Resolution Proofs and Skolem Functions in QBF Evaluation and Applications

Valeriy Balabanov and Jie-Hong R. Jiang

Department of Electrical Engineering / Graduate Institute of Electronics Engineering
National Taiwan University, Taipei 10617, Taiwan
{balabasik@gmail.com, jhjiang@cc.ee.ntu.edu.tw}

Abstract. Quantified Boolean formulae (QBF) allow compact encoding of many decision problems. Their importance motivated the development of fast QBF solvers. Certifying the results of a QBF solver not only ensures correctness, but also enables certain synthesis and verification tasks particularly when the certificate is given as a set of Skolem functions. To date the certificate of a true formula can be in the form of either a (cube) resolution proof or a Skolem-function model whereas that of a false formula is in the form of a (clause) resolution proof. The resolution proof and Skolem-function model are somewhat unrelated. This paper strengthens their connection by showing that, given a true QBF, its Skolem-function model is derivable from its cube-resolution proof of satisfiability as well as from its clause-resolution proof of unsatisfiability under formula negation. Consequently Skolem-function derivation can be decoupled from Skolemization-based solvers and computed from standard search-based ones. Fundamentally different from prior methods, our derivation in essence constructs Skolem functions following the variable quantification order. It permits constructing a subset of Skolem functions of interests rather than the whole, and is particularly desirable in many applications. Experimental results show the robust scalability and strong benefits of the new method.

1 Introduction

Quantified Boolean formulae (QBF) allow compact encoding of many decision problems, for example, hardware model checking [6], design rectification [17], program synthesis [18], two-player game solving [13], planning [15], and so on. QBF evaluation has been an important subject in both theoretical and practical computer sciences. Its broad applications have driven intensive efforts pursuing effective QBF solvers, despite the intractable PSPACE-complete complexity. Approaches to QBF evaluation may vary in formula representations, solving mechanisms, data structures, preprocessing techniques, etc. As a matter of fact, the advances of DPLL-style satisfiability (SAT) solving make search-based QBF evaluation [5] on prenex conjunctive normal form (PCNF) formulae the most popular approach.

As QBF evaluation procedures are much more complicated than their SAT solving counterparts, validating the results of a QBF solver is more critical than

that of a SAT solver. The commonly accepted certificate formats to date are mainly resolution proofs and Skolem-function models. More precisely, for a true QBF, a certificate can be in the syntactic form of a cube-resolution proof (e.g., available in solvers QuBE-cert [12] and yQuaffle [20]) or in the semantic form of a model consisting of a set of Skolem functions (e.g., available in sKizzo [1, 2], squolem [9], and Ebddres [9]); for a false QBF, it can be in the syntactic form of a clause-resolution proof (e.g., available in all the above solvers except for sKizzo). Despite some attempts towards a unified QBF proof checker [9], resolution proofs and Skolem-function models remain weakly related. Moreover, the asymmetry between the available certificate formats in the true and false QBF may seem puzzling.

From the application viewpoint, Skolem functions are more directly useful than resolution proofs. The Skolem-function model in solving a true QBF may correspond to, for example, a correct replacement in design rectification, a code fragment in program synthesis, a winning strategy in two-player game solving, a feasible plan in robotic planning, etc. Unfortunately, Skolem-function models are currently only derivable with Skolemization-based solvers, such as sKizzo, squolem, and Ebddres. Moreover, the derivation can be expensive as evidenced by empirical experience that Skolemization-based solvers usually take much longer time on solving true instances than false ones. In contrast, search-based solvers, such as QuBE-cert, can be more efficient and perform more symmetrically in terms of runtime on true and false instances.

This paper takes one step closer to a unified approach to QBF validation by showing that, for a true QBF, its Skolem-function model can be derived from its cube-resolution proof of satisfiability and also from its clause-resolution proof of unsatisfiability under formula negation, both in time linear with respect to proof sizes. Consequently, the aforementioned issues are addressed. Firstly, the connection between resolution proofs and Skolem functions is strongly established. Secondly, it practically conceives Skolem-function *countermodels* for false QBF, and thus yielding a symmetric view between satisfiability and unsatisfiability certifications. Finally, Skolem-function derivation can be decoupled from Skolemization-based solvers and achieved from the more popular search-based solvers, provided that resolution proofs are maintained. A key characteristic of the new derivation is that Skolem functions are generated for variables quantified from outside in, in contrast to the inside-out computation of Skolemization-based solvers. This feature gives the flexibility of computing some Skolem functions of interests, rather than all as in Skolemization-based solvers.

Experimental results show that search-based QBF solver QuBE-cert certifies more QBFEVAL instances[1] than Skolemization-based solvers sKizzo and squolem. Almost all of the Skolem-function models (respectively countermodels) are computable, under resource limits, from the cube-resolution proofs of the true cases (respectively clause-resolution proofs of the false cases). On the other

[1] Since negating the QBFEVAL formulae using Tseitin's conversion [19] may suffer from variable blow up, the Skolem functions are only derived with respect to the original formulae.

hand, for the relation determinization instances (all satisfiable), whose negations are concise by Tseitin's conversion from the circuit structures, their Skolem functions are obtained both from the cube-resolution proof of the original formulae and also from the clause-resolution proof of the negated formulae to compare. The latter tends to be much more robust and shows the unique value of the proposed method.

2 Preliminaries

A *literal* in a Boolean formula is either a variable (i.e., positive-phase literal) or the negation of the variable (i.e., negative-phase literal). In the sequel, the corresponding variable of a literal l is denoted $var(l)$. A *clause* is a Boolean formula consisting of a disjunction of a set of literals; a *cube* is a Boolean formula consisting of a conjunction of a set of literals. In the sequel, we may alternatively specify a clause or cube by a set of literals. A formula in *conjunctive normal form* (CNF) is a conjunction of a set of clauses whereas a *disjunctive normal form* (DNF) formula is a disjunction of a set of cubes. A (quantifier-free) formula ϕ over variables X subject to some truth assignment $\alpha : X' \to \{0, 1\}$ on variables $X' \subseteq X$ is denoted as $\phi|_\alpha$.

