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.

ISBN: 978-0-9835678-4-4. Copyright owned jointly by the authors and FMCAD Inc.

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).[1] A QBF in PCNF has the structure $\Pi.\psi$ where the prefix Π 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 ψ 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 $\mathsf{var}(l)$ where $\mathsf{var}(l) = x$ if $l = x$ or $l = \bar{x}$. The negation of a literal l is denoted by $\bar{l}$. 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 $i \leq j$. 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 variables occurring in the prefix of ϕ are given by $\mathsf{vars}(\phi)$. The subformula ψ_l consisting of all clauses of matrix ψ with literal l is defined by $\psi_l = \{C \mid l \in C, C \in \psi\}$. By $\top$ and $\bot$ we denote the truth constants true and false. QBFs are interpreted as follows: a QBF $\forall x\Pi.\psi$ 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. Two QBFs ϕ_1 and ϕ_2 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.

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 ϕ 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 $\mathcal{F}$ 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 ϕ, a valid Skolem set $\mathcal{F}$ 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 Π, as parameters.

To check that a Skolem set $\mathcal{F}$ is valid, it is necessary to substitute in ϕ all existential variables by their corresponding Skolem functions in $\mathcal{F}$ 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 $\mathsf{ALA}(\psi, C)$ is the unique clause obtained by repeatedly applying the extension rule

$$C := C \cup \{\bar{l}\} \text{ if } \exists l_1, \ldots, l_k \in C \text{ and } (l_1 \vee \ldots \vee l_k \vee l) \in \psi$$

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 ALA is model preserving, for any QBF $\phi = \Pi.\psi \wedge \{C\}$ holds that $\phi \approx \phi[C/C']$ where $C' = \mathsf{ALA}(\psi \setminus \{C\}, C)$ [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 squolem [15] as well as the BDD-based solver ebddres [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

[1] Note that any QBF of arbitrary structure can be efficiently transformed to a satisfiability equivalent formula in PCNF.

ISBN: 978-0-9835678-4-4. Copyright owned jointly by the authors and FMCAD Inc.

TABLE I
THE QRAT PROOF SYSTEM

	Rule	Preconditions	Postconditions
(N1)	$\Pi.\psi \xrightarrow{\mathsf{ATE}(C)} \Pi.\psi\backslash\{C\}$	C is an asymmetric tautology	
(N2)	$\Pi.\psi \xrightarrow{\mathsf{ATA}(C)} \Pi'.\psi \cup \{C\}$	C is an asymmetric tautology	$\Pi' = \Pi\exists X$ with $X = \{x \mid x \in \mathsf{vars}(C), x \notin \mathsf{vars}(\Pi)\}$
(E1)	$\Pi.\psi \xrightarrow{\mathsf{QRATE}(C,l)} \Pi.\psi\backslash\{C\}$	$C \in \psi$, $\mathsf{Q}(\Pi, l) = \exists$ C has QRAT on l w.r.t. ψ	
(E2)	$\Pi.\psi \xrightarrow{\mathsf{QRATA}(C,l)} \Pi'.\psi \cup \{C\}$	$C \notin \psi$, $\mathsf{Q}(\Pi, l) = \exists$ C has QRAT on l w.r.t. ψ	$\Pi' = \Pi\exists X$ with $X = \{x \mid x \in \mathsf{vars}(C), x \notin \mathsf{vars}(\Pi)\}$
(U1)	$\Pi.\psi \cup \{C\} \xrightarrow{\mathsf{QRATU}(C,l)} \Pi.\psi \cup \{C\backslash\{l\}\}$	$l \in C$, $\mathsf{Q}(\Pi, l) = \forall$, $\bar{l} \notin C$, C has QRAT on l w.r.t. ψ	
(U2)	$\Pi.\psi \cup \{C\} \xrightarrow{\mathsf{EUR}(C,l)} \Pi.\psi \cup \{C\backslash\{l\}\}$	$l \in C$, $\mathsf{Q}(\Pi, l) = \forall$, $\bar{l} \notin C$, C has EUR on l w.r.t. ψ	

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 looses 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 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 $\mathsf{ALA}(\psi\backslash\{C\}, C)$ is a tautology. As the addition of asymmetric literals is model preserving, it holds that $\Pi.\psi \approx \Pi.\psi \cup C$ iff $\mathsf{ALA}(\psi\{C\}, C)$ 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 ψ 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 σ, 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 ϕ preserves satisfiability and the deletion of C to ϕ 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 $\mathcal{O}(\Pi, C, l)$, is given by the clause $\{k \mid k \in C, k \leq_{\Pi} l, k \neq l\}$.

