Efficient Extraction of Skolem Functions from QRAT Proofs

Marijn J.H. Heule
The University of Texas at Austin
marijn@cs.utexas.edu

Martina Seidl and Armin Biere
Johannes Kepler University, Linz
{martina.seidl, biere}@jku.at

Abstract—Many synthesis problems can be solved by formulating them as a quantified Boolean formula (QBF). For such problems, a mere true/false answer is often not enough. Instead, expressing the answer in terms of Skolem functions reflecting the quantifier dependencies of the variables is required. Several approaches have been presented to extract such functions from term-resolution proofs. However, not all solvers and preprocessors are able to produce term-resolution proofs, especially when universal expansion is involved. In previous work, we developed the QRAT proof system consisting of three simple rules which allowed us to overcome this issue and to equip modern expansion-based tools like the preprocessor bloqqer with proof tracing. In this paper, we show how to extract Skolem functions from QRAT proofs. We present a general extraction tool and compare its performance to similar resolution-based tools. We show that the Skolem functions extracted from QRAT proofs are smaller than those produced by alternative approaches making our method in particular useful for synthesis applications.

I. INTRODUCTION

Synthesis problems, which aim at the automatic derivation of an implementation from a given specification, typically ask whether for all possible inputs by the environment there exists a strategy for the system such that certain properties like safety hold. If the answer is positive, then this strategy provides the means to synthesize the required implementation of the system which by construction obeys the given specification. Therefore, a tool solving such a synthesis problem should not only provide a yes/no answer, but also a strategy of the specification in the case of realizability.

A natural way to encode synthesis problems is offered by quantified Boolean formulas (QBFs) [1], [2], which extend propositional logic by quantifiers over propositional variables [3]. The QBF formalism provides a convenient framework for modeling finite two-player games [4] which is reflected by the popular game-based view on the semantics of QBFs. Here, the evaluation of a QBF is described as a game between the existential player who owns the existential variables and the universal player who owns the universal variables of the formula. The existential player wants to satisfy the formula, while the universal player wants to falsify the formula. The moves are assignments to the variables, where the order of the variables in the quantifier prefix has to be respected. If the formula is satisfiable, then there exists a strategy for the existential player to always win the game and respectively, there is a strategy for the universal player to win the game if the formula is unsatisfiable.

By using existential and universal quantification, QBFs allow for exponentially more succinct encodings than propositional logic, with the consequence that the satisfiability problem of QBF is PSPACE-complete [3]. Therefore, the field of application of QBF ranges from efficient encodings of verification problems like model checking tasks to planning (see [2] for a survey on QBF applications).

All these applications have in common that they require models if the formula encoding the application problem is satisfiable. Whereas in SAT a model is given by a variable assignment, in QBF the situation is more complicated [5]. A model is an assignment tree giving a winning strategy to the existential player. In practice, a more compact representation than given by an assignment tree is required. To this end, the concept of Skolem functions, which for example are used in first-order logic to eliminate existential quantifiers, is transferred to the context of QBF [5]. A Skolem function for an existential $x$ variable is a Boolean function over the universal variables preceding $x$ in the quantifier prefix.

Today, it is known how Skolem function extraction can be realized in the context of DPLL-based QBF solving [6], which is the solving paradigm realized by most state-of-the-art solvers. Basically, the possibility of search-based approaches is exploited to generate term-resolution proofs from which winning strategies for the existential player can be generated [7], [8]. Unfortunately, this approach does not apply to expansion-based techniques [9], [10], which have been shown to be extremely powerful if realized in preprocessors. Until recently, it was not possible to generate any proofs when preprocessing is applied, because it is an open question if expansion can be simulated by resolution [11]. However, when preprocessing is restricted or even omitted, the solving performance drastically decreases. To overcome this issue, we presented the QRAT proof system [12] which is able to capture all state-of-the-art preprocessing techniques by a few simple rules. We could further show that emitting QRAT proofs causes only small overhead and that the validation of QRAT proofs is computationally cheap. Hence we were able to provide a tool to certify the result of a preprocessor efficiently.

In this paper, we go one step further and show how to extract Skolem functions from QRAT proofs of satisfiability. With this work we solve one important issue hindering the practical application of QBF. Moreover, the size of the Skolem functions that we extract is smaller than in other approaches.
In the following, we first recapitulate QBF basics in Section II and review literature in Section III. Then we introduce the QRAT proof system in Section IV which is the basis for the Skolem function extraction approach presented in Section V. Implementation details are discussed in Section VI, followed by an experimental evaluation in Section VII. Finally, we conclude this paper with an outlook to future work.

