Noname manuscript No.
(will be inserted by the editor)

Unified QBF Certification and its Applications

Valeriy Balabanov · Jie-Hong R. Jiang

Received: date / Accepted: date

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. To date the certificate of a true formula can be in the form of either a syntactic cube-resolution proof or a semantic Skolem-function model whereas that of a false formula is only in the form of a syntactic clause-resolution proof. The semantic certificate for a false QBF is missing, and the syntactic and semantic certificates are somewhat unrelated. This paper identifies the missing Herbrand-function countermodel for false QBF, and strengthens the connection between syntactic and semantic certificates 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 special Skolemization-based solvers and computed from standard search-based ones. Experimental results show strong benefits of the new method.

Keywords Quantified Boolean formula · Resolution · Skolem function · Herbrand function

This work is an extended version of [4] and was supported in part by the National Science Council under grants NSC 99-2221-E-002-214-MY3, 99-2923-E-002-005-MY3, and 100-2923-E-002-008. VB has been partially supported by the Taiwan Scholarship Program.

V. Balabanov
Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan
E-mail: balabasik@gmail.com

J.-H. R. Jiang
Department of Electrical Engineering / Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan
E-mail: jhjiang@cc.ee.ntu.edu.tw

1 Introduction

Quantified Boolean formulae (QBF) allow compact encoding of many decision problems, for example, hardware model checking [7], design rectification [18], program synthesis [19], two-player game solving [14], planning [16], 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 [6] 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 [13] and yQuaffle [21]) or in the semantic form of a model consisting of a set of Skolem functions (e.g., available in sKizzo [1,2], squolem [10], and Ebddres [10]); 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 early attempts towards a unified QBF proof checker [10], 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 of 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 Skolemization-based solvers usually take much longer time on solving true instances than solving 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 further step 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 Herbrand-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 standard 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 a subset of Skolem functions of interest, rather than computing 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 Herbrand-function countermodels) are computable, under resource limits, from the cube-resolution proofs of the true formulae (respectively clause-resolution proofs of the false formulae). On the other hand, for the Boolean relation instances of [3] (true QBF), 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.

The rest of this paper is organized as follows. After the preliminaries given in Section 2, our main results on Herbrand-function derivation from clause-resolution proofs and Skolem-function derivation from cube-resolution proofs are presented in Sections 3 and 4, respectively. Section 5 discusses the issue of long-distance resolution. Applications to relation determinization in logic synthesis are elaborated in Section 6. After experimental evaluation summarized in Section 7, Section 8 concludes this paper.

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, we shall denote the corresponding variable of a literal l as $var(l)$. 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 $\phi|_\alpha$.

[1] Since negating the QBFEVAL formulae using Tseitin's conversion [20] may suffer from variable blow up, the Skolem functions are only derived with respect to the original formulae.

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

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

where $Q_1 x_1 \cdots Q_k x_k$, with $Q_i \in \{\exists, \forall\}$ and variables $x_i \neq x_j$ for $i \neq j$, is called the *prefix*, denoted Φ_{pfx}, and ϕ, a quantifier-free CNF formula in terms of variables X, is called the *matrix*, denoted Φ_{mtx}. 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. An 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 [12] 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 ([12]) *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 ([9]) *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 [22]. 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 [9].

2.3 Skolemization and Skolem Functions

A QBF Φ with variables X 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 appearance of $x_i \in X_\exists$ in Φ_{mtx} is replaced by its respective fresh function symbol, denoted $F_S[x_i]$, which refers only to $x_j \in X_\forall$ with $\ell(x_j) < \ell(x_i)$. These function symbols, corresponding to the so-called *Skolem functions* [17], 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. It was exploited in [1] for QBF evaluation. Essentially a QBF is true if and only if the Skolem functions of its Skolem normal form exist.

Example 1 Skolemizing the QBF

$$\forall x_1 \exists y_1 \forall x_2 \exists y_2 . (x_1 \vee y_1 \vee \neg y_2)(\neg x_1 \vee \neg x_2 \vee y_2)$$

yields

$$\exists F_S[y_1] \exists F_S[y_2] \forall x_1 \forall x_2 . (x_1 \vee F_S[y_1] \vee \neg F_S[y_2])(\neg x_1 \vee \neg x_2 \vee F_S[y_2])$$

where $F_S[y_1]$ is a 1-ary function symbol referring to x_1, and $F_S[y_2]$ is a 2-ary function symbol referring to x_1 and x_2. As can be verified, $F_S[y_1] = \neg x_1$ and $F_S[y_2] = x_1 \wedge x_2$, for instance, are legitimate Skolem functions.

In the sequel, we shall extend the notion of Skolem functions in their dual form, also known as the *Herbrand functions*. For a QBF Φ with variables X, in the new notion the Herbrand function $F_H[x_i]$ of variable $x_i \in X_\forall$ refers only to $x_j \in X_\exists$ with $\ell(x_j) < \ell(x_i)$. Essentially QBF Φ is false if and only if its Herbrand functions exist such that substituting $F_H[x_i]$ for $x_i \in X_\forall$ in Φ_{mtx} makes the new formula unsatisfiable.

