Journal of Automated Reasoning manuscript No.
(will be inserted by the editor)

Solution Validation and Extraction for QBF Preprocessing

Marijn Heule · Martina Seidl · Armin Biere

Received: date / Accepted: date

Abstract In the context of reasoning on quantified Boolean formulas (QBFs), the extension of propositional logic with existential and universal quantifiers, it is beneficial to use *preprocessing* for solving QBF encodings of real-world problems. Preprocessing applies rewriting techniques that preserve (satisfiability) equivalence and that do not destroy the formula's CNF structure. In many cases, preprocessing either solves a formula directly or modifies it such that information helpful to solvers becomes better visible and irrelevant information is hidden. The application of a preprocessor, however, prevented the extraction of proofs for the original formula in the past. Such proofs are required to independently validate the correctness of the preprocessor's rewritings and the solver's result as well as for the extraction of solutions in terms of Skolem functions. In particular for the important technique of universal expansion efficient proof checking and solution generation was not possible so far.

This article presents a unified proof system with three simple rules based on *quantified resolution asymmetric tautology* (QRAT). In combination with an extended version of universal reduction, we use this proof system to efficiently express *current* preprocessing techniques including universal expansion. Further, we develop an approach for extracting Skolem functions. We equip the preprocessor bloqqer with QRAT proof logging and provide a proof checker for QRAT proofs which is also able to extract Skolem functions.

This work was supported by the Austrian Science Fund (FWF) through the national research network RiSE (S11408-N23), Vienna Science and Technology Fund (WWTF) under grant ICT10-018, AFRL Award FA8750-15-2-0096, and the National Science Foundation under grant number CCF-1618574.

Marijn J.H. Heule
Department of Computer Science, The University of Texas at Austin, GDC building, room 7.714, Austin, USA
E-mail: marijn@cs.utexas.edu

Martina Seidl · Armin Biere
Institute for Formal Models and Verification, Johannes Kepler University Linz, Altenbergerstr. 69, 4040 Linz, Austria

1 Introduction

The upper part of Figure 1 shows the typical workflow for solving *quantified Boolean formulas*. Given a formula in conjunctive normal form (CNF), it is first passed to a *preprocessor* which rewrites the formula in a satisfiability preserving manner. Such a preprocessor typically implements a rich portfolio of sophisticated simplification techniques. Sometimes the preprocessor is able to solve the formula directly, otherwise it has to be passed to a complete solver.

Effectively checking the result returned by a QBF solver has been an open challenge for a long time [5, 27, 28, 32, 34, 41, 45]. The previous state-of-the-art was to simply dump Q-resolution proofs [33] and to validate their structure. This approach has two major problems. On the one hand the proofs might get extremely large and can often not be produced due to technical limitations. On the other hand, important solving and preprocessing techniques are based on calculi other than Q-resolution, which can not easily be simulated by Q-resolution if at all.

Due to the diversity of techniques in state-of-the-art preprocessors [10, 15], it is not straightforward to provide a checker which verifies the output of the preprocessor. In fact, it would be preferable to translate the different preprocessing techniques to a canonical representation which then can be checked easily. Some efforts in this direction use Q-resolution [28, 45]. If a Q-resolution proof is available, then checking is polynomial w.r.t. proof size. However, the proof itself might become exponentially large w.r.t. solving complexity and already writing down the proof is too costly.

In propositional logic, the RUP proof checking format [16] is extremely successful because it simply logs the learnt clauses and provides an easy checking criterion. For optimization purposes, recently, the DRUP extension has been presented [25] which provides elimination criteria for redundant clauses. It has been recognized that

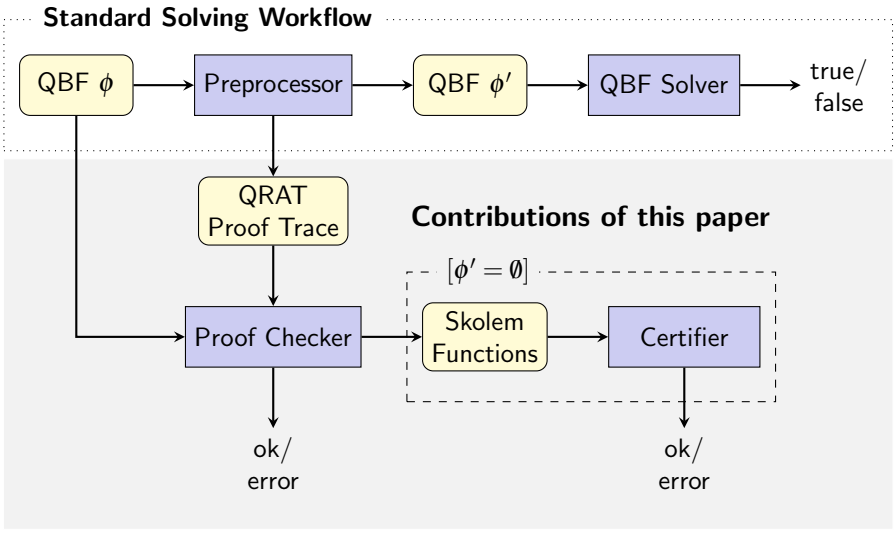

Fig. 1 Overview of extended QBF solving workflow.

RUP and DRUP can be characterized with the *resolution asymmetric tautology property* (RAT) which has been originally developed in the context of propositional preprocessing for characterizing and comparing the strength of the various techniques. In this article, following [19], we extend RAT [24] to QRAT, the *quantified resolution asymmetric tautology property*, and introduce novel clause addition and elimination techniques using QRAT. With these novel rules, a new powerful proof system is obtained, which is able to capture all state-of-the-art preprocessing techniques in a uniform and compact manner. This allows us to develop a checker verifying the correctness of a QBF preprocessor. Moreover, checking QRAT proofs is polynomial in the proof size. We integrated QRAT-based tracing in our preprocessor bloqqer [10] and implemented an efficient checker for QRAT proofs which is further able to extract Skolem functions from a proof of satisfiability.

In many applications, such as QBF-based synthesis [7, 11], a (checkable) proof is not enough. Users also need certificates in form of Skolem functions. Depending on the domain these Skolem functions correspond to synthesized circuits, universal plans in non-deterministic planning, game strategies etc. While it is also possible but cumbersome to directly generate certificates during solving [6], a more common approach is to extract them from a proof generated by a QBF solver. In this article, we follow the latter approach, where proofs provide an interface between solver and certificate extractor. This makes the extractor independent from the actual solver. Further, proofs can be checked independently and they might be useful for other applications such as core extraction, interpolation, etc.

The seminal results of [2, 3] showed how to extract Skolem functions from Q-resolution proofs as produced by search-based solvers (e.g., [37, 40, 47]) and [4] extends the approach to long-distance Q-resolution. An implementation of this approach has been presented in [41]. However, when preprocessing techniques such as universal expansion [8], are used, then no certificates could be obtained. We closed this gap by showing how to extract certificates from our QRAT proof system.

To summarize, this article presents a novel proof system which is the basis for the first coherent presentation of a complete workflow w.r.t. proof generation, proof checking, certificate extraction, as well as certificate checking for the arbitrary combination of state-of-the-art QBF preprocessing techniques (see lower part of Fig. 1).

Parts of this article have been presented at the 7th Int. Joint Conference on Automated Reasoning (IJCAR 2014) [19] and at the 14th Int. Conference on Formal Methods in Computer-Aided Design (FMCAD 2014) [18]. Some of the definitions are generalized and proofs have been streamlined accordingly. In addition, we also consider the preprocessing technique *blocked literal elimination* (BLE) which we presented at the 7th Int. NASA Formal Methods Symposium (NFM 2015) [22] in our workflow and we show experimental results with BLE enabled.

2 Preliminaries

The QBFs considered in this article are in *prenex conjunctive normal form* (PCNF). A QBF ϕ in PCNF has the structure $\Pi.\psi$ consisting of the *quantifier prefix* Π and the *matrix* ψ. The quantifier prefix Π has the form $Q_1X_1Q_2X_2\ldots Q_nX_n$ with disjoint

variable sets X_i, $Q_i \in \{\forall,\exists\}$, $Q_i \neq Q_{i+1}$, and the set of variables of ϕ defined as $\mathsf{vars}(\phi) = X_1 \cup \ldots \cup X_n$. The matrix ψ is a propositional formula in conjunctive normal form (CNF), i.e., a conjunction of clauses. A clause is a disjunction of literals and a literal is either a variable (e.g., x) or a negated variable (e.g., $\bar{x}$). The variable of a literal is denoted by $\mathsf{var}(l)$ where $\mathsf{var}(l) = x$ if $l = x$ or $l = \bar{x}$. In the first case, we call l a *positive literal*, in the second case, we call l a *negative literal*. The negation of a literal l is denoted by $\bar{l}$. As usual, $\bar{\bar{l}} = l$. The quantifier $\mathsf{Q}(\Pi, l)$ of a literal l is Q_i if $\mathsf{var}(l) \in X_i$. Let $\mathsf{Q}(\Pi, l) = Q_i$ and $\mathsf{Q}(\Pi, k) = Q_j$, then $l \leq_\Pi k$ if $i \leq j$. Note, that this order is in general not a partial order, since $l \leq_\Pi k$ as well as $k \leq_\Pi l$ for all literals l and k in the same scope, say X_i, even if l and k are different.

By $\top$ (empty matrix) and $\bot$ (empty clause) we denote the truth constants true and false. The polarity $\mathsf{pol}(l)$ of literal l with $\mathsf{var}(l) = x$ is $\top$ iff $l = x$ and $\bot$ iff $l = \bar{x}$. We sometimes write formulas in CNF as sets of clauses and clauses as sets of literals. We consider only closed QBFs, so ψ contains only variables which occur in the prefix. The subformula ψ_l consisting of all clauses of matrix ψ containing literal l is defined by $\psi_l = \{C \mid l \in C, C \in \psi\}$. A variable assignment $\tau : \mathsf{vars}(\phi) \to \{\bot, \top\}$ maps the variables of ϕ to truth constants. By $\tau(\phi)$ we denote ϕ under τ, i.e., each variable x is replaced by $\tau(x)$ and the assigned variables are removed from the prefix. A partial variable assignment maps only a subset of $\mathsf{vars}(\phi)$ to truth constants. A QBF $\forall x \Pi.\psi$ is false iff $\Pi.\psi[x/\top]$ or $\Pi.\psi[x/\bot]$ is false where $\Pi.\psi[x/t]$ is the QBF obtained by replacing all occurrences of variable x by t. Respectively, a QBF $\exists x \Pi.\psi$ is false iff both $\Pi.\psi[x/\top]$ and $\Pi.\psi[x/\bot]$ are false. If the matrix ψ of a QBF ϕ contains the empty clause after eliminating the truth constants according to standard rules, then ϕ is false. Accordingly, if the matrix ψ of QBF ϕ is empty, then ϕ is true.

Models and countermodels of QBFs can either be described intensionally in form of Herbrand and Skolem functions [5] or extensionally in form of subtrees of assignment trees. An *assignment tree* of a QBF ϕ is a complete binary tree of depth $|\mathsf{vars}(\phi)| + 1$ where non-leaf nodes of each level are *associated* with a variable of ϕ. The order of the associated variables in the tree respects the order of the variables in the prefix of ϕ. A non-leaf node associated with variable x has one outgoing edge *labelled* with x and one outgoing edge *labelled* with $\bar{x}$. Each path starting from the root of the tree represents a (partial) variable assignment. We also write a path as a sequence of literals. A path τ from the root node to a leaf is a complete assignment and the leaf is labelled with the value of the QBF under τ. Nodes associated with existential variables act as OR-nodes, while universal nodes act as AND-nodes. Respectively, a node is labelled either with $\top$ or with $\bot$. A QBF is true (satisfiable) iff its root is labelled with $\top$. A QBF is false (unsatisfiable) iff its root is labelled with $\bot$. A crucial notation in our work is the following.

Definition 1 (Prefix/Suffix Assignment) Let τ be a complete assignment, then τ^x and τ_x denote the partial *prefix* and *suffix* assignment w.r.t. to variable x obtained from the complete assignment τ as follows:

$$\tau^x = \{y \mapsto \tau(y) \mid y \leq_\Pi x, y \neq x\}, \qquad \tau_x = \{y \mapsto \tau(y) \mid y \not\leq_\Pi x\}$$

We also write $\tau = \tau^x l \tau_x$ with $\mathsf{var}(l) = x$ and $\tau(l) = \top$ in this case.

The corresponding non-explicit definitions in our previous work, e.g., [19] could be misinterpreted to assign literals k in the suffix τ_x, as opposed to assign them in prefix τ^x, even if they are in the same scope as l.

Example 2 Consider the QBF $\forall a \exists b,c \forall d \exists e.(a \vee b \vee \bar{c} \vee \bar{d} \vee e)$ and the full assignment $\tau = a\bar{b}c\bar{d}e$, then, e.g., $\tau(a) = \top$ and $\tau(b) = \bot$. Then we have partial assignments $\tau^b = ac$ and $\tau_b = \bar{d}e$, as well as $\tau^c = a\bar{b}$ and $\tau_c = \bar{d}e$, with $\tau = \tau^b\bar{b}\tau_b$ and $\tau = \tau^c c\tau_c$.

A *pre-model* M of QBF ϕ is a subtree of the assignment tree of ϕ such that (1) for each universal node in M, both children are in M; (2) for each existential node in M, exactly one of the children is in M; and (3) the root of the assignment tree is in M. A pre-model M of QBF ϕ is a *model* of ϕ if in addition each node in M is labelled with $\top$. Obviously, only a true QBF can have a model. A false QBF has at least one countermodel, which is defined dually: in a *pre-countermodel* M existential nodes have two children, whereas universal nodes have only one and the root of the assignment tree is in M. A pre-countermodel M is a *countermodel* if each node is labelled with $\bot$.

Two QBFs are *satisfiability equivalent* (written as $\phi_1 \sim \phi_2$) iff they have the same truth value. Two QBFs ϕ_1 and ϕ_2 are *logically equivalent* (written as $\phi_1 \approx \phi_2$) if they have the same set of (counter) models.

3 Quantified Implied Outer Resolvents

The following notion reduces a clause to those literals which are not inner to a given literal, i.e., outer or in the same scope but different. Thus it captures the "context" of the given literal w.r.t. that clause, which is then used to define a context-sensitive variant of resolvent.

