Strong (D)QBF Dependency Schemes via Pure Paths with Applications to Proof Checking

Leroy Chew
Czech Technical University in Prague, Czech Republic

Tomáš Peitl
TU Wien, Austria

Abstract

Certification for Quantified Boolean Formulas (QBF) and Dependency Quantified Boolean Formulas (DQBF) is an ongoing challenge. Recent proof complexity work has shown that the majority of QBF and DQBF techniques can be p-simulated by using the independent extension rule.

In propositional logic, extension rules are supported by proof checkers using a more general RAT (Resolution Asymmetric Tautology) rule. The next step in (D)QBF certification would be to update these modern RAT formats to match the strength of this independent extension rule.

In this paper we first introduce a new dependency scheme called $\mathcal{D}^{\forall\text{pure}}$. This rule is the missing ingredient that when added to Blinkhorn's proof system DQRAT allows it to be provably p-equivalent to the Independent Extended QU-Res, the most powerful of the known QBF and DQBF proof systems. Up until now, DQRAT has only existed in theory, so we implement a prototype checker `DQRAT-check` which includes our extra rule.

In addition to its inclusion in our proof checker we show $\mathcal{D}^{\forall\text{pure}}$ has other properties that have been found for previous dependency schemes, and each of these observations has potential in solving/checking including the sound integration into the dependency learning solver `Qute`.

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases DQBF, QBF, Qute, Proof Systems, Dependency Schemes, Dependency Learning, Skolem functions

Digital Object Identifier 10.4230/LIPIcs.SAT.2026.11

Supplementary Material *Software*: `https://doi.org/10.5281/zenodo.20203771`

Funding *Leroy Chew*: This project is supported by FWF ESPRIT grant number ESP-197, by the European Union under the project ROBOPROX (reg. no. CZ.02.01.01/00/22_008/0004590) and by the Czech Science Foundation project 24-12759S

Acknowledgements Thanks to Martina Seidl for discussions on this topic, to Mirek Olsak and Zarathustra Goertzel for help debugging DQRAT-check and to the anonymous reviewers for improvements to the paper.

1 Introduction

Our main motivation comes from the recent introduction of a new proof rule known as independent extension [18], which has desirable certification properties. The central observation of this paper is that we can readily simulate independent extension, with only a quick "fix" of existing certification rules. The fix is a sound relaxation of the order of quantification and also works with solvers. In this paper we explore how, why and to what extent this works, before implementing it in a proof checker.


© Leroy Chew and Tomáš Peitl;
licensed under Creative Commons License CC-BY 4.0

1.1 Background

The canonical NP-complete problem is propositional satisfiability (SAT), and if we extend it with a prefix of Boolean quantifiers we get the Quantified Boolean Formula (QBF) problem which is then considered the canonical PSPACE-complete problem. We can extend QBF even further into Dependency QBF (DQBF), where Boolean quantification still occurs, but the quantification order is not dictated by a linear prefix. Instead, every existentially quantified variable is given an explicit set of variables that it comes "after". DQBF is considered a more expressive language than QBF as it is NEXPTIME-complete rather than in PSPACE.

While research into DQBF is studied for its own sake, we are very interested where DQBFs appear in QBF research, because making changes to the quantification order in a QBF can make a QBF easier to solve or prove. Occasionally such changes require us to shift to a non-linear quantifier prefix. One common example is when solvers and proof systems employ *dependency schemes*, which calculate whether a dependency of one quantified variable to another is necessary or spurious. We can think of this as a transformation from a QBF to a DQBF as the prefix is recalculated to remove spurious dependencies. An example of well-studied dependency is the reflexive resolution path dependency scheme denoted by the symbol $\mathcal{D}^{\text{rrs}}$. This detects that a dependency is spurious if there is no sufficient pair of resolution paths (see Section 2 for the definition of a resolution path) from $\forall$ to $\exists$ literals.

The use of a whole range of transformations from QBF to DQBF, which is often done only implicitly, has made deciding a commonly agreed upon QBF proof format more challenging. Theoretical proof systems such as the sequent calculus G [31](named after Gentzen who pioneered sequent calculi) assume a neat symmetry between true and false as QBF is closed under negation and PSPACE is self-complementary, but DQBF is not alike in this way and has an inherent asymmetry between $\exists$ and $\forall$. In addition, theoretical proof systems make poor checking formats in a practical setting because they do not generalise existing practical formats and often proofs with polynomial size upper bounds are still too large in practice.

One noteworthy QBF format that attempts to tackle this issue is QRAT (Quantified Resolution Asymmetric Tautology)[24]. In addition to generalising the rules of the propositional proof system DRAT (Deletion Resolution Asymmetric Tautology), which is the closest system propositional logic has to a standard format, QRAT can detect the same side conditions that a dependency scheme detects. Instead of modifying the quantifier prefix, QRAT modifies a clause. Such a combination of rules ends up being incredibly powerful and QRAT can p-simulate a majority of the solving and preprocessing techniques in QBF, despite the large disparity of these techniques. This is incredibly fortunate overall, but it is not directly possible in QRAT to treat a QBF with a dependency scheme as a DQBF, as we soundly ought to be able to do. In fact, QRAT has been proven on a theoretical level to be no stronger than G [17], so even its detection of dependency schemes is no more than a sufficient combination of fundamental QBF steps. In an attempt to build QRAT into a genuine DQBF proof system, Blinkhorn proposed [15] the DQBF proof system DQRAT (Dependency Quantified Resolution Asymmetric Tautology) and DQRAT can handle dependency schemes directly. However, DQRAT has not been developed since its initial introduction, nor is there a substantial body of proof complexity follow up work that discusses it.

Nonetheless, work into strong DQBF proof systems may yet be fruitful. Recent developments in QBF proof complexity have shown that a new DQBF proof system IndExtQURes (where the power comes from an Independent Extension rule) can p-simulate practically everything in QBF and DQBF including G, QRAT, DQRAT and some dependency scheme rules such as $\mathcal{D}^{\text{rrs}}$ [18]. IndExtQURes is not yet a practical format, and the proposed next step in the paper by Chew and Peitl introducing IndExtQURes was to find a RAT version which was p-equivalent or more powerful.

In this paper we create a DQRAT proof checker and we add one additional rule to it. We demonstrate theoretically that adding this rule allows it to p-simulate IndExtQURes. This rule is simply a new dependency scheme, which we name the pure universal dependency scheme ($\mathcal{D}^{\forall\text{pure}}$).

We cover the proof theoretical and proof complexity properties of $\mathcal{D}^{\forall\text{pure}}$. Many of the properties, that we show, strengthen the case for $\mathcal{D}^{\forall\text{pure}}$ in various practical settings. For example, showing that $\mathcal{D}^{\forall\text{pure}}$ allows for strategy extraction in a resolution proof system shows that $\mathcal{D}^{\forall\text{pure}}$ can soundly be integrated into the solver `Qute`.

As we have already emphasised, our major contribution is that $\mathcal{D}^{\forall\text{pure}}$ can strengthen DQRAT to be at least as strong as all other well-studied QBF and DQBF proof systems theoretical and practical. We avoid having to show each of these p-simulations individually, instead we show that DQRAT+$\mathcal{D}^{\forall\text{pure}}$ and IndExtQURes are p-equivalent, and rely transitively on the p-simulation results on IndExtQURes shown in the paper by Chew and Peitl [18].

1.2 Related Work

QBF and DQBF proof complexity have been extensively studied, including work on QBF clause learning proof systems systems [2, 3, 10], expansion-based proof systems [25, 14, 12] and stronger proof systems that go beyond current solving techniques [24, 11, 31]. One specific subtopic, the line of dependency scheme research, has progressed over the last decade. The *standard dependency scheme* was developed originally by Samer and Szeider [42]. The *reflexive resolution path dependency scheme* was developed by Slivovsky and Szeider [43]. The *tautology-free dependency scheme* was developed by Beyersdorff, Blinkhorn and Peitl [8]. Later the same set of authors developed a framework of an infinite number of *implication-free dependency schemes* [9]. The schemes progress in difficulty and trend towards conceptual dependency schemes that may not have polynomial-time checkability.

The main motivation of this work was to capture the power of independent extension clauses [18] for DQBF. Independent extension generalises weaker forms of QBF extension in QBF [26, 11]. Independent extension involves conditioning on assignments, and similar ideas have been employed in other works such as conditional autarkies [27, 33]. There are many generalisations that capture extension clauses in other domains. Blocked clause elimination and addition considers redundant clauses that can be safely added or removed without changing satisfiability [32]. Resolution asymmetric tautologies (RAT) [22] generalise blocked clauses, and propagation redundancies generalise RAT even further [23].

This work involves the improvement of DQRAT to DQRAT+$\mathcal{D}^{\forall\text{pure}}$. DQRAT [15] is the result of generalising RAT addition rules in DQBF. Previously RAT was generalised into QBF through QRAT [24] and QRAT+ [35], these are QBF proof systems, but specifically designed for certification. As an alternative to these formats, the QRP format [36] can be used to provide more straightforward resolution type proofs, and recent work [40] aims to expand the potential of QRP proofs.

2 Preliminaries

2.1 Propositional Logic

We assume the reader is familiar with propositional logic, literals and conjunctive normal form (CNF). We use the $\bar{x}$ notation to switch between a negated and non-negated literals and acts as an involution: $\bar{0} = 1$, $\bar{1} = 0$, $\bar{x} = \neg x$ and $\overline{(\neg x)} = x$.

For CNF ϕ, $\text{var}(\phi)$ is the set of variables appearing in ϕ. A partial assignment α on ϕ is a partial function that maps $\text{var}(\phi)$ to $\{0,1\}$. We consider the restriction of ϕ: $\phi|_\alpha$ to be the copy of ϕ where if $\alpha(x) = 0$ literal x is replaced by 0 and literal $\neg x$ is replaced by 1 and likewise if $\alpha(x) = 1$ literal x is replaced by 1 and literal $\neg x$ is replaced by 0. We can overload α's functional notation to include literals so $\alpha(\neg l) = \neg\alpha(l)$. Additionally we can represent a partial assignment as a string of literals, i.e. $x\bar{z}$ is the partial assignment that maps x to 1 and z to 0.

The negation of a clause C, denoted by $\bar{C}$ or $\neg C$ is a partial assignment, but we can treat it syntactically as a CNF containing only the unit clauses of each of C's literals but negated. Given the equivalence between partial assignments, negated clauses and sets of unit clauses, we can simplify a CNF ϕ that contains unit clauses, by applying the literal of a unit clause as an assignment, removing satisfied clauses and removing falsified literals. This process can be repeated until no unit clauses remain and we defined this end point as $\texttt{UP}(\phi)$.

