Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Efficient Extraction of QBF (Counter)models from Long-Distance Resolution Proofs

Valeriy Balabanov[1], Jie-Hong R. Jiang[1], Mikoláš Janota[2], and Magdalena Widl[3]
[1]GIEE, National Taiwan University, Taipei, Taiwan
[2]INESC-ID, Lisbon, Portugal
[3]Vienna University of Technology, Austria

Abstract

Many computer science problems can be naturally and compactly expressed using quantified Boolean formulas (QBFs). Evaluating the truth or falsity of a QBF is an important task, and constructing the corresponding model or countermodel can be as important and sometimes even more useful in practice. Modern search and learning based QBF solvers rely fundamentally on resolution and can be instrumented to produce resolution proofs, from which in turn Skolem-function models and Herbrand-function countermodels can be extracted. These (counter)models are the key enabler of various applications. Not until recently the superiority of long-distance resolution (LQ-resolution) to short-distance resolution (Q-resolution) was demonstrated. While a polynomial algorithm exists for (counter)model extraction from Q-resolution proofs, it remains open whether it exists for LQ-resolution proofs. This paper settles this open problem affirmatively by constructing a linear-time extraction procedure. Experimental results show the distinct benefits of the proposed method in extracting high quality certificates from some LQ-resolution proofs that are not obtainable from Q-resolution proofs.

Introduction

Quantified Boolean formulas (QBFs) extend propositional formulas by adding quantifiers to the language of propositional logic. This extension lifts the computational complexity from the NP-complete propositional satisfiability (SAT) problem to the PSPACE-complete quantified Boolean satisfiability (QSAT) problem, and increases the descriptive power to allow compact encodings for a broad range of problems not economically expressible in propositional logic (Schaefer and Umans 2002). QBF applications include, for example, planning (Rintanen 2007), ontology reasoning (Kontchakov et al. 2009), formal verification (Dershowitz, Hanna, and Katz 2005; Benedetti and Mangassarian 2008), design debugging (Staber and Bloem 2007), logic synthesis (Jiang, Lin, and Hung 2009), etc. This broad range of QBF applications and the tremendous success of modern SAT solvers motivate the increasing effort being dedicated to QBF solving.

Many successful techniques in QBF solving were inspired by well-known techniques in SAT solving, notably, *conflict-driven clause learning* (CDCL) (Marques-Silva and Sakallah 1996), among others. Learning in CDCL-based SAT solvers hinges on *resolution*, a fundamental technique in automated reasoning. CDCL was generalized for QBF solving (Zhang and Malik 2002) based on Q-resolution (Kleine Büning, Karpinski, and Flögel 1995), a sound and complete inference rule for QBFs. QBF solving was further improved by allowing solution-driven cube learning, where Q-resolution on clauses is dualized to Q-resolution on cubes (Giunchiglia, Narizzano, and Tacchella 2006).[1] In addition to Q-resolution, long-distance Q-resolution (LQ-resolution) was introduced (Zhang and Malik 2002) and formalized in terms of inference rules (Balabanov and Jiang 2012). Although it strengthens the deductive power of Q-resolution, its superiority over Q-resolution was not clear until Egly *et al.* 2013 demonstrated (empirically with strong theoretical evidence) that LQ-resolution can produce exponentially shorter proofs than Q-resolution (Egly, Lonsing, and Widl 2013).

Similar to resolution in SAT, (Q and LQ) resolution in QBF solving is essential not only in learning but also in certification. Not until the recent unifying work (Balabanov and Jiang 2012; Goultiaeva, Van Gelder, and Bacchus 2011), QBF certification was incomplete with two unconnected forms of certificates: the syntactic form of Q-resolution proofs and the semantic form of Skolem-function models (but no Herbrand-function countermodels). QBF certification is important not only in ensuring the correctness of QBF solving, but also in real-life applications where models and countermodels have to be constructed. E.g., a (counter)model could represent a concrete plan leading to a goal state in a planning problem, a functional implementation of a Boolean relation in logic synthesis, a winning strategy in a two player game, etc. In prior work (Balabanov and Jiang 2012), a linear-time algorithm was devised to convert Q-resolution proofs to (counter)models. Thereby, (counter)models can be obtained from state-of-the-art learning based QBF solvers without relying on spe-

[1]In this paper, we do not specifically distinguish between clause and cube Q-resolutions, and we refer to the former and the latter when the falsity and truth, respectively, of a QBF are considered.

cial solvers using Skolemization, such as sKizzo (Benedetti 2005). Nevertheless, the method is limited to Q-resolution and cannot be applied to LQ-resolution proofs. As LQ-resolution has its distinct power producing short proofs, extracting (counter)models from LQ-resolution proofs is indispensable. Note that under a game-theoretic interpretation of QBF, although winning moves can be played interactively in time polynomial with respect to a Q-resolution proof (Goultiaeva, Van Gelder, and Bacchus 2011) and even an LQ-resolution proof (Egly, Lonsing, and Widl 2013), constructing a (counter)model circuit from instances of winning moves can be exponential. Despite some recent efforts (Egly, Lonsing, and Widl 2013; Balabanov, Widl, and Jiang 2014), it remains open whether there exists a polynomial algorithm extracting (counter)models from LQ-resolution proofs.

