x}) \wedge (\bar{u} \vee x) \wedge (\bar{u} \vee \bar{x})$. The trail violates the NCC, as we got a conflict directly after the decision x. The only learnable clause is $\text{ante}_{\mathcal{T}}(\bot) = \bar{u} \vee \bar{x}$, which is obviously already known.

Another example illustrates the case where we can learn a new clause but no asserting clause. Let the trail be $\mathcal{U} := (\mathbf{x}, y; \mathbf{u}, \bar{z}, \bot)$ for the false QCNF $\forall u \exists x, y, z \cdot (\bar{x} \vee y) \wedge (x \vee y) \wedge (u \vee \bar{y} \vee \bar{z}) \wedge (\bar{u} \vee \bar{y} \vee \bar{z}) \wedge (u \vee \bar{y} \vee z) \wedge (\bar{u} \vee \bar{y} \vee z)$. There are two new clauses we

	asserting clauses	only new clauses
asserting cubes	QCDCL	$\mathsf{QCDCL}^{\textsf{Exi-Any}}$
only new cubes	$\mathsf{QCDCL}^{\textsf{Uni-Any}}$	$\mathsf{QCDCL}^{\textsf{Any}}$

Fig. 2 Overview of guaranteed learnable constraints after a trail conflict in the corresponding models

could learn: $\bar{u} \vee \bar{y}$ or $\bar{u} \vee \bar{x}$. None of the two can become unit after backtracking since we used the decision heuristic for $\mathsf{QCDCL}^{\textsf{Exi-Any}}$, although we followed the NCC.

As a special case we obtain for our base model QCDCL the following situation.

Corollary 4 *[Folklore, cf. [23]] If $\mathcal{T}$ is a trail in a* QCDCL *proof for a formula Φ, that has run into a conflict, then $\mathfrak{L}(\mathcal{T})$ contains an asserting or empty clause or cube. If $\mathcal{T}$ has not run into a conflict, but we have assigned all variables in $\mathcal{T}$, then $\mathfrak{L}(\mathcal{T})$ contains at least a new cube C.*

Furthermore, if C is non-empty, there exists a point $[i, j]$ in the trail to which we can backtrack after learning C such that the NCC continues to hold.

Figure 2 provides an overview of the four systems and their ability to learn asserting clauses and cubes. As a consequence of always learning new constraints, we infer that our models are all complete and terminating proof methods.

Theorem 5 QCDCL, $\mathsf{QCDCL}^{\textsf{Any}}$, $\mathsf{QCDCL}^{\textsf{Uni-Any}}$ *and* $\mathsf{QCDCL}^{\textsf{Exi-Any}}$ *are sound and complete proof systems.*[6] *Additionally, as long as we follow the rules of decision making (especially the NCC), we will always learn the empty clause or cube at some point, no matter what decisions were made.*

Proof By Propositions 2 and 3 as well as Corollary 4 we conclude that from each trail (that has either run into a conflict or assigned all variables) we can always learn a new clause or cube. Note that these results have to be interpreted in the context of Proposition 1.

Since a given formula only consists of finitely many variables, we can only learn finitely many new clauses and cubes. We finish the proof as soon as we learn the empty clause or cube, which will happen at some point. Therefore all four systems are complete.

The soundness results from the fact that from each QCDCL, $\mathsf{QCDCL}^{\textsf{Any}}$, $\mathsf{QCDCL}^{\textsf{Uni-Any}}$ and $\mathsf{QCDCL}^{\textsf{Exi-Any}}$ proof ι we can extract a long-distance Q-resolution or long-distance Q-consensus proof $\mathfrak{R}(\iota)$ for the same formula. □

5 Separations of QCDCL Systems

In this section, we will exponentially separate our three new models—where decisions do not necessarily follow the prefix order—from the plain model QCDCL.

We will use the *gauge lower bound technique*, introduced in [9], which we will first review. This technique works on Σ_3^b QCNFs. To ease notation, we will assume that prefixes of Σ_3^b QCNFs have the form $\exists X \forall U \exists T$, for sets of literals X, U, T, and we will use the notions of X-, U- and T-variables and -literals. Further, we define certain types of clauses:

[6] I.e., they are proof systems for the language of false and true QBFs in the setting of [12]. Technically, in order not to trivialise the notion of such a proof system, we could consider proof systems for the language L of the marked union of true and false QBFs, i.e., $L = \{0\Phi \mid \Phi \text{ is a false QBF}\} \cup \{1\Phi \mid \Phi \text{ is a true QBF}\}$. In this way, L is still PSPACE complete.

- X-clauses consist of X-literals only (analogously we define U-clauses and T-clauses),
- XT-clauses consist of at least one X- and at least one T-literal, but no U-literals,
- XUT-clauses consist of at least one X-, U- and T-literal, respectively.

The gauge lower bound method works for a specific class of Σ_3^b QCNFs with the *XT-property*.

Definition 7 ([5]) We say that Φ fulfills the *XT-property*, if $\mathfrak{C}(\Phi)$ contains no XT-clauses, no T-clauses that are unit (or empty) and no two T-clauses from $\mathfrak{C}(\Phi)$ are resolvable.

The XT-property extends to entire QCDCL proofs, as stated in the next lemma.

Lemma 6 *([5]) If Φ is a Σ_3^b QCNF that fulfills the XT-property, then it is not possible to derive XT-clauses or new T-clauses via* long-distance Q-resolution *from Φ.*

The gauge lower-bound method from [9] uses the next two notions of fully reduced and primitive proofs (they were implicit in [9] and stated explicitly in [4]).

Definition 8 (fully reduced proofs [4, 9]) A long-distance Q-resolution refutation π of a QCNF Φ is *fully reduced*, if for each clause $C \in \pi$ that contains universal literals that are reducible, the reduction step is performed immediately and C is not used otherwise in the proof.

Fully reduced proofs are not much of a limitation. In fact, all long-distance Q-resolution proofs that we extract from a QCDCL run are already fully reduced by default. Also, we can always shorten a given long-distance Q-resolution proof by making it fully reduced.

Definition 9 (primitive proofs [4, 9]) A long-distance Q-resolution proof π from a Σ_3^b formula is *primitive*, if there are no two XUT-clauses in π that are resolved over an X-variable.

Unlike the fully reduced property, not all proofs extracted from QCDCL are primitive, in general.

Our lower bound method will not work for all QCDCL proofs, but needs *fully reduced primitive* Q-resolution proofs, which are better suited for a proof-complexity analysis. Later, the challenge will be to show that certain extracted proofs from QCDCL are primitive. Note that fully reduced primitive long-distance Q-resolution proofs are always Q-resolution proofs.

The main measure for the lower bound technique is the *gauge* of a formula, defined in [9].

Definition 10 ([9]) Let Φ be a Σ_3^b QCNF with prefix $\exists X \forall U \exists T$. We define W_Φ as the set of all Q-resolution proofs π from Φ of X-clauses C_π, such that π consists of resolutions over T-literals and reductions only. We define

$$\text{gauge}(\Phi) := \min\{|C_\pi| \ : \ C_\pi \ \text{ is the root of some } \pi \in W_\Phi\}.$$

Intuitively, gauge(Φ) is the minimal number of X-literals that are piled up during the process of deriving an X-clause without using resolutions over X-literals. In other words: to get rid of all T-literals from Φ, we have to pile up at least gauge(ϕ) many different X-literals.

All notions we introduced so far are combined into the following lower bound method:

Theorem 7 ([9]) *Each fully reduced primitive* Q-resolution *refutation of a Σ_3^b QCNF Φ that fulfils the XT-property has size $2^{\Omega(gauge(\Phi))}$.*

Proof We refer to the notion of quasi level-ordered proofs from [9] (there is no need to define this here). In that paper, it is explained how we can transform a QCDCL refutation of a formula that fulfils the XT-property into a quasi level-ordered Q − resolution refutation in polynomial time via an algorithm. However, the input proof does not need to be a QCDCL proof necessarily. It suffices that this proof is fully reduced and it does not contain an X-resolution over two XUT-clauses (these are the only two properties that were needed for proving Theorem 2 in [9]). In other words, this algorithm can be used to transform fully reduced primitive Q-resolution refutations into quasi level-ordered Q-resolution refutations in polynomial times.

The lower bound then follows from Theorem 5 of [9]. □

Our goal is to find formulas that separate QCDCL from $\mathsf{QCDCL}^{\textsc{Uni-Any}}$ and QCDCL from $\mathsf{QCDCL}^{\textsc{Exi-Any}}$, respectively. We start with the latter.

5.1 Separation on True Formulas

The advantage of $\mathsf{QCDCL}^{\textsc{Exi-Any}}$ (compared to QCDCL) is to decide existential literals out of order while still learning asserting cubes. Since cubes are important for certificates of true formulas, it makes sense to use true QBFs for the separation.

First, we discuss two generic modifications for QBFs. The twin construction doubles all universal variables. For all clauses with universal variables a copy is created in the twin variables.

Definition 11 (twin formulas) Let $\Phi = \exists X \forall U \exists T \cdot \mathfrak{C}(\Phi)$ be a QCNF. Let $U = \{u_1, \dots, u_m\}$ and let $v_1, \dots, v_m$ be variables not occuring in Φ. Then the *twin formula* of Φ is the QCNF $\mathtt{Twin}\Phi$ defined as

$$\mathtt{Twin}\Phi := \exists X \forall (U \cup \{v_1, \dots, v_m\}) \exists T \cdot \mathfrak{C}(\Phi) \wedge \bigwedge_{C \in \mathfrak{C}(\Phi)} C[u_1/v_1, \dots, u_m/v_m],$$

where u_i/v_i indicates that all occurrences of u_i are substituted by v_i.

The second modification is the *reversion* of a formula.

Definition 12 If $\Phi = \mathcal{Q}_1 V_1 \dots \mathcal{Q}_k V_k \cdot \bigwedge_{j=1}^{m} C_j$ is a QCNF with $\mathcal{Q}_i \in \{\exists, \forall\}$ and disjoint sets of variables V_i for $i = 1, \dots, k$, then the *reversion* $\mathrm{Rev}(\Phi)$ of Φ is the QCNF

$$\mathcal{Q}'_1 V_1 \dots \mathcal{Q}'_k V_k \forall w \exists c_1, \dots, c_m \cdot (\bar{c}_1 \vee \dots \vee \bar{c}_m) \wedge \bigwedge_{j=1}^{m} \bigwedge_{\ell \in C_j} (\bar{\ell} \vee w \vee c_j) \wedge (\bar{\ell} \vee \bar{w} \vee c_j)$$

where $\mathcal{Q}'_i = \forall$ if $\mathcal{Q}_i = \exists$, and $\mathcal{Q}'_i = \exists$ if $\mathcal{Q}_i = \forall$, and $w, c_1, \dots, c_m$ are new variables not contained in Φ.