Example 2 The QBF

$$\forall x_1 \exists y_1 \forall x_2 \exists y_2 . \phi$$

is false if and only if the formula

$$\exists F_H[x_1] \exists F_H[x_2] \exists y_1 \exists y_2 . \phi'$$

is unsatisfiable, where $F_H[x_1]$ is a 0-ary function symbol, $F_H[x_2]$ is a 1-ary function symbol referring to y_1, and ϕ' is the formula derived from ϕ with every occurrence of x_i in ϕ substituted with $F_H[x_i]$.

2.4 QBF Certificates

To validate the results of a QBF solver, resolution proofs and Skolem functions are commonly accepted certificates [13]. For a true QBF, either a cube-resolution proof or a Skolem-function model can certify its satisfiability. For a false QBF, a clause-resolution proof can certify its 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 [10]. In contrast, we show that a Herbrand-function countermodel can be obtained without formula negation, and thus is practical for certifying a false QBF.

3 Countermodel Construction from Clause-Resolution Proofs

This section shows a sound and complete approach to construct Herbrand functions for universal variables as the countermodel of a false QBF in time linear with respect to a clause-resolution proof of unsatisfiability.

We consider 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 [9], our discussion is applicable in general.[2]

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".

[2] Rewriting a proof involving long-distance resolution to one with only distance-1 resolution might potentially increase the proof size exponentially.

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_{mtx}, 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 more *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_{mtx} 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 legitimate countermodel to Φ.

Secondly, if $V_S = \emptyset$, then Φ_{mtx} 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 Herbrand function of $var(l_i)$ be

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

for $i = 1, \ldots, k$, forms a countermodel of Φ. (The Herbrand 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_{pfx}. 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_{mtx} and $\Phi_{\text{mtx}} \wedge v.clause$ are equisatisfiable under Φ_{pfx}.

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, and ϕ_1 *α-implies* ϕ_2.

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 every 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 $\Phi_{\text{mtx}}|_\alpha = (\Phi_{\text{mtx}} \wedge C)|_\alpha$.

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 Herbrand-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 Herbrand functions 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_Π(V_Π, E_Π)
  output: a countermodel in RFAO formulae
  begin
  01  foreach universal variable x of Φ
  02    RFAO_node_array[x] := ∅;
  03  foreach vertex v of G_Π in topological order
  04    if v.clause resulting from ∀-reduction on u.clause, i.e., (u, v) ∈ E_Π
  05      v.cube := ¬(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 3 Let Φ be a false QBF and Π be its resolution proof of unsatisfiability as below.

$$\Phi_{\text{pfx}} = \exists a \forall x \exists b \forall y \exists c$$

$$\Phi_{\text{mtx}} = (a \vee b \vee y \vee c)(a \vee x \vee b \vee y \vee \neg c)(x \vee \neg b)(\neg y \vee c)(\neg a \vee \neg x \vee b \vee \neg c)$$
$$(\neg x \vee \neg b)(a \vee \neg b \vee \neg y)$$

$$\Pi = \left\{ \begin{array}{l} 1.\ clause_8 = resolve(clause_1, clause_2) \\ 2.\ clause_9 = resolve(clause_3, clause_8) \\ 3.\ clause_{10} = resolve(clause_4, clause_5) \\ 4.\ clause_{11} = resolve(clause_{10}, clause_6) \\ 5.\ clause_{\text{empty}} = resolve(clause_{11}, clause_9) \end{array} \right\}$$

Note that the ∀-reduction steps are omitted in Π. They however can be easily recovered as shown in Figure 2, where the clauses are indexed by the subscript numbers and the ∀-reduction steps are indexed by the parenthesized numbers indicating their derivation order.

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

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

respectively. Note that the computed $F_H[y]$ depends on variable x, which can always be eliminated by substituting $F_H[x]$ for x in $F_H[y]$. 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

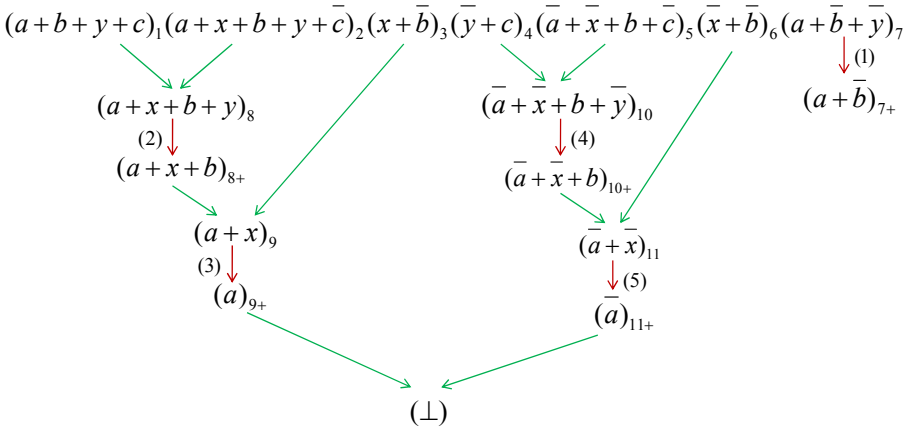

Fig. 2 DAG of resolution proof Π

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

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 every assignment to variables a, b, and c, formula Φ_{mtx} with variables x and y substituted with $F_H[x]$ and $F_H[y]$, respectively, is false.

