Electronic Colloquium on Computational Complexity, Report No. 61 (2023)

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

We formalize the notion of proof systems obtained by adding normal dependency schemes into the QCDCL proof system underlying algorithms for solving Quantified Boolean Formulas, by exploring the addition of the dependency schemes via two approaches: one as a preprocessing tool, and second in propagation and learnings in the QCDCL trails.

We show that QCDCL augmented with the reflexive resolution path dependency scheme $\mathtt{D}^{\mathtt{rrs}}$ produces three proof systems of interest: $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$, $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$ and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$. We show that these three systems are not only pairwise incomparable, but also each system is incomparable with the standard QCDCL and $\mathtt{QCDCL}^{\mathtt{cube}}$, as well as with $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$, Q-Res, QU-Res, and $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})$-Res .

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases QBF, CDCL, Resolution, Dependency schemes

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 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, [19, 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 [22, 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

ISSN 1433-8092

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 $\mathtt{QCDCL}_{\mathtt{RED}}^{\mathtt{LEV\text{-}ORD}}$, which we will refer to as QCDCL without any sub/super-script; for the other systems we will explicitly write the policies.

While the aforementioned 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. 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 engredient. 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.

DepQBF [15] 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 quite 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 Herbrandt/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, 21]), these heuristics are not aiming to solve DQBFs in general. Rather, they algorithmically detect spurious dependencies and disregard them as the algorithm proceeds. See [14, 20, 15, 16] for early work on this topic. Often the use of a dependency scheme makes the solvers run faster, and this is borne out by theoretical studies. 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, 23, 18]. 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 [18], 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 $\mathtt{QCDCL}_{\mathtt{RED}}^{\mathtt{LEV\text{-}ORD}}$ 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 [23, 10, 18]. 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 $\mathtt{D}$ 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 $\mathtt{QCDCL}$.

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 $\mathtt{D}_1, \mathtt{D}_2 \in \{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{rrs}}\}$, we study the four systems $\mathtt{D}_1 + \mathtt{QCDCL}(\mathtt{D}_2)$. As observed above, one of them is $\mathtt{QCDCL}$ 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 $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$, $\mathtt{Q\text{-}Res}$, $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})\text{-}\mathtt{Res}$, and $\mathtt{QU\text{-}Res}$(Theorem 5.2), as well as with $\mathtt{QCDCL}^{\mathtt{cube}}$(Theorem 5.3).

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 $\mathtt{QCDCL}(\mathtt{D})$ 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 $\mathtt{QCDCL}(\mathtt{D})$ 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 $\mathtt{QCDCL}(\mathtt{D})$, 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 $\mathtt{QCDCL}(\mathtt{D})$, $\mathtt{D} + \mathtt{QCDCL}$, and $\mathtt{D} + \mathtt{QCDCL}(\mathtt{D})$ are all incomparable with $\mathtt{QCDCL}^{\mathtt{cube}}$ is note-worthy and interesting as allowing for cube-learning always adds strength and makes things easier as compared to without cube-learning; $\mathtt{QCDCL}^{\mathtt{cube}}$ 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 [23, 10, 18, 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 prelimiaries 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 $\mathtt{D}_1 + \mathtt{QCDCL}(\mathtt{D}_2)$ 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 QCDCL,$\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$,Q-Res,$\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})$-Res,QU-Res,$\mathtt{QCDCL}^{\mathtt{cube}}$. We end with some concluding remarks in Section 6.

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 note, 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 $\Box$ 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 $C \vee D = \mathtt{res}(A, B, x)$), 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 stronger system LDQ-Res allows 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^*, provided it is to the right of the pivot x.

A proof system P is said to simulate 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.

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

Dependency schemes are mappings that associate every PCNF formula Φ with a binary relation on its variables in a manner that encodes constraints on the order of pairs of variables. The most basic of dependency schemes is the *trivial dependency scheme* $\mathtt{D}^{\mathtt{trv}}$, which is in fact the order of quantifier prefix: an existential variable x depends on a universal variable u (i.e. $(u, x) \in \mathtt{D}^{\mathtt{trv}}(\Phi)$) if x appears to the right of u in the quantifier prefix. A non-trivial dependency scheme D produces, for any formula Φ, a subset $\mathtt{D}(\Phi)$ of the trivial dependencies; it does not introduce new dependencies. Some non-trivial schemes are the standard scheme $\mathtt{D}^{\mathtt{std}}$ and the reflexive resolution path scheme $\mathtt{D}^{\mathtt{rrs}}$; see [23]. Roughly speaking, in $\mathtt{D}^{\mathtt{rrs}}$, (u, x) is in the dependency relation of a formula if $(u, x) \in \mathtt{D}^{\mathtt{trv}}$ 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, [23]). *For a QBF $\Phi = \mathcal{Q}\phi$, the pair (u, x) is in $\mathtt{D}^{\mathtt{rrs}}(\Phi)$ if and only if $(u, x) \in \mathtt{D}^{\mathtt{trv}}(\Phi)$ 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 $\mathtt{D}$, 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 $\mathtt{red\text{-}D}(C)$ the subclause obtained by removing all universal literals which are not blocked by any other literal in C. We denote by $\mathtt{red\text{-}D}(\Phi)$ the QBF Ψ obtained by replacing each clause C in the matrix of Φ with the clause $\mathtt{red\text{-}D}(C)$. When $\mathtt{D} = \mathtt{D}^{\mathtt{trv}}$, we use the notation $\mathtt{red}(C)$ and $\mathtt{red}(\Phi)$.

The proof systems `Q-Res` and `LDQ-Res`, augmented with a dependency scheme `D` [18], 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 $\mathtt{red\text{-}D}(C)$ from C.