In this work, we present such an algorithm of time complexity linear in the size of a given LQ-resolution proof. Experimental results demonstrate unique benefits of our algorithm in extracting simple (counter)model circuits. Several QBF instances whose (counter)models were not obtainable before can now be derived. Our algorithm further advances the practicality of QBF.

Preliminaries

A Boolean formula in *conjunctive normal form* (CNF) consists of a conjunction of clauses; a *clause* is a disjunction of literals; a *cube* is a conjunction of literals; a *literal* is either a Boolean variable (referred to as a literal of *positive phase* or a *positive literal*) or its negation (referred to as a literal of negative phase or a *negative literal*). A *Boolean variable* is interpreted over the binary domain $\{0, 1\}$. We denote the variable corresponding to a literal l by $var(l)$ and the set of variables appearing in a clause C by $vars(C)$. We alternatively specify a clause or cube by a set of literals. As a notational convention, we sometimes omit the Boolean connective conjunction ($\wedge$), denote disjunction ($\vee$) by the symbol "$+$," and represent negation ($\neg$) by an overline.

A Boolean formula ϕ over a set X of variables, subject to some truth assignment $\alpha : X' \to \{0, 1\}$ on the set $X' \subseteq X$ of variables is denoted by $\phi\mid_\alpha$. For an assignment α to variables X, we alternatively represent the mappings $\alpha(x) \mapsto 0$ and $\alpha(x) \mapsto 1$ for $x \in X$ as literals $\overline{x}$ and x, respectively. Therefore we consider an assignment α as a conjunction of literals or as a set of literals.

Given a variable x, a clause C containing both a positive literal x and a negative literal $\overline{x}$ is tautological. In the sequel, we replace the appearance of both literals x and $\overline{x}$ in C by a *merged literal*, denoted as x^*. Using the previous notation, we define $var(x^*) = x$. It should be noted that the negation of a merged literal is not defined. Hence, given a literal l, the presence of $\bar{l}$ in our discussion automatically asserts that l is not a merged literal.

A *quantified Boolean formula* (QBF) Φ over variables $X = \{x_1, \ldots, x_k\}$ in *prenex conjunctive normal form* (PCNF) is of the form $Q_1x_1 \cdots Q_kx_k.\phi$, where $Q_1x_1 \cdots Q_kx_k$, 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}. 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 over variables X, the *quantification level* of variable $x \in X$, denoted $lvl(x)$, is defined to be the number of quantifier alternations between the quantifiers $\exists$ and $\forall$ from left (outer) to right (inner) plus 1. The same level definition extends to a literal l, i.e., $lvl(l) = lvl(var(l))$.

A QBF $\Phi = \Phi_{\text{pfx}}.\phi(e_1, \ldots, e_m, u_1, \ldots, u_n)$ over existential variables $X_\exists = \{e_1, \ldots, e_m\}$ and universal variables $X_\forall = \{u_1, \ldots, u_n\}$ evaluates to *true* if and only if there exists a set of *Skolem functions* (Skolem 1928) $F[e_i] : \{0, 1\}^{|X_{e_i}|} \to \{0, 1\}$ for each $e_i \in X_\exists$ with $X_{e_i} = \{x \mid x \in X_\forall \text{ and } lvl(x) < lvl(e_i)\}$, $i = 1, \ldots, m$, such that substituting the existential variables in ϕ by their corresponding Skolem functions makes $\phi(F[e_1], \ldots, F[e_m], u_1, \ldots, u_n)$ a tautology. That is, the Skolem functions serve as a model for Φ. By duality, a QBF Φ evaluates to *false* if and only if there exists a set of *Herbrand functions* $F[u_i] : \{0, 1\}^{|X_{u_i}|} \to \{0, 1\}$ for each $u_i \in X_\forall$ with $X_{u_i} = \{x \mid x \in X_\exists \text{ and } lvl(x) < lvl(u_i)\}$, $i = 1, \ldots, n$, such that substituting universal variables in ϕ with their corresponding Herbrand functions makes $\phi(e_1, \ldots, e_m, F[u_1], \ldots, F[u_n])$ unsatisfiable. That is, the Herbrand functions serve as a countermodel for Φ.