Definition 3 (Outer Clause) The *outer clause* of clause C on literal $l \in C$ w.r.t. prefix Π is defined as $\mathsf{OC}(\Pi, C, l) = \{k \mid k \in C, k \leq_\Pi l, k \neq l\}$.

Definition 4 (Outer Resolvent) Let C be a clause with $l \in C$ and D a clause occurring in QBF $\Pi.\psi$ with $\bar{l} \in D$. The *outer resolvent* of C with D on literal l w.r.t. Π, denoted by $\mathsf{OR}(\Pi, C, D, l)$, is the clause $(C \setminus \{l\}) \cup \mathsf{OC}(\Pi, D, \bar{l})$.

Our considered proof system uses rules which can be interpreted as applications of instances of the following generic clause redundancy property.

Definition 5 (Quantified Implied Outer Resolvents) Clause C has property (QIOR) w.r.t. QBF $\Pi.\psi$ on literal $l \in C$ iff $\psi \models \mathsf{OR}(\Pi, C, D, l)$ for all $D \in \psi$ with $\bar{l} \in D$.

Note, that $\psi \models R$ is here and in the following interpreted in the classical propositional way, e.g., all assignments satisfying the precondition ψ also satisfy the conclusion R. Further, applied to formulas with a pure existential prefix, e.g., in the propositional case (SAT), the outer resolvent is the same as the (propositional) resolvent, which contains all literals of the resolved clauses except the pivot literals. In previous work, we considered only an instance (QRAT) of this generalization, which we will discuss further down in Section 4.

Lemma 6 *Given a clause C with* QIOR *w.r.t. QBF $\Pi.\psi$ on literal $l \in C$ with* $\mathsf{var}(l) = x$*. If there is an assignment τ that falsifies $C \setminus \{l\}$, but satisfies ψ, then the assignment τ^x satisfies all $D \in \psi$ with $\bar{l} \in D$.*

Proof Let $D \in \psi$ be a clause with $\bar{l} \in D$ and $R = \mathsf{OR}(\Pi, C, D, l)$. The definition of QIOR gives $\psi \models R$, and thus $\tau(R) = \tau((C \setminus \{l\}) \cup \mathsf{OC}(\Pi, D, \bar{l})) = \top$, because τ satisfies ψ. But since $\tau(C \setminus \{l\}) = \bot$ we need to have $\tau(\mathsf{OC}(\Pi, D, \bar{l})) = \top$. Notice that $\mathsf{OC}(\Pi, D, \bar{l})$ only consists of literals k different from l either occurring outside of x in Π or in the same scope, e.g., $k \leq_\Pi l$ and $k \neq \bar{l}$, and we obtain $\tau^x(\mathsf{OC}(\Pi, D, \bar{l})) = \top$. Finally, $\tau^x(D) = \top$ follows from $\mathsf{OC}(\Pi, D, \bar{l}) \subseteq D$. □

Theorem 7 *Given a QBF $\phi = \Pi.\psi$ and a clause $C \in \psi$ with* QIOR *on an existential literal $l \in C$ with respect to QBF $\phi' = \Pi'.\psi'$ where $\psi' = \psi \setminus \{C\}$ and Π' is Π without the variables of C not occurring in ψ'. Then ϕ and ϕ' are satisfiability equivalent.*

Proof If ϕ is satisfiable then ϕ' is also satisfiable, since all models of ϕ are also models of ϕ'. In the following, we show that if ϕ' is satisfiable then ϕ is also satisfiable. Let M' be a model for ϕ'. Let $M = M'$ except if violating assignments $\tau = \tau^x \bar{l} \tau_x$ with $\mathsf{var}(()l) = x$ in M' exist which satisfy $\psi' = \psi \setminus \{C\}$ but falsify C. In this case, we use $\tau^x l$ instead of $\tau^x \bar{l}$. Notice that such a change only modifies the polarity of a single label of an edge in M, if viewed as a tree, but changes potentially multiple assignments of the form $\tau^x \bar{l} \rho_x$. Each of those assignments now becomes $\tau^x l \rho_x$. This change only affects the clauses $D \in \psi$ with $\bar{l} \in D$. Since τ^x satisfies all these D (Lemma 6), the replacement does not affect satisfiability of any of them. On the other hand, $\tau^x l$ satisfies C. Thus the resulting pre-model M turns out to be a model of ϕ. □

The intuition behind QIOR is as follows: consider all potential outer resolvents of a clause on a certain literal with resolution candidates containing the negation of the picked literal. If all of them are "redundant", or more precisely logically implied, then this clause is redundant too and can be added or removed.

Theorem 8 *Given QBF $\phi_0 = \Pi.\psi$ and $\phi = \Pi.\psi \cup \{C\}$ where C has* QIOR *on a universal literal $l \in C$ with respect to ϕ_0. Further, let $\phi' = \Pi.\psi \cup \{C'\}$ with $C' = C \setminus \{l\}$. Then ϕ and ϕ' are satisfiability equivalent.*

Proof If ϕ' is satisfiable then ϕ is also satisfiable, since all models of ϕ' are also models of ϕ. In the following, we show that if ϕ is satisfiable then ϕ' is also satisfiable. Let M be a model for ϕ. Let $M' = M$ except if violating assignments τ in M exist which falsify C'. These τ have assignment prefix $\tau^x l$. Now consider the assignments of the form $\tau^x \bar{l} \rho_x$ in M. We will show below that the $\bar{l}$ in these assignments is redundant. Since $C' = C \setminus \{l\}$ is falsified by τ, all clauses $D \in \psi$ with $\bar{l} \in D$ are satisfied by τ^x (Lemma 6). Therefore, the *partial* assignment $\tau^x \rho_x$ satisfies $\psi \cup \{C\}$ (because $\tau^x \bar{l} \rho_x$ in M) and thus $\psi \cup \{C'\}$ (because C and C' only differ in l). We use this observation to construct M' such that is uses both $\tau^x l \rho_x$ and $\tau^x \bar{l} \rho_x$ for each $\tau^x \bar{l} \rho_x$ in M iff τ is a violating assignment. In the tree-view of a model, this replaces the labels in the subtree under $\tau^x l$ by a copy of the labels of the subtree under $\tau^x \bar{l}$. The resulting pre-model M' turns out to be a model of ϕ'. □

4 Quantified Resolution Asymmetric Tautologies (QRAT)

Checking whether an outer resolvent is implied by a formula, as stated in Def. 5 of QIOR is co-NP hard. This is too costly in practice. In order to remain polynomially computable, we use the following restriction, which still allows us to express all the interesting preprocessing techniques we are aware of.

Denote with $\psi \vdash_1 C$ that unit propagation shows that C is implied by ψ, where unit propagation is defined in the standard way. This is equivalent to and in practice computed by $\psi \wedge \overline{C} \vdash_1 \bot$, where $\overline{C}$ is the conjunction of the negation of all literals in C. Or in other words, C is implied by ψ via unit propagation iff unit propagation on the conjunction of ψ and all negated literals in C can derive the empty clause. A clause implied by unit propagation is also known as an *Asymmetric Tautology* (AT) [23, 20].

Definition 9 (Asymmetric Tautology) Clause C is an AT w.r.t. to ψ iff $\psi \vdash_1 C$.

Notice that the prefix of a QBF is ignored in the above definition and thus even for QBF, the notion of AT is purely propositional. It does not change models.

The considered restricted variant of QIOR replaces semantic implication ($\models$) by unit propagation ($\vdash_1$). It is also known as *Quantified Resolution Asymmetric Tautology* (QRAT) and defined below.

Definition 10 (Quantified Resolution Asymmetric Tautology) Clause C has QRAT on literal $l \in C$ w.r.t. QBF $\Pi.\psi$ iff $\psi \vdash_1 \mathsf{OR}(\Pi, C, D, l)$ for all $D \in \psi$ with $\bar{l} \in D$.

Note, that this definition differs from the original version [19] although both variants are equivalent. To improve readability, our new version uses unit propagation, instead of the less common notion of *Asymmetric Literal Addition* (ALA) used in [19].

Since QIOR implies QRAT, Theorem 7 gives the following corollary, which corresponds to Theorem 1 in [19].

Corollary 11 *Given a QBF $\phi = \Pi.\psi$ and a clause $C \in \psi$ with* QRAT *on an existential literal $l \in C$ with respect to QBF $\phi' = \Pi'.\psi'$ where $\psi' = \psi \setminus \{C\}$ and Π' is Π without the variables of C not occurring in ψ'. Then ϕ and ϕ' are satisfiability equivalent.*

The elimination of a clause which has QRAT w.r.t. a QBF ϕ is called QRAT *Elimination* (QRATE). Application of QRATE on a QBF $\Pi.\psi \cup \{C\}$ is written as

$$\Pi.\psi \cup \{C\} \xrightarrow{\mathsf{QRATE}(C,l)} \Pi.\psi \quad \text{or, more briefly, as} \quad \Pi.\psi \cup \{C\} \xrightarrow{\mathsf{QRATE}} \Pi.\psi$$

if C and l are clear from the context. Analogously, QRAT allows the introduction of a clause, which has QRAT w.r.t. a QBF ϕ, called QRAT *Addition* (QRATA), written

$$\Pi.\psi \xrightarrow{\mathsf{QRATA}(C,l)} \Pi'.\psi \cup \{C\} \quad \text{or, respectively, as} \quad \Pi.\psi \xrightarrow{\mathsf{QRATA}} \Pi'.\psi \cup \{C\}$$

if C and l are clear from the context. Note that the added clause may contain new variables which do not occur in the original QBF. Then the prefix has to be extended by these variables to obtain a closed QBF again. There is no restriction on how these variables are quantified nor where they are put within the prefix.

Example 12 Consider the true QBF $\Pi.\psi = \forall a \exists b,c.(a \vee b) \wedge (\bar{a} \vee c) \wedge (b \vee \bar{c})$. Clause $(a \vee c)$ has QRAT on c w.r.t. $\Pi.\psi$. The only clause with literal $\bar{c}$ is $(b \vee \bar{c})$, which produces the outer resolvent $(a \vee b)$. Then $\psi \vdash_1 (a \vee b)$, because $(a \vee b) \in \psi$. Hence, QRATA can add $(a \vee c)$ to ψ which yields $\forall a \exists b,c.(a \vee b) \wedge (\bar{a} \vee c) \wedge (b \vee \bar{c}) \wedge (a \vee b)$.

Example 13 Now let d be a new existential variable in the innermost quantifier block of the original formula $\Pi.\psi$ in the previous example. The clause $(\bar{b} \vee c \vee d)$ has QRAT on c (and d) w.r.t. ψ. Adding it through QRATA to ψ gives the true QBF $\forall a \exists b,c,d.(a \vee \bar{b}) \wedge (\bar{a} \vee c) \wedge (b \vee \bar{c}) \wedge (\bar{b} \vee c \vee d)$.

These two examples can also serve to explain elimination, e.g., in the resulting QBFs the added clause has QRAT and thus could be removed through QRATE to obtain the original formula.

For removing or adding a clause which has QRAT on literal l, it is necessary that l is existential. The following example illustrates that it would not be sound either to allow for universal variables or to ignore the variable dependency restrictions.

Example 14 Consider the false QBF $\exists a \forall b.(a \vee b) \wedge (\bar{a} \vee \bar{b})$. Clause $(a \vee b)$ has QRAT on b because the only outer resolvent, i.e., with $(\bar{a} \vee \bar{b})$, is a tautology. Eliminating $(a \vee b)$ does not preserve unsatisfiability. Hence, removing clauses justified by QRAT on universal literals is unsound. If we drop the variable dependency restriction, i.e., also allow inner literals in the definition of outer clause, then in this example $(a \vee b)$ would have QRAT on a since the only resolvent is a tautology. Again, removing $(a \vee b)$ does not preserve unsatisfiability.

However, as we will show below, one can remove universal literals if they have QRAT. This is a generalization of the pure literal elimination rule (see next section) which is a clause elimination technique if the pure literal is existentially quantified and which is a literal elimination technique if the pure literal is universally quantified.

Since QIOR implies QRAT, Theorem 8 gives the following corollary, which corresponds to Theorem 2 in [19].

Corollary 15 *Given QBF $\phi_0 = \Pi.\psi$ and $\phi = \Pi.\psi \cup \{C\}$ where C has* QRAT *on a universal literal $l \in C$ w.r.t. ϕ_0. Further, let $\phi' = \Pi.\psi \cup \{C'\}$ with $C' = C \setminus \{l\}$. Then ϕ and ϕ' are satisfiability equivalent.*

In principle, a universal literal on which a clause C has QRAT w.r.t. to a QBF ϕ may be safely removed from C or vice versa added. In the following, we only need the elimination of universal literals. The elimination of a universal literal l from a clause C which has QRAT on l w.r.t. a QBF ϕ is called QRATU, written

$$\Pi.\psi \cup \{C\} \xrightarrow{\mathsf{QRATU}(C,l)} \Pi.\psi \cup \{C \setminus \{l\}\} \quad \text{or} \quad \Pi.\psi \cup \{C\} \xrightarrow{\mathsf{QRATU}} \Pi.\psi \cup \{C \setminus \{l\}\}$$

if C and l are clear from the context.