An interesting and important subclass of dependency schemes are the so-called normal dependency schemes.

▶ **Definition 2.2** (Normal Dependency Scheme, [18]). *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 $\Phi = \forall X \mathcal{Q}.\phi$, every* `LDQ(D)-Res` *derivation P from Φ, and every $u \in X$, either u or $\bar{u}$ does not appear in P.*
- *normal if it is both monotone and simple.*

If `D` is simple, then 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`. However, for `LDQ-Res(D)`, this alone is not enough. The additional property of monotonicity, expressing that applying a partial assignment can possibly erase existing dependencies but cannot create new ones, suffices to ensure soundness.

The dependency schemes $\mathtt{D}^{\mathtt{trv}}$, $\mathtt{D}^{\mathtt{std}}$, $\mathtt{D}^{\mathtt{rrs}}$ are all *normal* dependency schemes. These *normal* dependency schemes are important to us because for these dependency schemes `LDQ(D)-Res` is a sound proof system[18].

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 decisions made by the algorithm and the others are propagated literals. Following the standard convention, we denote decision literals in a trail in boldface.) 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 $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{RED}}$, 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 [15, 17], 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.
   In the notation of [7], for a decision policy P and a propagation policy R, the corresponding QCDCL proof system is denoted $\mathtt{QCDCL}^{P}_{R}$. Thus standard QCDCL is $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{RED}}$. Other variants are also defined in [7]; in particular $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$.
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. ($\mathtt{ante}(\ell)$ *denotes the clause that causes literal ℓ to be propagated; i.e. the antecedent.)*

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

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

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

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

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 $\mathtt{QCDCL}^{\mathtt{cube}}$, which was shown to be strictly stronger than the standard QCDCL system i.e. $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{RED}}$.

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 $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{RED}}$.

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

For the propagation policy, the notion of unit clauses depends on the universal reductions allowed, and this in turn is affected by the dependency scheme. In the case of $R =$ NO-RED, no universal reductions are allowed anyway, so adding a dependency scheme to the proof system does not affect the policy. In the case of $R =$ RED, the definition of a unit propagation changes. A clause C propagates a literal ℓ at a position in the trail if the $\forall$(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 D leaves the single literal ℓ; $\texttt{red-D}(C|_\alpha) = \{\ell\}$. We denote this propagation policy as RED + 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 red everywhere with red-D, the $\forall$(D) rule for universal reduction with respect to the dependency.

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

These two ways of adding dependency schemes to QCDCL – (1) in the trail construction, propagation and learning itself, or (2) as pre-processing – can both be combined. For a particular dependency scheme D, we can think of three distinct proof systems:
- QCDCL(D): use D for unit propagations and learning, but not for preprocessing.
- D + QCDCL: use D only to preprocess the formula.
- D + QCDCL(D): use D for preprocessing first and then use it again 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 D′ + QCDCL(D):

▶ **Definition 3.1** (D′ + QCDCL(D) proof system)**.**
For a false QBF $\Psi = \mathcal{Q}\cdot\psi$ and a dependency scheme D*, a* QCDCL(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 LEV-ORD *and* RED + D*, each clause $C_j \in L_{T_j}$ is a clause learnable from T_j using the* 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* LDQ(D)-Res*.*

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

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

Note that QCDCL is exactly the proof system $\texttt{D}^{\texttt{trv}}$ + 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 completeness.

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

Proof. First we prove the soundness of the system. Suppose ι is a $\mathtt{D}_1 + \mathtt{QCDCL}(\mathtt{D}_2)$ refutation of a QBF Φ. By definition, this is a $\mathtt{QCDCL}(\mathtt{D}_2)$ refutation of the QBF $\Psi = \mathtt{red}\text{-}\mathtt{D}_1(\Phi)$. Now, every $\mathtt{QCDCL}(\mathtt{D})$ refutation has an underlying $\mathtt{LDQ}(\mathtt{D})\text{-}\mathtt{Res}$ refutation. Therefore, from ι we can extract a $\mathtt{LDQ}(\mathtt{D}_2)\text{-}\mathtt{Res}$ refutation Π of Ψ. Since $\mathtt{D}_2$ is a normal dependency scheme, $\mathtt{LDQ}(\mathtt{D}_2)\text{-}\mathtt{Res}$ is a sound proof system [18], and therefore Ψ is a false QBF. Now, by completeness of LDQ-Res, there exists a LDQ-Res refutation Π' of Ψ. The reductions made to obtain Ψ from Φ, followed by the derivation steps in Π', gives a $\mathtt{LDQ}(\mathtt{D}_1)\text{-}\mathtt{Res}$ refutation Π'' of Φ. Since $\mathtt{D}_1$ is also a normal dependency scheme, $\mathtt{LDQ}(\mathtt{D}_1)\text{-}\mathtt{Res}$ is also sound, and hence, the existence of Π'' implies that Φ is a false QBF.

Now we turn to completeness. In Theorem 3.16 of the full version of [7], 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. ◀

Having established soundness and completeness when normal dependency schemes are added, we now wish to look at how adding a particular 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 one of the most popular ones, and it is known that adding it to Q-Res gives a strictly stronger system in $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})\text{-}\mathtt{Res}$. 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 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 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 $\mathtt{red}\text{-}\mathtt{D}^{\mathtt{rrs}}(\Phi)$ is easy to refute in Res if and only if Φ is easy to refute 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 $\mathtt{red\text{-}D}^{\mathtt{rrs}}(\Phi)$ is not propositional.

