Dependency Schemes in CDCL-Based QBF Solving: A Proof-Theoretic Study

Abhimanyu Choudhury
The Institute of Mathematical Sciences, Chennai, India
Homi Bhabha National Institute, Training School Complex, Anushaktinagar, Mumbai, India

Meena Mahajan
The Institute of Mathematical Sciences, Chennai, India
Homi Bhabha National Institute, Training School Complex, Anushaktinagar, Mumbai, India

Abstract

In Quantified Boolean Formulas QBFs, dependency schemes help to detect spurious or superfluous dependencies that are implied by the variable ordering in the quantifier prefix but are not essential for constructing countermodels. This detection can provably shorten refutations in specific proof systems, and is expected to speed up runs of QBF solvers. The proof system QCDCL recently defined by Beyersdorff and Böhm (LMCS 2023) abstracts the reasoning employed by QBF solvers based on conflict-driven clause-learning (CDCL) techniques. We show how to incorporate the use of dependency schemes into this proof system, either in a preprocessing phase, or in the propagations and clause learning, or both. We then show that when the reflexive resolution path dependency scheme $\mathtt{D}^{\mathtt{rrs}}$ is used, a mixed picture emerges: the proof systems that add $\mathtt{D}^{\mathtt{rrs}}$ to QCDCL in these three ways are not only incomparable with each other, but are also incomparable with the basic QCDCL proof system that does not use $\mathtt{D}^{\mathtt{rrs}}$ at all, as well as with several other resolution-based QBF proof systems. A notable fact is that all our separations are achieved through QBFs with bounded quantifier alternation.

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases QBF, CDCL, Resolution, Dependency schemes

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.38

Related Version *Full Version*: https://eccc.weizmann.ac.il/report/2023/061/

Acknowledgements Part of this work was done when the second author was at the Simons Institute for the Theory of Computing at Berkeley during March–May 2023, participating in the Extended Reunion of the program Satisfiability: Theory, Practice, and Beyond.

1 Introduction

Despite the NP-hardness of propositional satisfiability, SAT solvers today are amazingly efficient in solving real-world instances. The best algorithms solving SAT in practice are based on the paradigm of conflict-driven clause learning CDCL, that revolutionised SAT solving in the nineties. Such algorithms use a generic template as follows: repeatedly decide values of some variables, propagate hard constraints (unit clauses) until a conflict is reached, "learn" a new clause from the conflict, backtrack and continue. For unsatisfiable formulas, the learning process yields a refutation in the proof system Resolution, and it was shown over a decade ago that resolution proofs can themselves be mimicked within this framework, so CDCL equals Resolution, [20, 1]. Hence, a proof-complexity-theoretic analysis of Resolution has revealed deep insights into the strengths and limitations of this CDCL paradigm.

With the success of propositional SAT solvers, there are many ambitious attempts now to tackle more expressive/succinct formalisms. In particular, for the PSPACE-complete problem of deciding the truth of Quantified Boolean Formulas QBF, there are now many solvers, as well as a rich (and still growing) theory about the underlying formal proof systems. Designing solvers for QBFs is a useful enterprise because many industrial applications seem to lend themselves more naturally to expressions involving both existential and universal quantifiers; see for instance [23, 9].

The proof system Resolution can be lifted to the QBF setting in many ways. The "CDCL way" is to add a universal reduction rule, giving rise to the system Q-Res and the more general QU-Res. Allowing contradictory literals to be merged under certain conditions gives rise to the system long-distance Q-Resolution LDQ-Res.

Another "CDCL" way is to lift the CDCL algorithm itself to a QCDCL algorithm: decide values of variables, usually respecting the order of quantified alternation, propagate unit constraints, interpreting unit modulo universal reductions, repeat until a conflict is reached, learn a new clause, backtrack and continue. For false formulas, the learning process yields a long-distance Q-resolution refutation. However, the QCDCL refutation itself is much more restricted than an LDQ-Res refutation. In [7], a formal proof system QCDCL was abstracted out of the QCDCL algorithm. Noting that potentially the decision policy and the propagation policy could be modified, the authors of [7] actually formalised four different QCDCL-based proof systems. The system underlying most solvers is $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{RED}}$, which we refer to as QCDCL without any sub/super-script; for the other systems we explicitly write the policies.

While the QCDCL proof system explains the correctness of solvers for false QBFs, it ignores cube-learning from satisfying assignments. In practice, cube-learning is essential to the completeness of a QCDCL solver; it is integral to proving a QBF to be true. (A QBF solver algorithm does not know in advance whether the input formula is true or false. It learns clauses and cubes, and concludes false/true when the empty clause/cube has been learnt.) The choice to ignore this was made in [7] because the focus there (as also here) was on refutational proof systems, proving QBFs false. In this setting, cube-learning is not an essential ingredient. However, in [13], the authors defined the system $\mathtt{QCDCL}^{\mathtt{cube}}$ that incorporates cube-learning on top of the original QCDCL, and found that cube-learning was in fact advantageous even in constructing shorter refutations for false QBFs. More recent work (see [12]) in fact redefines QCDCL as the augmented system with cube-learning incorporated. For this paper, we retain the notation from [7]. An interesting outcome from [12], though not directly relevant to our work, is that extracted refutations do not capture the full generality of LDQ-Res.

DepQBF [16] is the leading QCDCL solver and has many versions. Its base version still employs what the authors call "vanilla QCDCL", and its behaviour on false QBFs is explained by the proof system QCDCL which we are interested in exploring. Later versions of DepQBF provide options of turning on "cube learning" (when "turned on", its behaviour is explained by the proof system $\mathtt{QCDCL}^{\mathtt{cube}}$) and also offer heuristics like whether or not to allow "dependency scheme aware propagation" and/or apply "pure literal elimination".

A heuristic that has been found to be useful in many QBF solvers, and has been formalised in proof systems, is to eliminate easily-detectable spurious dependencies. In a prenex QBF, a variable "depends" on the variables preceding it in the quantifier prefix; where "depends" means that a Herbrand/Skolem function for the variable is a function of the preceding variables. However, a Herbrand function or countermodel may not really need to know the values of all preceding variables. A dependency scheme filters out as many of such unnecessary dependencies as it can detect, producing what is in effect a Dependency QBF, DQBF. Although DQBF is a significantly richer formalism that is known to be NEXP-complete (see [2, 22]), these heuristics are not aiming to solve DQBFs in general. Rather, they algorithmically detect spurious dependencies and disregard them as the algorithm

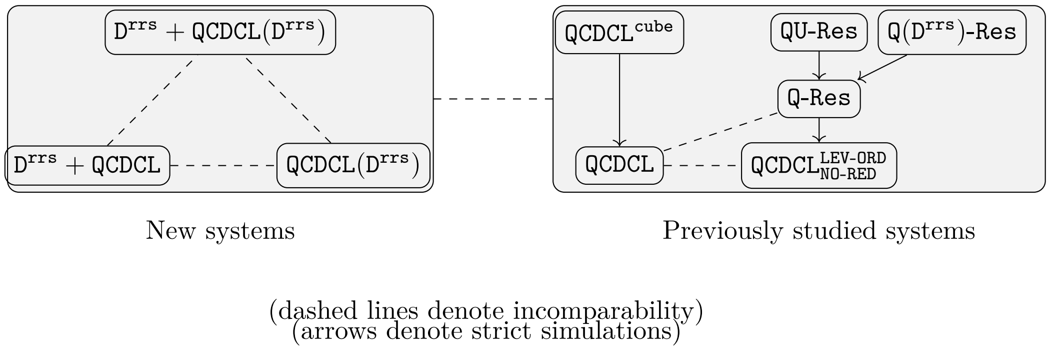

Figure 1 Relations between proof systems.

proceeds. See [15, 21, 16, 17] for early work on this topic. Often the use of a dependency scheme makes the solvers run faster. Now, the universal reduction rule in the proof systems (say in Q-Res, LDQ-Res) can be applied in more settings because there are fewer dependencies, and this can shorten refutations significantly. See for instance [10, 24, 19]. Note that the use of a dependency scheme must be proven to be sound and complete, and this in itself is often quite involved. The notion of a dependency scheme being "normal" was introduced in [19], where it is shown that adding any normal dependency scheme to LDQ-Res preserves soundness and completeness.

In this paper, we examine how the usage of a dependency scheme can affect proof systems underlying the QCDCL algorithm. As far as we are aware, such a theoretical study has not been undertaken before, even though many current QBF solvers are based on the QCDCL paradigm and also do use dependency schemes. Specifically, we focus on the proof system QCDCL (in the notation of [7], the $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{RED}}$ proof system), underlying most QCDCL-based solvers, and on the dependency scheme $\mathtt{D}^{\mathtt{rrs}}$ which has been studied in the context of Q-Res and LDQ-Res, see [24, 10, 19]. We note that the proof system QCDCL can be made aware of dependency schemes in more than one way. We identify two natural ways: (1) use a dependency scheme D to preprocess the formula, performing reductions in the initial clauses whenever permitted by the scheme, and (2) use a dependency scheme D in the QCDCL algorithm itself, in enabling unit propagations and in learning clauses. Denoting the first way as $\mathtt{D} + \mathtt{QCDCL}$ and the second as $\mathtt{QCDCL}(\mathtt{D})$, and noting that we could even use different dependency schemes in both these ways, we obtain the system $\mathtt{D}_1 + \mathtt{QCDCL}(\mathtt{D}_2)$. When $\mathtt{D}_1$ and $\mathtt{D}_2$ are both the trivial dependency scheme $\mathtt{D}^{\mathtt{trv}}$ inherited from the linear order of the quantifier prefix, this system is exactly QCDCL.