Table 1 Preprocessing Rules (the r,s,t indices are all universally quantified).

	name	rewriting rule	precondition	
E1.	*tautology elimination*	$\Pi.\psi, C \vee l \vee \bar{l} \xRightarrow{\mathsf{Taut}} \Pi.\psi$	none	clause elimination
E2.	*subsumption*	$\Pi.\psi, C, D \xRightarrow{\mathsf{Subs}} \Pi.\psi, C$	$C \subseteq D$	
E3.	*existential pure literal elimination*	$\Pi.\psi, C_1 \vee l, \ldots, C_n \vee l \xRightarrow{\mathsf{Pure}_\exists} \Pi.\psi$	$\mathsf{Q}(\Pi, l) = \exists$, $\mathsf{var}(l) \notin \mathsf{vars}(\psi \wedge C_r)$	
E4.	*blocked clause elimination*	$\Pi.\psi, C \vee l \xRightarrow{\mathsf{QBCE}} \Pi.\psi$	$\mathsf{Q}(\Pi, l) = \exists$, $\forall D \in \psi$ with $\bar{l} \in D$: $\exists k, \bar{k} \in C \otimes_l D$ with $k \leq_\Pi l$	
M1.	*universal reduction*	$\Pi.\psi, C \vee l \xRightarrow{\mathsf{URed}} \Pi.\psi, C$	$\mathsf{Q}(\Pi, l) = \forall$, $\forall k \in C$: $k \leq_\Pi l$	clause modification
M2.	*strengthening*	$\Pi.\psi, l \vee C, \bar{l} \vee D \xRightarrow{\mathsf{Str}} \Pi.\psi, C, \bar{l} \vee D$	$D \subseteq C$	
M3.	*unit literal elimination*	$\Pi.\psi, l, C_1 \vee \bar{l}, \ldots, C_n \vee \bar{l}, D_1 \vee l, \ldots, D_m \vee l \xRightarrow{\mathsf{Unit}} \Pi.\psi, C_1, \ldots, C_n$	$\mathsf{Q}(\Pi, l) = \exists$, $\mathsf{var}(l) \notin \mathsf{vars}(\psi \wedge C_r \wedge D_s)$	
M4.	*universal pure literal elimination*	$\Pi.\psi, C_1 \vee l, \ldots, C_n \vee l \xRightarrow{\mathsf{Pure}_\forall} \Pi.\psi, C_1, \ldots, C_n$	$\mathsf{Q}(\Pi, l) = \forall$, $\mathsf{var}(l) \notin \mathsf{vars}(\psi \wedge C_r)$	
M5.	*covered literal addition*	$\Pi.\psi, C \vee l \xRightarrow{\mathsf{QCLA}} \Pi.\psi, C \vee l \vee k$	$\mathsf{Q}(\Pi, l) = \exists$, $\forall D \in \psi$ with $\bar{l} \in D$: $k \in D$ with $k \leq_\Pi l$ or $\exists h, \bar{h} \in C \otimes_l D$ with $h \leq_\Pi l$	
M6.	*equivalence replacement*	$\Pi.\psi, \bar{l} \vee k, l \vee \bar{k} \xRightarrow{\mathsf{Equiv}} \Pi.\psi[l/k]$	$\mathsf{Q}(\Pi, l) = \exists$, $k \leq_\Pi l$	
M7.	*blocked literal elimination*	$\Pi.\psi, C \vee l \xRightarrow{\mathsf{BLE}} \Pi.\psi, C$	$\mathsf{Q}(\Pi, l) = \forall$, $\forall D \in \psi$ with $\bar{l} \in D$: $\exists k, \bar{k} \in C \otimes_l D$ with $k \leq_\Pi l$	
A1.	*variable elimination*	$\Pi \exists y.\psi, C_1 \vee \bar{y}, \ldots, C_n \vee \bar{y}, D_1 \vee y, \ldots, D_m \vee y \xRightarrow{\mathsf{VElim}} \Pi.\psi \bigwedge_{\substack{1 \leq i \leq n \\ 1 \leq j \leq m}} (C_i \cup D_j)$	$\mathsf{Q}(\Pi, y) = \exists$, $y \notin \mathsf{vars}(\psi \wedge C_r \wedge D_s)$	clause addition
A2.	*universal expansion*	$\Pi \forall x \exists Y.\psi, C_1 \vee \bar{x}, \ldots, C_n \vee \bar{x}, D_1 \vee x, \ldots, D_m \vee x, E_1, \ldots, E_p \xRightarrow{\mathsf{UExp}} \Pi \exists Y Y'.\psi, C_1, \ldots, C_n, D'_1, \ldots, D'_m, E_1, \ldots, E_p, E'_1, \ldots, E'_p$	$x \notin \mathsf{vars}(\psi \wedge C_r \wedge D_s \wedge E_t)$, $\exists y_i \in Y : y_i \in \mathsf{vars}(E_t)$, $\forall y_i \in Y : y_i \notin \mathsf{vars}(\psi)$, $D'_s = D_s[y_1/y'_1, \ldots, y_q/y'_q]$, $E'_t = E_t[y_1/y'_1, \ldots, y_q/y'_q]$	

5 Preprocessing for QBFs

To successfully solve quantified Boolean formulas (QBF), the introduction of an additional preprocessing step has been shown to be extremely beneficial in combination with search based solvers [30, 38]. The preprocessed formula even though still in CNF, is in general not logically equivalent, but only satisfiability equivalent. This requires—as we will see later—special actions to reconstruct solutions for the original formula. Below, we introduce techniques for CNF-based preprocessing used in state-of-the-art tools [10, 15, 20, 42] operating on a linear quantifier prefix.

We can distinguish three types of rules: (1) clause elimination rules; (2) clause modification rules; and (3) clause addition rules. Table 1 summarizes the preprocessing techniques and their necessary *preconditions*. The expression $C \otimes_x D$ denotes the resolvent over pivot variable x. We omit showing the soundness of the preprocessing rules as this is extensively discussed in the referenced literature.

Clause Elimination Rules remove clauses while preserving unsatisfiability. *Tautology elimination* (E1) removes clauses containing a positive and negative occurrence of a variable. *Subsumption* [8] (E2) removes clauses that are a superset of other clauses. *Existential pure literal elimination* [13] (E3) removes all clauses with an existential literal that occurs only positive or only negative in the formula. *Quantified blocked clause elimination* [20] (E4) removes clauses which contain an existential literal l producing only tautological resolvents (with opposite literals k and $\bar{k}$) when used as pivot. Note that in the case of QBF, the restriction $k, \bar{k} \leq_\Pi l$ has to hold.

Clause Modification Rules add, remove, and rename literals. The *universal reduction rule* [33] (M1) removes a universal literal if it is the innermost literal in a clause. The *strengthening rule* [8] (M2) relies on clauses produced by resolution which subsume one of its antecedents. If an existential literal l occurs in a clause of size one, then *unit literal elimination* [13] (M3) allows to remove clauses containing l and literal occurrences $\bar{l}$. *Universal pure literal elimination* [13] (M4) removes a universal literal if it occurs only in one polarity in the whole formula. Covered literal addition [20] (M5) extends a clause with literals that occur in all non-tautological resolvents. The *equivalence replacement rule* [8] (M6) substitutes the occurrence of a literal l (and $\bar{l}$) by a literal k (and $\bar{k}$) if clauses of the form $(l \vee \bar{k})$ and $(\bar{l} \vee k)$ are in the formula. Literal l must be existentially quantified and $l \geq_\Pi k$. Finally, the *blocked literal elimination rule* [22] (M7) removes a universal literal l from a clause if all resolvents obtained with l as pivot contain two opposite literals k and $\bar{k}$, i.e., they are tautological, with $k, \bar{k} \leq_\Pi l$.

Clause Addition Rules extend the formula with new clauses, while modifying and removing old ones. The *variable elimination rule* [8] (A1), also known as DP resolution [14], replaces the clauses in which a certain existential variable occurs by all non-tautological resolvents on that variable. The *universal expansion rule* [1] (A2) removes an innermost universal variable by duplicating and modifying all clauses that contain one or more innermost existential variables.

6 Representing Preprocessing Techniques with QRAT

The QRAT proof system as presented above provides clause elimination and addition rules as in propositional logic when the pivot variable is existentially quantified [29]. Further, QRAT allows for the removal/addition of variables in the case of universal pivots [22]. This is almost sufficient to express the preprocessing rules introduced in the previous section. The only missing element is universal reduction, which also marks the difference between propositional resolution and resolution for QBF [33]. To this end, we introduce the concept of *extended universal reduction*, based on Theorem 4.9 of Van Gelder's work on resolution path dependency schemes [44]. In the following, we do not introduce the concept of resolution path dependencies, but we

describe the universal literal elimination criterion according to the terminology used in the rest of the article.

Definition 16 (Inner Clause) Let C be a clause in the QBF $\Pi.\psi$. The *inner clause* of C on literal $l \in C$ is defined as $\mathsf{IC}(\Pi, C, l) = \{k \mid k \in C, k = \bar{l} \text{ or } k >_\Pi l\}$.

Definition 17 (Extended Inner Clause) The *extended inner clause* $\mathsf{EIC}(\Pi, C, l)$ with respect to a QBF $\Pi.\psi$ is the smallest clause such that

1. $C \subseteq \mathsf{EIC}(\Pi, C, l)$,
2. $\mathsf{IC}(\Pi, D, l) \subseteq \mathsf{EIC}(\Pi, C, l)$ if $k \in \mathsf{EIC}(\Pi, C, l)$, $\mathsf{Q}(\Pi, k) = \exists$ and $k >_\Pi l$.

In other words, the extended inner clause $\mathsf{EIC}(\Pi, C, l)$ is the unique clause obtained by repeatedly applying the following extension rule $C := C \cup \mathsf{IC}(\Pi, D, l)$ for $k \in C$ with $\mathsf{Q}(\Pi, k) = \exists$ and $k >_\Pi l$, and $\bar{k} \in D$ with $D \in \psi$ until fixpoint.

Lemma 18 *Given a QBF $\Pi.\psi$. Then for any clause $C \in \psi$ with universal literal $l \in C$ such that $\bar{l} \notin \mathsf{EIC}(\Pi, C, l)$, the removal of l from C is satisfiability preserving.*

Lemma 18 is a generalization of the universal reduction rule which we call *extended universal reduction* in the following. Its application is written as follows:

$$\Pi.\psi \cup \{C\} \xrightarrow{\mathsf{EUR}(C,l)} \Pi.\psi \cup \{C \setminus \{l\}\} \quad \text{or} \quad \Pi.\psi \cup \{C\} \xrightarrow{\mathsf{EUR}} \Pi.\psi \cup \{C \setminus \{l\}\}$$

if C and l are clear from the context.

Now we are able to express the preprocessing techniques shown in Table 1 with only four rules: QRATE, QRATA, QRATU, and EUR. Table 2 shows the translations for the clause elimination techniques, Table 3 for the clause modification techniques, and Table 4 for the clause addition techniques. We refer to Table 1 for preconditions to apply the preprocessing rules.

Tautologies, subsumed clauses as well as blocked clauses have QRAT, so only one application of QRATE is necessary for their removal. If an existential literal is pure then all clauses in which it occurs are blocked w.r.t. this literal (i.e., all resolvents are tautologies; this holds because l occurs only in one polarity and hence the set of resolvents is empty). Therefore such clauses can be omitted by multiple applications of QRATE—we indicate multiple applications of a rule by marking the rule with an asterisk, e.g., for the translation of existential pure we write QRATE$*$.

For strengthening a clause $C \vee l$, we first add the resolvent with $D \vee \bar{l}$ which is C. Now, $C \vee l$ is subsumed and can, as we have discussed before, be removed by QRATE. To express unit literal elimination, we first add clauses C_i, i.e., the resolvents of $C_i \vee \bar{l}$ and l. Then all $C_i \vee \bar{l}$ become QRAT and can be removed. Now the literal l occurs only in one polarity and hence, the clauses containing l can be removed by QRATE (cf., existential pure literal elimination). Universal pure literal elimination simply maps to multiple applications of QRATU such that l does not occur in the formula anymore. If a universal literal l is removed from a clause C, this can naturally be expressed by extended universal reduction.

If k is covered by literal l in $C \vee l$ w.r.t. $\Pi.\psi$, then $C \vee l \vee k$ has QRAT w.r.t. $\Pi.\psi$. After adding $C \vee l \vee k$ using QRATA, $C \vee l$ gets QRAT and can be removed using

Table 2 Clause Elimination Rules.

preprocessing rule	rewriting
$\Pi.\psi, C \vee l \vee \bar{l} \overset{\mathsf{Taut}}{\Longrightarrow} \Pi.\psi$	$\Pi.\psi, C \vee l \vee \bar{l} \xrightarrow{\mathsf{QRATE}} \Pi.\psi$
$\Pi.\psi, C, D \overset{\mathsf{Subs}}{\Longrightarrow} \Pi.\psi, C$	$\Pi.\psi, C, D \xrightarrow{\mathsf{QRATE}} \Pi.\psi, C$
$\Pi.\psi, C_1 \vee l, \ldots, C_n \vee l \overset{\mathsf{Pure}_\exists}{\Longrightarrow} \Pi.\psi$	$\Pi.\psi, C_1 \vee l, \ldots, C_n \vee l \xrightarrow{\mathsf{QRATE}*} \Pi.\psi$
$\Pi.\psi, C \overset{\mathsf{QBCE}}{\Longrightarrow} \Pi.\psi$	$\Pi.\psi, C \xrightarrow{\mathsf{QRATE}} \Pi.\psi$

Table 3 Clause Modification Rules.

preprocessing rule	rewriting
$\Pi.\psi, C \vee l, D \vee \bar{l}$ $\overset{\mathsf{Str}}{\Longrightarrow} \Pi.\psi, C, D \vee \bar{l}$	$\Pi.\psi, C \vee l, D \vee \bar{l}$ $\xrightarrow{\mathsf{QRATA}} \Pi.\psi, C, C \vee l, D \vee \bar{l} \xrightarrow{\mathsf{QRATE}} \Pi.\psi, C, D \vee \bar{l}$
$\Pi.\psi, C_1 \vee \bar{l}, \ldots, C_n \vee \bar{l},$ $l, D_1 \vee l, \ldots, D_m \vee l$ $\overset{\mathsf{Unit}}{\Longrightarrow} \Pi.\psi, C_1, \ldots, C_n$	$\Pi.\psi, C_1 \vee \bar{l}, \ldots, C_n \vee \bar{l}, l, D_1 \vee l, \ldots, D_m \vee l$ $\xrightarrow{\mathsf{QRATA}*} \Pi.\psi, C_1 \vee \bar{l}, \ldots, C_n \vee \bar{l},$ $l, D_1 \vee l, \ldots, D_m \vee l, C_1, \ldots, C_n$ $\xrightarrow{\mathsf{QRATE}*} \Pi.\psi, l, C_1, \ldots, C_n \xrightarrow{\mathsf{QRATE}} \Pi.\psi, C_1, \ldots, C_n$
$\Pi.\psi, C_1 \vee l, \ldots, C_n \vee l$ $\overset{\mathsf{Pure}_\forall}{\Longrightarrow} \Pi.\psi, C_1, \ldots, C_n$	$\Pi.\psi, C_1 \vee l, \ldots, C_n \vee l \xrightarrow{\mathsf{QRATU}*} \Pi.\psi, C_1, \ldots, C_n$
$\Pi.\psi, C \vee l$ $\overset{\mathsf{URed}}{\Longrightarrow} \Pi.\psi, C$	$\Pi.\psi, C \vee l \xrightarrow{\mathsf{EUR}} \Pi.\psi, C$
$\Pi.\psi, \bar{l} \vee k, l \vee \bar{k}$ $\overset{\mathsf{Equiv}}{\Longrightarrow} \Pi.\psi[l/k]$	$\Pi.\psi', C_1 \vee l, \ldots, C_n \vee l, D_1 \vee \bar{l}, \ldots, D_m \vee \bar{l}, \bar{l} \vee k, l \vee \bar{k}$ $\xrightarrow{\mathsf{QRATA}*} \Pi.\psi', C_1 \vee l, \ldots, C_n \vee l, D_1 \vee \bar{l}, \ldots, D_m \vee \bar{l},$ $\bar{l} \vee k, l \vee \bar{k}, C_1 \vee k, \ldots, C_n \vee k, D_1 \vee \bar{k}, \ldots, D_m \vee \bar{k}$ $\xrightarrow{\mathsf{QRATE}*}$ $\Pi.\psi', \bar{l} \vee k, l \vee \bar{k}, C_1 \vee k, \ldots, C_n \vee k, D_1 \vee \bar{k}, \ldots, D_m \vee \bar{k}$ $\xrightarrow{\mathsf{QRATE}*} \Pi.\psi', C_1 \vee k, \ldots, C_n \vee k, D_1 \vee \bar{k}, \ldots, D_m \vee \bar{k}$
$\Pi.\psi, C \vee l$ $\overset{\mathsf{QCLA}}{\Longrightarrow} \Pi.\psi, C \vee l \vee k$	$\Pi.\psi, C \vee l \xrightarrow{\mathsf{QRATA}} \Pi.\psi, C \vee l, C \vee l \vee k$ $\xrightarrow{\mathsf{QRATE}} \Pi.\psi, C \vee l \vee k$
$\Pi.\psi, C \vee l$ $\overset{\mathsf{BLE}}{\Longrightarrow} \Pi.\psi, C$	$\Pi.\psi, C \vee l \xrightarrow{\mathsf{QRATU}} \Pi.\psi, C$

QRATE. Quantified covered clause elimination [10] is a clause elimination procedure that extends clauses with covered literals until clauses become blocked. To represent this procedure, we add an intermediate clause for each covered literal addition. When the clause becomes blocked, it can be eliminated using QRATE.