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

▶ Remark 3.5. It is tempting to believe that if, for a QBF Φ, $\mathtt{red\text{-}D}(\Phi)$ is a propositional formula easy to refute in Res, then Φ is easy to refute in QCDCL(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 QCDCL (obtaining 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 bounds on refutation size 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 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*

$$\begin{array}{lcccc} & x_1 \vee x_2 \vee \bar{t}_2 & \bar{x}_1 \vee \bar{x}_2 \vee \bar{t}_2 & x_1 \vee \bar{x}_2 \vee t_2 & \bar{x}_1 \vee x_2 \vee t_2 \\ \textit{for } i = 2, \ldots, n: & x_i \vee t_{i-1} \vee \bar{t}_i & x_1 \vee \bar{t}_{i-1} \vee t_i & \bar{x}_i \vee t_{i-1} \vee t_i & \bar{x}_i \vee \bar{t}_{i-1} \vee \bar{t}_i \\ & & t_n \vee z & \bar{t}_n \vee \bar{z} & \end{array}$$

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

On the other hand, since these formulas are easy to refute in QCDCL (and hence also in $\mathtt{QCDCL}^{\mathtt{cube}}$), 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 next formula we study are another well-known family of QBF formulas, the Equality formulas, first introduced in [4].

▶ **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 PCNF matrix*

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

In [4] it was shown that these formulas are hard for QU-Res, and hence also for Q-Res and $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$. In [7] it was shown that they are hard for QCDCL as well.

However, as shown in [13], they are easy to refute in $\mathtt{QCDCL}^{\mathtt{cube}}$.

Further, as shown in [3], $\mathtt{D}^{\mathtt{rrs}}(\mathtt{Equality}) = \emptyset$, and there are short refutations in $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})$-Res.

Since $\mathtt{D}^{\mathtt{rrs}}(\mathtt{Equality}) = \emptyset$, no existential variable depends on any universal variable in the entire formula. Hence red-$\mathtt{D}^{\mathtt{rrs}}(\mathtt{Equality})$ is the propostional formula described below.

$$\text{red-}\mathtt{D}^{\mathtt{rrs}}(\mathtt{Equality}) : \quad \underbrace{(\bar{t}_1 \vee \cdots \vee \bar{t}_n)}_{T_n} \wedge \bigwedge_{i=1}^{n} \left[\underbrace{(x_i \vee t_i)}_{A'_i} \wedge \underbrace{(\bar{x}_i \vee t_i)}_{B'_i} \right]$$

This formula has a short Res refutation (resolve A'_i, B'_i to get t_i for all i, and then resolve the t_i's with T_n). Therefore by Proposition 3.4, the Equality formulas are easy to refute in $\mathtt{D}^{\mathtt{rrs}}$ + QCDCL and $\mathtt{D}^{\mathtt{rrs}}$ + QCDCL($\mathtt{D}^{\mathtt{rrs}}$).

Finally we come to the system QCDCL($\mathtt{D}^{\mathtt{rrs}}$). It turns out that these formulas are also easy to refute in this system, but this is not so straightforward. In particular, it does not follow merely because red-$\mathtt{D}^{\mathtt{rrs}}(\mathtt{Equality})$ is easy to refute in Res; see Remark 3.5. We describe the refutation below.

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

Proof. Since $\mathtt{D}^{\mathtt{rrs}}(\mathtt{Equality}_n) = \emptyset$, the propagation policy and clause learning always reduce the universal u variables from the corresponding clauses.

We will construct a polynomial size refutation for the $\mathtt{Equality}_n$ formulas containing $2(n-1)$ trails. Define the following clauses: For $i \in [n]$, $T_i = \bigvee_{j \leq i} \bar{t}_j$; for $i \in [n] \setminus \{1\}$, $L_i = \bar{x}_i \vee T_{i-1}$ and $R_i = x_i \vee T_{i-1}$.

We will construct the trails $\mathcal{U}_{n-1}, \mathcal{V}_{n-1}, \mathcal{U}_{n-2}, \mathcal{V}_{n-2}, \cdots, \mathcal{U}_1, \mathcal{V}_1$, and learn clauses $L_{n-1}, R_{n-1}, \cdots L_1, R_1$ corresponding to these trails. The $\mathcal{U}$ trails decide x variables (as many as is possible until conflict) positively; the $\mathcal{V}$ trails decide them negatively. Due to the RED policy, each decision propagates at least one t literal.

The initial trail is

$$\mathcal{U}_{n-1} := (\mathbf{x_1}, t_1; \mathbf{x_2}, t_2, ..., \mathbf{x_{n-1}}, t_{n-1}, \bar{t}_n, x_n, \Box)$$

and the antecedent clauses are $\mathtt{ante}(t_j) = B_j$ for $j \in [n-1]$, $\mathtt{ante}(\bar{t}_n) = T_n$, $\mathtt{ante}(x_n) = A_n$, and $\mathtt{ante}(\Box) = B_n$. From these set of clauses we learn the clause $L_{n-1} = \bar{x}_{n-1} \vee T_{n-2}$.

Restarting, create a symmetric trail to $\mathcal{U}_{n-1}$ flipping each decision:

$$\mathcal{V}_{n-1} := (\bar{\mathbf{x}}_\mathbf{1}, t_1; \bar{\mathbf{x}}_\mathbf{2}, t_2, ..., \bar{\mathbf{x}}_\mathbf{n-1}, t_{n-1}, \bar{t}_n, x_n, \Box)$$

where the antecedent clauses are $\mathtt{ante}(t_j) = A_j$ for $j \in [n-1]$, $\mathtt{ante}(\bar{t}_n) = T_n$, $\mathtt{ante}(x_n) = A_n$, and $\mathtt{ante}(\Box) = B_n$. From these set of clauses we learn the clause $R_{n-1} = x_{n-1} \vee T_{n-2}$.