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

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 $D \in \psi_{\bar{l}}$ that $\mathsf{ALA}(\psi \setminus \{C\}, R)$ is a tautology for the outer resolvent $R = \mathcal{R}(\Pi, C, D, l)$.

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

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

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 $\mathcal{OF}(\Pi, \psi, l)$ is falsified by an assignment then C is satisfied by that assignment; and 2) the outer formula $\mathcal{OF}(\Pi, \psi, l)$ 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 $\mathcal{F}$ 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 $\mathcal{F}'$ for $\Pi.\psi \wedge \{C\}$ by updating the Skolem function $f_{\mathsf{var}(l)}(U)$ as follows. First, create a new variable y and make $f_y(U) := f_{\mathsf{var}(l)}(U)$. Second, replace $f_{\mathsf{var}(l)}(U)$ by the function stating that if $\mathcal{OF}(\Pi, \psi, l)$ 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 “`d`”, while universal elimination lines have prefix “`u`”. Both are restricted in the following way: 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. 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 “`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 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 ψ 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. ψ (line 8), otherwise the proof is invalid (line 10). 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 14), otherwise it is invalid (line 12).

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

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 be to taken into account is that the other rules either strengthen ψ 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*, $\mathsf{last}[x] := x$ for existential variables and $\mathsf{last}[x] := 0$ 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: $\mathsf{pol}(l)$ and $\mathsf{eval}(F)$. The procedure $\mathsf{pol}(l)$ 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 $\mathsf{eval}(F)$ returns the clause set F under the current Skolem functions. It replaces each positive literal x with $f_{\mathsf{last}[x]}(U)$ and each negative literal $\bar{x}$ by $\neg f_{\mathsf{last}[x]}(U)$. For example, the expression $\mathsf{eval}((a \vee \bar{b}) \wedge (c))$ is replaced by $(f_{\mathsf{last}[a]}(U) \vee \neg f_{\mathsf{last}[b]}(U)) \wedge f_{\mathsf{last}[c]}(U)$.

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 $V$)
iS1 foreach $x \in V$ do
iS2   if $Q(\Pi.x) = \forall$ then $\mathsf{last}[x] := 0$ else $\mathsf{last}[x] := x$

    addSkolem (QBF formula $\Pi.\psi$, clause $C$, literal $l$)
aS1 let $x$ be $\mathsf{last}[\mathsf{var}(l)]$
aS2 let $y$ be a new existential variable
aS3 let $C' := \{k \in C \mid k \leq_\Pi l\}$
aS4 if $C'$ has QRAT on $l$ w.r.t. $\Pi.\psi$ then
aS5   $f_x(U) := \text{if}(\mathsf{eval}(\mathcal{O}(\Pi, C, l))\,\text{then}\, f_y(U)\,\text{else}\, \mathsf{pol}(l)$
aS6 else
aS7   $f_x(U) := \text{if}(\mathsf{eval}(\mathcal{OF}(\Pi, C, l)))\,\text{then}\, \mathsf{pol}(l)\,\text{else}\, f_y(U)$
aS8 $\mathsf{last}[\mathsf{var}(l)] := y$