▶ **Definition 1.** *We say $\phi \vdash_1 \bot$ if the resulting CNF of $\texttt{UP}(\phi)$ contains the empty clause.*

We can soundly derive clause C into a CNF ϕ if $\phi \wedge \bar{C}$ is a contradiction. Checking for a contradiction is CoNP-complete, so we use the weaker condition $\phi \wedge \bar{C} \vdash_1 \bot$. This is often known as reverse unit propagation, here we call it ATA (asymmetric tautology addition).

2.2 Quantified Boolean Formulas

A quantified Boolean formula in closed prenex conjunctive form $\Pi\phi$, contains a CNF ϕ and quantifier prefix Π. Π is a sequence of pairs containing a quantifier symbol from $(\forall, \exists)$ and a propositional variable, i.e. $\Pi = \mathcal{Q}_1 x_1 \dots \mathcal{Q}_k x_k$ where $\mathcal{Q}_i \in \{\forall, \exists\}$ for $1 \leq i \leq k$. $\text{var}(\Pi)$ is the set of variables appearing in prefix Π. Every variable in $\text{var}(\phi)$ occurs exactly once in the prefix Π (i.e. $\text{var}(\phi) \subseteq \text{var}(\Pi)$, we only consider closed sentences).

$\text{var}_\exists(\Pi)$ is the set of existential variables (variables bound by $\exists$) appearing in prefix Π and $\text{var}_\forall(\Pi)$ is the set of universal variables (variables bound by $\forall$) appearing in prefix Π. For assignment α, $\Pi\restriction_\alpha$ removes all variables from $\text{dom}(\alpha)$ and their attached quantification from the prefix. The prefix order $\lesssim_\Pi$ is a total pre-ordering of all variables in the prefix. $x \lesssim_\Pi y$ if and only if x appears left of y, or x and y are in the same uninterrupted contiguous block of variables with the same quantifier symbol.

Because we are working with closed prenex QBFs, the semantics can be defined using Skolem functions. We say that an existential variable x depends on u if u is quantified left of x in the prefix. In this way we can build a *dependency set* D_x^Π for each existentially quantified variable x containing exactly the universal variables that are left ($\lesssim_\Pi$) of x. A *Skolem function* for an existential variable x is a Boolean function $f_x : \{0,1\}^{\text{D}_x^\Pi} \to \{0,1\}$. A QBF is true if and only if there is a set of Skolem functions for each existential variable, such that the Skolem functions together would satisfy the propositional matrix under all complete assignments to $\text{var}_\forall(\Pi)$.

You can imagine a QBF as a game between $\exists$ and $\forall$ in which they set the values of their variables from left to right in the prefix. $\exists$ wins if and only if the matrix evaluates to 1 at the end of the game. The QBF is true if and only if $\exists$ has a winning strategy. Essentially this is no different to the idea of a set of satisfying Skolem functions. This works dually for $\forall$ and falsity and we name the dual of Skolem functions as Herbrand functions.

If we have a QBF $\Pi\phi$ and ϕ contains a clause $C \vee u$ where $\text{var}(u)$ is universal and not contained in the dependency set of any of the variables of C nor is there any $\bar{u}$ literal in C, then we can derive the clause C. This is because any winning $\exists$ strategy that satisfies $C \vee u$ will do so before arriving at u, hence C must be satisfied by the same winning strategy.

$$\frac{C \vee u}{C}\ (\text{UR})$$

One of the earliest observations about QBF was that if we combine the universal reduction rule with resolution, even when the resolution rule is restricted to cutting over existential literals only, then we get a complete refutational proof system for QBF known as Q-Res [30].

2.3 Dependency Quantified Boolean Formulas

Dependency quantified Boolean formulas (DQBF) extend the language of QBF. We will concentrate on the S-form DQBFs (Skolem-form). Like a QBF, every DQBF has two parts, a prefix and propositional matrix in CNF. Also like a QBF, in a DQBF every variable is either existentially or universally quantified in said prefix. In an S-form DQBF, each existential variable x has an arbitrary dependency set D^{Π}_x, the only restriction is that D^{Π}_x is a subset of the universal variables in the prefix Π. The prefix Π is written as $\Pi = \forall u_1 \dots u_p \exists x_1(D_{x_1}) \dots x_q(D_{x_q})$. Typically we omit the parentheses of the dependency set and list the elements. In a DQBF prefix the written ordering does not determine anything, because the essential information is all contained in the specifications of the dependency sets. Nonetheless any QBF can be represented by a DQBF by specifying the QBF's dependency sets explicitly.

A DQBF Φ is true if and only if there is a set $f = \{f_x : x \in \mathrm{var}_\exists(\Phi)\}$ of Skolem functions, one for each $\exists$ variable x, such that for every complete assignment β to all the universal variables the universal assignment completed with the values of the Skolem functions under that assignment, written as $\beta \cup f(\beta)$, form a satisfying assignment to the propositional matrix. For example

$$\forall u \forall v \exists x(u) \exists y(v)(v \vee x) \wedge (\bar{v} \vee \bar{x}) \wedge (u \vee y) \wedge (\bar{u} \vee \bar{y})$$

is false because the x Skolem function must satisfy the first clause when v is false and the second when v is true, but can only respond to u and not v. But

$$\forall u \forall v \exists x(v) \exists y(u)(v \vee x) \wedge (\bar{v} \vee \bar{x}) \wedge (u \vee y) \wedge (\bar{u} \vee \bar{y})$$

is true because the x Skolem function can be $\bar{v}$ and the y Skolem function can be $\bar{u}$. We call such a set of Skolem functions a *model*.

We sometimes informally write $\forall U \exists E \phi$ for an arbitrary S-form DQBF, where U is the set of universal variables, E the set of existential variables each with their own dependency set that we may hide for the time being, and ϕ a propositional matrix containing no quantifiers. We can define a subprefix and the relation $\Pi \subseteq \Omega$, whenever $\mathrm{var}_\exists(\Pi) \subseteq \mathrm{var}_\exists(\Omega)$, $\mathrm{var}_\forall(\Pi) \subseteq \mathrm{var}_\forall(\Omega)$ and if $x \in \mathrm{var}_\exists(\Pi)$ then $\mathrm{D}^{\Pi}_x = \mathrm{D}^{\Omega}_x$.

2.3.1 Pre-ordering a DQBF

Recall that a QBF prefix Π has a total pre-ordering ($\lesssim_\Pi$) whose equivalence classes are quantifier blocks. Instead let Π be a DQBF prefix. It turns out we can still define a pre-ordering on Π, but it may not be total. This allows QRAT to be generalised to the original definition DQRAT, which is the base system we aim to improve. For readers purely interested in the new dependency rule we introduce later, understanding of this pre-ordering is not needed, but it is important for the wider topic of DQBF certification.

First we overload the D^{Π} notation. For universal variables u we say $\mathrm{D}^{\Pi}_u = \{u\}$. For formulas ϕ (including partial assignments) we consider the dependency set D^{Π}_ϕ to be $\bigcup_{x \in \mathrm{var}(\phi)} \mathrm{D}^{\Pi}_x$.

We define the set of outer variables for each variable in Π. For an existential variable x in Π, the outer variables are the set of variables that can be calculated from the Skolem

functions using only the values for the dependency set. The outer variable set $\mathrm{O}^\Pi_x := \{y \in \mathrm{var}(\Pi) \mid \mathrm{D}^\Pi_y \subseteq \mathrm{D}^\Pi_x\}$. For a universal variable u to find its outer variables, we look at its inner existential variables, those existential variables that contain u in its dependency set i.e. $\mathrm{I}^\Pi_u := \{y \in \mathrm{var}(\Pi) \mid \mathrm{D}^\Pi_u \subseteq \mathrm{D}^\Pi_y\}$, and we include any universal variable that is common in all of these, we also include any existential variable that depends only on these common universal variables, though we make an exception and exclude it, if it depends on u, in order to keep the notion the $\forall$ variable comes before the $\exists$ variables that depend on it. $\mathrm{O}^\Pi_u = \{u\} \cup \bigcap_{x \in \mathrm{I}^\Pi_u} \{y \mid \mathrm{D}^\Pi_y \subseteq \mathrm{D}^\Pi_x \setminus \{u\}\}$. With a DQBF prefix Π we can define the relation $\lesssim_\Pi$, so $x \lesssim_\Pi y$ means $x \in \mathrm{O}^\Pi_y$, or equivalently that $\mathrm{O}^\Pi_x \subseteq \mathrm{O}^\Pi_y$.

2.3.2 Dependency Schemes

Dependency schemes are a method of evaluating individual dependency pairs to see if they are really necessary. Let $\Pi\phi$ be a DQBF (or QBF). We consider pairs (u, x), with u a universal variable and x an existential variable. *Spurious* dependencies occur when $u \in \mathrm{D}^\Pi_x$ but it does not change the truth of the DQBF to remove u from D^Π_x. Given a DQBF or QBF $\Pi\phi$ the trivial dependency scheme $\mathcal{D}^{\mathrm{trv}}(\Pi\phi)$ is the set of all pairs (u, x) where u is universal, x is existential and $u \in \mathrm{D}^\Pi_x$.

The most successful dependency schemes analyse resolution paths between clauses. We can represent a CNF as a multi graph where nodes are clauses. There is an edge from C_1 to C_2 whenever there is a variable p such that p is in one clause and $\neg p$ is in the other. We annotate the edge with variable p. A *resolution path* is a sequence of edges $\{e_i \mid 1 \leq i \leq k\}$ that form a path such that no two consecutive edges are annotated with the same variable. We can list a path using the clauses, and the literals in-between. If there is no resolution path between two clauses D and E for a CNF ϕ, then for any resolution refutation of ϕ both D and E cannot occur in the connected part of the proof. It is useful to restrict a resolution path to a particular subset of variables $\mathcal{S}$. So that $\{e_i \mid 1 \leq i \leq k\}$ is a resolution path if all edges are annotated with a variable from $\mathcal{S}$ and all consecutive edges coincide on a clause but have different annotations.

One of the most well used non-trivial dependency schemes is the *reflexive resolution path dependency scheme*. To compute which dependencies actually are required, which we denote as $(u, x) \in \mathcal{D}^{\mathrm{rrs}}(\Pi\phi)$, we use the notion of a resolution path. Let ψ be a DQBF, χ a subset of the clauses in ψ and $\mathcal{S}$ a subset of variables appearing in ψ. We define using a fixpoint a set of clauses $\mathfrak{C}^{\mathrm{rrs}}(\psi, \chi, \mathcal{S})$ as well as a set of literals $\mathfrak{L}^{\mathrm{rrs}}(\psi, \chi, \mathcal{S})$. Initially, $\mathfrak{C}^{\mathrm{rrs}}(\psi, \chi, \mathcal{S})$ contains all clauses from χ and $\mathfrak{L}^{\mathrm{rrs}}(\psi, \chi, \mathcal{S})$ contains all $\mathcal{S}$-literals in χ. We expand these sets in the following way: suppose $p \in \mathfrak{L}^{\mathrm{rrs}}(\psi, \chi, \mathcal{S})$, we include any ψ clause E with $\bar{p} \in E$ to $\mathfrak{C}^{\mathrm{rrs}}(\psi, \chi, \mathcal{S})$ and include the literals $\{x \in E \mid x \neq \bar{p}, x \in \mathcal{S}\}$ to $\mathfrak{L}^{\mathrm{rrs}}(\psi, \chi, \mathcal{S})$. $\bar{p}$'s non-inclusion is why we cannot just list the clauses.