We now go down now from $i = n-2$ down to $i = 2$. At stage i, we first construct trail $\mathcal{U}_i$ by deciding x variables positively; we reach a conflict after deciding x_i. The trail and antecedent clauses are as follows:

$$\mathcal{U}_i := (\mathbf{x_1}, t_1; \mathbf{x_2}, t_2, ..., \mathbf{x_i}, t_i, x_{i+1}, \Box)$$

with $\mathtt{ante}(t_j) = B_j$ for $j \in [i]$, $\mathtt{ante}(x_{i+1}) = L_{i+1}$, $\mathtt{ante}(\Box) = R_{i+1}$. From this we learn the clause L_i.

Next, we create the symmetrical trail by deciding the x variables negatively

$$\mathcal{V}_i := (\bar{\mathbf{x}}_\mathbf{1}, t_1; \bar{\mathbf{x}}_\mathbf{2}, t_2, ..., \bar{\mathbf{x}}_\mathbf{i}, t_i, x_{i+1}, \Box)$$

with antecedent clauses $\texttt{ante}(t_j) = A_j$ for $j \in [i]$, $\texttt{ante}(x_{i+1}) = L_{i+1}$, $\texttt{ante}(\Box) = R_{i+1}$. From this we learn the clause R_i.

The proof ends with the two trails

$$\mathcal{U}_1 = (\mathbf{x_1}, t_1, x_2, \Box)$$

with antecedents $\texttt{ante}(t_1) = B_1$, $\texttt{ante}(x_2) = L_2$, $\texttt{ante}(\Box) = R_2$, from which we learn $L_1 = \bar{x}_1$, and finally the last trail

$$\mathcal{V}_1 = (\bar{x_1}, t_1, x_2, \Box)$$

with antecedents $\texttt{ante}(\bar{x}_1) = L_1$, $\texttt{ante}(t_1) = B_1$, $\texttt{ante}(x_2) = L_2$, $\texttt{ante}(\Box) = R_2$, from which we learn the empty clause $\Box$, completing the refutation.
The $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$ refutation we have created has $O(n)$ trails and hence overall size $O(n^2)$. ◀

Thus the `Equality` formulas, which are hard for both `Q-Res` and `QCDCL`, become easy to refute when the power of $\texttt{D}^{\texttt{rrs}}$ 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], in order to compare `QCDCL` with the variant with the `NO-RED` policy. The idea is to juxtapose two proposotional formulas, one hard for `Res` and one easy for `Res`, and judiciously interject universal and existential variables tying the two together. The tying is done in such a way that QCDCL trails with the `NO-RED` policy can quickly get to the easy part, whereas with `RED` and the ensuing forced propagations, `QCDCL` is trapped into refuting the hard part. Thus for QCDCL proof systems, allowing reductions, which force more unit propagations in a trail, is not necessaririly a good thing.

▶ **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* $\texttt{PHP}_n^{n+1}$*, and the following matrix:*

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

In [7], it was shown that these formulas are easy to refute in $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$ and `Q-Res` (and hence also in $\texttt{Q}(\texttt{D}^{\texttt{rrs}})\texttt{-Res}$ and `QU-Res`), but are hard to refute in `QCDCL` because the reductions force unit propagations in the trails which send the solver down a "trap" of refuting `PHP`.

Clearly, $\texttt{D}^{\texttt{rrs}}(\texttt{Trapdoor}) = \emptyset$ since the universal variables appear in only one polarity. Thus $\texttt{red-D}^{\texttt{rrs}}(\texttt{Trapdoor})$is the following propositional formula:

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

This formula has a very short `Res` refutation involving the four y, t clauses for any i. Hence by Proposition 3.4, the `Trapdoor` formulas are easy to refute in $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}$ and $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$.

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

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

Proof. As seen before, $\texttt{D}^{\texttt{rrs}}(\texttt{Trapdoor}_n) = \emptyset$. Using this fact we can construct a $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$ 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. More precisely, the first trail is

$$T_1 := (\mathbf{y_1}, t, \square)$$

with $\texttt{red-D}^{\texttt{rrs}}(\texttt{ante}(\square)) = \texttt{red-D}^{\texttt{rrs}}(\bar{y}_1 \vee w \vee \bar{t}) = (\bar{y}_1 \vee \bar{t})$, and $\texttt{red-D}^{\texttt{rrs}}(\texttt{ante}(t)) = \texttt{red-D}^{\texttt{rrs}}(\bar{y}_1 \vee w \vee t) = (\bar{y}_1 \vee t)$. This allows us to learn the clause $(\bar{y}_1)$. The second trail begins by propagating $\bar{y}_1$.

$$T_2 := (\bar{y}_1, t, \square)$$

Here, $\texttt{red-D}^{\texttt{rrs}}(\texttt{ante}(\square)) = \texttt{red-D}^{\texttt{rrs}}(y_1 \vee w \vee \bar{t}) = y_1 \vee \bar{t}$, $\texttt{red-D}^{\texttt{rrs}}(\texttt{ante}(t)) = \texttt{red-D}^{\texttt{rrs}}(y_1 \vee w \vee t) = y_1 \vee t$ and $\texttt{ante}(\bar{y}_1) = \bar{y}_1$ Therefore, we can learn the empty clause $(\square)$, completing the refutation. ◀

Thus, we see that the Trapdoor formulas which were hard for QCDCL become easy to refute when the power of $\texttt{D}^{\texttt{rrs}}$ is added to the QCDCL system, be it in preprocessing, unit propagation or both. This is another demonstration of the advantage of discarding spurious dependencies.