Our contributions are as follows:

1. We formalise the proof system $\mathtt{D}' + \mathtt{QCDCL}(\mathtt{D})$ for dependency schemes $\mathtt{D}, \mathtt{D}'$, and note that whenever $\mathtt{D}', \mathtt{D}$ are normal schemes, $\mathtt{D}' + \mathtt{QCDCL}(\mathtt{D})$ is sound and complete (Theorem 3.2).
2. For $\mathtt{D}, \mathtt{D}' \in \{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{rrs}}\}$, we study the four systems $\mathtt{D}' + \mathtt{QCDCL}(\mathtt{D})$. As observed above, one of them is QCDCL itself, while the others are new systems. We compare these systems with each other and show that they are all pairwise incomparable (Theorem 5.1). We also show that each of them is incomparable with each of the systems $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$, Q-Res, $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})$-Res, and QU-Res(Theorem 5.2), as well as with $\mathtt{QCDCL}^{\mathtt{cube}}$(Theorem 5.3).

Relations among various proof systems are shown in Figure 1.

In other words, making QCDCL algorithms dependency-aware is a "mixed bag": in some situations this shortens runs while in others it is disadvantageous. Here are our thoughts on what this actually means.

That $\mathtt{QCDCL(D)}$ is stronger than $\mathtt{QCDCL}$ at times is to be expected; after all, that is why the heuristic evolved. That it can be weaker at times appears a bit surprising until one recalls that even when $\mathtt{QCDCL}$ was formalised in [7], it was shown that the no-reduction version $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$ can have an advantage over $\mathtt{QCDCL}$; for some formulas, enabling more reductions and unit propagations can send the trails down into a trap where refuting a hard sub-formula becomes inevitable. Since dependency schemes do exactly this enabling of more reductions and propagations, custom formulas can be designed where the difference is not just between no-reductions and reductions, but also between reductions and dependency-aware reductions. This is a consequence of the level-ordering of decisions and the forcing of all unit propogations with reduction, and may not hold for the other variants of $\mathtt{QCDCL}$.

That $\mathtt{D} + \mathtt{QCDCL}$ can be stronger than $\mathtt{QCDCL}$ is again to be expected. That it can be weaker seems really counter-intuitive, but is again related to the comment above: the preprocessing shortens clauses and thus enables more unit propagations in subsequent trails.

One direction of our separation between $\mathtt{D} + \mathtt{QCDCL}$ and $\mathtt{QCDCL(D)}$ was genuinely surprising to us. We construct formulas where after preprocessing (as in $\mathtt{D} + \mathtt{QCDCL}$) the resulting formula is propositional and easy to refute in Resolution, and hence the original formula is easy to refute in $\mathtt{D} + \mathtt{QCDCL}$. However, the same formula is hard for $\mathtt{QCDCL(D)}$, Section 4.5! In other words, it is not enough for the QCDCL algorithm to be dependency-aware; this awareness must be achieved at the right stage of the algorithm.

The fact that $\mathtt{QCDCL(D)}$, $\mathtt{D} + \mathtt{QCDCL}$, and $\mathtt{D} + \mathtt{QCDCL(D)}$ are all incomparable with $\mathtt{QCDCL}^{\mathsf{cube}}$ is note-worthy and interesting as allowing for cube-learning always adds strength and makes things easier as compared to without cube-learning; $\mathtt{QCDCL}^{\mathsf{cube}}$ as a proof system is known to be strictly more powerful than $\mathtt{QCDCL}$ [13]. Our results show that switching on cube-learning (which most current solvers do by default) and switching on dependency-awareness as proposed here are orthogonal options. Which option is better may depend on the setting from which the instances to be solved arise.

This work is based on formalisms in [24, 10, 19, 7]. See [7] for an extensive bibliography of relevant work.

The rest of this paper is organised as follows. After spelling out notation and required preliminaries in Section 2, including defining dependency schemes and describing the $\mathtt{QCDCL}$ proof system, we show in Section 3 that the addition of normal dependency schemes results in sound and complete proof systems. In Section 4 we present, for some previously studied formulas as well as for some newly designed formulas, lower and/or upper bounds in the $\mathtt{D}_1 + \mathtt{QCDCL(D_2)}$ systems when $\mathtt{D}_1, \mathtt{D}_2$ are in $\{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{rrs}}\}$. Using these bounds, we conclude in Section 5 that these new systems are pairwise incomparable with each other as well as with each of $\mathtt{QCDCL}$, $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$, $\mathtt{Q\text{-}Res}$, $\mathtt{Q(D^{rrs})\text{-}Res}$, $\mathtt{QU\text{-}Res}$, $\mathtt{QCDCL}^{\mathsf{cube}}$. We end with some concluding remarks in Section 6.

Some proofs are briefly sketched due to space constraints; full details can be found in [14].

2 Preliminaries

2.1 Basics

A Quantified Boolean Formula in prenex conjunction normal form (PCNF) consists of a prefix with an ordered list of variables, each quantified either existentitally or universally, and the matrix, which is a set of clauses over these variables. That is, it has the form

$$\Phi = \mathcal{Q}\vec{x} \cdot \varphi = Q_1 x_1 Q_2 x_2 \dots Q_n x_n \varphi(x_1, x_2, \dots, x_n)$$

where φ is a propositional formula in CNF.

The formula is true if there are (Skolem) functions s_i for each existentially quantified variable x_i, where each such s_i depends only on universally quantified variables x_j with $j < i$, such that substituting these s_i in φ yields a tautology. Similarly, the formula is false if there are (Herbrand) functions h_i for each universally quantified variable x_i, where each such h_i depends only on existentially quantified variables x_j with $j < i$, such that substituting h_i in φ yields an unsatisfiable formula.

In this paper, we focus on false formulas; refutations must rule out the existence of Skolem functions. In the proof system `Q-Res`, a refutation of a false QBF is a derivation of the empty clause $\square$ from the clauses in the matrix, using two rules: Resolution (from $A = C \vee x$ and $B = D \vee \neg x$, derive $C \vee D$, provided the pivot x is existential and $C \vee D$ is not tautological. We denote this as $C \vee D = \texttt{res}(A, B, x)$), and Universal Reduction (from $C \vee u$ derive C if u is universal and no existential variable in C appears to the right of u in the prefix). The proof system `QU-Res` generalises `Q-Res` by allowing resolution on universal pivots as well. The proof system `LDQ-Res` generalises `Q-Res` in a different way, allowing the derivation of seemingly-tautological clauses in resolution under certain conditions: a universal variable u appearing in opposite polarities in C and D is represented as the merged literal u^* in the resolvent, provided it is to the right of the pivot x.

A proof system `P` simulates a proof system `Q` if, for every formula, the size of the shortest `P` refutation is polynomial in the size of the shortest `Q` refutation. The systems `QU-Res` and `LDQ-Res` are both strictly more powerful than `Q-Res` and incomparable with each other.

For a set S of clauses and a literal ℓ, we use shorthand $\ell \vee S$ to denote the set of clauses $\{\ell \vee C \mid C \in S\}$.

2.2 Dependency Schemes

In a PCNF formula Φ, for any existential variable x, any witnessing Skolem function $s(x)$ (if such functions exist) can depend only on those universal variables that are quantified to the left of x in the quantifier prefix. The *trivial dependency scheme* $\texttt{D}^{\texttt{trv}}$ records this: for a PCNF formula Φ, the relation $\texttt{D}^{\texttt{trv}}(\Phi)$ consists of all pairs (u, x) where existential variable x appears to the right of universal variable u in the order of quantifier prefix. If a pair (u, x) appears in $\texttt{D}^{\texttt{trv}}(\Phi)$, we say that x depends on u.

A dependency scheme aims to weed out redundant pairs from $\texttt{D}^{\texttt{trv}}$ while preserving the (non)-existence of Skolem functions. Given a PCNF formula Φ, it maps each variable x to a set of variables whose values should be sufficient to determine how to set x. The most basic of dependency schemes weeds out nothing and is the trivial dependency scheme $\texttt{D}^{\texttt{trv}}$ itself. A non-trivial dependency scheme $\texttt{D}$ produces, for any formula Φ, a subset $\texttt{D}(\Phi)$ of the trivial dependencies $\texttt{D}^{\texttt{trv}}(\Phi)$; it does not introduce new dependencies. Some non-trivial schemes are the standard scheme $\texttt{D}^{\texttt{std}}$ and the reflexive resolution path scheme $\texttt{D}^{\texttt{rrs}}$; see [24]. Roughly speaking, in $\texttt{D}^{\texttt{rrs}}$, (u, x) is in the dependency relation of a formula if $(u, x) \in \texttt{D}^{\texttt{trv}}$ and there is a sequence of clauses with the first containing u, the last containing $\bar{u}$, some intermediate consecutive clauses containing x and $\bar{x}$, and where each pair of consecutive clauses has an existential variable, quantified after u, in opposite polarities. The non-existence of such a sequence implies that if at all there are Skolem functions for x, then there exists a Skolem function for x which does not use information about u; hence x need not depend on u. Formally, the dependence scheme is defined as follows:

▶ **Definition 2.1** (Reflexive Resolution Path Dependency Scheme, [24]). *For a QBF $\Phi = \mathcal{Q}\phi$, the pair (u, x) is in $\texttt{D}^{\texttt{rrs}}(\Phi)$ if and only if $(u, x) \in \texttt{D}^{\texttt{trv}}(\Phi)$ and there exists a sequence of clauses $C_1, \cdots, C_n \in \phi$ and a sequence of literals $l_1, \cdots, l_{n-1}$ such that:*

- $u \in C_1$ *and* $\bar{u} \in C_n$,
- $x = var(l_i)$ *for some* $i \in [n-1]$,
- $var(l_i) \neq var(l_{i+1})$ *for each* $i \in [n-2]$, *and*
- $(u, var(l_i)) \in \mathtt{D}^{\mathtt{trv}}(\Phi)$, $l_i \in C_i$ *and* $\bar{l}_i \in C_{i+1}$ *for each* $i \in [n-1]$.

