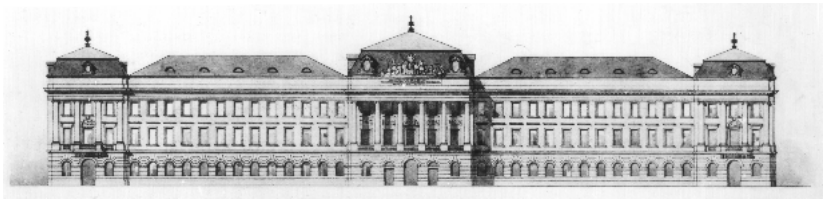
INFSYS
RESEARCH
REPORT



INSTITUT FÜR INFORMATIONSSYSTEME
ABTEILUNG WISSENSBASIERTE SYSTEME

DEPENDENCY SCHEMES AND Q-RESOLUTION

Friedrich Slivovsky Stefan Szeider

INFSYS RESEARCH REPORT 1843-14-09
DECEMBER 2014

Institut für Informationssysteme
Abtg. Wissensbasierte Systeme
Technische Universität Wien
Favoritenstraße 9-11
A-1040 Wien, Austria
Tel: +43-1-58801-18405
Fax: +43-1-58801-18493
sek@kr.tuwien.ac.at
www.kr.tuwien.ac.at

TU
TECHNISCHE UNIVERSITÄT WIEN

INFSYS Research Report
INFSYS Research Report 1843-14-09, December 2014

Dependency Schemes and Q-resolution

Friedrich Slivovsky[1*] Stefan Szeider[1*]

Abstract. We propose $Q(D)$-resolution, a proof system for Quantified Boolean Formulas. $Q(D)$-resolution is a generalization of Q-resolution parameterized by a dependency scheme D. This system is motivated by the generalization of the QDPLL algorithm using dependency schemes implemented in the solver DepQBF. We prove soundness of $Q(D)$-resolution for a dependency scheme D that is strictly more general than the standard dependency scheme; the latter is currently used by DepQBF. This result is obtained by proving correctness of an algorithm that transforms $Q(D)$-resolution refutations into Q-resolution refutations and could be of independent practical interest. We also give an alternative characterization of resolution-path dependencies in terms of directed walks in a formula's implication graph which admits an algorithmically more advantageous treatment.

[1] Institute of Information Systems, Vienna University of Technology, Vienna, Austria.
* Supported by the European Research Council (ERC), project 239962 (COMPLEX REASON).

Publication Information: This is the authors' self-archived copy of a paper that appeared in the *Proceedings of the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT 2014)*, dx.doi.org/10.1007/978-3-319-09284-3_21. The final publication is available at www.springer.com.

1 Introduction

The satisfiability problem of *Quantified Boolean Formulas* (QBFs) is a canonical PSPACE-complete decision problem [17]. QBFs offer a convenient language for representing problems from domains such as model checking, planning, or knowledge representation and reasoning. In practice, QBF solvers are expected to generate certificates witnessing the satisfiability or unsatisfiablity of input formulas. These certificates serve a dual purpose:

1. Certificates encode information that is valuable in applications settings (e.g., a plan, or a counterexample in model checking).

2. Certificates can be used to verify that the answer given by the solver is correct.

Search-based QBF solvers implementing the QDPLL algorithm [7] typically use variants of *Q-resolution* [5] as a certificate language [13]. When assigning variables during the search process, the QDPLL algorithm observes the order induced by the nesting of quantifiers in the input formula. This is often needlessly restrictive, in particular for formulas in the common QCNF format which places all quantifiers in a single, linear quantifier prefix. An appealing approach to dealing with this restriction is the generalization of QDPLL by means of *dependency schemes* implemented in DepQBF [2, 12]. A dependency scheme maps each QCNF formula to a binary relation on its variables that represents potential variable dependencies [14, 15]. This relation is used by DepQBF to gain additional freedom in decision making and in the definition of more powerful rules for constraint learning and unit propagation. The latter correspond to a generalization of the *forall-reduction* rule of Q-resolution [2]. Since certificates produced by DepQBF may involve applications of this more general rule, they do not correspond to ordinary Q-resolution proofs. With respect to the two purposes of certificates mentioned above, this has the following consequences.

1. The canonical algorithm for extracting Skolem/Herbrand models from Q-resolution refutations [1] does not work for certificates generated by DepQBF.

2. It is unclear whether certificates can serve as *proofs* of truth or falsity, that is, whether the underlying proof system is sound.

In this paper, we introduce $Q(D)$*-resolution* to study the proof system used by DepQBF to generate proofs. We define the *reflexive resolution-path dependency scheme* D^{rrs} and prove correctness of an algorithm that transforms $Q(D^{\mathrm{rrs}})$-resolution refutations into Q-resolution refutations, thus establishing soundness of $Q(D^{\mathrm{rrs}})$-resolution. Since D^{rrs} is strictly more general than the *standard dependency scheme* D^{std} currently implemented in DepQBF, the soundness result carries over to $Q(D^{\mathrm{std}})$-resolution. We also provide an alternative characterization of *resolution-path dependencies* [16, 18] in terms of directed walks in a formula's *implication graph* which admits an algorithmically more advantageous treatment in terms of strongly connected components. The dependency scheme D^{rrs} is a variant of the *resolution path dependency scheme* D^{res} [16]. We justify our working with D^{rrs} instead of D^{res} by demonstrating that $Q(D^{\mathrm{res}})$-resolution is unsound.

2 Preliminaries

Sequences. We write ε for the empty sequence. If $s = s_1 \dots s_k$ and $r = r_1 \dots r_l$ are sequences then by $s * r$ we denote the sequence $s_1 \dots s_k r_1 \dots r_l$. Let r be a sequence such that $r = p * s$. Then p is a *prefix* of r and s is a *suffix* of r. If t is a prefix (suffix) of r then t is a *proper* prefix (suffix) of r if $t \neq r$. The *length* of a sequence $s_1 \dots s_k$ is k. A *shortest* sequence with property P is a sequence of minimum length with property P. If p_i is a sequence for every $i \in \{1, \dots, k\}$ we use $\langle p_i \rangle_{i=1}^{k}$ as a shorthand for the sequence $p_1 * p_2 * \dots * p_k$. At the cost of introducing some ambiguity we write a for both the singleton set $\{a\}$ and the sequence containing only a.

Graphs. A *directed graph* G is a pair $(V(G), E(G))$ consisting of a finite set $V(G)$ and an irreflexive binary relation $E(G) \subseteq V(G) \times V(G)$. The elements of $V(G)$ and $E(G)$ are called *vertices* and *edges* of G, respectively. If $E(G)$ is symmetric then G is an *undirected graph*. We may write vw or $\{v, w\}$ for an edge $(v, w) \in E(G)$ if G is an undirected graph. A *walk* (in G) from v_1 to v_k is a sequence $v_1 \dots v_k$ such that $v_i \in V(G)$ for every $i \in \{1, \dots, k\}$ and $(v_i, v_{i+1}) \in E(G)$ for every $i \in \{1, \dots, k-1\}$. If $X \subseteq V(G)$ we write $G[X]$ for the *subgraph of G induced by X* with vertex set $V(G[X]) = X$ and edge set $E(G[X]) = E(G) \cap (X \times X)$. We define an equivalence relation $\sim_G$ on $V(G)$ as $v \sim_G w$ if and only if there is a walk from v to w in G and a walk from w to v in G. The equivalence classes of $\sim_G$ are called *strongly connected components* of G.