II. PRELIMINARIES

The language of QBF extends the language of propositional logic by existential and universal quantifiers over the propositional variables. As usual, we assume a QBF to be in prenex conjunctive normal form (PCNF). A QBF in PCNF has the structure $\Pi.\psi$ where the prefix $\Pi$ has the form $Q_1X_1Q_2X_2\ldots Q_nX_n$ with disjoint variable sets $X_i$ and $Q_i \in \{\forall, \exists\}$. The formula $\psi$ is a propositional formula in conjunctive normal form, i.e., a conjunction of clauses. A clause is a disjunction of literals and a literal is either a variable (positive literal) or a negated variable (negative literal). The variable of a literal is denoted by $\text{var}(l)$ where $\text{var}(l) = x$ if $l = x$ or $l = \bar{x}$. The negation of a literal $l$ is denoted by $\bar{l}$. The quantifier $Q(\Pi, l)$ of a literal $l$ is $Q_i$ if $\text{var}(l) \in X_i$. Let $Q(\Pi, l) = Q_i$ and $Q(\Pi, k) = Q_j$, then $l \leq_k \bar{k}$ if $i \leq j$. We sometimes write formulas in CNF as sets of clauses and clauses as sets of literals. We consider only closed QBFs, so $\psi$ contains only variables which occur in the prefix. The variables occurring in the prefix of $\phi$ are given by $\varphi(\phi)$. The subformula $\psi_i$ consisting of all clauses of matrix $\psi$ with literal $l$ is defined by $\psi_i = \{C_{l} \mid l \in C, C \in \psi\}$. By $\top$ and $\bot$ we denote the truth constants true and false. QBFs are interpreted as follows: a QBF $\forall\Pi.\psi$ is false iff $\Pi.\psi[x/\top]$ or $\Pi.\psi[x/\bot]$ is false where $\Pi.\psi[x/l]$ is the QBF obtained by replacing all occurrences of variable $x$ by $l$. Respectively, a QBF $\exists\Pi.\psi$ is false iff both $\Pi.\psi[x/\top]$ and $\Pi.\psi[x/\bot]$ are false. If the matrix $\psi$ of a QBF $\phi$ contains the empty clause after eliminating the truth constants according to standard rules, then $\phi$ is false. Accordingly, if the matrix $\psi$ of QBF $\phi$ is empty, then $\phi$ is true. Two QBFs $\phi_1$ and $\phi_2$ are satisfiability equivalent (written as $\phi_1 \sim_2 \phi_2$) iff they have the same truth value. Two QBFs $\phi_1$ and $\phi_2$ are logically equivalent (written as $\phi_1 \approx \phi_2$) if they have the same set of (counter) models.

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. Hence, QBF models are either expressed in form of subtrees of assignment trees or as Skolem functions.

**Definition 1.** Let $x$ be an existential variable of $\phi = \Pi.\psi$ and let $y_1, \ldots, y_n$ be all universals of $\phi$ with $y_i \leq \Pi x$. Then a propositional formula $f_x(y_1, \ldots, y_n)$ is a Skolem function for $x$, and called valid iff $\phi[x/f_x] \sim \phi$. A set of Skolem functions $F$ which contains exactly one Skolem function for every existential $x$ is called a Skolem set. It is called valid iff it only contains valid Skolem functions.

Obviously, for any satisfiable QBF $\phi$, a valid Skolem set $F$ gives a strategy for the existential player to satisfy the formula. In the remainder of this paper, given a QBF $\Pi.\psi$ containing an existential variable $x$, the function $f_x(U)$ denotes a Skolem function for $x$ with the set of universal variables $U$ that are outer to $x$ in $\Pi$, as parameters.

To check that a Skolem set $F$ is valid, it is necessary to substitute in $\phi$ all existential variables by their corresponding Skolem functions in $F$ and check that the resulting propositional formula is valid. This can be done by a SAT solver. In practice, it also needs to be checked 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 [5].

We conclude the preliminary section by introducing the concept of asymmetric literal addition.

**Definition 2 (Asymmetric Literal Addition).** Given a QBF $\Pi.\psi$ and a clause $C$. The clause $\text{ALA}(\psi, C)$ is the unique clause obtained by repeatedly applying the extension rule $C := C \cup \{l\}$ if $\exists l_1, \ldots, l_k \in C$ and $(l_1 \lor \ldots \lor l_k \lor l) \in \psi$ called asymmetric literal addition to $C$ until fixpoint.

Asymmetric literal addition is indifferent with respect to the quantification type of the involved literals. Originally it was introduced for propositional logic in order to uniformly characterize preprocessing and inprocessing techniques [13]. It turned out that asymmetric literal addition is the basis for several powerful redundancy criteria which allow to safely add and delete clauses in propositional logic. As $\text{ALA}$ is model preserving, for any QBF $\phi = \Pi.\psi \land \{C\}$ holds that $\phi \approx \phi[C/C']$ where $C' = \text{ALA}(\psi \setminus \{C\}, C)$ [12].

III. RELATED WORK

The importance of Skolem function generation for true QBFs has been acknowledged to be a vital problem. Yet for a long time, only solvers internally working with Skolemization like Skizzo [14] and squlem [15] as well as the BDD-based solver ebdires [16] were able to produce Skolem functions. All three solvers are not maintained any more and to best of our knowledge no recent solver is built based on internal Skolemization. Instead, two solving paradigms have shown to be successful over the last years: most solvers implement a variant of the search-based DPLL algorithm [6] with clause and cube learning which is closely related to the techniques found in state-of-the-art SAT solvers. Alternatively, expansion-based systems [17], [9], [10] are developed which use variable elimination and universal expansion for simplifying a formula. The latter techniques have been shown to be extremely powerful when used as preprocessing steps where they are not applied until completion but where they just transform the formula such that it becomes easier to solve for search-based solvers.