It is easy to prove that there exists a duality between the truth values of Φ and $\mathrm{Rev}(\Phi)$.

Lemma 8 *If Φ is a QCNF, then $Rev(\Phi)$ is true if and only if Φ is false.*

Proof Case 1: Φ is false.

Then there exists a winning strategy for the universal player of Φ. We will show that $\mathrm{Rev}(\Phi)$ has an existential winning strategy.

The existential player for $\mathrm{Rev}(\Phi)$ can just follow the universal winning strategy for Φ. That means there is at least one clause $C_j \in \mathfrak{C}(\Phi)$ falsified by the total assignment that consists of the assignment from the universal player and the corresponding response determined by the winning strategy. Then the clauses $\bar{\ell} \vee w \vee c_j$ and $\bar{\ell} \vee \bar{w} \vee c_j$ for each $\ell \in C_j$ are satisfied (for this particular j) by this assignment. Note that it does not matter how w was assigned. Therefore, the existential player for this modified formula can just set c_j to true and all the other c_i to false.

Case 2: Φ is true.

This case is analogous to Case 1. The universal player for the modified $\mathrm{Rev}(\Phi)$ version follows the existential winning strategy for Φ. Then the universal player can set w to true (it does not matter, actually). For each $j \in \{1, \ldots, m\}$ the clause C_j is satisfied, hence at least one literal $\ell \in C_j$ is set to true. Therefore for each c_j, the clause $\bar{\ell} \vee \bar{w} \vee c_j$ becomes the unit clause (c_j) at some point under the assignment determined by the strategy. That means the existential player for $\mathrm{Rev}(\Phi)$ has to set each c_j to true, falsifying the clause $\bar{c}_1 \vee \ldots \vee \bar{c}_m$.

We now have constructed a universal winning strategy for $\mathrm{Rev}(\Phi)$. □

We will use the reversion to lift hardness from false to true QCNFs. To verify a true formula, we need to create a proof using cubes. We will show that $\mathrm{Rev}(\Phi)$ is designed such that its initial cubes are basically the negated axiom clauses of Φ. Thus, a certificate of $\mathrm{Rev}(\Phi)$ can be transformed into a refutation of Φ.

Our reversion was inspired by the notion of the *negation* from [19]. The only change we made is adding the variable w. We did this to prevent a direct connection between an X- or U-block and an auxiliary variable c_j from the last block. Our lower bound technique is based on the fact that on certain formulas we cannot have direct connections (hence: cannot directly propagate) between outer and inner quantifier blocks. The added variable w helps to maintain this property.

The next two results shows how we can transform certificates of $\mathrm{Rev}(\Phi)$ into refutations of Φ by interpreting the cubes from the certificate as negated clauses of a refutation.

Lemma 9 *Let $\Phi = \mathcal{Q} \cdot \bigwedge_{j=1}^{m} C_j$ be a QCNF and let σ be an assignment that satisfies $\mathfrak{C}(Rev(\Phi))$. Then there exists some $C \in \mathfrak{C}(\Phi)$ with $\overline{C} \subseteq \sigma$.*

Proof Since we have to satisfy the clause $(\bar{c}_1 \vee \ldots \vee \bar{c}_m)$, there is some $j \in \{1, \ldots, m\}$ with $\bar{c}_j \in \sigma$. Then the clauses $\bar{\ell} \vee w \vee c_j$ and $\bar{\ell} \vee \bar{w} \vee c_j$ have to be satisfied for each $\ell \in C_j$. We do not need to assign w, but we need to set each ℓ to false, hence $\overline{C}_j \subseteq \sigma$. □

Proposition 10 *If Φ is a false QCNF and ρ is a* long-distance Q-consensus *certificate of $Rev(\Phi)$, then ρ can be transformed into a fully reduced* long-distance Q-resolution *refutation π of Φ with $|\pi| \leq |\rho|$.*

More precisely, for each clause $C \in \pi$ there is a cube $C' \in \rho$ with $\overline{C} \subseteq C'$. Furthermore, for each two clauses C, D that are resolved in π, the corresponding cubes C', D' are resolved in ρ, as well.

Proof By Lemma 9, for each initial cube $D \in \rho$ there is a clause $C \in \mathfrak{C}(\Phi)$ with $\mathrm{red}^{\exists}_{\mathrm{Rev}(\Phi)}(\overline{C}) \subseteq D$ (note that the assignment from Lemma 9 can still be reduced). We substitute each initial cube D with its corresponding $\mathrm{red}^{\exists}_{\mathrm{Rev}(\Phi)}(\overline{C})$ and shorten the proof, if necessary (i.e., delete redundant resolutions and reductions). We receive a subproof $\pi' \subseteq \rho$, that is still a certificate.

After that, we negate all cubes in π' and receive a proof π that consists of clauses. If we interpret π as a proof for Φ (or $\mathrm{red}(\Phi)$ to be precise), all resolutions and reductions are still sound because the quantifiers were flipped, as well.

We can assume that in π we will reduce as soon as possible, otherwise we could shorten the proof even more. Obviously, the last clause in π has not received any additional literals, therefore π is a long-distance Q-resolution refutation of Φ. $\square$

For our next results, we need an even stronger property than the XT-property: We require, that clauses from the formula contain at least one U- and T-literal.

Lemma 11 *If Φ is a Σ_3^b QCNF, in which all clauses contain at least one U- and T-literal, then Φ fulfils the XT-property.*

Proof Obviously, Φ does not contain any XT- or T-clauses and therefore the XT-property is fulfilled. $\square$

We combine the results above and obtain a new lower bound technique for true formulas, which builds on the gauge technique for false formulas.

Theorem 12 *Let Φ be a false Σ_3^b. Additionally, let all clauses $C \in \mathfrak{C}(\Phi)$ contain at least one U- and one T-literal. If the QCNF $\mathtt{Twin}\Phi$ needs fully reduced primitive* Q-resolution *refutations of size s, then* QCDCL *certificates for Rev$(\mathtt{Twin}\Phi)$ also need size s.*

Proof Let ι be a QCDCL certificate for Rev$(\mathtt{Twin}\Phi)$. We will show that there exists a fully reduced primitive Q-resolution refutation π for $\mathtt{Twin}\Phi$ with $|\pi| \leq |\mathfrak{R}(\iota)|$.

Let π be the long-distance Q-resolution refutation of $\mathtt{Twin}\Phi_n$ as described in Proposition 10. Then π is fully reduced. We will show that π is primitive.

Assume not. Then there are two XUT-clauses $B_1, B_2 \in \pi$ that are resolved over some $x \in X$. By the construction of π described in Proposition 10, we can find two cubes $D_1, D_2 \in \mathfrak{R}(\iota)$ such that $\text{var}(D_i) \cap U \neq \emptyset$ and $\text{var}(D_i) \cap T \neq \emptyset$ for $i = 1, 2$ which are resolved over x. One of these cubes was an antecedent cube for x in some trail $\mathcal{T} \in \mathfrak{T}(\iota)$, say $D_1 = \text{ante}_{\mathcal{T}}(x)$ (that means $\bar{x} \in D_1$).

In particular, there is some T-literal $t \in D_1$ such that $t <_{\mathcal{T}} x$ because D_1 must become unit. Remember that t is universal in Rev$(\mathtt{Twin}\Phi)$ and we can only reduce cubes existentially. Then either t was a regular decision, or a propagation.

<u>Case 1:</u> t was decided.

This is only possible if all U-variables were assigned before. Hence, for each $u \in U$ there is a literal ℓ_u with $\text{var}(\ell_u) = u$ and $\ell_u <_{\mathcal{T}} t <_{\mathcal{T}} x$. Because decisions have to be level-ordered in QCDCL, all ℓ_u had to have been propagated.

Let ℓ_u be the leftmost U-literal in $\mathcal{T}$. Consider its antecedent clause $A := \text{ante}_{\mathcal{T}}(\ell_u)$.

Claim: If ℓ_u is the leftmost U-literal in $\mathcal{T}$, then there exists an $i \in \{1, \ldots, m\}$ such that $c_i \in$ $\text{var}(\text{ante}_{\mathcal{T}}(\ell_u))$ *(where $c_1, \ldots, c_m$ are the variables from* Rev$(\mathtt{Twin}\Phi)$ *as in Definition* 12).

Proof of the claim. Assume not. We will show that $A := \text{ante}_{\mathcal{T}}(\ell_u)$ has to contain at least two different U-literals.

Assume that A only contains one U-literal, namely ℓ_u itself. Let Φ consist of the clauses $C_1, \ldots, C_{m'}$ and let $\mathtt{Twin}\Phi$ consist of the clauses $C_1, \ldots, C_m$ with $m > m'$. We can assume that ℓ_u is a copy of a literal from Φ by the construction of a twin formula. In particular, ℓ_u (and $\bar{\ell}_u$) cannot be contained in the clauses $C_1, \ldots, C_{m'}$.

Let ρ be the long-distance Q-resolution derivation of A that was constructed in ι, but not used for $\mathfrak{R}(\iota)$ since certificates can only make use of cubes. By assumption, A does not contain any c_i or $\bar{c}_i$. However, each axiom clause from Rev$(\mathtt{Twin}\Phi)$ includes at least one c_i or $\bar{c}_i$. Hence, we have to resolve over these variables somehow. In particular, we need

$\bar{c}_1 \vee \ldots \vee \bar{c}_m \in \rho$ since this is the only axiom clause where these variables occur in a negative polarity.

We will now construct another long-distance Q-resolution derivation ρ' by substituting $\bar{c}_1 \vee \ldots \vee \bar{c}_m$ with $\bar{c}_1 \vee \ldots \vee \bar{c}_{m'}$ in ρ and gradually deleting all redundant clauses. In particular, all clauses from $\mathrm{Rev}(\mathtt{Twin}\Phi)$ that contain ℓ_u or $\overline{\ell_u}$ will be deleted because the corresponding c_i is missing. Let A' be the last clause in ρ', hence ρ' is a long-distance Q-resolution proof of A' from $\mathrm{Rev}(\Phi)$. Obviously, we get $A' \subseteq A$ and $\ell_u \notin A'$ as well as $c_i, \bar{c}_i \notin A'$ for all $i = 1, \ldots, m$. Since ℓ_u was the only U-literal in A, the clause A' cannot have any U-literals. Therefore A' is a clause consisting of universal literals only. Reducing A' universally gives us the empty clause ($\bot$), which means that we can extend ρ' to a refutation of $\mathrm{Rev}(\Phi)$. But this is a contradiction to the fact that $\mathrm{Rev}(\Phi)$ is a true formula (by Lemma 8).