Trees. A *rooted binary tree* is a directed graph G such that (a) there exists a vertex $r \in V(G)$ (called the *root* of G) such that for every $w \in V(G)$ there is a unique walk from r to w in G, and (b) for every $v \in V(G)$ there are at most two distinct vertices $u, w \in V(G)$ such that $(v, u) \in E(G)$ and $(v, w) \in E(G)$. A *labelled rooted binary tree* is a triple $T = (V(T), E(T), \lambda)$ where $(V(T), E(T))$ is a rooted binary tree and λ is a function with domain $V(T) \cup E(T)$. We say $x \in V(T) \cup E(T)$ is *labelled with* $\lambda(x)$.

Formulas. We consider quantified Boolean formulas in *quantified conjunctive normal form* (QCNF). A QCNF formula is a pair QF, where Q is a (quantifier) *prefix* and F is a *CNF formula*, called the *matrix* of $\mathcal{F}$. A CNF formula is a finite set of *clauses*, where each clause is a finite set of *literals*. Literals are negated or unnegated propositional *variables*. If x is a variable, we put $\overline{x} = \neg x$ and $\overline{\neg x} = x$, and let $var(x) = var(\neg x) = x$. If X is a set of literals, we write $\overline{X}$ for the set $\{\, \overline{\mathtt{a}} : \mathtt{a} \in X \,\}$. For a clause C, we let $var(C)$ be the set of variables occuring (negated or unnegated) in C. For a QCNF formula $\mathcal{F}$ with matrix F, we put $var(\mathcal{F}) = var(F) = \bigcup_{C \in F} var(C)$. We call a clause *tautological* if it contains the same variable negated as well as unnegated. We assume that the matrix of a formula contains only non-tautological clauses. The prefix of a QCNF formula $\mathcal{F}$ is a sequence $Q_1 x_1 \dots Q_n x_n$, where $x_1 \dots x_n$ is a permutation of $var(\mathcal{F})$ and $Q_i \in \{\forall, \exists\}$ for $1 \leq i \leq n$. We define a total order $<_{\mathcal{F}}$ on $var(\mathcal{F})$ by letting $x_i <_{\mathcal{F}} x_j$ if and only if $i < j$. The sets of *existential* and *universal* variables occurring in $\mathcal{F}$ are given by

$var_\exists(\mathcal{F}) = \{ x_i : 1 \leq i \leq n, Q_i = \exists \}$ and $var_\forall(\mathcal{F}) = \{ x_i : 1 \leq i \leq n, Q_i = \forall \}$. Relative to $\mathcal{F}$, a literal a is existential (universal) if $var(a)$ is existential (universal). We define $R_\mathcal{F} = \{ (x, y) : x <_\mathcal{F} y \}$ and let $R_\mathcal{F}(x) = \{ y \in var(\mathcal{F}) : \ (x, y) \in R_\mathcal{F} \}$ for $x \in var(\mathcal{F})$. Moreover, we let $D^{\text{trv}}_\mathcal{F} = \{ (x_i, x_j) \in R_\mathcal{F} : Q_i \neq Q_j \}$. If F is a CNF formula and a is a literal then $F[a] = \{ C \setminus \overline{a} : C \in F, a \notin C \}$. We extend this to QCNF formulas $\mathcal{F} = QF$ by letting $\mathcal{F}[a] = Q'F[a]$, where Q' is obtained from Q by deleting $var(a)$ and its associated quantifier. Let $\mathcal{F}$ be a QCNF formula. If $var(\mathcal{F}) = \emptyset$ then $\mathcal{F}$ is *true* (or *satisfiable*) if $F = \emptyset$. Otherwise, let $\mathcal{F} = Q_x x \, \mathcal{F}'$. If $Q_x = \forall$ then $\mathcal{F}$ is *true* if both $\mathcal{F}'[x]$ and $\mathcal{F}'[\neg x]$ are true. If $Q_x = \exists$ then $\mathcal{F}$ is true if at least one of $\mathcal{F}'[x]$ and $\mathcal{F}'[\neg x_1]$ is true. If $\mathcal{F}$ is not true then $\mathcal{F}$ is *false* (or *unsatisfiable*).

3 Q(D)-Resolution

A *proto-dependency scheme* is a mapping D that associates each QCNF formula $\mathcal{F}$ with a binary relation $D_\mathcal{F} \subseteq D^{\text{trv}}_\mathcal{F}$. The *trivial dependency scheme* is the mapping $D^{\text{trv}} : \mathcal{F} \mapsto D^{\text{trv}}_\mathcal{F}$. A proto-dependency scheme D is *tractable* if the relation $D(\mathcal{F})$ can be computed in polynomial time for each QCNF formula $\mathcal{F}$.

We will represent $Q(D)$-resolution derivations as labelled rooted binary trees constructed by means of the following operations.[1]

- If C is a clause then $\triangle(C)$ denotes a labelled rooted binary tree consisting of a single (root) vertex labelled with C.

- Let $T_1 = (V_1, E_1, \lambda_1)$ and $T_2 = (V_2, E_2, \lambda_2)$ be labelled rooted binary trees. For every literal a, we define the operation $\odot_a$ as follows. We assume without loss of generality that V_1 and V_2 are disjoint. Let r_1 and r_2 denote the roots of T_1 and T_2, respectively, and let $C_1 = \lambda_1(r_1)$ and $C_2 = \lambda_2(r_2)$. Then $T_1 \odot_a T_2$ denotes the labelled rooted binary tree obtained by taking the union of T_1 and T_2, adding a new vertex r labelled with $C = (C_1 \setminus a) \cup (C_2 \setminus \overline{a})$, and making r the root by adding edges (r, r_1) and (r, r_2) labelled with a and $\overline{a}$, respectively.

- Let $T = (V, E, \lambda)$ be a labelled rooted binary tree with root r and $\lambda(r) = C$. For a literal a we construct the labelled rooted binary tree $T\|_a$ starting from T by adding a fresh vertex r' labelled with $C \setminus a$ and an edge (r', r) labelled with a.

Definition 1 (Tree-like $Q(D)$-resolution derivation)**.** Let D be a proto-dependency scheme and let $\mathcal{F} = QF$ be a QCNF formula. A *tree-like $Q(D)$-resolution derivation from $\mathcal{F}$* is a labelled rooted binary tree that can be constructed using the following rules.

1. If $C \in F$ then $\triangle(C)$ is a tree-like $Q(D)$-resolution derivation from $\mathcal{F}$.

[1] Our notation is inspired by [3].

2. Let T_1, T_2 be $Q(D)$-resolution derivations from $\mathcal{F}$ whose roots are labelled with C_1 and C_2, respectively. If $a \in C_1$ is an existential literal such that $\overline{a} \in C_2$, and $(C_1 \setminus a) \cup (C_2 \setminus \overline{a})$ is non-tautological then $T_1 \odot_a T_2$ is a tree-like $Q(D)$-resolution derivation from $\mathcal{F}$.

3. Let T be a $Q(D)$-resolution derivation with root label C. If $a \in C$ is a universal literal and there is no existential literal $b \in var(C)$ such that $(var(a), var(b)) \in D(\mathcal{F})$ then $T\|_a$ is a tree-like $Q(D)$-resolution derivation from $\mathcal{F}$.

