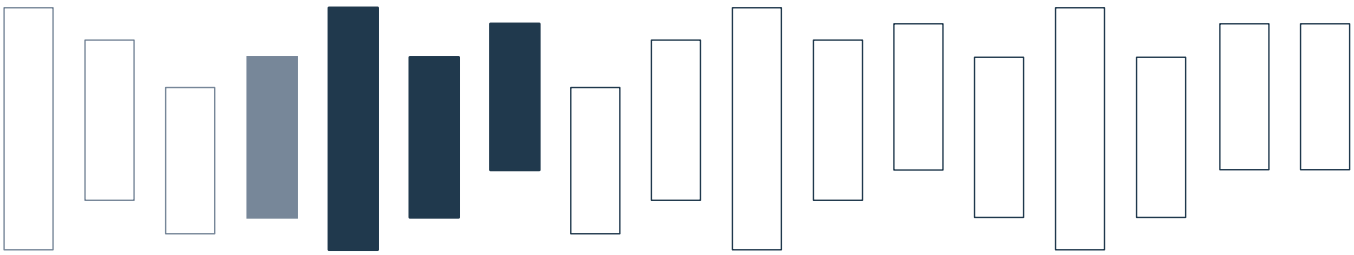
Algorithms and Complexity Group | Institute of Logic and Computation | TU Wien, Vienna, Austria

Technical Report AC-TR-18-003

July 2018

Polynomial-Time Validation of QCDCL Certificates

Tomáš Peitl, Friedrich Slivovsky, and Stefan Szeider



This is the authors' copy of a paper that appeared in the proceedings of SAT 2018, the 35th International Conference on Theory and Applications of Satisfiability Testing, held as part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018. LNCS 10929, pp. 253–269, 2018, DOI: 10.1007/978-3-319-94144-8_16.
www.ac.tuwien.ac.at/tr

Polynomial-Time Validation of QCDCL Certificates*

Tomáš Peitl, Friedrich Slivovsky, and Stefan Szeider

Algorithms and Complexity Group, TU Wien, Vienna, Austria
{peitl,fslivovsky,sz}@ac.tuwien.ac.at

Abstract. Quantified Boolean Formulas (QBFs) offer compact encodings of problems arising in areas such as verification and synthesis. These applications require that QBF solvers not only decide whether an input formula is true or false but also output a witnessing certificate, i.e. a representation of the winning strategy. State-of-the-art QBF solvers based on Quantified Conflict-Driven Constraint Learning (QCDCL) can emit Q-resolution proofs, from which in turn such certificates can be extracted. The correctness of a certificate generated in this way is validated by substituting it into the matrix of the input QBF and using a SAT solver to check that the resulting propositional formula (the *validation formula*) is unsatisfiable. This final check is often the most time-consuming part of the entire certification workflow. We propose a new validation method that does not require a SAT call and provably runs in polynomial time. It uses the Q-resolution proof from which the given certificate was extracted to directly generate a (propositional) proof of the validation formula in the RUP format, which can be verified by a proof checker such as DRAT-trim. Experiments with a prototype implementation show a robust, albeit modest, increase in the number of successfully validated certificates compared to validation with a SAT solver.

1 Introduction

Quantified Boolean Formulas (QBFs) offer succinct encodings for problems from domains such as formal verification, synthesis, and planning [3, 5, 7, 15, 21, 22]. Even though SAT-based approaches to these problems are generally still superior, the evolution of QBF solvers in recent years is starting to tip the scales in their favor [9]. In most of these applications, it is required that QBF solvers not only output a simple true/false answer but also produce a *strategy*, or *certificate*, that shows how this answer can be realized. For example, a certificate might encode a counterexample to the soundness of a software system, or a synthesized program.

Most state-of-the-art QBF solvers have the ability to generate such certificates, and some recently developed solvers have been explicitly designed with certification in mind [19, 20, 23]. Search-based solvers implementing Quantified

* This research was partially supported by FWF grants P27721 and W1255-N23.

Conflict-Driven Constraint Learning (QCDCL) [6, 26] can output Q-resolution proofs [4, 17, 18], from which in turn certificates can be extracted in linear time [1, 2].

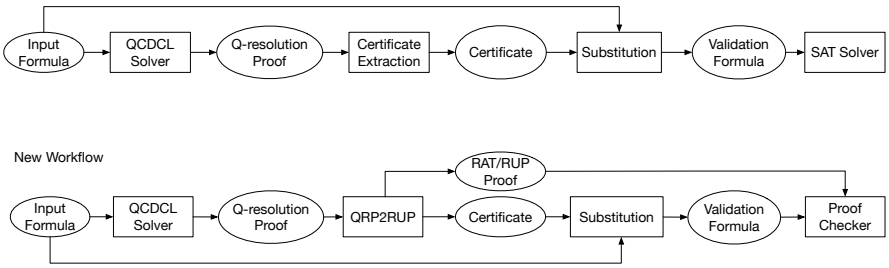


Fig. 1. Certificate extraction and validation for QCDCL solvers.

Since QBF solvers and (to a lesser degree) certificate extraction tools are complex pieces of software that may contain bugs, certificates obtained in this way ought to be independently *validated*. This can be achieved by substituting the certificate back into the matrix of the input QBF and using a SAT solver to check that the resulting propositional formula (which we call the *validation formula*) is unsatisfiable [17]. This certification workflow is illustrated in the top half of Figure 1. Once a certificate is validated, we can essentially trust its correctness as much as we trust in the correctness of the SAT solver used for validation.[1] However, since certificates tend to be large, the corresponding SAT call frequently amounts to the most time-consuming step in the entire certification workflow and even causes timeouts [17].

In this paper, we propose an alternative validation method for QCDCL that avoids this SAT call. Instead, it uses the Q-resolution proof from which the given certificate was extracted to generate a proof of the validation formula in the RUP format [12], whose correctness can then be verified by a propositional proof checker such as DRAT-trim [25]. This workflow is sketched in the lower half of Figure 1. Since this RUP proof can be computed from the Q-resolution proof in linear time and checked in polynomial time, we obtain a validation procedure that provably runs in polynomial time.

We implemented this new validation method in a tool named QRP2RUP and tested it on benchmark instances from several recent QBF evaluations. Our experiments show a robust, albeit modest, increase in the number of successfully validated certificates compared to validation with a SAT solver.

[1] We still have to make sure that the validation formula is constructed correctly so that it is not trivially unsatisfiable. We discuss this issue in Section 8.

2 Preliminaries

A *literal* is a negated or unnegated variable. If x is a variable, we write $\overline{x} = \neg x$ 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 $\{\, \overline{x} : x \in X \,\}$ and let $var(X) = \{\, var(\ell) : \ell \in X \,\}$. An *assignment* to a set X of variables is a mapping $\tau : X \to \{true, false\}$. An assignment σ is an extension of the assignment τ if σ assigns all variables that τ does, and to the same polarity. We extend assignments $\tau : X \to \{true, false\}$ to literals by letting $\tau(\neg x) = \neg\tau(x)$ for $x \in X$.

We consider Boolean *circuits* over $\{\neg, \wedge, \vee, false, true\}$ and write $var(\varphi)$ for the set of variables occurring in a circuit φ. If φ is a circuit and τ an assignment, $\varphi[\tau]$ denotes the circuit obtained by replacing each variable $x \in X \cap var(\varphi)$ by $\tau(x)$ and propagating constants. A circuit φ is *satisfiable* if there is an assignment τ such that $\varphi[\tau] = true$, otherwise it is *unsatisfiable*.