We define ϕ_u to be the set of clauses that contain u, and $\phi_{\bar{u}}$ to be those that contain $\bar{u}$, (assume no clauses are tautological). Let $\mathcal{S}_u$ be the set of existential variables that contain u in its dependency set. We say u has a reflexive resolution path to literal x (denoted $u \to_{\mathrm{rrs}} x$) if $x \in \mathfrak{L}^{\mathrm{rrs}}(\Pi\phi, \phi_u, \mathcal{S}_u)$, likewise $\bar{u} \to_{\mathrm{rrs}} x$ if $x \in \mathfrak{L}^{\mathrm{rrs}}(\Pi\phi, \phi_{\bar{u}}, \mathcal{S}_u)$. The purpose of $u \to_{\mathrm{rrs}} x$ is to identify that the existential player may be required to satisfy an x literal while dealing with a falsified u literal further back along the reflexive resolution path.

For the reflexive resolution path dependency scheme, $(u, x) \in \mathcal{D}^{\mathrm{rrs}}(\Pi\phi)$ if $u \in \mathrm{D}^\Pi_x$ and either: $u \to_{\mathrm{rrs}} x$ and $\bar{u} \to_{\mathrm{rrs}} \bar{x}$, or $\bar{u} \to_{\mathrm{rrs}} x$ and $u \to_{\mathrm{rrs}} \bar{x}$.

We sometimes omit $\Pi\phi$ when the DQBF is clear i.e. $(u, x) \in \mathcal{D}^{\mathrm{rrs}}$. The reader should interpret membership $(u, x) \in \mathcal{D}^{\mathrm{rrs}}$ as dependence, and $(u, x) \notin \mathcal{D}^{\mathrm{rrs}}$ as independence.

2.3.3 DQBF Proof Systems

While Q-Res has been proven incomplete for DQBFs it is still sound [1], so we can use parts of Q-Res in a refutation. In order to make it complete, so that every false DQBF has a refutation leading to the empty clause, Chew and Peitl [18] discovered an extension rule that created new variables with minimal dependency sets. This powerful calculus is known as IndExtQURes and is given in Figure 1.

$$\frac{}{L}\ (\text{Ax}) \qquad \frac{C \vee u}{C}\ (\text{Red}) \qquad \frac{E \vee \neg x \qquad F \vee x}{E \vee F}\ (\text{Res})$$

L is a clause in the propositional matrix ϕ. u is a $\forall$ literal. There is no $\exists$ literal l in C such that $\text{var}(u) \in \text{D}^{\Pi}_{\text{var}(l)}$, and there is no $\bar{u} \in C$.

$$\frac{}{(\bar{\alpha} \vee v \vee y_1), (\bar{\alpha} \vee v \vee y_2), (\bar{\alpha} \vee \bar{v} \vee \bar{y}_1 \vee \bar{y}_2)}\ (\text{IndExt})$$

v is a fresh $\exists$ variable, α is a conjunction of $\forall$ literals. $\text{D}^{\Pi}_{v} = (\text{D}^{\Pi}_{y_1} \cup \text{D}^{\Pi}_{y_2}) \setminus \text{D}^{\Pi}_{\alpha}$.

As an additional rule, the prefix Π may be weakened to Π' to add a new variable.

Figure 1 Proof rules of IndExtQURes.

▶ **Example 2.** Consider the following DQBF from [1, Theorem 7]:

$$\forall u \forall v\ \exists x(u) \exists y(v)\ (u \vee x \vee y)\ \wedge\ (\bar{u} \vee \bar{v} \vee x \vee y)\ \wedge\ (\bar{u} \vee v \vee x \vee \bar{y})$$
$$\wedge\ (u \vee \bar{x} \vee \bar{y})\ \wedge\ (\bar{u} \vee \bar{v} \vee \bar{x} \vee \bar{y})\ \wedge\ (\bar{u} \vee v \vee \bar{x} \vee y)$$

We can refute it by first adding the three clauses: $(\bar{u} \vee n \vee x), (\bar{u} \vee n \vee x), (\bar{u} \vee \bar{n} \vee \bar{x} \vee \bar{x})$ using the IndExt rule, although since we are only using one argument x, two of these clauses are identical and one can be simplified with idempotence. We can interpret the clauses as conditionally defining n, $(u \to (n = \bar{x}))$, because n is defined only when u is true, it can without penalty assume any value when u is false. This allows IndExtQURes to soundly let n not depend on u despite x doing so, therefore $\text{D}^{\Pi}_{n} = \{\}$.

We can resolve $(\bar{u} \vee n \vee x)$ with two clauses to get $(\bar{u} \vee \bar{v} \vee n \vee \bar{y})$ and $(\bar{u} \vee v \vee n \vee y)$ which reduce to $(\bar{v} \vee n \vee \bar{y})$ and $(v \vee n \vee y)$. Likewise, we can resolve our other extension clause $(\bar{u} \vee \bar{n} \vee \bar{x})$ with two axiom clauses to get $(\bar{u} \vee \bar{v} \vee \bar{n} \vee y)$ and $(\bar{u} \vee v \vee \bar{n} \vee \bar{y})$ which become $(\bar{v} \vee \bar{n} \vee y)$ and $(v \vee \bar{n} \vee \bar{y})$ after reduction. We show the remaining derivation as a DAG to the empty clause in Figure 2. Note that the rules of Q-Res have been shown to be inadequate to derive a contradiction in these formulas [1].

In reality, finding the optimal uses of the IndExt rule requires a high degree of non-determinism. IndExtQURes is a DQBF generalisation of Extended Resolution in propositional logic, which similarly allows the addition of extension clauses as a rule. In propositional logic most checkers have progressed from an extension clause checker to a RAT addition checker, so we are interested in the one RAT based DQBF proof system DQRAT.

DQRAT is a combination of rules that have complicated checking criteria (Figure 3). The attempt here is to generalise what was practically checked in propositional proofs, but we have to bring in both the DQBF notion of outer clauses from Section 2.3.1 and the reflexive resolution path from Section 2.3.2.

▶ **Example 3.** Once again we should look at the DQBF from Example 2. We will first use BPM (Basic Prefix Modification) to create a new $\exists$ variable n such that $\text{D}^{\Pi}_{n} = \{u\}$. In terms

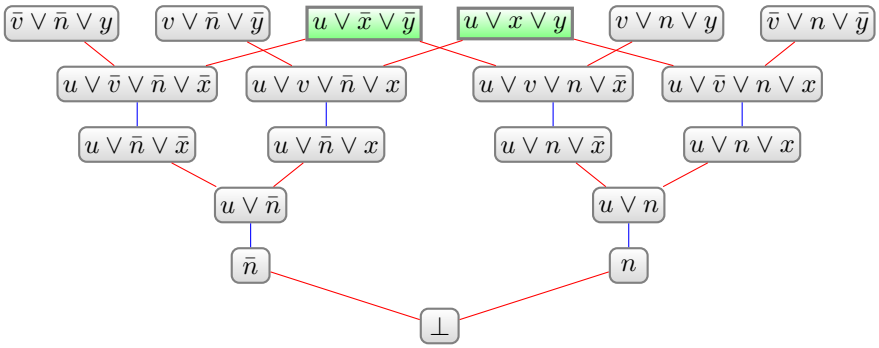

Figure 2 Proof DAG of the final resolution and reduction steps in Example 2 and Example 3.

of outer variables, n is in the same equivalence class as x and has outer variables $\{u, x, n\}$. We can trivially add clause $(n \vee x)$ via DQRAT$_\exists$ as no clause contains $\bar{n}$. We can then add $(\bar{n} \vee \bar{x})$ via DQRAT$_\exists$. This time it has to check against one clause with unit propagation but an easy contradiction between x and $\bar{x}$ exists on the left hand side so this is immediate.

Now we add the 4 clauses $(\bar{u} \vee \bar{v} \vee n \vee \bar{y})$, $(\bar{u} \vee v \vee n \vee y)$, $(\bar{u} \vee \bar{v} \vee \bar{n} \vee y)$ and $(\bar{u} \vee v \vee \bar{n} \vee \bar{y})$ which all can be done via ATA. The next part is important because it diverges from the proof in IndExtQURes. We need to delete the clauses $(n \vee x)$, $(\bar{n} \vee \bar{x})$. Only then can we apply the $\mathcal{D}^{\text{rrs}}$ rule. Since $u \not\to_{\text{rrs}} n$ and $u \not\to_{\text{rrs}} \bar{n}$, we can change the dependency set of n from $\{u\}$ to the empty set. After this point we can reduce the 4 clauses we added via ATA to get $(\bar{v} \vee n \vee \bar{y})$, $(v \vee n \vee y)$, $(\bar{v} \vee \bar{n} \vee y)$ and $(v \vee \bar{n} \vee \bar{y})$. Since ATA can be used to add instances of the resolution rules we can proceed to follow the same proof as in the end of the IndExtQURes proof (Figure 2). A machine readable proof of this formula is available at `https://ac.tuwien.ac.at/files/peitl/sat2026/`.

IndExtQURes and DQRAT are DQBF proof systems and can be compared via p-simulation. We say a proof system f *p-simulates* a proof system g if there is a polynomial time procedure that maps g proofs to f proofs of the same theorem. A p-simulation is impossible if there is a separating family of theorems, whose minimum proof size in f is bounded below by a super-polynomial function in the minimum proof size in g. IndExtQURes p-simulates DQRAT, but the converse is an open problem.

3 A New Dependency Scheme

In this section, we define an improvement on $\mathcal{D}^{\text{rrs}}$, which we call the *pure universal dependency scheme*. The idea is that we can exclude a resolution path from u to x if it is merely an extension of a resolution path from $\bar{u}$ to x, as only the minimal path should be considered.

3.1 Definition of the Pure Universal Dependency Scheme

For a DQBF $\Pi\phi$, we will denote membership $(u, x) \in \mathcal{D}^{\forall\text{pure}}(\Pi\phi)$ to mean that $\exists$-variable x really does depend on $\forall$ variable u after considering the pure universal dependency scheme.