Rules 2 and 3 are known as *resolution* and *forall-reduction*, respectively. We will usually refer to tree-like $Q(D)$-resolution derivations as $Q(D)$-*derivations* (or simply as *derivations* if D is clear from the context). Let $T = (V, E, \lambda)$ be a $Q(D)$-derivation from $\mathcal{F}$ with root r. We say that T is a *derivation of* $\lambda(r)$ or that T *derives* $\lambda(r)$, and call $\lambda(r)$ the *conclusion* of T. We call T a *refutation* of $\mathcal{F}$ if T derives the empty clause $\emptyset$. If T_1 and T_2 are derivations from $\mathcal{F}$ of clauses C_1 and C_2 such that $C_1 \subseteq C_2$ we say T_1 *subsumes* T_2. The *size* of T, denoted $|T|$, is defined to be $|V|$. A *position* of T is either a sequence of edge labels occurring on a walk in T starting from r or the empty sequence ε. Let π be a position of T. We let $T[\pi]$ denote the *subderivation of* T *at* π defined as

$$T[\pi] = \begin{cases} T & \text{if } \pi = \varepsilon, \\ T_1[\sigma] & \text{if } T = T_1 \odot_\ell T_2 \text{ and } \pi = \ell * \sigma, \\ T'[\sigma] & \text{if } T = T'\|_\ell \text{ and } \pi = \ell * \sigma. \end{cases}$$

If $T[\pi] = T_1 \odot_\ell T_2$ we refer to $T[\pi]$ as a *resolution step* (on $var(\ell)$); if $T[\pi] = S\|_a$ then $T[\pi]$ is a *forall-reduction step* (on ℓ). Let $\mathcal{F}$ be a QCNF formula, let x be a universal variable of $\mathcal{F}$, and let y be an existential variable of $\mathcal{F}$. Then $(x, y) \in D^{\mathrm{trv}}(\mathcal{F})$ if and only if $x <_{\mathcal{F}} y$. It follows that the forall-reduction rule of "ordinary" Q-resolution [5] corresponds to forall-reduction in $Q(D^{\mathrm{trv}})$-resolution. Accordingly, we define Q-resolution as follows.

Definition 2 (Q-resolution)**.** Let $\mathcal{F}$ be a QCNF formula. A *Q-resolution derivation from* $\mathcal{F}$ is a $Q(D^{\mathrm{trv}})$-derivation from $\mathcal{F}$, and a *Q-resolution refutation of* $\mathcal{F}$ is a $Q(D^{\mathrm{trv}})$-refutation of $\mathcal{F}$.

In spite of its simplicity, Q-resolution is a sound and complete proof system for unsatisfiable QCNF formulas.

Fact 1 ([5])**.** *Let $\mathcal{F}$ be a QCNF formula. There exists a Q-resolution refutation of $\mathcal{F}$ if and only if $\mathcal{F}$ is unsatisfiable.*

Let $\mathcal{F}$ be a QCNF formula, let a be a universal literal, and let b be an existential literal. We say that b *blocks* a (relative to $\mathcal{F}$) if $var(a) <_{\mathcal{F}} var(b)$. We extend this to clauses C and say that C *blocks* a if there is a literal $b \in C$ such that b blocks a. If $T = S\|_a$ is a $Q(D)$-derivation from $\mathcal{F}$ and C blocks a we say that C *blocks* T. In $Q(D)$-derivations, forall-reduction can be applied even in the presence of blocking literals. We refer to such occurrences of forall-reduction as D-*reductions*.

Definition 3 (D-reduction)**.** Let D be a proto-dependency scheme, let $\mathcal{F}$ be a QCNF formula, and let T be a $Q(D)$-derivation from $\mathcal{F}$. Let $T[\pi] = S\|_a$ be a derivation of clause C. If C blocks $S\|_a$ then $S\|_a$ is a *D-reduction* (of T). If, in addition, there is no D-reduction $R\|_b$ of T such that $var(b) <_{\mathcal{F}} var(a)$, then $T[\pi]$ is an *outermost D-reduction* (of T).

A $Q(D)$-derivation that does not contain D-reductions is already a Q-resolution derivation.

4 Dependency Schemes and Q(D)-resolution

In the literature, there are two definitions of *Dependency Scheme* that refine our abstract notion of a proto-dependency scheme:

1. Dependency schemes and so-called *cumulative dependency schemes* that characterize truth-value preserving permutations of a formula's prefix [15].
2. Dependency schemes based on a semantic notion of *independence* [14].

It turns out these notions are too weak to characterize soundness of $Q(D)$-resolution. For dependency schemes of type 2, this can be shown using the following formula:

$$\mathcal{F} = \forall x \forall y \exists z \, \{\{x, y, \neg z\}, \{\neg x, \neg y, z\}\}$$

The formula $\mathcal{F}$ is true if the value assigned to z matches the value assigned to x or the value assigned to y. Accordingly, both $f_z(x, y) = x$ and $f'_z(x, y) = y$ are *models* [6] of $\mathcal{F}$. This implies that z is both independent of x and independent of y in the sense of [14]. Let D be the proto-dependency scheme such that $D(\mathcal{F}) = \emptyset$ and $D(\mathcal{G}) = D^{\text{trv}}(\mathcal{G})$ for every QCNF formula $\mathcal{G} \neq \mathcal{F}$. Then D is a dependency scheme of type 2, but $Q(D)$-resolution is unsound: since $(x, z) \notin D(\mathcal{F})$ and $(y, z) \notin D(\mathcal{F})$, forall-reduction derives the clauses $\{x\}$ and $\{\neg x\}$ which yield a $Q(D)$-resolution refutation of $\mathcal{F}$.

The dependency scheme D in the above example is constructed with the counterexample in mind. For dependency schemes of type 1, we do not have to come up with artificial proto-dependency schemes. We will now show that $Q(D^{\text{res}})$-resolution is unsound, where D^{res} is the *resolution-path dependency scheme* [16, 18] (we give a simplified but equivalent version of the definition in [16]).

Definition 4 (Resolution Path)**.** Let $\mathcal{F} = QF$ be a QCNF formula and let $X \subseteq var_{\exists}(\mathcal{F})$. A *resolution path (from a_1 to a_{2k}) via X (in $\mathcal{F}$)* is a sequence $p = a_1 \dots a_{2k}$ of literals satisfying the following conditions.

1. There is a $C_i \in F$ such that $a_{2i-1}, a_{2i} \in C_i$, for all $i \in \{1, \dots, k\}$.
2. $var(a_{2i-1}) \neq var(a_{2i})$ for all $i \in \{1, \dots, k\}$.
3. $a_{2i}, a_{2i+1} \in X \cup \overline{X}$ for all $i \in \{1, \dots, k-1\}$.
4. $\overline{a_{2i}} = a_{2i+1}$ for all $i \in \{1, \dots, k-1\}$.

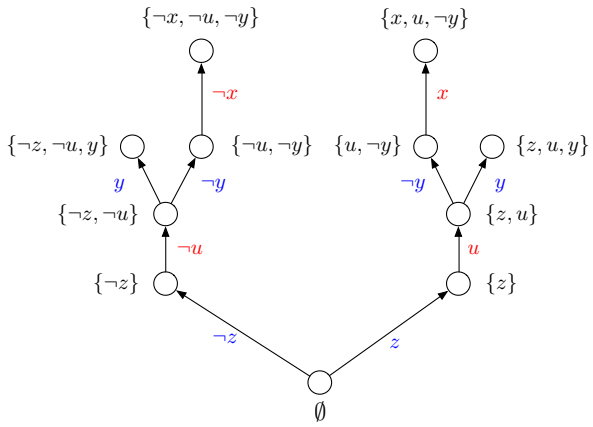


Figure 1: $Q(D^{\text{res}})$-refutation of $\mathcal{F}$ from Example 1.

For every $i \in \{1, \ldots, k\}$ we say that p *goes through* $var(a_{2i})$.