For a dependency scheme D, a QBF Φ, a universal literal $\ell_u \in \{u, \bar{u}\}$ and an existential literal $\ell_x \in \{x, \bar{x}\}$, we say that ℓ_x blocks ℓ_u if $(u, x) \in \mathtt{D}(\Phi)$; in particular, this implies that x is quantified after u. For a clause C we denote by red-D(C) the subclause obtained by removing all universal literals which are not blocked by any other literal in C. We denote by red-D(Φ) the QBF Ψ obtained by replacing each clause C in the matrix of Φ with the clause red-D(C). When $\mathtt{D} = \mathtt{D}^{\mathtt{trv}}$, we use the notation red(C) and red(Φ).

The proof systems Q-Res and LDQ-Res, augmented with a dependency scheme D [19], permit universal reduction of u under the more relaxed requirement that $(u, x) \notin \mathtt{D}$ for any existential variable $x \in C$. That is, they permit the derivation of red-D(C) from C.

In a QBF with a leading universal block, universal variables from the first block appear in a LDQ-Res derivation in only one polarity. This feature is useful in proving soundness of LDQ-Res. Soundness of LDQ(D)-Res can be proven in a similar vein, provided the dependency scheme D does not permit derivations violating this property, and provided that applying a partial assignment can possibly erase existing dependencies but cannot create new ones. This leads to the definition of the following important subclass of dependency schemes, the normal dependency schemes.

▶ **Definition 2.2** (Normal Dependency Scheme, [19]). *A dependency scheme* D *is*

- *monotone if for every PCNF formula ϕ, and every assignment τ to a subset of $var(\phi)$, $\mathtt{D}(\phi[\tau]) \subseteq \mathtt{D}(\phi)$. (Here $\phi[\tau]$ is the restriction of ϕ obtained by applying the partial assignment τ to it.)*
- *simple if for every PCNF formula Φ of the form $\Phi = \forall X_1 \exists Y_1 \cdots \forall X_n \exists Y_n \phi$, every (minimal)* LDQ(D)-Res *derivation P of a clause from Φ, and every $u \in X_1$, either u or $\bar{u}$ does not appear in P.*
- *normal if it is both monotone and simple.*

Normal dependency schemes are important because if D is normal, then LDQ(D)-Res is a sound proof system [19]. The dependency schemes $\mathtt{D}^{\mathtt{trv}}$, $\mathtt{D}^{\mathtt{std}}$, $\mathtt{D}^{\mathtt{rrs}}$ are all normal.

2.3 The proof system QCDCL

This proof system QCDCL defined in [7] formalises the reasoning in QCDCL algorithms. A refutation of a false QBF is a sequence of triples of the form (T, C, π) where T is a trail (in the QCDCL algorithm) ending in a conflict, C is the clause learnt from this trail, and π is the LDQ-Res derivation of C explaining how C is learnt. (Recall that in (Q)CDCL, a trail is a sequence of literals, some of which are set due to decisions made by the algorithm and the others are propagated literals set due to a unit constraint. Following the standard convention, we denote decision literals in a trail in boldface. A trail thus has the form

$$T := (p_{(0,1)}, \cdots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \cdots p_{(1,g_1)}; \mathbf{d_2}, \cdots\cdots\cdots; \mathbf{d_r}, p_{(r,1)}, \cdots p_{(r,g_r)})$$

where the literals d_i are decision literals, and the literals $p_{i,j}$ are propagated literals.) From the last triple in the sequence we can learn the empty clause, completing the refutation. Three factors affect the construction of the refutation.

1. The decision policy: how to choose the next variable to branch on. In standard QCDCL (i.e $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{RED}}$, the focus of this paper), decisions must respect the quantifier prefix level order. (Variables x, y are at the same level if they are quantified the same way, and no variable with a different quantification appears between them in the prefix order.) Other policies such as ASS-ORD, ASS-R-ORD, UNI-ANY, are also possible; see [7, 11].
2. The unit propagation policy. Upon a partial assignment α to some variables, when does a clause C propagate a literal? In the No-Reduction policy, a clause C is unit if exactly one literal ℓ of C is unset, and this literal is propagated. In the Reduction policy, used by most current QCDCL solvers [16, 18], a clause C propagates literal ℓ if after restricting C by α and applying all possible universal reductions, only ℓ remains. In standard QCDCL the Reduction policy is used. Also, propagations are made as soon as possible; see the description of natural trails (Def 3.4 in [7].
   In the notation of [7], for a decision policy P and a propagation policy R, the corresponding QCDCL proof system is denoted $\texttt{QCDCL}^{P}_{R}$. Thus standard QCDCL is $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{RED}}$. Other variants are also defined in [7]; in particular $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$.
3. The set of learnable clauses. These explain the conflict at the end of a trail.
   ▶ **Definition 2.3** (learnable clauses). *From a trail*

$$T := (p_{(0,1)}, \cdots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \cdots p_{(1,g_1)}; \mathbf{d_2}, \cdots\cdots\cdots; \mathbf{d_r}, p_{(r,1)}, \cdots p_{(r,g_r)})$$

ending in a conflict $p_{(r,g_r)} = \square$, the sequence L_T of learnable clauses has a clause associated with each propagation in the trail, and one more clause, described by tracing the conflict backwards through the trail as follows. ($\texttt{ante}(\ell)$ *denotes the clause that causes literal ℓ to be propagated; i.e. the antecedent.)*

- $C_{(r,g_r)} = \texttt{red}(\texttt{ante}(p_{(r,g_r)}))$.
- *For $i \in \{0, 1, \cdots, r\}$ and $j \in [g_i - 1]$,*

$$C_{(i,j)} = \begin{cases} \texttt{red}[\texttt{res}(C_{(i,j+1)}, \texttt{red}(\texttt{ante}(p_{(i,j)})), p_{(i,j)})] & \text{if } \bar{p}_{(i,j)} \in C_{(i,j+1)} \\ C_{(i,j+1)} & \text{otherwise} \end{cases}$$

- *For $i \in \{0, 1, \cdots, r-1\}$.*

$$C_{(i,g_i)} = \begin{cases} \texttt{red}[\texttt{res}(C_{(i+1,1)}, \texttt{red}(\texttt{ante}(p_{(i,g_i)})), p_{(i,g_i)})] & \text{if } \bar{p}_{(i,g_i)} \in C_{(i+1,1)} \\ C_{(i+1,1)} & \text{otherwise} \end{cases}$$

In the above formulation of the QCDCL system, we only consider trails that end in a conflict. Trails ending in a satisfying assignment are ignored. This is enough to ensure refutational completeness – the ability to prove all false QBFs false. From satisfying assignments, solvers can learn cubes (or terms), and this is necessary to prove true QBFs true. In [13] it was shown that allowing cube (or term) learning from satisfying assignments can also be advantageous while refuting false QBFs. This led to the definition of the proof system $\texttt{QCDCL}^{\texttt{cube}}$, which was shown to be strictly stronger than the standard QCDCL system i.e. $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{RED}}$.

Our focus, however, is on adding dependencies to the basic QCDCL system without cube learning, so wherever we talk about QCDCL as a proof system we refer to $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{RED}}$.

3 Adding Dependency Schemes to the QCDCL proof system

We first describe the generic addition of dependency schemes to QCDCL, and then show soundness and completenes for *normal* schemes. For a decision policy P and a propagation policy R, the corresponding QCDCL proof system is denoted $\texttt{QCDCL}^{P}_{R}$. Adding a dependency scheme D to this system can affect P, R, as well as the set of learnable clauses.

For the decision policy $P = \texttt{LEV-ORD}$, which is the focus of this work, adding a dependency scheme $\texttt{D}$ does not affect the decision policy.

For the propagation policy, the notion of unit clauses depends on the universal reductions allowed, which in turn is affected by the dependency scheme. When $R = \texttt{NO-RED}$, no universal reductions are allowed anyway, so adding a dependency scheme to the proof system does not affect the policy. When $R = \texttt{RED}$, the definition of a unit propagation changes. A clause C propagates a literal ℓ at a position in the trail if the $\forall(\texttt{D})$-reduction of C restricted to the trail so far is a unit clause. That is, the partial assignment α specified by the trail so far does not satisfy C, and after restricting C by α, applying all universal reductions allowed by $\texttt{D}$ leaves the single literal ℓ; $\texttt{red-D}(C|_\alpha) = \{\ell\}$. We denote this propagation policy as $\texttt{RED} + \texttt{D}$.

The dependency scheme modifies the reduction rule, which modifies the set of learnable clauses. The set of learnable clauses is now defined in a similar way as in Definition 2.3, but replacing $\texttt{red}$ everywhere with $\texttt{red-D}$, the $\forall(\texttt{D})$ rule for universal reduction with respect to the dependency.

A completely different way in which a dependency scheme $\texttt{D}$ can be added to $\texttt{QCDCL}$ proof systems is by adding it as a preprocessing step, by applying the $\texttt{red-D}$ rule on the axioms (clauses of the given formula). That is, produce $\texttt{QCDCL}$ refutations of $\texttt{red-D}(\Phi)$ instead of Φ.