That shows that A must contain more than one U-literal. Let $\ell_u \neq z \in A$ be another U-literal. Then we need $\bar{z} <_{\mathcal{T}} \ell_u$ since z is existential. However, this contradicts the choice of ℓ_u, which finishes the proof of the claim. ◻

We want to create a contradiction by applying the claim, for which we need to show that A does not contain any literal from $\{c_r, \bar{c}_r \,|\, r = 1, \ldots, m\}$.

Assume that there is such a literal. That means we can find the leftmost literal $c \in \{c_r, \bar{c}_r \,|\, r = 1, \ldots, m\}$ in $\mathcal{T}$, hence $c <_{\mathcal{T}} \ell_u <_{\mathcal{T}} t <_{\mathcal{T}} x$. Now, c cannot have been a decision since decisions must be level-ordered. That means that c has been propagated by an antecedent clause $F := \mathrm{ante}_{\mathcal{T}}(c)$. Because c was leftmost, F cannot be the clause $\bar{c}_1 \vee \ldots \vee \bar{c}_m$. It is easy to see that F then has to contain either w or $\bar{w}$ by the structure of a reversion (see Definition 12). W.l.o.g. let $w \in F$. Then we need $\bar{w} <_{\mathcal{T}} c <_{\mathcal{T}} \ell_u$. Because of the quantification order, $\bar{w}$ cannot be a decided literal. Hence $\bar{w}$ must have been propagated by some antecedent cube $E := \mathrm{ante}_{\mathcal{T}}(\bar{w})$. Let ρ be the subproof of E from $\mathfrak{R}(\iota)$. Then there exists an initial cube $G \in \rho$ with $w \in G$, which is not getting resolved away in ρ. Furthermore, G is also an initial cube in $\mathfrak{R}(\iota)$. By Lemma 9, there exists some $H \in \mathfrak{C}(\mathtt{Twin}\Phi)$ such that $\overline{H} \subseteq G$. Since each clause of Φ contains a U-literal, there is such a U-literal $v \in \overline{H} \subseteq G$ and also $v \in E$ because it cannot be resolved or reduced away. This means we need $v <_{\mathcal{T}} \bar{w} <_{\mathcal{T}} \ell_u$, which is a contradiction to the choice of ℓ_u.

We have now shown that A does not contain any $c_r, \bar{c}_r, r \in \{1, \ldots, m\}$. However, this is impossible by our claim. We conclude that Case 1 cannot occur.

Case 2: t was propagated.

Consider the antecedent cube $J := \mathrm{ante}_{\mathcal{T}}(t)$. Let τ be the subproof of J in $\mathfrak{R}(\iota)$. Then the first cubes in τ were (reduced) satisfying assignments for $\mathrm{Rev}(\mathtt{Twin}\Phi_n)$. At least one of these initial cubes in τ contains $\bar{t}$ which will not get resolved away since it appears in J. Let $I \in \tau$ be an initial cube with $\bar{t} \in I$ that does not get resolved away in τ. By Lemma 9, there exists a clause $K \in \mathfrak{C}(\mathtt{Twin}\Phi_n)$ such that $\overline{K} \subseteq I$. By our assumption, K contains at least one U- and one T-literal. But then also I contains at least one U-literal ℓ. Because ℓ is blocked by $\bar{t}$ all the time, it does not get reduced away in τ, hence $\ell \in J$.

Due to $\ell <_{\mathrm{Rev}(\mathtt{Twin}\Phi_n)} t$, we need $\ell <_{\mathcal{T}} t$ in order for J to become unit. W.l.o.g. let ℓ be the leftmost U-literal in $\mathcal{T}$ (the fact that $\ell \in J$ is not important anymore from this point on). Because of $x <_{\mathrm{Rev}(\mathtt{Twin}\Phi_n)} \ell$, the literal ℓ cannot be a regular decision. That means it must have been propagated.

We can repeat the argument from Case 1. We conclude that such an ℓ does not exist. Thus Case 2 does not occur and we get a contradiction regarding our assumption that π was not primitive. ◻

We now construct specific QBFs that meet the conditions of Theorem 12. We already know from [9] that the equality formulas $\mathtt{Eq}_n$ of [6] have linear gauge and therefore need exponential-size fully reduced primitive Q-resolution refutations. However, not all clauses from $\mathtt{Eq}_n$ contain a U-literal. We modify the formulas by adding an artificial U-literal p to the relevant clauses:

Definition 13 The QCNF $\mathtt{ModEq}_n$ consists of the prefix $\exists x_1, \ldots, x_n \forall u_1, \ldots, u_n, p \exists t_1, \ldots, t_n$
and the matrix $x_i \vee u_i \vee t_i,\ \bar{x}_i \vee \bar{u}_i \vee t_i,\ p \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n,\ \bar{p} \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n$ for $i = 1, \ldots, n$.

Neither this nor the $\mathtt{Twin}$ modification changes the gauge of the formulas. Hence we get:

Proposition 13 *It holds gauge*$(\mathtt{TwinModEq_n}) = n$. *Hence,* $\mathtt{TwinModEq_n}$ *needs exponential-size fully reduced primitive* Q-resolution *refutations.*

Proof Since all axiom clauses contain T-literals, we have to get rid of them somehow. The only four clauses that contain T-literals in a negative polarity are the clauses $p \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n$, $\bar{p} \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n$, $q \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n$ and $\bar{q} \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n$, where q is the copy of p. Hence, we have to use at least one of them in order to derive an X-clause. In particular, we have to resolve over each t_i. The only four clauses in which t_i occurs in a positive polarity are $x_i \vee u_i \vee t_i$, $\bar{x}_i \vee \bar{u}_i \vee t_i$, $x_i \vee v_i \vee t_i$ and $\bar{x}_i \vee \bar{v}_i \vee t_i$, where v_i is the copy of u_i. In each case we will pile up x_i or $\bar{x}_i$ for each resolution over t_i. Therefore, our X-clause at the end will contain at least n different X-literals.

Hence gauge$(\mathtt{TwinModEq_n}) = n$. The second claim then follows from Theorem 7. □

The lower bound for the true QBFs then follows with Theorem 12.

Corollary 14 *Rev*$(\mathtt{TwinModEq}_n)$ *needs exponential-size* QCDCL *certificates.*

We now use a direct construction to show that Rev$(\mathtt{TwinModEq}_n)$ is easy for QCDCL$^{\text{EXI-ANY}}$.

Proposition 15 *Rev*$(\mathtt{TwinModEq}_n)$ *has polynomial-size* QCDCL$^{\text{EXI-ANY}}$ *certificates.*

Proof Let us first list all the clauses of $\mathtt{TwinModEq}_n$. It consists of the prefix

$$\exists x_1, \ldots, x_n \forall u_1, \ldots, u_n, p, v_1, \ldots, v_n, q \exists t_1, \ldots, t_n$$

and the matrix

$$\begin{aligned} C_{(i,1)} &:= x_i \vee u_i \vee t_i & C_1 &:= p \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n \\ C_{(i,2)} &:= \bar{x}_i \vee \bar{u}_i \vee t_i & C_2 &:= \bar{p} \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n \\ C_{(i,3)} &:= x_i \vee v_i \vee t_i & C_3 &:= q \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n \\ C_{(i,4)} &:= \bar{x}_i \vee \bar{v}_i \vee t_i & C_4 &:= \bar{q} \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n \end{aligned}$$

for $i = 1, \ldots, n$.

Then the true QCNF Rev$(\mathtt{TwinModEq}_n)$ consists of the prefix

$$\forall x_1, \ldots, x_n \exists u_1, \ldots, u_n, p, v_1, \ldots, v_n, q \forall t_1, \ldots, t_n, w \exists M,$$

with $M := \{c_{(i,j)}, c_j \mid i = 1, \ldots, n,\ j = 1, \ldots, 4\}$, and the matrix

$$E := \bigvee_{i=1}^{n} \bigvee_{j=1}^{4} \bar{c}_{(i,j)} \vee \bigvee_{k=1}^{4} \bar{c}_k$$

$$\begin{array}{lll}
\bar{x}_i \vee w \vee c_{(i,1/3)} & \bar{u}_i \vee w \vee c_{(i,1)} & \bar{v}_i \vee w \vee c_{(i,3)} \\
\bar{x}_i \vee \bar{w} \vee c_{(i,1/3)} & \bar{u}_i \vee \bar{w} \vee c_{(i,1)} & \bar{v}_i \vee \bar{w} \vee c_{(i,3)} \\
x_i \vee w \vee c_{(i,2/4)} & u_i \vee w \vee c_{(i,2)} & v_i \vee w \vee c_{(i,4)} \\
x_i \vee \bar{w} \vee c_{(i,2/4)} & u_i \vee \bar{w} \vee c_{(i,2)} & v_i \vee \bar{w} \vee c_{(i,4)} \\
\bar{t}_i \vee w \vee c_{(i,1/2/3/4)} & & \\
\bar{t}_i \vee \bar{w} \vee c_{(i,1/2/3/4)} & &
\end{array}$$

$$\begin{array}{llll}
\bar{p} \vee w \vee c_1 & p \vee w \vee c_2 & \bar{q} \vee w \vee c_3 & q \vee w \vee c_4 \\
\bar{p} \vee \bar{w} \vee c_1 & p \vee \bar{w} \vee c_2 & \bar{q} \vee \bar{w} \vee c_3 & q \vee \bar{w} \vee c_4 \\
t_i \vee w \vee c_{1/2/3/4} & & & \\
t_i \vee \bar{w} \vee c_{1/2/3/4} & & &
\end{array}$$

for $i = 1, \dots, n$, where variables like $c_{(i,1/3)}$ decode two versions of this clause: One clause with $c_{(i,1)}$ and the other with $c_{(i,3)}$ (analogously with $c_{(i,2/4)}$, $c_{(i,1/2/3/4)}$ and $c_{1/2/3/4}$).