2.1 Quantified Boolean Formulae

A *quantified Boolean formula* (QBF) Φ over variables $X = \{x_1, \ldots, x_k\}$ in the *prenex conjunctive normal form* (PCNF) is of the form

$$Q_1 x_1, \ldots, Q_k x_k . \phi, \tag{1}$$

where $Q_1 x_1, \ldots, Q_k x_k$, with $Q_i \in \{\exists, \forall\}$ and variables $x_i \neq x_j$ for $i \neq j$, is called the *prefix*, denoted Φ_{prefix}, and ϕ, a quantifier-free CNF formula in terms of variables X, is called the *matrix*, denoted Φ_{matrix}. We shall assume that a QBF is in PCNF and is totally quantified, i.e., with no free variables. So the set X of variables of Φ can be partitioned into *existential variables* $X_\exists = \{x_i \in X \mid Q_i = \exists\}$ and *universal variables* $X_\forall = \{x_i \in X \mid Q_i = \forall\}$. A literal l is called an *existential literal* and a *universal literal* if $var(l)$ is in $X_\exists$ and $X_\forall$, respectively.

Given a QBF, the *quantification level* $\ell : X \to \mathbb{N}$ of variable $x_i \in X$ is defined to be the number of quantifier alternations between $\exists$ and $\forall$ from left (i.e., outer) to right (i.e., inner) plus 1. For example, the formula $\exists x_1, \exists x_2, \forall x_3, \exists x_4 . \phi$ has $\ell(x_1) = \ell(x_2) = 1$, $\ell(x_3) = 2$, and $\ell(x_4) = 3$. For convenience, we extend the definition of ℓ to literals, with $\ell(l)$ for some literal l meaning $\ell(var(l))$.

A clause C with literals $\{l_1, \ldots, l_j\}$ in a QBF Φ over variables X is called *minimal* if

$$\max_{l_i \in C, var(l_i) \in X_\forall} \{\ell(l_i)\} < \max_{l_i \in C} \{\ell(l_i)\}.$$

Otherwise, it is *non-minimal*. A non-minimal clause C can be minimized to a minimal clause C' by removing the literals

$$\{l \in C \mid var(l) \in X_\forall \text{ and } \ell(l) = \max_{l_i \in C}\{\ell(l_i)\}\}$$

from C. This process is called $\forall$-*reduction*. For a clause C of a QBF, we denote its $\forall$-reduced minimal clause as $\text{MIN}(C)$. Replacing C with $\text{MIN}(C)$ in a QBF does not change the formula satisfiability.

2.2 Q-Resolution

A clause is *tautological* if it contains both literals x and $\neg x$ of some variable x. Two non-tautological clauses C_1 and C_2 are of *distance* k if there are k variables $\{x_1, \ldots, x_k\}$ appearing in both clauses but with opposite phases. The ordinary *resolution* is defined on two clauses C_1 and C_2 of distance 1. If $C_1 = C_1' \vee x$ and $C_2 = C_2' \vee \neg x$, then resolving C_1 and C_2 on the *pivot variable* x yields the *resolvent* $C_1' \vee C_2'$.

Q-resolution [11] extends the ordinary resolution on CNF to PCNF formulae with two rules: First, only existential variables can be the pivot variables for resolution. Second, $\forall$-reduction is applied whenever possible. Unless otherwise said, "Q-resolution" is shortened to "resolution" in the sequel. In fact (Q-)resolution is a sound and complete approach to QBF evaluation.

Theorem 1 ([11]). *A QBF is false (unsatisfiable) if and only if there exists a clause resolution sequence leading to an empty clause.*

By duality, cube resolution can be similarly defined, and is also sound and complete for QBF evaluation.

Theorem 2 ([8]). *A QBF is true (satisfiable) if and only if there exists a cube resolution sequence leading to an empty cube.*

Modern search-based QBF solvers are equipped with conflict-driven learning, which performs resolution in essence. A tautological clause containing both positive and negative literals of a (universal) variable may result from resolution [21]. Since the clause is resolved from two clauses with distance greater than 1, it is referred to as *long-distance resolution*. Unlike the case in propositional satisfiability, such a clause is not totally redundant as it facilitates implication in QBF evaluation. Nevertheless, long-distance resolution is not essential, and can always be replaced by distance-1 resolution [8].

2.3 Skolemization and Skolem Functions

Any QBF Φ can be converted into the well-known *Skolem normal form* in mathematical logic, which consists of only two quantification levels, first existential and second universal. In the conversion, every existential variable x_i of Φ is replaced in Φ_{matrix} by its respective fresh function symbol, denoted $F[x_i]$, which

refers only to the universal variables x_j of Φ with $\ell(x_j) < \ell(x_i)$. These function symbols, corresponding to the so-called *Skolem functions* [16], are then existentially quantified in the first quantification level before the second level of universal quantification over the original universal variables. This conversion, called *Skolemization*, is satisfiability preserving. Essentially a QBF is true if and only if the Skolem functions of its Skolem normal form exist. (Skolemization was exploited in [1] for QBF evaluation.)

In the sequel, we shall extend the notion of Skolem functions in their dual form, also known as the *Herbrand functions*. That is, the Skolem normal form (in the dual) contains two quantification levels, first universal and second existential. For a QBF Φ, in the new notion the Skolem function $F[x_i]$ of a universal variable x_i of Φ refers only to the existential variables x_j of Φ with $\ell(x_j) < \ell(x_i)$. Essentially a QBF is false if and only if the Skolem functions of its Skolem normal form (in the dual) exist.

2.4 QBF Certificates

To validate the results of a QBF solver, resolution proofs and Skolem functions are commonly accepted certificates [12]. For a true QBF, either a cube-resolution proof or a collection of Skolem functions can certify the satisfiability. For a false QBF, a clause-resolution proof can certify the unsatisfiability. In theory, a false QBF can be negated to a true QBF, whose Skolem functions can then be used as a countermodel to the original false QBF. In practice, however, such a countermodel is hardly derivable because negation may result in substantial increase in the formula size or variable count [9]. In contrast, we show that a countermodel can be obtained without formula negation, and thus practical for certifying a false QBF.

3 Model/Countermodel Construction from Resolution Proofs

This section shows a sound and complete approach to construct Skolem functions for existential (respectively universal) variables as the model (respectively countermodel) of a true (respectively false) QBF in time linear with respect to a cube (respectively clause) resolution proof. Since cube and clause resolutions both obey similar deduction rules, we keep attention on the latter only and omit the former.