These two ways of adding dependency schemes to $\texttt{QCDCL}$ – (1) in the trail construction, propagation and learning itself, or (2) as pre-processing – can both be combined. For a particular dependency scheme $\texttt{D}$, we can think of three distinct proof systems:

- $\texttt{QCDCL}(\texttt{D})$: use $\texttt{D}$ for unit propagations and learning, but not for preprocessing.
- $\texttt{D} + \texttt{QCDCL}$: use $\texttt{D}$ only to preprocess the formula.
- $\texttt{D} + \texttt{QCDCL}(\texttt{D})$: use $\texttt{D}$ for preprocessing, and also during propagation and learning.

Going a step further, we can even use different dependency schemes in the preprocessing and in the actual trails. Thus, formally, we define the general proof system $\texttt{D}' + \texttt{QCDCL}(\texttt{D})$:

▶ **Definition 3.1** ($\texttt{D}' + \texttt{QCDCL}(\texttt{D})$ proof system)**.**
For a false QBF $\Psi = \mathcal{Q} \cdot \psi$ and a dependency scheme $\texttt{D}$, a $\texttt{QCDCL}(\texttt{D})$ derivation of a clause C from Ψ is a sequence of triples (T_i, C_i, π_i), or equivalently, a triple of sequences

$$\iota := ((T_1, \cdots, T_m), (C_1, \cdots, C_m), (\pi_1, \cdots, \pi_m))$$

where for each $i \in [m]$, the trail T_i follows policies $\texttt{LEV-ORD}$ and $\texttt{RED} + \texttt{D}$, each clause $C_j \in L_{T_j}$ is a clause learnable from T_j using the $\texttt{red-D}$ rule, and $C_m = C$. Each π_i is the derivation of C_i from $\mathcal{Q} \cdot (\psi \cup \{C_1, \cdots, C_{i-1}\})$ in $\texttt{LDQ}(\texttt{D})\texttt{-Res}$.

For a false QBF $\Phi = \mathcal{Q} \cdot \phi$ and dependency schemes $\texttt{D}, \texttt{D}'$, a $\texttt{D}' + \texttt{QCDCL}(\texttt{D})$ deriviation of a clause C from Φ is a $\texttt{QCDCL}(\texttt{D})$ derivation of C from $\Psi = \texttt{red-D}'(\Phi)$.

If $C = (\Box)$, then the derivation ι is called a refutation.

Note that $\texttt{QCDCL}$ is exactly the proof system $\texttt{D}^{\texttt{trv}} + \texttt{QCDCL}(\texttt{D}^{\texttt{trv}})$. Using other dependency schemes instead of $\texttt{D}^{\texttt{trv}}$ is a natural generalisation.

We now show that adding normal dependency schemes $\texttt{D}_1, \texttt{D}_2$ preserves soundness and completness. (The main focus of this work is the cases when $\texttt{D}_1, \texttt{D}_2$ are either $\texttt{D}^{\texttt{rrs}}$ or $\texttt{D}^{\texttt{trv}}$. However, the proofs we present below do not become any simpler for these special cases.)

▶ **Theorem 3.2.** *If $\texttt{D}_1$ and $\texttt{D}_2$ are normal dependency schemes, then $\texttt{D}_1 + \texttt{QCDCL}(\texttt{D}_2)$ is a sound and complete proof system.*

Proof. Soundness: Let ι be a $\texttt{D}_1 + \texttt{QCDCL}(\texttt{D}_2)$ refutation of a QBF Φ. By definition, this is a $\texttt{QCDCL}(\texttt{D}_2)$ refutation of the QBF $\Psi = \texttt{red-D}_1(\Phi)$. Now, every $\texttt{QCDCL}(\texttt{D})$ refutation has an underlying $\texttt{LDQ}(\texttt{D})\texttt{-Res}$ refutation. Thus from ι we can extract a $\texttt{LDQ}(\texttt{D}_2)\texttt{-Res}$ refutation Π of

Ψ. Since $\mathtt{D}_2$ is a normal dependency scheme, $\mathtt{LDQ}(\mathtt{D}_2)$-$\mathtt{Res}$ is a sound proof system [19], and therefore Ψ is a false QBF. By completeness of $\mathtt{LDQ\text{-}Res}$, there exists a $\mathtt{LDQ\text{-}Res}$ refutation Π' of Ψ. The reductions made to obtain Ψ from Φ, followed by the derivation steps in Π', gives a $\mathtt{LDQ}(\mathtt{D}_1)$-$\mathtt{Res}$ refutation Π'' of Φ. Since $\mathtt{D}_1$ is also a normal dependency scheme, $\mathtt{LDQ}(\mathtt{D}_1)$-$\mathtt{Res}$ is also sound, and hence, the existence of Π'' implies that Φ is a false QBF.

Completeness: In Theorem 3.16 of [7], $\mathtt{QCDCL}$ (denoted there as $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{RED}}$) is shown to be complete. Exactly the same proof, which is actually quite intricate, works also to show the completeness of $\mathtt{D}_1 + \mathtt{QCDCL}(\mathtt{D}_2)$. The idea is as follows: for a false formula Φ, in the 2-player evaluation game, the universal player has a winning strategy on Φ. Since each clause in Φ has a subclause in $\Psi = \mathtt{red}\text{-}\mathtt{D}_1(\Phi)$, the same strategy is also a winning strategy in the evaluation game on Ψ, so Ψ is false. Now, we can construct trails in level order that perform propagations whenever applicable, decide the polarity of existential variables arbitrarily, and decide the polarities of universal variables following this winning strategy. (This is possible because decisions are level-ordered.) The winning strategy guarantees that each such trail runs into a conflict. The set of learnable clauses either contains the empty clause, or is shown to contain an asserting clause – one which after backtracking becomes unit at some point in the trail – and an asserting clause is shown to be new. Thus each trail that does not terminate the refutation learns a new clause, and there are only finitely many clauses that can be added. All the arguments in this outline work also in the presence of a dependency scheme ($\mathtt{D}_2$) that is used in both propagation and learning. ◀

We now wish to examine how adding a particular normal dependency scheme affects the strength of these systems. In this work we focus on adding the *reflexive resolution path* dependency scheme $\mathtt{D}^{\mathtt{rrs}}$ as it is well-studied, and it is known that adding it to $\mathtt{Q\text{-}Res}$ gives a strictly stronger system in $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})$-$\mathtt{Res}$. Therefore it is interesting to see if the same parallel extends to the QCDCL systems. Thus in the system $\mathtt{D}_1 + \mathtt{QCDCL}(\mathtt{D}_2)$, we will henceforth assume that $\mathtt{D}_1, \mathtt{D}_2 \in \{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{rrs}}\}$. When a dependency scheme is $\mathtt{D}^{\mathtt{trv}}$, we will omit reference to it. Thus we have the systems $\mathtt{QCDCL}$, $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$, $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$, and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$.

Before proceeding further, the following propositions are noteworthy to keep in mind.

▶ **Proposition 3.3.** *For a QBF Φ, if $\mathtt{D}(\Phi) = \mathtt{D}^{\mathtt{trv}}(\Phi)$, then all of $\mathtt{QCDCL}$, $\mathtt{QCDCL}(\mathtt{D})$, $\mathtt{D} + \mathtt{QCDCL}$, and $\mathtt{D} + \mathtt{QCDCL}(\mathtt{D})$ are equivalent on Φ and produce the same refutations.*

This is simply because if $\mathtt{D} = \mathtt{D}^{\mathtt{trv}}$, then adding the dependency scheme gives nothing new to the system as no extra reductions are enabled.

▶ **Proposition 3.4.** *For a QBF Φ, $\mathtt{D}^{\mathtt{rrs}}(\Phi) = \emptyset$ if and only if $\mathtt{red}\text{-}\mathtt{D}^{\mathtt{rrs}}(\Phi)$ is a propositional formula (no universal variables in any clause).*

Further, if $\mathtt{D}^{\mathtt{rrs}}(\Phi) = \emptyset$, then $\mathtt{red}\text{-}\mathtt{D}^{\mathtt{rrs}}(\Phi)$ has polynomial size refutations in $\mathtt{Res}$ if and only if Φ has polynomial size refutations in $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$ and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$.

Proof (Sketch). If $\mathtt{D}^{\mathtt{rrs}}(\Phi) = \emptyset$, then by definition $\mathtt{red}\text{-}\mathtt{D}^{\mathtt{rrs}}(\Phi)$ is propositional. If $\mathtt{D}^{\mathtt{rrs}}(\Phi) \neq \emptyset$, then there is some reflexive resolution path involving a universal variable u, and the occurrence of u in the first clause of the path is blocked by an existential literal even with respect to $\mathtt{D}^{\mathtt{rrs}}$. So $\mathtt{red}\text{-}\mathtt{D}^{\mathtt{rrs}}(\Phi)$ is not propositional.

If $\mathtt{red}\text{-}\mathtt{D}^{\mathtt{rrs}}(\Phi)$ is propositional, then in $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$ and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$, after preprocessing, the universal variables have no role to play and the ensuing refutation is a standard $\mathtt{CDCL}$ refutation. Since $\mathtt{CDCL}$ is equivalent to $\mathtt{Res}$, the claim follows. ◀

▶ Remark 3.5. It is tempting to believe that if, for a QBF Φ, $\mathtt{red}\text{-}\mathtt{D}(\Phi)$ is a propositional formula easy to refute in $\mathtt{Res}$, then Φ is easy to refute in $\mathtt{QCDCL}(\mathtt{D})$ as well. However, this intuition is misleading. As we will show in Section 4.5, this is provably not the case.

4 Refutation size bounds for some formulas

In this section we examine the effect of adding the $\mathtt{D}^{\mathtt{rrs}}$ scheme to $\mathtt{QCDCL}$ (i.e. the three systems $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$, $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$ and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$) by computing refutation size bounds for some known QBF formulas, as well as for some newly-constructed QBF formulas.