For solvers and tools which are able to produce term-resolution proofs, the approaches presented by Balabanov

ISBN: 978-0-9835678-4-4. Copyright owned jointly by the authors and FMCAD Inc. 108
TABLE I
THE QRAT PROOF SYSTEM

<table>
<thead>
<tr>
<th>Rule</th>
<th>Preconditions</th>
<th>Postconditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N1)</td>
<td>$\Pi.\psi \xrightarrow{ATE(C)} \Pi.\psi \setminus {C}$</td>
<td>$C$ is an asymmetric tautology</td>
</tr>
<tr>
<td>(N2)</td>
<td>$\Pi.\psi \xrightarrow{ATA(C)} \Pi'.\psi \cup {C}$</td>
<td>$C$ is an asymmetric tautology</td>
</tr>
<tr>
<td>(E1)</td>
<td>$\Pi.\psi \xrightarrow{QRAT(U(C,l))} \Pi.\psi \setminus {C}$</td>
<td>$C \in \psi$, $Q(\Pi, l) = \exists$ $C$ has QRAT on $l$ w.r.t. $\psi$</td>
</tr>
<tr>
<td>(E2)</td>
<td>$\Pi.\psi \xrightarrow{QRAT(U(C,l))} \Pi'.\psi \cup {C}$</td>
<td>$C \notin \psi$, $Q(\Pi, l) = \exists$ $C$ has QRAT on $l$ w.r.t. $\psi$</td>
</tr>
<tr>
<td>(U1)</td>
<td>$\Pi.\psi \cup {C} \xrightarrow{EUR(U(C,l))} \Pi.\psi \cup {C} \setminus {l}$</td>
<td>$l \in C$, $Q(\Pi, l) = \forall, \bar{l} \notin C$, $C$ has QRAT on $l$ w.r.t. $\psi$</td>
</tr>
<tr>
<td>(U2)</td>
<td>$\Pi.\psi \cup {C} \xrightarrow{EUR(U(C,l))} \Pi.\psi \cup {C} \setminus {l}$</td>
<td>$l \in C$, $Q(\Pi, l) = \forall, \bar{l} \notin C$, $C$ has $EUR$ on $l$ w.r.t. $\psi$</td>
</tr>
</tbody>
</table>

and Jiang [7] and presented by Goultiaeva et al. [8] can be applied to extract strategies from the proofs. With these works it became possible to generate certificates for search-based solvers.

For expansion-based solvers and tools, however, the situation is different. As soon as universal expansion is involved in the solving process, it remains an open question if and how it can be translated to resolution. Therefore, it is not possible to produce resolution proofs for expansion-based systems [11], what was especially problematic if a formula is only solvable by the application of universal expansion. In previous work [18], we showed how to produce partial certificates for the variables of the outermost quantifier block, but here only single variable assignments are involved. Janota et al. [19] proposed to use only techniques that can be translated into resolution in order to bring certification and Skolem function extraction to state-of-the-art preprocessing. However, then the preprocessor loses a lot of its power.

To avoid any restriction of the applicable techniques when certification is required, we introduced the QRAT proof system [12] which is able to capture universal expansion as well as all state-of-the-art preprocessing techniques by three simple rules which can be checked easily. The obvious question is how to extract Skolem functions from such proofs which we answer in this paper.

IV. QRAT: QUANTIFIED RESOLUTION ASYMMETRIC TAUTOLOGIES

The QRAT proof system which we introduced in [12] is the first proof system for QBFs which captures all preprocessing techniques as well as expansion-based solving. The rules of the proof system are shown in Table II. The basic idea is to use syntactic redundancy criteria to add, remove, or modify clauses until the truth value of the formula is known. Soundness of the rules is shown in [12], completeness follows from the fact that the QRAT proof system simulates resolution. Please note that for the sake of readability, we work with a definition consisting of six rules in this paper instead of the more compact three rule variant of our QRAT proof format [12], where (N1)+(E1), (N2)+(E2), as well as (U1)+(U2) form the three rules. By splitting up the original three rules in six rules, we do not gain any additional expressiveness, but it allows us a more focused view on the problem of extracting Skolem functions. As we will see only those rules are relevant which delete clauses in a satisfiability equivalence preserving manner. This applies to (E1) only. Further, this allows us to present the rules irrelevant for the Skolem function extraction in an intuitive way by abstracting from the concrete technical details. For the complete formal definition of the proof system, we kindly refer to [12].

The rules (N1) and (N2) eliminate and respectively add asymmetric tautologies (AT). A clause $C$ is an asymmetric tautology w.r.t. a QBF $\Pi.\psi$ iff $\text{ALA}(\psi \setminus \{C\}, C)$ is a tautology. As the addition of asymmetric literals is model preserving, it holds that $\Pi.\psi \approx \Pi.\psi \cup C$ iff $\text{ALA}(\psi \setminus \{C\}, C)$ is a tautology. Hence, the application of (N1) and (N2) is model preserving [12].

Rule (E1) and (E2) apply the QRAT redundancy criterion which is defined below.

For some intuition about QRAT consider the following scenario. Let $\Pi.\psi$ be a QBF, $C$ a clause, and $l$ a literal in $C$. We are interested in the situation that for every assignment satisfying $\psi$ and falsifying $C$, it holds that all clauses $D_i \in \psi$ with $\bar{l} \in D_i$ are satisfied on literal $k \in D_i$ with $k \neq \bar{l}, k \leq \Pi. l$. As all clauses $D_i$ with $\bar{l} \in D_i$ are satisfied by at least two literals if $C$ is falsified by a variable assignment $\sigma$, the assignment can be modified by flipping the value of $l$ such that $C$ becomes satisfied while all $D_i$ stay satisfied. Hence, the addition of $C$ to $\phi$ preserves satisfiability and the deletion
of \( C \) to \( \phi \) preserves unsatisfiability. The reasoning for QRAT is similar and uses the following three definitions.

**Definition 3 (Outer Clause).** Let \( C \) be a clause occurring in QBF \( \Pi.\psi \). The outer clause of \( C \) on literal \( l \in C \), denoted by \( O(\Pi, C, l) \), is given by the clause \( \{ k | k \in C, k \leq l, k \neq l \} \).