In addition to the above semantic QBF evaluation, the truth or falsity of a QBF in PCNF can be evaluated via a syntactic way of applying the inference rules of Q-resolution (Kleine Büning, Karpinski, and Flögel 1995). The *resolution* of two given clauses C_1 and C_2, denoted $resolve(C_1, C_2)$, produces a clause $C_1 \setminus \{p\} \cup C_2 \setminus \{\overline{p}\}$, called the *resolvent*, for literals $p \in C_1$ and $\overline{p} \in C_2$. The literals p and $\overline{p}$ are called the *pivot literals*, and $var(p)$ is called the *pivot variable* of the resolution.

A *short-distance* (or ordinary) resolution refers to the resolution satisfying that, for any (including positive, negative, and merged) literals $l_1 \in C_1 \setminus \{p\}$ and $l_2 \in C_2 \setminus \{\overline{p}\}$, if $var(l_1) = var(l_2)$, then $l_1 = l_2$ and l_1 is not merged. Otherwise it is referred to as a *long-distance resolution*. A long-distance resolution is called *proper*, if the following *level restriction* holds. For any (including positive, negative, and merged) literals $l_1 \in C_1 \setminus \{p\}$ and $l_2 \in C_2 \setminus \{\overline{p}\}$, if $var(l_1) = var(l_2)$ and either $l_1 \neq l_2$ or l_1 is merged, then it holds that $var(l_1) \in X_\forall$ and $lvl(l_1) = lvl(l_2) > lvl(p)$. In the sequel we only consider the proper long-distance resolution, and for simplicity refer to it just by long-distance resolution.

Given a clause C, *universal reduction* on C, denoted $reduce(C)$, produces the reduced clause $C \setminus \{l \in C \mid var(l) \in X_\forall \text{ and } lvl(l) > lvl(l') \text{ for each } l' \in C \text{ with } var(l') \in X_\exists\}$, i.e., it removes from C all universal variables whose quantifier levels are greater than the largest level of any existential variable in C. Note that universal reduction applies to the merged literals from C in the same way as it applies to positive and negative literals.

Q-resolution (Kleine Büning, Karpinski, and Flögel 1995) consists of two rules: (short-distance) resolution over only existential variables and universal reduction. *LQ-resolution* (Balabanov and Jiang 2012) extends Q-resolution by allowing long-distance resolution.

Both Q-resolution and LQ-resolution form sound and complete proof systems for QBF (Balabanov and Jiang 2012; Kleine Büning, Karpinski, and Flögel 1995). That is, for a QBF in PCNF, it is false if and only if an empty clause can be derived using the Q-resolution (LQ-resolution) rules. The sequence of Q-resolution steps in a derivation of the empty clause (resp. cube) forms a proof of the falsity (resp. truth) of the QBF. State-of-the-art search based QBF solvers employ Q-resolution (LQ-resolution) as the underlying learning mechanism and therefore can produce Q-resolution (LQ-resolution) proofs for validation.

Extending prior work on (counter)model extraction from Q-resolution proofs (Balabanov and Jiang 2012), we consider the more general LQ-resolution proofs. Below we reproduce the definition of a *Right-First-And-Or (RFAO) formula* given in (Balabanov and Jiang 2012), as it is used throughout this work. An RFAO formula φ is recursively defined by

$$\varphi ::= clause \mid cube \mid clause \wedge \varphi \mid cube \vee \varphi,$$

where the symbol "::=" is read as "can be" and symbol "|" as "or". An RFAO formula can be specified as an (ordered) sequence of nodes, $node_1, node_2, \ldots, node_n$, where each node is either a clause or a cube. An RFAO formula has the following two important properties (Balabanov and Jiang 2012), to be used in the proof of Lemma 2.

1. If $node_i$ under some (partial) assignment of variables becomes a validated clause (i.e., a clause that valuates to 1) or a falsified cube (i.e., a cube that valuates to 0), then we can remove $node_i$ (unless it is the last node) from the formula.
2. If $node_i$ becomes a falsified clause (i.e., a clause that valuates to 0) or validated cube (i.e., a cube that valuates to 1), then the value of the formula is determined by $node_i$, and thus we can remove other nodes with indices greater than i.

(Counter)model Extraction from LQ-Resolution Proofs

Given an LQ-resolution proof of a false QBF, we present a procedure extracting a Herbrand-function countermodel in time linear with respect to the proof size. By duality, a Skolem-function model can be similarly extracted from a true QBF.

In the sequel, we consider an LQ-resolution proof Π of a false QBF Φ as a directed acyclic graph (DAG) $G_\Pi(V_\Pi, E_\Pi)$, where a vertex $v \in V_\Pi$ corresponds to a clause $v.clause$ in Π, and an edge $(u, v) \in E_\Pi \subseteq V_\Pi \times V_\Pi$ corresponds to the derivation of $v.clause$ from either an LQ-resolution step (i.e., $v.clause = resolve(u.clause, \cdot)$) or a universal reduction step (i.e., $v.clause = reduce(u.clause)$) over $u.clause$ in Π. For $(u, v) \in E_\Pi$, we call v a *child* of u, and u a *parent* of v. To generalize, for u that reaches v through a number of connected edges, we call v a *descendant* of u, and u an *ancestor* of v.