Definition 5 (Resolution-Path Dependency Pair)**.** Let $\mathcal{F}$ be a QCNF formula and let $X \subseteq var(\mathcal{F})$. We call (x, y) a *resolution-path dependency pair* of $\mathcal{F}$ with respect to X if there are literals a and b such that $var(a) = x$ and $var(b) = y$ and there exist resolution paths from a to b and from $\overline{a}$ to $\overline{b}$ via X.

Definition 6 (Resolution-Path Dependency Scheme)**.** The *resolution path dependency scheme* D^{res} maps each QCNF formula $\mathcal{F}$ to the relation $D^{\text{res}}_{\mathcal{F}} = \{\, (x, y) \in R_{\mathcal{F}} : \ (x, y)$ is a resolution-path dependency pair of $\mathcal{F}$ with respect to $(R_{\mathcal{F}}(x) \cap var_{\exists}(\mathcal{F})) \setminus y \,\}$.

The resolution-path dependency scheme is a cumulative dependency scheme [16]. However, the following example (taken from [14]) demonstrates that $Q(D^{\text{res}})$-resolution is unsound.

Example 1. Let $\mathcal{F} = \forall x \exists z \forall u \exists y\, F$, where F contains the clauses $\{x, u, \neg y\}$, $\{\neg x, \neg u, \neg y\}$, $\{z, u, y\}$, $\{\neg z, u, \neg y\}$, $\{\neg z, \neg u, y\}$, and $\{z, \neg u, \neg y\}$.

The formula $\mathcal{F}$ is satisfiable, but Figure 4 shows a $Q(D^{\text{res}})$-refutation of $\mathcal{F}$. It is straightforward to verify that the pair (x, y) is not in $D^{\text{res}}(\mathcal{F})$ since every resolution path from x or $\neg x$ to y goes through y. As a consequence, one can derive the clause $\{u, \neg y\}$ from $\{x, u, \neg y\}$, and the clause $\{\neg u, \neg y\}$ from $\{\neg x, \neg u, \neg y\}$ by forall-reduction in $Q(D^{\text{res}})$-resolution.

5 The Reflexive Resolution-Path Dependency Scheme

Motivated by Example 1, we define the following variant of D^{res} for which resolution paths inducing an (x, y)-dependency may also go through y.

Definition 7 (Reflexive Resolution-Path Dependency Scheme)**.** The *reflexive resolution-path dependency scheme* D^{rrs} maps each QCNF formula $\mathcal{F}$ to the relation $D^{\text{rrs}}_{\mathcal{F}} = \{\, (x, y) \in R_{\mathcal{F}} :$ (x, y) is a resolution-path dependency pair of $\mathcal{F}$ with respect to $R_{\mathcal{F}}(x) \cap var_{\exists}(\mathcal{F}) \,\}$.

Proto-dependency schemes can be partially ordered by a pointwise comparison of the relations they associate with QCNF formulas: we say that a proto-dependency scheme D_1 is *at least as general as* a proto-dependency scheme D_2 if $D_1(\mathcal{F}) \subseteq D_2(\mathcal{F})$ for every QCNF formula $\mathcal{F}$. If this inclusion is strict for some formulas we say D_1 is *strictly more general* than D_2. One can show that D^{rrs} is strictly more general than the *Standard Dependency Scheme* D^{std} used in DepQBF (the following definition is a streamlined version of the definition of D^{std} in [15]).

Definition 8 (Primal Graph)**.** Let $\mathcal{F}$ be a QCNF formula with matrix F. The *primal graph* of $\mathcal{F}$ is the undirected graph with vertex set $var(\mathcal{F})$ and edge set $\{\, xy : x, y \in var(\mathcal{F}), x \neq y,$ and there is a clause $C \in F$ such that $x, y \in var(C) \,\}$.

Definition 9 (Standard Dependency Pair)**.** Let $\mathcal{F}$ be a QCNF formula and let $X \subseteq var(\mathcal{F})$. We call $(x, y) \in D^{\text{trv}}_{\mathcal{F}}$ a *standard dependency pair* of $\mathcal{F}$ with respect to X if there is a walk from x to y in $G[X \cup \{x, y\}]$, where G denotes the primal graph of $\mathcal{F}$.

Definition 10 (Standard Dependency Scheme)**.** The *standard dependency scheme* D^{std} maps every QCNF formula $\mathcal{F}$ to the relation $D^{\text{std}}_{\mathcal{F}} = \{\, (x, y) \in R_{\mathcal{F}} : (x, y)$ is a standard dependency pair of $\mathcal{F}$ with respect to $R_{\mathcal{F}}(x) \cap var_{\exists}(\mathcal{F}) \,\}$.

Proposition 1. *D^{res} is strictly more general than D^{rrs}, and D^{rrs} is strictly more general than D^{std}.*

Proof. By definition, D^{res} is at least as general as D^{rrs}. In combination with Example 1 this implies that D^{res} is strictly more general than D^{rrs}. To see that D^{rrs} is strictly more general than D^{std}, let $\mathcal{F} = QF$ be a QCNF formula and let G denote the primal graph of $\mathcal{F}$. Let $(x, y) \in D^{\text{rrs}}(\mathcal{F})$ and let $X = R_{\mathcal{F}}(x) \cap var_{\exists}(\mathcal{F})$. Then (x, y) is resolution-path dependency pair with respect to X. Assume without loss of generality that there is a resolution path $p = a_1 \dots a_{2k}$ from x to y via X. By condition 1 of Definition 4 there is a $C_i \in F$ such that $a_{2i-1}, a_{2i} \in C_i$ for every $i \in \{1, \dots, k\}$. It follows that $\{var(a_{2i-1}), var(a_{2i})\}$ is an edge in G for every $i \in \{1, \dots, k\}$. Since $\overline{var(a_{2i})} = var(a_{2i+1})$ for every $i \in \{1, \dots, k-1\}$ by condition 4 of Definition 4 the sequence $p' = var(a_1) * \langle var(a_{2i-1})\rangle_{i=1}^{k-1} * var(a_{2k})$ is a walk from x to y in G. Since $var(a_{2i}) \in X$ for every $i \in \{1, \dots, k-1\}$ by condition 3 of Definition 4 the path p' is even a walk in $G[X \cup \{x, y\}]$. So (x, y) is a standard dependency pair with respect to X and thus $(x, y) \in D^{\text{std}}(\mathcal{F})$. This proves that D^{rrs} is at least as general as D^{std}. Let $\mathcal{G} = \forall x \exists y \, \{\{x, y\}\}$. It is not difficult to see that $(x, y) \in D^{\text{std}}(\mathcal{G})$ but $(x, y) \notin D^{\text{res}}(\mathcal{G})$. We conclude that D^{rrs} is strictly more general than D^{std}. □ It is not difficult to see that if D_1 is at least as general as D_2 and $Q(D_1)$-resolution is sound then $Q(D_2)$-resolution is sound as well. Thus soundness of $Q(D^{\text{rrs}})$-resolution (proved in Section 6) carries over to $Q(D^{\text{std}})$-resolution.

We now give an alternative characterization of resolution paths in terms of walks in the *implication graph* of a formula (also known as the formula's *associated graph* [11].)

Definition 11 (Implication graph)**.** Let $\mathcal{F} = QF$ be a QCNF formula. The *implication graph* of $\mathcal{F}$ is the directed graph with vertex set $var(\mathcal{F}) \cup \overline{var(\mathcal{F})}$ and edge set $\{\, (\overline{a}, b) :$ there is a $C \in F$ such that $a, b \in C$ and $a \neq b \,\}$.