We consider (Q-)resolution proofs of QBF unsatisfiability that involve no long-distance resolution. As long-distance resolution can always be avoided and replaced by distance-1 resolution [8], our discussion is applicable in general.

Before delving into the main construction, we first define the following formula structure.

Definition 1. *A* Right-First-And-Or (RFAO) formula φ *is recursively defined by*

$$\varphi ::= \mathit{clause} \mid \mathit{cube} \mid \mathit{clause} \wedge \varphi \mid \mathit{cube} \vee \varphi, \tag{2}$$

where the symbol "::=" is read as "can be" and symbol "|" as "or".

Note that the formula is constructed in order from left to right. Due to the particular building rule of an RFAO formula with priority going to the right, we save on parentheses to enhance readability. For example, formula

$$\begin{aligned}\varphi &= clause_1 \wedge clause_2 \wedge cube_3 \vee clause_4 \wedge cube_5 \vee cube_6 \\ &= (clause_1 \wedge (clause_2 \wedge (cube_3 \vee (clause_4 \wedge (cube_5 \vee cube_6))))).\end{aligned}$$

We sometimes omit expressing the conjunction symbol "$\wedge$" and interchangeably use "$+$" for "$\vee$" in a formula.

In our discussion we shall call a clause/cube in an RFAO formula a *node* of the formula, and omit a node's subsequent operator, which can be uniquely determined. Note that the ambiguity between a single-literal clause and a single-literal cube does not occur in an RFAO formula as the clause-cube attributes are well specified in our construction.

The RFAO formula has two important properties (which will be crucial in proving Theorem 3):

1. If $node_i$ under some (partial) assignment of variables becomes a validated clause (denoted 1-clause) or falsified cube (denoted 0-cube), then we can effectively remove $node_i$ (if it is not the last) from the formula without further valuating it.
2. If $node_i$ becomes a falsified clause (denoted 0-clause) or validated cube (denoted 1-cube), then we need not further valuate (namely, can remove) all other nodes with index greater than i.

Below we elaborate how to construct the countermodel expressed by the RFAO formula from a clause-resolution proof Π of a false QBF Φ. We treat the proof Π as a directed acyclic graph (DAG) $G_\Pi(V_\Pi, E_\Pi)$, where a vertex $v \in V_\Pi$ corresponds to a clause $v.clause$ obtained in the resolution steps of Π and a directed edge $(u, v) \in E_\Pi \subseteq V_\Pi \times V_\Pi$ from the *parent* u to the *child* v indicates that $v.clause$ results from $u.clause$ through either resolution or $\forall$-reduction. The clauses of Π can be partitioned into three subsets: those in Φ_{matrix}, those resulting from resolution, and those from $\forall$-reduction. Let V_M, V_S, and V_D denote their respective corresponding vertex sets. So $V_\Pi = V_M \cup V_S \cup V_D$. Note that in G_Π a vertex in V_M has no incoming edges and is a *source* vertex; a vertex in V_S has two incoming edges from its two parent vertices; a vertex in V_D has one incoming edge from its parent vertex. On the other hand, there can be one or many *sink* vertices, which have no outgoing edges. Since the final clause of Π is an empty clause, the graph G_Π must have the corresponding sink vertex.

The intuition behind our construction stems from the following observations. Firstly, if $V_D = \emptyset$, then the quantifier-free formula Φ_{matrix} is unsatisfiable by itself, and so is Φ. Since there exists an ordinary resolution proof, which involves no $\forall$-reduction, any functional interpretation on the universal variables forms a countermodel to Φ.

Secondly, if $V_S = \emptyset$, then Φ_{matrix} must contain a clause consisting of only universal variables. With only $\forall$-reduction, Φ can be falsified. Without loss of

generality, assume this clause is $(l_1 \vee \cdots \vee l_k)$. Then letting the Skolem function of $var(l_i)$ be

$$F[var(l_i)] = \begin{cases} 0 & \text{if } l_i = var(l_i), \text{ and} \\ 1 & \text{if } l_i = \neg var(l_i), \end{cases}$$

for $i = 1, \ldots, k$, forms a countermodel of Φ. (The Skolem functions of the universal variables not in the clause are unconstrained.)

Finally, we discuss the general case where V_D and V_S are non-empty. Every clause $w.clause$ of Π with $w \in V_S$ is implied by the conjunction $u.clause \wedge v.clause$ with $(u, w), (v, w) \in E_\Pi$. (That is, the clause resulting from resolution is *unconditionally* implied by the conjunction of its parent clauses.) Even if the pivot variable of the corresponding resolution were universally quantified, the implication would still hold. So the implication is regardless of Φ_{prefix}. On the other hand, a clause $v.clause$ of Π with $v \in V_D$ is not directly implied by $u.clause$ with $(u, v) \in E_\Pi$. (That is, the clause resulting from $\forall$-reduction is *conditionally* implied by its parent clause.) Nevertheless Φ_{matrix} and $\Phi_{matrix} \wedge v.clause$ are equisatisfiable under Φ_{prefix}.

To characterize the conditions for an implication (especially between the two clauses involved in a $\forall$-reduction step) to hold, we give the following definition.

Definition 2. *Let $\alpha : X \to \{0, 1\}$ be a full assignment on variables X. Given two (quantifier-free) formulae ϕ_1 and ϕ_2 over variables X, if the implication $\phi_1 \to \phi_2$ holds under α, then we say that ϕ_2 is* α-implied *by ϕ_1.*

For a resolution proof of a false QBF Φ, when we say a clause is *α-implied*, we shall mean it is α-implied by its parent clause or by the conjunction of its parent clauses depending on whether the clause results from $\forall$-reduction or resolution. A clause resulting from resolution is surely α-implied for any α, but a clause resulting from $\forall$-reduction may not be α-implied for some α. We further say that a clause C is *α-inherited* if all of its ancestor clauses (except for the clauses of the source vertices, which have no parent clauses and are undefined under α-implication) and itself are α-implied. Clearly, if C is α-inherited, then $\Phi_{matrix}|_\alpha = (\Phi_{matrix} \wedge C)|_\alpha$.