To study the implication relations among the clauses in an LQ-resolution proof, we use the following definition (Balabanov and Jiang 2012).

Definition 1 (α-implication). *Given two (quantifier-free) formulas ϕ_1 and ϕ_2 over variables X, let α be an assignment to X. If $(\phi_1 \rightarrow \phi_2)|_\alpha$, then we say that ϕ_2 is α-*implied *by ϕ_1.*

For a resolution proof Π of a false QBF Φ, when we say a clause C is *α-implied*, we mean C is α-implied by its parent clause for the case of universal reduction or by the conjunction of its two parent clauses for the case of resolution. Given a vertex $v \in V_\Pi$ such that $v.clause = resolve(u_1.clause, u_2.clause)$ in Π, the implication $u_1.clause \wedge u_2.clause \rightarrow v.clause$ always holds. Therefore, a clause resulting from resolution is α-implied under any α. We further say that a clause C is *α-inherited* if all of its ancestor clauses and C itself are α-implied. Clearly, if C is α-inherited, then $\Phi_{\mathrm{mtx}}|_\alpha = (\Phi_{\mathrm{mtx}} \wedge C)|_\alpha$.

As it was mentioned in Preliminaries, merged literals do not follow the same semantics as ordinary literals. The following simple example illustrates the problem for the notion of α-implication in the presence of tautological clauses.

Example 1. *Consider the QBF $\Phi = \exists a \forall x \exists b.(a + x + b)_1(\overline{a} + \overline{x} + b)_2(\overline{b})_3$ and the corresponding LQ-resolution proof Π with $\{C_4 = resolve(C_1, C_2); C_5 = resolve(C_3, C_4); C_{empty} = reduce(C_5)\}$. Note that $C_5 = \{x^*\}$, and if the semantics of a merged literal x^* is to be treated similarly to an ordinary literal, then $C_5|_\alpha = 1$ for any assignment α. Therefore C_{empty} cannot be α-implied. However, the empty clause is soundly deduced following the LQ-resolution proof system.*

Given a false QBF Φ over variables $X = X_\exists \cup X_\forall$ and its LQ-resolution proof Π, let $\alpha_\exists$ be an assignment to the existential variables $X_\exists$. For an arbitrary vertex $v \in V_\Pi$ and an arbitrary literal $l \in v.clause$, Table 1 defines some additional attributes associated with v or with l, including *parent literal*, *phase function*, *effective literal*, and *shadow clause*, which are to be used in our countermodel extraction algorithm.

A *phase function* intuitively represents the induced phase of a literal in a clause under a particular assignment to the existential variables. An *effective literal* represents the induced value of its corresponding literal. Observe that, by the definition of the phase function and the effective literal, $l.elit \leftrightarrow l$ holds whenever l is not a merged literal. In essence, effective literals represent the conditional origins of merged literals in connection to ordinary literals. A *shadow clause* corresponds to the disjunction of a set of effective literals. To illustrate, consider the merged literal $x^* \in C_5$ in Example 1. We have $x^*.elit = (\overline{a} \leftrightarrow x)$. For partial assignment $a = 0$, we have $x^*.elit = x$ and C_2 valuating to true. Hence, proof Π can be simplified to $\Pi|_{\{\overline{a}\}}$ with $\{C'_4 = resolve(C_1|_{\{\overline{a}\}}, C_3|_{\{\overline{a}\}}); C'_{empty} = reduce(C'_4)\}$. Observe that the merged literal $x^* \in C_5$ in proof Π now corresponds to the ordinary literal $x \in C'_4$,

Attribute	Definition
Parent literal	Literal $l' \in u.clause$ is called a *parent literal* of $l \in v.clause$, denoted $l.ancestor$, if $var(l') = var(l)$ and $(u, v) \in E_\Pi$. Note that l can only have 0, 1 or 2 parent literals.
Phase function	The *phase function*, denoted $l.phase$, of literal $l \in v.clause$ is defined as follows: • if l is positive, then $l.phase = 1$; • if l is negative, then $l.phase = 0$; • if l is merged and l has only one parent literal l', then $l.phase = l'.phase$; • if l is merged and l has two parent literals $l_1 \in u_1.clause$ and $l_2 \in u_2.clause$ with $(u_1, v) \in E_\Pi$ and $(u_2, v) \in E_\Pi$, then $l.phase = (l_1.phase \wedge \overline{p}) \vee (l_2.phase \wedge p)$, where pivot $p \in X_\exists$, $p \in u_1.clause$ and $\overline{p} \in u_2.clause$.
Effective literal	The *effective literal*, denoted $l.elit$, of $l \in v.clause$ is a literal that satisfies $l.elit \leftrightarrow (x \leftrightarrow l.phase)$, where $x = var(l)$.
Shadow clause	The *shadow clause*, denoted $v.shadcls$, of $v \in V_\Pi$ is the clause of effective literals of v: $$v.shadcls = \bigcup_{l \in v.clause} (l.elit).$$