The correctness of algorithm *Countermodel_construct* 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, 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 $\Phi_{\text{mtx}}|_\alpha = 0$. 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 $\Phi_{\text{mtx}}|_\alpha = 0$.

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

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 resolution 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 reducing 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 Herbrand function value of $F_H[x]$ 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 $F_H[x]$ 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 $\Phi_{\text{mtx}}|_\alpha = 0$.

Because every assignment $\alpha_\exists$ together with the corresponding induced assignment $\alpha_\forall$ makes $\Phi_{\text{mtx}}|_\alpha = 0$, the Herbrand 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 $F_H[x]$ be the Herbrand function computed by algorithm* Countermodel_construct *for the universal variable x in Φ. Then $F_H[x]$ 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 Herbrand functions.

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 3, it may be possible to exploit the redundancy for countermodel simplification.

4 Model Construction from Cube-Resolution Proofs

The discussion about countermodel construction can be easily extended under the duality principle to model construction of a true QBF from its cube-resolution proof of satisfiability as shown in this section.

Cube resolution can be defined as the dual of clause resolution. In contrast to clause resolution, where a resolvent is implied by the conjunction of its parent resolving clauses, a resolvent in cube resolution implies the disjunction of its parent resolving cubes. So if the empty cube is deduced through a sequence of cube resolutions, then the original formula is implied to be true. In fact, complementing a cube-resolution proof of a true QBF by negating each cube in the proof to a corresponding clause yields a clause-resolution proof of its negated false QBF, and vice versa.

Following a similar strategy to algorithm *Countermodel_construct*, algorithm *Model_construct* in Figure 4 computes Skolem functions from the resolution proof Π of a true QBF Φ by making the empty cube α-imply Φ_{mtx} under every α. Notice that it processes G_Π in a topological order, meaning that a cube in the resolution proof is processed only after all of its parent resolving cubes are processed. The correctness of the algorithm can be established in a way similar to Theorem 3 and its proof is omitted.

The following example illustrates how algorithm *Model_construct* works.

Example 4 Consider the true QBF $\Psi = \neg\Phi$ for Φ being the QBF of Example 3 and its resolution proof Π' shown below with $\exists$-reduction steps omitted.

$$\Psi_{\text{pfx}} = \forall a \exists x \forall b \exists y \forall c$$

$$\Psi_{\text{mtx}} = (\neg a \neg b \neg y \neg c) \vee (\neg a \neg x \neg b \neg y c) \vee (\neg x b) \vee (y \neg c) \vee (a x \neg b c) \vee (x b) \vee (\neg a b y)$$

$$\Pi' = \left\{ \begin{array}{l} 1.\ cube_8 = resolve(cube_1, cube_2) \\ 2.\ cube_9 = resolve(cube_3, cube_8) \\ 3.\ cube_{10} = resolve(cube_4, cube_5) \\ 4.\ cube_{11} = resolve(cube_{10}, cube_6) \\ 5.\ cube_{\text{empty}} = resolve(cube_{11}, cube_9) \end{array} \right\}$$

```
Model_construct
  input: a true QBF Φ and its cube-resolution DAG G_Π(V_Π, E_Π)
  output: a model in RFAO formulae
  begin
  01  foreach existential variable x of Φ
  02    RFAO_node_array[x] := ∅;
  03  foreach vertex v of G_Π in topological order
  04    if v.cube resulted from ∃-reduction on u.cube, i.e., (u, v) ∈ E_Π
  05      v.clause := ¬(v.cube);
  06      foreach existential variable x reduced from u.cube to get v.cube
  07        if x appears as positive literal in u.cube
  08          push v.cube to RFAO_node_array[x];
  09        else if x appears as negative literal in u.cube
  10          push v.clause to RFAO_node_array[x];
  11    if v.cube is the empty cube
  12      foreach existential variable x of Φ
  13        simplify RFAO_node_array[x];
  14      return RFAO_node_array's;
  end
```

Fig. 4 Algorithm: Model Construction

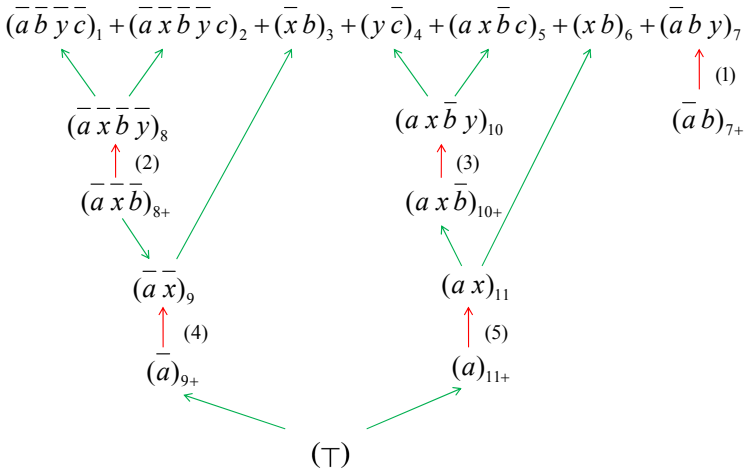

Fig. 5 DAG of resolution proof Π'

The DAG of Π' is shown in Figure 5, where the arrow direction signifies the implication direction. Note that Ψ above is not expressed in PCNF for ease of illustration. Realistically, Ψ should be in PCNF, and in this case, the cubes of Ψ_{mtx} above can be considered as the cubes learnt from solving Ψ.