For a false QBF Φ over variables $X = X_\exists \cup X_\forall$, let the assignment $\alpha : X \to \{0, 1\}$ be divided into $\alpha_\exists : X_\exists \to \{0, 1\}$ and $\alpha_\forall : X_\forall \to \{0, 1\}$. To construct the Skolem-function countermodel, our goal is to determine $\alpha_\forall$ for every $\alpha_\exists$ such that the empty clause of the resolution proof is α-inherited, or there exists an α-inherited clause C with $C|_\alpha = 0$. Therefore, for every assignment $\alpha_\exists$, Φ implies false. That is, such $\alpha_\forall$ provides a countermodel to Φ.

Figure 1 sketches the countermodel construction algorithm, where the Skolem functions for universal variables are computed in RFAO formulae, each of which is stored as an (ordered) array of nodes. Before proving the correctness of the algorithm, we take the following example to illustrate the computation.

Countermodel_construct
input: a false QBF Φ and its clause-resolution DAG $G_\Pi(V_\Pi, E_\Pi)$
output: a countermodel in RFAO formulae
begin
01 **foreach** universal variable x of Φ
02 `RFAO_node_array[`x`]` := $\emptyset$;
03 **foreach** vertex v of G_Π in topological order
04 **if** $v.clause$ resulting from $\forall$-reduction on $u.clause$, i.e., $(u, v) \in E_\Pi$
05 $v.cube := \neg(v.clause)$;
06 **foreach** universal variable x reduced from $u.clause$ to get $v.clause$
07 **if** x appears as positive literal in $u.clause$
08 **push** $v.clause$ to `RFAO_node_array[`x`]`;
09 **else if** x appears as negative literal in $u.clause$
10 **push** $v.cube$ to `RFAO_node_array[`x`]`;
11 **if** $v.clause$ is the empty clause
12 **foreach** universal variable x of Φ
13 simplify `RFAO_node_array[`x`]`;
14 **return** `RFAO_node_array`'s;
end

Fig. 1. Algorithm: Countermodel Construction

Example 1. Let Φ be a false QBF and Π be its resolution proof of unsatisfiability as below.

$$\Phi_{prefix} = \exists a \forall x \exists b \forall y \exists c$$
$$\Phi_{matrix} = (a \vee b \vee y \vee c)(a \vee x \vee b \vee y \vee \neg c)(x \vee \neg b)(\neg y \vee c)(\neg a \vee \neg x \vee b \vee \neg c)$$
$$(\neg x \vee \neg b)(a \vee \neg b \vee \neg y)$$
$$\Pi = \left\{ \begin{array}{l} 1.\ clause_8 = resolve(clause_1, clause_2) \\ 2.\ clause_9 = resolve(clause_3, clause_8) \\ 3.\ clause_{10} = resolve(clause_4, clause_5) \\ 4.\ clause_{11} = resolve(clause_{10}, clause_6) \\ 5.\ clause_{empty} = resolve(clause_{11}, clause_9) \end{array} \right\}$$

Note that the $\forall$-reduction steps are omitted in Π, which however can be easily filled in as shown in Figure 2, where the clauses are indexed by subscript numbers, and the $\forall$-reduction steps are indexed by the parenthesized numbers indicating the relative order.

By following the steps of the *Countermodel_construct* algorithm in Figure 1, the RFAO node-array contents after each $\forall$-reduction step in the proof of Figure 2 are listed in order of appearance in Figure 3. The resultant Skolem functions for universal variables x and y are

$$F[x] = (a) \wedge (a) = a, \text{ and}$$
$$F[y] = (\neg ab) \vee ((a \vee x \vee b) \wedge (ax \neg b)),$$

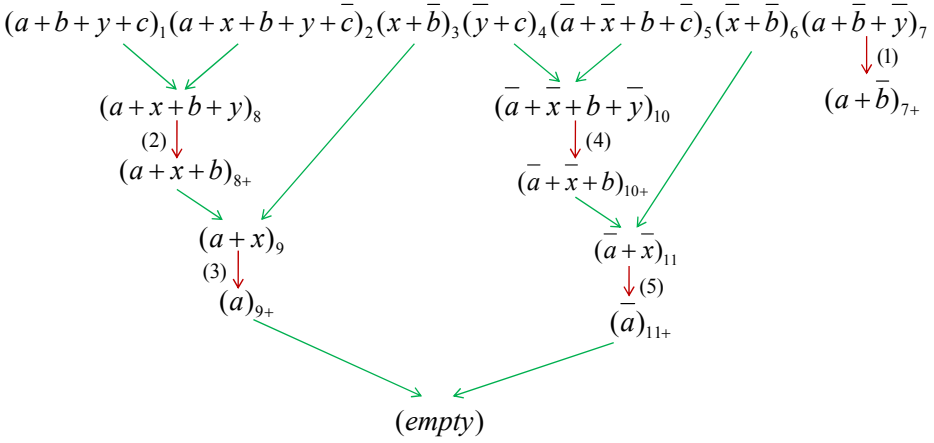

Fig. 2. DAG of resolution proof Π

respectively. Note that the computed $F[y]$ depends on variable x, which can always be eliminated by substituting $F[x]$ for x in $F[y]$. In fact, keeping such dependency may be beneficial as the countermodel can be represented in a multi-level circuit format with shared logic structures. Moreover, observe that clause 7, namely $(a \vee \neg b \vee \neg y)$, is not involved in the resolution steps leading to the empty clause. Its existence is optional in constructing the countermodel, and can be treated as don't cares for countermodel simplification. It can be verified that, for any assignment to variables a, b, and c, formula Φ_{matrix} with variables x and y substituted with $F[x]$ and $F[y]$, respectively, is false.

The correctness of the *Countermodel_construct* algorithm of Figure 1 is asserted below.

Theorem 3. *Given a false QBF Φ and a DAG G_Π corresponding to its resolution proof Π of unsatisfiability, the algorithm* Countermodel_construct*(Φ, G_Π) produces a correct countermodel for the universal variables of Φ.*

Proof. We show that, under every assignment $\alpha_\exists$ to existential variables of Φ, our constructed countermodel always induces some $\alpha_\forall$ such that $\Phi_{matrix}|_\alpha = 0$. There are two possible cases under every such α.

First, assume every clause $v.clause$ with $v \in V_D$ is α-implied. Then the empty clause must be α-inherited because other clauses resulting from resolution are always α-implied. Thus $\Phi_{matrix}|_\alpha = 0$.