Let us now construct a polynomial size $\mathsf{QCDCL}^{\mathsf{EXI\text{-}ANY}}$ certificate. At first, we would like to learn the cubes

$$\begin{aligned}
D_{(i,1)} &:= \bar{x}_i \wedge \bar{u}_i \wedge \bar{t}_i \\
D_{(i,2)} &:= x_i \wedge u_i \wedge \bar{t}_i \\
D_1 &:= \bar{p} \wedge t_1 \wedge \ldots \wedge t_n
\end{aligned}$$

for $i = 1, \dots, n$. In order to learn $D_{(i,1)}$, we will make (level-ordered) decisions that satisfy all literals from $D_{(i,1)}$, but falsify all the other $D_{(i',1)}$ for $i' \neq i$. For example, we set x_i, u_i and t_i to false, and we can assign all the other variables left of w arbitrarily. Note that until we reach w, we will never make any propagations since w or $\bar{w}$ is blocking them. After having decided all variables left of w, we will decide w and potentially trigger some propagations. However, the variable $c_{(i,1)}$ will never be propagated because all clauses containing it are already satisfied. After this we will set $c_{(i,1)}$ to false and all the remaining variables to true.

We now have satisfied the clause E. Furthermore, we have set all $c_{(i',j)}$ and c_k to true except $c_{(i,1)}$. Hence we have satisfied all clauses except the four clauses containing $c_{(i,1)}$. But these two clauses were already satisfied because we have satisfied the cube $D_{(i,1)}$ with the decisions left of w.

Let $\mathcal{T}_{(i,1)}$ be the trail we have constructed now. We can extract the cube

$$\bar{x}_i \wedge \bar{u}_i \wedge \bar{t}_i \wedge \bar{c}_{(i,1)} \wedge \bigwedge_{(i',j) \in (\{1,\dots,n\} \times \{1,2,3,4\}) \setminus \{(i,1)\}} c_{(i',j)} \wedge \bigwedge_{k=1}^{4} c_k,$$

which, as an assignment, already satisfies all clauses from $\mathrm{Rev}(\mathtt{TwinModEq}_n)$. This cube can be existentially reduced to $D_{(i,1)}$, which is the cube we learn from $\mathcal{T}_{(i,1)}$. Analogously, we can learn the cubes $D_{(i,2)}$ for $i = 1, \dots, n$ via some analogue trails $\mathcal{T}_{(i,2)}$.

It remains to learn the cube D_1, which represents the clause $C_1 \in \mathfrak{C}(\mathtt{TwinModEq}_n)$. We will construct a trail $\mathcal{T}_1$ which includes (level-ordered) decisions that satisfy D_1. But now we have to make sure not to trigger propagations via $D_{(i,1)}$ or $D_{(i,2)}$ since we must not set t_i to false. This can be done by setting all x_i to false and all u_i to true. Then we can set p to false and all t_i to true. The remaining variables left of w can again be decided arbitrarily. Then we set w to true and potentially trigger some propagations of $c_{(i,j)}$ or c_k, which is not a problem since c_1 will never be propagated (the clauses containing c_1 are already satisfied). Then we set c_1 to false and all remaining variables can be set to true.

As with $\mathcal{T}(i, 1)$, we have satisfied all clauses from $\text{Rev}(\texttt{TwinModEq}_n)$. We can extract the cube

$$\bar{p} \wedge t_1 \wedge \ldots \wedge t_n \wedge \bar{c}_1 \wedge \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{4} c_{(i,j)} \wedge \bigwedge_{k=2}^{4} c_k,$$

from $\mathcal{T}_1$, which already satisfies the matrix and can be existentially reduced to D_1.

We will now define the cubes

$$R_i := \bar{x}_i \wedge \bar{u}_i \wedge \bar{p} \wedge \bigwedge_{k=i+1}^{n} (u_k \wedge \bar{u}_k) \wedge \bigwedge_{\ell=1}^{i-1} t_\ell$$

$$L_i := x_i \wedge u_i \wedge \bar{p} \wedge \bigwedge_{k=i+1}^{n} (u_k \wedge \bar{u}_k) \wedge \bigwedge_{\ell=1}^{i-1} t_\ell$$

for $i = 2, \ldots, n-1$. We will construct trails $\mathcal{U}_{n-1}, \mathcal{V}_{n-1}, \ldots, \mathcal{U}_2, \mathcal{V}_2$ with which we will gradually learn the clauses $R_{n-1}, L_{n-1}, \ldots, R_2, L_2$.

We start with

$$\mathcal{U}_{n-1} := (\bar{\boldsymbol{p}}; \bar{\boldsymbol{x}}_{\boldsymbol{1}}; \bar{\boldsymbol{u}}_{\boldsymbol{1}}, t_1; \ldots; \bar{\boldsymbol{x}}_{\boldsymbol{n-1}}; \bar{\boldsymbol{u}}_{\boldsymbol{n-1}}, t_{n-1}, \bar{t}_n, \bar{x}_n, \top)$$

with antecedent cubes

$$\begin{aligned}
\text{ante}_{\mathcal{U}_{n-1}}(t_j) &= D_{(j,1)} \\
\text{ante}_{\mathcal{U}_{n-1}}(\bar{t}_n) &= D_1 \\
\text{ante}_{\mathcal{U}_{n-1}}(\bar{x}_n) &= D_{(n,2)} \\
\text{ante}_{\mathcal{U}_{n-1}}(\top) &= D_{(n,1)}
\end{aligned}$$

for $j = 1, \ldots, n-1$. We learn the cube $R_{n-1} = \left(\left(D_{(n,1)} \overset{x_n}{\bowtie} D_{(n,2)}\right) \overset{t_n}{\bowtie} D_1\right) \overset{t_{n-1}}{\bowtie} D_{(n-1,1)}$.

Analogously, by flipping some polarities, we construct the trail $\mathcal{V}_{n-1}$ and learn the cube $L_{n-1} = \left(\left(D_{(n,1)} \overset{x_n}{\bowtie} D_{(n,2)}\right) \overset{t_n}{\bowtie} D_1\right) \overset{t_{n-1}}{\bowtie} D_{(n-1,2)}$. Note that R_{n-1} will not interfere with the assignments in $\mathcal{V}_{n-1}$.

Assume we have already learned the clauses $R_{n-1}, L_{n-1}, \ldots, R_i, L_i$ for some $i \in \{3, \ldots, n-1\}$. Then we can construct the following trail:

$$\mathcal{U}_{i-1} := (\bar{\boldsymbol{p}}; \bar{\boldsymbol{x}}_{\boldsymbol{1}}; \bar{\boldsymbol{u}}_{\boldsymbol{1}}, t_1; \ldots; \bar{\boldsymbol{x}}_{\boldsymbol{i-1}}; \bar{\boldsymbol{u}}_{\boldsymbol{i-1}}, t_{i-1}, x_i, \top)$$

with antecedent cubes

$$\begin{aligned}
\text{ante}_{\mathcal{U}_{i-1}}(t_j) &= D_{(j,1)} \\
\text{ante}_{\mathcal{U}_{i-1}}(x_i) &= R_i \\
\text{ante}_{\mathcal{U}_{i-1}}(\top) &= L_i
\end{aligned}$$

for $j = 1, \ldots, i-1$. We learn the cube $R_{i-1} = \left(L_i \overset{x_i}{\bowtie} R_i\right) \overset{t_{i-1}}{\bowtie} D_{(i-1,1)}$. Analogously, we can construct the trail $\mathcal{V}_{i-1}$ and learn $L_{i-1} = \left(L_i \overset{x_i}{\bowtie} R_i\right) \overset{t_{i-1}}{\bowtie} D_{(i-1,2)}$.

After having learned the cubes $R_{n-1}, L_{n-1}, \ldots, R_2, L_2$, we construct two more trails, namely

$$\mathcal{U}_1 := (\bar{\boldsymbol{p}}; \bar{\boldsymbol{x}}_{\boldsymbol{1}}; \bar{\boldsymbol{u}}_{\boldsymbol{1}}, t_1, x_2, \top)$$

with antecedent cubes

$$\begin{aligned}\mathrm{ante}_{\mathcal{U}_1}(t_1) &= D_{(1,1)}\\ \mathrm{ante}_{\mathcal{U}_1}(x_2) &= R_2\\ \mathrm{ante}_{\mathcal{U}_1}(\top) &= L_2,\end{aligned}$$

from which we learn $[\bar{x}_1] = \left(L_2 \overset{x_2}{\bowtie} R_2\right) \overset{t_1}{\bowtie} D_{1,1}$, and the trail

$$\mathcal{V}_1 := (x_1;\, \bar{\boldsymbol{p}};\, \mathbf{u_1}, t_1, x_2, \top)$$

with antecedent cubes

$$\begin{aligned}\mathrm{ante}_{\mathcal{V}_1}(x_1) &= [x_1]\\ \mathrm{ante}_{\mathcal{V}_1}(t_1) &= D_{(1,2)}\\ \mathrm{ante}_{\mathcal{V}_1}(x_2) &= R_2\\ \mathrm{ante}_{\mathcal{V}_1}(\top) &= L_2,\end{aligned}$$

from which we learn the empty cube $[\top] = \mathrm{red}^{\exists}_{\mathrm{Rev}(\mathtt{TwinModEq_n})}\left(\left(L_2 \overset{x_2}{\bowtie} R_2\right) \overset{t_1}{\bowtie} D_{(1,2)}\right) \overset{x_1}{\bowtie} [x_1]$.

All in all, we have constructed a QCDCL$^{\text{Exi-Any}}$ certificate using the $4n-1$ trails

$$\mathcal{T}_{(1,1)}, \ldots, \mathcal{T}_{(n,1)}, \mathcal{T}_{(1,2)} \ldots, \mathcal{T}_{(n,2)}, \mathcal{T}_1, \mathcal{U}_{n-1}, \mathcal{V}_{n-1}, \ldots, \mathcal{U}_1, \mathcal{V}_1.$$

□

Corollary 16 QCDCL *and* QCDCL$^{\text{Exi-Any}}$ *are exponentially separated on true formulas.*

5.2 Separation on False Formulas

For separating QCDCL and QCDCL$^{\text{Uni-Any}}$, we recall the completion principle $\mathtt{CR}_n$ of [18].

Definition 14 ([18]) The false QCNF $\mathtt{CR}_n$ consists of the prefix $\exists X \forall U \exists T$ with

$$X := \{x_{(i,j)} \mid i, j \in \{1, \ldots, n\}\}, \quad U := \{u\}, \quad T := \{a_i, b_i \mid i \in \{1, \ldots, n\}\}$$

and the matrix

$$x_{(i,j)} \vee u \vee a_i \quad \bar{x}_{(i,j)} \vee \bar{u} \vee b_j \quad \bar{a}_1 \vee \ldots \vee \bar{a}_n \quad \bar{b}_1 \vee \ldots \vee \bar{b}_n$$

for $i, j = 1, \ldots, n$.