A *clause* (*term*) is a circuit consisting of a disjunction (conjunction) of literals. We write $\bot$ for the empty clause and $\top$ for the empty term. We call a clause *tautological* (and a term *contradictory*) if it contains the same variable negated as well as unnegated. A *CNF formula* (*DNF formula*) is a circuit consisting of a conjunction (disjunction) of non-tautological clauses (non-contradictory terms). Whenever convenient, we treat clauses and terms as sets of literals, and CNF and DNF formulas as sets of sets of literals. Throughout the paper, we make use of the fact that any circuit can be transformed into an equisatisfiable CNF formula of size linear in the size of the circuit [24].

A *unit clause* is a clause containing a single literal. A CNF formula ψ is derived from a CNF formula φ by the *unit clause rule* if (ℓ) is a unit clause of φ and $\psi = \varphi[\{\ell \mapsto true\}]$. *Unit propagation* in a CNF formula consists in repeated applictions of the unit clause rule. Unit propagation is said to *derive* the literal ℓ in a CNF formula φ if a CNF formula ψ with $(\ell) \in \psi$ can be derived from φ by unit propagation. We say that unit propagation causes a *conflict* if *false* can be derived by unit propagation. If unit propagation does not cause a conflict the set of literals that can be derived by unit propagation induces an assignment. The *closure* of an assignment τ with respect to unit propagation (in φ) is τ combined with the set of literals derivable by unit propagation in $\varphi[\tau]$.

A clause C has the *reverse unit propagation* (RUP) property with respect to a CNF formula φ if unit propagation in $\varphi[\{\, \ell \mapsto false : \ell \in C \,\}]$ causes a conflict. A *RUP proof of unsatisfiability* of a CNF formula φ is a sequence $C_1, \ldots, C_m$ of clauses such that $C_m = \bot$ and each clause C_i has the RUP property with respect to $\varphi \cup \{C_1, \ldots, C_{i-1}\}$, for $1 \leq i \leq m$.

A (prenex) *Quantified Boolean Formula* $\Phi = \mathcal{Q}.\varphi$ consists of a quantifier prefix $\mathcal{Q}$ and a circuit φ, called the *matrix* of Φ. A *quantifier prefix* is a sequence $\mathcal{Q} = Q_1 x_1 \ldots Q_n x_n$, where the x_i are pairwise distinct variables and $Q_i \in \{\forall, \exists\}$ for $1 \leq i \leq n$. Relative to Φ, variable x_i and its associated literals are called *existential* (*universal*) if $Q_i = \exists$ ($Q_i = \forall$). We write $E(\Phi)$ and $U(\Phi)$ for the sets of existential and universal variables of Φ, respectively. We assume that the set of variables occurring in φ is precisely $\{x_1, \ldots, x_n\}$ (in particular, we only consider closed QBFs) and let $var(\Phi) = \{x_1, \ldots, x_n\}$. We define a total order $<_\Phi$ on the

variables of Φ as $x_i <_\Phi x_j \Leftrightarrow i < j$ and let $D_\Phi(v) = \{\, w \in var(\Phi) : w <_\Phi v \,\}$ for $v \in var(\Phi)$. We drop the subscript from $<_\Phi$ and D_Φ whenever Φ is understood.

A *model circuit* of Φ for a variable $v \in var(\Phi)$ is a circuit f_v with $var(f_v) \subseteq D(v)$. A *model* of Φ is an indexed family $\{f_e\}_{e \in E(\Phi)}$ of model circuits such that $\varphi[\tau] = \mathit{true}$ for every assignment $\tau : var(\Phi) \to \{\mathit{true}, \mathit{false}\}$ that satisfies $f_e[\tau] = \tau(e)$ for $e \in E(\Phi)$. A *countermodel* of Φ is an indexed family $\{f_u\}_{u \in U(\Phi)}$ of model circuits such that $\varphi[\tau] = \mathit{false}$ for every assignment $\tau : var(\Phi) \to \{\mathit{true}, \mathit{false}\}$ that satisfies $f_u[\tau] = \tau(u)$ for $u \in U(\Phi)$. A QBF is *true* if it has a model, and *false* if it has a countermodel.

A QBF is a *PCNF (PDNF) formula* if its matrix is a CNF (DNF) formula. *Q-resolution* [14] and *long-distance Q-resolution* [1,27] are proof systems for false PCNF formulas. Let $\Phi = \mathcal{Q}.\varphi$ be a PCNF formula. A *Q-resolution refutation* of Φ is a sequence $\mathcal{P} = C_1, \ldots, C_m$ of non-tautological clauses where $C_m = \bot$ and each clause C_i is obtained in one of the following ways:

- $C_i \in \varphi$ is an *input clause*.
- $C_i = (C_j \setminus \{p\}) \cup (C_k \setminus \{\neg p\})$ is the *resolvent* of clauses C_j and C_k on *pivot* variable $p \in E(\Phi)$, where $1 \leq j, k < i$ and $p \in C_j$, $\neg p \in C_k$.
- $C_i = C_j \setminus L$ is obtained from C_j with $1 \leq j < i$ by *universal reduction*. This requires that every literal $\ell \in L$ is universal and that there is no existential variable $e \in var(C_i)$ such that $var(\ell) < e$.

The *size* of $\mathcal{P}$ is defined as $|\mathcal{P}| := \sum_{i=1}^{m} |C_i|$.

Long-distance Q-resolution [1] is a generalization of Q-resolution that permits the derivation of tautological clauses by modifying the resolution rule in the following way: if $\ell \in C_j$, $\overline{\ell} \in C_k$, and $var(\ell) \neq p$, then ℓ must be universal and $p < var(\ell)$. In this case we say that the literals ℓ and $\overline{\ell}$ are *merged*, and refer to the pair $\ell, \overline{\ell}$ as a *merged literal* of C_i.

Dual proof systems for true PDNF formulas operating on terms are known as *Q-consensus* and *long-distance Q-consensus*. The dual of universal reduction in these proof systems is called *existential reduction*.

3 Validation of Certificates

In this section, we will describe the setting of the problem of QBF certificate validation. Then, in Sections 4 and 5, we present an algorithm that computes a RUP proof that can be used to replace the final call to the SAT solver by a simple proof check. For the sake of simplicity, we will only focus on false PCNF formulas. The results generalize to true formulas by duality, which will be discussed in Section 6.

Let φ be a CNF formula, let C be a boolean circuit. The substitution of C into φ, denoted by $\varphi[C]$, is simply the CNF formula φ in conjunction with a CNF encoding of C (which may contain additional auxiliary variables). Let $\Phi = \mathcal{Q}.\varphi$ be a false QBF in PCNF, let C be a boolean circuit whose inputs are existential variables of Φ and whose outputs are universal variables of Φ.

The task of verifying that C is a countermodel of Φ is to verify that $\varphi[C]$ is unsatisfiable.

Some QCDCL QBF solvers are capable of outputting a *trace* that contains a (long-distance) Q-resolution refutation of the formula solved. From this refutation, a countermodel circuit can be computed by the Balabanov-Jiang (BJ) algorithm [1], or by the extended Balabanov-Jiang-Janota-Widl (BJJW) algorithm [2] for long-distance Q-resolution. Let $\Phi = \mathcal{Q}.\varphi$ be a QBF, let $\mathcal{P}$ be a (long-distance) Q-resolution refutation of it, let $\textsc{cc}(\mathcal{P})$ be the countermodel circuit computed by the appropriate version of BJ/BJJW. The CNF formula that results from substitution of $\textsc{cc}(\mathcal{P})$ into φ as described in the previous paragraph, i.e., $\varphi[\textsc{cc}(\mathcal{P})]$, is denoted by $\Phi[\mathcal{P}]$, and is called the *validation formula* for the QBF Φ and the proof $\mathcal{P}$. This is the formula that must be checked for unsatisfiability in order to verify the correctness of the certificate $\textsc{cc}(\mathcal{P})$. We will now present a way how to directly compute a RUP proof for the validation formula out of the proof $\mathcal{P}$, thus obviating the need to use a SAT solver and making validation checks solvable in polynomial time.