Lemma 2. *Let $\mathcal{F}$ be a QCNF formula and let $a, b \in var(\mathcal{F}) \cup \overline{var(\mathcal{F})}$ be distinct literals. Let $X \subseteq var(\mathcal{F})$ and let G denote the implication graph of $\mathcal{F}$. The following statements are equivalent.*

1. *There is a resolution path from a to b via X.*
2. *There is a walk from $\overline{a}$ to b in $G[X \cup \overline{X} \cup \{\overline{a}, b\}]$.*

Proof. Let $p = a_1 \dots a_{2k}$ be a resolution path from a to b via X. The sequence $p' = \overline{a_1} * \langle a_{2i} \rangle_{i=1}^{k}$ is a walk in the implication graph of $\mathcal{F}$. We have $\{a_2, \dots, a_{2k-1}\} \subseteq X \cup \overline{X}$ by Definition 4, so p is even a walk in $G[X \cup \overline{X} \cup \{\overline{a}, b\}]$. For the converse, let $p = a_1 \dots a_k$ be a shortest walk from $\overline{a}$ to b in $G[X \cup \overline{X} \cup \{\overline{a}, b\}]$. Then the sequence $p' = \overline{a_1} * \langle a_i \overline{a_i} \rangle_{i=2}^{k-1} * a_k$ is a resolution path in $\mathcal{F}$. Since p is a shortest walk we have $\overline{a}, b \notin \{a_2, \dots, a_{k-1}\}$. This implies $\{a_2, \dots, a_{k-1}\} \subseteq X \cup \overline{X}$ as well as $\overline{\{a_2, \dots, a_{k-1}\}} \subseteq X \cup \overline{X}$. So p' is a resolution path from $\overline{a_1} = a$ to b via X. □

The implication graph of a formula can be constructed in time quadratic in the size of $\mathcal{F}$ and directed reachability can be decided in linear time, so we obtain the following result.

Proposition 3. *Both D^{res} and D^{rrs} are tractable.*

For practical purposes the explicit construction of the implication graph can be avoided. Moreover, the following result shows that an overapproximation of D^{rrs} can be represented in terms of the strongly connected components of the implication graph.

Proposition 4. *Let $\mathcal{F}$ be a QCNF formula and let $(x, y) \in D^{\mathrm{trv}}(\mathcal{F})$. Let $X \subseteq var(\mathcal{F})$ and let G denote the implication graph of $\mathcal{F}$. If (x, y) is a resolution-path dependency pair with respect to X there is strongly connected component $\mathcal{C}$ of $G[X \cup \overline{X} \cup \{x, y, \neg y\}]$ such that $x, y \in \mathcal{C}$ or $x, \neg y \in \mathcal{C}$.*

Proof. Let (x, y) be a resolution-path dependency pair with respect to X. Assume without loss of generality that there are resolution paths p_1 and p_2 such that p_1 is a resolution path from $\neg x$ to y via X and p_2 is a resolution path from $\neg y$ to x via X. By Lemma 2 there are walks p_1' from x to y and p_2' from y to x in $G[X \cup \overline{X} \cup \{x, y\}]$, so x and y are in the same strongly connected component of $G[X \cup \overline{X} \cup \{x, y, \neg y\}]$. □

6 Soundness of $Q(D^{\mathrm{rrs}})$-resolution

This section is devoted to proving our main result, stated below.

Theorem 5. *For every QCNF formula $\mathcal{F}$ there is a $Q(D^{\mathrm{rrs}})$-resolution refutation of $\mathcal{F}$ if and only if $\mathcal{F}$ is unsatisfiable.*

In fact, we are going to prove the following, stronger statement.

Proposition 6. *Given a QCNF formula $\mathcal{F}$ and a $Q(D^{\mathrm{rrs}})$-refutation T of $\mathcal{F}$, one can compute a Q-resolution refutation of $\mathcal{F}$ of size at most $3^{|T|}$.*

We prove this proposition by demonstrating correctness and termination of an algorithm (Algorithm 3) that turns $Q(D^{\text{rrs}})$-refutations into Q-resolution refutations in Lemma 13. An outline of this algorithm is given below.

Algorithm outline. Let a be the universal literal removed by an outermost D-reduction of the input $Q(D)$-derivation. There are two cases.

1. If there is no clause containing $\overline{a}$ on the path from the D-reduction to the root of the derivation, we simply skip the D-reduction and add it at the root. If the clause derived at the root does not contain any literals blocking a, the D-reduction is turned into an ordinary forall-reduction. (This condition is satisfied by a refutation, and we can ensure that it holds for subderivations and their outermost D-reductions as well.)

2. Otherwise, the derivation must contain a resolution step on a variable x such that $x <_{\mathcal{F}} \mathit{var}(a)$ (see Lemma 12). We drop the lowermost such resolution step to the root of the derivation. This may introduce x-literals to the clauses on the path from the resolution step to the root. But since the D-reduction picked in the first step is outermost these literals will not interfere with D-reductions. Moreover, because the resolution step is lowermost, every clause on the path contains an existential variable y such that $\mathit{var}(a) <_{\mathcal{F}} y$, so introducing x to these clauses will not turn a forall-reduction into a D-reduction.

In this way, we obtain a derivation whose immediate subderivations are (a) strictly smaller than the original derivation and which (b) do not contain new D-reductions. We run the algorithm on these subderivations to rewrite them into Q-resolution derivations and add a final resolution or forall-reduction step.

Example 2. Consider the QCNF formula $\mathcal{F} = \exists e_1 \forall u \exists e_2 \exists e_3\ \{\{u, e_2\},\ \{\neg u, e_3\},\ \{\neg e_3, e_1\},\ \{\neg e_1, \neg e_2\}\}$. Derivation T_A of Figure 2 shows a $Q(D^{\text{rrs}})$-refutation of $\mathcal{F}$ with D-reductions at positions e_2 and $\neg e_2$. At the root of the derivation case 1 applies, so we skip the D-reduction $T_A[e_2]$ and add it at the root of the derivation, which leads to derivation T_B. We continue with the subderivation $T_B[u]$. Since u occurs on the path from the D-reduction $T_B[u\neg e_2]$ to the root, case 2 applies and we drop the resolution step $T_B[u\neg e_2\neg u\neg e_3]$ on $e_1 <_{\mathcal{F}} u$ to the root, resulting in the Q-resolution derivation T_C.

Example 2 also illustrates that known rewrite strategies for removing long-distance resolution steps from Q-resolution proofs [1, 8] cannot be applied to remove D-reductions from $Q(D)$-resolution refutations. If a long-distance resolution step leads to a clause containing a universal variable u in both polarities, one can assume that the variable resolved on does not block u. In a refutation, the literals blocking u have to be resolved out eventually, so one can remove the long-distance resolution step by successively lowering it [1] or by (recursively) resolving out blocking literals using clauses resolved closer to the root of the derivation [8]. In refutation T_A, resolving $\{u, e_2\}$ and $\{\neg u, \neg e_2\}$ would amount to a long-distance resolution step. But e_2 is both the variable resolved on and the variable blocking u in the premises, and we cannot further lower this resolution step.