**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. \( \Pi \), denoted by \( R(\Pi, C, D, l) \), is given by the clause \( O(\bar{l},C\setminus\{l\}) \) if \( Q(\Pi, l) = \forall \) and by \( O \cup C \) if \( Q(\Pi, l) = \exists \) assuming \( O = O(\Pi, D, \bar{l}) \).

**Definition 5 (Quantified Resolution Asymmetric Tautology (QRAT)).** Given a QBF \( \Pi.\psi \) and a clause \( C \). Then \( C \) has QRAT on literal \( l \in C \) with respect to \( \Pi.\psi \) iff it holds for all \( D \in \psi_l \) that \( ALA(\psi \setminus \{ C \}, R) \) is a tautology for the outer resolvent \( R = R(\Pi, C, D, l) \).

The rules (U1) and (U2) eliminate universal variable occurrences for which redundancy criteria ensure that on those the universal player will never be forced to use them to satisfy the clauses. In particular, if a clause \( C \) has QRAT on universal literal \( l \) w.r.t. to a QBF \( \phi = \Pi.\psi \) with \( C \in \psi \), then it can be shown that removing \( l \) from \( C \) preserves satisfiability. This rule subsumes universal pure literal elimination (i.e., literals occurring in one polarity), which is an indispensable rule for state-of-the-art QBF solvers. Finally, the rule (U2) allows the elimination of a universal literal by the means of extended universal reduction [12]. Universal reduction is part of the resolution calculus for QBFs. It removes a literal \( l \) from a non-tautological clause \( C \) iff \( C \) does not contain any existential literal occurring to the right of \( l \) in the prefix. From the game view this means that whenever the universal player has to assign \( l \), he can immediately falsify the clause, because there is no existential literal left allowing the existential player to satisfy the clause. The idea behind extended universal reduction goes in a similar direction, but here existential literals to the right of \( l \) in the prefix are allowed if they have certain properties. As these properties are irrelevant for the remainder of this paper, we refer the reader to [12] for the details.

**Example 1.** Consider the true QBF \( \Pi.\psi = \forall a \exists b, c, (a \lor b) \land (\bar{a} \lor c) \land (b \lor c) \). Clause \((a \lor c)\) has QRAT on \( c \) w.r.t. \( \Pi.\psi \); the only clause that contains literal \( \bar{c} \) is \((b \lor c)\), which produces the outer resolvent \((a \lor b \lor c)\). Since \( ALA(\psi \setminus \{(a \lor c)\}, (a \lor b \lor c)) = (a \lor \bar{a} \lor b \lor \bar{b} \lor c \lor \bar{c}) \) is a tautology, QRATA can add \((a \lor c)\) to \( \psi \). Now, consider a new existential variable \( \bar{d} \) in the innermost quantifier block. The clause \((b \lor c \lor \bar{d})\) has QRAT on \( c \) (and \( d \)) w.r.t. \( \psi \). Adding \((b \lor c \lor \bar{d})\) to \( \psi \) results in the true QBF \( \forall a \exists b, c, (a \lor b) \land (\bar{a} \lor c) \land (b \lor \bar{c}) \land (b \lor c \lor \bar{d}) \).

**Definition 6 (Outer Formula).** Let \( l \) be a literal occurring in QBF \( \Pi.\psi \). The outer formula of \( l \), denoted by \( OF(\Pi, l) \), is \( \{O(\Pi, D, \bar{l}) | D \in \psi, \bar{l} \in D\} \).

If a clause \( C \) has QRAT on \( l \in C \) w.r.t. a QBF formula \( \Pi.\psi \), we know that: 1) if the outer formula \( OF(\Pi, l) \) is falsified by an assignment then \( C \) is satisfied by that assignment; and 2) the outer formula \( OF(\Pi, \psi, l) \) is satisfied by an assignment, then \( l \) can be assigned to true, thereby satisfying \( C \). We use this property of QRAT clauses to construct Skolem functions. Given a QBF formula \( \Pi.\psi \), a valid Skolem set \( F \) for \( \Pi.\psi \) and a clause \( C \) that has QRAT on \( l \in C \) w.r.t. \( \Pi.\psi \). We can now make a valid Skolem set \( F' \) for \( \Pi.\psi \land \{ C \} \) by updating the Skolem function \( f_{\text{var}}(l) \) as follows. First, create a new variable \( y \) and make \( f_y(U) := f_{\text{var}}(U) \). Second, replace \( f_{\text{var}}(U) \) by the function stating that if \( OF(\Pi, \psi, l) \) evaluates to true then return polarity of \( l \) else return \( f_y(U) \).

**V. From Proof Validation to Skolem Functions**

In earlier work [12], we showed how to validate QRAT proofs. In this section, we describe how to extend that method to obtain Skolem functions after a satisfaction proof, i.e., a proof for a true QBF, has been validated. We start with a brief discussion on how to validate satisfaction proofs. Afterwards, we explain how to integrate the extraction of Skolem functions into the checking. We end this section with a running example of the algorithm.

**A. Validating QRAT Proofs**

QRAT proofs are sequences of clause additions and deletions. They are build using three kind of lines: addition (N2+E2), deletion (N1+E1), and universal elimination (U1+U2). In the QRAT proof 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 have prefix “\( \exists d \)”, while universal elimination lines have prefix “\( \exists u \)”. Both are restricted in the following way: the clause after a “\( \exists d \)” or “\( \exists u \)” prefix must be either present in the original formula or as a clause added earlier in the proof. For satisfaction proofs, a universal elimination line can be replaced by a clause addition (N2) and clause deletion line (N1). We assume that all universal elimination lines have been replaced and ignore their existence for the remaining part of the paper.

