Electronic Colloquium on Computational Complexity, Report No. 149 (2025)

Strong (D)QBF Dependency Schemes via Pure Universal Resolution Paths

Leroy Chew and Tomáš Peitl

TU Wien, Austria

Abstract. Certification for Quantified Boolean Formulas (QBF) and Dependency Quantified Boolean Formulas is an ongoing challenge (DQBF). 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 obvious 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 make a theoretical observation, that potentially makes this next step in certification easier. We observe that adding a new dependency scheme rule to the checking format DQRAT is p-equivalent to the inclusion of the independent extension variable. Our new dependency scheme rule, the pure-universal dependency scheme ($\mathcal{D}^{\forall\text{pure}}$) shares many similarities to tautology-free dependency schemes, but is special enough not to be covered in the previous literature.
In addition to soundness we show $\mathcal{D}^{\forall\text{pure}}$ has two other properties that have been found for previous dependency schemes, and each of these observations has potential in solving/checking. We demonstrate a strategy extraction theorem for LD-Q-resolution equipped with $\mathcal{D}^{\forall\text{pure}}$, meaning it can be incorporated soundly into the dependency learning solver `Qute`. And we also demonstrate how $\mathcal{D}^{\forall\text{pure}}$ can be incorporated in the same way Extended Universal Reduction has used previous dependency schemes.

1 Introduction

Our main motivation comes from the recent introduction of a new proof rule known as independent extension [17], 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.

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

ISSN 1433-8092

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, DQBFs often appear in QBF research, because making changes to the quantification order in a QBF can make a QBF easier to solve or prove, and 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 the universal to existential literals. Further investigation found the tautology-free dependency scheme $\mathcal{D}^{\text{tf}}$ [6], where it was apparent that we were able to soundly ignore a resolution path in $\mathcal{D}^{\text{rrs}}$ if it traversed what was essentially a tautology, thus we can remove more dependencies in $\mathcal{D}^{\text{tf}}$ than in $\mathcal{D}^{\text{rrs}}$. Tautology-free dependency schemes can be generalised into implication-free dependency schemes [7].

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 [30](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 considered too large in practice.

One noteworthy QBF format that attempts to tackle this issue is QRAT (Quantified Resolution Asymmetric Tautology)[23]. 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 instead. 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, 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 [12] 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 $\mathcal{D}^{\text{rrs}}$ [17]. 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 show that finding such a system, does not require us to rethink how RAT works in DQBF. In fact, it can be as straightforward as adding one additional rule to the existing system DQRAT. This rule is simply a new dependency scheme, which we name the pure universal dependency scheme ($\mathcal{D}^{\forall\text{pure}}$). $\mathcal{D}^{\forall\text{pure}}$ works in the same conceptual way as tautology-free dependency schemes, but has different side conditions. 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`. When we show that $\mathcal{D}^{\forall\text{pure}}$ can be used for clause modification this means that the proof checker `QRAT-trim` can be soundly generalised by changing a couple of lines. 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 [17].

1.2 Related Work

QBF and DQBF proof complexity has been extensively studied, including work on QBF clause learning proof systems systems [2,3,8], expansion-based proof systems [24,11,10] and stronger proof systems that go beyond current solving techniques [23,9,30]. 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 [43]. The *reflexive resolution path dependency scheme* was developed by Slivovsky and Szeider [44]. The *tautology-free dependency scheme* was developed by Beyersdorff, Blinkhorn and Peitl [6]. Later the same set of authors developed a framework of an infinite number of *implication-free dependency schemes* [7]. 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 [17] for DQBF. Independent extension generalises weaker forms of QBF extension in QBF [25,9]. Independent extension involves conditioning on assignments, and similar ideas have been employed in other works such as conditional autarkies [26,32]. 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 [31]. Resolution asymmetric tautologies (RAT) [22] generalise blocked clauses, and propagation redundancies generalise RAT even further [21].

This work involves the improvement of DQRAT to DQRAT+$\mathcal{D}^{\forall\text{pure}}$. DQRAT [12] is the result of generalising RAT addition rules in DQBF. Previously RAT was generalised into QBF through QRAT [23] and QRAT+ [34], 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

A propositional variable x is single variable that represents a Boolean value. A literal is either a Boolean constant $(0, 1)$, a propositional variable x or its negation $\neg x$. 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 $\overline{(\neg x)} = x$. A clause is a disjunction (logical or) of propositional literals e.g. $x \vee \neg y \vee \neg z$. The empty clause ($\bot$) is the clause that contains no variables, and should be interpreted as false. Clauses that contain 1 are considered satisfied clauses, which are equivalent to tautological clauses that contain a variable in both polarities as $x \vee \neg x$ implies 1. We typically will remove clauses that are satisfied or tautological. When a clause contains more than one copy of the same literal we can use idempotence to remove extra copies of that literal. We can also soundly remove any occurrence of the falsified literal 0 from a clause. Clauses that contain only one literal are known as unit clauses.

A conjunctive normal form (CNF) is a conjunction (logical and) of clauses. We can represent all Boolean formulas on propositional variables with a CNF. The CNF that contains no clauses is considered tautologically true, hence while the empty clause is equivalent to 0, the empty CNF is equivalent to 1. Where convenient, we can think of clauses as sets of literals, and CNFs as sets of clauses. It is common to use both logical notation and set notation and we switch between the two.

For CNF ϕ, var(ϕ) is the set of variables appearing in ϕ. A partial assignment α on ϕ is a partial function that maps var(ϕ) 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.

Clausal Inference 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 that contains unit clauses with the unit propagation procedure Algorithm 1.

Algorithm 1 Unit Propagation

```
1: procedure UP(φ)
2:     while φ contains a unit clause {a} do
3:         φ ← φ|_a
4:         φ ← RemoveSatisfiedClauses(φ)
5:         φ ← RemoveFalsifiedLiterals(φ)
6:     end while
7:     return φ
8: end procedure
```

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

Unit propagation can reach fix-point in polynomial-time but it is not refutationally complete, so for some unsatisfiable CNFs, unit-propagation reaches fix-point before the empty clause is derived. We can generalise to a complete proof system called resolution. A unit propagation procedure can be logged as a series of resolution steps. The resolution rule is as follows; if a CNF contains two clauses C_1 and C_2 such that literal $l \in C_1$ and $\bar{l} \in C_2$, we can create a new clause $C_1 \cup C_2 \setminus \{l, \bar{l}\}$ and add it to our CNF. For all unsatisfiable CNFs we can eventually add the empty clause in this way. A complete process that ends in the empty clause is known as a resolution refutation. This proof can be represented as a DAG (directed acyclic graph). Typically in a resolution refutation we omit all clauses that are not ancestors of the empty clause to concentrate on a connected DAG.

Proof checkers generalise resolution even further. We can soundly derive a clause C into a CNF ϕ if $\phi \wedge \bar{C}$ is a contradiction. However checking for a contradiction is CoNP-complete. Instead 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).

We can represent a CNF containing no tautological clauses as a multi graph. 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. Resolution paths are of semantic importance to formulas. 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. And since resolution is a sound and complete proof system, this indicates ϕ is unsatisfiable if and only if either $\phi \setminus \{D\}$ is unsatisfiable or $\phi \setminus \{E\}$ is unsatisfiable.

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. In particular when we introduce DQBF dependencies, we may restrict $\mathcal{S}$ to dependent variables.

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 $1 \leq i \leq k$. $\mathrm{var}(\Pi)$ is the set of variables appearing in prefix Π. No variable can occur in the prefix twice and every variable from $\mathrm{var}(\phi)$ must occur somewhere in the prefix (i.e. $\mathrm{var}(\phi) \subset \mathrm{var}(\Pi)$).

$\mathrm{var}_\exists(\Pi)$ is the set of existential variables (variables bound by $\exists$) appearing in prefix Π and $\mathrm{var}_\forall(\Pi)$ is the set of universal variables (variables bound by $\forall$) appearing in prefix Π. For assignment α, $\Pi \restriction_\alpha$ removes all variables from $\mathrm{dom}(\alpha)$ 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.