If a literal l shall be substituted by a literal k due to equivalence replacement, the formula has to contain the binary clauses $(l \vee \bar{k})$ and $(\bar{l} \vee k)$. Then first the clauses $C_i \vee k$ and $D_j \vee \bar{k}$ are added by resolution, i.e., by QRATA. As a consequence, all

Table 4 Clause Addition Rules. Clauses are added / removed in order of appearance.

preprocessing rule	**rewriting**
$\Pi\exists y.\psi, C_1 \vee \bar{y}, \ldots, C_n \vee \bar{y},$ $D_1 \vee y, \ldots, D_m \vee y$ $\xRightarrow{\mathsf{VElim}} \Pi.\psi \bigwedge_{\substack{1\le i\le n \\ 1\le j\le m}} (C_i \cup D_j)$	$\Pi\exists y.\psi, C_1 \vee \bar{y}, \ldots, C_n \vee \bar{y}, D_1 \vee y, \ldots, D_m \vee y$ $\xrightarrow{\mathsf{QRATA}*} \Pi\exists y.\psi, C_1 \vee \bar{y}, \ldots, C_n \vee \bar{y},$ $D_1 \vee y, \ldots, D_m \vee y, C_1 \cup D_1, \ldots, C_n \cup D_m$ $\xrightarrow{\mathsf{QRATE}*} \Pi.\psi, C_1 \cup D_1, \ldots, C_n \cup D_m$
$\Pi\forall x\exists Y.\psi,$ $C_1 \vee \bar{x}, \ldots, C_n \vee \bar{x},$ $D_1 \vee x, \ldots, D_m \vee x,$ $E_1, \ldots, E_p$ $\xRightarrow{\mathsf{UExp}} \Pi\exists YY'.\psi,$ $C_1, \ldots, C_n, E_1, \ldots, E_p,$ $D'_1, \ldots, D'_m, E'_1, \ldots, E'_p$	$\Pi\forall x\exists Y.\psi, C_1 \vee \bar{x}, \ldots, C_n \vee \bar{x},$ $D_1 \vee x, \ldots, D_m \vee x, E_1, \ldots, E_p$ $\xrightarrow{\mathsf{QRATA}*} \Pi\forall x\exists YY'.\psi, C_1 \vee \bar{x}, \ldots, C_n \vee \bar{x},$ $D_1 \vee x, \ldots, D_m \vee x, E_1, \ldots, E_p,$ $E_1 \vee \bar{x}, \ldots, E_p \vee \bar{x},$ $x \vee y_1 \vee \bar{y}'_1, \ldots, x \vee y_{\lvert Y\rvert} \vee \bar{y}'_{\lvert Y\rvert},$ $x \vee \bar{y}_1 \vee y'_1, \ldots, x \vee \bar{y}_{\lvert Y\rvert} \vee y'_{\lvert Y\rvert},$ $D'_1 \vee x, \ldots, D'_m \vee x, E'_1 \vee x, \ldots, E'_p \vee x$ $\xrightarrow{\mathsf{QRATE}*} \Pi\forall x\exists YY'.\psi, C_1 \vee \bar{x}, \ldots, C_n \vee \bar{x},$ $E_1 \vee \bar{x}, \ldots, E_p \vee \bar{x}, D'_1 \vee x, \ldots, D'_m \vee x,$ $E'_1 \vee x, \ldots, E'_p \vee x$ $\xrightarrow{\mathsf{EUR}*} \Pi\exists YY'.\psi, C_1, \ldots, C_n, E_1, \ldots, E_p,$ $D'_1, \ldots, D'_m, E'_1, \ldots, E'_p$

clauses containing l and $\bar{l}$ are asymmetric tautologies and can therefore be removed by QRATE.

A blocked literal is a universal literal which has QRAT, therefore, BLE can be simply expressed by QRATU.

Variable elimination is rewritten as follows. First all possible non-tautological resolvents on the pivot variable y are added with QRATA. Then all clauses containing y or $\bar{y}$ become QRAT and can be eliminated by QRATE.

Finally, we describe universal expansion using redundancy elimination and addition rules. Consider the QBF $\Pi\forall x\exists Y.\psi$ from which we want to eliminate the innermost universal variable x. Let $E = \{E_t \mid E_t \in \psi_Y, x \notin E_t, \bar{x} \notin E_t\}$. In the first step, we add clauses $E_t \vee \bar{x}$ (which are subsumed by E_t) using QRATA. This is necessary, because we later need to eliminate E_t. We introduce conditional equivalences represented by the clauses $x \vee y_i \vee \bar{y}'_i$ and $x \vee \bar{y}_i \vee y'_i$ for all $y_i \in Y$ and append $\exists Y'$ to the prefix. Now we copy all original clauses with literal y_i, but without $\bar{x}$ and add literal x in case it is not already present. The conditional equivalences allow to treat original and primed copies of clauses with x as alternative. One version can be exchanged for the other as long the equivalence clauses are there. We add the primed copies and afterwards remove the original ones. Now all clauses E_i are asymmetric tautologies and can be removed. Next, we remove the conditional equivalences $x \vee y_i \vee \bar{y}'_i$ and $x \vee \bar{y}_i \vee y'_i$ which have QRAT on the y_i after removal of the E_t clauses. At this point,

clauses containing variables from Y do not contain x and clauses with variables from Y' do not contain $\bar{x}$. So extended universal reduction can remove the literals x and $\bar{x}$.

7 QRAT Proofs

This section describes our new proof format for QBFs and how to check it. The syntax of the proof format is very similar to the DRAT proof format [46] for propositional formulas in CNF. It is extended to express elimination of universal literals though. Furthermore, the redundancy check in the original DRAT checker is adapted to take QBF quantifier dependencies into account.

7.1 The QRAT Proof Format

Proofs are sequences of clause additions, deletions, and modifications. They are built using three kind of lines: addition (QRATA), deletion (QRATE), and universal elimination (QRATU and EUR). In our format, addition lines have no prefix and are unconstrained in the sense that one can add any clause at any point in the proof. Clause deletion lines, with prefix "d", and universal elimination lines, with prefix "u", are restricted. The clause after a "d" or "u" prefix must be either present in the original formula or as a clause added earlier in the proof.

Definition 19 (Proof Step) Let P be a QRAT proof of length $|P|$ for a QBF $\Pi.\psi$. For each proof step $i \in \{0,\ldots,|P|\}$, we define ψ_P^i, a CNF formula, as below, where C_i refers to the clause in line i of P and l_i refers to the first literal on line i of P.

$$\psi_P^i := \begin{cases} \psi & \text{if } i = 0; \\ \psi_P^{i-1} \setminus \{C_i\} & \text{if the prefix of } C_i \text{ is "d"}; \\ \psi_P^{i-1} \setminus \{C_i\} \cup \{C_i \setminus \{l_i\}\} & \text{if the prefix of } C_i \text{ is "u"}; \\ \psi_P^{i-1} \cup \{C_i\} & \text{otherwise.} \end{cases}$$

A proof P is called a *satisfaction proof* (a proof of satisfiability) for QBF $\Pi.\psi$ if the following two properties hold. First, for all $i \in \{1,\ldots,|P|\}$, if clause C_i has prefix "d", then it must have QRAT on l_i with respect to ψ_P^i. In case l_i is universally quantified, we check whether C_i is an asymmetric tautology with respect to ψ_P^i. Second, $\psi_P^{|P|}$ must be empty.

A proof P is called a *refutation proof* for QBF $\Pi.\psi$ if the following three properties hold. First, for all $i \in \{1,\ldots,|P|\}$, if clause C_i has no prefix, then it must have QRAT on l_i with respect to ψ_P^{i-1}. In case l_i is universally quantified, we check whether C_i is an asymmetric tautology with respect to ψ_P^{i-1}. Second, for all $i \in \{1,\ldots,|P|\}$, if clause C_i has prefix "u", then l_i must be universally quantified. Additionally, C_i must have either QRAT on l_i with respect to ψ_P^{i-1}, or l_i can be removed using EUR. Third, $C_{|P|}$ must be the empty clause (without a prefix). Figure 2 shows a true and a false QBF and a QRAT proof for both.

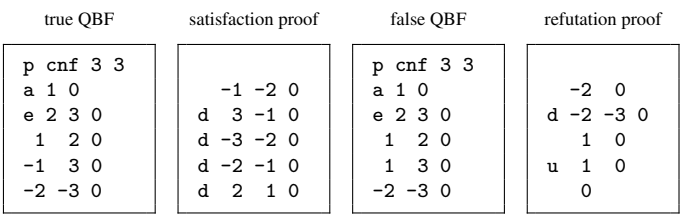

Fig. 2 Two QBFs in QDIMACS format and QRAT proofs. The true QBF $\forall a \exists b,c.(a \vee b) \wedge (\bar{a} \vee c) \wedge (\bar{b} \vee \bar{c})$ on the left with a satisfaction proof next to it. On the right the false QBF $\forall a \exists b,c.(a \vee b) \wedge (a \vee c) \wedge (\bar{b} \vee \bar{c})$ with a refutation next to it. The formulas and proofs are spaced to improve readability. Proofs consist of three kind of lines: addition (no prefix), deletion ("`d` " prefix) and universal elimination ("`u` " prefix).

A universal elimination line in satisfaction proofs can be replaced by a clause addition and deletion line to obtain another satisfaction proof. Therefore, the clause without its first literal is added and afterwards the subsumed clause is deleted. For example, consider the line "`u 1 2 3 0`" in a satisfaction proof. This line can be replaced by "`2 3 0`" followed by "`d 1 2 3 0`". Consequently, any satisfaction proof can be converted such that it contains only addition and deletion lines.

Recall that QRATA can add clauses that contain new variables. The QRAT proof format does not support the specification of the quantifier block where the new variables shall be placed. For all known preprocessing techniques, newly introduced variables are placed in the innermost active existential quantifier block. Consequently, the QRAT format assumes this convention for all new variables.

7.2 Checking QRAT Proofs

Although the syntax for QRAT proofs is identical for true and false QBFs, validating a proof is different. For true QBFs only the clause deletion lines (the ones with a "d" prefix) have to be checked, while for false QBFs, all the lines except the clause deletion lines have to be checked.

The easiest, but rather expensive, method to validate proofs checks the redundancy of each clause: for true QBFs all deletion lines and for false QBFs all addition and universal elimination lines.

Example 20 Consider the true QBF $\Pi.\psi = \forall a \exists b,c.(a \vee b) \wedge (\bar{a} \vee c) \wedge (\bar{b} \vee \bar{c})$. This is the same QBF as in Figure 2 (left). Figure 2 also shows the satisfaction proof $P := (\bar{a} \vee \bar{b}), \mathtt{d}(c \vee \bar{a}), \mathtt{d}(\bar{c} \vee \bar{b}), \mathtt{d}(\bar{b} \vee \bar{a}), \mathtt{d}(b \vee a)$. Satisfaction proofs are checked in chronological order. So, first, $(\bar{a} \vee \bar{b})$ is added, afterwards $(c \vee \bar{a})$ is removed, until all original clauses and all added clauses have been deleted.

Figure 3 shows the basic algorithm to validate satisfaction QRAT proofs. Let us ignore line `v1`, `v2`, `v9`, and `v13` for the moment, because they are only required to produce Skolem functions which we will discuss in the next section. We loop over the clauses in the proof in the order a QBF solver or preprocessor added or removed clauses (line `v3`). The first unexamined clause is obtained from the proof together with its flag and pivot (line `v4`). The flag can be either `add` or `delete` (in the proof format

```
     validateQRAT (QBF Π.ψ, QRAT proof P)
v1     let V = vars(P)
v2     initSkolem ()
v3     while P ≠ ∅ do
v4       ⟨flag, l, C⟩ := P.dequeue()
v5       if flag = delete then
v6         ψ := ψ \ {C}
v7         if ψ ⊢1 C then continue
v8         else if C has QRAT on l ∈ C w.r.t. Π.ψ then
v9           addSkolem (C, l)
v10        else return 'INVALID PROOF'
v11      else ψ := ψ ∪ {C}
v12    if ψ ≠ ∅ then return 'INVALID PROOF'
v13    finishSkolem ()
v14    return 'VALID PROOF'
```

Fig. 3 Procedure to check satisfaction QRAT proofs and output Skolem functions.

no prefix or a "d" prefix, respectively). If the flag is add, no checking is required because this is a strengthening step. The new clause is simply added to ψ (line v11). Else, the clause will be removed (line v6). This elimination step needs to be validated. We check if the clause is logically implied by ψ by computing whether the clause is an asymmetric tautology (line v7). If that is not the case, the clause needs to have QRAT w.r.t. ψ (line v8), otherwise the proof is invalid (line v10). This procedure continues until all clauses in the proof have been processed. At this point, ψ should be empty, showing that the original formula is satisfiability equivalent to the empty formula. If ψ is empty, the proof is valid (line v14), otherwise it is invalid (line v12).