4.1 The $\mathtt{QParity}_n$ formulas

The first family of formulas that we study are the $\mathtt{QParity}$ formulas, first defined in [8].

▶ **Formula 1** ($\mathtt{QParity}_n$). *The* $\mathtt{QParity}_n$ *formula has the prefix* $\exists x_1, \cdots, x_n \forall z \exists t_2, \cdots, t_n$ *and the matrix*

$$\begin{array}{lcccc} & x_1 \vee x_2 \vee \bar{t}_2 & \bar{x}_1 \vee \bar{x}_2 \vee \bar{t}_2 & x_1 \vee \bar{x}_2 \vee t_2 & \bar{x}_1 \vee x_2 \vee t_2 \\ \textit{for } i = 3, \ldots, n: & x_i \vee t_{i-1} \vee \bar{t}_i & x_i \vee \bar{t}_{i-1} \vee t_i & \bar{x}_i \vee t_{i-1} \vee t_i & \bar{x}_i \vee \bar{t}_{i-1} \vee \bar{t}_i \\ & & t_n \vee z & \bar{t}_n \vee \bar{z} & \end{array}$$

As shown in [8], these formulas are hard to refute in $\mathtt{QU\text{-}Res}$ (and hence also in $\mathtt{Q\text{-}Res}$ and $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$). In [7] it was shown that they have short refutations in $\mathtt{QCDCL}$.

It is straightforward to see that $\mathtt{D}^{\mathtt{rrs}}(\mathtt{QParity}) = \mathtt{D}^{\mathtt{trv}}(\mathtt{QParity})$: the last two clauses give the dependence (z, t_n), and this extends to (z, t_i) for all i using the remaining clauses. Hence the $\mathtt{QParity}$ formulas are hard to refute in $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})\text{-}\mathtt{Res}$ as well.

On the other hand, since these formulas are easy to refute in $\mathtt{QCDCL}$ (and hence also in $\mathtt{QCDCL}^{\mathsf{cube}}$), from Proposition 3.3 it follows that they have short refutations in all three systems: $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$, $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$, and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$.

4.2 The $\mathtt{Equality}_n$ formulas

The family of $\mathtt{Equality}$ formulas, introduced in [4], were shown to be hard for $\mathtt{QU\text{-}Res}$, and hence also for $\mathtt{Q\text{-}Res}$ and $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$. In [7] they are shown to be hard for $\mathtt{QCDCL}$ as well.

▶ **Formula 2** ($\mathtt{Equality}_n$).
The $\mathtt{Equality}_n$ *formula has the prefix* $\exists x_1 \cdots x_n \forall u_1 \cdots u_n \exists t_1 \cdots t_n$ *and the matrix*

$$\underbrace{(\bar{t}_1 \vee \cdots \vee \bar{t}_n)}_{T_n} \wedge \bigwedge_{i=1}^{n} \left[\underbrace{(x_i \vee u_i \vee t_i)}_{A_i} \wedge \underbrace{(\bar{x}_i \vee \bar{u}_i \vee t_i)}_{B_i} \right]$$

In [13], they are shown to be easy to refute in $\mathtt{QCDCL}^{\mathsf{cube}}$. In [3] it is shown that $\mathtt{D}^{\mathtt{rrs}}(\mathtt{Equality}) = \emptyset$, and that the $\mathtt{Equality}$ formulas have short refutations in $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})\text{-}\mathtt{Res}$.

$\mathtt{D}^{\mathtt{rrs}} = \emptyset$ also implies $\mathtt{red\text{-}D}^{\mathtt{rrs}}(\mathtt{Equality})$ is propositional and can be obtained from $\mathtt{Equality}$ by dropping all the universals. It is clear to see this formula has a short $\mathtt{Res}$ refutation. Therefore by Proposition 3.4 the $\mathtt{Equality}$ formulas are easy to refute in $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$ and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$.

The formulas are also easy to refute in $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$, but it is not as straightforward as with the aforementioned case; see Remark 3.5.

▶ **Lemma 4.1.** *The* $\mathtt{Equality}_n$ *formulas have* $O(n^2)$ *refutations in* $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$

Proof (Sketch). For decreasing values of i from $n-1$, construct pairs of trails $\mathcal{U}_i, \mathcal{V}_i$ deciding the variables $x_1 \ldots x_i$ all positively or all negatively, and choose the learnt clause cleverly. ◀

Thus the $\mathtt{Equality}$ formulas, which are hard for both $\mathtt{Q\text{-}Res}$ and $\mathtt{QCDCL}$, become easy to refute when the power of $\mathtt{D}^{\mathtt{rrs}}$ is added to these systems. Thus they showcase the power of dependency schemes and discarding spurious dependencies.

4.3 The $\texttt{Trapdoor}_n$ formulas

The Trapdoor formulas were introduced in [7] to compare QCDCL with $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$. The idea is to juxtapose two propositional formulas, one easy and one hard for Res, and judiciously interleave them with new variables such that QCDCL trails with NO-RED jump to refuting the easy part, while the ones with RED are trapped into refuting the hard part. Thus in QCDCL proof systems enabling more unit propagations is not necessarily an advantage.

▶ **Formula 3** ($\texttt{Trapdoor}_n$). *The* $\texttt{Trapdoor}_n$ *QBF has the prefix* $\exists y_1, \cdots, y_{s_n} \forall w \exists t \exists x_1, \cdots, x_{s_n} \forall u$, *where* s_n *is the number or variables in the propositional pigeonhole principle* $\texttt{PHP}^{n+1}_n$, *and the following matrix:*

$$\begin{array}{ll} & \texttt{PHP}^{n+1}_n(x_1, \cdots, x_{s_n}) \\ \text{for } \; i \in [s_n]: & \bar{y}_i \vee x_i \vee u \;,\; y_i \vee \bar{x}_i \vee u \\ \text{for } \; i \in [s_n]: & y_i \vee w \vee t \;,\; y_i \vee w \vee \bar{t} \;,\; \bar{y}_i \vee w \vee t \;,\; \bar{y}_i \vee w \vee \bar{t} \end{array}$$

Clearly, $\texttt{D}^{\texttt{rrs}}(\texttt{Trapdoor}) = \emptyset$ since the universal variables appear in only one polarity. Thus red-$\texttt{D}^{\texttt{rrs}}$(Trapdoor)is obtained by dropping all universals from the Trapdoor formula, and clearly have a short Res refutation using the four y, t clauses for any i. By Proposition 3.4, the Trapdoor formulas are easy to refute in $\texttt{D}^{\texttt{rrs}}$ + QCDCL and $\texttt{D}^{\texttt{rrs}}$ + QCDCL($\texttt{D}^{\texttt{rrs}}$). From [7], they are also easy for $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$ and Q-Res (and hence also Q($\texttt{D}^{\texttt{rrs}}$)-Res and QU-Res). But they are hard in QCDCL because the reductions force unit propagations in the trails which send the solver down a "trap" of refuting PHP.

We observe below that they are easy to refute in QCDCL($\texttt{D}^{\texttt{rrs}}$) as well.

▶ **Lemma 4.2.** *The* $\texttt{Trapdoor}_n$ *formulas have* $O(1)$*-size refutation in* QCDCL($\texttt{D}^{\texttt{rrs}}$)

Proof (Sketch). As seen before, $\texttt{D}^{\texttt{rrs}}(\texttt{Trapdoor}_n) = \emptyset$. Using this fact we can construct a QCDCL($\texttt{D}^{\texttt{rrs}}$) refutation consisting of two trails T_1 and T_2. The first trail decides y_1 and learns $\bar{y}_1$; the second trail has no decisions and learns the empty clause. ◀

4.4 The $\texttt{Dep-Trap}_n$ formulas

The previous sub-sections may suggest that adding $\texttt{D}^{\texttt{rrs}}$ to QCDCL creates a strictly stronger proof system (as is the case with Q-Res.) We design a formula to show that this is not the case. In [7], it is shown that neither of $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$ and QCDCL simulates the other. This motivates the designing of a formula where adding $\texttt{D}^{\texttt{rrs}}$ enables extra reductions that send the refutation down a "trap" that a seemingly weaker system jumps past. Based on this idea we introduce the Dep-Trap family, a slight modification of the Trapdoor formula.

▶ **Formula 4** ($\texttt{Dep-Trap}_n$). *The* $\texttt{Dep-Trap}_n$ *formula has the prefix* $\exists y_1, \cdots, y_{s_n} \forall w \exists t \forall u \exists x_1, \cdots, x_{s_n}$, *and the matrix is as given below.*

$$\begin{array}{ll} & \texttt{PHP}^{n+1}_n(x_1, \cdots, x_{s_n}) \\ \text{for } \; i \in [s_n]: & \bar{y}_i \vee u \vee x_i \;,\; y_i \vee u \vee \bar{x}_i \\ \text{for } \; i \in [s_n]: & y_i \vee w \vee t \;,\; y_i \vee w \vee \bar{t} \;,\; \bar{y}_i \vee w \vee t \;,\; \bar{y}_i \vee w \vee \bar{t} \\ & \bar{w} \vee \bar{t} \end{array}$$

There are two major differences from the $\texttt{Trapdoor}_n$ formulas. Firstly, u is now quantified before the x_i's; this prevents QCDCL going down the "trap". Secondly, an additional clause $\bar{w} \vee \bar{t}$ is introduced; this prevents the $\texttt{D}^{\texttt{rrs}}$-enabled system from bypassing the "trap".