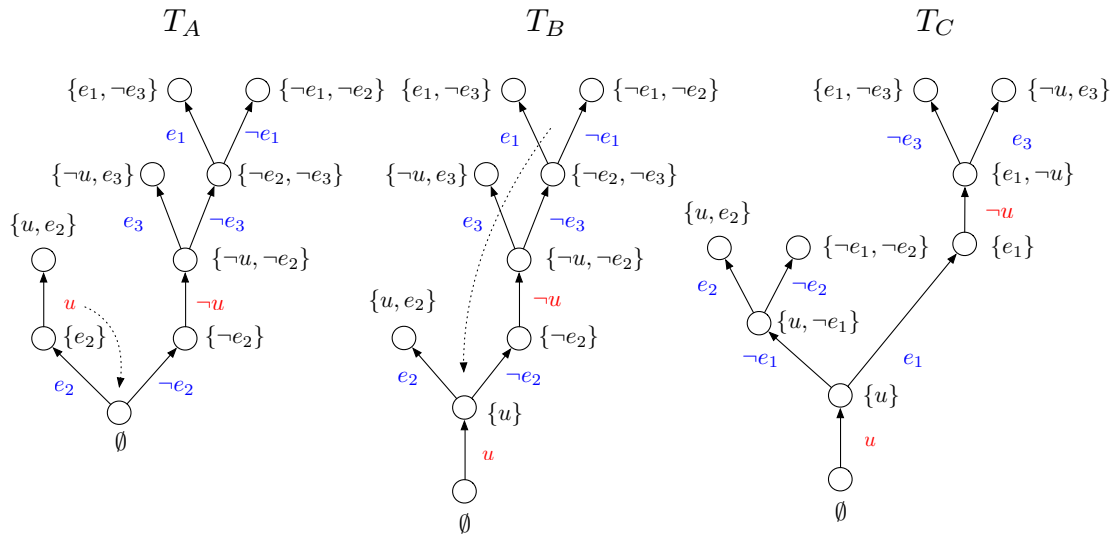


Figure 2: Rewriting a $Q(D^{\text{rrs}})$-resolution refutation (Example 2).

We now turn to a formal proof of Proposition 6. In order to state Algorithm 3 and prove its correctness and termination, we are going to define and characterize the following two operations:

1. Substitution (Definition 14 and Lemma 7).

2. Dropping a resolution step (Algorithm 2 and Lemma 11).

The second operation can in turn be represented by a successive "lowering" of a resolution step (Algorithm 1, Lemmas 9 and 10). This lowering operation essentially corresponds to the rewrite rules presented in [1] for turning long-distance resolution proofs into ordinary Q-resolution proofs.[2]

For the remainder of this section, let D be an arbitrary but fixed proto-dependency scheme, and let $\mathcal{F}$ be an arbitrary but fixed QCNF formula. In order to make formal statements to the effect that operations "do not create new D-reductions", we introduce the notion of *narrowing*.

Definition 12 (Narrowing)**.** Let $R_1 = S_1\|_a$ and $R_2 = S_2\|_a$ be $Q(D)$-derivations from $\mathcal{F}$ with conclusions C_1 and C_2, respectively. We write $R_1 \preceq R_2$ if every literal in C_1 that blocks a is contained in C_2. For $Q(D)$-derivations T_1 and T_2 we say that T_1 *narrows* T_2 if, for every forall-reduction step $T_1[\pi_1]$, there is a forall-reduction step $T_2[\pi_2]$ such that $T_1[\pi_1] \preceq T_2[\pi_2]$.

The narrowing relation defines a preorder on derivations. Throughout the rewriting process, we make sure that intermediate derivations narrow earlier ones, so as to not introduce "new" or "more complicated" D-reductions. The following statements are easily proved using the

[2]The cases covered in lines 5-8 of Algorithm 1 are not explicitly dealt with in [1].

definition of narrowing. Rewriting a derivation may cause literals to disappear from the derivation's conclusion, so that subsequent resolution or forall-reduction steps may become inapplicable. To suppress explicit case distinctions needed for situations of this kind we define "lazy" versions of resolution and forall-reduction as follows (cf. [9]).

Definition 13. Let T_1, T_2, and T be $Q(D)$-derivations from $\mathcal{F}$ of clauses C_1, C_2, and C.

$$T_1 \odot_a^L T_2 = \begin{cases} T_1 \odot_a T_2 & \text{if } a \in C_1 \text{ and } \overline{a} \in C_2, \\ T_1 & \text{if } a \notin C_1, \\ T_2 & \text{if } \overline{a} \notin C_2. \end{cases}$$

$$T\|_a^L = \begin{cases} T\|_a & \text{if } a \in C, \\ T & \text{otherwise.} \end{cases}$$

Definition 14 (Substitution)**.** Let T and S be $Q(D)$-derivations from $\mathcal{F}$. For a position π of T we define $T[\pi \leftarrow S]$ recursively as follows.

$$T[\pi \leftarrow S] = \begin{cases} S & \text{if } \pi = \varepsilon, \\ T_1[\sigma \leftarrow S] \odot_a^L T_2 & \text{if } T = T_1 \odot_a T_2 \text{ and } \pi = a * \sigma, \\ T'[\sigma \leftarrow S]\|_a^L & \text{if } T = T'\|_a \text{ and } \pi = a * \sigma. \end{cases}$$

Definition 15. Let T be a $Q(D)$-derivation from $\mathcal{F}$, let π is a position of T, and let a be a literal. We say that T *does not contain a below π* if, for every proper prefix σ of π, the conclusion of $T[\sigma]$ does not contain a.

Lemma 7. *Let T be a $Q(D)$-derivation from $\mathcal{F}$ of a clause C such that $T[\pi] = S\|_a$. If T does not contain $\overline{a}$ below π then $T[\pi \leftarrow S]$ is a $Q(D)$-derivation from $\mathcal{F}$ of a clause $C' \subseteq C \cup a$ and $T[\pi \leftarrow S]$ narrows T.*

Proof. The derivation $T[\pi \leftarrow S]$ simply omits the forall-reduction step on a, introducing a to clauses on the path from π to the root of the derivation (not necessarily all the way to the root, since there may be another forall-reduction step on a). By assumption, T does not contain $\overline{a}$ below π, so the result will be a $Q(D)$-derivation. □

Lemma 8. *Let T and S be $Q(D)$-derivations from $\mathcal{F}$. Let π be a position such that S subsumes and narrows $T[\pi]$. Then $T[\pi \leftarrow S]$ subsumes and narrows T.*

Proof. One can prove that $T[\pi \leftarrow S]$ subsumes T by an induction on the length of π. For the narrowing part, observe that every forall-reduction step $R\|_a$ of $T[\pi \leftarrow S]$ not already in T occurs in S or on the path from S to the root of $T[\pi \leftarrow S]$. If $R\|_a = S[\sigma]$ there is position ρ of $T[\pi]$ such that $S[\sigma] \preceq T[\pi][\rho]$ since S narrows $T[\pi]$. Otherwise, R derives a clause subsuming the clause to which the corresponding reduction is applied in T. □

For the lowering operation, we distinguish two cases based on whether the resolution step is lowered past a forall-reduction step (Lemma 9) or another resolution step (Lemma 10).

1 **Function** `lower`(T, b)
  input : A $Q(D)$-derivation T and a literal b.
2 **if** $T = (T_1 \odot_a T_2) \odot_b T_3$ **then**
3 **let** C_i *be the conclusion of* T_i *for* $i \in \{1, 2, 3\}$
4 **if** $a \notin C_3$ *and* $\overline{a} \notin C_3$ **then**
5 **return** $(T_1 \odot_b^L T_3) \odot_a (T_2 \odot_b^L T_3)$
6 **else if** $a \in C_3$ **then**
7 **return** $T_1 \odot_b^L T_3$
8 **else**
9 **return** $T_2 \odot_b^L T_3$
10 **else if** $T = (T_1 \odot_a T_2)\|_b$ **then**
11 **return** $T_1\|_b^L \odot_a T_2\|_b^L$
12 **else**
13 **return** T

Algorithm 1: Lowering a resolution step.

