Final Report

Benjamin Kiesl[1]
kiesl@kr.tuwien.ac.at

PhD Advisors: Hans Tompits[1] and Martina Seidl[2]
Supervisor at UT Austin: Marijn J.H. Heule[3]

[1] Institute of Information Systems, TU Wien, Austria
[2] Institute for Formal Models and Verification, JKU Linz, Austria
[3] Department of Computer Science, The University of Texas at Austin, USA

1 Summary

My work during my research stay at the University of Texas at Austin led to the following two publications at top-level venues in the field of automated reasoning:

Marijn J.H. Heule, Benjamin Kiesl, and Armin Biere:
Short Proofs Without New Variables.
In: Proceedings of the 26th International Conference on Automated Deduction (CADE-26). LNCS, vol. 10395, pp. 130–147. Springer, Cham (2017).

Benjamin Kiesl, Marijn J.H. Heule, and Martina Seidl:
A Little Blocked Literal Goes a Long Way.
In: Proceedings of the 20th International Conference on Theory and Applications of Satisfiability Testing (SAT 2017). LNCS, vol. 10491, pp. 281–297. Springer, Cham (2017).

The first of these two papers won the *best paper award* of the CADE-26 conference. The two papers are given below. Here, I want to discuss my contributions. Since many of the results have been obtained in discussions where several authors contributed ideas, it is not possible to associate every part of the papers with a single author. In both papers, the *introductions*, *preliminaries* and *conclusions* were basically written together by all authors.

In the first paper—*Short Proofs Without New Variables*—we introduce novel redundancy notions for SAT solving. We relate these new notions to theoretical concepts from the literature and illustrate the efficiency of proof systems that are based on these notions. I wrote most of the Sections 3–5. All authors developed the ideas for Sections 3 and 4—where we introduce the new redundancy notions—together, but I came up with the formal proofs for our theoretical results. The main idea of Section 5, in which we show how proof systems based on our redundancy notions yield short proofs for the well-known pigeon hole

formulas, is due to Marijn Heule. The evaluation in Section 6 was done by Marijn Heule while the rest of Section 6, in which we establish the computational complexity of our new methods, was written together by Marijn Heule and me. I wrote most parts of Section 7 (*Related Work*) although most of the ideas for this section were developed by all three authors together.

In the second paper—*A Little Blocked Literal Goes a Long Way*—we clarify the relationship between two proof systems for quantified Boolean formulas by proving that the QRAT proof system can polynomially simulate the long-distance-resolution proof system. The relationship between the two proof systems was not clear before and it was an open question in the research community whether or not a polynomial simulation like ours is possible. The idea for the whole simulation was developed jointly by all three authors. I wrote most parts of Sections 3–5, in which we introduce our simulation and prove that it is correct and that it is polynomially bounded. The evaluation in Section 6 was written together by all three authors although most of the work has been done by Marijn Heule and me. Martina Seidl contributed the framework—which she had already implemented in earlier projects—for the implementation of the tool (`ld2qrat`) that we presented and evaluated in Section 6. I modified Martina Seidl's framework accordingly to obtain this tool. Finally, I also performed a regression analysis to show that our empirical evaluation confirms our theoretical results.

Short Proofs Without New Variables*

Marijn J.H. Heule[1], Benjamin Kiesl[2], and Armin Biere[3]

[1] Department of Computer Science, The University of Texas at Austin
[2] Institute of Information Systems, Vienna University of Technology
[3] Institute for Formal Models and Verification, JKU Linz

Abstract. Adding and removing redundant clauses is at the core of state-of-the-art SAT solving. Crucial is the ability to add short clauses whose redundancy can be determined in polynomial time. We present a characterization of the strongest notion of clause redundancy (i.e., addition of the clause preserves satisfiability) in terms of an implication relationship. By using a polynomial-time decidable implication relation based on unit propagation, we thus obtain an efficiently checkable redundancy notion. A proof system based on this notion is surprisingly strong, even without the introduction of new variables—the key component of short proofs presented in the proof complexity literature. We demonstrate this strength on the famous pigeon hole formulas by providing short clausal proofs without new variables.

1 Introduction

Satisfiability (SAT) solvers are used for determining the correctness of hardware and software systems [1,2]. It is therefore crucial that these solvers justify their claims by providing proofs that can be independently verified. This holds also for various other applications that use SAT solvers. Just recently, long-standing mathematical problems were solved using SAT, including the Erdős Discrepancy Problem [3] and the Pythagorean Triples Problem [4]. Especially in such cases, proofs are at the center of attention, and without them, the result of a solver is almost worthless. What the mathematical problems and the industrial applications have in common, is that proofs are often of considerable size—in the case of the Pythagorean Triples Problem about 200 terabytes. As the size of proofs is influenced by the strength of the underlying proof system, the search for shorter proofs goes hand in hand with the search for stronger proof systems.

In this paper, we introduce highly expressive clausal proof systems that are closely related to state-of-the-art SAT solving. Informally, a clausal proof system allows the addition of redundant clauses to a formula in conjunctive normal form (CNF). Here, a clause is considered *redundant* if its addition preserves satisfiability. If the repeated addition of clauses allows us finally to add the empty clause—which is, by definition, unsatisfiable—the unsatisfiability of the original formula has been established.

* This work has been supported by the National Science Foundation under grant CCF-1526760 and the Austrian Science Fund (FWF) under project W1255-N23.

Since satisfiability equivalence is not efficiently decidable, practical proof systems only allow the addition of a clause if it fulfills some efficiently decidable criterion that ensures redundancy. For instance, the popular DRAT proof system [5], which is the de-facto standard in practical SAT solving, only allows the addition of so-called *resolution asymmetric tautologies* [6]. Given a formula and a clause, one can decide in polynomial time whether the clause is a resolution asymmetric tautology with respect to the formula and therefore the soundness of DRAT proofs can be efficiently checked.

We present new redundancy criteria by introducing a characterization of clause redundancy based on a simple implication relationship between formulas. By replacing the logical implication relation in this characterization with stronger notions of implication that are computable in polynomial time, we then obtain powerful redundancy criteria that are still efficiently decidable. We show that these redundancy criteria not only generalize earlier ones like the above-mentioned resolution asymmetric tautologies or *set-blocked clauses* [7], but that they are also related to other concepts from the literature, namely *autarkies* [8], *safe assignments* [9], *variable instantiation* [10], and *symmetry breaking* [11].

Proof systems based on our new redundancy criteria turn out to be highly expressive, even without the introduction of new variables. This is in contrast to resolution, which is considered relatively weak as long as one does not allow the introduction of new variables via definitions as in the stronger proof system of *extended resolution* [12,13]. The introduction of new variables, however, has a major drawback: the search space of variables and clauses one could possibly add to a proof is infinite, even when bounding the size of clauses. Finding useful clauses with new variables is therefore hard in practice, although there have been a few successes in the past [14,15].

We illustrate the strength of our strongest proof system by providing short clausal proofs for the famous pigeon hole formulas without introducing new variables. The size of the proofs is linear in the size of the formulas and the longest clauses in the proofs have length two. In these proofs, we add redundant clauses that are similar in nature to symmetry-breaking predicates [11,16]. To verify the correctness of proofs in our new system, we implemented a proof checker. The checker is built on top of `DRAT-trim` [5], the checker used to validate the unsatisfiability results of the recent SAT competitions [17]. We compare our proofs with existing proofs of the pigeon hole formulas in other proof systems and show that our new proofs are much smaller and cheaper to validate.

2 Preliminaries

We consider propositional formulas in *conjunctive normal form* (CNF), which are defined as follows. A *literal* is either a variable x (a *positive literal*) or the negation $\overline{x}$ of a variable x (a *negative literal*). The *complementary literal* $\bar{l}$ of a literal l is defined as $\bar{l} = \overline{x}$ if $l = x$ and $\bar{l} = x$ if $l = \overline{x}$. Accordingly, for a set L of literals, we define $\overline{L} = \{\bar{l} \mid l \in L\}$. A *clause* is a disjunction of literals. If not stated otherwise, we assume that clauses do not contain complementary literals.

A *formula* is a conjunction of clauses. We view clauses as sets of literals and formulas as sets of clauses. For a set L of literals and a formula F, we define $F_L = \{C \in F \mid C \cap L \neq \emptyset\}$. We sometimes write F_l to denote $F_{\{l\}}$.

An *assignment* is a partial function from a set of variables to the truth values 1 (*true*) and 0 (*false*). An assignment is *total* w.r.t. a formula if it assigns a truth value to every variable occurring in the formula. A literal l is *satisfied* (*falsified*) by an assignment α if l is positive and $\alpha(var(l)) = 1$ ($\alpha(var(l)) = 0$, respectively) or if it is negative and $\alpha(var(l)) = 0$ ($\alpha(var(l)) = 1$, respectively). We often denote assignments by the sequences of literals they satisfy. For instance, $x\,\overline{y}$ denotes the assignment that assigns x to 1 and y to 0. A clause is satisfied by an assignment α if it contains a literal that is satisfied by α. Finally, a formula is satisfied by an assignment α if all its clauses are satisfied by α. A formula is *satisfiable* if there exists an assignment that satisfies it. Two formulas are *logically equivalent* if they are satisfied by the same assignments. Two formulas F and F' are *satisfiability equivalent* if F is satisfiable if and only if F' is satisfiable.

We denote the empty clause by $\bot$ and the satisfied clause by $\top$. Given an assignment α and a clause C, we define $C|_\alpha = \top$ if α satisfies C, otherwise $C|_\alpha$ denotes the result of removing from C all the literals falsified by α. Moreover, for a formula F, we define $F|_\alpha = \{C|_\alpha \mid C \in F \text{ and } C|_\alpha \neq \top\}$. We say that a clause C *blocks* an assignment α if $C = \{x \mid \alpha(x) = 0\} \cup \{\overline{x} \mid \alpha(x) = 1\}$. A *unit clause* is a clause that contains only one literal. The result of applying the *unit clause rule* to a formula F is the formula $F|_\alpha$ with α being the assignment that satisfies exactly the unit clauses in F. The iterated application of the unit clause rule to a formula, until no unit clauses are left, is called *unit propagation*. If unit propagation yields the empty clause $\bot$, we say that it derived a *conflict*.

By $F \vDash F'$, we denote that F implies F', i.e., all assignments satisfying F also satisfy F'. Furthermore, by $F \vdash_1 F'$ we denote that for every clause $C \in F'$, unit propagation of the negated literals of C on F derives a conflict (thereby, the negated literals of C are viewed as unit clauses). For example, $x \wedge y \vdash_1 (x \vee z) \wedge y$, since unit propagation of the unit clauses $\overline{x}$ and $\overline{z}$ derives a conflict with x, and propagation of $\overline{y}$ derives a conflict with y. Similarly, $F \vdash_0 F'$ denotes that every clause in F' is subsumed by (i.e., is a superset of) a clause in F. Observe that $F \supseteq F'$ implies $F \vdash_0 F'$, $F \vdash_0 F'$ implies $F \vdash_1 F'$, and $F \vdash_1 F'$ implies $F \vDash F'$.

3 Clause Redundancy and Clausal Proofs

In this section, we introduce a formal notion of clause redundancy and demonstrate how it provides the basis for clausal proof systems. We start by introducing clause redundancy [7]:

Definition 1. *A clause C is* redundant *w.r.t. a formula F if F and $F \cup \{C\}$ are satisfiability equivalent.*

For instance, the clause $C = x \vee y$ is redundant w.r.t. $F = \{\overline{x} \vee \overline{y}\}$ since F and $F \cup \{C\}$ are satisfiability equivalent (although they are not logically equivalent). Since this notion of redundancy allows us to add redundant clauses to a formula without affecting its satisfiability, it gives rise to clausal proof systems.

Definition 2. *A* proof *of a clause C_m from a formula F is a sequence of pairs $(C_1, \omega_1), \ldots, (C_m, \omega_m)$, where each C_i $(1 \leq i \leq m)$ is a clause that is redundant w.r.t. $F \cup \{C_j \mid 1 \leq j < i\}$, and this redundancy can be efficiently checked using the (arbitrary)* witness *ω_i. If $C_m = \bot$, the proof is a* refutation *of F.*

Clearly, since every clause-addition step preserves satisfiability, and since the empty clause is unsatisfiable, a refutation certifies the unsatisfiability of F due to transitivity. Note that the ω_i can be arbitrary witnesses (they can be assignments, or even left out if no explicit witness is needed) that certify the redundancy of C_i w.r.t. $F \cup \{C_j \mid 1 \leq j < i\}$, and by requiring that the redundancy can be *efficiently checked*, we mean that it can be checked in polynomial time w.r.t. the size of $F \cup \{C_j \mid 1 \leq j < i\}$.

By specifying in detail what kind of redundant clauses—and corresponding witnesses—one can add to a proof, we obtain concrete proof systems. This is usually done by defining an efficiently checkable syntactic criterion that guarantees that clauses fulfilling this criterion are redundant. A popular example for a clausal proof system is DRAT [5], the de-facto standard for unsatisfiability proofs in practical SAT solving. DRAT allows the addition of a clause if it is a so-called *resolution asymmetric tautology* [6] (RAT, defined in the next section). As it can be efficiently checked whether a clause is a RAT, and since RATs cover a large portion of redundant clauses, the DRAT proof system is very powerful.

The strength of a clausal proof system depends on the generality of the underlying redundancy criterion. We say that a redundancy criterion $\mathcal{R}_1$ is *more general* than a redundancy criterion $\mathcal{R}_2$ if, whenever $\mathcal{R}_2$ identifies a clause C as redundant w.r.t. a formula F, then $\mathcal{R}_1$ also identifies C as redundant w.r.t. F. For instance, whenever a clause is subsumed in some formula, it is a RAT w.r.t. that formula. Therefore, the RAT redundancy criterion is more general than the subsumption criterion. In the next section, we develop redundancy criteria that are even more general than RAT. This gives rise to proof systems that are stronger than DRAT but still closely related to practical SAT solving.

4 Clause Redundancy via Implication

In the following, we introduce a characterization of clause redundancy that reduces the question whether a clause is redundant w.r.t. a certain formula to a simple question of implication. The advantage of this is that we can replace the logical implication relation by stronger, polynomially decidable implication relations to derive powerful redundancy criteria that are still efficiently checkable. These redundancy criteria can then be used to obtain highly expressive clausal proof systems.