The corresponding RFAO node-array content after each ∃-reduction step in the DAG of Figure 5 is the same as that of Figure 3. Consequently the obtained Skolem functions $F_S[x]$ and $F_S[y]$ for Ψ equal the previous Herbrand functions $F_H[x]$ and $F_H[y]$ for Φ, respectively. In fact, the equivalences are not a mere coincidence because the Skolem-function model for a true QBF Φ is

also the Herbrand-function countermodel for its negated false QBF $\neg\Phi$, and vice versa.

Note that algorithm *Countermodel_construct* (*Model_construct*) can be easily modified to compute the Herbrand (Skolem) functions for a subset of the universal (existential) variables of interest for a given QBF. Let k be the maximal quantification level among the universal (existential) variables whose Herbrand (Skolem) functions are of interest. Algorithm *Countermodel_construct* (*Model_construct*) only needs to maintain RFAO node arrays for universal (existential) variables with quantification level no greater than k. For Skolemization-based solvers, this partial derivation is not possible because Skolem functions are constructed on-the-fly during QBF solving, whereas our construction is performed after the entire proof is done.

5 (Counter)model Construction from Long-Distance Resolution Proofs

The above discussions focus on distance-1 resolution, where two resolving clauses have only the pivot variable appearing as literals in both positive and negative phases. There are solvers, such as QuBE-cert, occasionally producing proofs involving long-distance resolutions due to the omission of intermediate distance-1 resolutions. Although such proofs can always be transformed into regular proofs without long-distance resolutions by internal rearrangement, in practice it may slow down the solver.

This section characterizes a set of long-distance resolution rules as a sound and complete proof system. Moreover, we propose rewriting rules that eliminate long-distance resolution steps from a given resolution proof. We focus on clause resolution while similar results can be obtained for cube resolution.

In the sequel we shall denote $x \vee \neg x$, the disjunction of both literals of a variable x, as a special literal x^*. We formally define a resolution to be of long-distance if it belongs to one of the three cases:

$$\frac{C_1 \vee p \vee l \qquad C_2 \vee \neg p \vee \neg l}{\text{MIN}^*(C_1 \vee C_2 \vee l^*)}$$

$$\frac{C_1 \vee p \vee l \qquad C_2 \vee \neg p \vee l^*}{\text{MIN}^*(C_1 \vee C_2 \vee l^*)}$$

$$\frac{C_1 \vee p \vee l^* \qquad C_2 \vee \neg p \vee l^*}{\text{MIN}^*(C_1 \vee C_2 \vee l^*)}$$

where p and l are literals with $\ell(p) < \ell(l)$ and with $var(p)$ and $var(l)$ being existential and universal variables, respectively, and MIN^* extends the usual $\forall$-reduction MIN to reduce also l^* literal in the same way as the l and $\neg l$ literals. We call $var(l)$ the *merged* variable in the above resolutions. Notice that the above quantification level restriction applies to other merged variables in C_1 and C_2 as well. Otherwise long-distance resolution would be unsound. Also note that the resolution

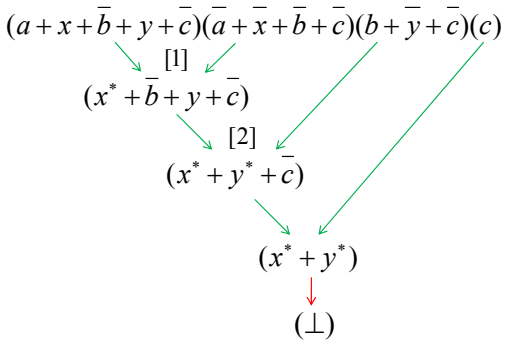

Fig. 6 DAG of resolution proof Π^*

$$\frac{C_1 \vee p \qquad C_2 \vee \neg p \vee l^*}{\mathrm{MIN}^*(C_1 \vee C_2 \vee l^*)}$$

is considered as the usual distance-1 resolution (since $var(l)$ is not considered as a merged variable for this resolution step, there is no quantification level restriction imposed on l).

In fact long-distance resolution can be exploited to condense proofs as the following example shows.

Example 5 Consider the following QBF, whose unsatisfiability is shown by the resolution proof Π^* detailed in Figure 6.

$$\Phi_{\mathrm{pfx}} = \exists a \forall x \exists b \forall y \exists c$$
$$\Phi_{\mathrm{mtx}} = (a \vee x \vee \neg b \vee y \vee \neg c)(\neg a \vee \neg x \vee \neg b \vee \neg c)(b \vee \neg y \vee \neg c)(c)$$
$$\Pi^* = \left\{ \begin{array}{l} 1.\ clause_5 = resolve(clause_1, clause_2) \\ 2.\ clause_6 = resolve(clause_3, clause_5) \\ 3.\ clause_{\mathrm{empty}} = resolve(clause_4, clause_6) \end{array} \right\}$$

As shown, the long-distance resolutions make the proof succinct.

The above resolution rules in addition to distance-1 resolutions form a sound and complete proof system for QBF evaluation as shown below.