Table 1: Attributes of each vertex $v \in V_\Pi$ of an LQ-resolution proof Π, represented as a DAG $G_\Pi(V_\Pi, E_\Pi)$, of a false QBF Φ.

which is equivalent to $x^*.elit$ under our partial assignment $a = 0$. On the other hand, under partial assignment $a = 1$, a similar correspondence can be observed. Therefore, effective literals intuitively represent how merged literals should be interpreted under a given (partial) assignment.

The procedure, *CountermodelExtractLQ*, to extract Herbrand functions from a given LQ-resolution proof is outlined in Figure 1. It is similar to the procedure *Countermodel_construct* in (Balabanov and Jiang 2012), but with two main differences: First, shadow clauses, rather than ordinary clauses, are used to construct RFAO formulas. Second, merged literals are processed as in Lines 14-16. Notice that Line 5 uses the definition of phase functions given in Table 1.

Example 2 illustrates the computation steps of algorithm *CountermodelExtractLQ*.

Example 2. *Consider the false QBF Φ with its prefix and matrix defined as follows.*

$$\Phi_{\text{pfx}} = \exists ab \forall x \exists cd \forall y \exists e$$

$$\Phi_{\text{mtx}} = (a, x, d)_1 (\overline{a}, b, \overline{x}, d)_2 (\overline{b}, x, c, y, e)_3 (\overline{c}, d, \overline{y}, e)_4 (d, \overline{e})_5 (\overline{d})_6$$

Its falsity can be established by the LQ-resolution proof Π visualized in the DAG of Figure 2 (a). The phase functions f_i and g_i corresponding to the literals of variables x and y present in the clause C_i can be derived as shown in Figure 2 (b) and (c), respectively. Note that, if $lit(x) \notin C_i$ (resp. $lit(y) \notin C_i$), then $f_i = \emptyset$ (resp. $g_i = \emptyset$).

The RFAO *array contents for variables x and y as generated by algorithm* CountermodelExtractLQ *after each $\forall$-reduction step in the proof of Figure 2 are shown below.*

0.	$x:$	$\begin{bmatrix} \ \end{bmatrix}$	$y:$	$\begin{bmatrix} \ \end{bmatrix}$
1.	$x:$	$\begin{bmatrix} \ \end{bmatrix}$	$y:$	$\begin{bmatrix} clause(f^e_{10}, d, \overline{g_{10}}) \\ cube(\overline{f^e_{10}}, \overline{d}, \overline{g_{10}}) \end{bmatrix}$
2.	$x:$	$\begin{bmatrix} clause(\overline{f_{12}}) \\ cube(\overline{f_{12}}) \end{bmatrix}$	$y:$	$\begin{bmatrix} clause(f^e_{10}, d, \overline{g_{10}}) \\ cube(\overline{f^e_{10}}, \overline{d}, \overline{g_{10}}) \end{bmatrix}$

Note that f^e_{10}, which equals $(x \wedge f_{10}) \vee (\overline{x} \wedge \overline{f_{10}})$, stands for the effective literal of x^* in C_{10}.

After re-expressing the RFAO formulas in terms of the existential variables, we get the Herbrand function of x, $F[x] = (\overline{f_{12}}) \wedge (\overline{f_{12}}) = \overline{(\overline{b}f_7 + bf_8)} = \overline{(\overline{b}(\overline{a}f_1 + af_2) + b)} = a\overline{b}$, and the Herbrand function of y, $F[y] = (f^e_{10} + d + \overline{g_{10}}) \wedge (\overline{f^e_{10}}\ \overline{d}\ \overline{g_{10}}) = (\overline{f^e_{10}}\ \overline{d}\ \overline{g_{10}}) = \overline{(xf_{10}) + (\overline{x}\overline{f_{10}})}\overline{d}\overline{g_8} = \overline{(x(\overline{b}\overline{a} + b)) + (\overline{x}\overline{(\overline{b}\overline{a} + b)})}\overline{d}c = \overline{d}c$. It can be verified that, after substituting $F[x]$ and $F[y]$ for variables x and y in Φ_{mtx}, the obtained quantifier-free formula is indeed unsatisfiable.