For the lower bound, we will use the modification $\mathtt{TwinCR}_n$. As we show, cube learning becomes rather useless with the $\mathtt{Twin}$ modification. This fact helps us to ensure that QCDCL refutations of $\mathtt{TwinCR}_n$ are primitive, and thus we can apply the gauge lower-bound method.

Similarly as in Proposition 13 we can compute the gauge.

Lemma 17 *It holds gauge*$(\mathtt{TwinCR}_n) = n$.

Proof For the derivation of an X-clause we need at least one of the clauses $\bar{a}_1 \vee \ldots \vee \bar{a}_n$ or $\bar{b}_1 \vee \ldots \vee \bar{b}_n$ since we have to get rid of all T-literals. In particular, w.l.o.g. we have to resolve over each a_i. For this, we need one of the clauses $x_{(i,j)} \vee u \vee a_i$ or $x_{(i,j)} \vee v \vee a_i$ for each i. That means for each i we will pile up at least one $x_{(i,j)}$ for some j. Therefore gauge$(\mathtt{TwinCR}_n) = n$. □

The main work is to check that QCDCL refutations of $\texttt{TwinCR}_n$ are primitive.

Proposition 18 *If ι is a* QCDCL *refutation of* $\texttt{TwinCR}_n$*, then* $\mathfrak{R}(\iota)$ *is fully reduced and primitive.*

Proof It suffices to show that $\mathfrak{R}(\iota)$ is primitive. Assume not.

Then there exists two XUT-clauses $C, D \in \mathfrak{R}(\iota)$ that are resolved over an X-literal, say x. One of these two clauses has to be the antecedent clause of x by the definition of clause learning, say $C = \text{ante}_{\mathcal{T}}(x)$ for some trail $\mathcal{T} \in \mathfrak{T}(\iota)$. Let $t_1 \in C$ be one of the T-literals. We want to show, that there exists a U-literal w with $w <_{\mathcal{T}} x$.

Assume that no such w exists. Since C had to become unit at the propagation of x, we need $\bar{t}_1 <_{\mathcal{T}} x$. The literal $\bar{t}_1$ cannot be a decision in $\mathcal{T}$, since this would mean that we assigned all U-variables earlier in the trail, which contradicts our assumption. Hence $\bar{t}_1$ must have been a propagation.

Starting with $i = 1$, we define $F_i := \text{ante}_{\mathcal{T}}(\bar{t}_i)$. Now, F_i cannot contain U-literals since we cannot falsify these literals before assigning $\bar{t}_i$. Because of the XT-property (and Lemma 6), F_i cannot contain X-literals, as well (otherwise it would be an XT-clause). But if the XT-property is fulfilled, we cannot derive unit T-clauses, therefore F_i has to contain at least one additional T-literal, say $t_{i+1} \in F_i$.

This argument can be repeated for each $i \in \mathbb{N}$, which means we could find an infinite amount of T-literals $\bar{t}_i$ that must be all contained in $\mathcal{T}$, which is obviously not possible. This shows that our assumption was false and we can indeed find such a U-literal $w <_{\mathcal{T}} \bar{t}_1 <_{\mathcal{T}} x$.

W.l.o.g. let w be the first (leftmost) U-literal in $\mathcal{T}$. Define $A := \text{ante}_{\mathcal{T}}(w)$. Clearly, A is a cube. We will show that A contains at least two different U-literals. Then, since w was the first U-literal in $\mathcal{T}$, A cannot become unit until at least one U-literal was assigned, which would be a contradiction.

Now, A is a cube that was derived during cube learning from cubes that represent satisfying (partial) assignments of the matrix of $\texttt{TwinCR}_\text{n}$. Let D be a cube that satisfies the matrix of $\texttt{TwinCR}_\text{n}$. Because we have to satisfy the clauses $\bar{a}_1 \vee \ldots \vee \bar{a}_n$ and $\bar{b}_1 \vee \ldots \vee \bar{b}_n$, there exists an $r \in \{1, \ldots, n\}$ with $\bar{a}_r \in D$ and an $s \in \{1, \ldots, n\}$ with $\bar{b}_s \in D$. Furthermore, we have to satisfy the clauses $x_{(r,s)} \vee u \vee a_r$, $x_{(r,s)} \vee v \vee a_r$, $\bar{x}_{(r,s)} \vee \bar{u} \vee b_s$ and $\bar{x}_{(r,s)} \vee \bar{v} \vee b_s$. That means we have to assign u in some polarity. W.l.o.g. let $u \in D$. Then we have to set $x_{(r,s)}$ to false, hence $\bar{x}_{(r,s)} \in D$. In order to satisfy $x_{(r,s)} \vee v \vee a_r$, we have to set v to true, as well. Therefore we get $v \in D$.

We conclude, that $u \in D$ if and only if $v \in D$, and analogously $\bar{u} \in D$ if and only if $\bar{v} \in D$. This means that we will never be able to resolve such two learned cubes in ι since we cannot create universal tautologies in cubes. In particular, we have proven that A contains at least two U-literals, which leads to a contradiction as described above. □

Applying Theorem 7 then yields the lower bound.

Corollary 19 $\texttt{TwinCR}_n$ *needs exponential-sized* QCDCL *refutations.*

On the other hand, $\texttt{TwinCR}_n$ is easy for $\text{QCDCL}^{\text{UNI-ANY}}$. Basically, we can simulate the Q-resolution refutation of $\texttt{CR}_n$ from [17], because we can decide universal literals out of order.

Proposition 20 $\texttt{TwinCR}_n$ *has polynomial-sized* $\text{QCDCL}^{\textit{UNI-ANY}}$ *refutations.*

Proof For each $k = 1, \ldots, n$ we construct the trail

$$\mathcal{T}_k := (\boldsymbol{\bar{x}_{(1,k)}}; \ldots; \boldsymbol{\bar{x}_{(n,k)}}; \boldsymbol{\bar{u}}, a_1, \ldots, a_n, \bot)$$

with antecedent clauses

$$\text{ante}_{\mathcal{T}_k}(a_i) = x_{(i,k)} \vee u \vee a_i, \quad \text{ante}_{\mathcal{T}_k}(\bot) = \bar{a}_1 \vee \ldots \vee \bar{a}_n,$$

for $i = 1, \ldots, n$.

Resolving $\bar{a}_1 \vee \ldots \vee \bar{a}_n$ over each $\text{ante}_{\mathcal{T}_k}(a_i)$ gives us the clause $E_k := x_{(1,k)} \vee \ldots \vee x_{(n,k)}$, which we will learn. Note that the trails and the learned clauses will not affect each other, hence the order in which we construct these n trails does not matter. Next, we construct the trails $\mathcal{U}_1, \ldots, \mathcal{U}_{n-1}$ (in that order). From each $\mathcal{U}_k$ we learn the clause $C_k := \bar{u} \vee b_k$. While constructing $\mathcal{U}_k$, we assume that $C_1, \ldots, C_{k-1}$ were already learned. Then, $\mathcal{U}_k$ looks as follows:

$$\mathcal{U}_k := (\mathbf{u}, b_1, \ldots, b_{k-1}; \mathbf{v}; \bar{\boldsymbol{b}}_{\boldsymbol{k}}, \bar{x}_{(1,k)}, \ldots, \bar{x}_{(n,k)}, \bot)$$

with antecedent clauses

$$\text{ante}_{\mathcal{U}_k}(b_j) = C_j, \quad \text{ante}_{\mathcal{U}_k}(\bar{x}_{(i,k)}) = \bar{x}_{(i,k)} \vee \bar{u} \vee b_k, \quad \text{ante}_{\mathcal{U}_k}(\bot) = E_k,$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, k-1$. Resolving E_k over each $\text{ante}_{\mathcal{U}_k}(\bar{x}_{(i,k)})$ leads to the learnable clause C_k. Having learned the clauses $C_1, \ldots, C_{n-1}$, we continue with the trail $\mathcal{V}$, which will be the last one. It looks as follows:

$$\mathcal{V} := (\mathbf{u}, b_1, \ldots, b_{n-1}, \bar{b}_n, \bar{x}_{(1,n)}, \ldots, \bar{x}_{(n,n)}, \bot)$$

with antecedent clauses

$$\text{ante}_{\mathcal{V}}(b_j) = C_j, \quad \text{ante}_{\mathcal{V}}(\bar{b}_n) = \bar{b}_1 \vee \ldots \vee \bar{b}_n, \quad \text{ante}_{\mathcal{V}}(\bar{x}_{(i,n)}) = \bar{x}_{(i,n)} \vee \bar{u} \vee b_n,$$
$$\text{ante}_{\mathcal{V}}(\bot) = E_n,$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, n-1$. Since we only made a universal decision, we can learn the empty clause $(\bot)$ from $\mathcal{V}$ by resolving over everything.

Thus we constructed a QCDCL$^{\text{UNI-ANY}}$ refutation using $2n+1$ trails. □

Besides $\texttt{TwinCR}_n$, we can find further separations between QCDCL and QCDCL$^{\text{UNI-ANY}}$. The QCNFs $\texttt{MirrorCR}_n$ were introduced in [4] as a modification of $\texttt{CR}_n$, where it was shown that the formula is hard for several variants of QCDCL, including our base model QCDCL. It is notable that the matrix of $\texttt{MirrorCR}_n$ is unsatisfiable, and therefore we will never perform cube learning.

Definition 15 The false QCNF $\texttt{MirrorCR}_n$ consists of the prefix

$$\exists x_{(1,1)}, \ldots, x_{(n,n)} \forall u \exists a_1, \ldots, a_n, b_1, \ldots, b_n$$

and the matrix

$$\begin{array}{lll} x_{(i,j)} \vee u \vee a_i & \bar{a}_1 \vee \ldots \vee \bar{a}_n & \\ \bar{x}_{(i,j)} \vee \bar{u} \vee b_j & \bar{b}_1 \vee \ldots \vee \bar{b}_n & \\ x_{(i,j)} \vee \bar{u} \vee \bar{a}_i & a_1 \vee \ldots \vee a_n & \\ \bar{x}_{(i,j)} \vee u \vee \bar{b}_j & b_1 \vee \ldots \vee b_n & \text{for } i, j \in \{1, \ldots, n\}. \end{array}$$

Proposition 21 *([4])* $\texttt{MirrorCR}_n$ *needs exponential-sized* QCDCL *refutations.*

Proposition 22 $\texttt{MirrorCR}_n$ *has polynomial-sized* QCDCL$^{\text{UNI-ANY}}$ *refutations.*

Proof At first, we will derive the clauses $A_k := x_{(1,k)} \vee \ldots \vee x_{(n,k)}$ for each $k = 1, \ldots, n$. Suppose, we have already learned $A_1, \ldots, A_{k-1}$. We construct the trail $\mathcal{T}_k$ as follows:

$$\mathcal{T}_k := (\bar{\boldsymbol{x}}_{\boldsymbol{(1,k)}}; \ldots; \bar{\boldsymbol{x}}_{\boldsymbol{(n,k)}}; \bar{\boldsymbol{u}}, a_1, \ldots, a_n, \bot)$$

with

$$\text{ante}_{\mathcal{T}_k}(a_i) = x_{(i,k)} \vee u \vee a_i$$
$$\text{ante}_{\mathcal{T}_k}(\bot) = \bar{a}_1 \vee \ldots \vee \bar{a}_n$$

for $i = 1, \ldots, n$. From this trail we can learn E_k by resolving over all a_i and then we restart.