Our characterization is based on the observation that a clause in a CNF formula can be seen as a constraint that blocks those assignments falsifying the clause. Therefore, a clause can be safely added to a formula if it does not constrain the formula too much. What we mean by this, is that after adding the clause, there should still exist other assignments (i.e., assignments not blocked

by the clause) under which the formula is at least as satisfiable as under the assignments blocked by the clause. Consider the following example:

Example 1. Let $F = \{x \vee y,\ x \vee z,\ \overline{x} \vee y \vee z\}$ and consider the (unit) clause $C = x$ which blocks all assignments that assign x to 0. The addition of C to F does not affect satisfiability: Let $\alpha = \overline{x}$ and $\omega = x$. Then, $F|\alpha = \{y,\ z\}$ while $F|\omega = \{y \vee z\}$. Clearly, every satisfying assignment of $F|\alpha$ is also a satisfying assignment of $F|\omega$ (i.e., $F|\alpha \vDash F|\omega$). Thus, F is at least as satisfiable under ω as it is under α. Moreover, ω satisfies C. The addition of C does therefore not affect the satisfiability of F. □

This motivates the characterization of clause redundancy we introduce next. Note that for a given clause C, "*the* assignment α blocked by C" can be a partial assignment, meaning that C actually rules out all assignments that extend α:

Theorem 1. *Let F be a formula, C a clause, and α the assignment blocked by C. Then, C is redundant w.r.t. F if and only if there exists an assignment ω such that ω satisfies C and $F|\alpha \vDash F|\omega$.*

Proof. For the "only if" direction, assume that F and $F \cup \{C\}$ are satisfiability equivalent. If $F|\alpha$ is unsatisfiable, then $F|\alpha \vDash F|\omega$ for every ω, hence the statement trivially holds. Assume now that $F|\alpha$ is satisfiable, implying that F is satisfiable. Then, since F and $F \cup \{C\}$ are satisfiability equivalent, there exists an assignment ω that satisfies both F and C. Since ω satisfies F, it holds that $F|\omega = \emptyset$ and so $F|\alpha \vDash F|\omega$.

For the "if" direction, assume that there exists an assignment ω such that ω satisfies C and $F|\alpha \vDash F|\omega$. Now, let γ be a (total) assignment that satisfies F and assume it falsifies C. As γ falsifies C, it coincides with α on $var(\alpha)$. Therefore, since γ satisfies F, it must satisfy $F|\alpha$ and since $F|\alpha \vDash F|\omega$ it must also satisfy $F|\omega$. Now, consider the following assignment γ':

$$\gamma'(x) = \begin{cases} \omega(x) & \text{if } x \in var(\omega), \\ \gamma(x) & \text{otherwise.} \end{cases}$$

Clearly, since ω satisfies C, γ' also satisfies C. Moreover, as γ satisfies $F|\omega$ and $var(F|\omega) \subseteq var(\gamma) \setminus var(\omega)$, γ' satisfies F. Hence, γ' satisfies $F \cup \{C\}$. □

This alternative characterization of redundancy allows us to replace the logical implication relation by stronger polynomially decidable relations. For instance, we can replace the condition $F|\alpha \vDash F|\omega$ by the stronger condition $F|\alpha \vdash_1 F|\omega$ (likewise, we could also use relations such as "$\vdash_0$" or "$\supseteq$" instead of "$\vdash_1$"). Now, if we are given a clause C—which implicitly gives us the blocked assignment α—and a *witnessing assignment* ω, then we can check in polynomial time whether $F|\alpha \vdash_1 F|\omega$, implying that C is redundant w.r.t. F. We can therefore use this implication-based redundancy notion to define proof systems. A proof is then a sequence $(C_1, \omega_1), \ldots, (C_m, \omega_m)$ where the ω_i are the witnessing assignments.

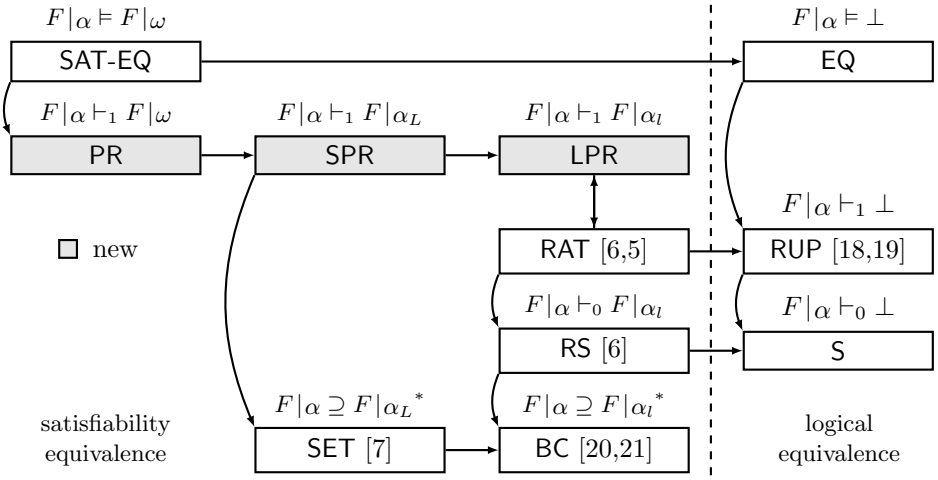


Fig. 1. Landscape of Redundancy Notions. SAT-EQ stands for all redundant clauses and EQ for implied clauses. A path from X to Y indicates that X is more general than Y. The asterisk (*) denotes that the exact characterization implies the shown one, e.g., for every set-blocked clause, the property $F|_\alpha \supseteq F|_{\alpha_L}$ holds, but not vice versa.

In the following, we use the propagation-implication relation "$\vdash_1$" to define the redundancy criteria of (1) *literal-propagation redundancy* (LPR), (2) *set-propagation redundancy* (SPR), and (3) *propagation redundancy* (PR). Basically, the three notions differ in the way we allow the witnessing assignment ω to differ from the assignment α blocked by a clause. The more freedom we give to ω, the more general the redundancy notion we obtain. We show that LPR clauses—the least general of the three—coincide with RAT. For the more general SPR clauses, we show that they generalize set-blocked clauses (SET) [7], which is not the case for LPR clauses. Finally, PR clauses are the most general ones. They give rise to an extremely powerful proof system that is still closely related to CDCL-based SAT solving. The new landscape of redundancy notions we thereby obtain is illustrated in Fig. 1. In the figure, RUP stands for the redundancy notion based on reverse unit propagation [18,19], S stands for subsumed clauses, RS for clauses with subsumed resolvents [6], and BC for blocked clauses [20,21].

As we will see, when defining proof systems based on LPR (e.g., the DRAT system) or SPR clauses, we do not need to add the explicit redundancy witnesses (i.e., the witnessing assignments ω) to a proof. In these two cases, a proof can thus just be seen as a sequence of clauses. A proof system based on SPR clauses can therefore have the same syntax as DRAT proofs, which makes it "downwards compatible". This is in contrast to a proof system based on PR clauses where, at least in general, we have to add the witnessing assignments to a proof, otherwise we cannot check the redundancy of a clause in polynomial time.

We start by introducing LPR clauses. In the following, given a (partial) assignment α and a set L of literals, we denote by α_L the assignment obtained

from α by flipping the truth values of the literals in L. If L contains only a single literal l, then we write α_l to denote $\alpha_{\{l\}}$.

Definition 3. *Let F be a formula, C a clause, and α the assignment blocked by C. Then, C is* literal-propagation redundant (LPR) *w.r.t. F if it contains a literal l such that $F|_\alpha \vdash_1 F|_{\alpha_l}$.*

Example 2. Let $F = \{x \vee y,\ x \vee \overline{y} \vee z,\ \overline{x} \vee z\}$ and let C be the unit clause x. Then, $\alpha = \overline{x}$ is the assignment blocked by C, and $\alpha_x = x$. Now, consider $F|_\alpha = \{y,\ \overline{y} \vee z\}$ and $F|_{\alpha_x} = \{z\}$. Clearly, $F|_\alpha \vdash_1 F|_{\alpha_x}$ and therefore C is literal-propagation redundant w.r.t. F. □

The LPR definition is quite restrictive, as it requires the witnessing assignment α_l to disagree with α on exactly one variable. Nevertheless, this already suffices for LPR clauses to coincide with RATs [6]:

Definition 4. *Let F be a formula and C a clause. Then, C is a* resolution asymmetric tautology (RAT) *w.r.t. F if it contains a literal l such that, for every clause $D \in F_{\bar{l}}$, $F \vdash_1 C \cup (D \setminus \{\bar{l}\})$.*

Theorem 2. *A clause C is literal-propagation redundant w.r.t. a formula F if and only if it is a resolution asymmetric tautology w.r.t. F.*

Proof. For the "only if" direction, assume that C is LPR w.r.t. F, i.e., it contains a literal l such that $F|_\alpha \vdash_1 F|_{\alpha_l}$. Now, let $D \in F_{\bar{l}}$. We have to show that $F \vdash_1 C \cup (D \setminus \{\bar{l}\})$. First, note that $F|_\alpha$ is exactly the result of propagating the negated literals of C on F (i.e., applying the unit clause rule with the negated literals of C but not performing further propagations). Moreover, since α_l falsifies $\bar{l}$, it follows that $D|_{\alpha_l} \subseteq (D \setminus \{\bar{l}\})$. But then, since $F|_\alpha \vdash_1 D|_{\alpha_l}$, it must hold that $F \vdash_1 C \cup (D \setminus \{\bar{l}\})$, hence C is a RAT w.r.t. F.

For the " if" direction, assume that C is a RAT w.r.t. F, i.e., it contains a literal l such that, for every clause $D \in F_{\bar{l}}$, $F \vdash_1 C \cup (D \setminus \{\bar{l}\})$. Now, let $D|_{\alpha_l} \in F|_{\alpha_l}$ for $D \in F$. We have to show that $F|_\alpha \vdash_1 D|_{\alpha_l}$. Since α_l satisfies l and α falsifies C, D does neither contain l nor any negations of literals in C except for possibly $\bar{l}$. If D does not contain $\bar{l}$, then $D|_\alpha = D|_{\alpha_l}$ is contained in $F|_\alpha$ and hence the claim immediately follows. Assume therefore that $\bar{l} \in D$.

As argued in the proof for the other direction, propagating the negated literals of C (and no other literals) on F yields $F|_\alpha$. Therefore, since $F \vdash_1 C \cup (D \setminus \{\bar{l}\})$ and $D \setminus \{\bar{l}\}$ does not contain any negations of literals in C (which could otherwise be the reason for a unit propagation conflict that only happens because of C containing a literal whose negation is contained in $D \setminus \{\bar{l}\}$), it must be the case that $F|_\alpha \vdash_1 D \setminus \{\bar{l}\}$. Now, the only literals of $D \setminus \{\bar{l}\}$ that are not contained in $D|_{\alpha_l}$ are the ones falsified by α, but those are anyhow not contained in $F|_\alpha$. It follows that $F|_\alpha \vdash_1 D|_{\alpha_l}$ and thus C is LPR w.r.t. F. □

By allowing the witnessing assignments to disagree with α on more than only one literal, we obtain the more general notion of set-propagation-redundant clauses:

Definition 5. *Let F be a formula, C a clause, and α the assignment blocked by C. Then, C is* set-propagation redundant (SPR) *w.r.t. F if it contains a non-empty set L of literals such that $F|_\alpha \vdash_1 F|_{\alpha_L}$.*

Example 3. Let $F = \{x \vee y,\ x \vee \overline{y} \vee z,\ \overline{x} \vee z,\ \overline{x} \vee u,\ \overline{u} \vee x\}$, $C = x \vee u$, and $L = \{x, u\}$. Then, $\alpha = \overline{x}\,\overline{u}$ is the assignment blocked by C, and $\alpha_L = x\,u$. Now, consider $F|_\alpha = \{y,\ \overline{y} \vee z\}$ and $F|_{\alpha_L} = \{z\}$. Clearly, $F|_\alpha \vdash_1 F|_{\alpha_L}$ and so C is set-propagation redundant w.r.t. F. Note also that C is not literal-propagation redundant w.r.t. F. □

Since L is a subset of C, we do not need to add it (or the assignment α_L) explicitly to an SPR proof. By requiring that L must consist of the first literals of C when adding C to a proof (viewing a clause as a sequence of literals), we can ensure that the SPR property is efficiently decidable. For instance, when a proof contains the clause $l_1 \vee \cdots \vee l_n$, we first check whether the SPR property holds under the assumption that $L = \{l_1\}$. If not, we proceed by assuming that $L = \{l_1, l_2\}$, and so on until $L = \{l_1, \ldots, l_n\}$. Thereby, only linearly many candidates for L need to be checked. In contrast to LPR clauses and RATs, the notion of SPR clauses generalizes set-blocked clauses [7]:

Definition 6. *A clause C is* set-blocked (SET) *by a non-empty set $L \subseteq C$ in a formula F if, for every clause $D \in F_{\overline{L}}$, the clause $(C \setminus L) \cup \overline{L} \cup D$ contains two complementary literals.*

To show that set-propagation-redundant clauses generalize set-blocked clauses, we first characterize them as follows:

Lemma 3. *Let F be a clause, C a formula, $L \subseteq C$ a non-empty set of literals, and α the assignment blocked by C. Then, C is set-blocked by L in F if and only if, for every $D \in F$, $D|_\alpha = \top$ implies $D|_{\alpha_L} = \top$.*

Proof. For the "only if" direction, assume that there exists a clause $D \in F$ such that $D|_\alpha = \top$ but $D|_{\alpha_L} \neq \top$. Then, since α and α_L disagree only on literals in L, it follows that D contains a literal $l \in \overline{L}$ and therefore $D \in F_{\overline{L}}$. Now, α_L falsifies exactly the literals in $(C \setminus L) \cup \overline{L}$ and since it does not satisfy any of the literals in D, it follows that there exists no literal $l \in D$ such that its complement $\overline{l}$ is contained in $(C \setminus L) \cup \overline{L}$. Therefore, C is not set-blocked by L in F.

For the "if" direction, assume that C is not set-blocked by L in F, i.e., there exists a clause $D \in F_{\overline{L}}$ such that $(C \setminus L) \cup \overline{L} \cup D$ does not contain complementary literals. Clearly, $D|_\alpha = \top$ since α falsifies L and $D \cap \overline{L} \neq \emptyset$. Now, since D contains no literal l such that $\overline{l} \in (C \setminus L) \cup \overline{L}$ and since α_L falsifies exactly the literals in $(C \setminus L) \cup \overline{L}$, it follows that α_L does not satisfy D, hence $D|_{\alpha_L} \neq \top$. □

Theorem 4. *If a clause C is set-blocked by a set L in a formula F, it is set-propagation redundant w.r.t. F.*

Proof. Assume that C is set-blocked by L in F. We show that $F|_\alpha \supseteq F|_{\alpha_L}$, which implies that $F|_\alpha \vdash_1 F|_{\alpha_L}$, and therefore that C is set-propagation redundant w.r.t. F. Let $D|_{\alpha_L} \in F|_{\alpha_L}$. First, note that D cannot be contained

in F_L, for otherwise $D|_{\alpha_L} = \top$ and thus $D|_{\alpha_L} \notin F|_{\alpha_L}$. Second, observe that D can also not be contained in $F_{\overline{L}}$, since that would imply that $D|_\alpha = \top$ and thus, by Lemma 3, $D|_{\alpha_L} = \top$. Therefore, $D \notin F_L \cup F_{\overline{L}}$ and so $D|_\alpha = D|_{\alpha_L}$. But then, $D|_{\alpha_L} \in F|_\alpha$. It follows that $F|_\alpha \supseteq F|_{\alpha_L}$. □

We thus know that set-propagation-redundant clauses generalize both resolution asymmetric tautologies and set-blocked clauses. Since there exist resolution asymmetric tautologies that are not set-blocked (and vice versa) [7], it follows that set-propagation-redundant clauses are actually a *strict* generalization of these two kinds of clauses.