Second, assume not every clause $v.clause$ with $v \in V_D$ is α-implied. Let $C_{\alpha\_violate}$ be the set of all such clauses violating α-implication. Suppose $v.clause \in C_{\alpha\_violate}$ is obtained by $\forall$-reduction from $u.clause$ with $(u, v) \in E_\Pi$ on some universal variables. Let $C_{u \setminus v}$ denote the subclause of $u.clause$ consisting of exactly the reduced literals in the $\forall$-reduction leading to $v.clause$. Then $v.clause$ must satisfy the criteria

0.	$x : [\,]$	$y : [\,]$
1.	$x : [\,]$	$y : [\,cube(\neg ab)\,]$
2.	$x : [\,]$	$y : \begin{bmatrix} cube(\neg ab), \\ clause(a \vee x \vee b) \end{bmatrix}$
3.	$x : [\,clause(a)\,]$	$y : \begin{bmatrix} cube(\neg ab), \\ clause(a \vee x \vee b) \end{bmatrix}$
4.	$x : [\,clause(a)\,]$	$y : \begin{bmatrix} cube(\neg ab), \\ clause(a \vee x \vee b), \\ cube(ax\neg b) \end{bmatrix}$
5.	$x : \begin{bmatrix} clause(a), \\ cube(a) \end{bmatrix}$	$y : \begin{bmatrix} cube(\neg ab), \\ clause(a \vee x \vee b), \\ cube(ax\neg b) \end{bmatrix}$

Fig. 3. Contents of RFAO node arrays

1. $v.clause|_\alpha = 0$ (otherwise $v.clause$ would be α-implied), and
2. $C_{u \setminus v}|_{\alpha\forall} = 1$ (otherwise $v.clause$ would have the same value as $u.clause$ and thus be α-implied).

It remains to show that, even if $C_{\alpha\_violate}$ is non-empty, there still exists some α-inherited clause C with $C|_\alpha = 0$, i.e., an induced empty clause under α.

Notice that algorithm *Countermodel_ construct* processes G_Π in a topological order, meaning that a clause in the resolution proof is processed only after all of its ancestor clauses are processed. Now we consider all clauses $v.clause$ with $v \in V_D$ in the topological order under the assignment α. Let $v'.clause$ be the first clause encountered with $v'.clause|_\alpha = 0$. (If there is no such $v'.clause$ under α, then it corresponds to the situation analyzed in the first case.) For every universal variable x being reduced from the parent clause $u'.clause$ of $v'.clause$, i.e., $(u', v') \in E_\Pi$, we examine its corresponding `RFAO_node_array[`x`]`. Suppose v' is the ith enumerated vertex that results from $\forall$-reduction involving the reduction of variable x. By the aforementioned two properties of the RFAO formula and by the way how `RFAO_node_array[`x`]` is constructed, we know that the Skolem function value of $F[x]$ under α is not determined by the first $i-1$ nodes, but by the ith node of `RFAO_node_array[`x`]`. In addition, the function value $F[x]$ makes the literal of variable x in clause $C_{u' \setminus v'}$ valuate to false. Because every literal in $C_{u' \setminus v'}$ is valuated to false, we have $u'.clause|_\alpha = 0$ and thus $v'.clause$ is α-implied. Moreover, since $v'.clause$ is the first clause encountered with $v'.clause|_\alpha = 0$, all its ancestor clauses must be α-implied. So $v'.clause$ is α-inherited, and thus $\Phi_{matrix}|_\alpha = 0$.

Because every assignment $\alpha_\exists$ together with the corresponding induced assignment $\alpha_\forall$ makes $\Phi_{matrix}|_\alpha = 0$, the Skolem functions computed by algorithm *Countermodel_ construct* form a correct countermodel to Φ. ■

Proposition 1. *Given a false QBF Φ and its resolution proof of unsatisfiability, let $F[x]$ be the Skolem function computed by algorithm* Countermodel_construct *for the universal variable x in Φ. Then $F[x]$ refers to some variable y in Φ only if $\ell(y) < \ell(x)$.*

Note that, by the above strict inequality, Proposition 1 asserts that no cyclic dependency arises among the computed Skolem functions.

In fact algorithm *Countermodel_ construct* of Figure 1 can be easily modified to compute the Skolem functions for some (not all) of the universal variables of a given QBF. Let k be the maximal quantification level among the universal variables whose Skolem functions are of interests. Then, by Proposition 1, algorithm *Countermodel_ construct* only needs to maintain RFAO node arrays for universal variables with quantification level no greater than k. For Skolemization-based solvers, this partial derivation is not possible because Skolem functions (for existential variables) are constructed on-the-fly during QBF solving, whereas our construction is performed after the entire proof is done.

Proposition 2. *Given a false QBF and its resolution proof of unsatisfiability, algorithm* Countermodel_construct *computes the countermodel in time linear with respect to the proof size.*

Proposition 3. *The RFAO formula size (in terms of nodes) for each universal variable computed by algorithm* Countermodel_construct *is upper bounded by the number of $\forall$-reduction steps in the resolution proof.*

The resolution proofs provided by search-based QBF solvers often contain (redundant) resolution steps unrelated to yielding the final empty clause. Algorithm *Countermodel_ construct* works for resolution proofs with and without redundant steps. Since a highly redundant proof may degrade the performance of the algorithm, it may be desirable to trim away redundant parts before countermodel construction. On the other hand, as illustrated in Example 1, it may be possible to exploit the redundancy for countermodel simplification.

The above discussion, concerned about countermodel construction, can be straightforwardly extended under the duality principle to model construction of a true QBF from its cube-resolution proof of satisfiability. We omit similar exposition.

4 Applications to Boolean Relation Determinization

We relate Skolem functions to the problem of *Boolean relation determinization*, which is useful in logic and property synthesis [9, 10]. A *Boolean relation* over *input variables* X and *output variables* Y is a characteristic function $R : \{0,1\}^{|X|+|Y|} \rightarrow \{0,1\}$ such that assignments $a \in \{0,1\}^{|X|}$ and $b \in \{0,1\}^{|Y|}$

make $R(a, b) = 1$ if and only if (a, b) is in the relation. Relations can be exploited to specify the permissible (non-deterministic) behavior of a system, by restricting its allowed input X and output Y combinations. To be implemented with circuits, a relation has to be *determinized* in the sense that each output variable $y_i \in Y$ can be expressed by some function $f_i : \{0,1\}^{|X|} \rightarrow \{0,1\}$. Formally it can be written as a QBF $\forall X, \exists Y.R(X, Y)$, and the determinization problem corresponds to finding the Skolem functions of variables Y.