We will define QBF semantics inductively. A quantifier free QBF, contains an empty prefix and its truth is determined by the evaluation of the propositional matrix ϕ, and given our restriction ϕ contains no variables and only constant symbols. A QBF $\forall x \Pi\phi$ is true if and only if $\Pi\phi|_x$ and $\Pi\phi|_{\bar{x}}$ are true. $\exists x \Pi\phi$ is false if and only if $\Pi\phi|_x$ and $\Pi\phi|_{\bar{x}}$ are false. Here the quantifier order matters, so for example $\forall x \exists y (x \vee y) \wedge (\bar{x} \vee \bar{y})$ is a true QBF, but $\exists y \forall x (x \vee y) \wedge (\bar{x} \vee \bar{y})$ is false.

Skolem functions 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 $f_x : \mathrm{D}_x^\Pi \to \{0, 1\}$. 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.

Universal Reduction 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 [29].

In the reduction rule, if C were to contain existential variables that had u in their dependency sets, then a satisfaction of $C \vee u$ may not automatically mean that C is satisfied regardless of u. In order to ensure that C is satisfied whenever $C \vee u$ is, the existential player must be able to costlessly commit to a strategy that determines the variables of C before u is reached. A basic example is if u is pure in all clauses of the QBF $\Pi\phi$, then the existential player is effectively able to ignore the actual play of u by assuming the worst case which is $u = 0$. This can be generalised further, we can reduce $C \vee u$ when u is pure only in the $\Pi\phi$ clauses that can be affected by our switch from $C \vee u$ to C. We can efficiently exclude some clauses from the set of relevant clauses by using resolution paths. Suppose there is no resolution path in the $\exists$ variables right of u from $C \vee u$ to D, then once the game is played up to u and we have current partial assignment α, then $(C \vee u)|_\alpha$ and $D|_\alpha$ cannot be in the same Q-Res refutation of $\Pi \restriction_{A'} \phi|_\alpha$ if it exists. Therefore, because the presence of any $\bar{u}$ literal in D is syntactically irrelevant to removing u from $C \vee u$, the presence of any $\bar{u}$ literal in D is semantically irrelevant to removing u from $C \vee u$.

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

This is how *Extended Universal Reduction* (EUR) works [23]. You may reduce $C \vee u$ to C as long as there is no $\mathcal{S}$-resolution path from C to any clause D with $\bar{u} \in D$. $\mathcal{S}$ here would be the set of existential variables that depend on u. EUR can be checked in polynomial time, but requires global knowledge of the clauses, it is therefore used in the QBF proof system QRAT [23] where a deletion rule allows deletion of clauses that may prevent the use of EUR.

2.3 Dependency Quantified Boolean Formulas

Dependency quantified Boolean formulas (DQBF) extends 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_x^{Π}, the only restriction is that D_x^{Π} is a subset of the universal variables in the prefix Π. The prefix Π is written as $\Pi = \forall u_1 \dots u_p \exists x_1(D_{x_1}) \dots x_q(D_{x_q})$. 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 is true if and only if there is a set of Skolem functions, one for each $\exists$ variable x, such that for every complete assignment to all the universal variables the universal assignment completed with the values of the Skolem functions under that assignment, form a satisfying assignment to the proposition 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 cannot respond to 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 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 $\Pi \subset \Omega$, 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 $\mathrm{D}_x^{\Pi} = \mathrm{D}_x^{\Omega}$.

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 preordering on Π, but it may not be total.

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 that the existential player can know the value of if they know the complete set of values for the dependency set. 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, 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.

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

For existential variable x, the outer variable set $\mathrm{O}_x^{\Pi} := \{y \in \mathrm{var}(\Pi) \mid \mathrm{D}_y^{\Pi} \subset \mathrm{D}_x^{\Pi}\}$. For a universal variable u the inner variables are all existential variables that depend on u i.e. $\mathrm{I}_u^{\Pi} := \{y \in \mathrm{var}(\Pi) \mid \mathrm{D}_u^{\Pi} \subset \mathrm{D}_y^{\Pi}\}$. The outer variables are defined as $\mathrm{O}_u^{\Pi} = \{u\} \cup \bigcap_{x \in \mathrm{I}_u^{\Pi}} \{y \mid \mathrm{D}_y^{\Pi} \subset \mathrm{D}_x^{\Pi} \setminus \{u\}\}$. With a DQBF prefix Π we can restore the relation $\lesssim_{\Pi}$, so $x \lesssim_{\Pi} y$ means $x \in \mathrm{O}_y^{\Pi}$, or equivalently that $\mathrm{O}_x^{\Pi} \subseteq \mathrm{O}_y^{\Pi}$. Since a proof that this was a pre-ordering was not included in the original paper, we provide it in the Appendix (Corollary 5)[1].

Example 1. Consider the prefix: $\forall u, v, w \exists a(u, v, w), b(), c(u, v), d(u, w), e(u)$. We gain the following sets of outer variables (Figure 1):

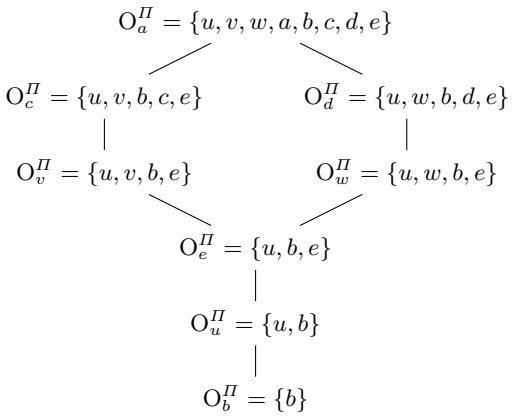

Fig. 1. Hasse diagram of equivalence classes of outer variables of an example DQBF prefix.

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 $u \in \mathrm{D}_x^{\Pi}$ but it does not change the truth of the DQBF to remove u from D_x^{Π}. 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 $u \in \mathrm{D}_x^{\Pi}$.

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

[1] This fact is not necessary for any of the new proofs we present in the paper, but to assist the reader with some intuition about the DQRAT proof system.

as a set of literals $\mathfrak{L}^{\text{rrs}}(\psi, \chi, \mathcal{S})$. Initially, $\mathfrak{C}^{\text{rrs}}(\psi, \chi, \mathcal{S})$ contains all clauses from χ and $\mathfrak{L}^{\text{rrs}}(\psi, \chi, \mathcal{S})$ contains all $\mathcal{S}$-literals in χ. We expand these sets in the following way: suppose $p \in \mathfrak{L}^{\text{rrs}}(\psi, \chi, \mathcal{S})$, we include any ψ clause E with $\bar{p} \in E$ to $\mathfrak{C}^{\text{rrs}}(\psi, \chi, \mathcal{S})$ and include the literals $\{x \in E \mid x \neq \bar{p}, x \in \mathcal{S}\}$ to $\mathfrak{L}^{\text{rrs}}(\psi, \chi, \mathcal{S})$. $\bar{p}$'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 $\mathcal{S}$ 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 $u \sim_{\text{rrs}} x$) if $x \in \mathfrak{L}^{\text{rrs}}(\Pi\phi, \phi_u, \mathcal{S})$, likewise $\bar{u} \sim_{\text{rrs}} x$ if $x \in \mathfrak{L}^{\text{rrs}}(\Pi\phi, \phi_{\bar{u}}, \mathcal{S})$. The purpose of $u \sim_{\text{rrs}} x$ 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, $(u, x) \in \mathcal{D}^{\text{rrs}}(\Pi\phi)$ if $u \in \mathrm{D}_x^{\Pi}$ and either:

1. $u \sim_{\text{rrs}} x$ and $\bar{u} \sim_{\text{rrs}} \bar{x}$
2. $\bar{u} \sim_{\text{rrs}} x$ and $u \sim_{\text{rrs}} \bar{x}$

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

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 [17] discovered an extension rule that created new variables with minimal dependency sets. This powerful calculus is known as IndExtQURes and is given in Figure 2.

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

L is a clause in the propositional matrix ϕ. u is a $\forall$ literal. There is no $\exists$ literal l in C such that $\text{var}(u) \in \mathrm{D}^{\Pi}_{\text{var}(l)}$, 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. $\mathrm{D}_v^{\Pi} = (\mathrm{D}_{y_1}^{\Pi} \cup \mathrm{D}_{y_2}^{\Pi}) \setminus \mathrm{D}_{\alpha}^{\Pi}$.

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

Fig. 2. 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 two of these clauses are identical and one can be simplified with idempotence. We can interpret the clauses as conditionally defining n, $(u \rightarrow (n = \bar{x}))$, 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 $\mathrm{D}_n^{\Pi} = \{\}$.