By giving practically full freedom to the witnessing assignments, i.e., by only requiring them to satisfy C, we finally arrive at propagation-redundant clauses, the most general of the three redundancy notions:

Definition 7. *Let F be a formula, C a clause, and α the assignment blocked by C. Then, C is* propagation redundant (PR) *w.r.t. F if there exists an assignment ω such that ω satisfies C and $F|_\alpha \vdash_1 F|_\omega$.*

Example 4. Let $F = \{x \vee y,\ \overline{x} \vee y,\ \overline{x} \vee z\}$, $C = x$, and let $\omega = x\,z$ be the witnessing assignment. Then, $\alpha = \overline{x}$ is the assignment blocked by C. Now, consider $F|_\alpha = \{y\}$ and $F|_\omega = \{y\}$. Clearly, unit propagation with the negated literal $\overline{y}$ of the unit clause $y \in F|_\omega$ derives a conflict on $F|_\alpha$. Therefore, $F|_\alpha \vdash_1 F|_\omega$ and so C is propagation redundant w.r.t. F. Note that C is not set-propagation redundant because for $L = \{x\}$, we have $\alpha_L = x$ and so $F|_{\alpha_L}$ contains the two unit clauses y and z, but it does not hold that $F|_\alpha \vdash_1 z$. The fact that ω satisfies z is crucial for ensuring propagation redundancy. □

Since the witnessing assignments ω are allowed to assign variables that are not contained in C, we need—at least in general—to add them to a proof to guarantee that redundancy can be efficiently checked. In the next section, we illustrate the power of a proof system that is based on the addition of PR clauses.

5 Short Proofs of the Pigeon Hole Principle

In a landmark paper, Haken [13] showed that pigeon hole formulas cannot be refuted by resolution proofs that are of polynomial size w.r.t. the size of the formulas. In contrast, by using the stronger proof system of *extended resolution*, Cook [22] proved that one can actually refute pigeon hole formulas in polynomial size. What distinguishes extended resolution from general resolution is that it allows for the introduction of new variables via definitions. Cook showed how the introduction of such definitions helps to reduce a pigeon hole formula of size n to a pigeon hole formula of size $n - 1$ over new variables. The problem with the introduction of new variables, however, is that the search space of possible variables—and therefore clauses—that could be added to a proof is infinite.

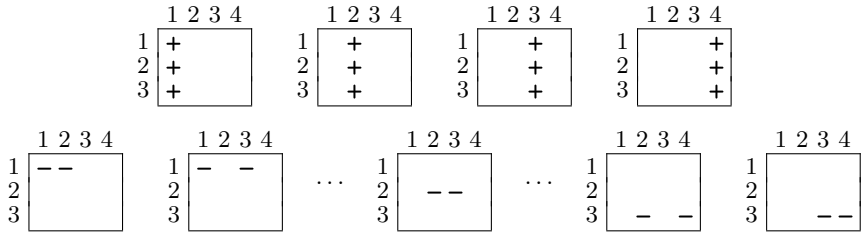
In this section, we illustrate how a clausal proof system that allows the addition of PR clauses can yield short proofs of pigeon hole formulas without the

need for introducing new variables. This shows that a proof system based on PR clauses is strictly stronger than the resolution calculus, even when we forbid the introduction of new variables. To recap, a pigeon hole formula PHP_n intuitively encodes that n pigeons have to be assigned to $n-1$ holes such that no hole contains more than one pigeon. In the encoding, a variable $x_{i,k}$ intuitively denotes that pigeon i is assigned to hole k:

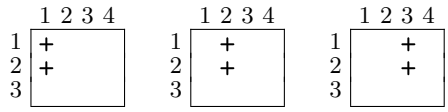
$$PHP_n := \bigwedge_{1 \leq i \leq n} (x_{i,1} \vee \cdots \vee x_{i,n-1}) \wedge \bigwedge_{1 \leq i < j \leq n} \bigwedge_{1 \leq k \leq n-1} (\overline{x}_{i,k} \vee \overline{x}_{j,k})$$

Clearly, pigeon hole formulas are unsatisfiable. The main idea behind our approach is similar to that of Cook, namely to reduce a pigeon hole formula PHP_n to the smaller PHP_{n-1}. The difference is, that in our case, PHP_{n-1} is still defined on the same variables as PHP_n. Therefore, reducing PHP_n to PHP_{n-1} boils down to deriving the clauses $x_{i,1} \vee \cdots \vee x_{i,n-2}$ for $1 \leq i \leq n-1$.

Following Haken [13], we use array notation for clauses: Every clause is represented by an array of n columns and $n-1$ rows. An array contains a "+" ("−") in the i-th column and k-th row if and only if the variable $x_{i,k}$ occurs positively (negatively, respectively) in the corresponding clause. Representing PHP_n in array notation, we have for every clause $x_{i,1} \vee \cdots \vee x_{i,n-1}$, an array in which the i-th column is filled with "+". Moreover, for every clause $\overline{x}_{i,k} \vee \overline{x}_{j,k}$, we have an array that contains two "−" in row k—one in column i and the other in column j. For instance, PHP_4 is given in array notation as follows:

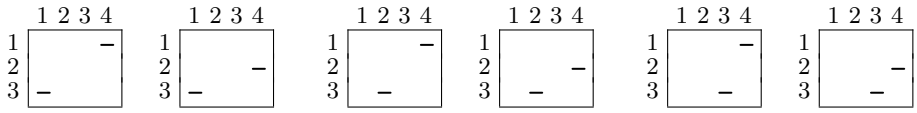


We illustrate the general idea for reducing a pigeon hole formula PHP_n to the smaller PHP_{n-1} on the concrete formula PHP_4. It should, however, become clear from our explanation that the procedure works for every $n > 1$. If we want to reduce PHP_4 to PHP_3, we have to obtain the following three clauses:



We can do so, by removing the "+" from the last row of every column full of "+", except for the last column, which can be ignored as it is not contained in PHP_3. The key observation is, that a "+" in the last row of the i-th column can be removed with the help of so-called "diagonal clauses" of the form $\overline{x}_{i,n-1} \vee \overline{x}_{n,k}$ $(1 \leq k \leq n-2)$. We are allowed to add these diagonal clauses since they are, as we will show, propagation redundant w.r.t. PHP_n. The arrays below represent

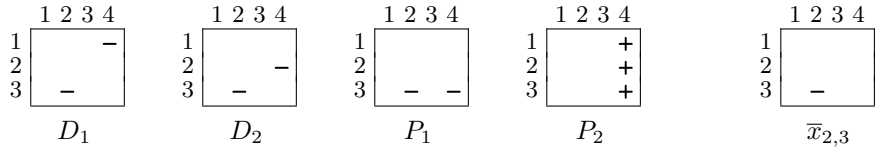
the diagonal clauses to remove the "+" from the last row of the first (left), second (middle), and third column (right):



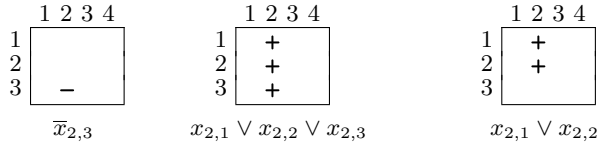
We next show how exactly these diagonal clauses allow us to remove the bottom "+" from a column full of "+", or, in other words, how they help us to remove the literal $x_{i,n-1}$ from a clause $x_{i,1} \vee \cdots \vee x_{i,n-1}$ $(1 \leq i \leq n-1)$. Consider, for instance, the clause $x_{2,1} \vee x_{2,2} \vee x_{2,3}$ in PHP_4. Our aim is to remove the literal $x_{2,3}$ from this clause. Before we explain the procedure, we like to remark that proof systems based on propagation redundancy can easily simulate resolution: Since every resolvent of clauses in a formula F is implied by F, the assignment α blocked by the resolvent must falsify F and thus $F|_\alpha \vdash_1 \bot$. We explain our procedure textually before we illustrate it in array notation:

First, we add the diagonal clauses $D_1 = \overline{x}_{2,3} \vee \overline{x}_{4,1}$ and $D_2 = \overline{x}_{2,3} \vee \overline{x}_{4,2}$ to PHP_4. After this, we can derive the unit clause $\overline{x}_{2,3}$ by resolving the two diagonal clauses D_1 and D_2 with the original pigeon hole clauses $P_1 = \overline{x}_{2,3} \vee \overline{x}_{4,3}$ and $P_2 = x_{4,1} \vee x_{4,2} \vee x_{4,3}$ as follows: We resolve D_1 with P_2 to obtain $\overline{x}_{2,3} \vee x_{4,2} \vee x_{4,3}$. Then, we resolve this clause with D_2 to obtain $\overline{x}_{2,3} \vee x_{4,3}$, which we resolve with P_1 to obtain $\overline{x}_{2,3}$. Note that our proof system actually allows us to add $\overline{x}_{2,3}$ immediately without carrying out all the resolution steps explicitly. Finally, we resolve $\overline{x}_{2,3}$ with $x_{2,1} \vee x_{2,2} \vee x_{2,3}$ to obtain the desired clause $x_{2,1} \vee x_{2,2}$.

We next illustrate this procedure in array notation. We start by visualizing the clauses D_1, D_2, P_1, and P_2 that can be resolved to yield the clause $\overline{x}_{2,3}$. The clauses are given in array notation as follows:



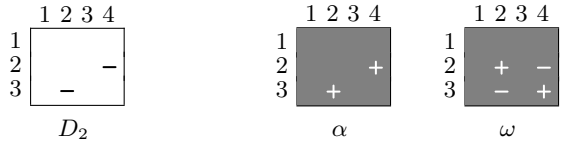
We can then resolve $\overline{x}_{2,3}$ with $x_{2,1} \vee x_{2,2} \vee x_{2,3}$ to obtain $x_{2,1} \vee x_{2,2}$:



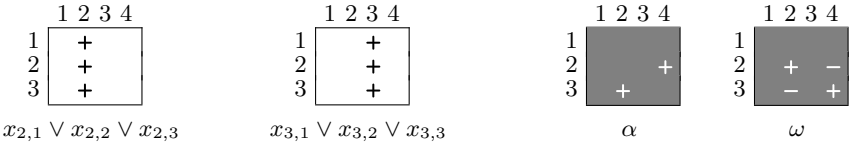
This should illustrate the general idea of how to reduce a clause of the form $x_{i,1} \vee \ldots x_{i,n-1}$ $(1 \leq i \leq n-1)$ to a clause $x_{i,1} \vee \ldots x_{i,n-2}$. By repeating this procedure for every column i with $1 \leq i \leq n-1$, we can thus reduce a pigeon hole formula PHP_n to a pigeon hole formula PHP_{n-1} without introducing new variables. Note that the last step, in which we resolve the derived unit clause $\overline{x}_{2,3}$ with the clause $x_{2,1} \vee x_{2,2} \vee x_{2,3}$, is actually not necessary for a valid PR proof of a pigeon hole formula, but we added it to simplify the presentation.

It remains to show that the diagonal clauses are indeed propagation redundant w.r.t. the pigeon hole formula. To do so, we show that for every assignment $\alpha = x_{i,n-1}\, x_{n,k}$ that is blocked by a diagonal clause $\overline{x}_{i,n-1} \vee \overline{x}_{n,k}$, it holds that for the assignment $\omega = \overline{x}_{i,n-1}\, \overline{x}_{n,k}\, x_{i,k}\, x_{n,n-1}$, $PHP_n|\alpha = PHP_n|\omega$, implying that $PHP_n|\alpha \vdash_1 PHP_n|\omega$. We also argue why other diagonal and unit clauses can be ignored when checking whether a new diagonal clause is propagation redundant.

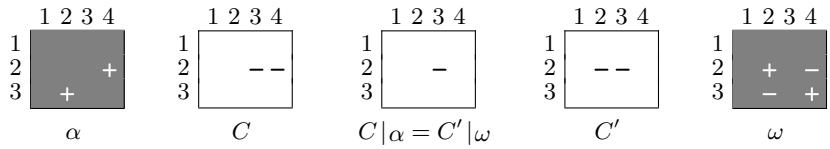
We again illustrate the idea on PHP_4. From now on, we use array notation also for assignments, i.e., a "+" ("–") in column i and row k denotes that the assignment assigns 1 (0, respectively) to variable $x_{i,k}$. Consider, for instance, the diagonal clause $D_2 = \overline{x}_{2,3} \vee \overline{x}_{4,2}$ that blocks $\alpha = x_{2,3}\, x_{4,2}$. The corresponding witnessing assignment $\omega = \overline{x}_{2,3}\, \overline{x}_{4,2}\, x_{2,2}\, x_{4,3}$ can be seen as a "rectangle" with two "–" in the corners of one diagonal and two "+" in the other corners:



To see that $PHP_4|\alpha$ and $PHP_4|\omega$ coincide on clauses $x_{i,1} \vee \cdots \vee x_{i,n-1}$, consider that whenever α and ω assign a variable of such a clause, they both satisfy the clause (since they both have a "+" in every column in which they assign a variable) and so they both remove it from PHP_4. For instance, in the following example, both α and ω satisfy $x_{2,1} \vee x_{2,2} \vee x_{2,3}$ while both do not assign a variable of the clause $x_{3,1} \vee x_{3,2} \vee x_{3,3}$:



To see that $PHP_4|\alpha$ and $PHP_4|\omega$ coincide on clauses of the form $\overline{x}_{i,k} \vee \overline{x}_{j,k}$, consider the following: If α falsifies a literal of $\overline{x}_{i,k} \vee \overline{x}_{j,k}$, then the resulting clause is a unit clause for which one of the two literals is not assigned by α (since α does not assign two variables in the same row). Now, one can show that the same unit clause is also contained in $PHP_4|\omega$, where it is obtained from another clause: Consider, for example, again the assignment $\alpha = x_{2,3}\, x_{4,2}$ and the corresponding witnessing assignment $\omega = \overline{x}_{2,3}\, \overline{x}_{4,2}\, x_{2,2}\, x_{4,3}$ from above. The assignment α turns the clause $C = \overline{x}_{3,2} \vee \overline{x}_{4,2}$ into the unit clause $C|\alpha = \overline{x}_{3,2}$. The same clause is contained in $PHP_4|\omega$, as it is obtained from $C' = \overline{x}_{2,2} \vee \overline{x}_{3,2}$ since $C'|\omega = C|\alpha = \overline{x}_{3,2}$:



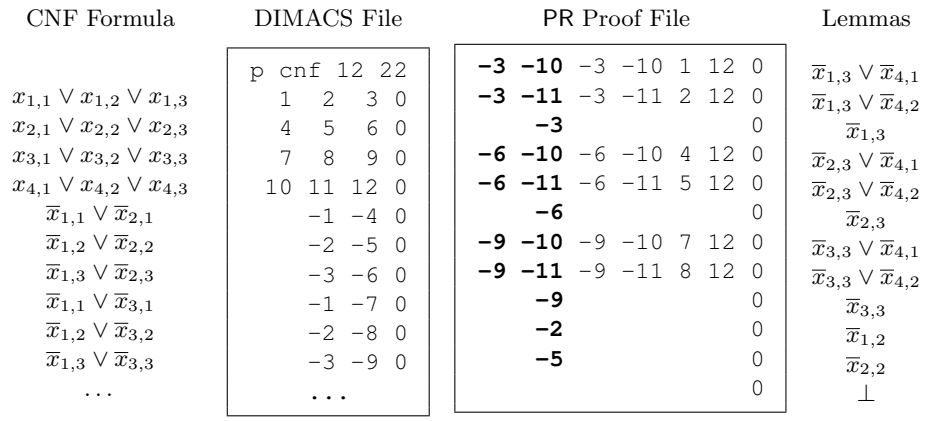


Fig. 2. Left, ten clauses of PHP_4 using the notation as elsewhere in this paper and next to it the equivalent representation of these clauses in the DIMACS format used by SAT solvers. Right, the full PR refutation consisting of clause-witness pairs. A repetition of the first literal indicates the start of the optional witness.