4 RUP Proofs from Ordinary Q-Resolution

We will begin by describing a countermodel, and in particular its CNF version obtained by the Tseitin conversion, computed by BJ. For a full explanation of the algorithm we refer to the original paper [1]. We illustrate the certificate extraction process on this example formula

$$\exists x_1, x_2\ \forall y\ \exists z \quad (x_1 \vee x_2 \vee y \vee z) \wedge (x_1 \vee x_2 \vee \overline{z}) \wedge (x_1 \vee \overline{x_2}) \\ \wedge\ (\overline{x_1} \vee \overline{x_2} \vee \overline{y} \vee \overline{z}) \wedge (\overline{x_1} \vee \overline{x_2} \vee z) \wedge (\overline{x_1} \vee x_2)$$

along with its Q-resolution refutation:

(1)	$x_1 \vee x_2 \vee y \vee z$	(input)
(2)	$\overline{x_1} \vee \overline{x_2} \vee \overline{y} \vee \overline{z}$	(input)
(3)	$x_1 \vee x_2 \vee \overline{z}$	(input)
(4)	$\overline{x_1} \vee \overline{x_2} \vee z$	(input)
(5)	$x_1 \vee \overline{x_2}$	(input)
(6)	$\overline{x_1} \vee x_2$	(input)
(7)	$x_1 \vee x_2 \vee y$	$(1, 3)$
(8)	$x_1 \vee x_2$	(7)
(9)	$\overline{x_1} \vee \overline{x_2} \vee \overline{y}$	$(2, 4)$
(10)	$\overline{x_1} \vee \overline{x_2}$	(9)
(11)	x_1	$(5, 8)$
(12)	$\overline{x_1}$	$(6, 10)$
(13)	$\bot$	$(11, 12)$

Let $\mathcal{P}$ be a Q-resolution refutation of a formula $\Phi = \mathcal{Q}.\varphi$. BJ processes the clauses of $\mathcal{P}$ forward, and everytime a conclusion R of a reduction step $R = R' - L$ (read the set of literals L is reduced from the clause R' to obtain the clause R) is encountered, for every literal ℓ from L either the clause R (if ℓ is positive) or the term $\overline{R}$ (if ℓ is negative) is pushed to what is called the *countermodel array* of $var(\ell)$ (cf. [1]). At the end, the arrays represent the countermodel functions for their respective variables, in the following way:

Let u be a universal variable, and let its countermodel array have the entries $X_1, \ldots, X_n$. This array is interpreted by constructing a set of partial circuits. Let $f_n^u = X_n$. Then we define

$$f_k^u = \begin{cases} X_k \wedge f_{k+1}^u & \text{if } X_k \text{ is a clause,} \\ X_k \vee f_{k+1}^u & \text{if } X_k \text{ is a term,} \end{cases}$$

and finally $f^u = f_1^u$. The circuit f^u represents the countermodel function for the variable u. Intuitively, these circuits find the first reduction step whose conclusion is falsified, and set all of the reduced literals in the premise so that they are falsified too, which ensures that the falsified clause is implied by the conjunction of input clauses and hence at least one of those is falsified too.

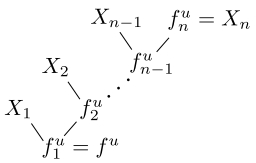


Fig. 2. Schematic depiction of a countermodel circuit extracted by BJ. Each f_i is either an "and" or an "or" gate, depending on the context.

Let us see what this means on the example formula and proof. There is only one universal variable, so we will only build one countermodel array. Processing the clauses forward, the first conclusion of a reduction step that we encounter is (8), y is reduced in positive polarity, so we push the clause $(x_1 \vee x_2)$ to the countermodel array. Next, we encounter the conclusion (10), here y is reduced in negative polarity, so we push the negation of the conclusion $(\overline{x_1} \vee \overline{x_2})$, the term $(x_1 \wedge x_2)$. There are no more reduction steps, so the final countermodel array for y is $[(x_1 \vee x_2), (x_1 \wedge x_2)]$. According to the interpretation above, this results in the circuit $y = ((x_1 \vee x_2) \wedge (x_1 \wedge x_2)) = (x_1 \wedge x_2)$. It can be easily verified that this is indeed a countermodel for the formula.

Let us now examine how the circuit f^u can be translated into CNF for substitution into Φ. We can observe that the circuit f^u has a nested structure, in which first the values of all of the X_k are evaluated, which are then further processed by the circuit to obtain the value for u. Every X_k is either a clause or a term corresponding to a conclusion of a reduction step in $\mathcal{P}$. Let $R_1, \ldots, R_N$ be all conclusions of reduction steps in $\mathcal{P}$, in the same order as they appear in the proof. Then for every X_k there is i_k such that $X_k = R_{i_k}$ or $X_k = \overline{R_{i_k}}$. Let us define variables $g_i = R_i$ for $1 \leq i \leq N$ using the set of clauses

$$G = \big\{\{(\overline{g_i} \vee R_i)\} \cup \{(g_i \vee \overline{\ell}) \mid \ell \in R_i\} \mid 1 \leq i \leq N\big\}.$$

Rather than encoding each countermodel circuit using its X_k members, we will leverage the fact that X_k is either equivalent to g_{i_k} or to $\overline{g_{i_k}}$ and replace it by

the suitable polarity. This way, the recursive definitions of f_k^u boil down to

$$f_n^u = \begin{cases} g_{i_n} & \text{if } X_n \text{ is a clause,} \\ \overline{g_{i_n}} & \text{if } X_n \text{ is a term,} \end{cases}$$

and for $1 \leq k < n$

$$f_k^u = \begin{cases} g_{i_k} \wedge f_{k+1}^u & \text{if } X_k \text{ is a clause,} \\ \overline{g_{i_k}} \vee f_{k+1}^u & \text{if } X_k \text{ is a term.} \end{cases}$$

At this point, since the countermodel arrays are populated in the order of the proof, we can observe the following:

Observation 1 *Whenever g_{i_k} and $g_{i_{k'}}$ appear in the same circuit and $k < k'$, i.e., g_{i_k} comes before $g_{i_{k'}}$ in the corresponding countermodel array, then also $i_k < i_{k'}$, i.e., the reduction step corresponding to g_{i_k} also comes before the one corresponding to $g_{i_{k'}}$.*

Using the simplified circuits with the variables g_i, we can finally produce an encoding into CNF. By using the Tseitin conversion, we get the clauses

$$F_n^u = \begin{cases} (f_n^u \vee \overline{g_{i_n}}) \wedge (\overline{f_n^u} \vee g_{i_n}) & \text{if } X_n \text{ is a clause,} \\ \underbrace{(f_n^u \vee g_{i_n})}_{F_{n,1}^u} \wedge \underbrace{(\overline{f_n^u} \vee \overline{g_{i_n}})}_{F_{n,2}^u} & \text{if } X_n \text{ is a term,} \end{cases}$$

and for $1 \leq k < n$

$$F_k^u = \begin{cases} (f_k^u \vee \overline{g_{i_k}} \vee \overline{f_{k+1}^u}) \wedge (\overline{f_k^u} \vee g_{i_k}) \wedge (\overline{f_k^u} \vee f_{k+1}^u) & \text{if } X_k \text{ is a clause,} \\ \underbrace{(\overline{f_k^u} \vee \overline{g_{i_k}} \vee f_{k+1}^u)}_{F_{k,1}^u} \wedge \underbrace{(f_k^u \vee g_{i_k})}_{F_{k,2}^u} \wedge \underbrace{(f_k^u \vee \overline{f_{k+1}^u})}_{F_{k,3}^u} & \text{if } X_k \text{ is a term.} \end{cases}$$

In our running example, we have two reduction steps, there are therefore two definitions of g-variables, namely $g_1 = (x_1 \vee x_2)$ and $g_2 = (\overline{x_1} \vee \overline{x_2})$. If we replace the actual entries in the countermodel array by the g-variables, we get the array $[g_1, \overline{g_2}]$ and the corresponding circuit $y = g_1 \wedge \overline{g_2}$. Its CNF encoding is

$$(y \vee \overline{g_1} \vee g_2) \wedge (\overline{y} \vee g_1) \wedge (\overline{y} \vee \overline{g_2}).$$

Starting from a formula $\Phi = \mathcal{Q}.\varphi$ and its Q-resolution refutation $\mathcal{P}$, G will denote the set of clauses defining the g_i and F will denote the set of clauses F_k^u (for all universals u and appropriate k) defining the countermodel. The validation formula $\Phi[\mathcal{P}]$ is then $\varphi \wedge G \wedge F$ and we will now present a RUP proof for it.