Proposition 4 *Let C_3 be the resolvent under (possible long-distance) resolution of clauses C_1 and C_2. Let C_1' and C_2' be the sub-clauses of C_1 and C_2, respectively, (with $l \subseteq l^*$ and $\neg l \subseteq l^*$). Then the resolvent C_3' of C_1' and C_2' under the same pivot variable must be a sub-clause of C_3.*

By the previous proposition, the following claim can be concluded.

Proposition 5 *Let Π be a (possibly long-distance) resolution proof of a false QBF $\Phi = Q_1 x_1 \cdots Q_n x_n.\phi$. For an existential variable x_i, the induced proof, denoted $\Pi_{\neg x_i}$ (respectively Π_{x_i}), from Π with variable x_i being assigned constant 0 (respectively constant 1) forms a legitimate proof of QBF $Q_1 x_1 \cdots Q_{i-1} x_{i-1} Q_{i+1} x_{i+1} \cdots Q_n x_n.\phi|_{\neg x_i}$ (respectively $Q_1 x_1 \cdots Q_{i-1} x_{i-1} Q_{i+1} x_{i+1} \cdots Q_n x_n.\phi|_{x_i}$).*

Theorem 4 *The long-distance (in addition to distance-1) resolution rules form a sound and complete proof system for QBF evaluation. That is, a QBF Φ is false if and only if there exists a resolution sequence leading to the empty clause.*

Proof For completeness, it follows directly from the fact that distance-1 resolution, a special subset of the rules, is complete for QBF evaluation [12].

For soundness, we prove by induction on the number k of quantifiers in the prefix of Φ. Let Π be a resolution proof of the falsity of Φ. For $k = 1$, $\Phi = \exists x.\phi$ or $\forall x.\phi$. Since there is no long-distance resolution involved in Π, Π must be sound (following from the fact that distance-1 resolution is sound).

Assume the soundness holds for every long-distance resolution proof of Φ with its quantifier number $k \leq n$. For $k = n+1$ and $\Phi = \forall x_{n+1} Q_n x_n \cdots Q_1 x_1.\phi$, variable x_{n+1} can be $\forall$-reduced at most once in Π (namely on the last step of Π). If x_{n+1} in Π is not $\forall$-reduced at all, it means all the clauses with the occurrence of variable x_{n+1} are not involved in the resolution proof, and thus can be removed from the matrix. By the induction hypothesis, the soundness holds. If x_{n+1} in Π is $\forall$-reduced, it must be reduced in the form of literal x_{n+1} or literal $\neg x_{n+1}$ (cannot be ${x_{n+1}}^*$ since long-distance resolution allows the quantification level of a pivot variable less than that of a merged variable). Without loss of generality, assume literal x_{n+1} is $\forall$-reduced. Since no steps except for the last one in Π involves reduction of variable x_{n+1}, the ancestor clauses of the empty clause cannot include literal $\neg x_{n+1}$. So if x_{n+1} is assigned as constant 0 (and thus removed from the prefix), the induced proof will still be a legitimate proof. By the induction hypothesis, the soundness holds.

On the other hand, for $k = n + 1$ and $\Phi = \exists x_{n+1} Q_n x_{n+1} \cdots Q_1 x_1.\phi$, according to Proposition 5 the induced proofs $\Pi_{x_{n+1}}$ and $\Pi_{\neg x_{n+1}}$ are both legitimate proofs leading to the empty clause. So the unsatisfiability proof of $Q_n x_n \cdots Q_1 x_1.\phi|_{\neg x_{n+1}}$ and that of $Q_n x_n \cdots Q_1 x_1.\phi|_{x_{n+1}}$ establish the falsity of Φ. Therefore by induction hypothesis the soundness holds. ■

To derive the Herbrand functions from a long-distance resolution proof, the following two rewriting rules can be first applied to eliminate long-distance resolution steps from the proof.

$$\dfrac{\dfrac{C_1 \vee p \vee q \vee l_1 \qquad C_2 \vee \neg p \vee l_2}{C_1 \vee C_2 \vee q \vee l^*} \qquad \neg q \vee C_3}{C_1 \vee C_2 \vee C_3 \vee l^*} \quad \longrightarrow$$

$$\dfrac{\dfrac{C_1 \vee p \vee q \vee l_1 \qquad \neg q \vee C_3}{C_1 \vee C_3 \vee p \vee l_1} \qquad C_2 \vee \neg p \vee l_2}{C_1 \vee C_2 \vee C_3 \vee l^*}$$

$$\dfrac{\dfrac{C_1 \vee p \vee q \vee l_1 \qquad C_2 \vee \neg p \vee q \vee l_2}{C_1 \vee C_2 \vee q \vee l^*} \qquad \neg q \vee C_3}{C_1 \vee C_2 \vee C_3 \vee l^*} \quad \longrightarrow$$

$$\dfrac{\dfrac{C_1 \vee p \vee q \vee l_1 \qquad \neg q \vee C_3}{C_1 \vee C_3 \vee p \vee l_1} \qquad \dfrac{C_2 \vee \neg p \vee q \vee l_2 \qquad \neg q \vee C_3}{C_2 \vee C_3 \vee \neg p \vee l_2}}{C_1 \vee C_2 \vee C_3 \vee l^*}$$

where p, q, l_1, and l_2 are literals with $l_1, l_2 \in \{l, \neg l, l^*\}$, and $l_1 \neq l_2$ or $l_1 = l_2 = l^*$. As stated in the following theorem, the above rewriting rules when applied in a proper order are complete in removing long-distance resolution steps from a resolution proof.