Note that diagonal clauses and unit clauses that have been derived earlier can be ignored when checking whether the current one is propagation redundant. For instance, assume we are currently reducing PHP_n to PHP_{n-1}. Then, the assignments α and ω under consideration only assign variables in PHP_n. In contrast, the unit and diagonal clauses used for reducing PHP_{n+1} to PHP_n (or earlier ones) are only defined on variables outside of PHP_n. They are therefore contained in both $PHP_n|\alpha$ and $PHP_n|\omega$. We can also ignore earlier unit and diagonal clauses over variables in PHP_n, i.e., clauses used for reducing an earlier column or other diagonal clauses for the current column: Whenever α assigns one of their variables, then ω satisfies them and so they are not in $PHP_n|\omega$.

Finally, we want to mention that one can also construct short SPR proofs (without new variables) of the pigeon hole formulas by first adding SPR clauses of the form $\overline{x}_{i,n-1} \vee \overline{x}_{n,k} \vee x_{i,k} \vee x_{n,n-1}$ and then turning them into diagonal clauses using resolution. We left these proofs out since they are twice as large as the PR proofs and their explanation is less intuitive. For DRAT, we consider it unlikely that such proofs exist.

6 Evaluation

We implemented a PR proof checker[4] on top of `DRAT-trim` [5]. Fig. 3 shows the pseudo code of the checking algorithm. The first "if" statement is not necessary but significantly improves the efficiency of the algorithm. The worst-case complexity of the algorithm is $\mathcal{O}(m^3)$, where m is the number of clauses in a proof. The reason for this is that there are m iterations of the outer for-loop and for

[4] The checker, benchmark formulas, and proofs are available at `http://www.cs.utexas.edu/~marijn/pr/`

```
PRcheck (CNF formula F; PR proof (C_1, ω_1), ..., (C_m, ω_m))
  for i ∈ {i, ..., m} do
    for D ∈ F do
      if D|_{ω_i} ≠ ⊤ and (D|_{α_i} = ⊤ or D|_{ω_i} ⊂ D|_{α_i}) then
        if F|_{α_i} ⊬_1 D|_{ω_i} then return failure
    F := F ∪ {C_i}
  return success
```

Fig. 3. Pseudo Code of the PR-Proof Checking Algorithm.

each of these iterations, the inner for-loop is performed $|F|$ times (i.e., once for every clause in F). Given that F contains n clauses at the start of the algorithm, we know that the size of F is bounded by $m+n$ (the original n clauses of F plus the m clauses of the proof that are added to F by the algorithm). It follows that the inner for-loop is performed $m(m+n)$ times. Now, there is a unit propagation test in the inner if-statement: If k is the maximal clause size and $m+n$ is an upper bound for the size of the formula, then the complexity of unit propagation is known to be at most $k(m+n)$. Hence, the overall worst-case complexity of the algorithm is bounded by $m(m+n)k(m+n) = \mathcal{O}(m^3)$.

This complexity is the same as for RAT-proof checking. In fact, the pseudo-code for RAT-proof checking and PR-proof checking is the same apart from the first if-statement, which is always true in the worst case, both for RAT and PR. Although the theoretical worst-case complexity makes proof checking seem very expensive, it can be done quite efficiently in practice: For the RAT proofs produced by solvers in the SAT competitions, we observed that the runtime of proof checking is close to linear with respect to the sizes of the proofs.

Moreover, we want to highlight that verifying the PR property of a clause is relatively easy as long as a witnessing assignment is given. For an arbitrary clause *without* a witnessing assignment, however, we conjecture that it is an NP-complete problem to decide whether the clause is PR. We therefore believe that in general, the verification of PR proofs is simpler than the actual solving/proving.

The format of PR proofs is an extension of DRAT proofs: the first numbers of line i denote the literals in C_i. Positive numbers refer to positive literals, and negative numbers refer to negative literals. In case a witness ω_i is provided, the first literal in the clause is repeated to denote the start of the witness. Recall that the witness always has to satisfy the clause. It is therefore guaranteed that the witness and the clause have at least one literal in common. Our format requires that such a literal occurs at the first position of the clause and of the witness. Finally, `0` marks the end of a line. Fig. 2 shows the formula and the PR proof of our running example PHP_4.

Table 1 compares our PR proofs with existing DRAT proofs of the pigeon hole formulas and of formulas from another challenging benchmark suite of the SAT competition that allow two pigeons per hole. For the latter suite, PR proofs can be constructed in a similar way as those of the classical pigeon hole formulas.

Table 1. The sizes (in terms of the number of variables and clauses) of pigeon hole formulas (top) and two-pigeons-per-hole formulas (bottom) as well as the sizes and validation times (in seconds) for their PR proofs (as described in Section 5) and their DRAT proofs (based on symmetry breaking [23]).

	input		PR proofs			DRAT proofs		
formula	#var	#cls	#var	#cls	time	#var	#cls	time
hole10.cnf	110	561	110	385	0.17	440	3,685	0.22
hole11.cnf	132	738	132	506	0.18	572	5,236	0.23
hole12.cnf	156	949	156	650	0.19	728	7,228	0.27
hole13.cnf	182	1,197	182	819	0.21	910	9,737	0.34
hole20.cnf	420	4,221	420	2,870	0.40	3,080	49,420	2.90
hole30.cnf	930	13,981	930	9,455	2.57	99,20	234,205	61.83
hole40.cnf	1,640	32,841	1,640	22,140	13.54	22,960	715,040	623.29
hole50.cnf	2,550	63,801	2,550	42,925	71.72	44,200	1,708,925	3,158.17
tph8.cnf	136	5,457	136	680	0.32	3,520	834,963	5.47
tph12.cnf	300	27,625	300	2,300	1.81	11,376	28,183,301	1,396.92
tph16.cnf	528	87,329	528	5,456	11.16	not available, too large		
tph20.cnf	820	213,241	820	10,660	61.69	not available, too large		

Notice that the PR proofs do not introduce new variables and that they contain fewer clauses than their corresponding formulas. The DRAT proof of PHP_n contains a copy of the formula PHP_k for each $k < n$. Checking PR proofs is also more efficient, as they are more compact.

7 Related Work

In this section, we shortly discuss how the concepts in this paper are related to *variable instantiation* [10], *autarkies* [8], *safe assignments* [9], and *symmetry breaking* [11]. If, for some literal l, it is possible to show $F|_{\bar{l}} \vDash F|_l$, then *variable instantiation*, as described by Andersson *et al.* [10], allows to assign the literal l in the formula F to 1. Analogously, we identify the unit clause l as redundant.

As presented by Kleine Büning and Kullmann [8], an assignment ω is an *autarky* for a formula F if it satisfies all clauses of F that contain a literal to which ω assigns a truth value. If an assignment ω is an autarky for a formula F, then F is satisfiability equivalent to $F|_\omega$. Similarly, propagation redundancy PR allows us to add all the unit clauses falsified by an autarky, with the autarky serving as a witness: Let ω be an autarky for some formula F, $C = \bar{l}$ for a literal l falsified by ω, and α the assignment blocked by C. Notice that $F|_\alpha \supseteq F|_\omega$ and thus C is propagation redundant w.r.t. F.

According to Weaver and Franco [9], an assignment ω is considered *safe* if, for every assignment α with $var(\alpha) = var(\omega)$, it holds that $F|_\alpha \vDash F|_\omega$. If an assignment ω is safe, then $F|_\omega$ is satisfiability equivalent to F. In a similar fashion, our approach allows us to block all the above-mentioned assignments $\alpha \neq \omega$. Through this, we obtain a formula that is logically equivalent to $F|_\omega$. Note that

safe assignments generalize autarkies and variable instantiation. Moreover, while safe assignments only allow the application of an assignment ω to a formula F if $F|_\alpha \vDash F|_\omega$ holds for *all* assignments $\alpha \neq \omega$, our approach enables us to block an assignment α as soon as $F|_\alpha \vDash F|_\omega$.

Finally, symmetry breaking [11] can be expressed in the DRAT proof system [23] but existing methods introduce many new variables and duplicate the input formula multiple times. It might be possible to express symmetry breaking without new variables in the PR proof system. For one important symmetry, row-interchangeability [16], the symmetry breaking using PR without new variables appears similar to the method we presented for the pigeon hole formulas.

8 Conclusion

Based on an implication relation between a formula and itself under different partial assignments, we obtain a clean and simple characterization of the most general notion of clause redundancy considered in the literature so far. Replacing the implication relation by stronger notions of implication, e.g., the superset relation or implication through unit propagation, gives then rise to various polynomially checkable redundancy criteria. One variant yields a proof system that turns out to coincide with the well-known DRAT, while we conjecture the proof systems produced by the other two variants to be much more powerful. We showed that these more general variants admit short clausal proofs for the famous pigeon hole formulas, without the need to introduce new variables. Experiments show that our proofs are much more compact than existing clausal proofs and also much faster to check. Our new proof systems simulate many other concepts from the literature very concisely, including autarkies, variable instantiation, safe assignments, and certain kinds of symmetry reasoning.

Interesting future work includes the separation of our new proof systems from the DRAT proof system on the lower end and from extended resolution on the upper end, under the additional restriction that our proof systems and DRAT do not introduce new variables. The relation to extended resolution is a particularly interesting aspect from the proof complexity point of view. Other open questions are related to the space and width bounds of the smallest PR proofs, again without new variables, for well-known other hard problems such as Tseitin formulas [12,24] or pebbling games [25]. On the practical side, we want to implement a formally verified proof checker for PR proofs. Moreover, we want to pursue some preliminary ideas for automatically generating short PR proofs during actual SAT solving: Our initial plan is to enumerate unit and binary clauses and to add them to a formula if they are propagation redundant. We already have a prototype implementation which is able to find short proofs of pigeon hole formulas, but we are still searching for efficient heuristics that help solvers with finding short PR clauses in general formulas.

References

1. Clarke, E.M., Biere, A., Raimi, R., Zhu, Y.: Bounded model checking using satisfiability solving. Formal Methods in System Design **19**(1) (2001) 7–34
2. Ivančić, F., Yang, Z., Ganai, M.K., Gupta, A., Ashar, P.: Efficient SAT-based bounded model checking for software verification. Theoretical Computer Science **404**(3) (2008) 256–274
3. Konev, B., Lisitsa, A.: Computer-aided proof of Erdős discrepancy properties. Artificial Intelligence **224**(C) (July 2015) 103–118
4. Heule, M.J.H., Kullmann, O., Marek, V.W.: Solving and verifying the Boolean Pythagorean Triples problem via Cube-and-Conquer. In: Proc. of the 19th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2016). Volume 9710 of LNCS., Cham, Springer (2016) 228–245
5. Wetzler, N.D., Heule, M.J.H., Hunt Jr., W.A.: DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In: Proc. of the 17th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2014). Volume 8561 of LNCS., Cham, Springer (2014) 422–429
6. Järvisalo, M., Heule, M.J.H., Biere, A.: Inprocessing rules. In: Proc. of the 6th Int. Joint Conference on Automated Reasoning (IJCAR 2012). Volume 7364 of LNCS., Heidelberg, Springer (2012) 355–370
7. Kiesl, B., Seidl, M., Tompits, H., Biere, A.: Super-blocked clauses. In: Proc. of the 8th Int. Joint Conference on Automated Reasoning (IJCAR 2016). Volume 9706 of LNCS., Cham, Springer (2016) 45–61
8. Kleine Büning, H., Kullmann, O.: Minimal unsatisfiability and autarkies. In Biere, A., Heule, M.J.H., van Maaren, H., Walsh, T., eds.: Handbook of Satisfiability. IOS Press (2009) 339–401
9. Weaver, S., Franco, J.V., Schlipf, J.S.: Extending existential quantification in conjunctions of BDDs. JSAT **1**(2) (2006) 89–110
10. Andersson, G., Bjesse, P., Cook, B., Hanna, Z.: A proof engine approach to solving combinational design automation problems. In: Proc. of the 39th Annual Design Automation Conference (DAC 2002), ACM (2002) 725–730
11. Crawford, J., Ginsberg, M., Luks, E., Roy, A.: Symmetry-breaking predicates for search problems. In: Proc. of the 5th Int. Conference on Principles of Knowledge Representation and Reasoning (KR 1996), Morgan Kaufmann (1996) 148–159
12. Tseitin, G.S.: On the complexity of derivation in propositional calculus. Studies in Mathematics and Mathematical Logic **2** (1968) 115–125
13. Haken, A.: The intractability of resolution. Theoretical Computer Science **39** (1985) 297–308
14. Audemard, G., Katsirelos, G., Simon, L.: A restriction of extended resolution for clause learning sat solvers. In: Proc. of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010), AAAI Press (2010)
15. Manthey, N., Heule, M.J.H., Biere, A.: Automated reencoding of boolean formulas. In: Proc. of the 8th Int. Haifa Verification Conference (HVC 2012). Volume 7857 of LNCS., Heidelberg, Springer (2013)
16. Devriendt, J., Bogaerts, B., Bruynooghe, M., Denecker, M.: Improved static symmetry breaking for SAT. In: Proc. of the 19th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2016). Volume 9710 of LNCS., Cham, Springer (2016) 104–122
17. Balyo, T., Heule, M.J.H., Järvisalo, M.: SAT competition 2016: Recent developments. To appear in: Proc. of the 31st AAAI Conference on Artificial Intelligence (AAAI 2017), AAAI Press (2017)

18. Goldberg, E.I., Novikov, Y.: Verification of proofs of unsatisfiability for CNF formulas. In: Proc. of the Conference on Design, Automation and Test in Europe (DATE 2003), IEEE Computer Society (2003) 10886–10891
19. Van Gelder, A.: Producing and verifying extremely large propositional refutations. Annals of Mathematics and Artificial Intelligence **65**(4) (2012) 329–372
20. Kullmann, O.: On a generalization of extended resolution. Discrete Applied Mathematics **96-97** (1999) 149–176
21. Järvisalo, M., Biere, A., Heule, M.J.H.: Simulating circuit-level simplifications on CNF. Journal on Automated Reasoning **49**(4) (2012) 583–619
22. Cook, S.A.: A short proof of the pigeon hole principle using extended resolution. SIGACT News **8**(4) (October 1976) 28–32
23. Heule, M.J.H., Hunt Jr., W.A., Wetzler, N.D.: Expressing symmetry breaking in DRAT proofs. In: Proc. of the 25th Int. Conference on Automated Deduction (CADE 2015). Volume 9195 of LNCS., Cham, Springer (2015) 591–606
24. Urquhart, A.: The complexity of propositional proofs. The Bulletin of Symbolic Logic **1**(4) (1995) 425–467
25. Nordström, J.: A simplified way of proving trade-off results for resolution. Information Processing Letters **109**(18) (August 2009) 1030–1035

A Little Blocked Literal Goes a Long Way*

Benjamin Kiesl[1], Marijn J.H. Heule[2], and Martina Seidl[3]

[1] Institute of Information Systems, Vienna University of Technology
[2] Department of Computer Science, The University of Texas at Austin
[3] Institute for Formal Models and Verification, JKU Linz

Abstract. Q-resolution is a generalization of propositional resolution that provides the theoretical foundation for search-based solvers of quantified Boolean formulas (QBFs). Recently, it has been shown that an extension of Q-resolution, called long-distance resolution, is remarkably powerful both in theory and in practice. However, it was unknown how long-distance resolution is related to QRAT, a proof system introduced for certifying the correctness of QBF-preprocessing techniques. We show that QRAT polynomially simulates long-distance resolution. Two simple rules of QRAT are crucial for our simulation—*blocked-literal addition* and *blocked-literal elimination*. Based on the simulation, we implemented a tool that transforms long-distance-resolution proofs into QRAT proofs. In a case study, we compare long-distance-resolution proofs of the well-known Kleine Büning formulas with corresponding QRAT proofs.