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 can resolve these over y pivots with more axioms to get $(u \vee \bar{v} \vee n \vee x)$, $(u \vee v \vee n \vee \bar{x})$, $(u \vee \bar{v} \vee \bar{n} \vee \bar{x})$ and $(u \vee v \vee \bar{n} \vee x)$. Having removed y we can reduce these to $(u \vee n \vee x)$, $(u \vee n \vee \bar{x})$, $(u \vee \bar{n} \vee \bar{x})$ and $(u \vee \bar{n} \vee x)$. These can be resolved together over x to get $(u \vee n)$ and $(u \vee \bar{n})$, once we resolve over n we obtain (u) and that reduces to the empty clause. We show this derivation as a DAG in Figure 3.

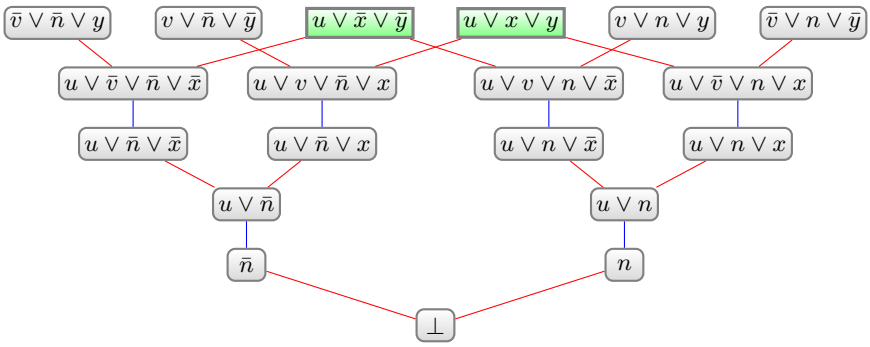

Fig. 3. Proof DAG of the final resolution and reduction steps in Example 2 and Example 3.

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 moved on 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 4). 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 and the reflexive resolution path from Section 2.3.

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 $\text{var}(l) \notin \text{D}^{\Pi}_{C}$.

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

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

Fig. 4. Proof rules of DQRAT [12].

Example 3. Once again we should look at the DQBF from Example 2. We will first use BPM to create a new $\exists$ variable n such that $\text{D}^{\Pi}_{n} = \{u\}$. In terms 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 $\text{DQRAT}_{\exists}$ as no clause contains $\bar{n}$. We can then add $(\bar{n} \vee \bar{x})$ via $\text{DQRAT}_{\exists}$. 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 $u \nsim_{\text{rrs}} n$ and $u \nsim_{\text{rrs}} \bar{n}$, 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 3).

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 existential variable x really does depend on universal variable u after considering the pure universal dependency scheme. 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 $\mathfrak{L}^{\forall\text{pure}}(u, \psi, \chi, \mathcal{S})$. 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 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 $\bar{u} \notin E$ into $\mathfrak{C}^{\forall\text{pure}}(u, \psi, \chi, \mathcal{S})$, and also include the literals $\{x \in D \mid x \neq \bar{p}, x \in \mathcal{S}\}$ 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 2. *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 $u \sim_{\forall\text{pure}} x$) if and only if $x \in \mathfrak{L}^{\forall\text{pure}}(u, \psi, \phi_u, \mathcal{S})$. Where ϕ_u is the set of clauses that contain literal u and $\mathcal{S}$ is the set of $\exists$ variables that contain u in its dependency set.*

Definition 3. *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 $u \in \mathrm{D}_x^{\Pi}$ and either:*

1. *$u \sim_{\forall\text{pure}} x$ and $\bar{u} \sim_{\forall\text{pure}} \bar{x}$, or*
2. *$\bar{u} \sim_{\forall\text{pure}} x$ and $u \sim_{\forall\text{pure}} \bar{x}$.*

Example 4. 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 $v \nsim_{\text{rrs}} x$ nor $v \nsim_{\text{rrs}} \bar{x}$, which means $v \nsim_{\forall\text{pure}} x$ nor $v \nsim_{\forall\text{pure}} \bar{x}$, 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 $u \sim_{\text{rrs}} y$ and $\bar{u} \sim_{\text{rrs}} \bar{y}$ (through $\bar{x}$, first). However $\bar{u} \nsim_{\forall\text{pure}} \bar{y}$ 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 $u \sim_{\text{rrs}} \bar{z}$ but as the path must go through $(\bar{u} \vee \bar{v} \vee \bar{x} \vee \bar{z})$, $u \nsim_{\forall\text{pure}} \bar{z}$.

3.2 Pure Universal Dependency Scheme as a Prefix Modification Rule

Definition of $\mathcal{D}^{\forall\text{pure}}$ 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 5. We can take the QBF from Example 4 and modify its prefix according to $\mathcal{D}^{\forall\text{pure}}$ to get the following DQBF:

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

Soundness

Theorem 1 (Soundness). *Let $\Pi\phi$ be a true DQBF. Let Π' be the prefix where $\text{var}_\exists(\Pi) = \text{var}_\exists(\Pi')$, $\text{var}_\forall(\Pi) = \text{var}_\forall(\Pi')$, $u \in \text{D}_x^{\Pi'}$ if and only if $(u, x) \in \mathcal{D}^{\text{rrs}}(\Pi\phi)$. Then $\Pi'\phi$ is true.*

Proof. Let f be a set of Skolem functions for $\Pi\phi$, Π' the $\mathcal{D}^{\forall\text{pure}}$-reduced prefix. We will show how to modify f to another set of f' of Skolem functions of $\Pi'\phi$. f may require such modification because its functions may use dependencies no longer allowed under Π'. We say a dependency is used by a Skolem function f_x if there is an assignment $\alpha : \text{D}_x^{\Pi} \to \{0, 1\}$ such that flipping the value of u (to obtain α^u) changes the value of f_x, i.e, $f_x(\alpha) \neq f_x(\alpha^u)$. If no dependency absent from Π' is used, then the unused inputs to f can simply be forgotten to obtain a model for $\Pi'\phi$.

Assume that there is an existential variable x and universal $u \in \text{D}_x^{\Pi}$, but $u \notin \text{D}_x^{\Pi'}$, and an assignment $\alpha : \text{D}_x^{\Pi}$ such that $f_x(\alpha) \neq f_x(\alpha^u)$, without loss of generality $\alpha(u) = 0$. Extend α to an arbitrary total assignment β to all universal variables. Define

$$f_x^0(\tau) = \begin{cases} f_x(\tau) \text{ if } \tau \neq \alpha \\ 1 - f_x(\tau) \text{ otherwise,} \end{cases}$$

and $f_y^0 = f_y$ for all $y \neq x$, i.e., flip the value of f_x for α. Consider the full assignment β, and the total response $f^0(\beta)$ of all Skolem functions f^0. If $\beta \cup f^0(\beta)$ satisfies the matrix ϕ we continue the process with a new pair of variables u and x, noting that we have removed an instance of dependency violation (u, x, α).

If $\beta \cup f^0(\beta)$ violates some clause $C \in \phi$, then, since $\beta \cup f(\beta)$ satisfies C, and since $f(\beta)$ and $f^0(\beta)$ differ only in the value of x, it must be the case that a literal on x is in C and it is the only satisfied literal under $\beta \cup f(\beta)$. Since $f^0(\beta^u) = f(\beta^u)$ and $\beta^u \cup f(\beta^u)$ satisfies C, we conclude that either $u \in C$, or some other variable y that depends on u (according to Π) is in C. In the former case, we have established a resolution path from some literal (the one satisfied by $f_x(\alpha)$) on x to u. In the latter case, we will overwrite the value of $f_y(\beta|_{D(y)})$ by $f_y(\beta^u|_{D(y)})$ as we did for x, and continue the process with a new clause that is falsified. As before with x, this new clause must contain $\bar{y}$, extending our resolution path.

At each point the number of dependency violations (u, x, α) decreases, and so the process must terminate. When it terminates, all clauses are satisfied, and either we successfully modified the model, or found a resolution path. If we find a resolution path, we restart the process from α^u, overwriting the value in the opposite way: if this process too terminates with a resolution path, we have a pair of resolution paths as required by Definition 3.