    finishSkolem (set of variables $V$)
fS1 foreach $x \in V$ do
fS2   if $\mathsf{last}[x] \neq 0$ then $f_{\mathsf{last}[x]}(U) := \top$
```

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:

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

Fig. 3 shows how this formula looks in the QDIMACS format, which is used by most QBF solvers and preprocessers (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 ("d " 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: $\mathsf{last}[a] := a$, $\mathsf{last}[b] := b$, $\mathsf{last}[c] := c$, and $\mathsf{last}[x] := 0$.

The proof consists of the following steps. First, $C := (\bar{a} \vee \bar{b})$ is removed from ψ using $\bar{b}$ as pivot. Since C has no inner literals, $C' := C$ and consequently C' has QRAT on $\bar{b}$ w.r.t. the new ψ (from which C has been removed). We introduce a new existential variable b_1. Now the Skolem function $f_b(U)$ will be – with $\mathsf{eval}(\bar{a})$ replaced by $\neg f_a(U)$, and $\mathsf{pol}(\bar{b})$ by $\bot$:

$$f_b(U) := \quad \mathrm{if}(\neg f_a(U))\,\mathrm{then}\, f_{b_1}(U)\,\mathrm{else}\,\bot$$

The second step in the proof is adding clause $(b \vee x \vee c)$ to ψ. Since this involves clause addition, no Skolem function is added. The third step is the most tricky one. Clause $C := (\bar{a} \vee x \vee c)$ is now removed using pivot $\bar{a}$ and checked for redundancy. Both x and c are inner to $\bar{a}$, so $C' := (\bar{a})$. C' does not have QRAT on $\bar{a}$ w.r.t. the new ψ (which now contains $(b \vee x \vee c)$, but no longer has $(\bar{a} \vee x \vee c)$). 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 $f_a(U)$ will be – with $\mathsf{eval}(F)$ replaced by $f_{b_1}(U)$ and $\mathsf{pol}(a)$ by $\bot$:

$$f_a(U) := \quad \mathrm{if}(f_{b_1}(U))\,\mathrm{then}\,\bot\,\mathrm{else}\, f_{a_1}(U)$$

The fourth step concerns the removal of $(a \vee b)$ using pivot a. This step is practically the same as the first step. Now $\mathsf{last}[a] = a_1$, so we compute for Skolem function $f_{a_1}(U)$:

$$f_{a_1}(U) := \quad \mathrm{if}(f_{b_1}(U))\,\mathrm{then}\, f_{a_2}(U)\,\mathrm{else}\,\top$$

In the fifth step, $C := (b \vee x \vee c)$, which was added in step two, is removed. Again, the literals x and c are inner to b. This results in $C' := (b)$. 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 $\bar{b}$.

$$f_{b_1}(U) := \quad \mathrm{if}(\bot)\,\mathrm{then}\, f_{b_2}(U)\,\mathrm{else}\,\top$$

Finally, the last clause $(\bar{x} \vee \bar{c})$ is removed with pivot $\bar{c}$. The Skolem function $f_c(U)$ will be:

$$f_c(U) := \quad \mathrm{if}(\neg x)\,\mathrm{then}\, f_{c_1}(U)\,\mathrm{else}\,\bot$$

After the QRAT proof has been validated, we call *finishSkolem* which does the following assignments: $f_{a_2}(U) := f_{b_2}(U) := f_{c_1}(U) := \top$. Using these Skolem functions, we can set the earlier Skolem functions to $f_a(U) := \bot$, $f_b(U) := \top$, and $f_c(U) := \neg x$.

VI. Implementation, Optimization and Validation