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

From the previous sub-sections it may appear that adding $\texttt{D}^{\texttt{rrs}}$ to QCDCL only adds to its strength and suggests that the addition of the dependency scheme gives us a strictly stronger proof system as with Q-Res. However, this is not the case; QCDCL's are tricky proof systems. In the previous sections we discussed the QParity and Trapdoor formulas; these were used in [7], to show that neither of $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$ and QCDCL simulates the other. For the Trapdoor formulas, the reduction sends the QCDCL refutation down a *"trap"* but not the $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$ refutation. This motivates the idea of designing a formula where adding the dependency scheme enables reductions that send the refuatation down a trap into which the seemingly weaker systems do not fall. Based on this idea, we introduce the family Dep-Trap which is a slight modification of the Trapdoor family, and is defined as follows:

▶ **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.*

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

Note that there are two differences from the $\texttt{Trapdoor}_n$ formulas. Firstly, the universal variable u which was earlier quantified at the end is now quantified just before the existential variables $x_1, \cdots, x_{s_n}$. Secondly, there is an additional clause $\bar{w} \vee \bar{t}$. We will see that these serve a dual purpose: the shifting of the position of u stops QCDCL from falling into the trap, making the formulas easy to refute in QCDCL, while the additional clause prevents the $\texttt{D}^{\texttt{rrs}}$ scheme from bypassing the trap as it used to in the $\texttt{Trapdoor}_n$ formulas, since now instead of being the empty set, $\texttt{D}^{\texttt{rrs}}(\texttt{Dep-Trap}) = \{(w, t)\}$. Therefore, $\texttt{red-D}^{\texttt{rrs}}(\texttt{Dep-Trap})$ isn't a propositional formula anymore; in fact it is the following QBF:

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

We observe below that for exactly the same reasons as `Trapdoor`, the `Dep-Trap` formulas are easy to refute in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$, and hence also in `Q-Res`, $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})$`-Res`, and `QU-Res`. Furthermore, the shifting of u to before the x_i's stops `QCDCL` from going down the "trap", and hence the formulas are easy to refute in `QCDCL` (and in $\mathtt{QCDCL}^{\mathtt{cube}}$).

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

Proof. First we will show that the `Dep-Trap` formulas have a linear-size refutation in `QCDCL`. We will do so by constructing a complete refutation. The first trail decides all y variables positively, decides w negatively, and then propagates t and a conflict. Unlike `Trapdoor` we don't propagate an x after an y decision, since the u cannot be reduced and so blocks unit propagation.

$$T_1 := (\mathbf{y_1}; \mathbf{y_2}; \cdots; \mathbf{y_{s_n}}; \bar{\mathbf{w}}, t, \square)$$

where $\mathtt{ante}(\square) = (\bar{y}_1 \vee w \vee \bar{t})$ and $\mathtt{ante}(t) = (\bar{y}_1 \vee w \vee t)$. Hence the set of learnable clauses for this trail is $L_{T_1} = ((\bar{y}_1 \vee w \vee \bar{t}), (\bar{y}_1))$; allowing us to learn the clause $(\bar{y}_1)$. Now the second trail propagates $(\bar{y}_1)$ followed by remaining decisions as before.

$$T_2 := (\bar{y}_1; \mathbf{y_2}; \cdots; \mathbf{y_{s_n}}; \bar{\mathbf{w}}, t, \square)$$

where $\mathtt{ante}(\square) = (y_1 \vee w \vee \bar{t})$, $\mathtt{ante}(t) = (y_1 \vee w \vee t)$, and $\mathtt{ante}(\bar{y}_1) = \bar{y}_1$. Therefore, the set of learnable clauses for this trail is $L_{T_2} = ((y_1 \vee w \vee \bar{t}), (y_1), (\square))$ which allows us to learn the empty clause $(\square)$, completing the refutation. The refutation size is $O(s_n)$, which is linear in the formula size.

Since none of the unit propagations in the two trails above employed a universal reduction, the same refutation is also a valid linear-size refutation in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$. Since all the other systems mentioned in the lemma simulate $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$, `Dep-Trap` has polynomial-size refutations in these proof systems as well. ◀

Next we show that these formulas are hard to refute in `QCDCL` variants that use $\mathtt{D}^{\mathtt{rrs}}$.

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

Proof. $\mathtt{D}^{\mathtt{rrs}}(\mathtt{Dep\text{-}Trap}_n) = \{(w,t)\}$. This means that the universal variable w cannot be reduced from any axiom clause it appears in as they all also contain the variable t, whereas the universal variable u can be reduced from the axiom clauses as no existential variable depends on it.

Let us first see hardness for $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$. Since u can be reduced but not w, the proof of hardness of `Trapdoor` in `QCDCL` from [7] carries over as is to hardness of `Dep-Trap` in $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$: the decisions on the y variables propagate x literals, sending the trails down the `PHP` trap. The additional clause cannot change the course of such trails, because its variables appear after the y part in the prefix and decisions are required to be level-ordered.

Next consider the other two variants. Preprocessing yields the QBF $\texttt{red-D}^{\texttt{rrs}}(\texttt{Dep-Trap})$ described above. Now all trails in a refutation of this formula must start with deciding the y_i's; these decisions propagate the x_i's; and w cannot be reduced from the clauses it exists in (irrespective of whether we allowed $\forall(\texttt{D})$-reductions or not) nor decided before all the y_i's are decided. Again, the proof of hardness of Trapdoor in QCDCL from [7] lifts to hardness of Dep-Trap in $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}$ and $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$. ◀

The Dep-Trap formulas are thus easy for 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 QCDCL system.