Fig. 1 shows the basic algorithm to validate QRAT proofs. Let us ignore line 1, 2, 9, and 13 for the moment, because they are only required to produce Skolem functions. We loop over the clauses in the proof in the order a QBF solver or preprocessor added or removed clauses (line 3). The first unexamined clause is obtained from the proof together with its flag and pivot (line 4). The flag can be either add or delete (in the proof format no prefix or a “\( \exists d \)” prefix, respectively). If the flag is add, no checking is required because this is a strengthening step. The new clause is simply added to \( \psi \) (line 11). Else, the clause will be removed (line 6). This elimination step needs to be validated. We check if the clause is logically implied by \( \psi \) by computing whether the clause is an asymmetric tautology (line 7). If that is not the case, the clause needs to have QRAT w.r.t. \( \psi \) (line 8), otherwise the proof is invalid (line 10). This procedure continues until all clauses in the proof have been processed. At this point, \( \psi \) should be empty, showing that the original formula is
satisfiability equivalent to the empty formula. If \( \psi \) is empty, the proof is valid (line 14), otherwise it is invalid (line 12).

validateQRAT (QBF formula \( \Pi.\psi \), QRAT proof \( P \))
\[
\begin{align*}
&\text{v1} \quad \text{let} \ V = \text{vars}(P) \\
&\text{v2} \quad \text{initSkolem (II, } V) \\
&\text{v3} \quad \text{while } P \neq \emptyset \text{ do} \\
&\text{v4} \quad \langle \text{flag, } l, C \rangle := P.\text{dequeue}() \\
&\text{v5} \quad \text{if } \text{flag} = \text{delete} \text{ then} \\
&\text{v6} \quad \psi := \psi \cup \{C\} \\
&\text{v7} \quad \text{if } \text{ALA}(\psi, C) \text{ is a tautology then continue} \\
&\text{v8} \quad \text{else if } C \text{ has QRAT on } l \in C \text{ w.r.t. } \Pi.\psi \text{ then} \\
&\text{v9} \quad \text{addSkolem (II, } \psi, C, l) \\
&\text{v10} \quad \text{else return } \text{'INVALID PROOF'} \\
&\text{v11} \quad \text{else } \psi := \psi \cup \{C\} \\
&\text{v12} \quad \text{if } \psi \neq \emptyset \text{ then return } \text{'INVALID PROOF'} \\
&\text{v13} \quad \text{finishSkolem (V)} \\
&\text{v14} \quad \text{return } \text{'VALID PROOF'}
\end{align*}
\]

Fig. 1. Procedure to check QRAT proofs and output Skolem functions.

B. Extracting Skolem Functions

The basic QRAT validation algorithm can easily be enhanced to produce Skolem functions. In Fig. 1 this is shown by the lines 1 and 2 (initialization), line 9 (producing Skolem functions), and line 13 (termination). Notice that although the QRAT proof system uses six rules, recall Table II, the Skolem functions only depend on one of them, i.e., quantified resolution asymmetric tautology elimination (E1). The reason why only (E1) has to be taken into account is that the other rules either strengthen \( \psi \) or preserve logical equivalence.

Fig. 2 shows the procedures used to extract the Skolem functions. An important part of the extraction algorithm is the global array last which contains for each variable in the QRAT proof a pointer to the last variable on which its Skolem function depends. Initially, see initSkolem, last[\( x \)] := \( x \) for existential variables and last[\( x \)] := \( 0 \) for universal variables, since there are no Skolem functions for universal variables. After the QRAT proof has been validated, for all variables \( x \) pointed to in the last array, a Skolem function \( f_x(U) \) is added that is always true (\( \top \)) (see finishSkolem).

The real work is done in the addSkolem procedure. This algorithm is inspired on the solution reconstruction procedure for RAT proofs, the variant of QRAT for propositional (SAT) formulas [20]. The algorithm works as follows: pick an arbitrary assignment. Loop over all clauses in the RAT proof in reverse order. If a clause is falsified by the current assignment, flip the truth value of the pivot. In order to make this algorithm produce Skolem functions out of QRAT proofs, some changes have to be made. Most importantly, we need to respect the quantifier prefix, because Skolem functions may not depend on variables that are more inner w.r.t. the quantifier prefix.

The addSkolem procedure uses two sub-procedures of which the pseudo-code is not shown: pol(l) and eval(\( F \)). The procedure pol(l) simply returns the polarity of literal \( l \), i.e., \( \top \) if \( l \) is a positive literal and \( \bot \) if \( l \) is a negative literal. The procedure eval(\( F \)) returns the clause set \( F \) under the current Skolem functions. It replaces each positive literal \( x \) with \( f_{\text{last}[x]}(U) \) and each negative literal \( \bar{x} \) by \( \neg f_{\text{last}[x]}(U) \). For example, the expression eval(\( (a \lor b) \land (c) \)) is replaced by \( (f_{\text{last}[a]}(U) \lor \neg f_{\text{last}[b]}(U)) \land f_{\text{last}[c]}(U) \).

At the end of Section IV, 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 \( C' \), a copy of \( C \) with all inner literals to the pivot removed, has QRAT as well. In that case, we only need to check whether the outer clause of \( C \) w.r.t. the pivot is falsified.