Our next goal is to learn the clauses $B_k := \bar{u} \vee b_k$ for each $k = 1, \ldots, n-1$. We now suppose that we have already learned $A_1, \ldots, A_n$ and $B_1, \ldots, B_{k-1}$. We construct the trail $\mathcal{U}_k$ as follows:

$$\mathcal{U}_k := (\mathbf{u}, b_1, \ldots, b_{k-1}; \bar{\boldsymbol{b}}_{\boldsymbol{k}}, \bar{x}_{(1,k)}, \ldots, \bar{x}_{(n,k)}, \bot)$$

with

$$\text{ante}_{\mathcal{U}_k}(b_j) = B_j$$
$$\text{ante}_{\mathcal{U}_k}(\bar{x}_{(i,k)}) = \bar{x}_{(i,k)} \vee \bar{u} \vee b_k$$
$$\text{ante}_{\mathcal{U}_k}(\bot) = A_k$$

for $j = 1, \ldots, k-1$ and $i = 1, \ldots, n$. We learn B_k by resolving A_k over all $x_{(i,k)}$. After this we backtrack back to the point where we decided $\bar{b}_k$.

Our last trail, from which we plan to learn the empty clause, looks as follows:

$$\mathcal{U}_n := (\mathbf{u}, b_1, \ldots, b_n, \bot)$$

with

$$\text{ante}_{\mathcal{U}_n}(b_j) = B_j$$
$$\text{ante}_{\mathcal{U}_n}(\bot) = \bar{b}_1 \vee \ldots \vee \bar{b}_n.$$

We resolve over all b_j and obtain $(\bot)$. □

Corollary 23 $\texttt{MirrorCR}_n$ *is hard for* QCDCL, *but easy for* QCDCL$^{\textsc{Uni-Any}}$.

Corollary 24 QCDCL *and* QCDCL$^{\textsc{Uni-Any}}$ *are exponentially separated on false formulas.*

We combine both separations into our main result:

Theorem 25 *a)* QCDCL$^{\textsc{Uni-Any}}$ *is exponentially stronger than* QCDCL *on false formulas.*
b) QCDCL$^{\textsc{Exi-Any}}$ *is exponentially stronger than* QCDCL *on true formulas.*
c) QCDCL$^{\textsc{Any}}$ *is exponentially stronger than* QCDCL *both on false and true formulas.*

6 Experiments

One of the aspirations of proof complexity is to explain and predict solver behaviour, in particular running time. In this section, we evaluate how well our proof-complexity results transfer to the ‘real world’ of QCDCL implemented in a solver.

For our experiments we picked the QCDCL solver Qute[7] [29, 33], and implemented each of the aforementioned QCDCL variants: $\text{QCDCL}^{\text{UNI-ANY}}$, $\text{QCDCL}^{\text{EXI-ANY}}$, and $\text{QCDCL}^{\text{ANY}}$ (Qute could already run in a mode that corresponds to QCDCL). In order to ensure compliance with the NCC (Definition 4), we needed to adapt some of Qute's internal data structures, and so for the sake of a fair comparison we also report on a version called QCDCL3: algorithmically plain QCDCL but with the new data structures that are required for the other variants (up to 3 watched literals rather than the usual 2, hence the name).[8]

We performed two experiments. In the first, we evaluated each QCDCL variant on the first 100 formulas from each separation family—`TwinCR`, `MirrorCR`, and Rev(`TwinModEq`$_n$)—running the solver with a time limit of 600 s on each individual formula on a machine with two 16-core Intel® Xeon® E5-2683 v4@2.10GHz CPUs and 512GB RAM running Ubuntu 20.04.3 LTS on Linux 5.4.0-48, organizing the computation with the help of GNU Parallel [35].

In the second, we additionally evaluated each QCDCL variant on the formulas from the latest two QBF Evaluations, 2020[9] and 2022[10] (there was no evaluation of QBF solvers in 2021), in both PCNF and QCIR categories, with the same time limit of 10 min. This was executed on a different cluster with heterogeneous machines powered by different Intel® Xeon® CPUs and AMD® EPYC® 7402@2.80GHz.

For all of our experiments we executed Qute with the same, default parameters for all heuristics. However, in order to obtain any meaningful results on the separation formulas, we had to tweak the initialization process of Qute's decision heuristic, which determines the next branching variable. Previously, the heuristic was initialized in prefix order, giving higher preference to variables earlier in the prefix. As a result, on our separation formulas, the solver kept branching on and learning clauses involving only outermost variables, and never made any decisions out of the prefix order even when allowed to, because the heurstic never suggested to. We changed this default initialization to go in reverse prefix order, and adopted this change for both experiments and all runs. We emphasize that this manual change affects only the initialization values (afterwards the heuristic updates according to the same rules as before), and also that the previous setting of in-order default initialization was an arbitrary choice.

6.1 Separation Formulas

In Figs. 3 and 4 we plot running times of the different QCDCL versions as a function of n. Any gaps in the plotted lines indicate the solver timed out at 600 seconds for that particular formula. In general, the proof complexity results are closely mirrored in solver performance, though there is occasionally a bit of surprise.

In Fig. 3, we see that for `TwinCR` the configuration $\text{QCDCL}^{\text{UNI-ANY}}$ is best and scales reasonably well up to $n = 100$. But there are also gaps—for some reason the solver's heuristics appear to be fooled for some particular formulas and fail to navigate towards the short proof. Overall, $\text{QCDCL}^{\text{UNI-ANY}}$ manages to solve 87 out the first 100 `TwinCR` formulas. $\text{QCDCL}^{\text{ANY}}$, which should theoretically be at least as good as $\text{QCDCL}^{\text{UNI-ANY}}$, comes a distant second and

7 https://github.com/fslivovsky/qute.

8 We invoked Qute with the parameters `–dependency-learning off –machine-readable -t 600` and, either `–watched-literals 3` for QCDCL3 or `–out-of-order-decisions off|existential|universal|all` for the other variants.

9 http://www.qbflib.org/event_page.php?year=2020.

10 http://www.qbflib.org/qbfeval2022_results.php.

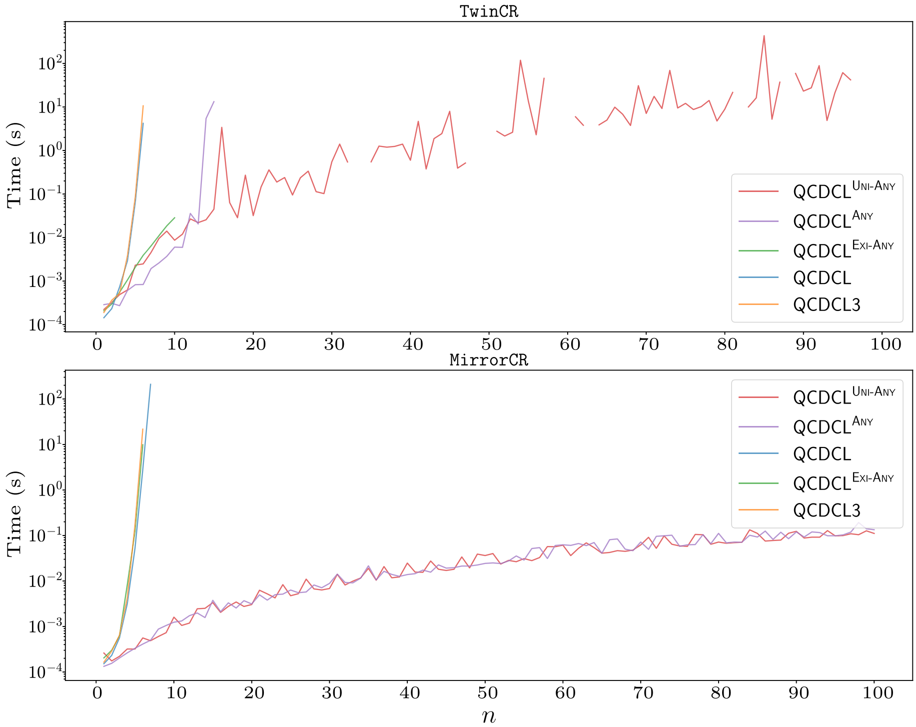

Fig. 3 Performance on $\texttt{TwinCR}_n$ (above) and $\texttt{MirrorCR}_n$ (below) Legends are sorted best-to-worst

fails to solve anything beyond $n = 16$. QCDCL$^{\text{Exi-Any}}$ appears to be off to a good start, but also quickly loses breath solving nothing after $n = 10$. The two vanilla variants QCDCL and QCDCL3 scale exponentially all the way as they should.

The picture on the related `MirrorCR` formulas (Fig. 3 below) is boring in comparison and perfectly corresponds to our theoretical results. The two variants that have short proofs—QCDCL$^{\text{Uni-Any}}$ and QCDCL$^{\text{Any}}$—are also fast in practice, and everything else is dead exponential.

Finally, Rev(`TwinModEq`) in Fig. 4 paint a picture somewhat similar to `TwinCR`, though with a different set of peculiarities. The best variant is QCDCL$^{\text{Exi-Any}}$, and unlike QCDCL$^{\text{Uni-Any}}$ on `TwinCR`, it solves all formulas up to $n = 100$ very fast. The second best is QCDCL$^{\text{Any}}$, but once again it drops out relatively early (last solved is $n = 26$) in spite of its theoretical superiority. An interesting thing seems to happen to QCDCL$^{\text{Uni-Any}}$, which appears to be helplessly off to an exponential path, but somehow recovers and solves $n = 15, 16$ fast, only to completely drop out afterwards. The two vanilla variants QCDCL and QCDCL3 are again dead exponential, as they should be.