It remains to be shown that these paths are universally pure. This follows easily from the fact that in each execution of the above process, any currently considered clause C is falsified under $\beta \cup f^i(\beta)$. All of these assignments agree on all universal variables: therefore only one polarity of u may ever be encountered along any such path.

Strategy Extraction in Long Distance Q-Resolution Theorem 1 establishes semantic soundness of $\mathcal{D}^{\forall\text{pure}}$: as a DQBF mapping it preserves both falsity (trivially) and truth (Theorem 1). 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. This also covers many scenarios in QBF solving and proof theory, where the use of a dependency scheme is coupled to the underlying proof system more tightly. However, an important case is left unaddressed by Theorem 1: *long-distance Q-resolution (LD-Q-Res)* [2]. LD-Q-Res (proof rules in Figure 5) is a sound proof system for QBF and is supported by the solvers `DepQBF` [33] and `Qute` [38], but it is not sound for DQBF [4], and we cannot infer from Theorem 1 that LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res (proof rules in Figure 6) is sound.

Our goal in this section is to prove the soundness of LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res. We adapt the proof of [39], who showed soundness of the closely related LD-Q($\mathcal{D}^{\text{rrs}}$)-Res proof system.

Theorem 2. *LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res is a sound and complete proof system for QBF. 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.*

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

L is a clause in the propositional matrix ϕ. u is a $\forall$ literal. There is no $\exists$ literal l in C such that $\text{var}(u) \in \text{D}^{\Pi}_{\text{var}(l)}$. In contrast to Figure 2, 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 $\text{var}(v) \in \text{D}^{\Pi}_{\text{var}(x)}$.

Fig. 5. Proof rules of LD-Q-Res applied to an input QBF $\Pi\phi$.

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

L is a clause in the propositional matrix ϕ. u is a $\forall$ literal. There is no $\exists$ literal l in C such that $(u, l) \in \mathcal{D}^{\forall\text{pure}}(\Pi\phi)$. In contrast to Figure 2, 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 $(v, x) \in \mathcal{D}^{\forall\text{pure}}(\Pi\phi)$.

Fig. 6. Proof rules of LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res applied to an input QBF $\Pi\phi$.

The proof follows the proof outline of [39, Theorem 2]. In particular, we will 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 $\mathcal{D}$.

In order to prove that $\mathcal{D}^{\forall\text{pure}}$ is normal, we follow the recipe of [39, Section 5.2]. As in there, we restrict ourselves to QBFs of the form $\forall u \exists x_1, \ldots, \exists x_n \phi$: with a single, outermost universal variable. It is easy to see that one can forget all literals on other universal variables without changing the validity of an LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res refutation.

Lemma 1. *If a clause C is derived by LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res from a QBF $\forall u \Pi\phi$, then $(u, x) \in \mathcal{D}^{\forall\text{pure}}(\Pi\phi) \iff (u, x) \in \mathcal{D}^{\forall\text{pure}}(\Pi\phi \wedge C)$ for all x, i.e., C can be soundly used to calculate $\mathcal{D}^{\forall\text{pure}}$ dependencies of an outermost variable u as if C were an input clause.*

Proof. Clause addition clearly does not take away any resolution-path connections. We need to show that it does not create new connections either.

The case of resolution steps is identical to the case of $\mathcal{D}^{\mathrm{rrs}}$ [39, Lemma 1]. Suppose C_1 and C_2 are resolved on x to obtain C, and a path uses the clause C and its two literals p_1, p_2. If $\{p_1, p_2\} \subseteq C_i$, replace C with C_i, and otherwise replace C with C_1, C_2 connected by the pivot literals $x, \bar{x}$. Either way, C can be replaced by C_1 and C_2 to show the same connections. u-purity is trivially preserved: if C does not contain u or $\bar{u}$, neither C_1 nor C_2 contain it.

The case of reduction steps, trivial for $\mathcal{D}^{\mathrm{rrs}}$, requires some care. Suppose the reduction step $C \vee u \to C$ introduces pure paths. This means there is now a path P from $\bar{u}$ to some x that uses C but could not use $C \vee u$ (because of universal impurity). So this path uses a literal $p \in C$ via which the clause C is entered. Consider the prefix of P that ends in $\bar{p} \in C'$, just before a transition from C' to C is made. This is a $\bar{u}$-pure path that shows $\bar{u} \sim_{\forall\mathrm{pure}} \bar{p}$. But since both $p, u \in (C \vee u)$, also $u \sim_{\forall\mathrm{pure}} p$, and thus $(u, p) \in \mathcal{D}^{\forall\mathrm{pure}}$, contradicting the soundness of the reduction step $C \vee u \to C$. Thus, the reduced clause C is not useful for any new resolution-path connections.

The rest of the proof is identical to the proof in [39].

Lemma 2. *A clause C derived by LD-Q($\mathcal{D}^{\forall\mathsf{pure}}$)-Res from a formula $\forall u \Pi \phi$ cannot contain both u and $\bar{u}$.*

Proof. Consider the resolution step of C_1 and C_2 over the pivot variable x that produced the first clause C with u and $\bar{u}$ (axioms are not tautological, so such a clause must have been produced by resolution). Without loss of generality $x, u \in C_1, \bar{x}, \bar{u} \in C_2$. But then by Lemma 1 $u \sim_{\forall\mathrm{pure}} x$ and $\bar{u} \sim_{\forall\mathrm{pure}} \bar{x}$. Thus, the resolution step is not valid in LD-Q($\mathcal{D}^{\forall\mathsf{pure}}$)-Res.

Proof (Proof of Theorem 2). Consider an LD-Q($\mathcal{D}^{\forall\mathsf{pure}}$)-Res refutation of a formula $\forall u \Pi \phi$ in which both u and $\bar{u}$ occur. Copy clauses to make the refutation tree-like and take a minimal subderivation of some (not necessarily empty) clause C that still contains both u and $\bar{u}$. Call this subderivation, which ends in the clause C, P. By Lemma 2, u and $\bar{u}$ do not occur in C: if $u \in C$, then omit all reduction steps on u, otherwise omit all reduction steps on $\bar{u}$. Call the resulting, still valid, LD-Q($\mathcal{D}^{\forall\mathsf{pure}}$)-Res derivation P^* and its final clause C^*, without loss of generality $u \in C^*, \bar{u} \notin C^*$. P^* must have a reduction step on $\bar{u}$ in the presence of some literal x, so that $(u, \mathrm{var}(x)) \notin \mathcal{D}^{\forall\mathrm{pure}}$. Take the lowermost such reduction step $C_0 \vee \bar{u} \to C_0$. It follows that the bottom of P^* is shaped as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{C_0 \vee \bar{u}}{C_0} \quad C_0'}{C_1} \quad C_1'}{\ddots \quad C_k} \quad C_k'}{C^*}$$

and no C_i, C_i' contains $\bar{u}$. Let p_i be the pivot for the resolution step producing C_i. There is a resolution path from $\bar{u} \in C^*$, through the pivots $p_k, \ldots, p_1$ to

$C_0 \vee \bar{u}$, establishing that $\bar{u} \sim_{\forall\text{pure}} p_1$ and $u \sim_{\forall\text{pure}} \bar{p}_1$, a contradiction with the soundness of the reduction step.

3.3 Pure Path Detection as a Reduction Rule

Definition of $\mathsf{Loc}^{\forall\mathbf{pure}}$-Red Let $\Pi\phi$ by a DQBF and let $C \vee u$ be a clause in ϕ with universal literal u, let $\mathcal{S}$ be the set of variables that depend on $\text{var}(u)$. In extended universal reduction we considered $\mathfrak{C}^{\text{rrs}}(\psi, C \vee u, \mathcal{S})$ to check that no clause in it contained $\bar{u}$. This prevents any $\mathcal{S}$ literal l in C from having $\bar{u} \sim_{\text{rrs}} \bar{l}$, and this was sufficient for u to be reduced in $C \vee u$.

We can improve on this using pure paths. Consider $\mathfrak{C}^{\forall\text{pure}}(\bar{u}, \psi, C \vee u, \mathcal{S})$ and check whether it contains a clause with $\bar{u}$ in, if it does not then $C \vee u$ can reduce to C. We will formally define how this proof rule works.