CountermodelExtractLQ
input: a false QBF Φ and its LQ-res DAG $G_\Pi(V_\Pi, E_\Pi)$
output: a countermodel in `RFAO` formulas
begin
01 **foreach** universal variable x of Φ
02 `RFAO`$[x] := \emptyset$;
03 **foreach** vertex $v \in V_\Pi$ in topological order
04 **foreach** merged literal $l \in v.clause$
05 update $l.phase$ from its parent literal(s);
06 **if** $v.clause = reduce(u.clause)$
07 $C := v.shadcls$;
08 **foreach** universal literal l reduced from $u.clause$
09 $x := var(l)$;
10 **if** $x \in u.clause$
11 **push back** C to `RFAO`$[x]$;
12 **else if** $\overline{x} \in u.clause$
13 **push back** $\overline{C}$ to `RFAO`$[x]$;
14 **else if** $x^* \in u.clause$
15 **push back** $(C \vee \overline{l.phase})$ to `RFAO`$[x]$;
16 **push back** $(\overline{C} \wedge \overline{l.phase})$ to `RFAO`$[x]$;
17 **if** $v.clause$ is the empty clause
18 **return** `RFAO` formulas;
end

Figure 1: Algorithm extracting a countermodel from an LQ-resolution proof.

The correctness of algorithm *CountermodelExtractLQ* of Figure 1 is asserted by the following theorem.

Theorem 1. *Given a false QBF Φ and its LQ-resolution proof Π, the algorithm CountermodelExtractLQ(Φ, G_Π) produces a countermodel of Herbrand functions for the universal variables of Φ.*

To prove Theorem 1, we need to show that 1) the Herbrand functions returned by *CountermodelExtractLQ* obey the prefix order dependency (i.e., the Herbrand function $F[x]$ of universal variable x only refers to the variables with

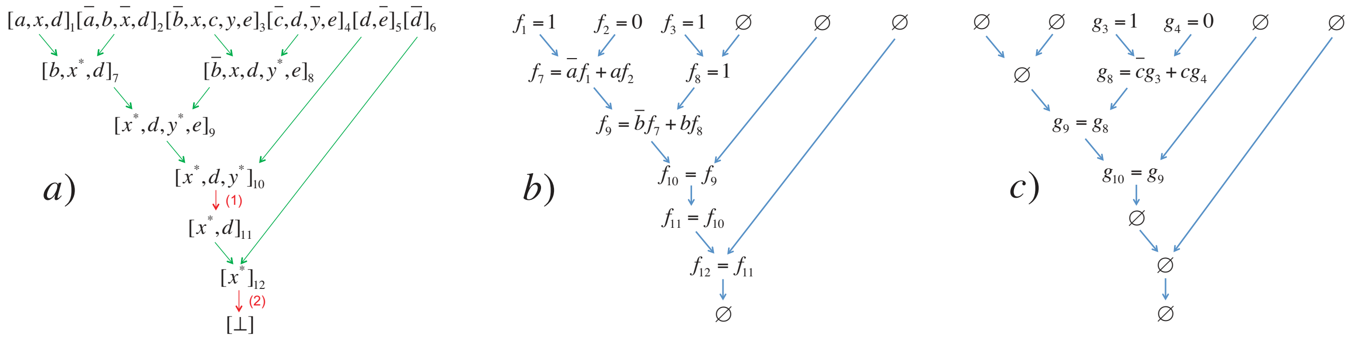

Figure 2: (a) DAG of LQ-resolution proof Π; (b) Phase functions for literals of variable x; (c) Phase functions for literals of variable y.

quantification levels less than that of x), and 2) their substitution for corresponding universal variables indeed makes the matrix Φ_{mtx} unsatisfiable. Proposition 1 establishes the first part, and Lemma 2 the second part.

Proposition 1. *Given a false QBF Φ and its LQ-resolution proof Π, let literal $l \in v.clause$ with $v \in V_\Pi$. If the truth or falsity of $l.elit$ refers (through recursive substitution of phase functions) to some variable x, then $lvl(x) \leq lvl(l)$.*

Proof. Observe that the proposition holds, by the definition of effective literals, for any literal l that is not a merged literal. Since the clauses in Φ_{mtx} do not involve any merged literals, the proposition holds for all the literals in the clauses of Φ_{mtx}. On the other hand, for a merged literal $l \in v.clause$, if $v.clause = reduce(u.clause)$ in Π, and l' denotes $l.ancestor$, then $l.elit = l'.elit$ according to the definition of effective literals. Now, if $v.clause = resolve(u_1.clause, u_2.clause)$ and $var(l) \notin vars(u_1.clause)$ (resp. $var(l) \notin vars(u_2.clause)$), then $l.elit = l'.elit(l)$, where l' represents $l.ancestor$. On the other hand, if $v.clause = resolve(u_1.clause, u_2.clause)$ under pivot variable p and $var(l) \in vars(u_1.clause)$ and $var(l) \in vars(u_2.clause)$, then this is an LQ-resolution step (if it is not, then $l \in v.clause$ cannot be a merged literal). By the rule of LQ-resolution, $lvl(p) < lvl(l)$. Therefore, regardless of the origin of v from either universal reduction or resolution, if the proposition holds for any literal in each parent of $v \in V_\Pi$, then it must also hold for the literals in $v.clause$. By induction, the proposition holds for any $l \in v.clause$ for any $v \in V_\Pi$. $\square$