1 Introduction

Quantified Boolean formulas (QBF) [19] extend propositional formulas with existential and universal quantifiers over the propositional variables. These quantifiers lead to increased expressiveness, which makes QBF attractive for reasoning problems in areas such as formal verification and artificial intelligence [3].

To obtain a better understanding of the strengths and limitations of different QBF-solving approaches, their underlying proof systems have been extensively analyzed, providing a comprehensive proof-complexity landscape for QBF [6, 10, 9, 4, 16]. Two kinds of proof systems have received particular attention: instantiation-based proof systems [5, 6], which provide the foundation for expansion-based solvers like `RAReQS` [18], and resolution-based proof systems [16, 20, 26, 1, 24, 2, 7, 17, 23], which provide the foundation for search-based solvers like `DepQBF` [22]. Apart from these, also sequent systems have been studied [10, 8]. There is, however, another practically useful proof system—quite different from the aforementioned ones—whose place in the complexity landscape was still unclear: the QRAT proof system [15].

The QRAT proof system is a generalization of DRAT [25] (the de-facto standard for proofs in practical SAT solving) that has its strengths when it comes

* This work has been supported by the Austrian Science Fund (FWF) under projects W1255-N23 and S11408-N23, and by the National Science Foundation (NSF) under grant number CCF-1618574.

to preprocessing: Many QBF solvers use preprocessing techniques to simplify a QBF before they actually evaluate its truth. With the QRAT system, it is possible to certify the correctness of virtually all preprocessing simplifications performed by state-of-the-art QBF solvers and preprocessors. Additionally, there exist efficient tools for checking the correctness of QRAT proofs as well as for extracting winning strategies (so-called *Skolem functions*) from QRAT proofs of satisfiability [15].

It can be easily seen that QRAT simulates the basic Q-resolution calculus [20] that allows only resolution upon existential variables. Likewise, it simulates the calculus QU-Res [24], which extends Q-resolution by allowing resolution upon universal variables. So far, however, it was unclear how QRAT is related to the long-distance-resolution calculus [26, 1]—a calculus that is particularly popular because it allows for short proofs both in theory and in practice [11].

In this paper, we prove that QRAT can polynomially simulate the long-distance-resolution calculus. For our simulation, we need only Q-resolution and universal reduction together with blocked-literal elimination and blocked-literal addition using fresh variables [14, 21]. These four rules are allowed in QRAT. To illustrate the power of blocked literals, we present handcrafted QRAT proofs of the formulas commonly used to display the strength of long-distance resolution—the well-known *Kleine Büning formulas* [20]. Our proofs are slightly smaller than the long-distance resolution proofs of these formulas described by Egly et al. [11].

To put our simulation into practice, we implemented a tool that transforms long-distance-resolution proofs into QRAT proofs. With this tool it is now possible to obtain QRAT proofs that certify the correctness of both the preprocessing and the actual solving, even when using a QBF solver based on long-distance resolution. We used our tool to transform long-distance-resolution proofs of the Kleine Büning formulas into QRAT proofs. We compare the resulting proofs with the handcrafted QRAT proofs as well as with the original proofs. Rounding off the picture, we locate QRAT in the proof-complexity landscape of resolution-based proof systems and discuss open questions.

2 Preliminaries

In the following, we introduce the background required to understand the rest of the paper. A *literal* is either a variable x (a *positive literal*) or the negation $\bar{x}$ of a variable x (a *negative literal*). The complement $\bar{l}$ of a literal l is defined as $\bar{x}$ if $l = x$ and as x if $l = \bar{x}$. A *clause* is a disjunction of literals. A (*propositional*) *formula* in *conjunctive normal form* (CNF) is a conjunction of clauses. A clause can be seen as a set of literals and a formula can be seen as a set of clauses.

A *quantifier prefix* has the form $Q_1 X_1 \dots Q_q X_q$, where all the X_i are mutually disjoint sets of variables, $Q_i \in \{\forall, \exists\}$, and $Q_i \neq Q_{i+1}$. A *quantified Boolean formula* (QBF) ϕ in *prenex conjunctive normal form* (PCNF) is of the form $\Pi.\psi$ where Π is a quantifier prefix and ψ, called the *matrix* of ϕ, is a propositional formula in CNF. The quantifier $Q(\Pi, l)$ of a literal l is Q_i if $var(l) \in X_i$. Let $Q(\Pi, l) = Q_i$ and $Q(\Pi, k) = Q_j$, then $l \leq_\Pi k$ if $i \leq j$, and $l <_\Pi k$ if $i < j$.

Using the truth constants 1 (*true*) and 0 (*false*), a QBF $\forall x \Pi.\psi$ is false iff at least one of $\Pi.\psi[x/1]$ and $\Pi.\psi[x/0]$ is false where $\Pi.\psi[x/t]$ is obtained from $\Pi.\psi$ by replacing all occurrences of x in ψ by t and removing x from Π. Respectively, a QBF $\exists x \Pi.\psi$ is false iff both $\Pi.\psi[x/1]$ and $\Pi.\psi[x/0]$ are false. If the matrix ψ of ϕ contains the empty clause (denoted by $\bot$) after eliminating the truth constants according to standard rules, then ϕ is false. If ψ is empty, ϕ is true.

2.1 Resolution-Based Calculi

In resolution-based calculi, a proof P of a QBF $\Pi.\psi = \Pi.C_1 \wedge \cdots \wedge C_m$ is a sequence $C_{m+1}, \ldots, C_n$ of clauses with $C_n = \bot$ and for every C_i $(m+1 \leq i \leq n)$, it holds that C_i has been derived from clauses in ψ or from earlier clauses in P (i.e., from clauses with index strictly smaller than i) by applications of either the $\forall$-*red* rule (also called *universal reduction*) or instantiations of the *resolution* rule which are defined as follows:

$$\frac{C \vee x}{C}\ (\forall\text{-red}) \qquad \frac{C \vee l \qquad D \vee \bar{l}}{C \vee D}\ (\text{resolution})$$

The rule $\forall$-red is only applicable if the literal x is universal and if for every existential literal $l \in C$, it holds that $l <_\Pi x$. In the resolution rule, the resolvent $C \vee D$ is derived from its two antecedent clauses. We assume that no clause in ψ contains complementary literals, otherwise the $\forall$-red rule is unsound.

The most basic resolution-based calculus for QBF is the *Q-resolution calculus* (Q-Res) [20]. It uses the resolution rule *Q-res* which requires that (1) l is existential and (2) C does not contain a literal x such that $\bar{x} \in D$. In contrast, the *long-distance-resolution calculus* (LQ-Res) [26, 1] uses a less restrictive variant of the resolution rule, called *LQ-res*, which requires that (1) l is existential and (2) for every literal $x \in C$ such that $\bar{x} \in D$, it holds that x is universal and $l <_\Pi x$. Note that every Q-res step is also an LQ-res step. In the rest of the paper, we refer to resolution steps as LQ-res steps only if they are not Q-res steps, otherwise we refer to them as Q-res steps. Note that in the literature a complementary pair $x, \bar{x}$ is also represented by a so-called *merged literal* x^*.

Example 1. Consider the QBF $\phi = \exists a \forall x \exists b \exists c.(\bar{a} \vee \bar{x} \vee c) \wedge (\bar{x} \vee b \vee \bar{c}) \wedge (a \vee x \vee b) \wedge (\bar{b})$. The following is a long-distance-resolution proof of ϕ: $\bar{a} \vee \bar{x} \vee b$, $x \vee \bar{x} \vee b$, $x \vee \bar{x}$, x, $\bot$. We explain this proof in more detail later (also see Fig. 1 on page 5).

2.2 The QRAT Proof System Light

In this paper, we do not need the power of the full QRAT proof system [15]. We therefore introduce only a very restricted version of QRAT that is sufficient for the simulation of the long-distance-resolution calculus.

One of the main concepts in this variant of QRAT is the concept of a *blocked literal*. For the definition of blocked literals, we first have to introduce so-called *outer resolvents*. Given two clauses $C \vee x, D \vee \bar{x}$ of a QBF $\Pi.\psi$, the outer resolvent $C \vee x \bowtie^x_\Pi D \vee \bar{x}$ of $C \vee x$ with $D \vee \bar{x}$ upon x is the clause consisting of all literals in C together with those literals of D that occur outer to x, i.e., the outer resolvent is the clause $C \cup \{l \mid l \in D \text{ and } l \leq_\Pi x\}$. We can now define blocked literals:

Definition 1. *A universal literal x is* blocked *in a clause $C \vee x$ w.r.t. a QBF $\Pi.\psi$ if, for every clause $D \vee \bar{x} \in \psi \setminus \{C \vee x\}$, the outer resolvent $C \vee x \bowtie^x_\Pi D \vee \bar{x}$ contains a pair of complementary literals.*

Example 2. Let $\phi = \exists a \forall x, y \exists b.(a \vee x \vee y) \wedge (\bar{a} \vee \bar{x} \vee b) \wedge (\bar{y} \vee \bar{x} \vee b)$. The literal x is blocked in $a \vee x \vee y$ w.r.t. ϕ: There are two outer resolvents of $a \vee x \vee y$ upon x, namely $a \vee y \vee \bar{a}$, obtained by resolving with $\bar{a} \vee \bar{x} \vee b$, and $a \vee y \vee \bar{y}$, obtained by resolving with $\bar{y} \vee \bar{x} \vee b$. Both contain a pair of complementary literals. □

If a literal is blocked in a clause, its removal is called *blocked-literal elimination* (BLE) [14]. If, after adding a literal to a clause, the literal is blocked in that clause, then this addition is called *blocked-literal addition* (BLA). Both BLE and BLA do not change the truth value of a formula.

In our restricted variant of QRAT, a *derivation* for a QBF ϕ is a sequence $M_1, \ldots, M_n$ of proof steps. Starting with $\phi_0 = \phi$, every M_i modifies ϕ_{i-1} in one of the following four ways, which results in a new formula ϕ_i: (1) It adds to ϕ_{i-1} a clause that is derived from two clauses in ϕ_{i-1} via a resolution step. (2) It adds to ϕ_{i-1} a clause C that is obtained from a clause $C \vee x \in \phi_{i-1}$ by a $\forall$-red step, with the additional restriction that C does not contain $\bar{x}$. (3) It adds a blocked literal to a clause in ϕ_{i-1}. (4) It removes a blocked literal from a clause in ϕ_{i-1}.

A QRAT derivation $M_1, \ldots, M_n$ therefore gradually derives new formulas $\phi_1, \ldots, \phi_n$ from the starting formula ϕ. If the final formula ϕ_n contains the empty clause $\bot$, then the derivation is a (*refutation*) *proof* of ϕ. Note that the $\forall$-red rule in QRAT is more restricted than the $\forall$-red rule from the resolution-based calculi, making it sound also when clauses contain complementary literals.

To simplify the presentation, we do not specify how the modification steps M_i are represented syntactically. We also do not include clause deletion. Note that certain proof steps can modify the quantifier prefix by introducing new or removing existing variables. Note also that Q-resolution proofs do not contain complementary literals, so they can be simply rewritten into QRAT proofs using only Q-res and $\forall$-red steps. Finally, we want to highlight that for our simulation, we do not need the unrestricted resolution rule; the Q-res rule suffices.

3 Illustration of the Simulation

We start by illustrating on an example how our restricted variant of QRAT can simulate the long-distance-resolution calculus. As already mentioned, the $\forall$-red rule used in QRAT is more restricted than the one in the long-distance-resolution calculus because it does not allow us to remove a literal x from a clause that contains $\bar{x}$. This means that once we derive a clause that contains both a literal x and its complement $\bar{x}$, we cannot simply get rid of the two literals by using the $\forall$-red rule. We therefore want to avoid the derivation of clauses with complementary literals entirely. Now, the only way the long-distance-resolution calculus can derive such clauses is via resolution (LQ-res) steps. So to avoid the complementary literals, we eliminate them already before performing the resolution steps. We demonstrate this on an example:

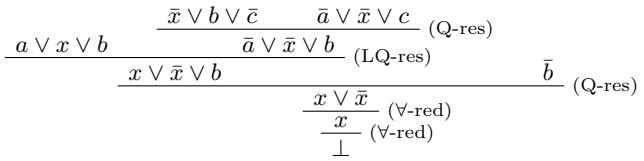


Fig. 1. LQ-res proof of QBF $\phi = \exists a \forall x \exists b \exists c.(\bar{a} \vee \bar{x} \vee c) \wedge (\bar{x} \vee b \vee \bar{c}) \wedge (a \vee x \vee b) \wedge (\bar{b})$.

Example 3. Consider the QBF $\phi = \exists a \forall x \exists b \exists c.(\bar{a} \vee \bar{x} \vee c) \wedge (\bar{x} \vee b \vee \bar{c}) \wedge (a \vee x \vee b) \wedge (\bar{b})$ from Example 1. To increase readability, we illustrate its long-distance-resolution proof as a proof tree in Fig. 1. To simulate this proof with QRAT, we first add the resolvent $\bar{a} \vee \bar{x} \vee b$ to ϕ via a Q-res step to obtain the new formula ϕ'. Now we cannot simply perform the next derivation step (the LQ-res step) because the resulting resolvent $x \vee \bar{x} \vee b$ would contain complementary literals. To deal with this, we try to eliminate x from the clause $a \vee x \vee b$. This is where the addition and elimination of blocked literals come into play.