Lemma 9. *Let $T = (T_1 \odot_a T_2)\|_b$ be a $Q(D)$-derivation from $\mathcal{F}$ such that a does not block b. Then* `lower`(T, b) *is a $Q(D)$-derivation from $\mathcal{F}$ that subsumes and narrows T.*

Proof. It is readily verified that `lower`(T, b) is a $Q(D)$-derivation that subsumes T. The derivation `lower`(T, b) may contain new forall-reduction steps $T_1\|_b$ and $T_2\|_b$, but since a does not block b we have $T_1\|_b \preceq (T_1 \odot_a T_2)\|_b$ and $T_2\|_b \preceq (T_1 \odot_a T_2)\|_b$, so `lower`$(T, b)$ narrows T. □

Lemma 10. *Let $T = (T_1 \odot_a T_2) \odot_b T_3$ be a $Q(D)$-derivation of from $\mathcal{F}$. Then* `lower`(T, b) *is a $Q(D)$-derivation of from $\mathcal{F}$ that subsumes and narrows T.*

Proof. The case distinction in lines 2-8 of Algorithm 1 is exhaustive, and for each case the derivation returned subsumes the original derivation. Every forall-reduction step of the resulting derivation is already present in T, so `lower`(T, b) narrows T. □

Definition 16. Let T be a $Q(D)$-resolution derivation from $\mathcal{F}$, let a be an existential literal, and let π be a position of T. We say that *a does not block in T below π* if, for every prefix ρ of π, whenever $T[\rho] = S\|_b$ then a does not block b.

Lemma 11. *Let T be a $Q(D)$-derivation from $\mathcal{F}$, and let $T[\pi]$ be a resolution step on a such that a does not block in T below π. Then $T' =$* `drop`(T, π, a) *is a $Q(D)$-derivation from $\mathcal{F}$ that subsumes and narrows T, and at least one of the following conditions*[3] *holds.*

[3]The proof of Lemma 13 is by induction on the size of the input derivation. These conditions ensure that we can apply the induction hypothesis after dropping a resolution step.

```
1  Function drop(T, π, a)
       input : A Q(D)-derivation T, a position π of T, and a literal a.
2      if π = ε then
3          return T
4      else if π = b ∗ ρ then
5          R := drop(T[b], ρ, a)
6          S := T[b ← R]
7          if R = S1 ⊙a S2 then
8              return lower(S, b)
9          else
10             return S
```

Algorithm 2: "Dropping" a resolution step.

1. $T' = T_1 \odot_a T_2$, *and* $|T_1| < |T|$ *as well as* $|T_2| < |T|$
2. $|T'| < |T|$

Proof. We proceed by induction on the length of π. The base case is trivial. For the inductive case, we use the induction hypothesis in line 5 and Lemma 8 in line 6 to conclude that $S = T[b \leftarrow R]$ subsumes and narrows $T[b]$. If $R \neq S_1 \odot_a S_2$ then $|R| < |T[b]|$ by induction hypothesis and $|\mathtt{drop}(T, \pi, a)| = |T[b \leftarrow R]| < |T|$. Otherwise, $|S_1| < |T[b]|$ and $|S_2| < |T[b]|$ by induction hypothesis. Suppose S is a forall-reduction step. Since a does not block in T below π it follows from Lemma 9 that $\mathtt{lower}(S, b) = S_1\|_b^L \odot_a S_2\|_b^L$ is a $Q(D)$-resolution derivation that subsumes and narrows T, and $|S_1\|_b^L| < |T|$ as well as $|S_2\|_b^L| < |T|$. Now suppose S is a resolution step. Then $\mathtt{lower}(S, b)$ is a $Q(D)$-resolution derivation that subsumes and narrows T by Lemma 10, and it is straightforward to verify that the derivation satisfies one of the above conditions. □

Lemma 12. *Let T be a $Q(D^{\mathrm{rrs}})$-derivation from $\mathcal{F}$ with conclusion C. If $T[\pi]$ is a forall-reduction step on literal a and $\overline{a} \in C$ then there is a position π of T such that $T[\sigma]$ is a resolution step on a variable x with $x <_{\mathcal{F}} var(a)$.*

Sketch. The proof is by an induction on the length of π, using the following claim (cf. [16, 18]).

Claim 1. Let S be a $Q(D)$-derivation from $\mathcal{F}$ with conclusion E, and let $resvar(S)$ denote the set of variables resolved on in S. If $a, b \in E$ are distinct literals, there is a resolution path from a to b via $resvar(S)$.

Suppose $|\pi| = 1$ and let $\pi = b$. Then $T = T_1 \odot_b T_2$ is a resolution step since the conclusion of $T[\pi * a]$ is non-tautological. Let C_1 be the clause derived by $T[\pi * a] = T[ba]$ and let C_2 be the clause derived by T_2. Then $a, b \in C_1$ and $\overline{a, b} \in C_2$. It follows from the above claim that there are resolution paths from a to b and from $\overline{a}$ to $\overline{b}$ via $resvar(T)$. That is,

```
1  Function normalize(F, T)
       input : A QCNF formula F and a Q(D^rrs)-derivation T from F.
2      if T does not contain a D-reduction then
3          return T
4      else if T = S‖_a then
5          return normalize(F, S)‖_a^L
6      else if T = T_1 ⊙_a T_2 then
7          let T[π] = T'‖_b be an outermost D-reduction of T
8          if T does not contain b̄ below π then
9              S := T[π ← T']
10             return normalize(F, S)‖_b^L
11         else
12             let σ be a shortest position such that T[σ] = R_1 ⊙_c R_2 and var(c) <_F var(b)
13             S := drop(T, σ, c)
14             if S ≠ S_1 ⊙_c S_2 then
15                 return normalize(F, S)
16             else
17                 return normalize(F, S_1) ⊙_c^L normalize(F, S_2)
```

Algorithm 3: Converting $Q(D^{\mathrm{rrs}})$-derivations to Q-resolution derivations.

$(var(a), var(b))$ is a resolution-path dependency pair of $\mathcal{F}$ with respect to $resvar(T)$. If $resvar(T) \subseteq R_{\mathcal{F}}(var(a))$ then $(var(a), var(b)) \in D^{\mathrm{rrs}}(\mathcal{F})$, a contradiction. Thus there must be an $x \in resvar(T)$ such that $x <_{\mathcal{F}} var(a)$. The inductive case is proved by a straightforward generalization of this argument. □

Lemma 13. *Let T be a $Q(D^{\mathrm{rrs}})$-derivation of C from $\mathcal{F}$ such that C does not block a D-reduction of T. Then* normalize$(\mathcal{F}, T)$ *returns a Q-resolution derivation of size at most $3^{|T|}$ that subsumes T.*