Definition 4. *Local Pure Literal Reduction $\mathsf{Loc}^{\forall pure}$-Red allows us to make the following derivation*

$$\frac{\Pi\phi \wedge C \vee u}{\Pi\phi \wedge C} \; (\mathsf{Loc}^{\forall pure}\text{-Red})$$

Where Π is a DQBF prefix, ϕ is a CNF, C is a clause and u is a universal literal and where $\text{var}(u)$ *is found in Π. $\mathcal{S}$ is the set of existential variables x in Π such that $u \in \mathrm{D}_x^{\Pi}$. For brevity, ψ is defined as the full DQBF $\Pi\phi \wedge C \vee u$. The main side condition is that $\mathfrak{C}^{\forall\text{pure}}(\bar{u}, \psi, C \vee u, \mathcal{S})$ does not contain any clause that contains $\bar{u}$.*

Soundness We have to demonstrate soundness which we will do in DQBF, we can prove soundness by showing a p-simulation by a sound DQBF proof system and we use IndExtQURes for this. While there my be a shorter proof of soundness we will need this lemma for Section 5.

Lemma 3. *We can p-simulate the $\mathsf{Loc}^{\forall pure}$-Red rule with* IndExtQURes*, by adding new variables and clauses to DQBF.*

Proof. Let Π be the DQBF prefix and $\phi \wedge C \vee u$ be the propositional matrix of the DQBF ψ. Suppose we reduce from $\Pi\phi \wedge C \vee u$ to $\Pi\phi \wedge C$. Consider the $\mathcal{S}$-literals of C, where $\mathcal{S}$ is the set of existential variables that depend on u. In order to make this p-simulate, for each $x \in \mathcal{S}$ we will replace x with x' where $\mathrm{D}_{x'}^{\Pi} \subseteq \mathrm{D}_x^{\Pi}$. Essentially we will recreate $\Pi\phi \wedge C$ as $\Pi'\phi' \wedge C'$ by clause additions, where we substitute each x' for x to change Π to Π', ϕ to ϕ' and C to C'. Technically there is no deletion rule in IndExtQURes, we consider "deletion" a persistent ignoring of the clause, thereafter.

Recall that in the definition $\mathfrak{C}^{\forall\text{pure}}(\bar{u}, \psi, \chi, \mathcal{S})$, we had some subset of clauses χ which we used as a start point. We will study pure path reachability from two different start points, firstly the singular clause set $\{C \vee u\}$. Once we have found all reachable clauses from $\{C \vee u\}$, those remaining unreachable clauses form the second start point. This way we have two "spheres" of reachable clauses that

cover the entire set of clauses. Counter-intuitively, these spheres are not necessarily disjoint because reachability requires us not to immediately re-use literals on the resolution path. The intersection of these spheres will be an important special case that we must handle. Let $L^0 = \mathfrak{L}^{\forall\text{pure}}(\bar{u}, \psi, C \vee u, \mathcal{S})$, $\chi_0 = \mathfrak{C}^{\forall\text{pure}}(\bar{u}, \psi, C \vee u, \mathcal{S})$. Then let $L^1 = \mathfrak{L}^{\forall\text{pure}}(\bar{u}, \psi, \phi \setminus \chi_0, \mathcal{S})$, $\chi_1 = \mathfrak{C}^{\forall\text{pure}}(\bar{u}, \psi, \phi \setminus \chi_0, \mathcal{S})$. We introduce conditional definition $\bar{u} \to (x^{\bar{u}} \leftrightarrow x)$ using the IndExt rule for each variable that has a literal in L^0.

For each clause D in χ_0 we replace each literal $x \in \mathcal{S}$ with $x^{\bar{u}} \vee u$ via resolving with the definition clauses. Note that χ_0 contains no $\bar{u}$ literals by the side condition of the rule, and no other clause except $C \vee u$ contains a u literal by the path purity. Removing all $\mathcal{S}$-literals, means we can reduce any u-literals from what was once χ_0, including the one originating from C and those introduced from resolving away the $\mathcal{S}$-literals. We call this set $\chi_0^{\bar{u}}$ because it is what you would get in the $\bar{u}$ expansion of χ_0.

Now consider the literals x in L_0 such that $x \in L_1$. In $\chi_0^{\bar{u}}$ we weaken all $x^{\bar{u}}$ literals to $x^{\bar{u}} \vee x$ and in χ_1 we weaken all x literals to $x^{\bar{u}} \vee x$. $\bar{x}$ can only appear in clauses in the intersection $\chi_0 \cap \chi_1$, because if x in L_0 all clauses with $\bar{x}$ are in χ_0, and likewise with χ_1. Furthermore, $\bar{x}$ cannot be in L_0 otherwise there is a u-free path from C to $\bar{x}$ and a u-free path from x to some $D \in \phi \setminus \chi_0$ meaning D is actually in χ_0. Likewise, $\bar{x}$ cannot be in L_1 otherwise there is a u-free path from C to x and a u-free path from $\bar{x}$ to some $D \in \phi \setminus \chi_0$ meaning D is actually in χ_0.

Each clause $D \in \chi_0 \cap \chi_1$ has a unique entry literal $\bar{x}$ which cannot be in L_0 nor L_1. This means its sufficient just to have two copies, one originating from χ_0 and one originating from χ_1. Note that IndExtQURes p-simulates Frege rules because it is p-equivalent to IndExtFrege + ∀red, so after the weakening we can use distributivity on the two copies of the intersection clause, to replace $\bar{x}$ with $\bar{x}^{\bar{u}} \wedge \bar{x}$.

The full replacement scheme is as follows

$$x' = \begin{cases} x^{\bar{u}} & \text{if } x \in L_0, x \notin L_1, \\ x^{\bar{u}} \vee x & \text{if } x \in L_0, x \in L_1, \\ x^{\bar{u}} \wedge x & \text{if } \bar{x} \in L_0, \bar{x} \in L_1, \\ x & \text{otherwise.} \end{cases}$$

x' has dependency set $\mathrm{D}^{\Pi}_{x'} \subseteq \mathrm{D}^{\Pi}_{x}$, and replaces x in all clauses, all extra literals introduced have been reduced.

Therefore we have derived a set of clauses under a set of variables that have the same structure as the original set such that the new set of clauses is the correct substitution of variables of the original set of clauses, with the exception that the substitution of $C \vee u$ is replaced by a substitution of C. Despite IndExtQURes not having a deletion rule, we can ignore all variables replaced by substitution and all original clauses and clauses used as in intermediate part of of this proof. This is practically deletion. The only difference is that $\mathrm{D}^{\Pi}_{x'}$ may be

strictly smaller than D_x^{Π}, but this does not prevent any future steps. One can easily imagine IndExtQURes with a dependency weakening rule.

Corollary 1. *Local Pure Literal Reduction (Loc$^{\forall pure}$-Red) is a sound QBF/DQBF Rule.*

For valid (Loc$^{\forall\text{pure}}$-Red) steps, note that a checker should, in theory, require no more computation than an simple EUR checker, because the resolution paths are shorter. Every valid EUR step is also an Loc$^{\forall\text{pure}}$-Red step, so a QRAT checker (such as `QRAT-trim`) need only check for Loc$^{\forall\text{pure}}$-Red and not EUR.

The advantage of using Loc$^{\forall\text{pure}}$-Red over EUR is that when p-simulating expansion based solving [23,28], one will not have to delete the definitions $\alpha \rightarrow (x = x^{\alpha})$ in order to remove universal literals.

4 Separations

In Section 3.2 we showed how $\mathcal{D}^{\forall\text{pure}}$ can be combined with LD-Q-Res in the same way that $\mathcal{D}^{\text{rrs}}$ has worked with LD-Q-Res in the past. 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 [3,11] 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 5 (ts-LQParity(N)). *Let* $\mathrm{xor}_l(o_1, o_2, o, z)$ *be the set of clauses* $\{(z \vee \neg o_1 \vee \neg o_2 \vee \neg o), (z \vee o_1 \vee o_2 \vee \neg o), (z \vee \neg o_1 \vee o_2 \vee o), (z \vee o_1 \vee \neg o_2 \vee o)\}$

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

Lemma 4. *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 $z, \bar{z}$ or z^* literal. 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 z^* without a blocking existential literal can be immediately reduced without cost to the proof size, so $z, \bar{z}, z^*$ literals and inner existential literals can only coincide in reduced clauses. For more details follow the argument from [11].