The recurring theme in Figs. 3 and 4 is that the theoretically strongest system QCDCL$^{\text{Any}}$ is outperformed by the specialized version for each formula type. One appealing explanation would be that the specialized systems QCDCL$^{\text{Uni-Any}}$ and QCDCL$^{\text{Exi-Any}}$ profit from their ability to guarantee learning asserting clauses and cubes respectively. But this does not appear to be the real reason: QCDCL$^{\text{Any}}$ also (like QCDCL$^{\text{Uni-Any}}$) learns almost exclusively asserting clauses on `TwinCR` (96% on average, more than 99% in over 70% of cases), and similarly QCDCL$^{\text{Any}}$ (like QCDCL$^{\text{Exi-Any}}$) learns almost exclusively asserting cubes on Rev(`TwinModEq`) (98%

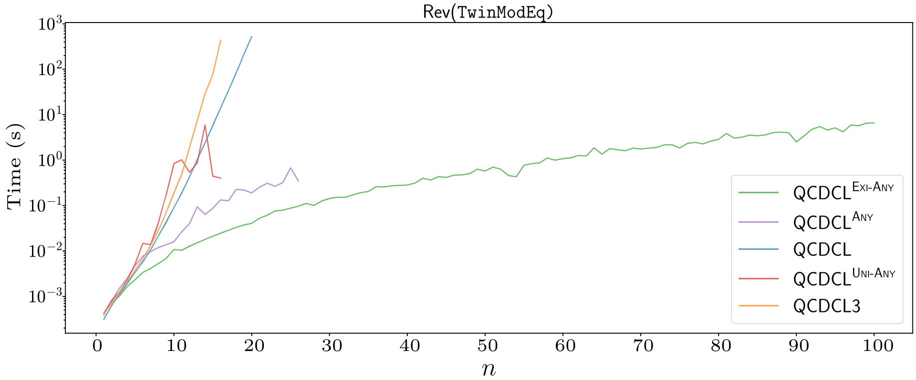

Fig. 4 Running time in seconds on Rev($\texttt{TwinModEq}_n$). The legend is sorted from best downwards

Table 1 Results of the QCDCL variants on QBF Eval 2020 and 2022. VBS stands for the *virtual best solver*, the best performing solver on each instance. $\top$ gives the number of true solved formulas, $\perp$ the number of false solved formulas, $\Sigma = \top + \perp$. Column maxima are in bold (excluding VBS)

	QBF Eval 20						QBF Eval 22					
	PCNF			QCIR			PCNF			QCIR		
	$\top$	$\perp$	Σ	$\top$	$\perp$	Σ	$\top$	$\perp$	Σ	$\top$	$\perp$	Σ
QCDCL	33	137	170	**87**	**37**	**124**	**18**	**53**	**71**	**30**	55	85
QCDCL3	33	139	172	80	30	110	14	50	64	**30**	**58**	**88**
QCDCL$^{\text{Any}}$	29	130	159	73	31	104	10	33	43	12	35	47
QCDCL$^{\text{Uni-Any}}$	**35**	**169**	**204**	74	32	106	7	28	35	10	42	52
QCDCL$^{\text{Exi-Any}}$	29	112	141	59	28	87	9	28	37	15	38	53
VBS	41	195	236	95	42	137	18	53	71	34	61	95

on average, more than 99% in over 70% of cases). Thus, the advantage of the specialized systems is unlikely to be explicable solely by the *quantity* of asserting constraints, but rather by their *quality*. This is also supported by the erratic performance of several of the variants on both $\texttt{TwinCR}$ and Rev($\texttt{TwinModEq}$)—it appears that the existing short runs are hard for the solver to discover. Investigating this properly might require opening up the solver even more, and recording decisions and other details of the search path. We want to keep this paper focused on the theory part, and leave further investigation of this behaviour to future work.

6.2 QBF Evaluations

Table 1 and Figs. 5, 6, 7, and 8 show the performance on PCNF and QCIR (circuit) formulas from the QBF Evaluations (QBFEval) 2020 and 2022. Even though the theoretical part is concerned with PCNF formulas only, here we evaluate the algorithms on circuit formulas as well, as the circuit format is a standard part of QBF Evaluations (in fact, it is preferred due to its greater flexibility for both encoding and solving). Circuit formulas are internally translated into a pair of PCNF formulas by Qute.

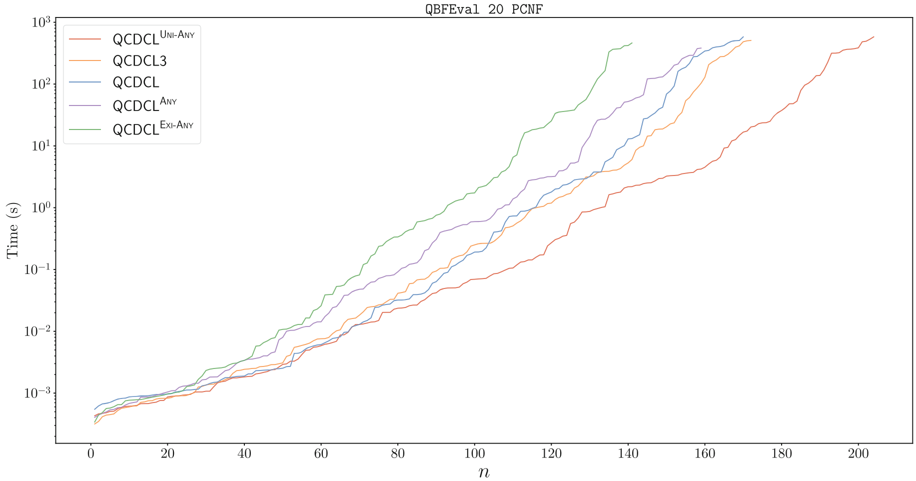

Fig. 5 Performance on QBFEval 2020 PCNF instances. Cactus plot: (x, y) means the configuration solved x instances in y seconds. Lower and right is better

In Fig. 5, we see a decisive victory of QCDCL$^{\text{Uni-Any}}$, which beat the second QCDCL3 by a margin of 32 solved instances. QCDCL$^{\text{Uni-Any}}$ solved 27 instances from this benchmark set uniquely, of which 26 were false formulas. These 27 uniquely solved formulas include application formulas encoding bounded model checking problems, as well as several crafted formulas.

In all other cases, the winner is vanilla QCDCL; twice QCDCL, once QCDCL3. This, as well as the relative ranking of QCDCL and QCDCL3 in Fig. 5, proves that the 3-watched-literal scheme, a by-product of the implementation, considered in its own right, is in fact competitive with the traditional 2-watched literal scheme, at least on these formulas.

With the already mentioned exception of QCDCL$^{\text{Uni-Any}}$, no other QCDCL variant beats vanilla QCDCL in any of the other cases. Each of QCDCL$^{\text{Any}}$, QCDCL$^{\text{Uni-Any}}$, QCDCL$^{\text{Exi-Any}}$ beats the other two on at least one benchmark set. In all cases except PCNF 22, the virtual best solver (VBS) is strictly better than any individual algorithm, meaning that there were always formulas not solver by the best variant, which were solved by another variant.

Such mixed results should perhaps not surprise. The act of performing out-of-order decisions amounts to revealing a future move in the game earlier than forced to. This should be advantageous, philosophically speaking, when strong moves exist that can already be played early. It is not clear how often such situations should arise in application formulas, into which they are not baked the way they are into separation formulas.

In any case, the experiments show that both the new algorithms as well as the technical implementation are competitive. Further analysis would be needed to determine whether there are application formula families on which one QCDCL variant is significantly better then others. We provide all of our experimental data as supplementary material.

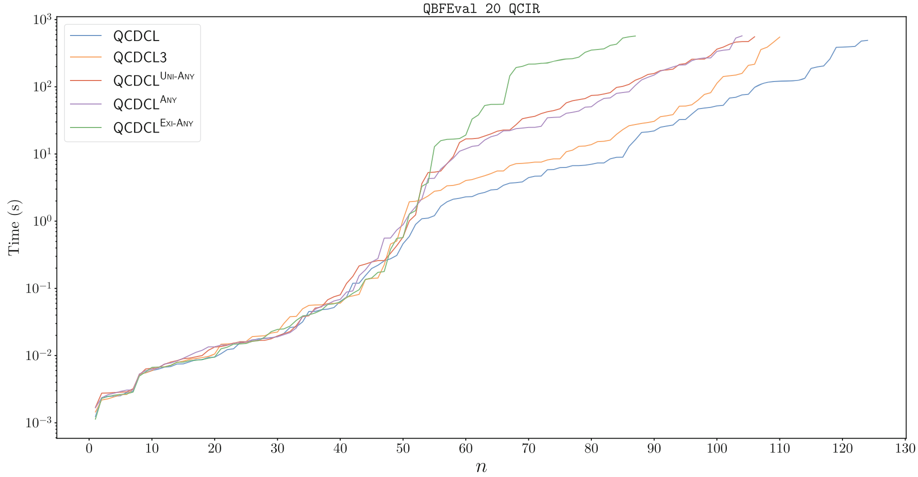

Fig. 6 Performance on QBFEval 2020 QCIR instances. Cactus plot: (x, y) means the configuration solved x instances in y seconds. Lower and right is better

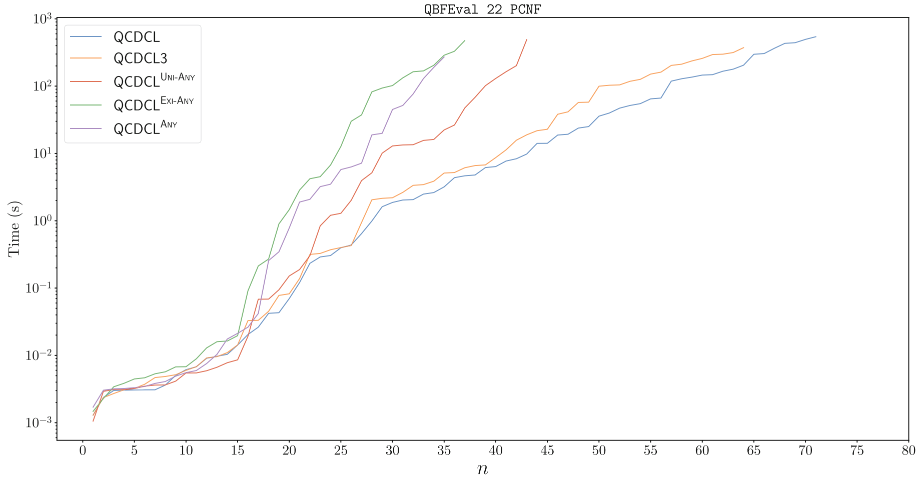

Fig. 7 Performance on QBFEval 2022 PCNF instances. Cactus plot: (x, y) means the configuration solved x instances in y seconds. Lower and right is better

7 Conclusion

We have laid the theoretical foundations for new flavours of QCDCL with the ability to ignore all quantification order for decisions. In this paper we focused on proof complexity, showing exponential advantage for the new systems over vanilla QCDCL. We complemented this with a proof-of-concept implementation in Qute, which validates the feasibility of our approach. Our preliminary experiments on crafted formulas already raise some interesting questions about poor solver performance on theoretically easy formulas.