We will need the following notation. Let x, y be variables of a propositional formula φ, let τ be an assignment to variables of φ. We write $x \cong_\tau^\varphi y$ if, for every extension σ of τ that defines x or y, either unit propagation in $\varphi[\sigma]$ causes a conflict or $\sigma'(x) = \sigma'(y)$, where σ' is the closure of σ with respect to unit propagation. If φ is understood from the context, we may drop the superscript, likewise, if τ is the empty assignment, we may drop the subscript.

Lemma 1. *Let u be a universal variable of Φ whose countermodel array has n entries and the corresponding g-variables are $g_{i_1}, \ldots, g_{i_n}$. For $1 \leq k \leq n$ let τ_k be a partial assignment (to variables of $\Phi[\mathcal{P}]$) which sets $g_{i_1}, \ldots, g_{i_{k-1}}$ to true. Then $f^u \cong_{\tau_k} f^u_k$.*

Proof. We can see that the clauses $F^u_{j,2}[\tau_k]$ are satisfied by g_{i_k} and $\overline{g_{i_k}}$ disappears from $F^u_{j,1}[\tau_k]$ for $1 \leq j < k$. The clauses $F^u_{j,1}[\tau_k]$ and $F^u_{j,3}[\tau_k]$ we are left with encode precisely $f^u_j \cong f^u_{j+1}$. Together, we have that under the assignment τ_k, $f^u = f^u_1 \cong f^u_k$, or in other words $f^u \cong_{\tau_k} f^u_k$. □

The following lemma asserts that the intuition about how countermodel circuits find the first falsified conclusion and set the variable accordingly is indeed true.

Lemma 2. *For $1 \leq i \leq N$ let τ_i be a partial assignment (to variables of $\Phi[\mathcal{P}]$) which sets $g_1, \ldots, g_{i-1}$ to true and g_i to false. Let ℓ be a universal literal that is reduced in the reduction step leading to R_i. Under the assignment τ_i unit propagation (in $\Phi[\mathcal{P}]$) causes a conflict or derives $\overline{\ell}$.*

Proof. Let us assume unit propagation does not cause a conflict. Let $u = var(\ell)$, g_i occurs in the countermodel array of u as some g_{i_k}. If ℓ is positive, $F^u_{k,2}$ together with $\overline{g_{i_k}}$ propagate $\overline{f^u_k}$. If ℓ is negative, $F^u_{k,2}$ together with $\overline{g_{i_k}}$ propagate f^u_k. We can use Observation 1 to see that all $g_{i_{k'}}$ with $k' < k$ are set to true and Lemma 1 applies, so that $f^u \cong f^u_k$ and the value for u propagated is false if ℓ is negative and true if ℓ is positive. Either way, this means that $\overline{\ell}$ is propagated. □

With Lemma 2, we can describe how to construct a RUP proof for $\Phi[\mathcal{P}]$ from $\mathcal{P}$.

Theorem 1. *Let $\mathcal{P}$ be a Q-resolution refutation of the formula $\Phi = \mathcal{Q}.\varphi$. Then there exists a RUP proof of unsatisfiability of the validation formula $\Phi[\mathcal{P}]$ of size $O(|\mathcal{P}|)$, and this proof can be computed in $O(|\mathcal{P}|)$ time.*

Proof. Let $\mathcal{P}'$ be $\mathcal{P}$ with each conclusion R_i replaced by the unit clause (g_i), and with the input clauses omitted. We claim that $\mathcal{P}'$ is a RUP proof of unsatisfiability of $\Phi[\mathcal{P}]$. Since resolvents are always RUP with respect to their premises we only need to verify that all (g_i) are RUP too. Let $R_i = R'_i - L$ be a reduction step, let $\ell \in L$ be one of the universal literals reduced to obtain R_i, let $u = var(l)$. We need to prove that setting (g_i) to false causes a conflict by unit propagation. At the time when (g_i) is inserted into the proof, all (g_j) with $j < i$ have already been inserted and since they are unit clauses, all g_j with $j < i$ are set to true by unit propagation. Adding to that the assignment $\overline{g_i}$, the conditions of Lemma 2 are satisfied and so either unit propagation causes a conflict (in which case we are done), or $\overline{\ell}$ is propagated. Since ℓ was chosen without loss of generality, all literals in L are propagated to false, and since $\overline{g_i}$ trivially propagates all literals of R_i to false, R'_i is falsified and a conflict is reached as required. Clearly, the size of $\mathcal{P}'$ is bounded by the size of $\mathcal{P}$, and it can be computed in time $O(|\mathcal{P}|)$ as the amount of work per each clause of $\mathcal{P}$ is proportional to its size. □

For example, the RUP proof constructed according to Theorem 1 from the example Q-resolution proof would consist in the following sequence of clauses:

$$(x_1 \vee x_2 \vee y),\ (g_1),\ (\overline{x_1} \vee \overline{x_2} \vee \overline{y}),\ (g_2),\ (x_1),\ (\overline{x_1}),\ \bot$$

5 RUP Proofs from Long-Distance Q-Resolution

With long-distance Q-resolution, we cannot directly use the clauses of the refutation in the RUP proof as we did in the proof of Theorem 1, because these clauses may be tautological. Instead, we adopt the approach that was used in the paper of Balabanov et al. [2] in order to generalize BJ to long-distance Q-resolution proofs. The following definition is taken from the paper of Balabanov et al. [2], with a slight change of notation.

Definition 1. *Let $\mathcal{P}$ be a long-distance Q-resolution refutation of the QBF $\Phi = \mathcal{Q}.\varphi$. Let $C \in \mathcal{P}$ be a clause, $\ell \in C$ a literal and $u = var(\ell)$. The* phase function *of the variable u in the clause C, denoted by $u^\phi(C)$, is a boolean function defined recursively as follows:*

- *if C is an input clause, then $u^\phi(C) = 1$ if $\ell = u$, otherwise $u^\phi(C) = 0$*
- *if C is the result of application of universal reduction on the clause C', $u^\phi(C) = u^\phi(C')$*
- *if C is the resolvent of C_1 and C_2 on the pivot literal p, $p \in C_1$, $\overline{p} \in C_2$, then if $u \notin var(C_1)$, then $u^\phi(C) = u^\phi(C_2)$, if $u \notin var(C_2)$ or $u^\phi(C_1) = u^\phi(C_2)$, then $u^\phi(C) = u^\phi(C_1)$, otherwise $u^\phi(C) = (p \wedge u^\phi(C_2)) \vee (\overline{p} \wedge u^\phi(C_1))$*

The effective literal *of ℓ in C, denoted by $\ell^\epsilon(C)$, is a literal that satisfies $\ell^\epsilon(C) \Leftrightarrow (u \Leftrightarrow u^\phi(C))$. The* shadow clause *of C is the clause $C^\sigma = \bigvee_{\ell \in C} \ell^\epsilon(C)$.*

The phase function intuitively tells us, under a given assignment to previous variables in the quantifier prefix, what is the phase in which a given universal variable would have appeared in a given clause, had we restricted the proof using that assignment. The effective literal is a literal which, based on an assignment to previous existential variables, is equivalent to the polarity of its variable indicated by the phase function. Note that in the case when the phase function is constant, i.e. 0 or 1, the effective literal of any literal is simply the literal itself. In such cases we say that the literal is *unmerged*. Literals that are not unmerged are *merged*.