Proof. By induction on the size of T. If T consists of a single node it does not contain a D-reduction, and the algorithm simply returns T (line 2). Suppose the lemma holds for derivations of size strictly less than $|T|$. If T does not contain a D-reduction it is already a Q-resolution derivation (line 2). Otherwise, if T ends with a forall-reduction step the induction hypothesis implies that the derivation returned in line 5 is a Q-resolution derivation from $\mathcal{F}$ that subsumes T. Suppose T ends with a resolution step (line 6). There must be some outermost D-reduction $T[\pi] = T'\|_b$ of T (line 7). There are two cases. (a) If T does not contain $\bar{b}$ below π (line 8) then by Lemma 7 the derivation $S = T[\pi \leftarrow T']$ narrows T and derives a clause $C' \subseteq C \cup b$. Since by assumption C does not block a D-reduction of T

the clause C' does not block a D-reduction of S. Moreover $|S| < |T|$, so we can apply the induction hypothesis and conclude that $\texttt{normalize}(\mathcal{F}, S)$ is a Q-resolution derivation that subsumes S. It follows that $\texttt{normalize}(\mathcal{F}, S)\|_b^L$ is a Q-resolution derivation that subsumes T (line 10). (b) If T contains $\bar{b}$ below π then by Lemma 12 there must be a (shortest) position σ of T such that $T[\sigma]$ is a resolution step on $var(c)$ for some literal c, and $var(c) <_{\mathcal{F}} var(b)$ (line 12). We claim that c does not block in T below σ. Towards a contradiction assume that there is a proper prefix ψ of σ such that $T[\psi] = R\|_d$ and $var(d) <_{\mathcal{F}} var(c)$. By choice of σ the conclusion of R must contain an existential literal e such that $var(b) <_{\mathcal{F}} var(e)$. Thus $var(d) <_{\mathcal{F}} var(c) <_{\mathcal{F}} var(b) <_{\mathcal{F}} var(e)$ and e blocks d, so $R\|_d$ must be a D-reduction. But $T[\pi] = T'\|_b$ is an outermost D-reduction of T and $var(d) <_{\mathcal{F}} var(b)$, a contradiction. So c does not block in T below σ. Thus by Lemma 11 the derivation $S = \texttt{drop}(T, \sigma, c)$ subsumes and narrows T (line 13). Since $T'\|_b$ is an outermost D-reduction of T and $c <_{\mathcal{F}} var(b)$, neither c nor $\neg c$ block a D-reduction of S because S narrows T. If $S \neq S_1 \odot_c S_2$ then $|S| < |T|$ by Lemma 11 and by induction hypothesis $\texttt{normalize}(\mathcal{F}, S)$ is a Q-resolution derivation that subsumes T (line 15). Otherwise $|S_1| < |T|$ and $|S_2| < |T|$ by Lemma 11 and the conclusions of S_1 and S_2 do not block D-reductions of S by choice of c. By induction hypothesis $\texttt{normalize}(\mathcal{F}, S_1)$ and $\texttt{normalize}(\mathcal{F}, S_2)$ are Q-resolution derivations subsuming S_1 and S_2, respectively, so $\texttt{normalize}(\mathcal{F}, S_1) \odot_c^L \texttt{normalize}(\mathcal{F}, S_2)$ is a Q-resolution derivation that subsumes T (line 17). It is easy to verify that the output of $\texttt{normalize}(\mathcal{F}, T)$ satisfies the size bound claimed in the statement of the lemma. □

Proof of Proposition 6. Immediate from Lemma 13 and the observation that the empty clause cannot not block a D-reduction. □

7 Conclusion

We proposed and studied $Q(D)$-resolution, a generalization of Q-resolution, to capture the certificates generated by DepQBF. We introduced the reflexive resolution-path dependency scheme D^{rrs}, proved soundness of $Q(D^{\text{rrs}})$-resolution, and provided an alternative characterization of resolution paths that lends itself to an efficient implementation. In this manner, we hope to have created a solid theoretical basis for future incarnations of DepQBF using dependency schemes more general than D^{std}.

QBF solvers based on quantifier expansion can also benefit from an analysis of variable dependencies [4, 12, 14, 15]. Recently, Janota et al. [10] introduced a proof system to capture the behavior of these solvers. We plan to study how such systems can be augmented with dependency schemes as part of future work. Another intriguing topic for further research is the relative complexity of Q-resolution and $Q(D)$-resolution.

References

[1] Valeriy Balabanov and Jie-Hong R. Jiang. Unified QBF certification and its applications. *Formal Methods in System Design*, 41(1):45–65, 2012.

[2] Armin Biere and Florian Lonsing. Integrating dependency schemes in search-based QBF solvers. In Ofer Strichman and Stefan Szeider, editors, *Theory and Applications of Satisfiability Testing - SAT 2010*, volume 6175 of *Lecture Notes in Computer Science*, pages 158–171. Springer Verlag, 2010.

[3] Joseph Boudou and Bruno Woltzenlogel Paleo. Compression of propositional resolution proofs by lowering subproofs. In Didier Galmiche and Dominique Larchey-Wendling, editors, *Automated Reasoning with Analytic Tableaux and Related Methods - Tableaux 2013*, volume 8123 of *Lecture Notes in Computer Science*, pages 59–73. Springer Verlag, 2013.

[4] Uwe Bubeck. *Model-based transformations for quantified Boolean formulas*. PhD thesis, University of Paderborn, 2010.

[5] H. Kleine Büning, M. Karpinski, and A. Flögel. Resolution for quantified Boolean formulas. *Information and Computation*, 117(1):12–18, 1995.

[6] Hans Kleine Büning, K. Subramani, and Xishun Zhao. Boolean functions as models for quantified Boolean formulas. *Journal of Automated Reasoning*, 39(1):49–75, 2007.

[7] Marco Cadoli, Marco Schaerf, Andrea Giovanardi, and Massimo Giovanardi. An algorithm to evaluate Quantified Boolean Formulae and its experimental evaluation. *Journal of Automated Reasoning*, 28(2), 2002.

[8] Enrico Giunchiglia, Massimo Narizzano, and Armando Tacchella. Clause/term resolution and learning in the evaluation of Quantified Boolean Formulas. *J. Artif. Intell. Res.*, 26:371–416, 2006.

[9] Alexandra Goultiaeva, Allen Van Gelder, and Fahiem Bacchus. A uniform approach for generating proofs and strategies for both true and false QBF formulas. In Toby Walsh, editor, *Proceedings of IJCAI 2011*, pages 546–553. IJCAI/AAAI, 2011.

[10] Mikolás Janota and Joao Marques-Silva. On propositional QBF expansions and Q-resolution. In Matti Järvisalo and Allen Van Gelder, editors, *Theory and Applications of Satisfiability Testing - SAT 2013*, volume 7962 of *Lecture Notes in Computer Science*, pages 67–82. Springer Verlag, 2013.

[11] Hans Kleine Büning and Theodor Lettman. *Propositional logic: deduction and algorithms*. Cambridge University Press, Cambridge, 1999.

[12] Florian Lonsing. *Dependency Schemes and Search-Based QBF Solving: Theory and Practice*. PhD thesis, Johannes Kepler University, Linz, Austria, April 2012.

[13] 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*, volume 7317 of *Lecture Notes in Computer Science*, pages 430–435. Springer Verlag, 2012.

[14] Marko Samer. Variable dependencies of quantified CSPs. In Iliano Cervesato, Helmut Veith, and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 15th International Conference, LPAR 2008, Doha, Qatar, November 22-27, 2008. Proceedings*, volume 5330 of *Lecture Notes in Computer Science*, pages 512–527. Springer Verlag, 2008.

[15] Marko Samer and Stefan Szeider. Backdoor sets of quantified Boolean formulas. *Journal of Automated Reasoning*, 42(1):77–97, 2009.

[16] Friedrich Slivovsky and Stefan Szeider. Computing resolution-path dependencies in linear time. In Alessandro Cimatti and Roberto Sebastiani, editors, *Theory and Applications of Satisfiability Testing - SAT 2012*, volume 7317 of *Lecture Notes in Computer Science*, pages 58–71. Springer Verlag, 2012.

[17] Larry J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time. In *Proc. Theory of Computing*, pages 1–9. ACM, 1973.

[18] Allen Van Gelder. Variable independence and resolution paths for quantified Boolean formulas. In Jimmy Lee, editor, *Principles and Practice of Constraint Programming - CP 2011*, volume 6876 of *Lecture Notes in Computer Science*, pages 789–803. Springer Verlag, 2011.