As argued in [11], long distance steps (merge steps) are possible, but can never be reduced as this can only 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 5. *The shortest LD-Q-Res refutations of ts-*LQParity*(N) are exponential in N.*

Proof. Citing [11, 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 $x_1 \dots x_n$. Parity has exponential lower bounds in bounded-depth circuits [19,20], 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 6 (Bridged ts-LQParity).

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

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 4 and 5. But previously there were no resolution paths between t-variables and s-variables. But now b acts as a bridge.

Lemma 6. *$\mathcal{D}^{\text{rrs}}$ and $\mathcal{D}^{\text{trv}}$ are equivalent on Bridged ts-*LQParity.

Proof. **Induction hypothesis** (on i): $z \sim_{\text{rrs}} t_{N-i}$ and $\neg z \sim_{\text{rrs}} \neg t_{N-i}$.
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 2. *LD-Q($\mathcal{D}^{\text{rrs}}$)-Res requires exponential-size proofs of Bridged ts-*LQPARITY_N.

Lemma 7. *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 3. *LD-Q($\mathcal{D}^{\text{rrs}}$)-Res does not p-simulate LD-Q($\mathcal{D}^{\forall\text{pure}}$)-Res.*

5 P-Equivalence with IndExtQURes

In this section, we observe a new key connection between our new dependency scheme $\mathcal{D}^{\forall\text{pure}}$ and the new proof system IndExtQURes by Chew and Peitl [17]. When looking at proof complexity, IndExtQURes has been proven to be very powerful relative to the other proof systems for both QBF and DQBF. In fact IndExtQURes has been shown to p-simulate the majority of QBF and DQBF proof systems (see Figure 7 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 4), 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 $u \notin \mathrm{D}_x^{\Omega}$ only if $u \notin \mathrm{D}_x^{\Pi}$ or $(u, x) \notin \mathcal{D}^{\forall\text{pure}}(\Pi\phi)$.

Definition 7. *(The refutational version of) DQRAT+$\mathcal{D}^{\forall\text{pure}}$ 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 $\mathcal{D}^{\forall\text{pure}}$.*

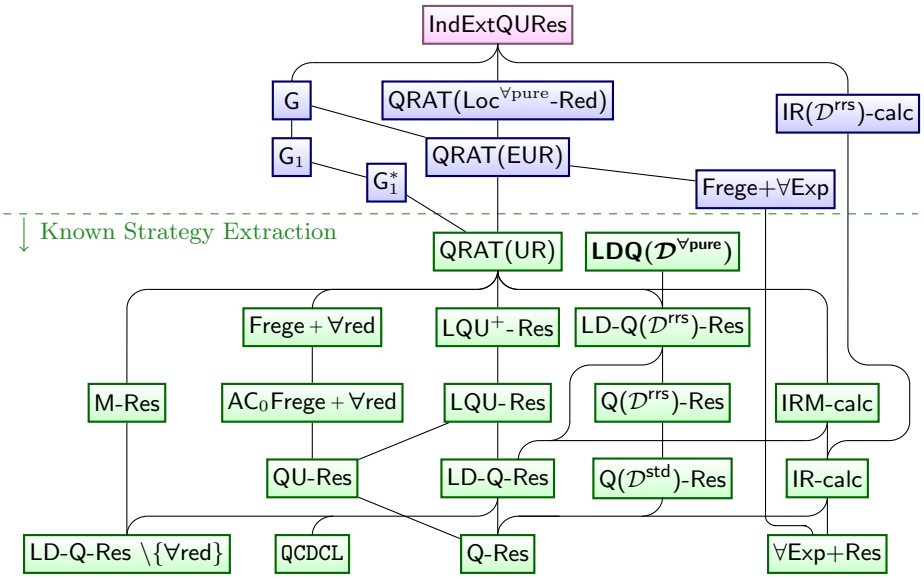

Fig. 7. The p-simulation structure of refutational QBF proof systems [2,3,5,8,9,11,15,16,18,17,23,28,29,30,39,42,45,46].

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. It is the direction with the simpler proof as IndExtQURes only has a small number of clause addition rules.

Theorem 3. *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 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 $(\bar{\alpha} \vee n \vee a), (\bar{\alpha} \vee n \vee b), (\bar{\alpha} \vee \bar{n} \vee \bar{a} \vee \bar{b})$. $\mathrm{D}_n^{\Pi} = \mathrm{D}_a^{\Pi} \cup \mathrm{D}_b^{\Pi} \setminus \mathrm{D}_\alpha^{\Pi}$ using the **(IndExt)** rule. We first choose to add the existential variables n with $\mathrm{D}_n^{\Pi} = \mathrm{D}_a^{\Pi} \cup \mathrm{D}_b^{\Pi}$ 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 $\mathrm{D}_n^{\Pi} = \mathrm{D}_a^{\Pi} \cup \mathrm{D}_b^{\Pi}$, 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 a literal u is a conjunct in the assignment α. All paths from $\bar{u}$ to n or $\bar{n}$ pass through u. Therefore we can drop var(u) from D_n^{Π}. We can do this for each literal in α. At the end $\mathrm{D}_n^{\Pi} = \mathrm{D}_a^{\Pi} \cup \mathrm{D}_b^{\Pi} \setminus \mathrm{D}_\alpha^{\Pi}$.

We can also show the reverse, that IndExtQURes p-simulates DQRAT+$\mathcal{D}^{\forall\mathrm{pure}}$. For the six original rules of DQRAT we already know how to do a p-simulation [17]. So our efforts will be to p-simulate the $\mathcal{D}^{\forall\mathrm{pure}}$ rule. Note that previously IndExtQURes has been shown to p-simulate the $\mathcal{D}^{\mathrm{rrs}}$ rule and we use that proof to construct ours here. In that proof, an important lemma was that IndExtQURes could p-simulate the Extended Universal Reduction(EUR) rule. The EUR is a localised form of $\mathcal{D}^{\mathrm{rrs}}$ that allows a reduction based on the $\mathcal{D}^{\mathrm{rrs}}$ scheme but does not modify the prefix. For $\mathcal{D}^{\forall\mathrm{pure}}$ the analogue to EUR is Loc$^{\forall\mathrm{pure}}$-Red, and fortunately we have proved its p-simulation by IndExtQURes in Lemma 3.

Theorem 4. IndExtQURes *p-simulates DQRAT+*$\mathcal{D}^{\forall\mathrm{pure}}$.

Proof. We know IndExtQURes p-simulates DQRAT already. What we have to show is that IndExtQURes p-simulates the $\mathcal{D}^{\forall\mathrm{pure}}$ rule. To do this we follow a similar proof to IndExtQURes p-simulating the $\mathcal{D}^{\mathrm{rrs}}$ rule, for each spurious dependency (u, x) we replace x with another variable x' where $D_{x'} = D_x \setminus \{u\}$.

Given a DQBF with propositional CNF matrix ϕ and let u be some universal variable in the prefix, and x be an existential variable such that x depends on u, but only spuriously: $u \in \mathrm{D}_x^{\Pi}, (u, x) \notin \mathcal{D}^{\forall\mathrm{pure}}$. We define χ_u to be the subset of ϕ where all clauses contain literal u and $\chi_{\bar{u}}$ to be the subset of ϕ where all clauses contain literal $\bar{u}$. Define $L^u = \mathfrak{L}^{\forall\mathrm{pure}}(u, \psi, \chi_u, \mathcal{S})$, $L^{\bar{u}} = \mathfrak{L}^{\forall\mathrm{pure}}(\bar{u}, \psi, \chi_{\bar{u}}, \mathcal{S})$, where $\mathcal{S}$ is the set of existential variables that contain u in its dependency set.