We will now present a description of the countermodel computed by BJJW from a long-distance Q-resolution refutation. In order to do that, we adapt the notation from Section 4. Let $\mathcal{P}$ be a long-distance Q-resolution refutation of a formula $\Phi = \mathcal{Q}.\varphi$. The conclusions of reduction steps in $\mathcal{P}$, in the same order as they appear, are denoted by $R_1, \ldots, R_N$. The variables g_i, $1 \leq i \leq N$, are now equivalent to the shadow clauses ${R_i}^\sigma$ instead of R_i themselves. Since BJJW keeps track of the phase function of every universal variable in every clause, we will use a variable $u^\phi(C)$ to denote the output of the phase function. We will also have variables $\ell^\epsilon(C)$ for the effective literals. In the case of unmerged literals, this will simply be ℓ. By H we will denote the conjunction of all clauses that encode the circuits which define phase variables and effective literals.

The partial countermodel circuits f_k^u from the previous section are slightly more complicated now. Let $R_i = R'_i - L$ be a reduction step, let $\ell \in L$ be a literal that is being reduced, let $u = var(\ell)$. If ℓ is unmerged, ${R_i}^\sigma$ is pushed into

the countermodel array of u, similarly as in the case of ordinary Q-resolution. However, if ℓ is merged, we first require that both ℓ and $\overline{\ell}$ be reduced at the same time (merged literals arise from merges, so they are always in both polarities in a clause), and as such two entries are pushed into the countermodel array of u, namely ${R_i}^{\sigma} \vee \overline{u^{\phi}(R'_i)}$ and right afterwards $\overline{{R_i}^{\sigma}} \wedge \overline{u^{\phi}(R'_i)}$. The intuition for why these entries are added is the following: if the phase $u^{\phi}(R'_i)$ of ℓ in R'_i is positive, and the (shadow clause of the) conclusion is falsified, set u to false, otherwise if the phase is negative and the conclusion is falsified, set u to true, each time falsifying the effective literal $\ell^{\epsilon}(R'_i)$. This is analogous to the ordinary case, where when the conclusion is falsified, the reduced literal is set so that it is falsified, only in this case we falsify the effective literal.

Now, for the sake of simplicity of presentation, we will treat unmerged literals the same way as merged ones. This means that even for unmerged reduced literals we push two entries into the countermodel array, $R_i \vee \overline{u^{\phi}(R'_i)}$ and $\overline{R_i} \wedge \overline{u^{\phi}(R'_i)}$. It is easy to see that if $u^{\phi}(R'_i) = 1$, the term becomes falsified and the clause reduces to just R_i, while if $u^{\phi}(R'_i) = 0$, the clause becomes satisfied and the term reduces to just $\overline{R_i}$. In each case, the circuit is equivalent to what we would have produced by pushing just the one entry as previously.

Let $X_1, \ldots, X_{2n}$ be the entries in the countermodel array of a universal variable u. Each X_{2k-1} is ${R_{i_k}}^{\sigma} \vee \overline{u^{\phi}(R'_{i_k})}$ and X_{2k} is $\overline{{R_{i_k}}^{\sigma}} \wedge \overline{u^{\phi}(R'_{i_k})}$. We have already defined $g_i = {R_i}^{\sigma}$, but since each entry in the countermodel array still contains two variables even after replacing ${R_{i_k}}^{\sigma}$ with g_{i_k}, we will define the auxiliary variables $f'^{u}_{2k-1} = g_{i_k} \vee \overline{u^{\phi}(R'_{i_k})}$ and $f'^{u}_{2k} = \overline{g_{i_k}} \wedge \overline{u^{\phi}(R'_{i_k})}$ using the following sets of clauses (for $1 \leq k \leq n$):

$$F'^{u}_{2k-1} = \overbrace{\left(\overline{f'^{u}_{2k-1}} \vee g_{i_k} \vee \overline{u^{\phi}(R'_{i_k})}\right)}^{F'^{u}_{2k-1,1}} \wedge \overbrace{\left(f'^{u}_{2k-1} \vee \overline{g_{i_k}}\right)}^{F'^{u}_{2k-1,2}} \wedge \overbrace{\left(f'^{u}_{2k-1} \vee u^{\phi}(R'_{i_k})\right)}^{F'^{u}_{2k-1,3}}$$

$$F'^{u}_{2k} = \underbrace{\left(f'^{u}_{2k} \vee g_{i_k} \vee u^{\phi}(R'_{i_k})\right)}_{F'^{u}_{2k,1}} \wedge \underbrace{\left(\overline{f'^{u}_{2k}} \vee \overline{g_{i_k}}\right)}_{F'^{u}_{2k,2}} \wedge \underbrace{\left(\overline{f'^{u}_{2k}} \vee \overline{u^{\phi}(R'_{i_k})}\right)}_{F'^{u}_{2k,3}}$$

Let F' be the conjunction of all F'^{u}_{k} for all universal variables u and all appropriate k. The following is immediate from the clauses F'.

Observation 2 *Setting g_{i_k} to true causes unit propagation to set f'^{u}_{2k-1} and $\overline{f'^{u}_{2k}}$.*

Finally, we are ready to present the set F of clauses which encode the countermodel circuit:

$$F^{u}_{2n,1} = (f^{u}_{2n} \vee \overline{f'^{u}_{2n}}), \quad F^{u}_{2n,2} = (\overline{f^{u}_{2n}}, f'^{u}_{2n}),$$

and for $1 \leq k < 2n$

$$F^{u}_{k} = \begin{cases} \underbrace{(f^{u}_{k} \vee \overline{f'^{u}_{k}} \vee \overline{f^{u}_{k+1}})}_{F^{u}_{k,1}} \wedge \underbrace{(\overline{f^{u}_{k}}, f'^{u}_{k})}_{F^{u}_{k,2}} \wedge \underbrace{(\overline{f^{u}_{k}}, f^{u}_{k+1})}_{F^{u}_{k,3}} & \text{if } k \text{ is odd,} \\ \underbrace{(\overline{f^{u}_{k}} \vee f'^{u}_{k} \vee f^{u}_{k+1})}_{F^{u}_{k,1}} \wedge \underbrace{(f^{u}_{k}, \overline{f'^{u}_{k}})}_{F^{u}_{k,2}} \wedge \underbrace{(f^{u}_{k}, \overline{f^{u}_{k+1}})}_{F^{u}_{k,3}} & \text{if } k \text{ is even.} \end{cases}$$

Similarly as before, let F be the conjunction of all F_k^u for all appropriate u and k, and let G be the conjunction of the clauses defining the equivalences $g_i \Leftrightarrow {R_i}^\sigma$. Then, the validation formula for Φ and $\mathcal{P}$ is $\Phi[\mathcal{P}] = \varphi \wedge F \wedge F' \wedge G \wedge H$.

The following are analogues of Lemmas 1 and 2.