Now we have all elements to explain the addSkolem procedure. A new existential variable \( y \) is created (line 2). Afterwards, we compute \( C' \), a copy of \( C \) with all inner literals to the pivot are removed (line 3). If \( C' \) has QRAT on \( l \) w.r.t. \( \Pi.\psi \) (line 4) then the Skolem function for the pivot becomes as shown in the pseudo-code on line 5. Otherwise, we need to compute the outer formula of the pivot and use it for the Skolem function (line 7). The procedure terminates by updating the last array using \( y \) (line 8).

initSkolem (prefix \( \Pi \), set of variables \( V \))
\[
\begin{align*}
&\text{iS1} \quad \text{foreach } x \in V \text{ do} \\
&\text{iS2} \quad \text{if } Q(\Pi.x) = \top \text{ then last}[x] := 0 \text{ else last}[x] := x \\
&\text{addSkolem (QBF formula } \Pi.\psi, \text{ clause } C, \text{ literal } l) \\
&\text{aS1} \quad \text{let } x = \text{last[}\text{var}(l)\text{]} \\
&\text{aS2} \quad \text{let } y = \text{a new existential variable} \\
&\text{aS3} \quad \text{let } C' := \{k \in C \mid k \leq l\} \\
&\text{aS4} \quad \text{if } C' \text{ has QRAT on } l \text{ w.r.t. } \Pi.\psi \text{ then} \\
&\text{aS5} \quad f_x(U) := \text{if}\text{eval(QF}(\Pi,[C,l])\text{)}\text{ then } f_x(U) \text{ else } \text{pol(l)} \\
&\text{aS6} \quad \text{else} \\
&\text{aS7} \quad f_x(U) := \text{if}\text{eval(QF}(\Pi,[C,l])\text{)}\text{ then } \text{pol(l)} \text{ else } f_x(U) \\
&\text{aS8} \quad \text{last[}\text{var}(l)\text{]} := y \\
\end{align*}
\]

finishSkolem (set of variables \( V \))
\[
\begin{align*}
&\text{fS1} \quad \text{foreach } x \in V \text{ do} \\
&\text{fS2} \quad \text{if last}[x] \neq 0 \text{ then } f_{\text{last}[x]}(U) := \top
\end{align*}
\]

Fig. 2. Procedures to init, add, and finish Skolem functions.

C. Running Example

The true QBF below is used to illustrate how the extraction of Skolem functions from a QRAT proofs works:
\[
\Pi.\psi := \exists a, b \forall x, \exists c. (a \lor b) \land (\bar{a} \lor \bar{b}) \land (\bar{a} \lor x \lor c) \land (\bar{a} \lor \bar{c})
\]

Fig. 3 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 \( C \). The initialization
true QBF formula in DIMACS

| p cnf 4 4 |
| e 1 2 0 |
| a 4 0 |
| e 3 0 |
| 1 2 0 |
| -1 -2 0 |
| -1 3 4 0 |
| -3 -4 0 |

satisfaction QRAT proof

| d -2 -1 0 |
| 2 3 4 0 |
| d -1 3 4 0 |
| d 1 2 0 |
| d 2 3 4 0 |
| d -3 -4 0 |

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

of Skolem functions will assign the last array as follows:

last[a] := a, last[b] := b, last[c] := c, and last[x] := 0.

The proof consists of the following steps. First, C := (a ∨ b) is removed from ψ using b as pivot. Since C has no inner literals, C′ := C and consequently C′ has QRAT on b w.r.t. the new ψ (from which C has been removed). We introduce a new existential variable b1. Now the Skolem function fb(U) will be – with eval(α) replaced by ¬fa(U), and pol(b) by ↓:

fb(U) := if(¬fa(U)) then fa(U) else ⊥

The second step in the proof is adding clause (b ∨ x ∨ c) to ψ. Since this involves clause addition, no Skolem function is added. The third step is the most tricky one. Clause C := (a ∨ x ∨ c) is now removed using pivot a and checked for redundancy. Both x and c are inner to a, so C′ := (a). C′ does not have QRAT on a w.r.t. the new ψ (which now contains (b ∨ x ∨ c), but no longer has (a ∨ x ∨ c)). In this case the outer formula: F := (b). Although F is small, it can be almost as large as ψ in the worst case. Now, the Skolem function fa(U) will be – with eval(F) replaced by fb(U) and pol(α) by ↓:

fa(U) := if(fb(U)) then ⊥ else fa1(U)

The fourth step concerns the removal of (a ∨ b) using pivot a. This step is practically the same as the first step. Now last[a] = a1, so we compute for Skolem function fa1(U):

fa1(U) := if(fb(U)) then fa2(U) else ⊤

In the fifth step, C := (b ∨ x ∨ c), which was added in step two, is removed. Again, the literals x and c are inner to b. This results in C′ := (b). In contrast to step three, this C′ has QRAT on b w.r.t. the current ψ because ψ no longer contains any clause with literal b.

fb(U) := if(⊥) then fa(U) else ⊤

Finally, the last clause (x ∨ c) is removed with pivot c. The Skolem function fc(U) will be:

fc(U) := if(¬x) then fc1(U) else ⊥

After the QRAT proof has been validated, we call finishSkolem which does the following assignments:

f02(U) := f02(U) := f03(U) := ⊤. Using these Skolem functions, we can set the earlier Skolem functions to f0(U) := ⊥, f0(U) := ⊤, and f0(U) := ¬x.

VI. IMPLEMENTATION, OPTIMIZATION AND VALIDATION

We enhanced our QRAT checking tool, called QRAT-trim, with Skolem function extraction capabilities. The Skolem functions can be emitted as a propositional formula in DIMACS format or as an and-inverter-graph in AIGER format. This section describes some details about our implementation, optimizations and validation of Skolem function extraction.