Theorem 5 *Given a long-distance resolution proof Π of a false QBF Φ, the above two rewriting rules when applied on the long-distance resolution steps in a reverse topological order eventually yield a new proof without long-distance resolution.*

Proof Consider the last (i.e., in the reverse topological order, the first encountered) long-distance resolution step S of Π. Let C'_1 and C'_2 be the resolving clauses at S. Then there must exist another clause C'_3 such that these three clauses match the three top-level clauses of either the first or the second rewriting rule above. Since no descendants of S are long-distance resolution steps, clause C'_3 cannot include variable $var(l)$. After applying the corresponding rewriting on these clauses, the new last long-distance resolution step S will move one step closer to the empty clause without introducing any new long-distance resolution step to the proof. Since S is again the closest to the empty clause, the rewriting rules will be applied to it again and again until all the existential variables in its C'_1 and C'_2 clauses disappear and $var(l)$ is eventually $\forall$-reduced. Consequently every last long-distance resolution step will be eventually removed. Iterating this process all long-distance resolution can be eliminated and leaving a new proof with only distance-1 resolutions. ■

Example 6 Consider the following QBF Φ with a long-distance resolution Π.

$$\begin{aligned}
\Phi_{\text{pfx}} &= \exists a \exists b \forall x \exists c \\
\Phi_{\text{mtx}} &= (a \vee x \vee c)(\neg a \vee b \vee \neg x \vee c)(\neg b \vee x \vee c)(\neg c) \\
\Pi &= \left\{ \begin{array}{l} 1.\ clause_5 = resolve(clause_1, clause_2) \\ 2.\ clause_6 = resolve(clause_3, clause_5) \\ 3.\ clause_{\text{empty}} = resolve(clause_4, clause_5) \end{array} \right\}
\end{aligned}$$

Figure 7 shows the rewriting process converting Π to a new proof without long-distance resolution steps.

6 Applications to Boolean Relation Determinization

We relate Skolem functions to the problem of *Boolean relation determinization*, which is useful in logic and property synthesis [10,11]. 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 $b \in \{0,1\}^{|Y|}$

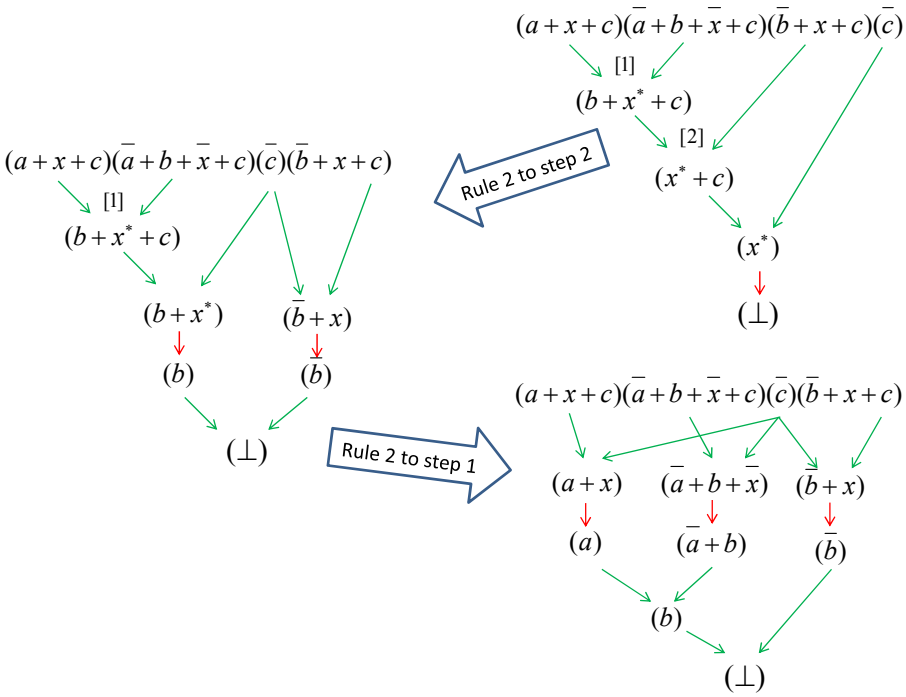

Fig. 7 Rewriting long-distance resolution

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.

Table 1 Summary for QBFEVAL Benchmarks.

		all	sKizzo		squolem+ResQu			QuBE-cert+ResQu		
		#sv	#sv	time (sv)	#sv	time (sv)	time (md)	#sv/#pg	time (sv)	time (md)
true	'05	84	69	1707.3	50	1490.8	—	19/19	414.7	54.7
	'06	48	29	295.2	25	199.8	—	44/44	859.6	152.2
	total	132	98	2002.5	75	1690.6	—	63/63	1274.3	207.0
false	'05	77	0	0	42	1467.5	12.6	46/25	2369.9	13.0
	'06	29	0	0	9	86.0	0.8	28/22	916.6	2.3
	total	106	0	0	51	1553.4	13.4	74/47	3286.5	15.3

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

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.