Lemma 3. *Let u be a universal variable of Φ whose countermodel array has $2n$ entries and the corresponding g-variables are $g_{i_1}, \ldots, g_{i_{2n}}$. For $1 \leq k \leq 2n$ let τ_k be a partial assignment (to variables of $\Phi[\mathcal{P}]$) which sets $g_{i_1}, \ldots, g_{i_{k-1}}$ to true. Then $f^u \cong_{\tau_k} f^u_{2k-1}$.*

Proof. Let $1 \leq j < k$. Applying Observation 2, we see that f'^u_{2j-1} and $\overline{f'^u_{2j}}$ are propagated, in each case, inspecting the restricted clauses that remain, we see that $f^u_{2j-1} \cong_{\tau_k} f^u_{2j}$ and $f^u_{2j} \cong_{\tau_k} f^u_{2j+1}$. Altogether, we get $f^u \cong_{\tau_k} f^u_k$. □

Lemma 4. *For $1 \leq i \leq N$ let τ_i be a partial assignment (to variables of $\Phi[\mathcal{P}]$) which sets $g_1, \ldots, g_{i-1}$ to true and g_i to false. Let u be a universal variable of Φ in whose countermodel g_i appears as some g_{i_k}. Let R_i be the corresponding reduction step, obtained from R'_i. Then, under either of the assignments $\tau_i \cup u^\phi(R'_i)$ and $\tau_i \cup \overline{u^\phi(R'_i)}$, unit propagation (in $\Phi[\mathcal{P}]$) causes a conflict or derives $\overline{u^\epsilon(R'_i)}$.*

Proof. Assume unit propagation not cause a conflict. Let us assume $u^\phi(R'_i)$ first. Since we have $\overline{g_{i_k}} \wedge u^\phi(R'_i)$, the clause $F'^u_{2k-1,1}$ propagates $\overline{f'^u_{2k-1}}$, which in turn propagates $\overline{f^u_{2k-1}}$. Since $g_1, \ldots, g_{i-1}$ are set to true, Lemma 3 applies and the value of f^u_{2k-1} is propagated for the value of u, meaning $\overline{u}$ is propagated. Together with the assumption $u^\phi(R'_i)$, we have that the effective literal $u^\epsilon(R'_i)$ is set to false by unit propagation.

If on the other hand we assume $\overline{u^\phi(R'_i)}$, f'^u_{2k-1} is propagated from $F'^u_{2k-1,3}$, which means that the restricted clauses F^u_{2k-1} now encode $f^u_{2k-1} \cong f^u_{2k}$. Also, $F'^u_{2k,1}$ propagates f'^u_{2k}, which in turn propagates f^u_{2k}. Since $g_1, \ldots, g_{i-1}$ are set to true, Lemma 3 applies and the value of f^u_{2k} is propagated for the value of u, meaning u is propagated. Together with the assumption $\overline{u^\phi(R'_i)}$, we have that the effective literal $u^\epsilon(R'_i)$ is set to false by unit propagation. □

While in the case of ordinary Q-resolution, the resolvent of two clauses is always RUP with respect to those clauses, this is not true in the case of long-distance Q-resolution. This is due to the fact that if a merge occurs, a fresh effective literal is introduced in the resolvent, and just falsifying this new fresh literal without the knowledge of the value of the corresponding phase variable does not cause the effective literals in the premises of the resolution step to become falsified. Therefore, we first prove that a set of extra clauses can be derived from the definitions of phase functions and effective literals. These clauses will then empower unit propagation to deal with merged effective literals the same way as with unmerged ones.

Let C be the resolvent of C_1 and C_2 on the pivot literal $p \in C_1$ (and $\overline{p} \in C_2$). Let $\ell \in C_1$, $\overline{\ell} \in C_2$, $u = var(\ell)$ be a universal literal such that $u^\phi(C_1) \neq u^\phi(C_2)$,

i.e. u is being merged in this resolution step. Then the clauses E^u_{C,C_1} and E^u_{C,C_2} are defined as follows:

$$E^u_{C,C_1} = (u^\epsilon(C) \vee p \vee \overline{u^\epsilon(C_1)}), \quad E^u_{C,C_2} = (u^\epsilon(C) \vee \overline{p} \vee \overline{u^\epsilon(C_2)}).$$

We will denote by E the set of all $E^u_{C,D}$ for appropriate premise D, resolvent C, and merged literal u. The clauses of E will provide us with a direct relationship between successive effective literals of one variable. They express one direction of the conditional dependence of an effective literal on the previous effective literals—if an effective literal is false, then based on the value of the pivot variable, the corresponding previous effective literal must be false too.

Lemma 5. *All clauses of E are derivable by RUP from H. The combined size of the RUP proofs is $O(|\mathcal{P}|)$ and they are computable in $O(|\mathcal{P}|)$ time.*

Proof. Let $E^u_{C,D} \in E$, let $p \in D$ be the pivot literal. It can be easily verified by unit propagation on the definitions of phase functions and effective literals that the following is the required RUP proof:

$$(u^\epsilon(C) \vee p \vee \overline{u^\epsilon(D)} \vee u^\phi(C)),\ (E^u_{C,D})$$

Clearly, per each resolution step, these proofs only take up constant space and are computable in constant time, resulting in an overall linear bound. □

We now state the main result of this section (we omit the proof due to space constraints).

Theorem 2. *Let $\mathcal{P}$ be a long-distance Q-resolution refutation of the formula $\Phi = \mathcal{Q}.\varphi$. Then there exists a RUP proof of unsatisfiability of the validation formula $\Phi[\mathcal{P}]$ of size $O(|\mathcal{P}|)$, and this proof can be computed in $O(|\mathcal{P}|)$ time.*

Finally, let us point out that even though we presented concrete CNF encodings for many of the circuits, other encodings can work as well. Namely, it is sufficient if the encodings contain the g-variables (because these are present in the RUP proof) and satisfy the unit-propagation properties of the lemmas.

6 True Formulas

In this section we show how to derive analogues of Theorems 1 and 2 for true formulas. Let us start with the case of a (long-distance) Q-consensus proof $\mathcal{P}$ of a true PDNF formula $\Phi = \mathcal{Q}.\varphi$. In this case the validation formula $\Phi[\mathcal{P}]$ for the model $\textsc{cc}(\mathcal{P})$ is the DNF φ in disjunction with $\textsc{dnf}(\textsc{cc}(\mathcal{P}))$. The task of validation of the model $\textsc{cc}(\mathcal{P})$ is to check that $\Phi[\mathcal{P}]$ is valid, and checking the validity of $\Phi[\mathcal{P}]$ is equivalent to checking that the CNF $\overline{\Phi[\mathcal{P}]}$ is unsatisfiable.

Theorem 3. *Let $\mathcal{P}$ be a long-distance Q-consensus proof of the PDNF formula $\Phi = \mathcal{Q}.\varphi$. Then there exists a RUP proof of unsatisfiability of the negated validation formula $\overline{\Phi[\mathcal{P}]}$ of size $O(|\mathcal{P}|)$, and it can be computed in $O(|\mathcal{P}|)$ time.*

Proof. We observe that the countermodels extracted by BJ/BJJW from $\mathcal{P}$ and from its negation $\overline{\mathcal{P}}$ are in fact the same (we have not discussed the variants of BJ/BJJW for true formulas here, but check the definitions in [1, 2] to see that this trivially holds), which means that their CNF and DNF encodings are negations of one another. This means that

$$\overline{\Phi[\mathcal{P}]} = \overline{\varphi \vee \textsc{dnf}(\textsc{cc}(\mathcal{P}))} = \overline{\varphi} \wedge \overline{\textsc{dnf}(\textsc{cc}(\mathcal{P}))} = \overline{\varphi} \wedge \textsc{cnf}(\textsc{cc}(\overline{\mathcal{P}})) = \overline{\Phi}[\overline{\mathcal{P}}],$$

and we can apply Theorem 2 on $\overline{\Phi}$ and $\overline{\mathcal{P}}$. □

For a Q-consensus proof $\mathcal{P}$ of a true PCNF formula $\Phi = \mathcal{Q}.\varphi$ let us first clarify what the validation formula looks like. We would need to check the validity of $\varphi \vee \textsc{dnf}(\textsc{cc}(\mathcal{P}))$, but φ is a CNF and $\textsc{cc}(\mathcal{P})$ must be encoded as a DNF for validity checking. Therefore, we need to first transform φ to DNF using the Tseitin transformation as follows. Suppose $\varphi = C_1 \wedge \cdots \wedge C_n$. We will define the clause variables $c_i = C_i$ and represent $\textsc{dnf}(\varphi)$ as follows:

$$\textsc{dnf}(\varphi) = \bigvee_{i=1}^{n} \left[(c_i \wedge \overline{C_i}) \vee \bigvee_{\ell \in C_i} (\overline{c_i} \wedge \ell) \right] \vee (c_1 \wedge \cdots \wedge c_n).$$

The validation formula $\Phi[\mathcal{P}]$ is then $\textsc{dnf}(\varphi) \vee \textsc{dnf}(\textsc{cc}(\mathcal{P}))$. As before, instead of checking the validity of $\Phi[\mathcal{P}]$, we will check the unsatisfiability of $\overline{\Phi[\mathcal{P}]}$.