▶ **Definition 4.** *Let ψ be a DQBF, χ a subset of the clauses in ψ, $\mathcal{S}$ a subset of variables appearing in ψ and u a universal literal. We define the sets $\mathfrak{C}^{\forall\text{pure}}(u, \psi, \chi, \mathcal{S})$ and $\mathfrak{L}^{\forall\text{pure}}(u, \psi, \chi, \mathcal{S})$ as follows. Initially $\mathfrak{C}^{\forall\text{pure}}(u, \psi, \chi, \mathcal{S})$ contains all clauses from χ and $\mathfrak{L}^{\forall\text{pure}}(u, \psi, \chi, \mathcal{S})$ contains all $\mathcal{S}$-literals in χ. We expand these sets in a similar way as*

In all rules, let Π, Ω be DQBF prefixes, ϕ be a CNF, C be a clause and l be a literal with $\text{var}(\phi), \text{var}(C), \text{var}(l)$ assumed to be subsets of $\text{var}(\Pi)$.

$$\frac{\Pi\phi}{\Pi\phi \wedge C}\ (\text{ATA}) \qquad \frac{\Pi\phi \wedge C}{\Pi\phi}\ (\text{Del}) \qquad \frac{\Pi\phi \wedge (C \vee l)}{\Pi\phi \wedge C}\ (\text{UR})$$

ATA: $\phi \wedge \bar{C} \vdash_1 \bot$ is required. **Del**: there is no side condition on C.
UR: we require that l is universal and $\text{var}(l) \notin \text{D}^{\Pi}_{C}$.

$$\frac{\Pi\phi}{\Pi\phi \wedge (C \vee l)}\ (\text{DQRAT}_{\exists}) \qquad \frac{\Pi\phi \wedge (C \vee l)}{\Pi\phi \wedge C}\ (\text{DQRAT}_{\forall})$$

DQRAT: l is existential in $\text{DQRAT}_{\exists}$, universal in $\text{DQRAT}_{\forall}$. For all clauses D in ϕ with $\bar{l} \in D$, the following must hold:
($\exists$): $\phi \wedge \neg C \wedge \bar{l} \wedge \bigwedge_{x \in D, x \neq \bar{l}}^{\text{var}(x) \lesssim_{\Pi} \text{var}(l)} \bar{x} \vdash_1 \bot$. ($\forall$): $\phi \wedge \neg C \wedge l \wedge \bigwedge_{x \in D, x \neq \bar{l}}^{\text{var}(x) \lesssim_{\Pi} \text{var}(l)} \bar{x} \vdash_1 \bot$.

$$\frac{\Pi\phi}{\Omega\phi}\ (\text{BPM}) \qquad \frac{\Pi\phi}{\Omega\phi}\ (\mathcal{D}^{\text{rrs}})$$

BPM: $\Pi \subseteq \Omega$.
$\boldsymbol{\mathcal{D}^{\text{rrs}}}$: $\text{var}_{\exists}(\Pi) = \text{var}_{\exists}(\Omega)$, $\text{var}_{\forall}(\Pi) = \text{var}_{\forall}(\Omega)$, $u \notin \text{D}^{\Omega}_{x}$ only if $(u, x) \notin \mathcal{D}^{\text{rrs}}(\Pi\phi)$.

Figure 3 Proof rules of DQRAT [15].

before, suppose $p \in \mathfrak{L}^{\forall\text{pure}}(u, \psi, \chi, \mathcal{S})$ *we include any* ψ *clause* E *with* $\bar{p} \in E$ ***and crucially*** $\bar{u} \notin E$ *into* $\mathfrak{L}^{\forall\text{pure}}(u, \psi, \chi, \mathcal{S})$*, and also include the literals* $\{x \in E \mid x \neq \bar{p}, x \in \mathcal{S}\}$ *into* $\mathfrak{L}^{\forall\text{pure}}(u, \psi, \chi, \mathcal{S})$*, and as we increase this set we can further propagate until we reach fix-point.*

▶ **Definition 5.** *Given a DQBF* $\Pi\phi$ *with CNF matrix* ϕ*. We say there is a pure path from universal literal* u *to existential literal* x *(denoted* $u \to_{\forall\text{pure}} x$*) if and only if* $x \in \mathfrak{L}^{\forall\text{pure}}(u, \psi, \phi_u, \mathcal{S}_u)$*. Where* ϕ_u *is the set of clauses that contain literal* u *and* $\mathcal{S}_u$ *is the set of* $\exists$ *variables that contain* u *in its dependency set.*

▶ **Definition 6.** *Given a DQBF* $\Pi\phi$ *with CNF matrix* ϕ*, let* (u, x) *be a pair with a* $\forall$ *variable* u *and* $\exists$ *variable* x*.* $(u, x) \in \mathcal{D}^{\forall\text{pure}}(\Pi\phi)$ *(*$\in \mathcal{D}^{\forall\text{pure}}$ *when unambiguous) if and only if* $u \in \text{D}^{\Pi}_{x}$ *and either:* ***1.*** $u \to_{\forall\text{pure}} x$ *and* $\bar{u} \to_{\forall\text{pure}} \bar{x}$*, or* ***2.*** $\bar{u} \to_{\forall\text{pure}} x$ *and* $u \to_{\forall\text{pure}} \bar{x}$*.*

▶ **Example 7.** Consider the following QBF:

$$\forall u \forall v \exists x \exists y \exists z\ (u \vee v \vee y)\ \wedge\ (u \vee \bar{v} \vee x \vee \bar{y})\ \wedge\ (\bar{u} \vee v \vee z)\ \wedge\ (\bar{u} \vee \bar{v} \vee \bar{x} \vee \bar{z}).$$

We can first make some observations that are true for both $\mathcal{D}^{\text{rrs}}$ and for $\mathcal{D}^{\forall\text{pure}}$. $(v, x) \notin \mathcal{D}^{\forall\text{pure}}$ and $(v, x) \notin \mathcal{D}^{\text{rrs}}$ as $v \not\to_{\text{rrs}} x$ nor $v \not\to_{\text{rrs}} \bar{x}$, which means $v \not\to_{\forall\text{pure}} x$ nor $v \not\to_{\forall\text{pure}} \bar{x}$, it is always the case that $(v, x) \notin \mathcal{D}^{\text{rrs}}$ entails $(v, x) \notin \mathcal{D}^{\forall\text{pure}}$. However $(v, y), (v, z) \in \mathcal{D}^{\forall\text{pure}}$ as we can make immediate paths, likewise for $(u, x) \in \mathcal{D}^{\forall\text{pure}}$.

Where $\mathcal{D}^{\forall\text{pure}}$ differs from $\mathcal{D}^{\text{rrs}}$ can be seen for (u, y) and (u, z). (u, y) is in $\mathcal{D}^{\text{rrs}}$ because $u \to_{\text{rrs}} y$ and $\bar{u} \to_{\text{rrs}} \bar{y}$ (through $\bar{x}$, first). However $\bar{u} \not\to_{\forall\text{pure}} \bar{y}$ as the path from $\bar{u}$ cannot use $(u \vee \bar{v} \vee x \vee \bar{y})$ as it contains a positive u. We observe a similar situation for (u, z) where $u \to_{\text{rrs}} \bar{z}$ but as the path must go through $(\bar{u} \vee \bar{v} \vee \bar{x} \vee \bar{z})$, $u \not\to_{\forall\text{pure}} \bar{z}$.

3.2 The $\mathcal{D}^{\forall\text{pure}}$ Prefix Modification Rule

Given a universal literal u, we can calculate the set of literals $\mathfrak{L}^{\forall\text{pure}}(u, \psi, \chi, \mathcal{S})$ in linear time in the total number of individual literals appearing in ψ. Once we have found the variables that lack sufficient paths for $\mathcal{D}^{\forall\text{pure}}$, we can subtract u from their dependency sets, and this will not affect the process if we repeat it for any different universal literals. Therefore it takes polynomial time in ψ to completely recalculate the prefix according to $\mathcal{D}^{\forall\text{pure}}$ and can be considered a polynomial-time checkable proof system. We prove it sound in this section.

▶ **Example 8.** We can take the QBF from Example 7 and modify its prefix according to $\mathcal{D}^{\forall\text{pure}}$ to get the following DQBF:

$$\forall u \forall v\ \exists x(u) \exists y(v) \exists z(v)\ (u \vee v \vee y)\ \wedge\ (u \vee \bar{v} \vee x \vee \bar{y})\ \wedge\ (\bar{u} \vee v \vee z)\ \wedge\ (\bar{u} \vee \bar{v} \vee \bar{x} \vee \bar{z}).$$

For the proof of Theorem 9 recall that for a total universal assignment β and a set of Skolem functions f we define the completed assignment $\beta \cup f(\beta)$ by

$$\big[\beta \cup f(\beta)\big](x) = \begin{cases} \beta(x) & x \in \text{var}_\forall, \\ f_x(\beta|_{\mathrm{D}_x^\Pi}) & x \in \text{var}_\exists, \end{cases}$$

in other words, the assignment that is β on universal variables and computes a value according to the respective Skolem function for existential variables.

▶ **Theorem 9** (Soundness). *Let $\Pi\phi$ be a true DQBF. Let Π' be the prefix where $\text{var}_\exists(\Pi) = \text{var}_\exists(\Pi')$, $\text{var}_\forall(\Pi) = \text{var}_\forall(\Pi')$, $u \in \mathrm{D}_x^{\Pi'}$ if and only if $(u, x) \in \mathcal{D}^{\forall\text{pure}}(\Pi\phi)$. Then $\Pi'\phi$ is true.*

Proof. Let $f = \{f_e \mid e \in \text{var}_\exists(\Pi)\}$ be a set of Skolem functions that satisfies $\Pi\phi$. We will construct a set of Skolem functions $f^* = \{f_e^* \mid e \in \text{var}_\exists(\Pi)\}$ that satisfies $\Pi'\phi$.

We use the following definitions. If α is partial assignment to the universal variables, α^v is obtained by flipping the Boolean value of universal variable v. Given f a *dependency witness* is a triple (α, y, v) where $f_y(\alpha) \neq f_y(\alpha^v)$, y is an existential variable, v is a universal variable, α is an assignment to D_y^Π. The set $\mathcal{S}_u = \{z : u \in \mathrm{D}_z^\Pi\}$. For universal literal l: $\phi_l = \{C \in \phi \mid l \in C\}$. For an existential variable z, and assignment $\gamma \in \{0,1\}^U$: $\gamma_z = \gamma|_{\mathrm{D}_z^\Pi}$.