7 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 [15] and relation determinization ones modified from [3].

We compared different Skolem-function derivation scenarios using QBF solvers sKizzo [1], squolem [10], and QuBE-cert, which are equipped with certification capabilities.[3] 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 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

[3] We did not experiment with Ebddres [10] and yQuaffle [21] as the former tends to generate larger certificates for false QBF compared to squolem, and the latter has characteristics similar to QuBE-cert.

ResQu were verified using MiniSat [8] 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. 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.[4] 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.

Table 1 also reveals that for the QBFEVAL'05 true instances sKizzo and squolem outperformed QuBE-cert in solving more instances, while the situation reversed for the QBFEVAL'06 true cases. Investigating the reasons, we noted that, for the solved QBFEVAL'05 true instances, the average numbers of quantification levels, existential variables, universal variables, and clauses are 22, 1393, 80, and 8028, respectively; for QBFEVAL'06, they are 3, 672, 78, and 1690, respectively. The statistics might suggest that sKizzo and squolem could work better for true QBF with fewer existential variables (since there are fewer Skolem functions to derive), whereas search-based solvers could work better for cases with fewer quantification levels (since resolution depths are shallower and $\exists$-reduction can be more effective).

We also compared the sizes of (counter)models in terms of and-inverter graph (AIG) nodes. For models, we noticed that those constructed by ResQu from the proofs of QuBE-cert can be up to a few orders of magnitude larger than those from sKizzo and squolem. Similarly for countermodels, those constructed by ResQu from the proofs of QuBE-cert tend to be much larger than those from the proofs of squolem. These AIGs can be very redundant

[4] 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 [10] suggested that it might be unlikely to cover much more cases than sKizzo.

Table 2 Results for Relation Determinization Benchmarks.

	(#i, #o, #e, #C)	sKizzo		sQuolem+ResQu		QuBE-cert+ResQu	
		time (sv)	size	time (sv/md/vf)	size	time (sv/md/vf)	size
22 original (true) instances	(7, 3, 55, 322)	0.1	377	(0.1 , —, —)	134	(0.0, 0.0, 0.0)	28
	(20, 10, 1k, 6k)	0.9	1.3k	(0.8, —, —)	1.4k	(0.1, 0.0, 0.0)	118
	(21, 9, 1k, 8k)	NA		(NA, —, —)		(5.3, 1.7, 1.2)	149k
	(24, 12, 2k, 11k)	NA		(1.2, —, —)	179	(1.0, 0.1, 0.0)	1.9k
	(28, 14, 2k, 11k)	NA		(NA, —, —)		(0.4, 0.1, 0.0)	61
	(32, 16, 3k, 20k)	NA		(NA, —, —)		(1.2, 0.2, 0.0)	1.2k
	(36, 18, 6k, 35k)	NA		(NA, —, —)		(0.2, 0.2, 0.0)	91
	(42, 20, 7k, 42k)	NA		(NA, —, —)		(0.3, 0.1, 0.0)	3
	(39, 19, 10k, 59k)	NA		(NA, —, —)		(3.1, 0.5, 0.0)	307
	(46, 22, 11k, 63k)	NA		(NA, —, —)		(1.9, 0.3, 0.0)	58
	(49, 19, 11k, 68k)	NA		(NA, —, —)		(NA, NA, NA)	
	(32, 18, 13k, 81k)	NA		(NA, —, —)		(NA, NA, NA)	
	(50, 24, 15k, 89k)	NA		(NA, —, —)		(3.1, 0.8, 0.1)	3.5k
	(53, 25, 16k, 96k)	NA		(NA, —, —)		(3.4, 0.4, 0.0)	283
	(56, 26, 20k, 118k)	NA		(NA, —, —)		(8.1, 1.1, 0.1)	905
	(59, 27, 26k, 157k)	NA		(NA, —, —)		(3.9, 0.6, 0.0)	187
	(65, 29, 29k, 174k)	NA		(NA, —, —)		(7.2, 0.9, 0.1)	232
	(62, 28, 30k, 182k)	NA		(NA, —, —)		(9.3, 1.3, 0.1)	731
	(72, 32, 36k, 215k)	NA		(NA, —, —)		(NA, NA, NA)	
	(68, 30, 51k, 303k)	NA		(NA, —, —)		(3.0, 0.6, 0.1)	11
	(95, 35, 58k, 346k)	NA		(NA, —, —)		(NA, NA, NA)	
	(41, 23, 90k, 538k)	NA		(NA, —, —)		(NA, NA, NA)	
22 complemented (false) instances	(7, 3, 55, 322)	NA		(0.0, 0.0, 0.0)	6	(0.1, 0.0, 0.0)	17
	(20, 10, 1k, 6k)	NA		(1.1, 0.0, 0.0)	53	(0.1, 0.1, 0.0)	61
	(21, 9, 1k, 8k)	NA		(0.2, 0.0, 0.0)	4	(1.2, 0.1, 0.0)	5
	(24, 12, 2k, 11k)	NA		(0.3, 0.0, 0.0)	0	(0.3, 0.3, 0.0)	30
	(28, 14, 2k, 11k)	NA		(0.3, 0.0, 0.0)	3	(1.0, 0.5, 0.0)	150
	(32, 16, 3k, 20k)	NA		(2.0, 0.0, 0.0)	3	(1.0, 0.2, 0.0)	3
	(36, 18, 6k, 35k)	NA		(3.1, 0.1, 0.1)	3	(4.2, 1.3, 0.0)	11
	(42, 20, 7k, 42k)	NA		(3.2, 0.1, 0.1)	3	(9.2, 3.6, 0.0)	1.2k
	(39, 19, 10k, 59k)	NA		(9.4, 0.1, 0.1)	5	(10.0, 1.6, 0.0)	201
	(46, 22, 11k, 63k)	NA		(9.9, 0.2, 0.1)	3	(3.6, 0.6, 0.0)	31
	(49, 19, 11k, 68k)	NA		(8.3, 0.2, 0.1)	3	(14.1, 0.9, 0.0)	1
	(32, 18, 13k, 81k)	NA		(10.4, 0.2, 0.1)	3	(10.4, 1.3, 0.0)	0
	(50, 24, 15k, 89k)	NA		(15.8, 0.3, 0.1)	4	(510, 89.9, 0.0)	20k
	(53, 25, 16k, 96k)	NA		(23.7, 0.3, 0.1)	5	(7.2, 0.9, 0.0)	4
	(56, 26, 20k, 118k)	NA		(30.2, 0.4, 0.1)	3	(25.3, 2.3, 0.0)	93
	(59, 27, 26k, 157k)	NA		(74.2, 0.4, 0.1)	3	(203, 122, 0.0)	12k
	(65, 29, 29k, 174k)	NA		(46.9, 0.4, 0.2)	0	(24.5, 1.5, 0.0)	5
	(62, 28, 30k, 182k)	NA		(84.5, 0.5, 0.3)	4	(94.9, 3.3, 0.0)	570
	(72, 32, 36k, 215k)	NA		(130, 0.4, 0.2)	3	(80.1, 2.6, 0.0)	662
	(68, 30, 51k, 303k)	NA		(363, 0.7, 7.3)	3	(26.1, 2.3, 0.0)	0
	(95, 35, 58k, 346k)	NA		(359, 1.0, 8.2)	2	(86.1, 2.5, 1.2)	6
	(41, 23, 90k, 538k)	NA		(NA, NA, NA)		(142, 5.1, 0.0)	170