However, one can check proofs more efficiently by marking involved clauses during each redundancy check. That way the checker can be restricted to validate marked clauses only. The marking procedure is a bit tricky. It would be easy to mark all propagating clauses, but this unnecessarily marks too many clauses, thereby increasing the clauses that need to be checked. Instead, it is possible to mark only a subset of the propagated clauses, similarly as computing the clauses that are involved in a conflict via conflict analysis in SAT solvers. Checking only marked clauses was proposed to check clausal proofs of CNF formulas efficiently [16].

For false QBFs, checking is similar to the SAT case: during initialization the empty clause is marked. Refutation proofs are validated in reverse order, starting with the marked empty clause. The right part of Figure 2 shows a false QBF and a refutation proof for that QBF. For true QBFs the procedure differs: initially all original clauses are marked and satisfaction proofs are checked in chronological order. The left part of Figure 2 shows a true QBF and a satisfaction proof for that QBF.

We equipped our preprocessor bloqqer [10] with QRAT-based tracing as described in Section 6. In contrast to previous extensions of bloqqer [28, 36] we hardly had to modify its internal behavior. Hence, with QRAT-based tracing, we have the first QBF preprocessor fully supporting proof generation for true and false formulas.

We implemented an efficient QRAT checker QRATtrim,[1] which is based on DRATtrim [46], a clausal proof checking tool for CNF. It uses the optimizations of Section 7.2, such as validating marked clauses only and checking satisfaction and refutation proofs in chronological and reverse order, respectively.

8 Extracting Skolem Functions

In this section, we introduce an approach to extract Skolem functions from QRAT proofs. We present this approach on three levels: first, we provide formal definitions and prove soundness of our approach. Second, we describe pseudo code, explaining how the approach can be implemented efficiently. Third, we discuss more technical details of the implementation.

8.1 Skolem Functions for QBF

Whereas in propositional logic a model of a formula is given by a satisfying variable assignment, for a QBF a "model" has to reflect the variable dependencies between existential and universal variables. Beside assignment trees as above, another convenient and potentially much more succinct way is to express QBF models as Skolem functions which are defined as follows.

Definition 21 (Skolem Function) Let x be an existential variable of a satisfiable QBF $\phi = \Pi.\psi$ and let $y_1, \ldots, y_n$ be all universals of ϕ with $y_i \leq_\Pi x$. Then a propositional formula $f(y_1, \ldots, y_n)$ is a *Skolem function* for x.

Definition 22 (Skolem Set) Given a QBF ϕ. A mapping F of Skolem functions which maps every existential variable $x \in \mathsf{vars}(\phi)$ to a Skolem function $\mathsf{F}(x)$ of x is called a *Skolem set*.

We do not assume that Skolem functions are in CNF, i.e., we allow any Boolean connective like $\vee, \wedge, \rightarrow, \neg, \ldots$ under its standard semantics. In particular, we use the *if-then-else* connective with $\mathsf{ite}(\psi_1; \psi_2; \psi_3)$ defined as $(\psi_1 \rightarrow \psi_2) \wedge (\bar{\psi}_1 \rightarrow \psi_3)$.

Definition 23 A Skolem function $f(y_1, \ldots, y_n)$ is *valid* on a QBF ϕ iff $\phi[x/f] \sim \phi$. A Skolem set is *valid* on a QBF ϕ iff it is a set of valid Skolem functions on ϕ.

Obviously, every Skolem set on ϕ can be interpreted as a symbolic (potentially exponentially more succinct) representation of a pre-model of ϕ and vice versa. Further a Skolem set is valid iff its corresponding pre-model is actually a model.

Note that in practice we might also want to consider Skolem functions that depend on existential variables too. This has the potential of much more succinct Skolem sets, particularly if represented as trees instead of circuits. If we want existential variables to occur then we can extend the prefix order $\leq_\Pi$ to a total order on all variables and

[1] The QRATtrim version with Skolem function extraction is available on `http://www.cs.utexas.edu/~marijn/skolem/`.

require that Skolem functions depend only on outer universal and existential variables w.r.t. this total order. All the arguments of this section apply to this extended definition as well, after eliminating existential variables in Skolem functions through simple syntactic substitution. Note, that substitution is a linear operation if Skolem functions are represented as circuits, e.g., with and-inverter-graphs (AIGs).

Given a QBF $\Pi.\psi$ containing an existential variable x, the function $f(U)$ denotes a Skolem function for x with the set of universal variables U that are outer to x in Π, as parameters. Note that in the following, we omit U if clear from the context. We also write $\mathsf{F}(x) = f$ for a Skolem set F and f the Skolem function of x.

The substitution of existential literals by their Skolem functions is defined as follows, yielding a formula with universal quantification only.

Definition 24 (Substitution by Skolem Set) Consider $\{x_1,\ldots,x_n\}$, the set of all the existential variables of a QBF ϕ. Further let F be an (extended) Skolem set of ϕ. Then define $\phi[\mathsf{F}] = \phi[x_1/\mathsf{F}(x_1),\ldots,x_n/\mathsf{F}(x_n)]$.

The following notion of *outer formula* is used for constructing Skolem functions.

Definition 25 (Outer Formula) The *outer formula* of an existential l w.r.t. the QBF $\Pi.\psi$, denoted by $\mathsf{OF}(\Pi,\psi,l)$, is defined as $\{\mathsf{OC}(\Pi,D,\bar{l}) \mid D \in \psi, \bar{l} \in D\}$.

We interpret an outer formula as a CNF, i.e., a conjunction of the respective outer clauses and use the following properties for the construction of Skolem functions, which are immediate consequences of Theorem 7 (or Corollary 11).

Corollary 26 *Let clause C have* QRAT *on an existential literal $l \in C$ w.r.t. a QBF $\Pi.\psi$ with $\mathsf{var}(l) = x$. If an assignment $\tau = \tau^x \bar{l} \tau_x$ falsifies C but satisfies ψ, then τ^x satisfies the outer formula $OF(\Pi,\psi,l)$.*

Corollary 27 *Let clause C have* QRAT *on an existential literal $l \in C$ w.r.t. a QBF $\Pi.\psi$ with $\mathsf{var}(l) = x$. Further, let $\tau = \tau^x \bar{l} \tau_x$ be a satisfying assignment of ψ. If the outer formula $OF(\Pi,\psi,l)$ is falsified by assignment τ^x then C is satisfied by τ too.*

Although we do not manipulate the prefix Π for ψ_P^i in Def. 19 explicitly, we assume that variables not occurring in the current prefix are removed and new variables are added appropriately. Refer to this updated prefix as Π_P^i and define $\phi_P^i = \Pi_P^i.\psi_P^i$.

In order to construct the Skolem functions, we traverse the QRAT proof in reverse order. Using the last corollary above, we can satisfy a clause C deleted at step i in a QRAT proof with $\mathsf{ite}(\mathsf{OF}(\Pi_P^i,\psi_P^{i+1},l_i)[\mathsf{F}_P^{i+1}];\mathsf{pol}(l_i);\mathsf{F}_P^{i+1})$. Now we have all ingredients to define the construction of Skolem functions.

Definition 28 (Construction of Skolem Functions) Let $\Pi.\psi$ be a satisfiable QBF and P be a QRAT satisfaction proof of $\Pi.\psi$ with length $|P|$. Then a Skolem trace T_P of P is a sequence of Skolem sets $(\mathsf{F}_P^0,\ldots,\mathsf{F}_P^{|P|})$ for proof steps $(\psi_P^0,\ldots,\psi_P^{|P|})$ defined as follows. In the last proof step $i = |P|$ for all existential variables $x \in \mathsf{vars}(P)$ pick constant Skolem functions $\mathsf{F}_P^{|P|}(x) \in \{\bot,\top\}$ arbitrarily. In earlier proof steps $\mathsf{F}_P^i(x) = \mathsf{F}_P^{i+1}(x)$ unless the proof step has prefix "d" and l is the first literal of clause C_i with $\mathsf{var}(l) = x$. In this case set $\mathsf{F}_P^i(x) = \mathsf{ite}(\mathsf{OF}(\Pi_P^i,\psi_P^{i+1},l)[\mathsf{F}_P^{i+1}];\mathsf{pol}(l);\mathsf{F}_P^{i+1}(x))$.

In the definition above, we introduce a set of propositional formulas by traversing the proof in reverse order: (1) In the last proof step, i.e., when the empty formula has been derived, all variables occurring in the proof are initialized with an arbitrary truth constant. (2) If a clause is deleted at step i, the current Skolem function f'_x of the previous step is modified. (3) If a clause is updated at step i or added, then the set of propositional formulas remains the same as in step $i+1$.

Theorem 29 *Let $\phi = \Pi.\psi$ be a satisfiable QBF and P be a* QRAT *proof of ϕ of length n, i.e., $|P| = n$. Let $\mathsf{F}_P = \mathsf{F}_P^0$ be a Skolem set with F_P^0 constructed from a Skolem trace $(\mathsf{F}_P^0, \ldots, \mathsf{F}_P^n)$ as described above. Then F_P is valid on ϕ.*

Proof We show by reverse induction that Skolem set F_P^i is valid on ϕ_i, e.g., $\phi_i[F_P^i] \sim \top$. The base case $i = n$ is trivial. Assume $\phi_{i+1}[\mathsf{F}_P^{i+1}] \sim \top$ for $i < n$. If proof step i with clause C_i has no prefix (addition) or prefix "u", we have $\psi_P^{i+1} \models \psi_P^i$ and the Skolem set does not change which concludes the induction step.

Otherwise proof step i has prefix "d". Let l be the first (existential) literal in C_i. With the induction hypothesis we get the model M of ϕ_P^{i+1} corresponding to F_P^{i+1} such that $\tau(C) = \top$ for every clause $C \in \psi_P^{i+1}$ and τ in M. Let M' be the pre-model of ϕ_P^i which corresponds to F_P^i. We show that M' is a model of ϕ_P^i or equivalently $\tau(C) = \top$ for all clauses $C \in \psi_P^i$ and for all τ in M'. Due to the construction of Skolem traces, M and M' might only differ (and thus potentially not satisfy ψ_P^i) for assignments τ on $x = \mathsf{var}(l)$, for which $\mathsf{F}_P^i(x) = \mathsf{ite}(\mathsf{OF}(\Pi_P^i, \psi_P^{i+1}, l)[\mathsf{F}_P^{i+1}]; \mathsf{pol}(l); \mathsf{F}_P^{i+1}(x))$. First, if $\tau(\mathsf{OF}(\Pi_P^i, \psi_P^{i+1})[\mathsf{F}_P^{i+1}]) = \top$, then τ trivially satisfies all clauses $C \in \psi_P^i$ with $l \in C$ (including C_i), because $\tau(x) = \mathsf{pol}(l)$ and thus $\tau(l) = \top$. It further satisfies all clauses $D \in \psi_P^i$ with $\bar{l} \in D$ since $\mathsf{OF}(\Pi_P^i, \psi_P^{i+1}) \models D$. In the other case, where $\tau(\mathsf{OF}(\Pi_P^i, \psi_P^{i+1})[\mathsf{F}_P^{i+1}]) = \bot$, we have $\tau \in M$ by construction, τ satisfies ψ_P^{i+1} and then also C_i by Cor. 27. Therefore all clauses $C \in \psi_P^i$ are satisfied. □

To check that a Skolem set F_P generated from proof P is valid, each existential variable x of QBF ϕ has to be substituted by its corresponding Skolem function $\mathsf{F}_P(x)$. The resulting propositional formula must be valid. This validity check can be done by a SAT solver. In practice, it also needs to be ensured that a given Skolem function for x does not contain universal variables y_i with $y_i >_\Pi x$. This syntactic criterion can easily be checked. Thus while the satisfiability checking problem of QBF is PSPACE complete, checking validity of a Skolem set is in co-NP [35].

8.2 Skolem Function Extraction

Figure 4 shows the procedures used to extract Skolem functions. These procedures are plugged into the proof validation procedure in Fig. 3, which goes forward over the proof, while the formal Skolem functions construction in Def. 28 and Thm. 29 goes backward over the proof.

A forward version can be obtained by introducing a new existential variable y for every update to a Skolem function of a variable x. This y acts as place holder for the still to be determined Skolem function of x at this point, which could be obtained by processing the tail of the proof in reverse oder, as in the backward construction. This

```
      initSkolem ()
iS1   foreach existential x ∈ V do
iS2     L(x) := x,  G(x) := x

      addSkolem (clause C, literal l)
aS1   let x be L(var(l))
aS2   let y be a new existential variable
        in the same quantifier block as x
aS3   L(y) := y,  G(y) := y
aS4   let O := OC(Π,C,l) ∪ {l}
aS5   if O has QRAT on l w.r.t. Π.ψ then
aS6     G(x) := ite(OC(Π,C,l)[L];y;pol(l))
aS7   else
aS8     G(x) := ite(OF(Π,ψ,l)[L];pol(l);y)
aS9   L(x) := y

      finishSkolem ()
fS1   foreach existential x ∈ V do
fS2     G(L(x)) := ⊤
```

Fig. 4 Procedures to *init*, *add*, and *finish* Skolem functions (used in Fig. 3).

recursive process is similar to lazy evaluation in functional programming or to the concept of "holes" in advanced Prolog programming, i.e., the Skolem function of the new y will also be determined later during validation. At the end, see *finishSkolem*, all the not yet determined Skolem functions are set to be constant (line `fS2`), which corresponds to the base case in Def. 28 and Thm. 29.

In general this lazy forward approach will yield Skolem functions which depend on existential variables too. However, as already discussed above, these dependencies can be eliminated by syntactic substitution. Our implementation will perform this substitution on-the-fly, while dumping the Skolem functions as AIGs.