Suppose f has a dependency witness (α, x, u) where $u \in \mathrm{D}_x^\Pi \setminus \mathrm{D}_x^{\Pi'}$. Let l_u be the literal on u satisfied by α, l_x the literal on x satisfied by $f_x(\alpha)$. Since $u \notin \mathrm{D}_x^{\Pi'}$, by definition either $\bar{l}_u \not\to_{\forall\text{pure}} l_x$ or $l_u \not\to_{\forall\text{pure}} \bar{l}_x$. Without loss of generality, let $\neg l_u \not\to_{\forall\text{pure}} l_x$ (in the other case, swap the roles of α and α^u).

Define $\mathcal{M} = \mathcal{M}(f, \bar{l}_u, l_x, \alpha)$ as the set of all candidate Skolem sets $f' = \{f'_x \mid x \in \text{var}_\exists(\Pi)\}$ respecting the prefix Π with the following properties:

1. The set of dependency witnesses in f' and u are a *proper* subset of dependency witnesses in f and u.
2. For every existential variable z, for every $\gamma_z \in \{0,1\}^{\mathrm{D}_z^\Pi}$, if $f'_z(\gamma_z) \neq f_z(\gamma_z)$ then γ_z satisfies l_u; i.e. $u \in \mathrm{D}_z^\Pi$ and l_u's polarity matches with γ_z.
3. For every existential variable z, for every $\gamma_z \in \{0,1\}^{\mathrm{D}_z^\Pi}$, if $f'_z(\gamma_z) \neq f_z(\gamma_z)$ then for the literal l_z satisfied by $f_z(\gamma_z)$, $\neg l_u \not\to_{\forall\text{pure}} l_z$.

We first show that $\mathcal{M}$ is non-empty. Let

$$f^0 = \{f_z^0 \mid z \in \underset{\exists}{\text{var}}(\Pi)\} \quad f_z^0(\tau) = \begin{cases} \neg f_x(\tau) & \text{if } \tau = \alpha \text{ and } z = x \\ f_z(\tau) & \text{otherwise,} \end{cases}$$

1. f^0 satisfies property 1 because (α, x, u) and (α^u, x, u) are dependency witnesses that are removed. For any dependency witness (δ, w, u) present in f^0 then $w \neq x$ or ($\delta \neq \alpha$ and $\delta \neq \alpha^u$) and so $f_w^0(\delta) = f_w(\delta)$ and $f_w^0(\delta^u) = f_w(\delta^u)$ so (δ, w, u) is a witness in f.

2. f^0 only differs from f on α and its extensions, but α satisfies l_u.
3. f^0_z only differs from f_z when $z = x$ however we have assumed that $\neg l_u \not\to_{\forall\text{pure}} l_x$ and $f_x(\alpha)$ satisfies l_x.

Next we create an inductive step. We show that if $f' \in \mathcal{M}$ is not a model there is $f'' \in \mathcal{M}$ where the dependency witnesses of f'' and u are a proper subset of the witnesses of f'.

First we need to construct such a f''. We know that if f' is not a model of $\Pi\phi$, then there is some assignment $\gamma \in \{0,1\}^{\text{var}_\forall(\Pi)}$ and some clause $C \in \phi$ such that the assignment $\gamma \cup f'(\gamma)$ falsifies C. However we know that $\gamma \cup f(\gamma)$ satisfies C because f is a model. Hence there is some existential variable z such that $f'_z(\gamma_z) \neq f_z(\gamma_z)$ and some literal l_z in z such that $f_z(\gamma_z)$ satisfies l_z and $f'_z(\gamma_z)$ falsifies l_z. By property 2: γ_z satisfies l_u, and by property 3: $\bar{l}_u \not\to_{\forall\text{pure}} l_z$. On γ^u, we have that l_u is falsified so f' and f are equal by property 2. This means that $f'_z(\gamma^u_z) = f_z(\gamma^u_z)$. Therefore if $f_z(\gamma_z) \neq f_z(\gamma^u_z)$ (i.e. (γ_z, z, u) is not a dependency witness in f) then $f'_z(\gamma_z) \neq f_z(\gamma_z) = f_z(\gamma^u_z) = f'_z(\gamma^u_z)$ (i.e. (γ_z, z, u) is a dependency witness in f') violating property 1, so $f_z(\gamma^u_z) \neq f_z(\gamma^u_z)$ and $f'_z(\gamma_z) = f'_z(\gamma^u_z)$.

This means there is some other literal $l_y \in C$ with variable y such that it is satisfied in $\gamma^u \cup f(\gamma^u)$. As f and f' must agree on γ^u, l_y is also satisfied by $\gamma^u \cup f'(\gamma^u)$. $\gamma \cup f(\gamma)$ does not satisfy l_y as it does not satisfy C. Therefore the only $\forall$ variable y can be is u, however l_y cannot be l_u, otherwise $\gamma \cup f(\gamma)$ would satisfy this (we come back to this for pure paths), and l_y cannot be $\neg l_u$ otherwise we would have $\neg l_u \not\to_{\forall\text{pure}} l_z$. Therefore y is existential. Let

$$f'' = \{f''_w \mid w \in \underset{\exists}{\text{var}}(\Pi)\} \quad f''_w(\tau) = \begin{cases} \neg f'_w(\tau) & \text{if } \tau = \gamma_y \text{ and } w = y \\ f'_w(\tau) & \text{otherwise,} \end{cases}$$

we show that $f'' \in \mathcal{M}$:

1. (+ smaller than f') As $\gamma \cup f(\gamma)$ falsifies l_y and $\gamma \cup f(\gamma^u)$ satisfies l_y, (γ_y, y, u) and (γ^u_y, y, u) are dependency witnesses for f' and therefore must be for f by the induction hypothesis. $f''_y(\gamma_y) = f''_y(\gamma^u_y)$ and thus these dependency witnesses are removed for f''. For any dependency witness (δ, w, u) in f'', $w \neq y$ or ($\delta \neq \gamma_y$ and $\delta \neq \gamma^u_y$), and so $f''_w(\delta) = f'_w(\delta)$ and $f''_w(\delta^u) = f'_w(\delta^u)$ so (δ, w, u) is a dependency witness in f' and thus also in f.
2. f'' only differs from f' on γ_y and its extensions. Since (γ_y, y, u) is a dependency witness for f', then $u \in D(y)$, and since it is consistent with γ then γ_y satisfies l_u.
3. f''_w only differs from f'_w when $w = y$. We know that (γ_y, y, u) is a dependency witness for f, so as $f_y(\gamma^u_y)$ satisfies l_y then $f_y(\gamma_y)$ satisfies $\bar{l}_y$. Assume for contradiction, that $\neg l_u \to_{\forall\text{pure}} \neg l_y$. We know that if l_u was in C then γ would satisfy C which is does not (this is the only observation needed to get from $\mathcal{D}^{\text{rrs}}$ to $\mathcal{D}^{\forall\text{pure}}$). Hence extending $\neg l_u \to_{\forall\text{pure}} \neg l_y$ through the positive l_y to l_z gives us a $\neg l_u$ pure path from $\neg l_u$ to l_z giving us a contradiction.

Since the set of dependency witnesses is finite the induction step eventually terminates with model where (α, x, u) is no longer a dependency witness, and we have strictly fewer dependency witnesses on u. We can repeat this for any other witness (α', x, u) until all witnesses of (α', x, u) are removed to get model f^*. We can restrict the domain of f^*_x to remove u and it will still be well defined and a winning Skolem function. ◀

We note that a known dependency witness (α, x, u) can be eliminated in polynomial time (there is at most one flip per existential variable). On the other hand, checking whether a given model f has a forbidden dependency witness is NP-complete.

▶ **Theorem 10.** *Given a true (D)QBF Φ and its model f, it is NP-complete to decide if f has a dependency witness (α, x, u) for universal u and existential x with $(u, x) \notin \mathcal{D}^{\forall\text{pure}}(\Phi)$.*

Proof. By reduction from SAT. Take a CNF $\phi(V)$ in the variables V and construct the true DQBF Φ with prefix $\forall V \cup \{u\} \exists x(V \cup \{u\})$ where $\{u, x\} \cap V = \emptyset$ and matrix with the single clause $\bigvee_{v \in V} v \vee u \vee \bar{x}$, with the Skolem function $f_x = \phi(V) \wedge u$. f_x is a model, since whenever u is set to false, f_x yields 0 satisfying the only clause. $(u, x) \notin \mathcal{D}^{\forall\text{pure}}(\Phi)$ since the literal x does not occur in Φ at all. Yet for any universal assignment α, (α, u, x) is a dependency witness for f_x if and only if $\alpha|_V$ is a satisfying assignment to ϕ. ◀

In addition to the prefix modification rule, we also have found a reduction rule that has $\mathcal{D}^{\forall\text{pure}}$ inbuilt in it. The rule and its proven properties are omitted for space reasons.

3.3 Strategy Extraction in Long Distance Q-Resolution

Theorem 9 establishes semantic soundness of $\mathcal{D}^{\forall\text{pure}}$: as a DQBF mapping it preserves both falsity (trivially) and truth (Theorem 9). This means that $\mathcal{D}^{\forall\text{pure}}$ can be soundly used in any DQBF proof system, and in fact the dependency scheme is not even 'used in' the proof system any more, it operates entirely outside of it. An important case not directly addressed by Theorem 9 is that of *long-distance Q-resolution (LD-Q-Res)* [2]. LD-Q-Res is a sound proof system for QBF and is supported by the solvers `DepQBF` [34] and `Qute` [38], but it is not sound for DQBF [5], so from Theorem 9 alone we cannot infer that LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res is sound (we give the definition of long-distance Q-resolution with a dependency scheme $\mathcal{D}$ in Figure 4 and this works for $\mathcal{D} = \mathcal{D}^{\forall\text{pure}}$; for long distance Q-resolution without a dependency scheme we take $\mathcal{D} = \mathcal{D}^{\text{trv}}$, the trivial dependency scheme, where $(u, x) \in \mathcal{D}^{\text{trv}}(\Pi\phi)$ if and only if $u \in \mathrm{D}_x^{\Pi}$). Fortunately, Theorem 8 from [4] addresses precisely this: it shows that LD-Q($\mathcal{D}$)-Res is sound when $\mathcal{D}$ preserves DQBF truth value (the technical term for this used in [4] is 'full exhibition').

▶ **Theorem 11.** *LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res is a sound and complete proof system for QBF.*

$$\frac{}{L}\ (\text{Ax}) \qquad \frac{C \vee u}{C}\ (\text{Red})$$

L is a clause in the propositional matrix ϕ. u is a $\forall$ literal. There is no $\exists$ literal l in C such that $(u, l) \in \mathcal{D}(\Pi\phi)$. In contrast to Figure 1, there **may** be $\bar{u} \in C$.

$$\frac{E \vee \neg x \qquad F \vee x}{E \vee F}\ (\text{Res})$$

There is no $\forall$ literal v in E such that $\bar{v} \in F$ and $(v, x) \in \mathcal{D}(\Pi\phi)$.