As part of future work, we plan to advance on the practical front, polishing and possibly improving the implementation technically, performing a more thorough experimental evaluation, and combining the approaches presented here with other techniques like Qute's native

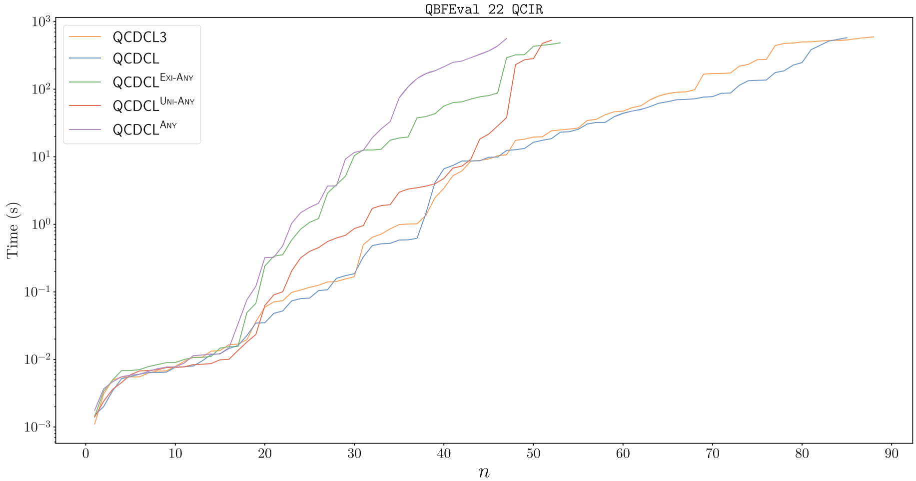

Fig. 8 Performance on QBFEval 2022 QCIR instances. Cactus plot: (x, y) means the configuration solved x instances in y seconds. Lower and right is better

dependency learning (and possibly dependency schemes). We would also like to dive deeper into the analysis of how learning asserting constraints affects solver performance.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Atserias, A., Fichte, J.K., Thurley, M.: Clause-learning algorithms with many restarts and bounded-width resolution. J. Artif. Intell. Res. **40**, 353–373 (2011)
2. Balabanov, V., Jiang, J.-H.R.: Unified QBF certification and its applications. Form. Methods Syst. Des. **41**(1), 45–65 (2012)
3. Beame, P., Kautz, H.A., Sabharwal, A.: Towards understanding and harnessing the potential of clause learning. J. Artif. Intell. Res. **22**, 319–351 (2004)
4. Beyersdorff, O., Böhm, B.: QCDCL with cube learning or pure literal elimination—what is best? Electron. Colloquium Comput. Complex. 131 (2021)
5. Beyersdorff, O., Böhm, B.: Understanding the relative strength of QBF CDCL solvers and QBF resolution. In: Proceedings of Innovations in Theoretical Computer Science (ITCS), pp. 12–11220 (2021)
6. Beyersdorff, O., Blinkhorn, J., Hinde, L.: Size, cost, and capacity: a semantic technique for hard random QBFs. Logical Methods Comput. Sci. **15**(1), 13 (2019)
7. Beyersdorff, O., Janota, M., Lonsing, F., Seidl, M.: Quantified Boolean formulas. In: Biere, A., Heule, M., Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability. Frontiers in Artificial Intelligence and Applications, pp. 1177–1221. IOS Press, Amsterdam (2021)
8. Böhm, B., Peitl, T., Beyersdorff, O.: Should decisions in QCDCL follow prefix order? In: Meel, K.S., Strichman, O. (eds.) 25th International Conference on Theory and Applications of Satisfiability Testing (SAT). LIPIcs, Vol. 236, pp. 11–11119. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022)

9. Böhm, B., Beyersdorff, O.: Lower bounds for QCDCL via formula gauge. In: Li, C.-M., Manyà, F. (eds.) Theory and Applications of Satisfiability Testing—SAT 2021, pp. 47–63. Springer, Cham (2021)
10. Buss, S., Nordström, J.: Proof complexity and SAT solving. In: Biere, A., Heule, M., Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability. Frontiers in Artificial Intelligence and Applications, pp. 233–350. IOS Press, Amsterdam (2021)
11. Cadoli, M., Giovanardi, A., Schaerf, M.: An algorithm to evaluate quantified Boolean formulae. In: Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference (AAAI), pp. 262–267 (1998)
12. Cook, S.A., Reckhow, R.A.: The relative efficiency of propositional proof systems. J. Symb. Logic **44**(1), 36–50 (1979)
13. Giunchiglia, E., Marin, P., Narizzano, M.: Reasoning with quantified Boolean formulas. In: Biere, A., Heule, M., Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability. Frontiers in Artificial Intelligence and Applications, vol. 185, pp. 761–780. IOS Press, Amsterdam (2009)
14. Heule, M.: Proofs of unsatisfiability. In: Biere, A., Heule, M., Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability. Frontiers in Artificial Intelligence and Applications, 2nd edn., pp. 635–668. IOS Press, Amsterdam (2021)
15. Heule, M.J.H., Seidl, M., Biere, A.: Solution validation and extraction for QBF preprocessing. J. Autom. Reason. **58**(1), 97–125 (2017)
16. Hoos, H.H., Peitl, T., Slivovsky, F., Szeider, S.: Portfolio-based algorithm selection for circuit qbfs. In: Hooker, J.N. (ed.) Principles and Practice of Constraint Programming - 24th International Conference, CP 2018, Lille, France, August 27-31, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11008, pp. 195–209. Springer (2018). https://doi.org/10.1007/978-3-319-98334-9_13
17. Janota, M.: On Q-Resolution and CDCL QBF solving. In: Proceedings of International Conference on Theory and Applications of Satisfiability Testing (SAT), pp. 402–418 (2016)
18. Janota, M., Marques-Silva, J.: Expansion-based QBF solving versus Q-resolution. Theor. Comput. Sci. **577**, 25–42 (2015)
19. Janota, M., Marques-Silva, J.: An Achilles' heel of term-resolution. In: Oliveira, E., Gama, J., Vale, Z., Lopes Cardoso, H. (eds.) Progress in Artificial Intelligence, pp. 670–680. Springer, Cham (2017)
20. Kleine Büning, H., Karpinski, M., Flögel, A.: Resolution for quantified Boolean formulas. Inf. Comput. **117**(1), 12–18 (1995)
21. Lonsing, F., Egly, U.: DepQBF 6.0: A search-based QBF solver beyond traditional QCDCL. In: Proceedings of International Conference on Automated Deduction (CADE), pp. 371–384 (2017)
22. Lonsing, F., Egly, U.: Evaluating QBF solvers: Quantifier alternations matter. In: Proceedings of Principles and Practice of Constraint Programming (CP), pp. 276–294. Springer (2018)
23. Lonsing, F.: Dependency schemes and search-based QBF solving: Theory and practice. PhD thesis, Johannes Kepler University Linz (2012)
24. Marques Silva, J.P., Lynce, I., Malik, S.: Conflict-driven clause learning SAT solvers. In: Biere, A., Heule, M., Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability. Frontiers in Artificial Intelligence and Applications. IOS Press (2021)
25. Marques Silva, J.P., Sakallah, K.A.: GRASP - a new search algorithm for satisfiability. In: Proceedings of IEEE/ACM International Conference on Computer-aided Design (ICCAD), pp. 220–227 (1996)
26. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: engineering an efficient sat solver. In: Proceedings of the 38th Design Automation Conference (IEEE Cat. No.01CH37232), pp. 530–535 (2001). 11.1145/378239.379017
27. Peitl, T., Slivovsky, F., Szeider, S.: Combining resolution-path dependencies with dependency learning. In: Janota, M., Lynce, I. (eds.) Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11628, pp. 306–318. Springer (2019). https://doi.org/10.1007/978-3-030-24258-9_22
28. Peitl, T., Slivovsky, F., Szeider, S.: Long-distance Q-resolution with dependency schemes. J. Autom. Reason. **63**(1), 127–155 (2019)
29. Peitl, T., Slivovsky, F., Szeider, S.: Dependency learning for QBF. J. Artif. Intell. Res. **65**, 180–208 (2019)
30. Pipatsrisawat, K., Darwiche, A.: On the power of clause-learning SAT solvers as resolution engines. Artif. Intell. **175**(2), 512–525 (2011)
31. Samer, M., Szeider, S.: Backdoor sets of quantified Boolean formulas. J. Autom. Reason. **42**(1), 77–97 (2009). https://doi.org/10.1007/s10817-008-9114-5
32. Shukla, A., Biere, A., Pulina, L., Seidl, M.: A survey on applications of quantified Boolean formulas. In: Proceedings of IEEE International Conference on Tools with Artificial Intelligence (ICTAI), pp. 78–84 (2019)

33. Slivovsky, F., Peitl, T., Perebor, Heisinger, M.: Fslivovsky/qute: Out-of-order Decisions. https://doi.org/10.5281/zenodo.10149885
34. Slivovsky, F., Szeider, S.: Soundness of Q-resolution with dependency schemes. Theor. Comput. Sci. **612**, 83–101 (2016)
35. Tange, O.: GNU Parallel - The command-line power tool. login: The USENIX Magazine February, 42–47 (2011)
36. Vinyals, M.: Hard examples for common variable decision heuristics. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI) (2020)
37. Zhang, L., Madigan, C.F., Moskewicz, M.W., Malik, S.: Efficient conflict driven learning in Boolean satisfiability solver. In: Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 279–285 (2001)
38. Zhang, L., Malik, S.: Conflict driven learning in a quantified Boolean satisfiability solver. In: Proceedings of IEEE/ACM International Conference on Computer-aided Design (ICCAD), pp. 442–449 (2002)