A. Reducing the Size of the Outer Formula

The outer formula computed in line 7 of the addSkolem procedure (Fig. 2) 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 ∈ ψ with i ∈ D, we compute the outer resolvent R := O(Π, D, i) ∪ C′. We check whether AIA(ϕ, R) is a tautology and store all O(Π, D, i) for which the corresponding R is not a tautology. The alternative outer formula becomes the conjunction of these O(Π, D, i) together with the negation of C′ \ {i}.

B. Value of Final Skolem Functions

We presented finishSkolem such that it assigns all Skolem functions flast[x](U) := ⊤. However, for some variables, flast[x](U) := ⊥ is much more effective. We observed that the best truth value for final Skolem functions ffinal[x] is based on the polarity of a literal that was a pivot for a QRAT check. For some variables x (typically a few hundred for each benchmark), there are QRAT checks with literal x as a pivot, but no QRAT checks with literal x̄ as pivot (or the other way around). By assigning ffinal[x](U) := ⊤ (or ffinal[x](U) := ⊥, respectively), and apply simplification, we obtain the Skolem functions f̄x(U) := ⊤ (or f̄x(U) := ⊥, respectively).

C. Validation of Skolem Functions

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 ψ. 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 F to check that no input gate gi (universal variable) influences the truth value of output gate go (existential variable) with gi >Π go. 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 that

2The QRAT-trim version with Skolem function extraction is available on http://www.cs.utexas.edu/~marijn/skolem/.
checks whether the Skolem functions are correct. A tool, called CertCheck [21], 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 2.** Consider the formula \( \exists a \forall \forall c. (a \lor b \lor c) \land (\bar{a} \lor \bar{c}) \). A possible QRAT proof for this formula removes first \((a \lor b \lor c)\) with pivot \( c \) and afterwards \((\bar{a} \lor \bar{c})\) with pivot \( \bar{a} \). The latter is allowed because after removing \((a \lor b \lor c)\) the prefix collapses to \( \exists a \). Our procedure for extracting Skolem functions can result in \( f_a(U) = f_c(U) \) and \( f_c(U) = \top \) (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 \( a \).

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.

**TABLE II**

<table>
<thead>
<tr>
<th>formula</th>
<th>sol-t</th>
<th>ext-t</th>
<th>tr-s</th>
<th>ch-t</th>
<th>qbc-s</th>
</tr>
</thead>
<tbody>
<tr>
<td>c3BMC_p1_k2</td>
<td>1.08</td>
<td>2.10</td>
<td>1777</td>
<td>0.04</td>
<td>71</td>
</tr>
<tr>
<td>counter_8</td>
<td>0.39</td>
<td>0.38</td>
<td>266</td>
<td>0.07</td>
<td>37</td>
</tr>
<tr>
<td>itc-b13-fxpoint-8</td>
<td>16.32</td>
<td>75.53</td>
<td>12496</td>
<td>337.80</td>
<td>14756</td>
</tr>
<tr>
<td>k_branch_n-16</td>
<td>6.12</td>
<td>137.40</td>
<td>13127</td>
<td>7.14</td>
<td>1296</td>
</tr>
<tr>
<td>k4_n-7</td>
<td>3.71</td>
<td>25.33</td>
<td>25449</td>
<td>337.41</td>
<td>7467</td>
</tr>
<tr>
<td>k4_d_n-10</td>
<td>1.08</td>
<td>3.23</td>
<td>5688</td>
<td>31.45</td>
<td>1988</td>
</tr>
<tr>
<td>k4_d_n-11</td>
<td>1.22</td>
<td>3.97</td>
<td>6401</td>
<td>42.20</td>
<td>2244</td>
</tr>
<tr>
<td>k4_d_n-14</td>
<td>1.58</td>
<td>7.36</td>
<td>9379</td>
<td>25.05</td>
<td>3158</td>
</tr>
<tr>
<td>k4_d_n-15</td>
<td>1.75</td>
<td>8.78</td>
<td>10536</td>
<td>86.55</td>
<td>3459</td>
</tr>
<tr>
<td>k4_d_n-20</td>
<td>2.53</td>
<td>18.94</td>
<td>16637</td>
<td>94.76</td>
<td>4885</td>
</tr>
<tr>
<td>k4_d_n-21</td>
<td>2.75</td>
<td>22.08</td>
<td>18442</td>
<td>140.58</td>
<td>5296</td>
</tr>
<tr>
<td>k_dum_n-10</td>
<td>0.17</td>
<td>0.42</td>
<td>262</td>
<td>0.06</td>
<td>62</td>
</tr>
<tr>
<td>k_dum_n-11</td>
<td>0.23</td>
<td>0.49</td>
<td>335</td>
<td>0.07</td>
<td>74</td>
</tr>
<tr>
<td>k_dum_n-12</td>
<td>0.26</td>
<td>0.47</td>
<td>342</td>
<td>0.07</td>
<td>63</td>
</tr>
<tr>
<td>k_dum_n-16</td>
<td>0.26</td>
<td>0.60</td>
<td>432</td>
<td>0.09</td>
<td>95</td>
</tr>
<tr>
<td>k_dum_n-20</td>
<td>0.33</td>
<td>0.85</td>
<td>657</td>
<td>0.18</td>
<td>158</td>
</tr>
<tr>
<td>k_dun-n-21</td>
<td>0.44</td>
<td>0.93</td>
<td>761</td>
<td>0.32</td>
<td>204</td>
</tr>
<tr>
<td>lights3_021_l_022</td>
<td>0.18</td>
<td>0.46</td>
<td>253</td>
<td>0.09</td>
<td>54</td>
</tr>
<tr>
<td>lights3_021_l_033</td>
<td>0.23</td>
<td>0.49</td>
<td>335</td>
<td>0.07</td>
<td>74</td>
</tr>
<tr>
<td>lights3_035_l_059</td>
<td>0.39</td>
<td>0.70</td>
<td>639</td>
<td>0.34</td>
<td>111</td>
</tr>
<tr>
<td>rankfunc0_unsigned_64</td>
<td>1.70</td>
<td>7.55</td>
<td>3244</td>
<td>36.69</td>
<td>4985</td>
</tr>
<tr>
<td>rankfunc16_unsigned_16</td>
<td>0.33</td>
<td>1.42</td>
<td>783</td>
<td>0.83</td>
<td>342</td>
</tr>
<tr>
<td>rankfunc24_signed_32</td>
<td>0.56</td>
<td>1.92</td>
<td>1049</td>
<td>10.37</td>
<td>902</td>
</tr>
<tr>
<td>rankfunc27_unsigned_32</td>
<td>0.44</td>
<td>0.80</td>
<td>1446</td>
<td>1.90</td>
<td>569</td>
</tr>
<tr>
<td>rankfunc52_signed_64</td>
<td>1.80</td>
<td>9.23</td>
<td>3652</td>
<td>363.51</td>
<td>5058</td>
</tr>
<tr>
<td>s3330_d2_s</td>
<td>10.60</td>
<td>19.22</td>
<td>122659</td>
<td>3.59</td>
<td>798</td>
</tr>
<tr>
<td>stmt137_003_911</td>
<td>0.39</td>
<td>0.54</td>
<td>1037</td>
<td>0.27</td>
<td>112</td>
</tr>
<tr>
<td>stmt1629_630</td>
<td>0.72</td>
<td>1.20</td>
<td>1671</td>
<td>0.74</td>
<td>187</td>
</tr>
<tr>
<td>stmt17_99_98</td>
<td>1.22</td>
<td>2.69</td>
<td>3388</td>
<td>2.19</td>
<td>373</td>
</tr>
<tr>
<td>stmt27_584_603</td>
<td>0.42</td>
<td>0.63</td>
<td>975</td>
<td>0.27</td>
<td>132</td>
</tr>
<tr>
<td>stmt27_946_955</td>
<td>0.39</td>
<td>0.53</td>
<td>977</td>
<td>0.32</td>
<td>113</td>
</tr>
<tr>
<td>stmt4_118_131</td>
<td>0.36</td>
<td>0.52</td>
<td>774</td>
<td>0.68</td>
<td>166</td>
</tr>
</tbody>
</table>