Figure 4 Proof rules of LD-Q($\mathcal{D}$)-Res applied to an input QBF $\Pi\phi$ for dependency scheme $\mathcal{D}$.

It can also be shown using methods from [39] that LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res admits polynomial-time strategy extraction. The proof follows the proof outline of [39, Theorem 2]. In particular, one can show that $\mathcal{D}^{\forall\text{pure}}$ is a *normal* dependency scheme. A dependency scheme $\mathcal{D}$ is normal [39, Definition 7] if any LD-Q($\mathcal{D}$)-Res refutation of a QBF with outermost universal variables contains outermost variables in at most one polarity (and additionally the proofs are closed under application of partial existential assignments in a natural way). This unique polarity prescribes the winning strategy for the outermost variables, and this idea can be captured in a polynomial-size circuit, yielding both soundness and strategy extraction for LD-Q($\mathcal{D}$)-Res for normal $\mathcal{D}$ [39, Theorem 1]. We omit the proof of normality of $\mathcal{D}^{\forall\text{pure}}$ here and state only the result.

▶ **Theorem 12.** *There is a polynomial time algorithm that, given an LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res refutation of a QBF $\Pi\phi$, computes a strategy for the universal player.*

3.4 Separations

LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res trivially p-simulates LD-Q($\mathcal{D}^{\text{rrs}}$)-Res as all LD-Q($\mathcal{D}^{\text{rrs}}$)-Res refutations are in fact LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res refutations already. Not all LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res proofs are LD-Q($\mathcal{D}^{\text{rrs}}$)-Res proofs as some reduction and resolution steps could be prohibited and in fact we show that there is no workaround.

In QBF, the QParity formulas are a family of false formulas which require the $\forall$ player to play the parity function in order to win. The parity function's hardness on bounded-depth formulas usually translates [3, 14] to proof size lower bounds, but in some cases gadgets can be used to find short proofs depending on the proof system. Therefore, by tuning these gadgets we can use variations of QParity to separate different QBF proof systems.

▶ **Definition 13** (ts-LQParity(N))**.** *Let* $\text{xor}_l(o_1, o_2, o, z)$ *be the set of clauses* $\{(z \vee \neg o_1 \vee \neg o_2 \vee \neg o), (z \vee o_1 \vee o_2 \vee \neg o), (z \vee \neg o_1 \vee o_2 \vee o), (z \vee o_1 \vee \neg o_2 \vee o)\}$

$$\exists x_1, \ldots, x_N \; \forall z \; \exists t_2, \ldots, t_N, s_2, \ldots, s_N. \quad \bigwedge \text{xor}_l(x_1, x_2, t_2, z) \wedge \bigwedge_{i=3}^{N} \text{xor}_l(t_{i-1}, x_i, t_i, z)$$

$$\wedge \bigwedge \text{xor}_l(x_1, x_2, s_2, \neg z) \wedge \bigwedge_{i=3}^{N} \text{xor}_l(s_{i-1}, x_i, s_i, \neg z) \wedge (z \vee t_N) \wedge (\neg z \vee \neg s_N).$$

▶ **Lemma 14.** *The shortest LD-Q-Res refutations of ts-*LQParity*(N) are Q-Res refutations.*

Proof. At the beginning of the proof, every clause contains a z or $\bar{z}$ literal block by some inner existential literal. Every derived clause in LD-Q-Res that contains an inner existential literal must therefore contain a $z, \bar{z}$ literal or both. This should be intuitive as universal literals linger unless they can be reduced which they cannot in the presence of these inner existentials. In addition any derived clause that contains a $z, \bar{z}$ or both without a blocking existential literal can be immediately reduced without cost to the proof size, so $z, \bar{z}$ literals and inner existential literals can only coincide in reduced clauses. For more details follow the argument from [14]. As argued in [14], long distance (merge) steps are possible, but can never be reduced as this must be done after resolving the inner existential literal. Resolving the inner existential literal cannot be done because it will always be an illegal merge step. ◀

▶ **Lemma 15.** *The shortest LD-Q-Res refutations of ts-*LQParity*(N) are exponential in N.*

Proof. Citing [14, Theorem 26] we use the well established strategy extraction lower bound technique. The winning universal strategy extracted from a Q-Res proof is always a bounded-depth circuit. In this case the winning strategy for z is the parity function on $x_1 \ldots x_n$. Parity has exponential lower bounds in bounded-depth circuits [20, 21], therefore since the strategy extraction was done in polynomial time in the size of the proof, the proofs must be at least exponential size. The shortest LD-Q-Res proofs are the shortest Q-Res proofs, so these are exponentially bounded below as well. ◀

We can observe that actually ts-LQParity(N) will not be a hard problem when using the proof systems of LD-Q($\mathcal{D}^{\text{rrs}}$)-Res or LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res because $\mathcal{D}^{\text{rrs}}$ and $\mathcal{D}^{\forall\text{pure}}$ are empty and the problem reduces to the SAT benchmark `Dubois` which is a known easy family of formulas. While we could use this as a separating example between LD-Q-Res and LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res, we can do better and add a gadget so that it also becomes a separating family between LD-Q($\mathcal{D}^{\text{rrs}}$)-Res and LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res.

▶ **Definition 16** (Bridged ts-LQPARITY)**.**

$$\exists x_1, \ldots, x_N\ \forall z\ \exists t_2, \ldots, t_N,\ \exists s_2, \ldots, s_N,\ \exists b\ (\textit{all clauses from ts-LQPARITY}) \wedge \\ (z \vee \neg t_N \vee b) \wedge (z \vee t_N \vee \neg b) \wedge (\neg z \vee \neg s_N \vee b) \wedge (\neg z \vee s_N \vee \neg b)$$

The b variable must be equal to t_N when z is false and equal to s_N when z is true, and that is the only condition needed to satisfy these clause. Making this modification does not change the proofs of Lemmas 14 and 15. But previously there were no resolution paths between t-variables and s-variables. But now b acts as a bridge.

▶ **Lemma 17.** *$\mathcal{D}^{\text{rrs}}$ and $\mathcal{D}^{\text{trv}}$ are equivalent on Bridged ts-*LQPARITY*.*

Proof.
Induction hypothesis (on i): $z \to_{\text{rrs}} t_{N-i}$ and $\neg z \to_{\text{rrs}} \neg t_{N-i}$.
Base case: $z \vee t_N$ is an axiom. $(\neg z \vee s_N \vee \neg b)$ and $(z \vee \neg t_N \vee b)$ link $\neg z$ to $\neg b$ and to $\neg t_N$.
Induction step: We can extend a path from t_{N+1-i} to t_{N-i} and from $\bar{t}_{N+1-i}$ to $\bar{t}_{N-i}$ using the clauses of $\text{xor}_l(\bar{t}_{N-i}, x_{N+1-i}, t_{N+1-i}, z)$.
We can symmetrically do the same induction for the s_i variables. Finally, although it is not necessary for hardness, b appears in an axiom with z and $\neg b$ appears with $\neg z$. ◀

▶ **Corollary 18.** *LD-Q($\mathcal{D}^{\text{rrs}}$)-Res requires exponential-size proofs of Bridged ts-*LQPARITY$_N$*.*

▶ **Lemma 19.** *There are short refutations of Bridged ts-*LQPARITY$_N$ *in LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res.*

Proof. Every clause with a t variable contains a positive z literal. Every clause with an s variables contains a $\neg z$ literal. Therefore, for $2 \leq i \leq N$ there are no $\mathcal{D}^{\forall\text{pure}}$ resolution paths that go from $\neg z$ to t_i, from $\neg z$ to $\neg t_i$, from z to s_i nor from z to $\neg s_i$. In fact the only dependency pair in $\mathcal{D}^{\forall\text{pure}}$ is (z, b). In the derivation, with the exception of the new clause which we will not use anyway, we can reduce all z and $\neg z$ literals immediately. Now we have a false existential formula with a known short resolution proof. Starting with t_2 and s_2 we inductively derive $t_i \leftrightarrow s_i$ (as clauses $\neg t_i \vee s_i$ and $\neg s_i \vee t_i$). Once we reach $(\neg t_i \vee s_i)$ we can contradict this with t_i and $\neg s_i$. ◀

▶ **Corollary 20.** *LD-Q($\mathcal{D}^{\text{rrs}}$)-Res does not p-simulate LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res.*

4 P-Equivalence with IndExtQURes

When looking at proof complexity, IndExtQURes [18] has been proven to be very powerful relative to the other proof systems for both QBF and DQBF. IndExtQURes has been shown to p-simulate the majority of QBF and DQBF proof systems (see Fig 5 for QBF proof systems), despite IndExtQURes only using a small number of simple rules. The p-simulation of many QBF and DQBF techniques presents an opportunity to improve certification for both logics.

One way to do this is to adapt an existing certification format with new rules so that it p-simulates IndExtQURes, therefore transitively p-simulating most of the techniques in QBF and DQBF. Our main idea is that we can combine DQRAT with the $\mathcal{D}^{\forall\text{pure}}$ prefix modification rule. Doing so gives a proof system that we will show is p-equivalent to IndExtQURes. Recall the definition of DQRAT (Figure 3), we introduce $\mathcal{D}^{\forall\text{pure}}$ as a replacement of the $\mathcal{D}^{\text{rrs}}$ rule:

$$\frac{\Pi\phi}{\Omega\phi}(\mathcal{D}^{\forall\text{pure}})$$

Where Π and Ω are DQBF prefixes and ϕ is a CNF. The condition on Ω is that it contains the same variables as Π, with a modification of the dependency sets with the restriction that $u \notin \text{D}_x^\Omega$ only if $u \notin \text{D}_x^\Pi$ or $(u, x) \notin \mathcal{D}^{\forall\text{pure}}(\Pi\phi)$.