4.5 The $\texttt{TwoPHPandCT}_n$ formulas

The formulas in the previous sections seem to suggest that Proposition 3.4 could also extend to include $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$. We show now that this is not the case. The motivation for defining the following formula also comes from the Trapdoor formulas, using the propositional hardness of PHP and the "easiness" of the (negation of) complete tautology on two variables. The added new element is the use of two disjoint 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*

$$\begin{gathered} u \vee \texttt{PHP}(x_1, \cdots, x_{s_n}) \\ \bar{u} \vee \texttt{PHP}(y_1, \cdots, y_{s_n}) \\ v \vee z_1 \vee z_2 \ , \ v \vee \bar{z}_1 \vee z_2 \ , \ v \vee z_1 \vee \bar{z}_2 \ , \ v \vee \bar{z}_1 \vee \bar{z}_2 \end{gathered}$$

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

$$\begin{gathered} \texttt{PHP}(x_1, \cdots, x_{s_n}) \\ \texttt{PHP}(y_1, \cdots, y_{s_n}) \\ z_1 \vee z_2 \ , \ \bar{z}_1 \vee z_2 \ , \ z_1 \vee \bar{z}_2 \ , \ \bar{z}_1 \vee \bar{z}_2 \end{gathered}$$

This formula is easy to refute in Res using the z_1, z_2 clauses; hence by Proposition 3.4, the original QBFs are easy to refute in $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}$ and $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$.

The final question arises now as to how hard they are to refute in QCDCL, $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$ and $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$. And the answer is that it is hard for all three systems.

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

Proof. None of the three systems QCDCL, $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$or $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$ allow for any preprocessing. Hence the first decision in each of these three systems must be on u, which allows for no propagations in any case. And depending on the choice of u, the next set of decisions are either all on x variables or all on y variables. The variable v could be dropped during unit propagation in the $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$ system, but neither z_1 or z_2 could be decided or propagated before all the y or x variables are decided/propagated. Therefore, all conflicts these trails hit come directly from the PHP clauses. Thus refuting TwoPHPandCT in QCDCL, $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$ or $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$ is equivalent to refuting PHP in CDCL, requiring exponential size. ◀

These formulas highlight two important facts: firstly that $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$ is not the same as $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}$ or $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$ 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 $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$.

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 a formulas comes from trying to ascertain whether after preprocessing with $\texttt{D}^{\texttt{rrs}}$, does allowing reductions using $\texttt{D}^{\texttt{rrs}}$ 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 prefix has the variables of `Equality` sandwiched between a and b. The underlying idea in the formulation is that unlike $\texttt{D}^{\texttt{rrs}}(\texttt{Equality})$ which is empty, the additional C_i clauses make $\texttt{D}^{\texttt{rrs}}(\texttt{RRSTrapEq}) = \{(u_i, b) : i \in [n]\}$. Further, adding b to the x, u, t clauses along with that results in $\texttt{red-D}^{\texttt{rrs}}(\texttt{RRSTrapEq}) = \texttt{RRSTrapEq}$. Now, since preprocessing by $\texttt{D}^{\texttt{rrs}}$ does not change the formula at all, therefore the refutational hardness and in fact the refutation for these two formulas will be exactly the same for the pair of systems `QCDCL` and $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}$, and similarly for the pair $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$ and $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$.

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

Proof. Since, $\texttt{red-D}^{\texttt{rrs}}(\texttt{RRSTrapEq})= \texttt{RRSTrapEq}$, there is no effect of preprocessing, and all 4 systems start off by refuting the same formula. Now, for any of the four proof systems, if you consider any trail $\mathcal{T}$ of a refutation, the first decision must be on the variable a, and irrespective of the manner of that decision we propagate $\bar{b}$, which satisifies all the C_i clauses and removes the b literal from the A_i and B_i clauses. The remaining formula at this point is exactly the `Equality` formula, i.e. $\texttt{RRSTrapEq}|_{a=*,b=0} = \texttt{Equality}$, and hence refuting the `RRSTrapEq` formulas is equivalent to refuting the `Equality` formulas in `QCDCL` and $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$, which we know are hard and easy respectively.

Therefore, `RRSTrapEq` formulas have polynomial size refutations in $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$ and $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$, but require exponential size refutations in `QCDCL` and $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}$ ◀

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 $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$. However, they are easily seen to have short refutations in $\texttt{Q}(\texttt{D}^{\texttt{rrs}})\texttt{-Res}$: resolve the last two clauses to derive $\bar{b}$, use it to remove b from the A_i and B_i, now use $\texttt{D}^{\texttt{rrs}}$ reductions to remove the u literals, resolve on x to derive unit t_i clauses, and remove them from T_n sequentially.

4.7 The $\texttt{PreDepTrap}_n$ formulas

The previous section underlined that preprocessing by $\texttt{D}^{\texttt{rrs}}$ may not necessarily make $\texttt{D}^{\texttt{rrs}}$ during propagation obsolete and/or give an advantage. It is reasonable to believe that at least it won't make things worse. But the following example shows that this is not the case: preprocessing before allowing a QCDCL system to refute could in fact make the refutation harder.

The construction of the formula is pretty straightforward. It is the disjoint union of the two formulas Dep-Trap and Equality, connected by a universal quantified right at the beginning, appearing in the two sub-formulas in opposite polarites.