Theorem 4. *Let $\mathcal{P}$ be a long-distance Q-consensus proof of the PCNF formula $\Phi = \mathcal{Q}.\varphi$ with the set of initial terms μ. If every clause from $\overline{\mu}$ is RUP with respect to $\overline{\textsc{dnf}(\varphi)}$, then there exists a RUP proof of unsatisfiability of the negated validation formula $\overline{\Phi[\mathcal{P}]}$ of size $O(|\mathcal{P}|)$, and it can be computed in $O(|\mathcal{P}|)$ time.*

Proof. Let $M = \mathcal{Q}.\mu$ be the PDNF consisting of the initial terms. Using Theorem 3, we obtain a RUP proof for the negated validation formula $\overline{M[\mathcal{P}]} = \overline{M}[\overline{\mathcal{P}}] = \overline{\mu} \wedge \textsc{cnf}(\textsc{cc}(\overline{\mathcal{P}}))$. By prepending $\overline{\mu}$ to this proof, we obtain a RUP proof of $\textsc{dnf}(\varphi) \wedge \textsc{cnf}(\textsc{cc}(\overline{\mathcal{P}})) = \overline{\Phi[\mathcal{P}]}$ of size $O(|\mathcal{P}| + |\mu|) = O(|\mathcal{P}|)$. □

There are two common ways of obtaining initial terms. One is to transform the CNF φ to DNF [13], in which case there is nothing to prove, because the negated initial terms are directly members of $\overline{\textsc{dnf}(\varphi)}$ and therefore RUP. The other way is to produce hitting sets of the clauses of φ. In this case, since every initial term is a hitting set of the clauses $C_1, \ldots, C_n$, we have that for every initial term I and for every clause C_i, there is always a clause of $\textsc{cnf}(\overline{\varphi})$ of the form $(c_i \vee \overline{\ell})$, such that $\ell \in I$. Therefore, by assuming the negation of a negated initial term, i.e. the term itself, unit propagation will propagate c_i for all i, which in turn causes a conflict with the clause $(\overline{c_1} \vee \cdots \vee \overline{c_n})$. Therefore, every clause in $\neg\mu$ is indeed RUP with respect to $\overline{\textsc{dnf}(\varphi)}$ and Theorem 4 applies.

Finally, in the paragraph above we mentioned that initial terms are hitting sets of the clauses of φ (in one of the cases). In fact, this need not always be true, since the hitting sets might have existential reduction applied to them first according to the *model generation* rule [10]. Since it is no problem for the QBF

solver to output the original hitting set without applying existential reduction, but very difficult (NP-hard in general) for the proof-checker to recover it, we suggest to strengthen the conditions on the QRP proof format by requiring that the initial terms be full hitting sets. If this condition is not met our algorithm may fail to produce valid RUP proofs for true PCNF formulas. Fortunately DepQBF always generated terms that happened to be full hitting sets in our experiments.

7 Experiments

We implemented the algorithm of Theorem 2, which generalizes Theorem 1, in a tool called `qrp2rup` (https://www.ac.tuwien.ac.at/research/certificates/) and evaluated the performance compared to various other approaches to certificate validation. In particular, since our tool is also capable of emitting deletion information for DRAT-trim, we evaluated the following six configurations of certificate extractors and validators:

- `qrp2rup` with *deletion* information and validation by DRAT-trim,
- `qrp2rup` without deletion information (*plain*) and validation by DRAT-trim,
- `qrp2rup` and validation by Lingeling (ignoring the RUP proof),
- `qrp2rup` and validation by Glucose (ignoring the RUP proof),
- QBFcert and validation by Lingeling,
- QBFcert and validation by Glucose.

We also experimented with configurations of DRAT-trim that used forward checking (instead of the default backward checking), but excluded the results due to systematically inferior performance. Note that since QBFcert cannot handle long-distance Q-resolution, only the first four configurations were used for the experiments with long-distance proofs. To produce both ordinary and long-distance Q-resolution proofs, we used DepQBF 6.03 in a configuration that allowed tracing (i.e., with most of the advanced techniques off) with a cut-off time of 900 CPU seconds and a memory limit of 4GB. The validation process was limited to 1800 CPU seconds and 7GB of memory. The experiments were run on a cluster of heterogeneous machines running 64-bit Ubuntu 16.04.3 LTS (GNU/Linux 4.10.0-42). We evaluated the tools on the PCNF benchmark sets from the QBF Evaluations 2017, 2016, and 2010. The numbers of true and false validated instances for each configuration and benchmark set are reported in the tables below. The column "total" reports the total number of proofs for true and false formulas produced by DepQBF.

The results indicate that our approach is beneficial mainly on true formulas, but performs well across the board. Interestingly, even though QBFcert tends to produce smaller certificates than `qrp2rup`, Glucose performs worse on them. QBFcert internally uses AIG-based optimizations to shrink the certificates, and it is conceivable that these optimizations hurt Glucose's performance.

		QBFcert+SAT-solver		`qrp2rup`+SAT-solver		`qrp2rup`+DRAT-trim	
year	total	Lingeling	Glucose	Lingeling	Glucose	deletion	plain
2010	162+230	88+215	88+216	88+225	92+**228**	**99**+224	**99**+223
2016	157+206	124+196	123+197	116+202	128+**203**	**136**+202	**136**+200
2017	18+62	**12**+58	**12**+58	11+62	**12**+**63**	**12**+**63**	**12**+62

Table 1. Ordinary Q-resolution proofs: number of true+false formulas validated.

		`qrp2rup`+SAT-solver		`qrp2rup`+DRAT-trim	
year	total	Lingeling	Glucose	deletion	plain
2010	149+222	93+215	95+**217**	**100**+215	**100**+215
2016	160+250	120+197	131+**200**	**137**+196	**137**+196
2017	17+59	12+**59**	**13**+**59**	**13**+**59**	**13**+**59**

Table 2. Long-distance Q-resolution proofs: number of true+false formulas validated.

8 Concluding Remarks

We have presented a way of using (long-distance) Q-resolution/Q-consensus proofs in the process of validating QBF certificates. Our approach does not require a SAT call and comes with a polynomial runtime guarantee. Since it allows us to generate proofs in a format that is routinely used to verify the answers produced by SAT solvers and that has prompted the development of formally verified checkers [8, 11, 16], we can have a high degree of confidence in the correctness of certificates validated in this manner.

However, one subtle challenge remains. When constructing the validation formula $\Phi[\mathcal{P}]$, we take the matrix of Φ and append a CNF encoding of the countermodel. In principle, if we instead appended a small unsatisfiable CNF formula such as $(x) \wedge (\overline{x})$, we could be led to believe that it represents a countermodel when in reality it is much more restrictive than a countermodel is allowed to be (a formula that does not encode a set of functions). It would be desirable to have a way of checking that what we appended to the original matrix is indeed a set of functions (with the correct dependencies) for universal variables. This may require formal verification of parts of the certificate extraction algorithm.