We enhanced our QRAT checking tool, called QRAT-trim, with Skolem function extraction capabilities.[2] 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 $D \in \psi$ with $\bar{l} \in D$, we compute the outer resolvent $R = \mathcal{O}(\Pi, D, \bar{l}) \cup C'$. We check whether $\mathsf{ALA}(\varphi, R)$ is a tautology and store all $\mathcal{O}(\Pi, D, \bar{l})$ for which the corresponding R is *not* a tautology. The alternative outer formula becomes the conjunction of these $\mathcal{O}(\Pi, D, \bar{l})$ together with the negation of $C' \setminus \{l\}$.

B. Value of Final Skolem Functions

We presented *finishSkolem* such that it assigns all Skolem functions $f_{\mathsf{last}[x]}(U) := \top$. However, for some variables, $f_{\mathsf{last}[x]}(U) := \bot$ is much more effective. We observed that the best truth value for final Skolem functions $f_{\mathsf{last}[x]}$ 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 $\bar{x}$ as pivot (or the other way around). By assigning $f_{\mathsf{last}[x]}(U) := \top$ (or $f_{\mathsf{last}[x]}(U) := \bot$, respectively), and apply simplification, we obtain the Skolem functions $f_x(U) := \top$ (or $f_x(U) := \bot$, 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 $\mathcal{F}$ 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 $\mathcal{F}$. 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 that

[2] The 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 $\exists a \forall b \exists c.(a \vee b \vee c) \wedge (\bar{a} \vee \bar{c})$. A possible QRAT proof for this formula removes first $(a \vee b \vee c)$ with pivot c and afterwards $(\bar{a} \vee \bar{c})$ with pivot $\bar{a}$. The latter is allowed because after removing $(a \vee b \vee c)$ the prefix collapses to $\exists a, c$. Our procedure for extracting Skolem functions can result in $f_a(U) = \neg f_c(U)$ 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
STATISTICS OF EXTRACTING SKOLEM FUNCTIONS FROM QRAT PROOFS PRODUCED BY BLOQQER ON QBF EVAL 12 BENCHMARKS.

formula	sol-t	ext-t	tr-s	ch-t	qbc-s
c3_BMC_p1_k2	1.08	2.10	1777	0.04	71
counter_8	0.39	0.38	266	0.07	37
itc-b13-fixpoint-8	16.32	75.53	124969	337.80	14756
k_branch_n-16	6.12	137.40	13127	7.14	1296
k_branch_n-7	3.71	25.33	25449	337.41	7467
k_d4_n-10	1.08	3.23	5688	31.45	1988
k_d4_n-11	1.22	3.97	6401	42.20	2244
k_d4_n-14	1.58	7.36	9379	25.05	3158
k_d4_n-15	1.75	8.78	10536	86.55	3459
k_d4_n-20	2.53	18.94	16637	94.76	4885
k_d4_n-21	2.75	22.08	18442	140.58	5296
k_dum_n-10	0.17	0.42	262	0.06	62
k_dum_n-11	0.23	0.49	335	0.07	74
k_dum_n-12	0.26	0.47	342	0.07	63
k_dum_n-16	0.26	0.60	432	0.09	95
k_dum_n-20	0.33	0.85	657	0.18	158
k_dum_n-21	0.44	0.93	761	0.32	204
lights3_021_1_022	0.18	0.46	253	0.09	54
lights3_021_1_033	0.23	0.41	388	0.07	47
lights3_035_1_059	0.39	0.70	639	0.34	111
rankfunc0_unsigned_64	1.70	7.55	3244	36.69	4985
rankfunc16_unsigned_16	0.33	1.42	783	0.83	342
rankfunc24_signed_32	0.56	1.82	1049	10.37	902
rankfunc27_unsigned_32	0.44	0.80	1446	1.90	569
rankfunc52_signed_64	1.80	9.23	3652	363.51	5058
s3330_d2_s	10.60	19.23	122569	3.59	798
stmt137_903_911	0.39	0.54	1037	0.27	112
stmt1_629_630	0.72	1.20	1671	0.74	187
stmt17_99_98	1.22	2.69	3388	2.19	373
stmt27_584_603	0.42	0.63	975	0.27	132
stmt27_946_955	0.39	0.53	977	0.32	113
stmt41_118_131	0.36	0.52	774	0.68	166

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

VII. Experimental Evaluation