Similarly to Proposition 1, we can prove that for any $l \in v.clause$, function $l.phase$ only refers to the variables with quantification level less than $lvl(l)$. Taking into account the construction of procedure *CountermodelExtractLQ*, the following corollary follows.

Corollary 1. *Herbrand functions returned by the algorithm CountermodelExtractLQ obey the prefix order dependency.*

Proof. Given a universal reduction step $v.clause = reduce(u.clause)$, and a literal $l \in u.clause$ such that $l \notin v.clause$ (i.e., l is a reduced literal), it holds that $lvl(l) > lvl(l')$ for any $l' \in v.clause$ by the definition of universal reduction. Therefore, by Proposition 1, the truth or falsity of $v.shadcls$ only refers to the variables with quantification level less than $lvl(l)$. Similarly function $l.phase$ only refers to the variables with quantification level less than $lvl(l)$. Hence for each universal variable $var(l)$, its corresponding RFAO node array only refers to the variables with quantification level less than $lvl(l)$. $\square$

To prove that Herbrand functions returned by *CountermodelExtractLQ* form a countermodel, we follow a similar line of reasoning as in (Balabanov and Jiang 2012). However, the algorithm *CountermodelExtractLQ* stores shadow clauses (cubes) in RFAO arrays rather than ordinary clauses (cubes) and it considers universal reduction on merged literals. Note that, if no LQ-resolution step is present in a proof (i.e., no merged literal appears), then *CountermodelExtractLQ* returns exactly the same Herbrand functions as *Countermodel_construct*.

Regardless of the change from ordinary to shadow clauses (cubes), the two RFAO formula properties listed in Preliminaries remain intact. In the following, when we say that some shadow clause of vertex v is α-implied, we mean it is α-implied by the shadow clause (the conjunction of the shadow clauses) corresponding to the parent vertex (parent vertices) of v. Lemma 1 shows the properties of α-implication among shadow clauses.

Lemma 1. *Given a false QBF Φ and its LQ-resolution proof Π, let $v \in V_\Pi$ and $v.clause = resolve(u_1.clause, u_2.clause)$ in Π with pivot literals $p \in u_1.clause$ and $\overline{p} \in u_2.clause$. Then $(u_1.shadcls|_\alpha \wedge u_2.shadcls|_\alpha) \rightarrow v.shadcls|_\alpha$ under any assignment α to the variables in Φ.*

Proof. Let $vars(u_1.clause) = \{p\} \cup L_1 \cup M$ and $vars(u_2.clause) = \{p\} \cup L_2 \cup M$, where L_1 and L_2 are the sets of variables *local* to $u_1.clause$ and $u_2.clause$, respectively, and M is the set of their *common* variables, excluding the pivot variable p. If $l.elit|_\alpha = 1$ for some $l \in v.clause$, then by the definition of shadow clauses $v.shadcls|_\alpha = 1$. Therefore $(u_1.shadcls|_\alpha \wedge u_2.shadcls|_\alpha) \rightarrow v.shadcls|_\alpha$.

Consider the other case that $l.elit|_\alpha = 0$ for each $l \in v.clause$. Without loss of generality, assuming $\overline{p} \in \alpha$, we prove $u_1.shadcls|_\alpha = 0$ in the following. (Assuming $p \in \alpha$, $u_2.shadcls|_\alpha = 0$ can be proved similarly).

For each l_1 with $var(l_1) \in L_1$ by the definition of effective literals, it holds that $l_1.elit = l'_1.elit$, where $l'_1 \in u_1$ is a parent of l_1. Hence if $l_1.elit|_\alpha = 0$, then $l'_1.elit|_\alpha = 0$. Further, for each literal l with $var(l) \in M$, we have $l.phase|_\alpha = ((\overline{p} \wedge l'.phase) \vee (p \wedge l''.phase))|_\alpha = l'.phase|_\alpha$, where $l' \in u_1$ and $l'' \in u_2$ are the parents of l. Therefore $l.elit|_\alpha = (x \leftrightarrow l.phase)|_\alpha = (x \leftrightarrow l'.phase)|_\alpha = l'.elit|_\alpha = 0$, where $x = var(l)$. Consequently, $l'.elit|_\alpha = 0$ for each l' with $var(l') \in \{p\} \cup L_1 \cup M$, and thus $u_1.shadcls|_\alpha = 0$.