#i: number of input variables in a relation; #o: number of output variables in a 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

and can be substantially simplified by ABC [5] synthesis commands, e.g., `balance`, `dc2`, `collapse`, etc. If heavy synthesis commands such as `collapse` succeeded, all the (counter)models derived in different ways for the same instance can be of comparable sizes.

Table 2 shows the results of our second experiment on 22 relation determinization benchmarks. All the original 22 instances are true (satisfiable). We

Table 3 Summary for Relation Determinization Benchmarks.

	all	sKizzo		squolem+ResQu			QuBE-cert+ResQu		
	#sv	#sv	time (sv)	#sv	time (sv)	time (md)	#sv/#pg	time (sv)	time (md)
true	17	2	1.0	3	2.1	—	17/17	51.3	8.8
false	22	0	0	21	1175.8	5.2	22/0	1254.3	242.9

(Legend same as in Table 1)

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 6. 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.

It was observed that squolem is easier to produce a clause-resolution proof than Skolem functions for a given relation determinization instance. This phenomenon might be explained by the common large number of innermost existential variables, each of which requires building a Skolem function. It was similarly observed that QuBE-cert is relatively easier to produce a clause-resolution proof than a cube-resolution one.

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. ResQu internally applied the aforementioned rewriting rules to convert them into new proofs without long-distance resolution and constructed their countermodels. All the instances were successfully solved and verified. 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.

8 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.

Acknowledgements The authors are grateful to Roderick Bloem and Georg Hofferek for providing the relation determinization benchmarks.

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* (DATE), 2007.
4. V. Balabanov and J.-H. R. Jiang. Resolution Proofs and Skolem Functions in QBF Evaluation and Applications. In *Proc. Int'l Conf. on Computer Aided Verification* (CAV), pp.149-164, 2011.
5. Berkeley Logic Synthesis and Verification Group. *ABC: A System for Sequential Synthesis and Verification*. `http://www.eecs.berkeley.edu/~alanmi/abc/`
6. 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.
7. 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.
8. 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.
9. E. Giunchiglia, M. Narizzano, and A. Tacchella. Clause-Term Resolution and Learning in Quantified Boolean Logic Satisfiability. *Artificial Intelligence Research*, 26:371-416, 2006.
10. 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.
11. 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.
12. H. Kleine-Büning, M. Karpinski and A. Flögel. Resolution for Quantified Boolean Formulas. *Information and Computation*, 117(1):12-18, 1995.
13. 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.
14. C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
15. QBF Solver Evaluation Protal. `http://www.qbflib.org/qbfeval/`
16. J. Rintanen. Constructing Conditional Plans by a Theorem-Prover. *Journal of Artificial Intelligence Research*, 10:323-352, 1999.
17. 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.]
18. 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.
19. 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.
20. G. Tseitin. On the Complexity of Derivation in Propositional Calculus. *Studies in Constructive Mathematics and Mathematical Logic*, pp. 466-483, 1970.
21. 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* (ASP-DAC), 2005.
22. 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.