A potential limitation of our approach is that it is sensitive to certain aspects of the CNF encoding of the countermodel to be validated, and therefore does not necessarily work with certificates extracted by other tools. However, our method ought to be compatible with simple circuit-level simplifications of certificates. Moreover, we hope to improve performance by generating GRAT [16] proofs of validation formulas as part of future work.

References

1. V. Balabanov and J. R. Jiang. Unified QBF certification and its applications. *Formal Methods in System Design*, 41(1):45–65, 2012.
2. V. Balabanov, J. R. Jiang, M. Janota, and M. Widl. Efficient extraction of QBF (counter)models from long-distance resolution proofs. In B. Bonet and S. Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 3694–3701. AAAI Press, 2015.
3. M. Benedetti and H. Mangassarian. QBF-based formal verification: Experience and perspectives. *J on Satisfiability, Boolean Modeling and Computation*, 5(1-4):133–191, 2008.
4. A. Biere and F. Lonsing. Integrating dependency schemes in search-based QBF solvers. In O. Strichman and S. Szeider, editors, *Theory and Applications of Satisfiability Testing - SAT 2010*, volume 6175 of *Lecture Notes in Computer Science*, pages 158–171. Springer Verlag, 2010.
5. R. Bloem, R. Könighofer, and M. Seidl. SAT-based synthesis methods for safety specs. In K. L. McMillan and X. Rival, editors, *Verification, Model Checking, and Abstract Interpretation - VMCAI 2014*, volume 8318 of *Lecture Notes in Computer Science*, pages 1–20. Springer Verlag, 2014.
6. M. Cadoli, M. Schaerf, A. Giovanardi, and M. Giovanardi. An algorithm to evaluate Quantified Boolean Formulae and its experimental evaluation. *Journal of Automated Reasoning*, 28(2), 2002.
7. M. Cashmore, M. Fox, and E. Giunchiglia. Partially grounded planning as Quantified Boolean Formula. In D. Borrajo, S. Kambhampati, A. Oddi, and S. Fratini, editors, *23rd International Conference on Automated Planning and Scheduling, ICAPS 2013*. AAAI, 2013.
8. L. Cruz-Filipe, M. J. H. Heule, W. A. Hunt, M. Kaufmann, and P. Schneider-Kamp. Efficient certified rat verification. In L. de Moura, editor, *Automated Deduction – CADE 26*, pages 220–236, Cham, 2017. Springer International Publishing.
9. P. Faymonville, B. Finkbeiner, M. N. Rabe, and L. Tentrup. Encodings of bounded synthesis. In A. Legay and T. Margaria, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 354–370, Berlin, Heidelberg, 2017. Springer Berlin Heidelberg.
10. E. Giunchiglia, M. Narizzano, and A. Tacchella. Clause/term resolution and learning in the evaluation of Quantified Boolean Formulas. *J. Artif. Intell. Res.*, 26:371–416, 2006.
11. M. Heule, W. Hunt, M. Kaufmann, and N. Wetzler. Efficient, verified checking of propositional proofs. In M. Ayala-Rincón and C. A. Muñoz, editors, *Interactive Theorem Proving*, pages 269–284, Cham, 2017. Springer International Publishing.
12. M. Heule, W. A. H. Jr., and N. Wetzler. Trimming while checking clausal proofs. In *Formal Methods in Computer-Aided Design, FMCAD 2013*, pages 181–188. IEEE Computer Soc., 2013.
13. M. Janota and J. Marques-Silva. An Achilles' heel of term-resolution. In E. C. Oliveira, J. Gama, Z. A. Vale, and H. L. Cardoso, editors, *Progress in Artificial Intelligence - 18th EPIA Conference on Artificial Intelligence, EPIA 2017*, volume 10423 of *Lecture Notes in Computer Science*, pages 670–680. Springer Verlag, 2017.
14. H. Kleine Büning, M. Karpinski, and A. Flögel. Resolution for quantified Boolean formulas. *Information and Computation*, 117(1):12–18, 1995.

15. M. Kronegger, A. Pfandler, and R. Pichler. Conformant planning as benchmark for QBF-solvers. In *International Workshop on Quantified Boolean Formulas - QBF 2013*, 2013. http://fmv.jku.at/qbf2013/.
16. P. Lammich. Efficient verified (un)sat certificate checking. In L. de Moura, editor, *Automated Deduction – CADE 26*, pages 237–254. Springer International Publishing, 2017.
17. A. Niemetz, M. Preiner, F. Lonsing, M. Seidl, and A. Biere. Resolution-based certificate extraction for QBF. In A. Cimatti and R. Sebastiani, editors, *Theory and Applications of Satisfiability Testing - SAT 2012*, volume 7317 of *Lecture Notes in Computer Science*, pages 430–435. Springer Verlag, 2012.
18. T. Peitl, F. Slivovsky, and S. Szeider. Dependency learning for QBF. In S. Gaspers and T. Walsh, editors, *Theory and Applications of Satisfiability Testing – SAT 2017*, pages 298–313. Springer International Publishing, 2017.
19. M. N. Rabe and S. A. Seshia. Incremental determinization. In N. Creignou and D. L. Berre, editors, *Theory and Applications of Satisfiability Testing - SAT 2016*, volume 9710 of *Lecture Notes in Computer Science*, pages 375–392. Springer Verlag, 2016.
20. M. N. Rabe and L. Tentrup. CAQE: A certifying QBF solver. In R. Kaivola and T. Wahl, editors, *Formal Methods in Computer-Aided Design - FMCAD 2015*, pages 136–143. IEEE Computer Soc., 2015.
21. J. Rintanen. Asymptotically optimal encodings of conformant planning in QBF. In *22nd AAAI Conference on Artificial Intelligence*, pages 1045–1050. AAAI, 2007.
22. S. Staber and R. Bloem. Fault localization and correction with QBF. In J. Marques-Silva and K. A. Sakallah, editors, *Theory and Applications of Satisfiability Testing - SAT 2007*, volume 4501 of *Lecture Notes in Computer Science*, pages 355–368. Springer Verlag, 2007.
23. L. Tentrup. Non-prenex QBF solving using abstraction. In N. Creignou and D. L. Berre, editors, *Theory and Applications of Satisfiability Testing - SAT 2016*, volume 9710 of *Lecture Notes in Computer Science*, pages 393–401. Springer Verlag, 2016.
24. G. S. Tseitin. On the complexity of derivation in propositional calculus. *Zap. Nauchn. Sem. Leningrad Otd. Mat. Inst. Akad. Nauk SSSR*, 8:23–41, 1968. Russian. English translation in J. Siekmann and G. Wrightson (eds.) *Automation of Reasoning. Classical Papers on Computer Science 1967–1970*, Springer Verlag, 466–483, 1983.
25. N. Wetzler, M. J. H. Heule, and W. A. Hunt. Drat-trim: Efficient checking and trimming using expressive clausal proofs. In C. Sinz and U. Egly, editors, *Theory and Applications of Satisfiability Testing – SAT 2014*, pages 422–429. Springer Verlag, 2014.
26. L. Zhang and S. Malik. Conflict driven learning in a quantified Boolean satisfiability solver. In L. T. Pileggi and A. Kuehlmann, editors, *Proceedings of the 2002 IEEE/ACM International Conference on Computer-aided Design, ICCAD 2002, San Jose, California, USA, November 10-14, 2002*, pages 442–449. ACM / IEEE Computer Society, 2002.
27. L. Zhang and S. Malik. The quest for efficient boolean satisfiability solvers. In D. Brinksma and K. G. Larsen, editors, *Computer Aided Verification: 14th International Conference (CAV 2002)*, volume 2404 of *Lecture Notes in Computer Science*, pages 17–36, 2002.