For exactly the same reasons as Trapdoor, the Dep-Trap formulas are easy to refute in $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$ and QCDCL, thus also in Q-Res, Q($\texttt{D}^{\texttt{rrs}}$)-Res, QU-Res, and $\texttt{QCDCL}^{\texttt{cube}}$. However these formulas are hard to refute in all QCDCL variants that use $\texttt{D}^{\texttt{rrs}}$.

▶ **Lemma 4.3.** *The* $\texttt{Dep-Trap}$ *formulas have polynomial-size refutations in* $\texttt{QCDCL}$*,* $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$*,* $\texttt{Q-Res}$*,* $\texttt{QU-Res}$ *and* $\texttt{Q(D}^{\texttt{rrs}}\texttt{)-Res}$*.*

Proof (Sketch). For $\texttt{QCDCL}$ and $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$, there is a two-trail refutation, the first deciding all y's positively, then w negatively and learning $(\bar{y}_1)$ from the conflict. The second trail propagates $(\bar{y}_1)$, makes the same decisions to conflict, and learns the empty clause. ◀

▶ **Lemma 4.4.** *Refutations of the* $\texttt{Dep-Trap}_n$ *formulas in* $\texttt{QCDCL(D}^{\texttt{rrs}}\texttt{)}$*,* $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}$ *and* $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL(D}^{\texttt{rrs}}\texttt{)}$ *require exponential size.*

Proof (Sketch). $\texttt{D}^{\texttt{rrs}}(\texttt{Dep-Trap}_n) = \{(w,t)\}$; so w cannot be reduced from axiom clauses during propagation, but u can be reduced. Therefore every y_i decision propagates x_i in the opposite polarity, forcing the trails to refute $\texttt{PHP}$ exactly the same way as hardness of $\texttt{Trapdoor}$ in $\texttt{QCDCL}$ [7]. ◀

The $\texttt{Dep-Trap}$ formulas are thus easy for $\texttt{QCDCL}$, but become hard to refute when $\texttt{D}^{\texttt{rrs}}$ is added to the system, demonstrating that allowing more reductions and removing spurious dependencies does not necessarily help for the $\texttt{QCDCL}$ system.

4.5 The $\texttt{TwoPHPandCT}_n$ formulas

The previous sections suggest Proposition 3.4 could also hold for $\texttt{QCDCL(D}^{\texttt{rrs}}\texttt{)}$, but the following formula refutes that argument. The motivation for this construction also comes from the $\texttt{Trapdoor}$ formulas, with the new element of using two copies of the hard part.

▶ **Formula 5** ($\texttt{TwoPHPandCT}_n$)**.** *The* $\texttt{TwoPHPandCT}_n$ *formulas has the prefix* $\mathcal{Q} = \forall u \exists x_1 \cdots x_{s_n} \ \exists y_1 \cdots y_{s_n} \ \forall v \exists z_1, z_2$ *and the matrix*

$$\begin{array}{cc} u \vee \mathrm{PHP}(x_1, \cdots, x_{s_n}) & \bar{u} \vee \mathrm{PHP}(y_1, \cdots, y_{s_n}) \\ \multicolumn{2}{c}{v \vee z_1 \vee z_2 \ , \ v \vee \bar{z}_1 \vee z_2 \ , \ v \vee z_1 \vee \bar{z}_2 \ , \ v \vee \bar{z}_1 \vee \bar{z}_2} \end{array}$$

Observe that these formulas are easy to refute in $\texttt{Q-Res}$, using the four z_1, z_2 clauses, and hence also easy to refute in $\texttt{Q(D}^{\texttt{rrs}}\texttt{)-Res}$ and $\texttt{QU-Res}$.

Since v appears in only one polarity and $u, \bar{u}$ appear in clauses with disjoint variables, hence $\texttt{D}^{\texttt{rrs}}(\texttt{TwoPHPandCT}) = \emptyset$ and $\texttt{red-D}^{\texttt{rrs}}(\texttt{TwoPHPandCT})$ is the propositional formula of two copies of $\texttt{PHP}$ and four clauses of the complete tautology on two variables, and hence is easy to refute in $\texttt{Res}$. By Proposition 3.4, the original QBFs are easy to refute in $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}$ and $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL(D}^{\texttt{rrs}}\texttt{)}$.

However they are hard for all the three systems $\texttt{QCDCL}$, $\texttt{QCDCL(D}^{\texttt{rrs}}\texttt{)}$ and $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$.

▶ **Lemma 4.5.** *The QBF formulas* $\texttt{TwoPHPandCT}_n$ *require exponential size refutations in* $\texttt{QCDCL}$*,* $\texttt{QCDCL(D}^{\texttt{rrs}}\texttt{)}$*and* $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$*.*

Proof (Sketch). All three systems have no preprocessing and must first decide u, which causes no propagations, and then are reduced to refuting $\texttt{PHP}$ (in either x or y), which requires exponential size refutations. ◀

These formulas highlight two important facts: firstly that $\texttt{QCDCL(D}^{\texttt{rrs}}\texttt{)}$ is not the same as $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}$ or $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL(D}^{\texttt{rrs}}\texttt{)}$ and does not simulate them either. And secondly, even in the case when $\texttt{D}^{\texttt{rrs}} = \emptyset$ and reducing the formula by $\texttt{D}^{\texttt{rrs}}$ gives us an easy propositional formula, it can still be hard to refute for $\texttt{QCDCL(D}^{\texttt{rrs}}\texttt{)}$.

4.6 The $\texttt{RRSTrapEq}_n$ formulas

The next family of formulas are obtained by making a slight modification to the `Equality` formulas. The motivation to define such formulas comes from trying to ascertain whether after preprocessing with $\mathtt{D}^{\mathtt{rrs}}$, does allowing reductions using $\mathtt{D}^{\mathtt{rrs}}$ for unit propagation add any power over trivial universal reductions.

▶ **Formula 6** ($\texttt{RRSTrapEq}_n$)**.** *The* $\texttt{RRSTrapEq}_n$ *formula has the prefix* $\exists a \exists x_1 \cdots x_n \forall u_1 \cdots u_n \exists t_1 \cdots t_n \exists b$ *and the PCNF matrix as below:*

$$\underbrace{(\bar{t}_1 \vee \cdots \vee \bar{t}_n)}_{T_n} \wedge \bigwedge_{i=1}^{n} \left[\underbrace{(x_i \vee u_i \vee t_i \vee b)}_{A_i} \wedge \underbrace{(\bar{x}_i \vee \bar{u}_i \vee t_i \vee b)}_{B_i} \right] \wedge \bigwedge_{i=1}^{n} \underbrace{(u_i \vee \bar{b})}_{C_i} \wedge (a \vee \bar{b}) \wedge (\bar{a} \vee \bar{b})$$

The underlying idea in the formulation is that unlike $\mathtt{D}^{\mathtt{rrs}}$(`Equality`) which is empty, we add the C_i clauses to make $\mathtt{D}^{\mathtt{rrs}}$(`RRSTrapEq`) $= \{(u_i, b) : i \in [n]\}$, and add b to the x, u, t clauses to obtain red-$\mathtt{D}^{\mathtt{rrs}}$(`RRSTrapEq`) = `RRSTrapEq`. Consequently, preprocessing by $\mathtt{D}^{\mathtt{rrs}}$ does not change the formula at all. Therefore the refutation for these formulas will be exactly the same for `QCDCL` and $\mathtt{D}^{\mathtt{rrs}}$ + `QCDCL`, and similarly for `QCDCL`($\mathtt{D}^{\mathtt{rrs}}$) and $\mathtt{D}^{\mathtt{rrs}}$ + `QCDCL`($\mathtt{D}^{\mathtt{rrs}}$).

▶ **Lemma 4.6.** *The* `RRSTrapEq` *formulas have polynomial size refutations in* `QCDCL`($\mathtt{D}^{\mathtt{rrs}}$) *and* $\mathtt{D}^{\mathtt{rrs}}$ + `QCDCL`($\mathtt{D}^{\mathtt{rrs}}$)*, but require exponential size refutations in* `QCDCL` *and* $\mathtt{D}^{\mathtt{rrs}}$ + `QCDCL`

Proof (Sketch). Since, red-$\mathtt{D}^{\mathtt{rrs}}$(`RRSTrapEq`)= `RRSTrapEq`, preprocessing has no effect. After the first decision, hardness reduces to hardness of the `Equality` formulas (Section 4.2) in the `QCDCL` or `QCDCL`($\mathtt{D}^{\mathtt{rrs}}$)systems. ◀

Additionally, since the `Equality` formulas are embedded in the `RRSTrapEq` formula, and since `QU-Res` is closed under restrictions, the `RRSTrapEq` formulas are hard for `QU-Res` and in turn `Q-Res` and $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$. However, they have easily seen short refutations in `Q`($\mathtt{D}^{\mathtt{rrs}}$)`-Res`.

4.7 The $\texttt{PreDepTrap}_n$ formulas

The previous section underlines that preprocessing using $\mathtt{D}^{\mathtt{rrs}}$ does not make $\mathtt{D}^{\mathtt{rrs}}$ during propagation obsolete. However, one might expect that at the very least preprocessing won't make things worse. However that is not so; the following example highlights that fact.