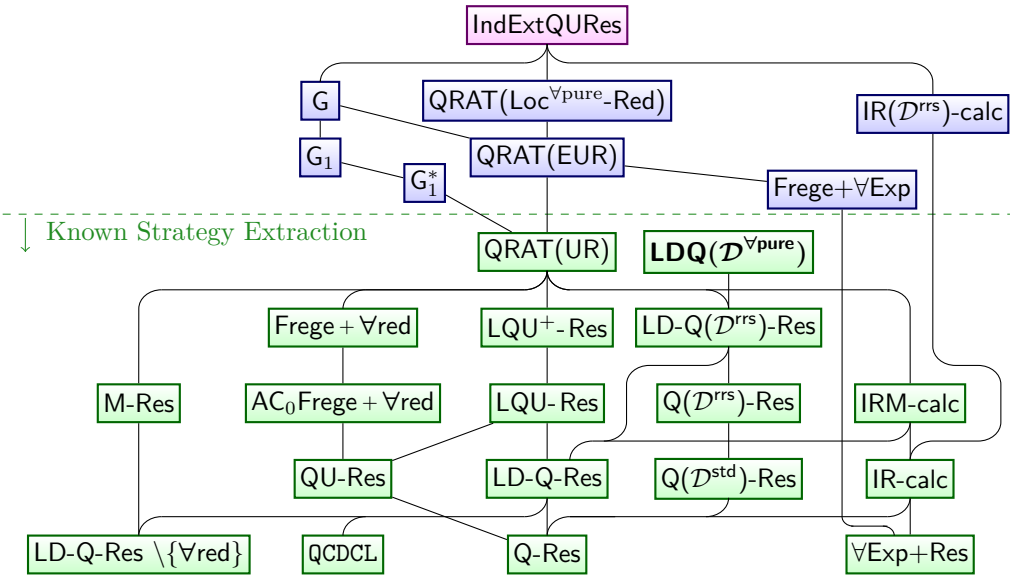

Figure 5 The p-simulation structure of refutational QBF proof systems [2, 3, 7, 10, 11, 14, 16, 17, 19, 18, 24, 29, 30, 31, 39, 41, 43, 44].

▶ **Definition 21.** *(The refutational version of) DQRAT+$\mathcal{D}^{\forall pure}$ is a proof system that allows proofs where each line is an S-form DQBF $\Pi\phi$. Refutation is shown in the same way as DQRAT. Each subsequent line follows from the previous by one of the seven rules: ATA, Del, UR, DQRAT$_\exists$, DQRAT$_\forall$, BPM and $\mathcal{D}^{\forall pure}$.*

First we show that DQRAT+$\mathcal{D}^{\forall pure}$ p-simulates IndExtQURes. This direction is the more important of the two if we want to use DQRAT+$\mathcal{D}^{\forall pure}$ for certification.

▶ **Theorem 22.** *DQRAT+$\mathcal{D}^{\forall pure}$ p-simulates* IndExtQURes.

Proof. We consider an IndExtQURes refutation π of $\Pi\phi$ as a sequence of clauses $C_1 \dots C_n$ with $C_n = \bot$. We will p-simulate π by creating a DQRAT+$\mathcal{D}^{\forall pure}$ derivation $L_1 \dots L_m$. Within that sequence there will be a subsequence $(L'_{f(i)})_{i=1}^n$ where $L'_{f(i)} = \Omega\psi$ and ψ contains clauses $C_1 \dots C_i$ as well as any clauses from ϕ and Ω quantifies all variables appearing in ψ, with the same quantifiers and dependency sets as in the IndExtQURes proof.

For the **(Ax)** rule we ensure that we keep all clauses from ϕ in ψ. For **adding variables to the prefix** we can use BPM. For the **(Red)** rule if we want to use clause $C \vee u$ to get C we use the UR rule in DQRAT. Technically we need to keep a copy of $C \vee u$ in ψ which can be returned by using ATA. For the **(Res)** rule we can use ATA to add any resolvent since if $D_1 \vee \bar{x}$ and $D_2 \vee x$ are in ψ then $\psi \wedge \neg D_1 \wedge \neg D_2$ is a propositional contradiction, furthermore we can derive it via reverse unit propagation as the units of $\neg D_1$ simplify $D_1 \vee \bar{x}$ to just $\bar{x}$ and the units of $\neg D_2$ simplify $D_2 \vee x$ to just x, we then propagate to the empty clause.

Suppose we add independent extension clauses $(\bar{\alpha} \vee n \vee a), (\bar{\alpha} \vee n \vee b), (\bar{\alpha} \vee \bar{n} \vee \bar{a} \vee \bar{b})$. $D_n^\Pi = D_a^\Pi \cup D_b^\Pi \setminus D_\alpha^\Pi$ using the **(IndExt)** rule. We first choose to add the existential variable n with $D_n^\Pi = D_a^\Pi \cup D_b^\Pi$ using BPM. Using DQRAT$_\exists$ we can add the first two clauses $(\bar{\alpha} \vee n \vee a), (\bar{\alpha} \vee n \vee b)$, provided n is a new variable. In order to add the final clause, we need the outer variables of $(\bar{\alpha} \vee n \vee a), (\bar{\alpha} \vee n \vee b)$ to each have some non n literal to be opposite of a literal in $(\bar{\alpha} \vee \bar{n} \vee \bar{a} \vee \bar{b})$, this can only be a and b. So in this case because we chose that $D_n^\Pi = D_a^\Pi \cup D_b^\Pi$, this is sufficient to add the final clause $(\bar{\alpha} \vee \bar{n} \vee \bar{a} \vee \bar{b})$.

Finally we need to remove the dependencies of n that are in α. Suppose we have a universal variable u and a literal l_u is a conjunct in the assignment α. All paths from l_u to n or $\bar{n}$ pass through $\bar{l}_u$ as our definition clauses are the only clauses that contain n and $\bar{n}$ at this point. Thus $(u, n) \notin \mathcal{D}^{\forall\text{pure}}$. Therefore we can drop $\text{var}(u)$ from D_n^Π. We can do this for each literal in α. At the end $\text{D}_n^\Pi = \text{D}_a^\Pi \cup \text{D}_b^\Pi \setminus \text{D}_\alpha^\Pi$. ◀

The reverse is true, that IndExtQURes p-simulates DQRAT+$\mathcal{D}^{\forall\text{pure}}$. For the original rules of DQRAT we already know how to do a p-simulation [18]. Only the $\mathcal{D}^{\forall\text{pure}}$ rule remains, however this ends up being similar enough to the p-simulation of the $\mathcal{D}^{\text{rrs}}$ rule.

▶ **Theorem 23.** IndExtQURes *p-simulates DQRAT+*$\mathcal{D}^{\forall\text{pure}}$.

5 Practical Implications

We took existing code from the QBF solver `Qute` [38] that detected $\mathcal{D}^{\text{rrs}}$ and naturally extended it to $\mathcal{D}^{\forall\text{pure}}$. Using this we developed in C++ the prototype DQRAT+$\mathcal{D}^{\forall\text{pure}}$ checker called `DQRAT-check` and we increased the capabilities of `Qute` to include $\mathcal{D}^{\forall\text{pure}}$.[1]

5.1 DQRAT+$\mathcal{D}^{\forall\text{pure}}$-check

`dqrat-check` accepts a `.dqdimacs` file as its first argument containing the DQBF or QBF, the second argument is a proof written in the `.dqrat` format. We designed the `dqrat` format to follow the `qrat` format as closely as possible. The only deviation is in how extension variables are introduced: in `qrat` they must be existential, and are quantified automatically based on other literals in the clause. In `dqrat`, the proof can explicitly declare any existential variable and its dependencies before used in a clause (if undeclared, we adopt the QRAT convention of taking all universal variables as dependencies).

No alphabetical symbols indicates clause addition using ATA or DQRAT$_\exists$. The `a` indicates a new $\forall$ variable. Unlike in `.dqdimacs`, `d` indicates clause deletion (for QRAT compatibility), which leaves `e` for introduction or modification of an existential variable. Each `e` line takes multiple integers, the first is the variable, and then subsequently are the $\forall$ variables to be added or removed from the dependency set. Negative integers indicate removal. It is possible to make an $\exists$ variable depend another $\exists$, so it will inherit the entire dependency set.

Like in QRAT, a line starting with `u` indicates a UR or DQRAT$_\forall$ rule, in each case reducing the first literal in the clause, which must be universal.

We evaluated `DQRAT-check` on formulas from this paper, as well as on proofs obtained via `Qute` on QBFEval 2022 benchmarks. To obtain DQRAT proofs out of `Qute`, we extended the LD-Q-Res checker `qrp2rup`[2] [37] to convert the LD-Q-Res proof produced by Qute to (D)QRAT.[3] Proof checking (`qrp2rup` + `DQRAT-check`) took 9% longer than solving on average (geometric mean over all 43 false instances that `Qute` could solve). Notably, `DQRAT-check` alone accounts for only 3% of the overall proof checking effort; 97% is proof extraction from the QRP trace and conversion into DQRAT, both done by `qrp2rup`.

This QBF-based scenario does not cover the full scope of what DQRAT can do; it does not use $\mathcal{D}^{\forall\text{pure}}$ and DQRAT$_\forall$ (not needed for our translation from LD-Q-Res to DQRAT). We further tested `DQRAT-check` on crafted formulas: the formula from Example 2, the bridged

[1] Available at `https://github.com/peitl/dqrat-check` and `https://github.com/fslivovsky/qute`.
[2] Available at `https://github.com/peitl/qrp2rup`.
[3] The proof is de facto QRAT, but in DQRAT format with explicit dependency sets for extension variables.

ts-LQParity formulas from Section 3.4, and other crafted formulas from QBF literature (Equality [6] and QParity [13, 14]). We generated proofs for these formulas manually (without a solver), with the exception of bridged ts-LQParity, which we solved with `Qute` with $\mathcal{D}^{\forall\text{pure}}$ and modified the prefix in such a way that the extraction through `qrp2rup` (which only works for LD-Q-Res, not LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res) goes through. This is possible, since the result of applying $\mathcal{D}^{\forall\text{pure}}$ to bridged ts-LQParity is again a QBF, and so the LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res proof can be interpreted as a sound LD-Q-Res proof of another QBF (this holds for any QBF with a single universal variable). `DQRAT-check` correctly verified all of these proofs; details and instruction for reproducing are in the supplementary material.

More experiments will be interesting to evaluate the performance of `DQRAT-check` in different scenarios, but for them to be practical, it is first necessary to develop certifying DQBF solvers that output proofs, ideally taking advantage of all DQRAT rules.

5.2 Qute + $\mathcal{D}^{\forall\text{pure}}$

While we have motivated this work on proof checking, we were interested in the solving capabilities of $\mathcal{D}^{\forall\text{pure}}$. To this end we integrated $\mathcal{D}^{\forall\text{pure}}$ into the dependency learning solver `Qute`. As with $\mathcal{D}^{\text{rrs}}$, it is unknown whether `Qute` can soundly use $\mathcal{D}^{\forall\text{pure}}$ for term learning (to prove truth of formulas), so we only turn on $\mathcal{D}^{\forall\text{pure}}$ for clause learning.

We found $\mathcal{D}^{\forall\text{pure}}$ detects no independence on the newest QBFEval 2022 benchmarks, and `Qute` performs better without the overhead of $\mathcal{D}^{\text{rrs}}$ or $\mathcal{D}^{\forall\text{pure}}$. On QBFEval 2020 benchmarks $\mathcal{D}^{\forall\text{pure}}$ shows promise: compared to $\mathcal{D}^{\text{rrs}}$, 7× less time is spent on computing dependencies, 2× more independent pairs are found, and 3× more $\mathcal{D}^{\forall\text{pure}}$ reductions are executed (geometric means), but this did not yield more solved instances or better PAR2 score. For ts-LQParity formulas which separate $\mathcal{D}^{\text{rrs}}$ from $\mathcal{D}^{\forall\text{pure}}$ `Qute` +$\mathcal{D}^{\forall\text{pure}}$ performs well as expected.