At the moment, our preprocessor bloqqer is the only tool able to produce QRAT proofs. As it is not a complete solver, we consider the true formulas of the benchmark suite from QBF Eval 12 which can be solved by bloqqer. We showed in earlier work [12] that bloqqer with and without QRAT proof logging solves the same instances. In the following, we evaluate how our Skolem function extraction algorithm performs on the formulas for which QRAT proofs are available. Table II shows the results of our Skolem function extraction tool, i.e., a modified version of QRAT-trim. We converted all our proofs to the QBC format in order to make a comparison with other tools more clear. Notice that the extracted Skolem functions are typically smaller than the used QRAT proofs. We validated our Skolem functions using cheskol which checks the dependencies and computes whether the Skolem functions imply the formula using the SAT solver lingeling [22].

Janota et al. [19] 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 32). 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 [15], [16], [14] that are shown in Table III. The only tool that has comparable performance compared to our bloqqer+QRAT approach is bloqqer+RES+depQBF — although it cannot solve four of the harder benchmarks in the test suite.

A. Comparing the Size of Skolem Functions

If all preprocessing techniques are turned on, the average size of the Skolem functions produced by bloqqer+QRAT is larger than those from bloqqer+RES+depQBF. Recall that bloqqer+RES+depQBF does not support several preprocessing techniques. Consequently, unsupported techniques such as covered clause elimination (QCCE) [23] are turned off. Although bloqqer+QRAT supports QCCE, using it has a negative impact on the size of Skolem functions. On harder benchmarks, the Skolem functions are about four times larger due to this technique. Hence, turning QCCE off reduces the size of Skolem functions significantly — at the cost of solving one formula less (s3330_d2_s).

For a fair comparison between the size of Skolem functions produced by both approaches, we turn off QCCE for bloqqer+QRAT as done by bloqqer+RES+depQBF. The AIG files of the bloqqer+QRAT approach are converted into QBC certificates to have the same file format. The scatter plot shown in Fig. 4 illustrates that the Skolem functions extracted by bloqqer+QRAT are smaller than those produced by bloqqer+RES+depQBF, especially for the harder benchmarks.

TABLE III
RESULTS OF SKOLEM FUNCTION PRODUCING TOOLS ON QBF EVAL 12

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

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.

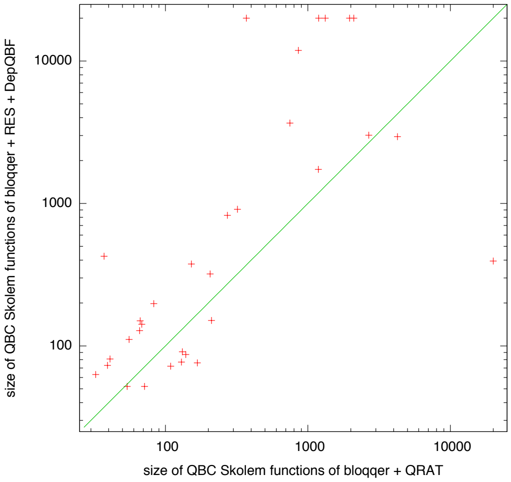

Fig. 4. Comparison between the size (in kilobyte) of QBC Skolem functions produced by the bloqqer+QRAT and bloqqer+RES+depQBF approaches on QBF Eval 2012 benchmarks. Above the line means that bloqqer+QRAT produces smaller files. Unsolved formulas are shown as a 20,000 kb file.

REFERENCES

[1] R. Bloem, R. Könighofer, and M. Seidl, “SAT-Based Synthesis Methods for Safety Specs,” in *VMCAI*, ser. LNCS, vol. 8318. Springer, 2014.
[2] M. Benedetti and H. Mangassarian, “QBF-Based Formal Verification: Experience and Perspectives,” *JSAT*, vol. 5, no. 1-4, pp. 133–191, 2008.
[3] H. Kleine Büning and U. Bubeck, “Theory of Quantified Boolean Formulas,” in *Handbook of Satisfiability*, 2009.