▶ **Formula 7** ($\texttt{PreDepTrap}_n$)**.** *The* $\texttt{PreDepTrap}_n$ *formula has the prefix* $\forall a \ \exists y_1 \cdots y_{s_n} \forall w \exists t \forall u$ $\exists x_1 \cdots x_{s_n} \ \exists p_1 \cdots p_n \forall q_1 \cdots q_n \exists r_1 \cdots r_n$*, and the matrix*

$$a \vee \texttt{Dep-Trap}(y_1, \cdots, y_{s_n}, w, t, u, x_1, \cdots, x_{s_n})$$
$$\bar{a} \vee \texttt{Equality}(p_1, \cdots, p_n, q_1, \cdots, q_n, r_1, \cdots, r_n)$$

Looking at $\mathtt{D}^{\mathtt{rrs}}$ for the formula, the literals a and $\bar{a}$ appear in clauses with disjoint non-interacting sets of variables, and $\mathtt{D}^{\mathtt{rrs}}$(`Equality`) $= \emptyset$. So $\mathtt{D}^{\mathtt{rrs}}$(`PreDepTrap`) = $\mathtt{D}^{\mathtt{rrs}}$(`Dep-Trap`)

Consider the systems $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$, `QCDCL` and `QCDCL`($\mathtt{D}^{\mathtt{rrs}}$). Every trail must start with setting the variable a. In the latter system, let the trails start with decision a, which reduces the formula to exactly the `Equality` formulas, which are easy to refute in `QCDCL`($\mathtt{D}^{\mathtt{rrs}}$), (Lemma 4.1). Thus the same refutation as Lemma 4.1 with a leading decision a in every trail gives an easy `QCDCL`($\mathtt{D}^{\mathtt{rrs}}$) refutation. In the two former systems, let the trails decide $\bar{a}$. Then the formula reduces exactly to `Dep-Trap` formula which is easy to refute in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$ and `QCDCL` (Section 4.4) and as a consequence in `Q-Res`, `Q`($\mathtt{D}^{\mathtt{rrs}}$)`-Res`, `QU-Res`, and $\mathtt{QCDCL}^{\mathtt{cube}}$.

Finally, we show that preprocessing the formulas makes refutations exponentially long.

▶ **Lemma 4.7.** *The* PreDepTrap *formulas require exponential size refutations in* $\mathtt{D}^{\mathtt{rrs}}+\mathtt{QCDCL}$ *and* $\mathtt{D}^{\mathtt{rrs}}+\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$

Proof (Sketch). As in the case of Trapdoor, QCDCL trails on red-$\mathtt{D}^{\mathtt{rrs}}$(PreDepTrap) are forced to refute the PHP part. ◀

4.8 The PropDep-Trap$_n$ formulas

Section 4.7 illustrates that $\mathtt{D}^{\mathtt{rrs}}$ as a preprocessing technique was detrimental; it was better to use $\mathtt{D}^{\mathtt{rrs}}$ *only* in unit propagation. This begs the question – could there be a formula such that $\mathtt{D}^{\mathtt{rrs}}$ only as preprocessing is strictly better than $\mathtt{D}^{\mathtt{rrs}}$ for just propagation or both preprocessing and propagation? The answer is yes, and is witnessed by the following slight modification of the Dep-Trap formulas

▶ **Formula 8** (PropDep-Trap$_n$). *The* PropDep-Trap$_n$ *formulas have the prefix* $\exists s\ \ \exists y_1 \cdots y_{s_n} \forall w \exists t \forall b_1, b_2\ \exists x_1 \cdots x_{s_n}\ \exists z_1, z_2$ *and the matrix as given below.*

$$\begin{array}{rl} & \mathtt{PHP}_n^{n+1}(x_1, \cdots, x_{s_n}) \\ \textit{for}\ \ i \in [s_n]: & \bar{y}_i \vee b_1 \vee x_i \vee z_1\ ,\ y_i \vee b_2 \vee \bar{x}_i \vee z_2 \\ & s \vee w \vee t\ ,\ s \vee w \vee \bar{t}\ ,\ \bar{s} \vee w \vee t\ ,\ \bar{s} \vee w \vee \bar{t}\ ,\quad \bar{w} \vee \bar{t}\ , \\ & \bar{b}_1 \vee \bar{z}_1\ ,\ \bar{b}_2 \vee \bar{z}_2\ ,\ \bar{z}_1\ ,\ \bar{z}_2 \end{array}$$

Notice that $\mathtt{D}^{\mathtt{rrs}}(\mathtt{PropDep\text{-}Trap}) = \{(w,t),(b_1,z_1),(b_2,z_2)\}$. Therefore preprocessing has no effect on the formula; red-$\mathtt{D}^{\mathtt{rrs}}$(PropDep-Trap)= PropDep-Trap.

The presence of the four s, w, t clauses make this formula very easy to refute in Q-Res and hence also in Q($\mathtt{D}^{\mathtt{rrs}}$)-Res and QU-Res. They are also easy to refute in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$, QCDCL, $\mathtt{D}^{\mathtt{rrs}}+\mathtt{QCDCL}$ and $\mathtt{QCDCL}^{\mathsf{cube}}$ because, apart from the initial propagation of $\bar{z}_1, \bar{z}_2$, the s,y_i decisions cause no propagation, then we can decide $\bar{w}$ and reach a conflict using the s, w, t clauses. If the trail decided $s(\bar{s})$, from this conflict we can learn $\bar{s}(s)$; the next trail propagtes $\bar{s}(s)$, and after deciding $\bar{w}$ in the same manner the empty clause can be learnt.

These formulas are however hard to refute in $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$ and $\mathtt{D}^{\mathtt{rrs}}+\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$.

▶ **Lemma 4.8.** *The* PropDep-Trap *formulas require exponential size refutations in* $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$ *and* $\mathtt{D}^{\mathtt{rrs}}+\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$

Proof (Sketch). red-$\mathtt{D}^{\mathtt{rrs}}$(PropDep-Trap)= PropDep-Trap; therefore preprocessing has no effect. $\mathtt{D}^{\mathtt{rrs}}$ during propagation causes propagation of x_i due to y_i forcing the refutation down the same "trap" as the Trapdoor formulas for QCDCL[7]. ◀

4.9 The TwinEq$_n$ formulas

The TwinEq formulas were introduced in [13] to show hardness in the system $\mathtt{QCDCL}^{\mathsf{cube}}$. They are formally defined as follows:

▶ **Formula 9** (TwinEq$_n$ [13]). *The* TwinEq$_n$ *formula has the prefix* $\exists x_1 \cdots x_n \forall u_1 \cdots u_n, w_1, \cdots, w_n \exists t_1 \cdots t_n$ *and the PCNF matrix*

$$\underbrace{(\bar{t}_1 \vee \cdots \vee \bar{t}_n)}_{T_n} \wedge \bigwedge_{i=1}^{n} \left[\underbrace{(x_i \vee u_i \vee t_i)}_{A_i} \wedge \underbrace{(\bar{x}_i \vee \bar{u}_i \vee t_i)}_{B_i} \right] \wedge \bigwedge_{i=1}^{n} \left[\underbrace{(x_i \vee w_i \vee t_i)}_{C_i} \wedge \underbrace{(\bar{x}_i \vee \bar{w}_i \vee t_i)}_{D_i} \right]$$

These formulas are hard for $\texttt{QCDCL}^{\texttt{cube}}$, and hence also for $\texttt{QCDCL}$ and $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$. For the same reason as for $\texttt{Equality}$ (the size-cost-capacity theorem from [4]), they are also hard for $\texttt{QU-Res}$ and $\texttt{Q-Res}$.

It is easy to show that $\texttt{D}^{\texttt{rrs}}(\texttt{TwinEq}) = \emptyset$ (just as $\texttt{D}^{\texttt{rrs}}(\texttt{Equality})$ is shown to be $\emptyset$). Therefore, $\texttt{red-D}^{\texttt{rrs}}(\texttt{TwinEq})$ is a propositional formula; in fact is the same formula as $\texttt{red-D}^{\texttt{rrs}}(\texttt{Equality})$, which has a short Resolution proof. Hence, $\texttt{TwinEq}$ is easy to refute in $\texttt{Q}(\texttt{D}^{\texttt{rrs}})\texttt{-Res}$, and from Proposition 3.4, is also easy to refute in $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}$, $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$. They are also easy to refute in $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$, following the same argument as for the $\texttt{Equality}$ formulas.

5 Relations between proof systems

Putting together the bounds from the previous section, we can now place the newly-defined proof systems within the simulation order. First we observe that the four versions of $\texttt{QCDCL}$ that use or do not use $\texttt{D}^{\texttt{rrs}}$ in either of the two ways are all pairwise incomparable. Next we observe that each of the three new versions of $\texttt{QCDCL}$ is also incomparable with $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$, $\texttt{Q-Res}$, $\texttt{Q}(\texttt{D}^{\texttt{rrs}})\texttt{-Res}$, and $\texttt{QU-Res}$. Finally, we observe that even when we add cube-learning to standard $\texttt{QCDCL}$, the system $\texttt{QCDCL}^{\texttt{cube}}$ is still incomparable with all the three versions of $\texttt{QCDCL}$ with dependency scheme added.

▶ **Theorem 5.1.** *The proof systems in* $\{\texttt{QCDCL}, \texttt{QCDCL}(\texttt{D}^{\texttt{rrs}}), \texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}, \texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})\}$ *are pairwise incomparable.*

Proof. Of the four systems under consideration, the $\texttt{Trapdoor}$ formulas (Section 4.3) are hard only for $\texttt{QCDCL}$, and the $\texttt{Dep-Trap}$ formulas (Section 4.4)are easy only in $\texttt{QCDCL}$. Hence $\texttt{QCDCL}$ is incomparable with all three systems obtained by adding $\texttt{D}^{\texttt{rrs}}$.

Among the three systems using $\texttt{D}^{\texttt{rrs}}$, the $\texttt{TwoPHPandCT}$ formulas (Section 4.5) are hard only in $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$, while the $\texttt{PreDepTrap}$ formulas (Section 4.7) are easy only in $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$. Hence $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$ is incomparable with the systems that use preprocessing.