▶ **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*

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

(Recall that $a \vee$ Dep-Trap *is the disjunction of* a *with every clause of* Dep-Trap*, and similarly for* $\bar{a} \vee$ Equality*.)*

First consider the $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$ and QCDCL systems. Since there is no preprocessing, the first decision of every trail must set the variable a. For a trail that starts with the decision $\bar{a}$, the PreDepTrap formula reduces exactly to Dep-Trap formula which we know is easy to refute in $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$ as well as QCDCL (Section 4.4). Therefore, the PreDepTrap formulas are easy to refute in QCDCL and $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$, and as a consequence in Q-Res, $\texttt{Q(D}^{\texttt{rrs}}\texttt{)-Res}$, QU-Res, and $\texttt{QCDCL}^{\texttt{cube}}$.

Next, consider adding $\texttt{D}^{\texttt{rrs}}$. Since the opposite polarities of a appear in clauses with disjoint non-interacting sets of variables, no existential variable depends on a. Since $\texttt{D}^{\texttt{rrs}}(\texttt{Equality}) = \emptyset$, no variable depends on a q variable. Thus $\texttt{D}^{\texttt{rrs}}(\texttt{PreDepTrap}) = \texttt{D}^{\texttt{rrs}}(\texttt{Dep-Trap})$,

In the $\texttt{QCDCL(D}^{\texttt{rrs}}\texttt{)}$ system, where there is no preprocessing, the first decision of every trail must again set the variable a. For trails that start with the decison a, the formula immediately reduces exactly to the Equality formulas, which are easy to refute in $\texttt{QCDCL(D}^{\texttt{rrs}}\texttt{)}$, Lemma 4.1. The refutation in the proof of Lemma 4.1, preceded by the decision a, gives a valid refutation for the PreDepTrap formulas. Therefore these formulas are easy to refute in $\texttt{QCDCL(D}^{\texttt{rrs}}\texttt{)}$.

Finally, consider the case when the formula is preprocessed. We show that this makes refutations exponentially long.

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

Proof. Since $\texttt{D}^{\texttt{rrs}}(\texttt{PreDepTrap}) = \texttt{D}^{\texttt{rrs}}(\texttt{Dep-Trap})$, the formula $\texttt{red-D}^{\texttt{rrs}}(\texttt{PreDepTrap})$ has the matrix

$$\texttt{PHP}^{n+1}_n(x_1, \cdots, x_{s_n})$$
$$\text{for } \ i \in [s_n]: \quad \bar{y}_i \vee x_i\ ,\ y_i \vee \bar{x}_i$$
$$\text{for } \ i \in [s_n]: \quad y_i \vee w \vee t\ ,\ y_i \vee w \vee \bar{t}\ ,\ \bar{y}_i \vee w \vee t\ ,\ \bar{y}_i \vee w \vee \bar{t}$$
$$\bar{w} \vee \bar{t}$$
$$()\bar{r}_1 \vee \cdots \vee \bar{r}_n)$$
$$\text{for } \ i \in [n] \quad (p_i \vee r_i)$$
$$\text{for } \ i \in [n] \quad (\bar{p}_i \vee r_i)$$

Now the variable a, though still in the quantifier prefix, has no effect on the trails. It can be decided arbitrarily in the beginning, and then the goal is to refute $\texttt{red-D}^{\texttt{rrs}}(\texttt{PreDepTrap})$ in QCDCL and $\texttt{QCDCL(D}^{\texttt{rrs}}\texttt{)}$. At this point, therefore, all trails must start with deciding the y_i's, propagating the x_i's. Since the only remaining universal w cannot be reduced from the clauses it occurs in nor decided before all the y_i's are decided, and since the p_i's or r_i's also cannot be decided before the y_i's, the trails are led into the trap of refuting PHP. The proof of hardness of Trapdoor in QCDCL from [7] carries over exactly as it is to show hardness for $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$ and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$. ◀

4.8 The $\mathtt{PropDep\text{-}Trap}_n$ formulas

The previous section illustrated an example where having $\mathtt{D}^{\mathtt{rrs}}$ as a preprocessing technique was a detriment to refuting it and in fact it was better to use $\mathtt{D}^{\mathtt{rrs}}$ only in unit propagation, and not also for preprocessing. This leads to the question — could there be a formula where having $\mathtt{D}^{\mathtt{rrs}}$ only for preprocessing was strictly better than having it for both preprocessing and propagation? Addressing this led to the birth of the following formula which is a slight modification of the Dep-Trap formulas, and which witnesses that the answer is yes.

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

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

First observe that the presence of the four s, w, t clauses make this formula very easy to refute in Q-Res and hence in $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})\text{-}\mathtt{Res}$ and QU-Res.

Next, notice that the formulas are also easy to refute in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$, QCDCL, and $\mathtt{QCDCL}^{\mathtt{cube}}$ because, after the initial propagation of the unit clauses $\bar{z}_1, \bar{z}_2$ in level 0, there are no propagations possible before a w decision; the decisions on s and all the y_i's cause no propagations. A trail that decides s and $\bar{w}$ quickly reaches a conflict and learns $\bar{s}$; a next trail that propagates $\bar{s}$ and decides $\bar{w}$ then learns the empty clause.

Coming to $\mathtt{D}^{\mathtt{rrs}}$, it can be seen that $\mathtt{D}^{\mathtt{rrs}}(\mathtt{PropDep\text{-}Trap}) = \{(w, t), (b_1, z_1), (b_2, z_2)\}$. Therefore, $\mathtt{red\text{-}D}^{\mathtt{rrs}}(\mathtt{PropDep\text{-}Trap})= \mathtt{PropDep\text{-}Trap}$. This means that a QCDCL refutation is also a $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$ refutation, and therefore PropDep-Trap has short refutations in $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$.

We now show that refuting these formulas in $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$ and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$ is hard.