We cannot yet eliminate x from ϕ' because x is not blocked in $a \vee x \vee b$ with respect to ϕ': For x to be blocked, all outer resolvents of $a \vee x \vee b$ upon x must contain complementary literals. The clauses that can be resolved with $a \vee x \vee b$ are $\bar{a} \vee \bar{x} \vee c$, $\bar{a} \vee \bar{x} \vee b$, and $\bar{x} \vee b \vee \bar{c}$. While the outer resolvents with the former two clauses contain the complementary literals a and $\bar{a}$, the outer resolvent $a \vee b$, obtained by resolving with $\bar{x} \vee b \vee \bar{c}$, does not contain complementary literals.

Now we use a feature of QRAT to make x blocked in $a \vee x \vee b$: We add a new literal x' (which goes to the same quantifier block as x) to $a \vee x \vee b$ to turn it into $a \vee x' \vee x \vee b$. The addition of x' is clearly a blocked-literal addition as there are no outer resolvents of $a \vee x' \vee x \vee b$ upon x'. Likewise, we add the complement $\bar{x}'$ of x' to $\bar{x} \vee b \vee \bar{c}$ to turn it into $\bar{x}' \vee \bar{x} \vee b \vee \bar{c}$. Again this is a blocked-literal addition since $a \vee x' \vee x \vee b$ (which is the only clause containing the complement x' of $\bar{x}'$) contains x while $\bar{x}' \vee \bar{x} \vee b \vee \bar{c}$ contains $\bar{x}$.

Now the complementary pair $x', \bar{x}'$ is contained in the outer resolvent of $a \vee x' \vee x \vee b$ with $\bar{x}' \vee \bar{x} \vee b \vee \bar{c}$ upon x. Thus, the literal x becomes blocked in $a \vee x' \vee x \vee b$ and so we can remove it to obtain $a \vee x' \vee b$. We have thus replaced x in $a \vee x \vee b$ by x' and now we can resolve $a \vee x' \vee b$ with $\bar{a} \vee \bar{x} \vee b$ upon a to obtain the resolvent $x' \vee \bar{x} \vee b$ (instead of $x \vee \bar{x} \vee b$ as in the original proof). Finally, we resolve $x' \vee \bar{x} \vee b$ with $\bar{b}$ to obtain $x' \vee \bar{x}$ from which we derive the empty clause $\bot$ via $\forall$-red steps. □

To summarize, we start by adding clauses of a given long-distance-resolution proof to our formula until we bump into an LQ-res step. To avoid complementary literals in the resolvent of the LQ-res step, we then use blocked-literal addition and blocked-literal elimination to replace these literals. After this, we can derive a resolvent without complementary literals and move on until we encounter the next LQ-res step, which we again eliminate. We repeat this procedure until the whole long-distance-resolution proof is turned into a QRAT proof.

Note that the modification of existing clauses has an impact on later derivations. For instance, by replacing $a \vee x \vee b$ in the above example with $a \vee x' \vee b$, we not only affected the immediate resolvent $x \vee \bar{x} \vee b$, which we turned into $x' \vee \bar{x} \vee b$, but also the later resolvent $x \vee \bar{x}$, which became $x' \vee \bar{x}$. We therefore have to show that these modifications are harmless in the sense that they do not lead to an invalid proof. We do so in the next section, where we define our simulation in detail before proving that it indeed produces a valid QRAT proof.

4 Simulation

We first describe our simulation procedure on a high level before we specify the details and prove its correctness. As we have seen, given a long-distance-resolution proof, we can use QRAT to derive all clauses up to the first LQ-res step. The crucial part of the simulation is then the elimination of complementary literals from this LQ-res step, which might involve the modification of several clauses via the addition and elimination of blocked literals.

Let $\phi = \Pi.C_1 \wedge \cdots \wedge C_m$ be a QBF and $P = C_{m+1}, \ldots, C_r, \ldots, C_n$ be a long-distance-resolution proof of ϕ where C_r is the first clause derived via an LQ-res step. If there is no such C_r, the proof can be directly translated to QRAT. Otherwise, in a first step, our procedure produces a QRAT derivation that adds all the clauses $C_{m+1}, \ldots, C_{r-1}$ to ϕ by using Q-res and $\forall$-red steps. It then uses blocked-literal addition and blocked-literal elimination to avoid complementary literals in the resolvent C_r, which it thereby turns into a different resolvent C'_r. After this, it adds C'_r to ϕ via a Q-res step. The result is a QRAT derivation of a formula ϕ' from ϕ. We explain this first step in Section 4.1.

In a second step, the procedure first removes all the clauses $C_{m+1}, \ldots, C_r$ from P since they—or their modified variants—are now all contained in ϕ'. As several clauses have been modified via blocked-literal addition and blocked-literal elimination in the first step, it then propagates these modifications through the remaining part of P. This turns P into a long-distance resolution proof P' of ϕ'. We explain this second step in Section 4.2.

By repeating these two steps for every LQ-res step, we finally obtain a QRAT proof of ϕ. Thus, we have to show that after the above two steps (i.e., after one iteration of our procedure), ϕ' is obtained by a valid QRAT derivation and the proof P' is a valid long-distance-resolution proof of ϕ' that is shorter than P. The correctness of the simulation follows then simply by induction.

To simplify the presentation, we assume that the long-distance resolvent C_r contains only one pair of complementary literals, i.e., $C_r = C \vee D \vee x \vee \bar{x}$ was derived from two clauses $C \vee l \vee x$ and $D \vee \bar{l} \vee \bar{x}$ where C does not contain a literal k such that $\bar{k}$ is contained in D. Although this assumption leads to a loss of generality, we show later that our argument can be easily extended to the more general case where C and D are allowed to contain multiple pairs of complementary literals.

4.1 QRAT Derivation of the Formula ϕ'

Below we describe the QRAT derivation of ϕ' from ϕ. Initially, $\phi' = \phi$.

1. Add the clauses $C_{m+1}, \dots C_{r-1}$ to ϕ' via Q-res and $\forall$-red steps.
2. Consider the LQ-res step that derived $C_r = C \vee D \vee x \vee \bar{x}$ from two clauses $C \vee l \vee x$ and $D \vee \bar{l} \vee \bar{x}$:
$$\frac{C \vee l \vee x \qquad D \vee \bar{l} \vee \bar{x}}{C \vee D \vee x \vee \bar{x}} \text{ (LQ-res)}$$
Towards making x blocked in $C \vee l \vee x$, add a new literal x' (that goes to the same quantifier block as x) to $C \vee l \vee x$ to turn it into $C \vee l \vee x' \vee x$.
3. Add $\bar{x}'$ to each clause $C_i \in \phi'$ for which (1) $\bar{x} \in C_i$, and (2) the outer resolvent of $C \vee l \vee x' \vee x$ and C_i upon x is not a tautology.
4. Now x is a blocked literal in $C \vee l \vee x' \vee x$. Eliminate it to obtain $C \vee l \vee x'$.
5. Add the clause $C \vee D \vee x' \vee \bar{x}$ to ϕ' by performing a Q-res step of $C \vee l \vee x'$ and $D \vee \bar{l} \vee \bar{x}$ upon l.

To see that this results in a valid QRAT derivation, observe the following: In step 2, the addition of x' is a blocked-literal addition, since $\bar{x}'$ is not contained in any of the clauses. In step 3, for every C_i with $\bar{x} \in C_i$, the addition of $\bar{x}'$ is a blocked-literal addition as only $C \vee l \vee x' \vee x$ can be resolved with C_i upon $\bar{x}'$ and the corresponding outer resolvent contains x and $\bar{x}$. Note that instead of eliminating x from $C \vee l \vee x$, we could have also eliminated $\bar{x}$ from $D \vee \bar{l} \vee \bar{x}$. It remains to modify the long-distance-resolution proof P of ϕ so that it becomes a valid proof of ϕ'.

4.2 Modification of the Long-Distance-Resolution Proof

We next turn the proof $P = C_{m+1}, \dots, C_r, \dots, C_n$ of ϕ into a proof P' of ϕ'. First, we remove the clauses $C_{m+1}, \dots, C_r$ from P since ϕ' already contains variants $C'_{m+1}, \dots, C'_r$ of these clauses. Second, since we have modified the clauses in ϕ', we have to propagate these modifications through the remaining proof.

Assume, for instance, that in P the clause C_{r+1} has been obtained by resolving a clause C_i with a clause C_j. Both C_i and C_j might have been affected by blocked-literal additions so that they are now different clauses $C'_i, C'_j \in \phi'$. To account for these modifications of C_i and C_j, we replace C_{r+1} in P by the resolvent of C'_i and C'_j. Moreover, in cases where P removes x from a clause via a $\forall$-red step, we now also remove x'. Analogously for $\bar{x}'$ and $\bar{x}$.

To formalize these modifications, we first assign to every clause C_i with $1 \leq i \leq r$ its corresponding clause of ϕ' as follows:

$$C'_i = \begin{cases} C_i \cup \{\bar{x}'\} & \text{if } \bar{x} \in C_i \text{ and the outer resolvent of } C \vee l \vee x \vee x' \\ & \text{and } C_i \text{ upon } x \text{ is not a tautology;} \\ (C_i \setminus \{x\}) \cup \{x'\} & \text{if } C_i = C_r \text{ or } C_i = C \vee l \vee x; \\ C_i & \text{otherwise.} \end{cases}$$

Note that, by construction, $C'_i \in \phi'$ for $1 \leq i \leq r$. For every i such that $r < i \leq n$, we step-by-step, starting with $i = r + 1$, define C'_i based on the derivation rule that was used for deriving C_i in P. We distinguish between clauses derived by resolution steps and clauses derived by $\forall$-red steps:

Case 1: C_i has been derived via a resolution step of two clauses $C_j = C \vee l$ and $C_k = D \vee \bar{l}$ upon l, i.e., $C_i = C \vee D$. We define $C'_i = C'_j \setminus \{l\} \vee C'_k \setminus \{\bar{l}\}$.

Case 2: C_i has been derived from a clause C_j via a $\forall$-red step. If the $\forall$-red step removes a literal l with $var(l) \neq var(x)$, we define $C'_i = C'_j \setminus \{l\}$. If it removes x, we define $C'_i = C'_j \setminus \{x, x'\}$, and if it removes $\bar{x}$, we define $C'_i = C'_j \setminus \{\bar{x}, \bar{x}'\}$.

Note that $\forall$-red steps of x and $\bar{x}$ in P' might remove two literals at once. Although such $\forall$-red steps do not constitute valid derivation steps in a strict sense, this is not a serious problem: These steps can be easily rewritten into two distinct $\forall$-red steps since x and x' are in the same quantifier block. For instance, the left step below can be rewritten into the two steps on the right:

$$\frac{C \vee x \vee x'}{C}\ (\forall\text{-red}) \qquad \frac{\dfrac{C \vee x \vee x'}{C \vee x}\ (\forall\text{-red})}{C}\ (\forall\text{-red})$$

Next, we show that the resulting proof P' is—apart from the minor detail just mentioned—a valid long-distance-resolution proof of ϕ'.

4.3 Correctness of the Simulation

To prove the correctness of our simulation, we first introduce a lemma that guarantees that the modified long-distance-resolution proof P' has a similar structure as the original proof P:

Lemma 1. *Let $\phi' = \Pi'.C'_1 \wedge \cdots \wedge C'_r$ and $P' = C'_{r+1}, \ldots, C'_n$ be obtained from $\phi = \Pi.C_1 \wedge \cdots \wedge C_m$ and $P = C_{m+1}, \ldots, C_r, \ldots, C_n$ as defined above. Then, for every clause C'_i with $1 \leq i \leq n$, the following holds: (1) If x' or x is in C'_i, then $x \in C_i$. (2) If $\bar{x}'$ or $\bar{x}$ is in C'_i, then $\bar{x} \in C_i$. (3) C'_i agrees with C_i on all literals whose variables are different from x and x', i.e., $C'_i \setminus \{x, \bar{x}, x', \bar{x}'\} = C_i \setminus \{x, \bar{x}\}$.*

Proof. By induction on i.

Base Case ($i \leq r$): The claim holds by the definition of C'_i.

Induction Step ($r < i$): Consider the clause C_i in P that corresponds to C'_i. We proceed by a case distinction based on how C_i was derived in P.

Case 1: C_i is a resolvent $C_j \setminus \{l\} \vee C_k \setminus \{\bar{l}\}$ of two clauses C_j, C_k. In this case, $C'_i = C'_j \setminus \{l\} \vee C'_k \setminus \{\bar{l}\}$. By the induction hypothesis, the statement holds for C'_j and C'_k. Now, if C'_i contains x' or x, then at least one of C'_j and C'_k must contain x' or x and thus one of C_j and C_k must contain x, hence $x \in C_i$. Analogously, if C'_i contains $\bar{x}'$ or $\bar{x}$, then C_i contains $\bar{x}$. Now, C'_j agrees with C_j on all literals

whose variables are different from x and x', and the same holds for C'_k and C_k. Thus, C'_i agrees with C_i on all literals whose variables are different from x and x'.

Case 2: C_i has been derived from a clause C_j via a $\forall$-red step, i.e., $C_i = C_j \setminus \{y\}$ for some universal literal y. By the induction hypothesis, the statement holds for C'_j. If $var(y) \neq var(x')$, then $C'_i = C'_j \setminus \{y\}$ and thus the claim holds. If $y = x$, then $C'_i = C'_j \setminus \{x, x'\}$ and thus the claim holds too. The case where $y = \bar{x}$ is analogous to the case where $y = x$. □

We can now show that the proof P', produced by our simulation procedure, is a valid long-distance-resolution proof of ϕ':

Theorem 2. *Let $\phi' = \Pi'.C'_1 \wedge \cdots \wedge C'_r$ and $P' = C'_{r+1}, \ldots, C'_n$ be obtained from $\phi = \Pi.C_1 \wedge \cdots \wedge C_m$ and $P = C_{m+1}, \ldots, C_r, \ldots, C_n$ by our procedure. Then, P' is a valid long-distance-resolution proof of ϕ'.*

Proof. We have to show that every clause C'_i in P' has been derived from clauses in $C'_1, \ldots, C'_{i-1}$ via a valid application of a derivation rule and that $C'_n = \bot$. To show that every clause in P' has been derived via a valid application of a derivation rule, let C'_i be a clause in P'. We proceed by a case distinction based on the rule via which its counterpart C_i has been derived in P:

Case 1: C_i has been derived from two clauses C_j, C_k via a Q-res step or an LQ-res step upon some existential literal l. In this case, $C'_i = C'_j \setminus \{l\} \vee C'_k \setminus \{\bar{l}\}$. We have to show that $l \in C'_j$, $\bar{l} \in C'_k$, and for every literal $l' \in C'_j$ such that $l' \neq l$ and $\bar{l}' \in C'_k$, it holds that l' is universal and $l <_{\Pi'} l'$. By Lemma 1, C'_j agrees with C_j on all literals whose variables are different from the universal literals x and x'. Likewise for C'_k and C_k. Therefore, $l \in C'_j$ and $\bar{l} \in C'_k$.