Often the formula $R(X, Y)$ is not in CNF, but rather in some circuit structure. By Tseitin's transformation, it can be rewritten in CNF $\phi_R(X, Y, Z)$ with the cost of introducing some new intermediate variables Z. Therefore the QBF is rewritten as $\forall X, \exists Y, \exists Z.\phi_R(X, Y, Z)$. By our model construction, the Skolem functions can be computed from its cube-resolution proof of satisfiability. Alternatively, we may compute the Skolem functions by finding the countermodel of $\exists X, \forall Y.\neg R(X, Y)$, which can be again by Tseitin's transformation translated into PCNF $\exists X, \forall Y, \exists Z'.\phi_{\neg R}(X, Y, Z')$ with Z' being the newly introduced intermediate variables in the circuit of $\neg R(X, Y)$. Note that after the negation, the number of quantification levels increases from two to three; on the other hand, ϕ_R and $\phi_{\neg R}$ can be simplified to have the same number of clauses and $|Z| = |Z'|$. The above two approaches are to be studied in the experiments.

It is interesting to note that, since the quantification order of a QBF affects the support variables of a Skolem functions, QBF prefix reordering may be exploited to synthesize Skolem functions with some desired variable dependencies. Moreover, in addition to the relation determinization application, the duality between model and countermodel construction may be useful in other applications whose original formulae are in circuit representation.

5 Experimental Results

The proposed method, named ResQu, was implemented in the C++ language. The experiments were conducted on a Linux machine with a Xeon 2.53 GHz CPU and 48 GB RAM for two sets of test cases: the QBF evaluation benchmarks downloaded from [14] and relation determinization ones modified from [3].

We compared various Skolem-function derivation scenarios using QBF solvers with certification capability, including sKizzo [1], squolem [9], and QuBE-cert.[2] For true QBF instances, sKizzo and squolem were applied to obtain Skolem-function models whereas the cube-resolution proofs produced by QuBE-cert were converted to Skolem-function models by ResQu. For false QBF instances, sKizzo was applied on the negated formulae to obtain Skolem-function countermodels whereas the clause-resolution proofs produced by squolem and QuBE-cert were converted to Skolem-function countermodels by ResQu.

Table 1 summarizes the results of our first experiment on the QBFEVAL'05 and QBFEVAL'06 test sets, which contain 211 and 216 instances, respectively. In

[2] We did not experiment with Ebddres [9] and yQuaffle [20] as the former tends to generate larger certificates for false QBF compared to squolem, and the latter has characteristics similar to QuBE-cert.

Table 1. Summary for QBFEVAL Benchmarks.

		overall	sKizzo		squolem+ResQu				QuBE-cert+ResQu			
		#sv	#sv	time (sv)	#sv	time (sv)	#md	time (md)	#sv/#pg	time (sv)	#md	time (md)
true	'05	84	69	1707.27	50	1490.84	—	—	19/19	414.65	19	54.73
	'06	48	29	295.24	25	199.79	—	—	44/44	859.64	44	152.22
	total	132	98	2002.51	75	1690.63	—	—	63/63	1274.29	63	206.95
false	'05	77	0	0	42	1467.45	42	12.60	46/25	2369.91	25	12.99
	'06	29	0	0	9	85.96	9	0.80	28/22	916.57	22	2.34
	total	106	0	0	51	1553.41	51	13.40	74/47	3286.48	47	15.33

#sv: number of instances solved; #pg: number of proofs involving no long-distance resolution; #md: number of (counter)models generated by ResQu; time (sv/md): CPU time in seconds for QBF evaluation/(counter)model generation; —: data not available due to inapplicability of ResQu

the experiment, all the QBF solvers, including sKizzo, squolem, and QuBE-cert, are given a 600-second time limit and a 1-GB memory limit for solving each instance. Under the given resource limits, all solvers, together, solved 132 true and 106 false instances. All the (counter)models produced by ResQu were verified using MiniSat [7] while the models produced by sKizzo and squolem were assumed correct without verification.

It should be mentioned that the resolution proofs produced by QuBE-cert were not simplified, that is, including resolution steps unrelated to producing the final empty clause (or empty cube). The unrelated resolution steps were first removed (with runtime omitted) before the (counter)model construction of ResQu. Moreover, approximately 20% of all the proofs involved long-distance resolution, and ResQu did not construct their (counter)models. On the other hand, the clause-resolution proofs produced by squolem were simplified already and involved no long-distance resolution. Hence ResQu had no problems constructing their countermodels.

We compared the numbers of instances whose (counter)models generated by ResQu and by other tools. When models are concerned, ResQu (via the proofs from QuBE-cert) covered 63 (19 in QBFEVAL'05 and 44 in QBFEVAL'06), whereas sKizzo and squolem in combination covered 105 (75 in QBFEVAL'05 and 30 in QBFEVAL'06). When countermodels are concerned, ResQu (via the proofs from squolem and QuBE-cert) covered 83 (60 in QBFEVAL'05 and 23 in QBFEVAL'06), whereas sKizzo covered 0.[3] Notably, ResQu circumvents the DNF-to-CNF conversion problem and is unique in generating countermodels.

While all the (counter)models can be constructed efficiently (for proofs without long-distance resolution), some of them can be hard to verify. In fact, about 84% of the 161 (counter)models constructed by ResQu were verified within 1 second using MiniSAT; there are 5 models of the true instances in QBFEVAL'06 that remained unverifiable within 1000 seconds.

[3] In addition to sKizzo, in theory squolem can also compute Skolem-function countermodels of false QBF instances by formula negation. We only experimented with sKizzo, which can read in DNF formulae and thus requires no external DNF-to-CNF conversion, arising due to formula negation. Although squolem is not experimented in direct countermodel generation by formula negation, prior experience [9] suggested that it might be unlikely to cover much more cases than sKizzo.

Table 2. Results for Relation Determinization Benchmarks.