▶ **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. Since $\mathtt{red\text{-}D}^{\mathtt{rrs}}(\mathtt{PropDep\text{-}Trap})= \mathtt{PropDep\text{-}Trap}$, a $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$ refutation in this case is just a $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$ refutation, so it suffices to show hardness for the latter. Let us observe how a $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$ refutation looks. The two unit clauses $\bar{z}_1, \bar{z}_2$ get propagated initially and then s and the y_i's have to be decided. A decision on s, irrespective of the polarity, causes no further propagations until w is also decided, as every clause containing the variable s also contains the variables w and t, and the reduction of w is blocked by t. Since neither (b_1, x_i) nor (b_2, x_i) is in $\mathtt{D}^{\mathtt{rrs}}$, a decision on variable y_i propagates either x_i (due to the clause containing $\bar{y}_i$, where b_1 can now be reduced), or $\bar{x}_i$ (due to the clause containing y_i, where b_2 can now be reduced). The propagating of x_i due to y_i sends the $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$ refutation down the same "trap" as the Trapdoor formulas for QCDCL[7], and thus $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$ requires exponential size refutations to refute these formulas. ◀

4.9 The $\mathtt{TwinEq}_n$ formulas

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

▶ **Formula 9** ($\texttt{TwinEq}_n$ [13]). *The* $\texttt{TwinEq}_n$ *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, and in fact is the same formula as $\texttt{red-D}^{\texttt{rrs}}(\texttt{Equality})$. Therefore, due to the same argument as for the $\texttt{Equality}$ formulas in Section 4.2, the $\texttt{TwinEq}$ formulas are easy to refute in $\texttt{Q(D}^{\texttt{rrs}}\texttt{)-Res}$, $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}$, $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL(D}^{\texttt{rrs}}\texttt{)}$ and $\texttt{QCDCL(D}^{\texttt{rrs}}\texttt{)}$.

5 Relation between proof systems

The previous section saw us study the bounds for known as well as our newly-constructed formulas in a plethora of proof systems. Using these obtained bounds, in this section we look to obtain the relations of our newly defined proof systems using $\texttt{D}^{\texttt{rrs}}$ with $\texttt{QCDCL}$ among themselves, as well as their relation with other QBF proof systems.

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.

▶ **Theorem 5.1.** *The proof systems in* $\{\texttt{QCDCL}, \texttt{QCDCL(D}^{\texttt{rrs}}\texttt{)}, \texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}, \texttt{D}^{\texttt{rrs}} + \texttt{QCDCL(D}^{\texttt{rrs}}\texttt{)}\}$ *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 $\texttt{QCDCL(D}^{\texttt{rrs}}\texttt{)}$, while the $\texttt{PreDepTrap}$ formulas (Section 4.7) are easy only in $\texttt{QCDCL(D}^{\texttt{rrs}}\texttt{)}$. Hence $\texttt{QCDCL(D}^{\texttt{rrs}}\texttt{)}$ is incomparable with the systems that use preprocessing.

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

Next we observe that each of the three new versions of $\texttt{QCDCL}$ is also incomparable with $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$, $\texttt{Q-Res}$, $\texttt{Q(D}^{\texttt{rrs}}\texttt{)-Res}$, and $\texttt{QU-Res}$.

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

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

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

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

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 but never lengthens refutations, 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.

One aspect which is unresolved in our work is the nature of our upper bounds. Since `QCDCL` is not in itself an algorithm but a template, there are multiple instantiations of it based on the choice of decision heuristics, propagation policy, and learning scheme. We have restricted ourselves here to the decision heuristic and propagation policy used in most state-of-the-art solvers, namely, level-ordered decisions and propagations with reductions. However, we have not specified the learning scheme. 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. Thus the impact of the learning scheme itself is still improperly understood for dependency-aware `QCDCL`.

In this work, we only consider decisions in `LEV-ORD`. Whether incorporating dependency schemes in the decision policy itself makes a difference or not is an interesting question. Formalisation seems tricky as it seems to be more like `D-ORD` than `LEV-ORD(D)`. As recent work in [12] shows, other orders are not necessarily unsound, so this can still be meaningful. However, this is quite a different direction to explore. There is also the dependency-learning setup which is quite different: the solver starts off assuming there are no dependencies, and gradually builds up a set of dependencies. (That is, add required dependencies starting from $\emptyset$, rather than remove spurious dependencies starting from $\mathtt{D}^{\mathtt{trv}}$.) In this approach, explored in [17] and used in the QBF solver Qute, learning dependencies does affect decision order. An in-depth future comparison of these two approaches could be interesting to explore.

Another aspect we have not directly addressed, other than showing soundness and completeness, is the effect of dependency schemes other than $\mathtt{D}^{\mathtt{rrs}}$. Since $\mathtt{D}^{\mathtt{rrs}}$ is what is actually used in most QBF solvers, it is important to understand its effect first. But there are other schemes: the less (than $\mathtt{D}^{\mathtt{rrs}}$) general standard dependency scheme $\mathtt{D}^{\mathtt{std}}$ introduced in [20], 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.

Finally, an aspect to our work that could also be interesting to investigate, with potential ramifications to practical solvers, is exploring the addition of cube-learning to our dependency-aware QCDCL systems. 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 [18]).

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. URL: https://eccc.weizmann.ac.il/report/2021/135, 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. URL: https://lmcs.episciences.org/11193, 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? *Electron. Colloquium Comput. Complex.*, page 040., 2022. (to appear in SAT 2022). URL: https://eccc.weizmann.ac.il/report/2022/040.

12 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.

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, 7 2022. Main Track. doi:10.24963/ijcai.2022/248.

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

15 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.

16 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.

17 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.

18 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.

19 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.

20 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.

21 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.

22 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.

23 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.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il