Finally, the systems $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}$ and $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$ are separated by the formulas $\texttt{RRSTrapEq}$ (Section 4.6) easy only in $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$, and the formulas $\texttt{PropDep-Trap}$ (Section 4.8) easy only in $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}$. ◀

▶ **Theorem 5.2.** *Any two proof systems* $\texttt{P}_1 \in \{\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}}), \texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}, \texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})\}$, *and* $\texttt{P}_2 \in \{\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}, \texttt{Q-Res}, \texttt{Q}(\texttt{D}^{\texttt{rrs}})\texttt{-Res}, \texttt{QU-Res}\}$, *are incomparable.*

Proof. The $\texttt{QParity}$ formulas (Section 4.1) require exponential size refutations in $\texttt{P}_2$ but have polynomial size refutations in $\texttt{P}_1$.

The $\texttt{Dep-Trap}$ formulas (Section 4.4) have constant size refutations in $\texttt{P}_2$ but require exponential size refutations in $\texttt{P}_1$. ◀

▶ **Theorem 5.3.** *Every proof system in* $\{\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}}), \texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}, \texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})\}$ *is incomparable with* $\texttt{QCDCL}^{\texttt{cube}}$*.*

Proof. The $\texttt{TwinEq}$ formulas (Section 4.9) require exponential size refutations in $\texttt{QCDCL}^{\texttt{cube}}$ but have poly-size refutations in $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}$, $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$ and $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$.

The $\texttt{Dep-Trap}$ formulas (Section 4.4) require exponential size refutations in $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}$, $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$ and $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$, but have short refutations in $\texttt{QCDCL}^{\texttt{cube}}$. ◀

6 Conclusion

We have examined, from a rigourous proof-theoretic viewpoint, the effect of incorporating heuristics based on dependency schemes into QBF solving algorithms based on the conflict-driven clause learning paradigm. Our results show that unlike in the case of the proof system `Q-Res`, where dependency-awareness can shorten refutations but never lengthens them, here the picture is much more nuanced, and all kinds of shortenings as well as lengthenings can be observed. Thus the decision of whether or not to make a `QCDCL` solver account for spurious dependencies is itself a challenging one, and it is likely the domain from where instances are to be solved may indicate what choice is more suitable.

Some directions for further studies include

1. the impact of other dependency schemes; e.g. the less (than $\mathtt{D}^{\mathtt{rrs}}$) general standard dependency scheme $\mathtt{D}^{\mathtt{std}}$ introduced in [21], and more general schemes based on tautology-free and implication-free paths introduced in [5, 6]. It seems reasonable to expect that a similar nuanced picture will present when considering such schemes as well.
2. the impact of specific learning schemes on dependency-aware `QCDCL`. Our lower bounds hold for any `QCDCL`-based solver as long as the learning scheme picks a clause only from the learnable-clause sequence as defined in Section 2. However, the upper bounds hold for specific choices of learnt clauses, and this, in some sense, reflects a certain non-determinism in the algorithm. (This is somewhat akin to the non-determinism inherent in `CDCL` algorithms in the statement that `CDCL` simulates Resolution.) Arguably, the upper bounds will be more meaningful if achieved with actually-used learning schemes. While some of our upper bounds are achieved with such schemes, specifically the UIP policy, some others make ad hoc choices with respect to which clauses to learn.
3. formalising the dependency-learning approach (used in the solver Qute [18]). Here the solver adds detected dependencies starting from $\emptyset$, rather than removing spurious dependencies starting from $\mathtt{D}^{\mathtt{trv}}$. This would change the decision order from `LEV-ORD`. Recent work in [11] shows that other orders are not necessarily unsound, but we are quite short on techniques to analyse any-decision-order.
4. formalising the combination of cube-learning and dependency schemes for all true and false QBFs. This may be somewhat non-trivial and nuanced, as adding dependency schemes to the formal proof system of long-distance term resolution (the proof system in which proofs can be extracted from runs of solvers on true QBFs, Q-consensus) is not known to be sound (see the Discussion section in [19]).

References

1 Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. Clause-learning algorithms with many restarts and bounded-width resolution. *J. Artif. Intell. Res.*, 40:353–373, 2011. doi:10.1613/jair.3152.

2 Salman Azhar, Gary Peterson, and John Reif. Lower bounds for multiplayer non-cooperative games of incomplete information. *Journal of Computers and Mathematics with Applications*, 41:957–992, 2001.

3 Olaf Beyersdorff and Joshua Blinkhorn. Dynamic QBF dependencies in reduction and expansion. *ACM Trans. Comput. Log.*, 21(2):8:1–8:27, 2020. doi:10.1145/3355995.

4 Olaf Beyersdorff, Joshua Blinkhorn, and Luke Hinde. Size, cost, and capacity: A semantic technique for hard random QBFs. *Log. Methods Comput. Sci.*, 15(1), 2019. doi:10.23638/LMCS-15(1:13)2019.

5 Olaf Beyersdorff, Joshua Blinkhorn, and Tomás Peitl. Strong (D)QBF dependency schemes via tautology-free resolution paths. In Luca Pulina and Martina Seidl, editors, *Theory and Applications of Satisfiability Testing – SAT 2020 – 23rd International Conference, Alghero, Italy, July 3-10, 2020, Proceedings*, volume 12178 of *Lecture Notes in Computer Science*, pages 394–411. Springer, 2020. doi:10.1007/978-3-030-51825-7_28.

6 Olaf Beyersdorff, Joshua Blinkhorn, and Tomás Peitl. Strong (D)QBF dependency schemes via implication-free resolution paths. *Electron. Colloquium Comput. Complex.*, TR21-135, 2021. arXiv:TR21-135.

7 Olaf Beyersdorff and Benjamin Böhm. Understanding the Relative Strength of QBF CDCL Solvers and QBF Resolution. *Logical Methods in Computer Science*, Volume 19, Issue 2, April 2023. (Preliminary version in ITCS'21). doi:10.46298/lmcs-19(2:2)2023.

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

9 Olaf Beyersdorff, Mikolás Janota, Florian Lonsing, and Martina Seidl. Quantified boolean formulas. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability – Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 1177–1221. IOS Press, 2021. doi:10.3233/FAIA201015.

10 Joshua Blinkhorn and Olaf Beyersdorff. Shortening QBF proofs with dependency schemes. In *Theory and Applications of Satisfiability Testing – SAT*, volume 10491 of *LNCS*, pages 263–280. Springer, 2017. doi:10.1007/978-3-319-66263-3_17.

11 Benjamin Böhm, Tomás Peitl, and Olaf Beyersdorff. Should decisions in QCDCL follow prefix order? In *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, August 2-5, 2022, Haifa, Israel*, volume 236 of *LIPIcs*, pages 11:1–11:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SAT.2022.11.

12 Benjamin Böhm and Olaf Beyersdorff. QCDCL vs QBF resolution: Further insights. In *26th Theory and Applications of Satisfiability Testing – SAT*, 2023. full version in ECCC, TR23-051.

13 Benjamin Böhm, Tomáš Peitl, and Olaf Beyersdorff. QCDCL with cube learning or pure literal elimination – What is best? In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 1781–1787. International Joint Conferences on Artificial Intelligence Organization, July 2022. Main Track. doi:10.24963/ijcai.2022/248.

14 Abhimanyu Choudhury and Meena Mahajan. Dependency schemes in CDCL-based QBF solving: a proof-theoretic study. *Electron. Colloquium Comput. Complex.*, TR23-061, 2023. arXiv:TR23-061.

15 Florian Lonsing. *Dependency schemes and search-based QBF solving: theory and practice*. PhD thesis, Johannes Kepler University, Linz, Austria, 2012.

16 Florian Lonsing and Armin Biere. Depqbf: A dependency-aware QBF solver. *J. Satisf. Boolean Model. Comput.*, 7(2-3):71–76, 2010. doi:10.3233/sat190077.

17 Florian Lonsing and Armin Biere. Integrating dependency schemes in search-based QBF solvers. In Ofer Strichman and Stefan Szeider, editors, *Theory and Applications of Satisfiability Testing – SAT 2010, 13th International Conference, SAT 2010, Edinburgh, UK, July 11-14, 2010. Proceedings*, volume 6175 of *Lecture Notes in Computer Science*, pages 158–171. Springer, 2010. doi:10.1007/978-3-642-14186-7_14.

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

19 Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Long-distance Q-resolution with dependency schemes. *J. Autom. Reasoning*, 63(1):127–155, 2019. doi:10.1007/s10817-018-9467-3.

20 Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artif. Intell.*, 175(2):512–525, 2011. doi:10.1016/j.artint.2010.10.002.

21 Marko Samer and Stefan Szeider. Backdoor sets of quantified boolean formulas. *J. Autom. Reason.*, 42(1):77–97, 2009. doi:10.1007/s10817-008-9114-5.

22 Christoph Scholl and Ralf Wimmer. Dependency quantified Boolean formulas: An overview of solution methods and applications – extended abstract. In Olaf Beyersdorff and Christoph M. Wintersteiger, editors, *International Conference on Theory and Practice of Satisfiability Testing SAT*, volume 10929 of *LNCS*, pages 3–16. Springer, 2018.

23 Ankit Shukla, Armin Biere, Luca Pulina, and Martina Seidl. A survey on applications of quantified boolean formulas. In *31st IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2019, Portland, OR, USA, November 4-6, 2019*, pages 78–84. IEEE, 2019. doi:10.1109/ICTAI.2019.00020.

24 Friedrich Slivovsky and Stefan Szeider. Soundness of Q-resolution with dependency schemes. *Theor. Comput. Sci.*, 612:83–101, 2016. doi:10.1016/j.tcs.2015.10.020.