sol-t:ext-t:ch-t: solving/extraction/checking time (sec) tr-s/qbc-s: size of QRAT file/qbc file (kilobyte)
TABLE III
RESULTS OF SKOLEM FUNCTION PRODUCING TOOLS ON QBF EVAL 12

<table>
<thead>
<tr>
<th>solver</th>
<th>sol-#</th>
<th>sol-t</th>
<th>ch-#</th>
<th>ch-t</th>
<th>cer-s</th>
</tr>
</thead>
<tbody>
<tr>
<td>bloqqer+QRAT</td>
<td>32</td>
<td>1</td>
<td>32</td>
<td>47</td>
<td>1851</td>
</tr>
<tr>
<td>bloqqer+RES</td>
<td>22</td>
<td>1</td>
<td>22</td>
<td>1</td>
<td>861</td>
</tr>
<tr>
<td>bloqqer+RES+depQBF</td>
<td>28</td>
<td>113</td>
<td>27</td>
<td>13</td>
<td>1040</td>
</tr>
<tr>
<td>depQBF</td>
<td>2</td>
<td>843</td>
<td>2</td>
<td>1</td>
<td>224</td>
</tr>
<tr>
<td>ebdd</td>
<td>15</td>
<td>491</td>
<td>7</td>
<td>118</td>
<td>409479</td>
</tr>
<tr>
<td>squolem</td>
<td>16</td>
<td>465</td>
<td>16</td>
<td>2</td>
<td>382</td>
</tr>
<tr>
<td>sKizzo</td>
<td>23</td>
<td>275</td>
<td>23</td>
<td>1</td>
<td>108750</td>
</tr>
</tbody>
</table>

sol-#: # solved formulas, sol-t: avg. solving time (s),
ch-#: checked certificates, ch-t: avg. checking time (s)
cer-s: avg. certificate size (kilobyte)

VIII. CONCLUSION AND FUTURE WORK

The QRAT proof system is the first framework that allows verification of all preprocessing techniques for QBFs. We developed an algorithm that extracts Skolem functions of QRAT proofs. Hence, the techniques presented in this paper allow us to obtain Skolem functions for all QBF preprocessing techniques. Moreover, these Skolem functions are smaller than those produced by alternative approaches – a very favorable property for many synthesis applications.

We expect that the produced Skolem functions can be further reduced in size: we applied the circuit simplification tool ABC [24] on the AIGs representing the set of Skolem functions and noticed a significant reduction. However, the simplified Skolem functions are not necessarily valid, as ABC is not aware of the dependency restrictions. In future, we want to consider Skolem function reduction by circuit simplification while taking into account the dependencies.

ACKNOWLEDGEMENTS

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, DARPA contract number N66001-10-2-4087 and National Science Foundation grant no. CCF-1153558.

REFERENCES