The algorithm maintains two global data structures. The first one L maps each (original and introduced) existential variable to its *last* introduced place holder variable, on which its Skolem function depends. Initially, see line `iS2` in *initSkolem*, $\mathsf{L}(x) := x$ for all existential variables in the original QBF $\phi = \Pi.\psi$. This map is extended for newly introduced variables y in the same way (line `aS3`). If the clause C_i in line i of the proof P is processed by *addSkolem* the place holder $x = \mathsf{L}(\mathsf{var}(l))$ corresponds to $\mathsf{F}^i_P(\mathsf{var}(l))$ of Def. 28.

The second global data structure G maps each existential variable to its partial Skolem function. These Skolem functions are built top-down. Each update (line `aS6`, `aS8`) in essence fills in a hole (x) and introduces a new one (y). At the end, after filling the remaining holes by constants in *finishSkolem*, $\mathsf{G}(x)$ corresponds to $\mathsf{F}^0_P(x)$ for all original existential variables x. The final Skolem function of x is obtained by $\mathsf{G}(x)[\mathsf{G}]$, which substitutes partial Skolem functions of all existential variables recursively.

While describing the construction of Skolem functions (Def. 28), we discussed how to update Skolem functions when adding a QRAT clause C. This update step requires to evaluate the outer formula of the pivot. The outer formula can be large which in turn would make the Skolem functions large. Therefore, we first check whether we can avoid computing the outer formula. This can be done when O, a copy of C with all inner literals to the pivot removed, has QRAT as well.

The argument is as follows. First, add O to the current formula. Since O subsumes C, the clause C can be removed, while preserving logical equivalence. We know that O still has QRAT, because removing a subsumed clause does not influence that property. As a consequence, applying Lemma 6, we obtain the following. If an assignment τ satisfies the current formula, but falsifies $O \setminus \{l\}$, then τ^x with $\mathsf{var}(l) = x$ satisfies $\mathsf{OF}(\Pi, O, l)$. Furthermore τ^x falsifies $O \setminus \{l\}$, because O has no inner literals to l. Hence instead of checking that τ^x satisfies $\mathsf{OF}(\Pi, O, l)$, we could check whether τ^x falsifies $O \setminus \{l\}$. Observe that this check is different in the case τ^x satisfies both $\mathsf{OF}(\Pi, O, l)$ and $O \setminus \{l\}$. In this case, we could either keep the current Skolem function (because τ^x satisfies $O \setminus \{l\}$) or flip the polarity of l to true (because τ^x satisfies $\mathsf{OF}(\Pi, O, l)$). The alternative method would perform the former, while the default method performs the latter.

Now we have all elements to explain the *addSkolem* procedure. A new existential variable y is created (line `aS2`), in the same quantifier block as x. Afterwards, we compute O, a copy of C with all inner literals to the pivot removed (line `aS4`), which is the same as the outer clause of C w.r.t. l but with l added back. If O has QRAT on l w.r.t. $\Pi.\psi$ (line `aS4`) then the Skolem function only needs to check whether the outer clause of C w.r.t. the pivot is falsified. In this case, the Skolem function for the pivot is updated as shown in the pseudo-code on line `aS6`.

Otherwise, if this optimization is not possible, the Skolem function is updated to check the outer formula of the pivot as shown in line `aS8`, which exactly matches the way how Skolem functions are updated in Def. 28. The procedure terminates by updating L in line `aS9`, which marks y as new place holder for x. The substitutions denoted "$\ldots[\mathsf{L}]$" in these two updates replace all occurrences of an existential variable z by $\mathsf{L}(z)$ and $\neg z$ by $\neg\mathsf{L}(z)$. This corresponds to the substitution "$\ldots[\mathsf{F}_P^{i+1}]$" in Def. 28, which uses the same notation.

Example 30 The true QBF below and in Fig. 5 is used to illustrate how the extraction of Skolem functions from a QRAT proof works:

$$\Pi.\psi := \exists a, b \forall x, \exists c.(a \vee b) \wedge (\bar{a} \vee \bar{b}) \wedge (\bar{a} \vee x \vee c) \wedge (\bar{x} \vee \bar{c})$$

Figure 5 also shows how this formula looks in the QDIMACS format, which is used by most QBF solvers and preprocessors (left) and a QRAT proof for that formula (right). Note that in QRAT proofs, the pivot of clause C is the first literal appearing in the clause deletion line corresponding to C. During initialization L and G are assigned as follows: $\mathsf{L}(a) := \mathsf{G}(a) := a$, $\mathsf{L}(b) := \mathsf{G}(b) := b$, and $\mathsf{L}(c) := \mathsf{G}(c) := c$.

The proof consists of the following steps. First, $C_1 := (\bar{a} \vee \bar{b})$ is removed from $\psi_P^0 = \psi$ using $\bar{b}$ as pivot. Since C_1 has no inner literals, $O_1 := C_1$ and consequently O_1 has QRAT on $\bar{b}$ w.r.t. the new ψ_P^1 obtained from ψ_p^0 after removing $C_1 = (\bar{a} \vee \bar{b})$:

$$\psi_P^1 \equiv (a \vee b) \wedge (\bar{a} \vee x \vee c) \wedge (\bar{x} \vee \bar{c})$$

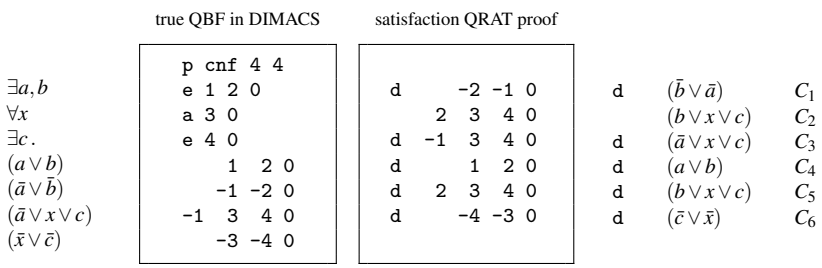

Fig. 5 A true QBF (left) with a satisfaction proof (right). The boxed ASCII versions of the formula and the proof are spaced to improve readability. Proofs consist of two kind of lines: addition (no prefix) and deletion ("d " prefix). The formula and proof represent our example in the QDIMACS and QRAT format, respectively. Variables in both formats are numbers. The following mapping is used (a,b,x,c) corresponds to $(1,2,3,4)$. Negative literals are shown as negative numbers.

We introduce a new existential variable b_1 and update G in line aS6 as follows

$$\begin{aligned}\mathsf{G}(b) := \mathsf{ite}\big(\mathsf{OC}(\Pi_P^0, C_1, \bar{b})[\mathsf{L}]; b_1; \bot\big) &\equiv \mathsf{ite}\big(\mathsf{OC}(\Pi_P^0, (\bar{a}\vee\bar{b}), \bar{b})[\mathsf{L}]; b_1; \bot\big)\\ \equiv \mathsf{ite}\big((\bar{a})[\mathsf{L}]; b_1; \bot\big) &\equiv \mathsf{ite}\big(\neg\mathsf{L}(a); b_1; \bot\big)\\ \equiv \mathsf{ite}\big(\bar{a}; b_1; \bot\big) &\equiv \bar{a}\wedge b_1\end{aligned}$$

For the new variable b_1 we initialize $\mathsf{L}(b_1) := \mathsf{G}(b_1) := b_1$ in line aS3 and remember $\mathsf{L}(b) = b_1$ in line aS9. Then the second step in the proof adds clause $C_2 = (b\vee x\vee c)$ to ψ_P^1. Since this involves clause addition, no Skolem function is added, but leads to:

$$\psi_P^2 \;\equiv\; (a\vee b)\wedge(\bar{a}\vee x\vee c)\wedge(\bar{x}\vee\bar{c})\wedge(b\vee x\vee c)$$

The third step is the most tricky one. Clause $C_3 := (\bar{a}\vee x\vee c)$ is now removed using pivot $\bar{a}$. It has QRAT on $\bar{a}$ on the following formula ψ_P^3 w.r.t. $\bar{a}$, since its single outer resolvent $\mathsf{OR}\big(\Pi_P^2, C_3, (a\vee b), \bar{a}\big) = (b\vee x\vee c)$ is even subsumed.

$$\psi_P^3 \;\equiv\; (a\vee b)\wedge(\bar{x}\vee\bar{c})\wedge(b\vee x\vee c)$$

In C_3 both x and c are inner to $\bar{a}$, so $O_3 := (\bar{a})$. The clause O_3 does not have QRAT on $\bar{a}$ w.r.t. ψ_P^3, and we can not use the optimization in line aS6. Instead we have to use the regular update in line aS8, and then remember $\mathsf{L}(a) = a_1$ with a_1 the new variable.

$$\mathsf{G}(a) := \mathsf{ite}\big(\mathsf{OF}(\Pi_P^2, \psi_P^3, \bar{a})[\mathsf{L}]; \bot; a_1\big) \equiv \mathsf{ite}\big((b)[\mathsf{L}]; \bot; a_1\big) \equiv \mathsf{ite}(b_1; \bot; a_1) \equiv a_1\wedge\bar{b}_1$$

The fourth step removes $C_4 = (a\vee b)$ using pivot a. This step is practically the same as the first step, and with $\mathsf{L}(a) = a_1$ and $\mathsf{L}(b) = b_1$ (before the update) we get

$$\mathsf{G}(a_1) := \mathsf{ite}\big(\mathsf{OC}(\Pi_P^3, C_4, a)[\mathsf{L}]; a_2; \top\big) \equiv \mathsf{ite}\big(b_1; a_2; \top\big) \equiv a_2\vee\bar{b}_1$$

Then we remember $\mathsf{L}(a) = a_2$ (after the update) with a_2 the new introduced variable.

$$\psi_P^4 \;\equiv\; (\bar{x}\vee\bar{c})\wedge(b\vee x\vee c)$$

In the fifth step, clause $C_5 := (b\vee x\vee c)$, which was added in step two, is removed. Again, the literals x and c are inner to b. This results in $O_5 := (b)$. In contrast to step

three, this O_5 has QRAT on b w.r.t. the new $\psi_P^5 \equiv (\bar{x} \vee \bar{c})$ because it no longer contains any clause with literal $\bar{b}$. With $\mathsf{L}(b) = b_1$ the optimized update in line aS6 gives

$$\mathsf{G}(b_1) := \mathsf{ite}\big(\mathsf{OC}(\Pi_P^4, C_5, b)[\mathsf{L}]; b_2; \top\big) \equiv \mathsf{ite}\big(\bot; b_2; \top\big) \equiv \top$$

Then we update $\mathsf{L}(b) = b_2$ with the new variable b_2. Finally, the last remaining clause $C_6 = (\bar{x} \vee \bar{c})$ with pivot $\bar{c}$ is removed. The Skolem function update in line aS6 gives

$$\mathsf{G}(c) := \mathsf{ite}\big(\mathsf{OC}(\Pi_P^5, C_6, \bar{c})[\mathsf{L}]; c_1; \bot\big) \equiv \mathsf{ite}\big(\bar{x}; c_1; \bot\big) \equiv \bar{x} \wedge c_1$$

After the QRAT proof has been validated, we call *finishSkolem* which does the following assignments: $\mathsf{G}(a_2) := \mathsf{G}(b_2) := \mathsf{G}(c_1) := \top$ and after collecting and parallel substitution of all those partial Skolem functions G we get

$$\begin{aligned}
\mathsf{F}_P(a) &= \mathsf{F}_P^0(a) = \mathsf{G}(a)[\mathsf{G}] = (a_1 \wedge \bar{b}_1)[\mathsf{G}] = \big((a_2 \vee \bar{b}_1) \wedge \bot\big)[\mathsf{G}] = \bot \\
\mathsf{F}_P(b) &= \mathsf{F}_P^0(b) = \mathsf{G}(b)[\mathsf{G}] = (\bar{a} \wedge b_1)[\mathsf{G}] = \top \wedge \top = \top \\
\mathsf{F}_P(c) &= \mathsf{F}_P^0(c) = \mathsf{G}(c)[\mathsf{G}] = (\bar{x} \wedge c_1)[\mathsf{G}] = (\bar{x} \wedge \top) = \bar{x}
\end{aligned}$$

which is the resulting Skolem set.

8.3 Representation and Size of Skolem Functions

The representation of Skolem functions has a big impact on their size. Representing Skolem functions as a CNF using only the universal variables occurring earlier in the prefix, may result in Skolem functions that are exponential in the size of the QRAT proof. However, one can construct Skolem functions from a QRAT proof that are in worst case polynomial in the size of the proof (and linear in practice).

Example 31 Consider the true QBF expressing $y_i := \mathrm{XOR}(x_0, \ldots, x_i)$:

$$\forall x_0..x_n \exists y_1..y_n.(x_0 \vee x_1 \vee \bar{y}_1) \wedge (x_0 \vee \bar{x}_1 \vee y_1) \wedge (\bar{x}_0 \vee x_1 \vee y_1) \wedge (\bar{x}_0 \vee\ \bar{x}_1 \vee \bar{y}_1) \wedge \bigwedge_{i \in \{2..n\}} (y_{i-1} \vee x_i \vee \bar{y}_i) \wedge (y_{i-1} \vee \bar{x}_i \vee y_i) \wedge (\bar{y}_{i-1} \vee x_i \vee y_i) \wedge (\bar{y}_{i-1} \vee \bar{x}_i \vee \bar{y}_i)$$

All clauses in this formula can be removed by QRATE (in reverse order as shown above), resulting in a QRAT proof that is linear in the size of the formula. Expressing $y_n := \mathrm{XOR}(x_0, \ldots, x_n)$ using only universal variables requires a CNF formula of 2^{n+1} clauses. Our procedure will produce Skolem functions which are linear in the size of the QRAT proof by reusing the Skolem functions of variables occurring earlier in the prefix: $\mathsf{F}(y_1) := \mathrm{XOR}(x_0, x_1)$ and $\mathsf{F}(y_i) := \mathrm{XOR}(x_i, \mathsf{F}(y_{i-1}))$ with $i \in \{2..n\}$.

Given a QRAT proof P with k being the length of the largest clause in P, the size of the Skolem functions produced by our method is $\mathcal{O}(k|P|^2)$. Consider the pseudo-code in Figure 4. In the worst case, all lines in P are deletion steps forcing the Skolem functions to be updated $|P|$ times. The size of the update depends on whether the check on line aS5 succeeds. We express the update using and-inverter-graphs (AIGs). If it succeeds, the size of the update is $\mathcal{O}(k)$ gates. Otherwise, the size of the update is the size of ψ which is $\mathcal{O}(k|P|)$ gates.