6 Conclusion

We have shown that DQRAT+$\mathcal{D}^{\forall\text{pure}}$ is p-equivalent to IndExtQURes, and thus a number of useful p-simulations via transitivity. What we cannot show by transitivity is a p-simulation of the newly created LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res, because long-distance resolution steps do not have a sound meaning in DQBF. Nonetheless we conjecture such a p-simulation will exist and we point to the literature [28, 16, 37] that demonstrate that there are a variety of successful of formalisations and p-simulations of long-distance resolution. We aim to further improve and explore the capabilities of our prototype proof checker `DQRAT-check`.

References

1 Valeriy Balabanov, Hui-Ju Katherine Chiang, and Jie-Hong R Jiang. Henkin quantifiers and Boolean formulae: A certification perspective of DQBF. *Theoretical Computer Science*, 523:86–100, 2014. doi:10.1016/j.tcs.2013.12.020.

2 Valeriy Balabanov and Jie-Hong R. Jiang. Unified QBF certification and its applications. *Formal Methods in System Design*, 41(1):45–65, 2012. doi:10.1007/s10703-012-0152-6.

3 Valeriy Balabanov, Magdalena Widl, and Jie-Hong R. Jiang. QBF resolution systems and their proof complexities. In *SAT 2014*, pages 154–169, 2014. doi:10.1007/978-3-319-09284-3_12.

4 Olaf Beyersdorff and Joshua Blinkhorn. Dependency schemes in QBF calculi: Semantics and soundness. In *Principles and Practice of Constraint Programming - 22nd International Conference (CP)*, pages 96–112, 2016. doi:10.1007/978-3-319-44953-1_7.

5 Olaf Beyersdorff, Joshua Blinkhorn, Leroy Chew, Renate A. Schmidt, and Martin Suda. Reinterpreting dependency schemes: Soundness meets incompleteness in DQBF. *Journal of Automated Reasoning*, 63(3):597–623, 2019. doi:10.1007/s10817-018-9482-4.

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

7 Olaf Beyersdorff, Joshua Blinkhorn, and Meena Mahajan. Building strategies into qbf proofs. *Journal of Automated Reasoning*, 65(1):125–154, January 2021. © The Author(s) 2020. Building Strategies into QBF Proofs This is an open access article under the terms of the Creative Commons Attribution 4.0 International (CC BY 4.0) (https://creativecommons.org/licenses/by/4.0/). doi:10.1007/s10817-020-09560-1.

8 Olaf Beyersdorff, Joshua Blinkhorn, and Tomáš 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*, pages 394–411, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-51825-7_28.

9 Olaf Beyersdorff, Joshua Blinkhorn, and Tomáš Peitl. Strong (D)QBF dependency schemes via implication-free resolution paths. *ACM Trans. Comput. Theory*, 16(4), November 2024. doi:10.1145/3689345.

10 Olaf Beyersdorff and Benjamin Böhm. Understanding the relative strength of QBF CDCL solvers and QBF resolution. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 12:1–12:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ITCS.2021.12.

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

12 Olaf Beyersdorff, Leroy Chew, Judith Clymo, and Meena Mahajan. Short proofs in qbf expansion. In Mikoláš Janota and Inês Lynce, editors, *Theory and Applications of Satisfiability Testing – SAT 2019*, pages 19–35, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-24258-9_2.

13 Olaf Beyersdorff, Leroy Chew, and Mikoláš Janota. Proof complexity of resolution-based QBF calculi. In *Proc. Symposium on Theoretical Aspects of Computer Science*, pages 76–89. LIPIcs series, 2015. doi:10.4230/LIPIcs.STACS.2015.76.

14 Olaf Beyersdorff, Leroy Chew, and Mikoláš 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.

15 Joshua Blinkhorn. Simulating DQBF preprocessing techniques with resolution asymmetric tautologies. *Electron. Colloquium Comput. Complex.*, TR20, 2020. URL: https://eccc.weizmann.ac.il/report/2020/112.

16 Leroy Chew. Proof simulation via round-based strategy extraction for QBF. In Toby Walsh, Julie Shah, and Zico Kolter, editors, *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 – March 4, 2025, Philadelphia, PA, USA*, pages 11176–11184. AAAI Press, 2025. doi:10.1609/aaai.v39i11.33215.

17 Leroy Chew and Marijn J. H. Heule. Relating existing powerful proof systems for QBF. In Kuldeep S. Meel and Ofer Strichman, editors, *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, Haifa, Israel, August 2-5, 2022*, volume 236 of *LIPIcs*, pages 10:1–10:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SAT.2022.10.

18 Leroy Chew and Tomáš Peitl. Better Extension Variables in DQBF via Independence. In Jeremias Berg and Jakob Nordström, editors, *28th International Conference on Theory and Applications of Satisfiability Testing (SAT 2025)*, volume 341 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:24, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SAT.2025.11.

19 Leroy Chew and Friedrich Slivovsky. Towards uniform certification in QBF. *Log. Methods Comput. Sci.*, 20(1), 2024. doi:10.46298/lmcs-20(1:14)2024.

20 Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984. doi:10.1007/BF01744431.

21 J. Håstad. *Computational Limitations of Small Depth Circuits*. MIT Press, Cambridge, 1988.

22 Marijn J. H. Heule, Warren A. Hunt Jr., and Nathan Wetzler. Verifying refutations with extended resolution. In *24th International Conference on Automated Deduction (CADE)*, pages 345–359, 2013. doi:10.1007/978-3-642-38574-2_24.

23 Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere. Short proofs without new variables. In Leonardo de Moura, editor, *Automated Deduction – CADE 26*, pages 130–147, Cham, 2017. Springer International Publishing. doi:10.1007/978-3-319-63046-5_9.

24 Marijn J. H. Heule, Martina Seidl, and Armin Biere. A unified proof system for QBF preprocessing. In *7th International Joint Conference on Automated Reasoning (IJCAR)*, pages 91–106, 2014. doi:10.1007/978-3-319-08587-6_7.

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

26 Toni Jussila, Armin Biere, Carsten Sinz, Daniel Kröning, and Christoph M. Wintersteiger. A first step towards a unified proof checker for QBF. In *SAT 2007*, pages 201–214, 2007. doi:10.1007/978-3-540-72788-0_21.

27 Benjamin Kiesl, Marijn J. H. Heule, and Armin Biere. Truth assignments as conditional autarkies. In Yu-Fang Chen, Chih-Hong Cheng, and Javier Esparza, editors, *Automated Technology for Verification and Analysis*, pages 48–64, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-31784-3_3.

28 Benjamin Kiesl, Marijn J. H. Heule, and Martina Seidl. A little blocked literal goes a long way. In *SAT 2017*, volume 10491 of *Lecture Notes in Computer Science*, pages 281–297. Springer, 2017. doi:10.1007/978-3-319-66263-3_18.

29 Benjamin Kiesl and Martina Seidl. QRAT polynomially simulates ∀Exp+Res. In Mikolás Janota and Inês Lynce, editors, *Theory and Applications of Satisfiability Testing – SAT 2019 – 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings*, volume 11628 of *Lecture Notes in Computer Science*, pages 193–202. Springer, 2019. doi:10.1007/978-3-030-24258-9_13.

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

31 Jan Krajíček and Pavel Pudlák. Quantified propositional calculi and fragments of bounded arithmetic. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 36:29–46, 1990. doi:10.1002/malq.19900360106.

32 Oliver Kullmann. On a generalization of extended resolution. *Discret. Appl. Math.*, 96-97(1):149–176, October 1999. doi:10.1016/S0166-218X(99)00037-2.

33 Oliver Kullmann and Ankit Shukla. Autarkies for DQCNF. In Clark W. Barrett and Jin Yang, editors, *2019 Formal Methods in Computer Aided Design, FMCAD 2019, San Jose, CA, USA, October 22-25, 2019*, pages 179–183. IEEE, 2019. doi:10.23919/FMCAD.2019.8894263.

34 Florian Lonsing and Armin Biere. DepQBF: A dependency-aware QBF solver. *JSAT*, 7(2-3):71–76, 2010. doi:10.3233/sat190077.

35 Florian Lonsing and Uwe Egly. QRAT+: Generalizing QRAT by a more powerful qbf redundancy property. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *Automated Reasoning*, pages 161–177, Cham, 2018. Springer International Publishing. doi:10.1007/978-3-319-94205-6_12.

36 Aina Niemetz, Mathias Preiner, Florian Lonsing, Martina Seidl, and Armin Biere. Resolution-based certificate extraction for QBF – (tool presentation). In Alessandro Cimatti and Roberto Sebastiani, editors, *Theory and Applications of Satisfiability Testing – SAT 2012 – 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings*, volume 7317 of *Lecture Notes in Computer Science*, pages 430–435, Berlin, Heidelberg, 2012. Springer. doi:10.1007/978-3-642-31612-8_33.

37 Tomáš Peitl, Friedrich Slivovsky, and Stefan Szeider. Polynomial-time validation of QCDCL certificates. In Olaf Beyersdorff and Christoph M. Wintersteiger, editors, *Theory and Applications of Satisfiability Testing – SAT 2018 – 21st International Conference, SAT 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*, volume 10929 of *Lecture Notes in Computer Science*, pages 253–269. Springer, 2018. doi:10.1007/978-3-319-94144-8_16.

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

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

40 Mark Peyrer and Martina Seidl. QRP+Gen: A Framework for Checking Q-Resolution Proofs with Generalized Axioms. In Jeremias Berg and Jakob Nordström, editors, *28th International Conference on Theory and Applications of Satisfiability Testing (SAT 2025)*, volume 341 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:10, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SAT.2025.25.

41 Markus N. Rabe. A resolution-style proof system for DQBF. In Serge Gaspers and Toby Walsh, editors, *Theory and Applications of Satisfiability Testing – SAT 2017*, pages 314–325, Cham, 2017. Springer International Publishing. doi:10.1007/978-3-319-66263-3_20.

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

43 Friedrich Slivovsky and Stefan Szeider. Variable dependencies and Q-resolution. In *Theory and Applications of Satisfiability Testing – SAT 2014 – 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, pages 269–284, 2014. doi:10.1007/978-3-319-09284-3_21.

44 Allen Van Gelder. Variable independence and resolution paths for quantified Boolean formulas. In *CP*, pages 789–803, 2011. doi:10.1007/978-3-642-23786-7_59.