If $(u, x) \notin \mathcal{D}^{\forall\mathrm{pure}}(\Pi\phi)$ then we have four cases (up to symmetry) of x's membership in L^u and $L^{\bar{u}}$:

1. $x \notin L_u$, $\bar{x} \notin L_u$, $x \notin L_{\bar{u}}$ and $\bar{x} \notin L_{\bar{u}}$
2. $x \in L_u$, $\bar{x} \notin L_u$, $x \notin L_{\bar{u}}$ and $\bar{x} \notin L_{\bar{u}}$
3. $x \in L_u$, $\bar{x} \notin L_u$, $x \in L_{\bar{u}}$ and $\bar{x} \notin L_{\bar{u}}$
4. $x \in L_u$, $\bar{x} \in L_u$, $x \notin L_{\bar{u}}$ and $\bar{x} \notin L_{\bar{u}}$

For cases 1, 2 and 3, we can take advantage of the fact that $\bar{x}$ has no pure path to either u or $\bar{u}$. In fact there is no path at all from $\bar{x}$ to either u or $\bar{u}$, as a non-pure path could be minimised to a pure path of one of the polarities. We replace x with $x^{\bar{u}} \vee x^u$. Variables $x^{\bar{u}}, x^u$ are defined by the conditional definitions

$\bar{u} \to (x^{\bar{u}} \leftrightarrow x)$, $u \to (x^u \leftrightarrow x)$. We have to replace each x literal with $x^{\bar{u}} \vee x^u$, this is quite straightforward as we can derive $\bar{x} \vee x^{\bar{u}} \vee x^u$ by resolving over u in the definition clauses. Replacing $\bar{x}$ is a more difficult matter. For a clause C we can replace $\bar{x}$ with $\bar{x}^{\bar{u}} \vee u$ but we would require a way to remove the u literal. We can remove the u once we have replaced all variables in $\mathcal{S}$ that have spurious dependencies on u. In this case there cannot be any $l \in C$ such that there is a resolution path from $\bar{l}$ to $\bar{u}$, otherwise there would be a path from $\bar{u}$ to $\bar{x}$, nor can there be a path from $\bar{l}$ to u for the same reason. Hence l is also a case 1, 2 or 3 literal. This means we can reduce the u we introduce to clause C. Thus we can replace $\bar{x}$ with $\bar{x}^{\bar{u}}$ and symmetrically we can do the same to create another copy where we replace $\bar{x}$ with $\bar{x}^u$. By using distributivity we get $\bar{x}$ replaced by $\bar{x}^{\bar{u}} \wedge \bar{x}^u$, which fortunately is the negation of what we replaced x with. Thus we have $x' = x^{\bar{u}} \vee x^u$

The remaining case 4, requires us to replace x with $x^{\bar{u}}$. Initially via resolution we can replace x with $x^{\bar{u}} \vee u$ in some clause C. In the case that u already was in C we do not need to remove it. Otherwise we argue there are no other literals l in C such that there is a pure path from $\bar{l}$ to $\bar{u}$, because there would be a pure path from $\bar{u}$ to x. This means we can use local pure literal elimination to remove said u, because there is no u-free path from C to $\bar{u}$.

Corollary 4. IndExtQURes *and* $DQRAT{+}\mathcal{D}^{\forall\text{pure}}$ *are p-equivalent.*

6 Practical Implications

The QBF solver `Qute` [38] already supports $\mathcal{D}^{\text{rrs}}$, and so we implemented $\mathcal{D}^{\forall\text{pure}}$ support for `Qute`. One may think that $\mathcal{D}^{\forall\text{pure}}$ not only finds more independence than $\mathcal{D}^{\text{rrs}}$, but it also makes calculations faster, the latter because resolution paths that would need to be explored for $\mathcal{D}^{\text{rrs}}$ might be aborted early due to universal impurity. This is not quite as simple.

For both $\mathcal{D} \in \{\mathcal{D}^{\text{rrs}}, \mathcal{D}^{\forall\text{pure}}\}$, one can compute all x with $(u, x) \in \mathcal{D}$ in linear time, and so all dependent pairs in overall quadratic time. For $\mathcal{D}^{\text{rrs}}$ it is additionally possible to compute all dependencies *of* some existential variable x in quasilinear time with a Dijkstra-style algorithm [37]. The latter seems hard for $\mathcal{D}^{\forall\text{pure}}$ as resolution paths from a fixed existential literal to many universal targets may be polluted with different subsets of universal literals, leading to an exponential blowup.

This affects the implementation of $\mathcal{D}^{\text{rrs}}$ and $\mathcal{D}^{\forall\text{pure}}$ in `Qute`. `Qute` uses QCDCL (quantified conflict-driven clause learning) with *dependency learning* to learn dependencies between variables dynamically. In dependency learning, the solver starts branching and propagating as if there were no dependencies (equivalently, implicitly reducing all universal literals). Only if this leads to a *dependency conflict*, which is a resolution step in LD-Q-Res that should have been valid but is not because of forbidden v literals from Figure 6, does the solver learn missing dependencies to prevent the conflict from taking place again. At this point a dependency scheme $\mathcal{D}$ can be inserted: if all blocking literals v are found independent, the resolution step can soundly be carried out in LD-Q($\mathcal{D}$)-Res. In

order to avoid a quadratic blow up from computing upfront and storing all $\mathcal{D}^{\text{rrs}}$ dependencies, `Qute` only computes the required dependencies on demand during dependency conflicts (and then stores them forever). Because multiple universal variables v may be blocking in a dependency conflict, `Qute` computes all dependencies of the pivot variable x. As shown in [37], all $\mathcal{D}^{\text{rrs}}$ dependencies of an existential variable can be found in quasilinear time with a Dijkstra-style algorithm. The same, however, does not seem possible for $\mathcal{D}^{\forall\text{pure}}$: in the search for resolution paths, any currently explored path can be impure for any subset of universal literals, and thus the actual number of paths to explore is in general exponential. For this reason, our implementation of $\mathcal{D}^{\forall\text{pure}}$ computes all existential variables that depend on a given universal variable u, which can be done in linear time with depth or breadth-first search. The price we pay (compared to `Qute`'s $\mathcal{D}^{\text{rrs}}$ implementation) is that in a dependency conflict we need to compute dependencies on every blocker v. We evaluated `Qute` with $\mathcal{D}^{\forall\text{pure}}$ on the formulas from Section 4. Even with $\mathcal{D}^{\text{rrs}}$, `Qute`'s running time scales exponentially, while with $\mathcal{D}^{\forall\text{pure}}$ the formulas are solved instantly (Figure 8). This is all in line with proof complexity; we note that such clean mirroring of proof complexity in solver performance is far from given: previous work on QCDCL proof complexity found this business to be tricky with `Qute` unable to solve theoretically easy formulas quickly [13,14]. We also evaluated `Qute` on the PCNF track of QBFEval 2022, but saw no improvement over $\mathcal{D}^{\text{rrs}}$ or vanilla `Qute`.

Experimental details In 15 minutes and out of 434 instances of the PCNF track of QBFEval 2022[2], vanilla `Qute` solved 74, with $\mathcal{D}^{\text{rrs}}$ 72, and with $\mathcal{D}^{\forall\text{pure}}$ 70 instances. The set of instances solved with $\mathcal{D}^{\forall\text{pure}}$ was a subset of those solved with $\mathcal{D}^{\text{rrs}}$, which in turn was a subset of those solved by vanilla `Qute`. More experiments will be necessary to determine whether there are practical instances on which $\mathcal{D}^{\forall\text{pure}}$ (or even $\mathcal{D}^{\text{rrs}}$ for that matter) provides a boost.

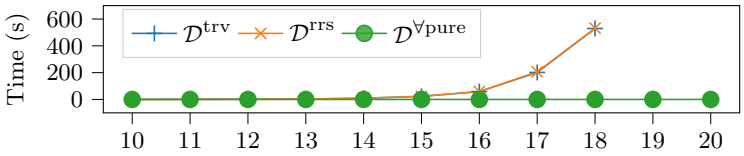

Fig. 8. Vanilla `Qute` ($\mathcal{D}^{\text{trv}}$) vs `Qute` with $\mathcal{D}^{\text{rrs}}$ and $\mathcal{D}^{\forall\text{pure}}$ on Bridged ts-LQParity. The x-axis gives n. $\mathcal{D}^{\text{trv}}$ and $\mathcal{D}^{\text{rrs}}$ timed out at 10 minutes for $n \geq 19$.

The experiments on `Qute` are somewhat tangential to the motivations of this paper, which were mainly certification rather than solving, but we are interested in the results regardless. Unfortunately, as there is no implementation of DQRAT we were unable to implement any of the p-simulation ideas into a practical

[2] `https://www.qbflib.org/qbfeval2022_results.php` [35,41]

certification example. But we would like to do this in future work and one of our next goals is to construct a DQRAT checker.

7 Conclusion

As we have shown that DQRAT+$\mathcal{D}^{\forall\text{pure}}$ is p-equivalent to IndExtQURes, we therefore show 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 work by Kiesl, Heule and Seidl [27] that shows how QRAT can p-simulate long distance as well as the work by Chew [15], which shows how extension variables can handle dependency schemes, as potential techniques.