		(#in, #out, #e, #C)	sKizzo		squolem+ResQu		QuBE-cert+ResQu	
			time (sv)	size	time (sv/md/vf)	size	time (sv/md/vf)	size
true	1	(7, 3, 55, 322)	0.09	377	(0.06 , —, —)	134	(0.03, 0.01, 0.01)	28
	2	(20, 10, 963, 5772)	0.86	1311	(0.79, —, —)	1378	(0.12, 0.03, 0.02)	118
	3	(21, 9, 1280, 7672)	NA		(NA, —, —)		(5.28, 1.74, 1.23)	148883
	4	(24, 12, 1886, 11300)	NA		(1.23, —, —)	179	(0.94, 0.11, 0.03)	1947
	5	(28, 14, 1833, 10992)	NA		(NA, —, —)		(0.35, 0.05, 0.02)	61
	6	(32, 16, 3377, 20250)	NA		(NA, —, —)		(1.21, 0.18, 0.04)	1193
	7	(36, 18, 5894, 35354)	NA		(NA, —, —)		(0.23, 0.15, 0.03)	91
	8	(42, 20, 6954, 41718)	NA		(NA, —, —)		(0.27, 0.12, 0.03)	3
	9	(39, 19, 9823, 58932)	NA		(NA, —, —)		(3.08, 0.50, 0.01)	307
	10	(46, 22, 10550, 63294)	NA		(NA, —, —)		(1.89, 0.25, 0.01)	58
	11	(49, 19, 11399, 68384)	NA		(NA, —, —)		(NA, NA, NA)	
	12	(32, 18, 13477, 80856)	NA		(NA, —, —)		(NA, NA, NA)	
	13	(50, 24, 14805, 88822)	NA		(NA, —, —)		(3.14, 0.77, 0.05)	3458
	14	(53, 25, 16037, 96216)	NA		(NA, —, —)		(3.41, 0.35, 0.02)	283
	15	(56, 26, 19700, 118194)	NA		(NA, —, —)		(8.10, 1.10, 0.05)	905
	16	(59, 27, 26117, 156696)	NA		(NA, —, —)		(3.85, 0.59, 0.03)	187
	17	(65, 29, 29038, 174222)	NA		(NA, —, —)		(7.16, 0.88, 0.05)	232
	18	(62, 28, 30294, 181756)	NA		(NA, —, —)		(9.29, 1.32, 0.05)	731
	19	(72, 32, 35806, 214828)	NA		(NA, —, —)		(NA, NA, NA)	
	20	(68, 30, 50513, 303070)	NA		(NA, —, —)		(2.97, 0.62, 0.05)	11
	21	(95, 35, 57717, 346294)	NA		(NA, —, —)		(NA, NA, NA)	
	22	(41, 23, 89624, 537738)	NA		(NA, —, —)		(NA, NA, NA)	
false	1	(7, 3, 55, 322)	NA		(0.03, 0.01, 0.01)	6	(0.05, NA, NA)	
	2	(20, 10, 963, 5772)	NA		(1.14, 0.02, 0.01)	53	(0.13, NA, NA)	
	3	(21, 9, 1280, 7672)	NA		(0.20, 0.02, 0.01)	4	(1.19, NA, NA)	
	4	(24, 12, 1886, 11300)	NA		(0.31, 0.02, 0.03)	0	(0.30, NA, NA)	
	5	(28, 14, 1833, 10992)	NA		(0.29, 0.02, 0.01)	3	(1.02, NA, NA)	
	6	(32, 16, 3377, 20250)	NA		(1.95, 0.04, 0.03)	3	(0.95, NA, NA)	
	7	(36, 18, 5894, 35354)	NA		(3.08, 0.06, 0.05)	3	(4.22, NA, NA)	
	8	(42, 20, 6954, 41718)	NA		(3.23, 0.07, 0.06)	3	(9.15, NA, NA)	
	9	(39, 19, 9823, 58932)	NA		(9.41, 0.11, 0.08)	5	(10.01, NA, NA)	
	10	(46, 22, 10550, 63294)	NA		(9.87, 0.15, 0.07)	3	(3.62, NA, NA)	
	11	(49, 19, 11399, 68384)	NA		(8.33, 0.20, 0.08)	3	(14.09, NA, NA)	
	12	(32, 18, 13477, 80856)	NA		(10.42, 0.23, 0.10)	3	(10.41, NA, NA)	
	13	(50, 24, 14805, 88822)	NA		(15.82, 0.25, 0.10)	4	(509.84, NA, NA)	
	14	(53, 25, 16037, 96216)	NA		(23.65, 0.27, 0.11)	5	(7.19, NA, NA)	
	15	(56, 26, 19700, 118194)	NA		(30.18, 0.35, 0.14)	3	(25.33, NA, NA)	
	16	(59, 27, 26117, 156696)	NA		(74.19, 0.43, 0.14)	3	(202.80, NA, NA)	
	17	(65, 29, 29038, 174222)	NA		(46.90, 0.42, 0.21)	0	(24.45, NA, NA)	
	18	(62, 28, 30294, 181756)	NA		(84.48, 0.46, 0.25)	4	(94.93, NA, NA)	
	19	(72, 32, 35806, 214828)	NA		(129.84, 0.41, 0.22)	3	(80.12, NA, NA)	
	20	(68, 30, 50513, 303070)	NA		(363.12, 0.70, 7.31)	3	(26.14, NA, NA)	
	21	(95, 35, 57717, 346294)	NA		(359.40, 0.96, 8.15)	2	(86.10, NA, NA)	
	22	(41, 23, 89624, 537738)	NA		(NA, NA, NA)		(142.24, NA, NA)	

#in: number of input variables in the relation; #out: number of output variables in the relation; #e: number of innermost existential variables added due to circuit-to-CNF conversion; #C: number of clauses in final CNF; size: number of AIG nodes after performing ABC command `dc2` with negligible runtime not shown; time (sv/md/vf): CPU time in seconds for QBF evaluation/(counter)model generation/verification; NA: data not available due to computation out of resource limit; —: data not available due to inapplicability of ResQu

Table 3. Summary for Relation Determinization Benchmarks.

	overall	sKizzo		squolem+ResQu				QuBE-cert+ResQu			
	#sv	#sv	time (sv)	#sv	time (sv)	#md	time (md)	#sv/#pg	time (sv)	#md	time (md)
true	17	2	0.95	3	2.08	—	—	17/17	51.32	17	8.77
false	22	0	0	21	1175.84	21	5.20	22/0	1254.28	0	0

(Legend same as in Table 1)

Table 2 shows the results of our second experiment on 22 relation determinization benchmarks. All the original 22 instances are true (satisfiable). We compared their models obtained in two ways: by direct model construction from the satisfiability proofs of the original formulae and by indirect model construction from the unsatisfiability proofs of their negations. Unlike the QBFEVAL cases, negating these formulae by Tseitin's transformation does not result in variable- and clause-increase, as discussed in Section 4. The experiment was conducted under the resource limit same as before. For the original instances, ResQu could have generated Skolem functions only for the existential variables of interests (namely, the output variables of a Boolean relation rather than the intermediate variables), but it generated all for the verification purpose.