[4] C. Ansótegui, C. P. Gomes, and B. Selman, “The achilles’ heel of qbf,” in *AAAI*. AAAI Press / The MIT Press, 2005, pp. 275–281.
[5] H. Kleine Büning, K. Subramani, and X. Zhao, “Boolean functions as models for quantified boolean formulas,” *J. Aut. Reas.*, vol. 39, no. 1, 2007.
[6] E. Giunchiglia, P. Marin, and M. Narizzano, “Reasoning with quantified boolean formulas,” in *Handbook of Satisfiability*. IOS Press, 2009, vol. 185, pp. 761–780.
[7] V. Balabanov and J.-H. R. Jiang, “Resolution Proofs and Skolem Functions in QBF Evaluation and Applications,” in *CAV*, ser. LNCS, vol. 6806. Springer, 2011, pp. 149–164.
[8] A. Goultiaeva, A. Van Gelder, and F. Bacchus, “A Uniform Approach for Generating Proofs and Strategies for Both True and False QBF Formulas,” in *IJCAI*. IJCAI/AAAI, 2011, pp. 546–553.
[9] A. Biere, “Resolve and expand,” in *SAT (Selected Papers)*, ser. LNCS, vol. 3542. Springer, 2004, pp. 59–70.
[10] A. Biere, F. Lonsing, and M. Seidl, “Blocked clause elimination for QBF,” in *CADE 2011*, ser. LNCS, vol. 6803. Springer, 2011.
[11] M. Janota and J. Marques-Silva, “On Propositional QBF Expansions and Q-Resolution,” in *SAT 2013*, ser. LNCS, vol. 7962. Springer, 2013, pp. 67–82.
[12] M. J. H. Heule, M. Seidl, and A. Biere, “A Unified Proof System for QBF Preprocessing,” in *IJCAR 2014*, ser. LNCS, vol. 8562. Springer, 2014, pp. 91–106.
[13] M. J. H. Heule, M. Järvisalo, and A. Biere, “Clause elimination procedures for CNF formulas,” in *LPAR-17*, ser. LNCS, vol. 6397. Springer, 2010, pp. 357–371.
[14] M. Benedetti, “Skizzo: A suite to evaluate and certify QBFs,” in *CADE-20*, ser. LNCS, vol. 3632. Springer, 2005, pp. 369–376.
[15] T. Jussila, A. Biere, C. Sinz, D. Kröning, and C. Wintersteiger, “A first step towards a unified proof checker for QBF,” in *SAT 2007*, ser. LNCS. Springer, 2007, vol. 4501, pp. 201–214.
[16] T. Jussila, C. Sinz, and A. Biere, “Extended resolution proofs for symbolic sat solving with quantification,” in *SAT*, ser. LNCS, A. Biere and C. P. Gomes, Eds., vol. 4121. Springer, 2006, pp. 54–60.
[17] M. Janota, W. Klieber, J. Marques-Silva, and E. M. Clarke, “Solving QBF with Counterexample Guided Refinement,” in *SAT 2012*, ser. LNCS, vol. 7317. Springer, 2012.
[18] R. Könighofer and M. Seidl, “Partial witnesses from preprocessed quantified boolean formulas,” in *DATE*. IEEE, 2014, pp. 1–6.
[19] M. Janota, R. Grigore, and J. Marques-Silva, “On QBF Proofs and Preprocessing,” in *LPAR*, ser. LNCS, vol. 8312. Springer, 2013.
[20] M. Järvisalo, M. J. H. Heule, and A. Biere, “Inprocessing rules,” in *IJCAR*, ser. LNCS, vol. 7364. Springer, 2012, pp. 355–370.
[21] A. Niemetz, M. Preiner, F. Lonsing, M. Seidl, and A. Biere, “Resolution-Based Certificate Extraction for QBF,” in *SAT 2012*, ser. LNCS, vol. 7317, 2012, pp. 430–435.
[22] A. Biere, “Lingeling, Plingeling and Treengeling entering the SAT competition 2013,” in *Proceedings of SAT Competition 2013*, 2013.
[23] M. J. H. Heule, M. Järvisalo, and A. Biere, “Covered clause elimination,” in *LPAR-17-short*, ser. EPiC Series, A. Voronkov, G. Sutcliffe, M. Baaz, and C. Fermüller, Eds., vol. 13. EasyChair, 2013, pp. 41–46.
[24] Berkeley Logic Synthesis and Verification Group, ABC: A System for Sequential Synthesis and Verification. http://www.eecs.berkeley.edu/~alanmi/abc/.