Now, assume C'_j contains a literal l' such that $l' \neq l$ and $\bar{l}' \in C'_k$. If the variable of l' is different from x and x', then it must be the case that l' is universal and $l <_{\Pi'} l'$, for otherwise the derivation of C_i in P were not valid. Assume thus that the variable of l' is either x or x'. If l' is either x or x', then Lemma 1 implies that C_j contains x and also, since $\bar{l}' \in C'_k$, that C_k contains $\bar{x}$. Therefore, it holds that $l <_{\Pi'} x$ (since otherwise the derivation of C_i in P were not valid) and since x' and x are in the same quantifier block, it also holds that $l <_{\Pi'} x'$, hence $l <_{\Pi'} l'$. The case where l' is $\bar{x}$ or $\bar{x}'$ is symmetric.

Case 2: C_i has been derived from a clause C_j via a $\forall$-red step, that is, by removing a universal literal y such that for every existential literal $l' \in C_j$, it holds that $l' <_{\Pi} y$. If $var(y) \neq x$, then $C'_j = C'_i \setminus \{y\}$ and since, by Lemma 1, C'_i coincides with C_i on all existential variables, it holds for every existential literal $l' \in C'_i$ that $l' <_{\Pi'} y$. If $var(y) = x$, then C'_j is of the form $C'_i \setminus \{x, x'\}$ or $C'_i \setminus \{\bar{x}, \bar{x}'\}$. Now, x and x' are in the same quantifier block and thus, with the same argument as for $var(y) = x$, it holds for every existential literal $l' \in C'_j$ that $l' <_{\Pi'} y$.

Finally, to see that $C'_n = \bot$, observe the following: By Lemma 1, since x and $\bar{x}$ are not in C_n, it follows that x' and $\bar{x}'$ are not in C'_n. Moreover, again by Lemma 1, C_n and C'_n agree on all other literals. Therefore, $C'_n = C_n = \bot$. □

We can also show that our simulation does not introduce new LQ-res steps. Hence, if a long-distance-resolution proof contains n LQ-res steps, our simulation terminates after at most n iterations (the proof is omitted due to space reasons):

Theorem 3. *Let P' be obtained from $\phi = \Pi.\psi$ and P by our procedure. Then, P' contains fewer LQ-res steps than P.*

4.4 Clashes of Several Universal Literals

Until now, we assumed that LQ-res steps involve only one pair of complementary universal literals. When multiple such pairs are involved, the procedure changes only slightly: Instead of eliminating only a single literal from one of the clauses that are involved in the LQ-res step, we now eliminate several of them. If we start with the outermost one and gradually move inwards, we ensure that at most one blocked literal is added per clause. We illustrate this on an example. Consider the QBF $\phi = \exists a \exists b \forall x \exists c \forall y \exists d.(b \vee x \vee c \vee y \vee d) \wedge (a \vee \bar{x} \vee c) \wedge (\bar{a} \vee \bar{b} \vee \bar{y} \vee d)$ and the following derivations in a long-distance-resolution proof:

$$\dfrac{b \vee x \vee c \vee y \vee d \qquad \dfrac{a \vee \bar{x} \vee c \qquad \bar{a} \vee \bar{b} \vee \bar{y} \vee d}{\bar{b} \vee \bar{x} \vee c \vee \bar{y} \vee d}\ \text{(Q-res)}}{x \vee \bar{x} \vee c \vee y \vee \bar{y} \vee d}\ \text{(LQ-res)}$$

In the LQ-res step, there are two pairs of complementary universal literals, namely $x, \bar{x}$ and $y, \bar{y}$. We therefore try to get rid of both x and y in the left antecedent $L = b \vee x \vee c \vee y \vee d$ of the LQ-res step. As in the case where only one literal is removed, we start by deriving in QRAT all clauses that occur before the LQ-res step. In this case, we add $\bar{b} \vee \bar{x} \vee c \vee \bar{y} \vee d$ to ϕ via a Q-res step and denote the resulting formula by ϕ'.

Now we want to remove x from L via blocked-literal elimination. In order for x to be blocked in ϕ', all outer resolvents of L upon x have to be tautologies. The formula ϕ' contains two clauses that can be resolved with L upon x, namely $\bar{b} \vee \bar{x} \vee c \vee \bar{y} \vee d$ and $a \vee \bar{x} \vee c$. As the first clause contains $\bar{b}$ and L contains b, the corresponding outer resolvent upon x contains $b, \bar{b}$. But there are no complementary literals in the outer resolvent $a \vee b$ with the second clause. We therefore add a fresh literal x' to L and add its complement $\bar{x}'$ to $\bar{a} \vee \bar{x} \vee c$ to obtain $\phi' = \exists a \exists b \forall x \forall x' \exists c \forall y \exists d.(b \vee x \vee x' \vee c \vee y \vee d) \wedge (a \vee \bar{x} \vee \bar{x}' \vee c) \wedge (\bar{a} \vee \bar{b} \vee \bar{y} \vee d) \wedge (\bar{b} \vee \bar{x} \vee c \vee \bar{y} \vee d)$.

We can now remove the blocked literal x from $(b \vee x \vee x' \vee c \vee y \vee d)$ to obtain $L' = b \vee x' \vee c \vee y \vee d$. If we now resolved L' with $\bar{b} \vee \bar{x} \vee c \vee \bar{y} \vee d$, we would get the following LQ-res step:

$$\dfrac{b \vee x' \vee c \vee y \vee d \qquad \bar{b} \vee \bar{x} \vee c \vee \bar{y} \vee d}{x' \vee \bar{x} \vee c \vee y \vee \bar{y} \vee d}\ \text{(LQ-res)}$$

Since there is still a clash of y and $\bar{y}$, we need to get rid of y in L'. We are lucky because we do not need to perform any blocked-literal additions: The only clauses in ϕ' that contain $\bar{y}$ are $\bar{a} \vee \bar{b} \vee \bar{y} \vee d$ and $\bar{b} \vee \bar{x} \vee c \vee \bar{y} \vee d$, and the outer resolvents of L' with both of them contain complementary literals. We can thus remove y from L' and use a Q-res step to add the resulting resolvent to ϕ':

$$\frac{b \vee x' \vee c \vee d \qquad \bar{b} \vee \bar{x} \vee c \vee \bar{y} \vee d}{x' \vee \bar{x} \vee c \vee \bar{y} \vee d}\ \text{(Q-res)}$$

Similarly to the case where we only eliminated one literal, we then propagate the corresponding changes through the rest of the proof to turn it into a valid long-distance resolution proof of ϕ'.

5 Complexity of the Simulation

After showing how a long-distance-resolution proof can be translated into a QRAT proof, we still have to prove that the size (the number of derivation steps) of the resulting QRAT proof is polynomial w.r.t. the size of the original proof and the formula. We have seen that the long-distance-resolution proof and the QRAT proof correspond one-to-one on resolution steps and $\forall$-red steps. Therefore, we only need to estimate the number of blocked-literal addition and blocked-literal elimination steps to obtain an upper bound on the size of the QRAT proof.

Consider a long-distance-resolution proof $C_{m+1}, \ldots, C_r, \ldots, C_n$ of a QBF $\Pi.C_1 \wedge \cdots \wedge C_m$, where C_r is the first clause that is derived via an LQ-res step:

$$\frac{C \vee l \vee x_1 \vee \cdots \vee x_k \qquad D \vee \bar{l} \vee \bar{x}_1 \vee \cdots \vee \bar{x}_k}{C_r = C \vee D \vee x_1 \vee \bar{x}_1 \vee \cdots \vee x_k \vee \bar{x}_k}\ \text{(LQ-res)}$$

We can make the following observation: To remove all the literals $x_1, \ldots, x_k$ from $C \vee l \vee x_1 \vee \cdots \vee x_k$ via blocked-literal elimination, we have to add at most one new literal of the form $\bar{x}'_i$ to every clause $C_1, \ldots, C_{r-1}$ if we start by eliminating the outermost universal literal x_1 and step-by-step work ourselves towards the innermost literal x_k. The reason this works is as follows:

Assume we have added the literal x'_1 to $C \vee l \vee x_1 \vee \cdots \vee x_k$ and the corresponding literal $\bar{x}'_1$ to another clause $C_i = C'_i \vee \bar{x}_1$ to obtain complementary literals in the outer resolvent of the resulting clauses $C \vee l \vee x_1 \vee x'_1 \vee \cdots \vee x_k$ and $C' \vee \bar{x}_1 \vee \bar{x}'_1$ upon x_1. Then, the outer resolvent of $C \vee l \vee x_1 \vee x'_1 \vee \cdots \vee x_k$ with $C' \vee \bar{x}_1 \vee \bar{x}'_1$ upon a literal x_j that is inner to x_1 (i.e., $x_1 <_\Pi x_j$) contains the complementary pair $x'_1, \bar{x}'_1$, so we have to add no further literals to $C' \vee \bar{x}_1 \vee \bar{x}'_1$.

Hence, the number of blocked-literal additions for literals of the form $\bar{x}'_i$ is bounded by the number of clauses, that is, by n. Moreover, for every addition of a literal $\bar{x}'_i$ to some clause, there is at most one addition of the corresponding literal x'_i. Therefore, there are at most $2n$ blocked-literal additions per LQ-res step. Now, for every addition of a literal x'_i, there is exactly one elimination of the corresponding literal x_i. Thus, overall there are at most $3n$ blocked-literal additions and eliminations for every LQ-res step. Since the number of LQ-res steps is bounded by the number of clauses in the proof, the size of the QRAT derivation is at most $3n^2$. It follows that whenever a QBF has a long-distance-resolution proof of polynomial size, it also has a polynomial-size QRAT proof:

Theorem 4. *The* QRAT *proof system polynomially simulates the long-distance-resolution calculus.*

6 Evaluation

We now know that QRAT can polynomially simulate long-distance resolution. But what does it mean in practice? Can we have short QRAT proofs for formulas that have short long-distance-resolution proofs? To answer this question at least partly, we consider the formulas well-known for having short long-distance-resolution proofs while only having long Q-resolution proofs—the Kleine Büning formulas [20]. A Kleine Büning formula of size n, in short $KBKF_n$, has the prefix $\exists a_0, a_1, b_1 \forall x_1 \exists a_2, b_2 \forall x_2 \ldots \exists a_n, b_n \forall x_n \exists c_1, c_2, \ldots, c_n$ and the following clauses:

I :	$\bar{a}_0$	I' : $a_0 \vee \bar{a}_1 \vee \bar{b}_1$	
A_i :	$a_i \vee \bar{x}_i \vee \bar{a}_{i+1} \vee \bar{b}_{i+1}$	B_i : $b_i \vee x_i \vee \bar{a}_{i+1} \vee \bar{b}_{i+1}$	for $i \in \{1..n-1\}$
C :	$a_n \vee \bar{x}_n \vee \bar{c}_1 \vee \cdots \vee \bar{c}_n$	C' : $b_n \vee x_n \vee \bar{c}_1 \vee \cdots \vee \bar{c}_n$	
X_i :	$\bar{x}_i \vee c_i$	X'_i : $x_i \vee c_i$	for $i \in \{1..n\}$

We can reduce a formula $KBKF_n$ to a formula $KBKF_{n-1}$ by using only Q-res, blocked-literal elimination, and clause-deletion steps[4] (no $\forall$-red steps or resolution upon universal literals). To do so, we use the clauses A_n, B_n, C, C', X_n, and X'_n of $KBKF_n$ to construct the clauses C and C' of $KBKF_{n-1}$. The required 12 steps are shown below. The last two clauses (11 and 12) respectively correspond to the clauses C and C' of $KBKF_{n-1}$.

1.	$a_n \vee \bar{x}_n \vee \bar{c}_1 \vee \cdots \vee \bar{c}_{n-1}$	(Q-res of C and X_n)
2.	$b_n \vee x_n \vee \bar{c}_1 \vee \cdots \vee \bar{c}_{n-1}$	(Q-res of C' and X'_n)
3.		(delete C, C', X_n, X'_n)
4.	$a_{n-1} \vee \bar{x}_{n-1} \vee \bar{b}_n \vee \bar{x}_n \vee \bar{c}_1 \vee \cdots \vee \bar{c}_{n-1}$	(Q-res of 1 and A_{n-1})
5.	$b_{n-1} \vee x_{n-1} \vee \bar{a}_n \vee x_n \vee \bar{c}_1 \vee \cdots \vee \bar{c}_{n-1}$	(Q-res of 2 and B_{n-1})
6.	$a_{n-1} \vee \bar{x}_{n-1} \vee \bar{b}_n \vee \bar{c}_1 \vee \cdots \vee \bar{c}_{n-1}$	(BLE of $\bar{x}_n$ from 4)
7.	$b_{n-1} \vee x_{n-1} \vee \bar{a}_n \vee \bar{c}_1 \vee \cdots \vee \bar{c}_{n-1}$	(BLE of x_n from 5)
8.	$a_{n-1} \vee \bar{x}_{n-1} \vee x_n \vee \bar{c}_1 \vee \cdots \vee \bar{c}_{n-1}$	(Q-res of 6 and B_{n-1})
9.	$b_{n-1} \vee x_{n-1} \vee \bar{x}_n \vee \bar{c}_1 \vee \cdots \vee \bar{c}_{n-1}$	(Q-res of 7 and A_{n-1})
10.		(delete 4, 5, 6, 7, A_{n-1}, B_{n-1})
11.	$a_{n-1} \vee \bar{x}_{n-1} \vee \bar{c}_1 \vee \cdots \vee \bar{c}_{n-1}$	(BLE of x_n from 8)
12.	$b_{n-1} \vee x_{n-1} \vee \bar{c}_1 \vee \cdots \vee \bar{c}_{n-1}$	(BLE of $\bar{x}_n$ from 9)

Table 1 shows the sizes of the Kleine Büning formulas as well as of the corresponding long-distance-resolution proofs (in the QRP format) and QRAT proofs. The latter are obtained by the construction mentioned in this section. The size of both types of proofs is linear in the size of the formula. Although QRAT proofs use about twice as many proof steps (including deletion steps), the file size of QRAT proofs is smaller. The explanation for this is that long-distance-resolution proofs increase the length of clauses, while QRAT proofs decreases their length.

Short proofs of the $KBKF$ formulas can also be obtained by using resolution upon universal variables as in the calculus QU-Res [24]. There is, however, a variant of the $KBKF$ formulas, called $KBKF_n{-}qu$ [2], which has only exponential proofs in the QU-Res calculus. A $KBKF_n{-}qu$ formula is obtained from $KBKF_n$

[4] Clause deletion was not used in the simulation, but is allowed in the QRAT system.