8.4 Implementation, Optimization and Validation

We enhanced our QRAT checking tool, called QRATtrim,[1] with Skolem function extraction capabilities. The Skolem functions are emitted as a propositional formula in DIMACS format or as and-inverter-graphs (AIGs) in AIGER format. This section describes our implementation, optimizations and validation of Skolem functions.

The outer formula computed in line `a58` of the *addSkolem* procedure (Figure 4) is typically much larger than necessary and consequently makes Skolem functions larger than necessary. In order to produce smaller Skolem functions, we implemented the following optimization (using the notation in *addSkolem*): For all $D \in \psi$ with $\bar{l} \in D$, we compute the outer resolvent $R = \mathsf{OC}(\Pi, D, \bar{l}) \cup (C \setminus \{l\})$. We check whether R is an asymmetric tautology with respect to ψ and store all $\mathsf{OC}(\Pi, D, \bar{l})$ for which the corresponding R is *not* an asymmetric tautology. The alternative outer formula becomes the conjunction of these $\mathsf{OC}(\Pi, D, \bar{l})$ together with the negation of $C \setminus \{l\}$.

The presented *finishSkolem* assigns all Skolem functions $\mathsf{G}(\mathsf{L}(x)) := \top$. However, for some variables, $\mathsf{G}(\mathsf{L}(x)) := \bot$ is much more effective. Note, that Thm. 29 allows to pick an arbitrary constant. We observed that the best truth value for final Skolem functions $\mathsf{G}(\mathsf{L}(x))$ is based on the polarity of a literal used as pivot for a QRAT check. For some variables x (typically a few hundred), there are QRAT checks with literal x as a pivot, but no QRAT checks with literal $\bar{x}$ as pivot (or the other way around). By assigning $\mathsf{G}(\mathsf{L}(x)) := \top$ (or $\mathsf{G}(\mathsf{L}(x)) := \bot$, respectively), and applying simplification, we obtain the Skolem functions $\mathsf{F}_P(x) \equiv \top$ (or $\mathsf{F}_P(x) \equiv \bot$, respectively).

Validating a set of Skolem functions consists of two checks. If both checks succeeds, the set of Skolem functions is valid. Let F be a set of Skolem functions for a QBF $\Pi.\psi$. The first check consists of substituting the existential variables in ψ by all the Skolem functions in F. The resulting formula is negated and checked by a SAT solver to be unsatisfiable.

The second check uses the AIG representation of the Skolem functions and Π to check that no input gate g_i (universal variable) influences the truth value of output gate g_o (existential variable) with $g_i >_\Pi g_o$. So no universal variable influences the truth of an inner-more existential variable.

Apart from implementing a tool that extracts Skolem functions from a QRAT proof, we also implemented a tool cheskol that checks whether the Skolem functions are correct. A tool, called CertCheck [41], has the same functionality but uses a more strict check for the second part, i.e., whether the truth of no existential variable x depends on the truth value of any variable (also existential) inner to x. This check is too restrictive to validate our Skolem functions.

Example 32 Consider the formula $\exists a \forall b \exists c.(a \vee b \vee c) \wedge (\bar{a} \vee \bar{c})$. A possible QRAT proof removes first $(a \vee b \vee c)$ on pivot c and afterwards $(\bar{a} \vee \bar{c})$ on pivot $\bar{a}$. The latter is allowed because the prefix collapses to $\exists a, c$ after removing $(a \vee b \vee c)$. Our procedure for extracting Skolem functions gives $\mathsf{F}_P(a) = \neg \mathsf{F}_P(c)$ and $\mathsf{F}_P(c) = \top$ (depending on which optimizations are used). Although the Skolem function for a depends on the Skolem function for c which is inner to a, the Skolem functions are correct because the Skolem function of a does not depend on a universal variable inner to a.

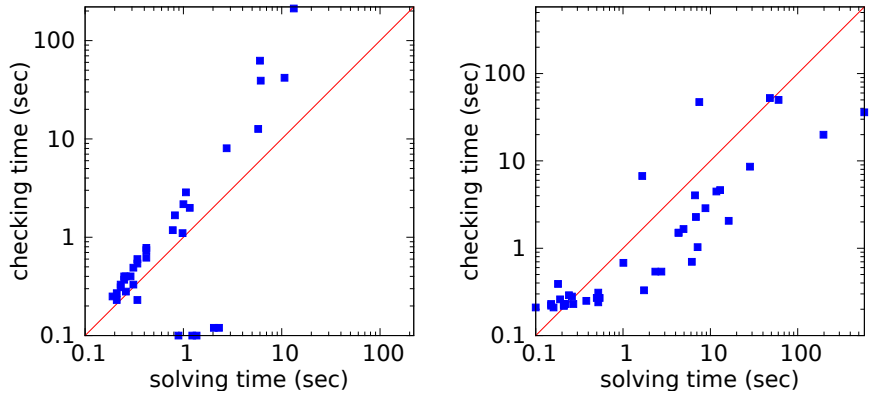

Fig. 6 Solving times versus checking times for true and false formulas.

Our validation tool called cheskol uses the less restrictive dependency check and emits the result of substitution and negation as a formula in DIMACS format (after Tseitin encoding), the typical input format for SAT solvers.

9 Evaluation

The most recent version v037 of the publicly available preprocessor bloqqer[2] directly solves 37 true instances and 41 false instances of the QBF Eval 2012 benchmark set, on which the CNF tracks of QBF Gallery 2013 and QBF Gallery 2014 are based. Enabling QRAT proof tracing did not decrease the number of solved formulas. All proofs of these 78 formulas could be checked with our QRAT checker QRATtrim.[1] Figure 6 compares solving times and checking times. on average, checking takes twice as long as solving for satisfiable formulas. On the other, for unsatisfiable formulas, checking is considerably faster than solving.

Table 5 shows results for our Skolem function extraction tool, i.e., a modified version of QRATtrim. We converted our certificates to the QBC format allowing for size comparison with certificates produced by other tools. Note that the extracted Skolem functions are typically smaller than the QRAT proofs. We validated our Skolem functions using the tool cheskol[1] which checks the dependencies and computes whether the Skolem functions imply the formula using the SAT solver lingeling [9].

Janota et al. [28] restricted bloqqer such that it is able to produce resolution (RES) proofs. However, several preprocessing techniques are not supported by that approach. As a consequence, their modified version of bloqqer solves less formulas (only 22 out of 37). If a formula cannot be solved by bloqqer they use the solver depQBF to compute a resolution proof of the simplified formula. They merge the certificate (resolution proof) obtained from depQBF with the partial certificate obtained from their bloqqer. The results of resolution-based approaches, our approach and some older tools [6, 32, 31] are shown in Table 6. The only tool that has comparable performance compared to our bloqqer+QRAT approach is bloqqer+RES+depQBF.

[2] `http://fmv.jku.at/bloqqer`

Table 5 Skolem functions extraction from QRAT proofs produced by bloqqer on QBF Eval 12.

formula	sol-t	ext-t	ch-t	tr-s	qbc-s
b11_PR_7_20	0.65	0.81	0.01	4,018.35	33.31
biu...b001-p010-OPF02-c08	0.92	1.75	0.32	3,138.93	137.56
c3_BMC_p1_k2	1.00	2.15	0.03	1,776.21	70.80
counter_8	0.33	0.44	0.08	265.87	36.21
itc-b13-fixpoint-8	11.29	75.71	351.57	123,077.00	25,286.80
k_branch_n-10	2.98	16.28	1.15	4,619.94	605.10
k_branch_n-14	6.32	82.31	1.99	15,349.60	968.72
k_branch_n-16	6.45	136.06	3.86	12,914.60	1,292.48
k_branch_n-20	13.83	442.66	8.08	20,131.90	1,951.84
k_branch_n-7	1.40	3.75	0.27	2,222.69	191.96
k_d4_n-10	0.88	3.20	19.80	5,591.46	2,019.67
k_d4_n-11	1.00	4.01	27.38	6,706.69	2,316.07
k_d4_n-14	1.40	7.39	47.92	10,261.00	3,384.55
k_d4_n-15	1.51	8.76	59.30	11,503.70	3,711.80
k_d4_n-20	2.14	18.50	78.91	16,441.90	4,898.16
k_d4_n-21	2.31	22.06	116.35	20,660.20	5,709.78
k_dum_n-10	0.26	0.44	0.04	258.61	47.53
k_dum_n-11	0.26	0.44	0.05	299.73	62.98
k_dum_n-12	0.20	0.52	0.05	330.77	64.57
k_dum_n-16	0.33	0.70	0.09	455.66	88.54
k_dum_n-20	0.33	0.88	0.15	613.37	134.16
k_dum_n-21	0.36	1.04	0.40	767.64	234.31
lights3_021_1_022	0.26	0.44	0.09	252.41	53.43
lights3_021_1_033	0.15	0.43	0.08	387.24	46.00
lights3_035_1_059	0.36	0.77	0.22	638.58	110.29
rankfunc0_unsigned_64	1.16	7.59	26.78	3,243.67	4,984.67
rankfunc16_unsigned_16	0.53	1.47	0.82	782.88	341.07
rankfunc24_signed_32	0.59	1.86	3.03	1,053.36	901.13
rankfunc27_unsigned_32	0.36	0.81	1.62	1,445.79	568.83
rankfunc52_signed_64	1.40	9.33	138.77	3,588.83	5,052.10
s3330_d2_s	5.82	19.22	5.66	122,569.00	797.96
stmt137_903_911	0.30	0.59	0.26	1,036.93	111.89
stmt1_629_630	0.54	1.17	0.45	1,670.66	186.70
stmt17_99_98	0.90	2.62	1.02	3,387.52	372.50
stmt27_584_603	0.32	0.59	0.28	974.67	131.92
stmt27_946_955	0.30	0.58	0.27	976.93	112.21
stmt41_118_131	0.25	0.49	0.47	773.02	165.15

sol-t/ext-t/ch-t: solving/extraction/checking time (sec)
tr-s/qbc-s: size of QRAT file/qbc file (kilobyte)

If all preprocessing techniques are turned on, the average size of the Skolem functions produced by bloqqer+QRAT is in general larger than those produced by bloqqer+RES+depQBF. Recall that bloqqer+RES+depQBF does not support several preprocessing techniques. Consequently, unsupported techniques such as covered clause elimination (QCCE) [17] are turned off. Although bloqqer+QRAT supports QCCE, using it has a negative impact on the size of Skolem functions. Turning QCCE off reduces the size of Skolem functions significantly — at the cost of solving three formulas less (itc-b13-fixpoint-8, k_branch_n-20, s3330_d2_s). The left scatter plot of Figure 7 compares the size of Skolem functions obtained from proof traces generated with and without QCCE. Note that we converted the AIG representations

Table 6 Comparison of solving/checking times (some formula names are abbreviated).

formula	bloqqer QRAT	bloqqer RES	bloqqer depQBF	depQBF	ebdd	squolem	sKizzo
b11_PR_7_20	0.7/0.0	-/-	0.1/0.2	0.0/0.0	-/-	4.9/0.4	-/-
biu..b001-p010-..	0.9/0.3	-/-	-/-	-/-	-/-	-/-	-/-
c3_BMC_p1_k2	1.0/0.0	0.2/1.3	0.2/1.3	0.1/0.1	-/-	486/31	3.8/0.6
counter_8	0.3/0.1	-/-	0.1/0.7	0.0/0.1	12/831	-/-	4.5/0.9
itc-b13-fixpoint-8	11/351	3.2/16.0	3.2/16.0	-/-	-/-	-/-	-/-
k_branch_n-10	3.0/1.1	-/-	2.0/	-/-	-/-	-/-	-/-
k_branch_n-14	6.3/2.0	-/-	-/-	-/-	-/-	-/-	-/-
k_branch_n-16	6.5/3.9	4.6/1.6	4.6/1.6	-/-	-/-	-/-	-/-
k_branch_n-20	13.8/8.1	-/-	-/-	-/-	-/-	-/-	-/-
k_branch_n-7	1.4/0.3	-/-	0.6/18.8	-/-	-/-	-/-	-/-
k_d4_n-10	0.9/19.8	-/-	0.3/28.6	-/-	30.1/-	3.3/0.9	13.0/9.0
k_d4_n-11	1.0/27.4	-/-	0.4/295	-/-	36.0/-	3.3/0.6	24/19.5
k_d4_n-14	1.4/47.9	-/-	0.5/	-/-	49.2/-	3.6/0.9	578/0.2
k_d4_n-15	1.5/59.3	-/-	-/-	-/-	52.6/-	3.6/0.7	-/-
k_d4_n-20	2.1/78.9	-/-	-/-	-/-	72.1/-	4.2/1.1	-/-
k_d4_n-21	2.3/116	-/-	-/-	-/-	75.0/-	4.4/1.0	-/-
k_dum_n-10	0.3/0.0	0.1/1.1	0.1/1.1	-/-	1.6/1.1	2.9/0.3	4.2/0.6
k_dum_n-11	0.3/0.1	0.1/1.4	0.1/1.4	-/-	1.6/1.3	3.3/0.7	4.0/0.6
k_dum_n-12	0.2/0.1	0.1/1.5	0.1/1.5	-/-	1.6/1.4	3.4/0.6	0.5/2.4
k_dum_n-16	0.3/0.1	0.1/1.3	0.1/1.3	-/-	1.7/1.7	3.0/0.7	4.2/0.7
k_dum_n-20	0.3/0.1	0.2/1.3	0.2/1.3	-/-	1.7/1.8	2.5/0.6	4.1/0.8
k_dum_n-21	0.4/0.4	0.2/0.5	0.2/0.5	-/-	1.6/2.0	3.0/0.4	4.0/0.9
lights3_021_1_022	0.3/0.1	0.1/0.8	0.1/0.8	-/-	30.3/-	2.1/0.3	4.4/0.7
lights3_021_1_033	0.1/0.1	0.1/1.2	0.1/1.2	-/-	29.9/-	2.9/0.3	4.7/0.4
lights3_035_1_059	0.4/0.2	0.1/0.6	0.1/0.6	-/-	-/-	3.1/0.6	6.2/1.0
rf0_unsigned_64	1.2/27	1.6/1.1	1.6/1.1	-/-	-/-	-/-	4.3/3.9
rf16_unsigned_16	0.5/0.8	0.3/0.5	0.3/0.5	-/-	-/-	-/-	4.1/2.2
rf24_signed_32	0.6/3.0	0.5/1.3	0.5/1.3	-/-	-/-	-/-	-/-
rf27_unsigned_32	0.4/1.6	0.3/1.3	0.3/1.3	-/-	-/-	-/-	1.5/2.5
rf52_signed_64	1.4/138	1.6/1.1	1.6/1.1	-/-	-/-	-/-	-/-
s3330_d2_s	5.8/5.7	-/-	1.6/0.2	-/-	-/-	-/-	-/-
stmt137_903_911	0.3/0.3	0.1/0.0	0.1/0.0	-/-	-/-	-/-	0.3/0.2
stmt1_629_630	0.5/0.5	0.1/0.1	0.1/0.1	-/-	-/-	-/-	17.4/0.5
stmt17_99_98	0.9/1.0	-/-	-/-	-/-	-/-	-/-	2.9/1.7
stmt27_584_603	0.3/0.3	0.1/0.1	0.1/0.1	-/-	-/-	-/-	0.3/0.2
stmt27_946_955	0.3/0.3	0.1/0.1	0.1/0.1	-/-	-/-	-/-	0.3/0.2
stmt41_118_131	0.2/0.5	0.1/0.1	0.1/0.1	-/-	-/-	-/-	0.3/0.5

into QBC certificates to have the same file format as bloqqer+RES+depQBF. Finally, the right scatter plot of Figure 7 compares the sizes of the Skolem functions produced by bloqqer+QRAT (without QCCE) and bloqqer+RES+depQBF. It illustrates that the Skolem functions extracted by bloqqer+QRAT are in general smaller than those produced by bloqqer+RES+depQBF, especially for the harder benchmarks.