Acknowledgments

Thanks to Joshua Blinkhorn and Martina Seidl for their discussions. This work is supported by FWF Project ESP197.

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.
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.
4. O. Beyersdorff, J. Blinkhorn, L. Chew, R. A. Schmidt, and M. 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.
5. Olaf Beyersdorff, Joshua Blinkhorn, and M. Mahajan. Building strategies into QBF proofs. In *Electron. Colloquium Comput. Complex.*, 2018.
6. 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.
7. Olaf Beyersdorff, Joshua Lewis 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.
8. 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.

9. Olaf Beyersdorff, Ilario Bonacina, Leroy Chew, and Jan Pich. Frege systems for quantified Boolean logic. *J. ACM*, 67(2), April 2020.
10. Olaf Beyersdorff, Leroy Chew, Judith Clymo, and Meena Mahajan. Short proofs in QBF expansion. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:102, 2018. URL: `https://eccc.weizmann.ac.il/report/2018/102`.
11. 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.
12. Joshua Blinkhorn. Simulating DQBF preprocessing techniques with resolution asymmetric tautologies. *Electron. Colloquium Comput. Complex.*, TR20, 2020. URL: `https://api.semanticscholar.org/CorpusID:221159787`.
13. Benjamin Böhm, Tomás Peitl, and Olaf Beyersdorff. QCDCL with cube learning or pure literal elimination - what is best? *Artif. Intell.*, 336:104194, 2024. URL: `https://doi.org/10.1016/j.artint.2024.104194`, `doi:10.1016/J.ARTINT.2024.104194`.
14. Benjamin Böhm, Tomás Peitl, and Olaf Beyersdorff. Should decisions in QCDCL follow prefix order? *J. Autom. Reason.*, 68(1):5, 2024. URL: `https://doi.org/10.1007/s10817-024-09694-6`, `doi:10.1007/S10817-024-09694-6`.
15. 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. URL: `https://doi.org/10.1609/aaai.v39i11.33215`, `doi:10.1609/AAAI.V39I11.33215`.
16. Leroy Chew and Marijn J. H. Heule. Relating existing powerful proof systems for QBF. *Electron. Colloquium Comput. Complex.*, 27:159, 2020.
17. 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. URL: `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.SAT.2025.11`, `doi:10.4230/LIPIcs.SAT.2025.11`.
18. 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`.
19. Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.
20. J. Håstad. *Computational Limitations of Small Depth Circuits*. MIT Press, Cambridge, 1988.
21. 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.
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.
23. 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.
24. Mikoláš Janota and Joao Marques-Silva. Expansion-based QBF solving versus Q-resolution. *Theor. Comput. Sci.*, 577:25–42, 2015.

25. 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.
26. 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.
27. 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.
28. Benjamin Kiesl and Martina Seidl. QRAT polynomially simulates ∀Exp+Res. In *SAT 2019*, volume 11628 of *Lecture Notes in Computer Science*, pages 193–202. Springer, 2019.
29. Hans Kleine Büning, Marek Karpinski, and Andreas Flögel. Resolution for quantified Boolean formulas. *Inf. Comput.*, 117(1):12–18, 1995.
30. 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.
31. O. Kullmann. On a generalization of extended resolution. *Discrete Appl. Math.*, 96–97(1):149–176, October 1999. doi:10.1016/S0166-218X(99)00037-2.
32. Oliver Kullmann and Ankit Shukla. Introducing autarkies for dqcnf, 07 2019. doi:10.48550/arXiv.1907.12156.
33. Florian Lonsing and Armin Biere. DepQBF: A dependency-aware QBF solver. *JSAT*, 7(2-3):71–76, 2010.
34. 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.
35. Massimo Narizzano, Luca Pulina, and Armando Tacchella. The qbfeval web portal. In *European Workshop on Logics in Artificial Intelligence*, pages 494–497. Springer, 2006.
36. Aina Niemetz, Mathias Preiner, Florian Lonsing, Martina Seidl, and Armin Biere. Resolution-based certificate extraction for qbf. In Alessandro Cimatti and Roberto Sebastiani, editors, *Theory and Applications of Satisfiability Testing – SAT 2012*, pages 430–435, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
37. Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Combining resolution-path dependencies with dependency learning. 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 306–318. Springer, 2019. doi:10.1007/978-3-030-24258-9_22.
38. Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Dependency learning for QBF. *J. Artif. Intell. Res.*, 65:180–208, 2019.
39. Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Long-distance Q-Resolution with dependency schemes. *J. Autom. Reason.*, 63(1):127–155, 2019.
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. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.SAT.2025.25, doi:10.4230/LIPIcs.SAT.2025.25.
41. Luca Pulina and Martina Seidl. The 2016 and 2017 QBF solvers evaluations (qbfeval'16 and qbfeval'17). *Artif. Intell.*, 274:224–248, 2019. URL: https://doi.org/10.1016/j.artint.2019.04.002, doi:10.1016/J.ARTINT.2019.04.002.
42. 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.
43. Marko Samer and Stefan Szeider. Backdoor sets of quantified Boolean formulas. *J. Autom. Reasoning*, 42(1):77–97, 2009.
44. Friedrich Slivovsky and Stefan Szeider. Variable dependencies and Q-Resolution. International Workshop on Quantified Boolean Formulas, 2013.
45. Friedrich Slivovsky and Stefan Szeider. Variable dependencies and Q-resolution. In Carsten Sinz and Uwe Egly, editors, *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*, volume 8561, pages 269–284. Springer, 2014. URL: http://dx.doi.org/10.1007/978-3-319-09284-3_21, doi:10.1007/978-3-319-09284-3_21.
46. Allen Van Gelder. Variable independence and resolution paths for quantified Boolean formulas. In Jimmy Ho-Man Lee, editor, *CP*, volume 6876, pages 789–803. Springer, 2011.

A Appendix

A.1 Further Details on Outer Variables in DQBF

Here we expand on the description of outer variables from Section 2.3.

Theorem 5. *Let Π be a DQBF prefix. Let x and y be variables in Π. If $y \in \mathrm{O}_x^\Pi$ then $\mathrm{O}_y^\Pi \subseteq \mathrm{O}_x^\Pi$.*

Proof. We will assume throughout the proof that $y \in \mathrm{O}_x^\Pi$

First suppose x is existential and y also existential. Then $\mathrm{D}_y^\Pi \subseteq \mathrm{D}_x^\Pi$. If $z \in \mathrm{O}_y^\Pi$ then $\mathrm{D}_z^\Pi \subset \mathrm{D}_y^\Pi$ and thus $\mathrm{D}_z^\Pi \subset \mathrm{D}_x^\Pi$ meaning $z \in \mathrm{O}_x^\Pi$.

Now suppose x is existential and y is universal, then $y \in \mathrm{D}_x^\Pi$. Now consider K_y^Π the set of universal variables in O_y^Π. Each of these also appears in every dependency set y is in including D_x^Π, hence they also appear in O_x^Π. Now suppose $z \in \mathrm{O}_y^\Pi$ and z is existential, then D_z^Π only contains variables in K_y^Π thus $\mathrm{D}_z^\Pi \subseteq \mathrm{D}_x^\Pi$ and $z \in \mathrm{O}_x^\Pi$.

Next suppose x is universal and y is universal. We say y is in K_x^Π, meaning y is in all the dependency sets that x is in. If $z \in \mathrm{O}_y^\Pi$ then this is defined based on all the dependency sets y is in, which contain all the dependency sets of x. So if z is in the kernel of y it is in the kernel of x, and if D_x^Π contains only variables of K_y^Π it contains only variables of K_x^Π. Thus, whether z is universal or existential it is contained in O_x^Π.

Finally suppose x is universal and y is existential. Then D_y only contains variables from K_x^Π. Let z be such that $z \in \mathrm{O}_y^\Pi$. If z is existential then $\mathrm{D}_z^\Pi \subseteq \mathrm{D}_y^\Pi$ which contains only variables from K_x^Π. Neither contain x so $z \in \mathrm{O}_x^\Pi$. If z is universal then z is in the dependency set of y, and thus in the kernel of x.

Corollary 5. *For a DQBF $\Pi\phi$, $\lesssim_\Pi$ is a pre-order.*

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il