Table 1. The size of Kleine Büning formulas in the number of variables (#var) and clauses (#cls). Additionally, the size of their long-distance-resolution proofs (in the QRP format) in the number of Q-res steps (#Q), LQ-res steps (#L), ∀-red steps (#∀), and the file size in KB (ignoring the part that represents the formula). On the right, the number of Q-res (#Q), BLE (#B), and deletion (#D) steps as well as the file size for the manual QRAT proofs.

	input		LD proofs (QRP)				QRAT proofs			
formula	#var	#cls	#Q	#L	#∀	file size	#Q	#B	#D	file size
$KBKF_{10}$	41	42	41	18	38	6	57	38	92	6
$KBKF_{50}$	201	202	201	98	198	138	297	198	492	112
$KBKF_{100}$	401	402	401	198	398	573	597	398	992	421
$KBKF_{200}$	801	802	801	398	798	2321	1197	798	1992	1627
$KBKF_{500}$	2001	2002	2001	998	1998	16 259	2997	1998	4992	11 890

by adding a universal literal y_i (occurring in the same quantifier block as x_i) to every clause in $KBKF_n$ that contains x_i, and a literal $\bar{y}_i$ to every clause in $KBKF_n$. For these formulas, blocked-literal elimination can remove all the y_i and $\bar{y}_i$ literals, which reduces a $KBKF_n{-}qu$ formula to a $KBKF_n$ formula that can then be efficiently proved using resolution upon universal literals.

In addition to the handcrafted QRAT proofs, we implemented a tool (called `ld2qrat`) that, based on our simulation, transforms long-distance-resolution proofs into QRAT proofs. We used `ld2qrat` to transform the long-distance-resolution proofs of the $KBKF_n$ formulas (by Egly et al. [11]) into QRAT proofs and validated the correctness of these proofs with the proof checker `QRAT-trim`. In the plain mode, `ld2qrat` closely follows our simulation. Additionally, it features two optimizations: (1) Given an LQ-res step upon l with the antecedents $C \vee l \vee x$ and $D \vee \bar{l} \vee \bar{x}$, if one of x or $\bar{x}$ is already a blocked literal, it is removed with blocked-literal elimination. This avoids the introduction of new variables. (2) Clauses are deleted as soon as they are not needed later in the proof anymore.

Table 2 shows properties of the QRAT proofs produced by `ld2qrat` from the long-distance-resolution proofs of the $KBKF$ formulas. On the left are the sizes of proofs obtained without the clause-deletion optimization. On the right are the sizes of proofs with this optimization. A (least squares) regression analysis confirms that the length (number of steps) of the QRAT proofs without deletion is quadratically related to the length of the corresponding long-distance-resolution proofs: The function $f(x) = 0.22x^2 - 4.48x + 54.58$ (where x is the length of the long-distance-resolution proof and $f(x)$ is the length of the QRAT proof) fits the data from the above tables perfectly (the error term R^2 of the regression is 1).

7 QRAT in the Complexity Landscape

After the analysis of QRAT in theory and practice, we now locate it in the proof-complexity landscape of resolution-based calculi for QBF, which is shown in Fig. 2. Besides the long-distance-resolution calculus LQ-Res, another well-known proof system is the calculus QU-Res [24], which extends the basic Q-resolution

Table 2. Comparison of the QRAT proofs obtained by applying `ld2qrat` to long-distance-resolution proofs (in the QRP format) of the Kleine Büning formulas. The file size is given in KB and the time for translating the proof (time) is given in seconds.

	QRP to QRAT w/o deletion				QRP to QRAT w/ deletion			
formula	#var	#step	file size	time	#var	#step	file size	time
$KBKF_{10}$	59	1690	103	0.07	59	448	26	0.01
$KBKF_{50}$	299	52 170	18 774	0.45	299	6288	2227	0.12
$KBKF_{100}$	599	214 270	154 299	3.77	599	22 588	16 192	0.86
$KBKF_{200}$	1199	868 470	1 309 559	30.70	1199	85 188	126 375	7.95
$KBKF_{500}$	2999	5 471 070	23 622 369	497.32	2999	512 988	2 229 195	124.10

calculus (Q-Res) by allowing resolution upon universals literals if the resulting resolvent does not contain complementary literals. As QRAT also allows resolution upon universal literals, it simulates QU-Res. Balabanov et al. [2] showed the incomparability between LQ-Res and QU-Res by exponential separations. It thus follows that QRAT is strictly stronger than both LQ-Res and QU-Res.

Another system that is stronger than both LQ-Res and QU-Res is the calculus LQU$^+$-Res [2], which extends LQ-Res by allowing (long-distance) resolution upon universals literals. We know that either QRAT is strictly stronger than LQU$^+$-Res or the two systems are incomparable: On purely existentially-quantified formulas, LQU$^+$-Res boils down to ordinary propositional resolution (without complementary literals in resolvents) whereas the QRAT system boils down to the RAT system [25]. As the RAT system is strictly stronger than resolution—there exist polynomial-size RAT proofs of the well-known pigeon hole formulas [13] while resolution proofs of these formulas are necessarily exponential in size [12]—LQU$^+$-Res cannot simulate QRAT.

On the other hand, QRAT might be able to simulate LQU$^+$-Res, but not with our simulation of the long-distance-resolution calculus, because the simulation cannot convert all LQU$^+$-Res proofs into QRAT proofs. To see this, consider the QBF $\exists a \forall x \forall y \exists b.(a \vee x \vee b) \wedge (\bar{a} \vee \bar{x} \vee b) \wedge (x \vee \bar{b}) \wedge (\bar{x} \vee \bar{y} \vee \bar{b})$ with the following LQU$^+$-Res proof [2]: $x \vee \bar{x} \vee b,\ \bar{y} \vee \bar{b},\ x \vee \bar{x} \vee \bar{y},\ x \vee \bar{x},\ x,\ \bot$. The proof can be illustrated as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{a \vee x \vee b \qquad \bar{a} \vee \bar{x} \vee b}{x \vee \bar{x} \vee b}\,\text{(LQ-res)} \qquad \dfrac{x \vee \bar{b} \qquad \bar{x} \vee \bar{y} \vee \bar{b}}{\bar{y} \vee \bar{b}}\,\text{(QU-res)}}{x \vee \bar{x} \vee \bar{y}}\,\text{(Q-res)}}{x \vee \bar{x}}\,(\forall\text{-red})}{x}\,(\forall\text{-red})}{\bot}\,(\forall\text{-red})$$

In our simulation, we first replace the literal x in $a \vee x \vee b$ by x' before resolving the resulting clause $a \vee x' \vee b$ with $\bar{a} \vee \bar{x} \vee b$. The replacement of x by x' also leads to the addition of $\bar{x}'$ to $\bar{x} \vee \bar{y} \vee \bar{b}$. If we now perform the universal resolution step of $x \vee \bar{b}$ with $\bar{x} \vee \bar{x}' \vee \bar{y} \vee \bar{b}$, then we obtain the following partial proof:

$$\dfrac{\dfrac{a \vee x' \vee b \qquad \bar{a} \vee \bar{x} \vee b}{x' \vee \bar{x} \vee b}\,\text{(Q-res)} \qquad \dfrac{x \vee \bar{b} \qquad \bar{x} \vee \bar{x}' \vee \bar{y} \vee \bar{b}}{\bar{x}' \vee \bar{y} \vee \bar{b}}\,\text{(QU-res)}}{}$$

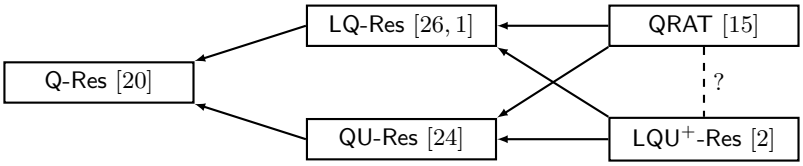


Fig. 2. Complexity landscape including QRAT. A directed edge from a proof system A to a proof system B means that A is strictly stronger than B.

The Q-res step upon b is now impossible because x' is in $x' \vee \bar{x} \vee b$ and $\bar{x}'$ is in $\bar{x}' \vee \bar{y} \vee \bar{b}$. We also cannot eliminate x' from $x' \vee \bar{x} \vee b$ via blocked-literal elimination: This would require us to add a new literal x'' to $x' \vee \bar{x} \vee b$ and to add $\bar{x}''$ to $\bar{x}' \vee \bar{y} \vee \bar{b}$ leading to the new pair $x'', \bar{x}''$ of complementary literals.

Our key result, Lemma 1, does not hold anymore when allowing resolution over universal literals. Lemma 1 guarantees that whenever a new literal $\bar{x}'$ is in a proof clause C_i' of the modified long-distance-resolution proof, then $\bar{x}$ was contained in the corresponding clause C_i in the original proof. The above example shows that resolution over universal literals destroys this property: Although $\bar{x}'$ is contained in the clause $\bar{x}' \vee \bar{y} \vee \bar{b}$, the literal x is not contained in the corresponding clause $y \vee \bar{y} \vee b$ of the original proof because we resolved it away.

8 Conclusion

We showed that the QRAT proof system polynomially simulates long-distance resolution. In our simulation, we used only a small subset of the QRAT rules: Q-resolution, universal reduction, blocked-literal addition, and blocked-literal elimination. Based on our simulation, we implemented a tool that transforms long-distance-resolution proofs into QRAT proofs. The tool allows to merge a QRAT derivation produced by a QBF-preprocessor with a long-distance-resolution proof produced by a search-based solver. The correctness of the resulting QRAT proof can then be checked with a proof checker such as `QRAT-trim` [15]. We evaluated the tool on long-distance-resolution proofs of the well-known Kleine Büning formulas and manually constructed QRAT proofs of these formulas that are smaller than their long-distance counterparts.

We further noted that our simulation breaks down if the long-distance-resolution calculus is extended by resolution upon universal literals, as in the calculus LQU$^+$-Res. Investigating the exact relationship between LQU$^+$-Res and QRAT therefore remains open for future work. Another open question is whether blocked-literal elimination can be polynomially simulated in LQU$^+$-Res. We also do not know whether it is possible to simulate long-distance resolution with only Q-resolution, universal reduction, clause deletion, and blocked-literal elimination (but no blocked-literal addition). Finally, what is still unclear is how QRAT relates to instantiation-based proof systems and sequent proof systems. Answers to these questions will shed more light on the proof-complexity landscape of QBF.

References

1. Balabanov, V., Jiang, J.R.: Unified QBF certification and its applications. Formal Methods in System Design 41(1), 45–65 (2012)
2. Balabanov, V., Widl, M., Jiang, J.R.: QBF resolution systems and their proof complexities. In: Proc. of the 17th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2014). LNCS, vol. 8561, pp. 154–169. Springer, Cham (2014)
3. Benedetti, M., Mangassarian, H.: QBF-based formal verification: Experience and perspectives. Journal on Satisfiability, Boolean Modeling and Computation (JSAT) 5(1-4), 133–191 (2008)
4. Beyersdorff, O., Bonacina, I., Chew, L.: Lower bounds: From circuits to QBF proof systems. In: Proc. of the 2016 ACM Conference on Innovations in Theoretical Computer Science (ITCS 2016). pp. 249–260. ACM (2016)
5. Beyersdorff, O., Chew, L., Janota, M.: On unification of QBF resolution-based calculi. In: Proc. of the 39th Int. Symposium on Mathematical Foundations of Computer Science (MFCS 2014). LNCS, vol. 8635, pp. 81–93. Springer, Heidelberg (2014)
6. Beyersdorff, O., Chew, L., Janota, M.: Proof complexity of resolution-based QBF calculi. In: Proc. of the 32nd Int. Symposium on Theoretical Aspects of Computer Science (STACS 2015). LIPIcs, vol. 30, pp. 76–89. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2015)
7. Beyersdorff, O., Chew, L., Mahajan, M., Shukla, A.: Are short proofs narrow? QBF resolution is not simple. In: Proc. of the 33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016). LIPIcs, vol. 47, pp. 15:1–15:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016)
8. Beyersdorff, O., Pich, J.: Understanding Gentzen and Frege Systems for QBF. In: Proc. of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2016). pp. 146–155. ACM (2016)
9. Chen, H.: Proof Complexity Modulo the Polynomial Hierarchy: Understanding Alternation as a Source of Hardness. In: Proc. of the 43rd Int. Colloquium on Automata, Languages, and Programming (ICALP 2016). LIPIcs, vol. 55, pp. 94:1–94:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016)
10. Egly, U.: On stronger calculi for QBFs. In: Proc. of the 19th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2016). LNCS, vol. 9710, pp. 419–434. Springer, Cham (2016)
11. Egly, U., Lonsing, F., Widl, M.: Long-distance resolution: Proof generation and strategy extraction in search-based QBF solving. In: Proc. of the 19th Int. Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-19). LNCS, vol. 8312, pp. 291–308. Springer, Heidelberg (2013)
12. Haken, A.: The intractability of resolution. Theoretical Computer Science 39, 297–308 (1985)
13. Heule, M.J.H., Hunt Jr., W.A., Wetzler, N.D.: Expressing symmetry breaking in DRAT proofs. In: Proc. of the 25th Int. Conference on Automated Deduction (CADE 2015). LNCS, vol. 9195, pp. 591–606. Springer, Cham (2015)
14. Heule, M.J.H., Seidl, M., Biere, A.: Blocked literals are universal. In: Proc. of the 7th Int. NASA Symposium on Formal Methods (NFM 2015). LNCS, vol. 9058, pp. 436–442. Springer, Cham (2015)
15. Heule, M.J.H., Seidl, M., Biere, A.: Solution validation and extraction for QBF preprocessing. Journal of Automated Reasoning pp. 1–29 (2016)

16. Janota, M.: On Q-Resolution and CDCL QBF solving. In: Proc. of the 19th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2016). LNCS, vol. 9710, pp. 402–418. Springer, Cham (2016)
17. Janota, M., Grigore, R., Marques-Silva, J.: On QBF proofs and preprocessing. In: Proc. of the 19th Int. Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-19). LNCS, vol. 8312, pp. 473–489. Springer, Heidelberg (2013)
18. Janota, M., Klieber, W., Marques-Silva, J., Clarke, E.M.: Solving QBF with counterexample guided refinement. Artificial Intelligence 234, 1–25 (2016)
19. Kleine Büning, H., Bubeck, U.: Theory of Quantified Boolean Formulas. In: Handbook of Satisfiability, pp. 735–760. IOS Press (2009)
20. Kleine Büning, H., Karpinski, M., Flögel, A.: Resolution for quantified boolean formulas. Information and Computation 117(1), 12–18 (1995)
21. Kullmann, O.: On a generalization of extended resolution. Discrete Applied Mathematics 96-97, 149–176 (1999)
22. Lonsing, F., Egly, U.: DepQBF 6.0: A search-based QBF solver beyond traditional QCDCL. CoRR abs/1702.08256 (2017)
23. Slivovsky, F., Szeider, S.: Variable dependencies and Q-resolution. In: Proc. of the 17th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2014). LNCS, vol. 8561, pp. 269–284. Springer, Cham (2014)
24. Van Gelder, A.: Contributions to the theory of practical quantified boolean formula solving. In: Proc. of the 18th Int. Conference on Principles and Practice of Constraint Programming (CP 2012). LNCS, vol. 7514, pp. 647–663. Springer, Heidelberg (2012)
25. Wetzler, N., Heule, M.J.H., Hunt Jr., W.A.: DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In: Proc. of the 17th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2014). LNCS, vol. 8561, pp. 422–429. Springer, Cham (2014)
26. Zhang, L., Malik, S.: Conflict driven learning in a quantified boolean satisfiability solver. In: Proc. of the 2002 IEEE/ACM Int. Conference on Computer-aided Design (ICCAD 2002). pp. 442–449. ACM/IEEE Computer Society (2002)