10 Conclusion

We presented a proof system which captures recent preprocessing and solving techniques for QBF in a uniform manner. Based on *asymmetric tautologies*, the proof

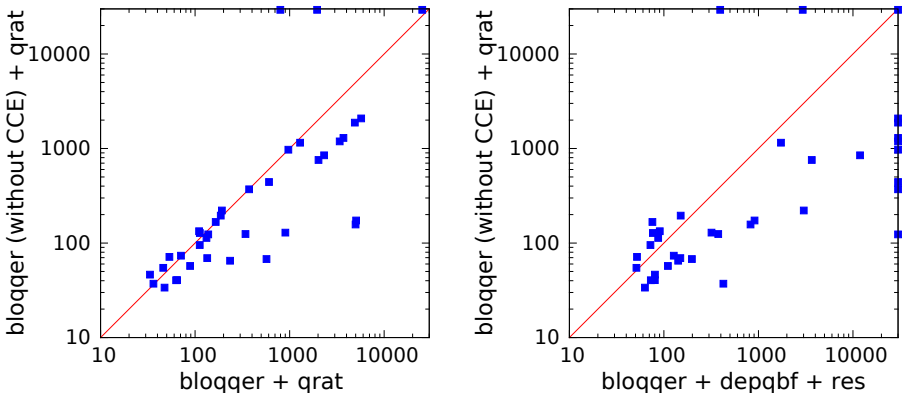

Fig. 7 Comparison of certificate sizes (KB). Unsolved formulas are shown with size 30,000 KB.

system consists only of four simple rules. We showed how state-of-the-art preprocessing techniques can be represented within this proof system. Our rules QRATE, QRATU, and QRATA may be applied as preprocessing rules themselves similar as QBCE and we plan to integrate them in our preprocessor. We deal with all the challenges regarding certificates and preprocessing for QBF that were recently listed [28], namely: can we (1) produce polynomially-verifiable certificates for true QBFs in the context of preprocessing, (2) narrow the performance gap between solving with and without certificate generation, (3) develop methods to deal with universal expansion and other techniques; and (4) extract Skolem functions from QRAT proofs.

This work opens several streams of future work. First, we extract only Skolem functions from formulas which are solved by the preprocessor. In some applications, also the extraction of Herbrand function is of interest. In contrast to the Skolem functions, for which the deleted clauses of a proof are considered, the newly introduced clauses have to be taken into account for the generation of the Herbrand functions.

Second, if the preprocessor cannot solve a formula, then a complete solver has to be consolidated. Such a solver probably produces proofs in a different proof system. For understanding how proofs of different proof systems can be translated or combined, it has to be investigated how our new QRAT-based proof systems compares to other proof systems like Q-resolution (the clause and its dual cube variant), sequent calculi for QBF, or proof systems which provide explicit rules for universal expansion [26]. Based on these results it will become possible to integrate orthogonal solving techniques as it is done in [39] and to obtain Skolem and Herbrand functions for the original input QBF.

A third direction of future work concerns the integration of advanced dependency schemes into our new proof system (like in [43] for Q-resolution) including other variants of expansion [12]. Further, it would be interesting to investigate how QRAT can be used for symmetry breaking as it is successfully done in SAT [21].

References

1. Ayari A, Basin DA (2002) QUBOS: Deciding Quantified Boolean Logic Using Propositional Satisfiability Solvers. In: Proc. of the 5th Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD 2002), Springer, LNCS, vol 2517, pp 187–201
2. Balabanov V, Jiang JR (2011) Resolution Proofs and Skolem Functions in QBF Evaluation and Applications. In: Proc. of the 23rd Conf. on Computer Aided Verification (CAV 2011), Springer, LNCS, vol 6806, pp 149–164
3. Balabanov V, Jiang JR (2012) Unified QBF certification and its applications. Formal Methods in System Design 41(1):45–65
4. Balabanov V, Jiang JR, Janota M, Widl M (2015) Efficient extraction of QBF (counter)models from long-distance resolution proofs. In: Proc. of the 29th Conf. on Artificial Intelligence, (AAAI 2015), AAAI Press, pp 3694–3701
5. Benedetti M (2005) Extracting Certificates from Quantified Boolean Formulas. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005), Professional Book Center, pp 47–53
6. Benedetti M (2005) sKizzo: A Suite to Evaluate and Certify QBFs. In: Proc. of the 20th Int. Conf. on Automated Deduction (CADE 2005), LNCS, vol 3632, Springer, pp 369–376
7. Benedetti M, Mangassarian H (2008) QBF-Based Formal Verification: Experience and Perspectives. Journal on Satisfiability, Boolean Modeling and Computation 5(1-4):133–191
8. Biere A (2005) Resolve and expand. In: Proc. of the 7th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT 2004), Springer, LNCS, vol 3542, pp 59–70
9. Biere A (2013) Lingeling, Plingeling and Treengeling entering the SAT competition 2013. In: Proc. of SAT Competition 2013
10. Biere A, Lonsing F, Seidl M (2011) Blocked clause elimination for QBF. In: Proc. of the 23th Int. Conf. on Automated Deduction (CADE 2011), Springer, LNCS, vol 6803, pp 101–115
11. Bloem R, Könighofer R, Seidl M (2014) SAT-Based Synthesis Methods for Safety Specs. In: Proc. of the 15th Int. Conf. on Verification, Model Checking, and Abstract Interpretation (VMCAI 2014), Springer, LNCS, vol 8318, pp 1–20
12. Bubeck U, Kleine Büning H (2007) Bounded universal expansion for preprocessing QBF. In: Proc. of the 10th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT 2007), Springer, LNCS, vol 4501, pp 244–257
13. Cadoli M, Giovanardi A, Schaerf M (1998) An algorithm to evaluate quantified boolean formulae. In: Proc. of the 15th National Conf. on Artificial Intelligence and 10th Innovative Applications of Artificial Intelligence Conf., (AAAI 98/IAAI 98), AAAI Press / The MIT Press, pp 262–267
14. Davis M, Putnam H (1960) A computing procedure for quantification theory. Journal of the ACM 7(3):201–215
15. Giunchiglia E, Marin P, Narizzano M (2010) sQueezeBF: An Effective Preprocessor for QBFs Based on Equivalence Reasoning. In: Proc. of the 13th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT 2010), Springer,

LNCS, vol 6175, pp 85–98
16. Goldberg EI, Novikov Y (2003) Verification of proofs of unsatisfiability for CNF formulas. In: Proc. of the Design, Automation and Test in Europe Conf. and Exposition (DATE 2003), IEEE, pp 10,886–10,891
17. Heule M, Järvisalo M, Biere A (2013) Covered clause elimination. In: Proc. of the 17th Int. Conf. on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2013), EasyChair Proc. in Computing, vol 13, pp 41–46
18. Heule M, Seidl M, Biere A (2014) Efficient extraction of Skolem functions from QRAT proofs. In: Proc. of the 17th Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD 2014), IEEE, pp 107–114
19. Heule M, Seidl M, Biere A (2014) A unified proof system for QBF preprocessing. In: Proc. of the 7th Int. Joint Conf. on Automated Reasoning (IJCAR 2014), Springer, LNCS, vol 8562, pp 91–106
20. Heule M, Järvisalo M, Lonsing F, Seidl M, Biere A (2015) Clause elimination for SAT and QSAT. Journal on Artificial Intelligence Research (JAIR) 53:127–168
21. Heule M, Jr WAH, Wetzler N (2015) Expressing symmetry breaking in DRAT proofs. In: Proc. of the 25th Int. Conf. on Automated Deduction (CADE 2015), Springer, LNCS, vol 9195, pp 591–606
22. Heule M, Seidl M, Biere A (2015) Blocked literals are universal. In: Proc. of the 7th Int. Symposium on NASA Formal Methods (NFM 2015), Springer, LNCS, vol 9058, pp 436–442
23. Heule MJH, Järvisalo M, Biere A (2010) Clause elimination procedures for CNF formulas. In: Proc. 14th Int. Conf. on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2010), Springer, LNCS, vol 6397, pp 357–371
24. Heule MJH, Hunt WA, Wetzler N (2013) Verifying refutations with extended resolution. In: Proc. of the 24th Int. Conf. on Automated Deduction (CADE 2013), Springer, LNAI, vol 7898, pp 345–359
25. Heule MJH, Hunt, Jr WA, Wetzler N (2013) Trimming while checking clausal proofs. In: Proc. of the 16th Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD 2013), IEEE, pp 181–188
26. Janota M, Marques-Silva J (2015) Expansion-based QBF solving versus Q-resolution. Theor Comput Sci 577:25–42
27. Janota M, Grigore R, Marques-Silva J (2012) On Checking of Skolem-based Models of QBF. In: Proc. of the Int. Workshop on Experimental Evaluation of Algorithms for solving problems with combinatorial explosion
28. Janota M, Grigore R, Marques-Silva J (2013) On QBF Proofs and Preprocessing. In: Proc. of the 17th Int. Conf. on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2013), Springer, LNCS, vol 8312, pp 473–489
29. Järvisalo M, Heule MJH, Biere A (2012) Inprocessing rules. In: Proc. of the 6th Int. Joint Conf. on Automated Reasoning (IJCAR 2012), Springer, LNCS, vol 7364, pp 355–370
30. Jordan C, Seidl M (2014) The QBF Gallery 2014. `http://qbf.satisfiability.org/gallery/`
31. Jussila T, Sinz C, Biere A (2006) Extended resolution proofs for symbolic sat solving with quantification. In: Proc. of the 9th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT 2006), Springer, vol 4121, pp 54–60

32. Jussila T, Biere A, Sinz C, Kröning D, Wintersteiger CM (2007) A first step towards a unified proof checker for QBF. In: Proc. of the 7th Int. Conf. Theory and Applications of Satisfiability Testing (SAT 2007), LNCS, vol 4501, Springer, pp 201–214
33. Kleine Büning H, Karpinski M, Flögel A (1995) Resolution for quantified boolean formulas. Information and Computation 117(1):12–18
34. Kleine Büning H, Subramani K, Zhao X (2007) Boolean Functions as Models for Quantified Boolean Formulas. Journal of Automated Reasoning 39(1):49–75
35. Kleine Büning H, Subramani K, Zhao X (2007) Boolean functions as models for quantified boolean formulas. Journal of Automated Reasoning 39(1)
36. Könighofer R, Seidl M (2014) Partial witnesses from preprocessed quantified boolean formulas. In: Proc. of Design, Automation & Test in Europe Conf. & Exhibition, (DATE 2014), IEEE, pp 1–6
37. Lonsing F, Biere A (2010) DepQBF: A Dependency-Aware QBF Solver. Journal on Satisfiability, Boolean Modeling and Computation 7(2-3):71–76
38. Lonsing F, Seidl M, Van Gelder A (2013) The QBF Gallery: Behind the Scenes. CoRR abs/1508.01045, URL `http://arxiv.org/abs/1508.01045`
39. Lonsing F, Bacchus F, Biere A, Egly U, Seidl M (2015) Enhancing search-based QBF solving by dynamic blocked clause elimination. In: Proc. of the 20th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2015), Springer, LNCS, vol 9450, pp 418–433
40. Narizzano M, Peschiera C, Pulina L, Tacchella A (2009) Evaluating and certifying qbfs: A comparison of state-of-the-art tools. AI Com 22(4):191–210
41. Niemetz A, Preiner M, Lonsing F, Seidl M, Biere A (2012) Resolution-based certificate extraction for QBF - (tool presentation). In: Proc. of the 15th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT 2012), Springer, LNCS, vol 7317, pp 430–435
42. Samulowitz H, Davies J, Bacchus F (2006) Preprocessing QBF. In: Proc. of the 12th Int. Conf. on Principles and Practice of Constraint Programming (CP 2006), Springer, LNCS, vol 4204, pp 514–529
43. Slivovsky F, Szeider S (2016) Soundness of Q-resolution with dependency schemes. Theor Comput Sci 612:83–101
44. Van Gelder A (2011) Variable Independence and Resolution Paths for Quantified Boolean Formulas. In: Proc. of the 17th Int. Conf. on Principles and Practice of Constraint Programming (CP 2011), Springer, LNCS, vol 6876, pp 789–803
45. Van Gelder A (2013) Certificate Extraction from Variable-Elimination QBF Preprocessors. In: Proc. of the 1st Int. Workshop on Quantified Boolean Formulas (QBF 2013), pp 35–39
46. Wetzler N, Heule MJH, Hunt WA (2014) DRAT-trim: Efficient Checking and Trimming Using Expressive Clausal Proofs. In: Proc. of the 17th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT 2014), LNCS, vol 8561, Springer, pp 422–429
47. Yu Y, Malik S (2005) Validating the result of a quantified boolean formula (QBF) solver: theory and practice. In: Proc. of the Conf. on Asia South Pacific Design Automation, (ASP-DAC 2005), ACM Press, pp 1047–1051