As summarized in Table 3, for the true cases, sKizzo and squolem in combination can construct models for only 3 instances, whereas from the 17 proofs of QuBE-cert, ResQu can generate (and verify) all models. For the negated cases, all the proofs provided by QuBE-cert involved long-distance resolution, so ResQu did not construct their countermodels. Nevertheless, squolem solved 21 out of 22 instances, and ResQu can generate (and verify) all their countermodels (i.e., models for the original QBF). It is interesting to see that, in the relation determinization application, countermodel generation for the negated formulae can be much easier than model generation for the original formulae. It reveals the essential value of ResQu.

6 Conclusions and Future Work

A new approach has been proposed to compute Skolem functions in the context of QBF evaluation. As a result, Skolem-function derivation is decoupled from Skolemization-based solvers, and is available from standard search-based solvers, provided that proper resolution proofs are given. The approach gives a balanced and unified view on certifying both true and false QBF using models and countermodels. Moreover, its practical value has been strongly supported by experiments. As Skolem functions can be important in various areas, we hope our results may encourage and enable QBF applications. Our on-going work is to extract Skolem functions from proofs with the presence of long-distance resolution.

Acknowledgments

The authors are grateful to Roderick Bloem and Georg Hofferek for providing the relation determinization benchmarks. This work was supported in part by the National Science Council under grants NSC 99-2221-E-002-214-MY3 and NSC 99-2923-E-002-005-MY3.

References

1. M. Benedetti. Evaluating QBFs via Symbolic Skolemization. In *Proc. Int'l Conf. on Logic for Programming, Artificial Intelligence and Reasoning* (LPAR), 2004.

2. M. Benedetti. Extracting Certificates from Quantified Boolean Formulas. In *Proc. Int'l Joint Conf. on Artificial Intelligence* (IJCAI), 2005.
3. R. Bloem, S. Galler, B. Jobstmann, N. Piterman, A. Pnueli, and M. Weiglhofer. Automatic Hardware Synthesis from Specifications: A Case Study. In *Proc. Design Automation and Test in Europe*, 2007.
4. Berkeley Logic Synthesis and Verification Group. *ABC: A System for Sequential Synthesis and Verification.* `http://www.eecs.berkeley.edu/~alanmi/abc/`
5. M. Cadoli, M. Schaerf, A. Giovanardi, M. Giovanardi. An Algorithm to Evaluate Quantified Boolean Formulae and Its Experimental Evaluation. *Journal of Automated Reasoning*, 28(2):101-142, 2002.
6. N. Dershowitz, Z. Hanna and J. Katz. Bounded Model Checking with QBF. In *Proc. Int'l Conf. on Theory and Applications of Satisfiability Testing* (SAT), 2005.
7. N. Eén and N. Sörensson. An Extensible SAT-Solver. In *Proc. Int'l Conf. on Theory and Applications of Satisfiability Testing* (SAT), pp. 502-518, 2003.
8. E. Giunchiglia, M. Narizzano, and A. Tacchella. Clause-Term Resolution and Learning in Quantified Boolean Logic Satisfiability. *Artificial Intelligence Research*, 26:371-416, 2006.
9. T. Jussila, A. Biere, C. Sinz, D. Kröning, and C. Wintersteiger. A First Step Towards a Unified Proof Checker for QBF. In *Proc. Int'l Conf. on Theory and Applications of Satisfiability Testing* (SAT), pp. 201-214, 2007.
10. J.-H. R. Jiang, H.-P. Lin, and W.-L. Hung. Interpolating Functions from Large Boolean Relations. In *Proc. Int'l Conf. on Computer-Aided Design* (ICCAD), pp., 779-784, 2009.
11. H. Kleine-Büning, M. Karpinski and A. Flögel. Resolution for Quantified Boolean Formulas. *Information and Computation*, 117(1):12-18, 1995.
12. M. Narizzano, C. Peschiera, L. Pulina, and A. Tacchella. Evaluating and Certifying QBFs: A Comparison of State-of-the-Art Tools. In *AI Communications*, 2009.
13. C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.
14. QBF Solver Evaluation Protal. `http://www.qbflib.org/qbfeval/`
15. J. Rintanen. Constructing Conditional Plans by a Theorem-Prover. *Journal of Artificial Intelligence Research*, 10:323-352, 1999.
16. Th. Skolem. *Über die Mathematische Logik. Norsk. Mat. Tidsk.*, 10:125-142, 1928. [Translation in *From Frege to Gödel, A Source Book in Mathematical Logic*, J. van Heijenoort, Harvard Univ. Press, 1967.]
17. S. Staber and R. Bloem. Fault Localization and Correction with QBF. In *Proc. Int'l Conf. on Theory and Applications of Satisfiability Testing* (SAT), pp. 355-368, 2007.
18. A. Solar-Lezama, L. Tancau, R. Bodík, S. Seshia, and V. Saraswat. Combinatorial Sketching for Finite Programs. In *Proc. Int'l Conf. on Architectural Support for Programming Languages and Operating Systems* (ASPLOS), pp. 404-415, 2006.
19. G. Tseitin. On the Complexity of Derivation in Propositional Calculus. *Studies in Constructive Mathematics and Mathematical Logic*, pp. 466-483, 1970.
20. Y. Yu and S. Malik. Validating the Result of a Quantified Boolean Formula (QBF) Solvers: Theory and Practice. In *Proc. Asia and South Pacific Design Automation Conference*, 2005.
21. L. Zhang and S. Malik. Conflict Driven Learning in a Quantified Boolean Satisfiability Solver. In *Proc. Int'l Conf. on Computer-Aided Design* (ICCAD), pp. 442-449, 2002.