Thereby $(u_1.shadcls|_\alpha \wedge u_2.shadcls|_\alpha) \rightarrow v.shadcls|_\alpha$ under any assignment α, and the lemma follows. $\square$

Finally, the following lemma shows that the substitution of all universal variables by their corresponding Herbrand functions returned by *CountermodelExtractLQ*(Φ, G_Π) indeed makes Φ_{mtx} unsatisfiable, and thus completes the proof of Theorem 1.

Lemma 2. *Given a false QBF Φ and its LQ-resolution proof Π, the algorithm CountermodelExtractLQ(Φ, G_Π) returns Herbrand functions whose substitution for the corresponding universal variables makes the matrix Φ_{mtx} unsatisfiable.*

Proof. Given an assignment $\alpha_\exists$ to the existential variables of Φ, we show below that the constructed Herbrand functions induce an assignment $\alpha_\forall$ to the universal variables of Φ such that $\Phi_{\text{mtx}}|_\alpha = 0$ for $\alpha = \alpha_\exists \cup \alpha_\forall$.

Let V_D be the set of all vertices $v \in V_\Pi$ whose clauses were obtained by universal reduction in Π (i.e., $v.clause = reduce(u.clause)$ for some $u \in V_\Pi$). Notice that algorithm *CountermodelExtractLQ* processes G_Π in a topological order, meaning that a clause in Π is processed only after all of its ancestor clauses are processed. Therefore we consider all shadow clauses $v.shadcls$ with $v \in V_D$ in the topological order under the assignment α. First, assume that for each $v \in V_D$ its corresponding shadow clause $v.shadcls$ satisfies $v.shadcls|_\alpha = 1$, and therefore is α-implied. By Lemma 1, we conclude that in this case $v.shadcls$ is α-implied for any $v \in V_\Pi$. Hence the empty shadow clause (that corresponds to the empty clause) is α-inherited, and thus $\Phi_{\text{mtx}}|_\alpha = 0$.

Second, assume that for some vertex $v \in V_D$, the corresponding shadow clause $v.shadcls|_\alpha = 0$. Let $v'.shadcls$ be the first such encountered shadow clause. Denote u' as the parent of v'. Note that $u'.shadcls$ and all its ancestors must be α-inherited (as all the ancestors of $u'.shadcls$ are α-implied). Let $C_{u' \setminus v'}$ be the set of all the universal literals being reduced from $u'.clause$ to get $v'.clause$. By the definition of shadow clauses, we have $u'.shadcls = v'.shadcls \bigcup_{l \in C_{u' \setminus v'}} l.elit$. It follows that

$$u'.shadcls|_\alpha = v'.shadcls|_\alpha \vee \bigvee_{l \in C_{u' \setminus v'}} l.elit|_\alpha = \bigvee_{l \in C_{u' \setminus v'}} l.elit|_\alpha.$$

Next we show that our construction yields $l.elit|_\alpha = 0$ for any $l \in C_{u' \setminus v'}$, therefore leading to $u'.shadcls|_\alpha = 0$. For each variable $x = var(l)$ with $l \in C_{u' \setminus v'}$, we examine its corresponding `RFAO[`x`]`. Since $w.shadcls|_\alpha = 1$ for any ancestor $w \in V_D$ of v', the value of Herbrand function $F[x]$ under α is not determined by any of the RFAO nodes that were added to the RFAO array before the reduction of x happens in u' (by Property 1 of RFAO arrays mentioned in Preliminaries). We now analyze the following three cases for the reduction of x in $u'.clause$:

1. For x being reduced as a positive literal l, the corresponding RFAO clause node evaluates to 0 (since $v'.shadcls|_\alpha = 0$), and hence $F[x]|_\alpha = 0$ (by Property 2 of RFAO arrays mentioned in Preliminaries).
2. Similarly, for x being reduced as a negative literal l, the corresponding RFAO cube node evaluates to 1 (since $(v'.shadcub)|_\alpha = 1$), and hence $F[x]|_\alpha = 1$.
3. For x being reduced as a merged literal l, by algorithm *CountermodelExtractLQ* two RFAO nodes, namely, clause node $Node_1 = (v'.shadcls \vee \overline{l.phase})$ and cube node $Node_2 = (v'.shadcub \wedge \overline{l.phase})$, are added to the RFAO array. Now, if $l.phase|_\alpha = 0$, then $Node_1|_\alpha = Node_2|_\alpha = 1$. Therefore $F[x]|_\alpha = 1$ (determined by the RFAO cube $Node_2|_\alpha = 1$) and $l.elit|_\alpha = ((F[x] \wedge l.phase) \vee (\overline{F[x]} \wedge \overline{l.phase}))|_\alpha = 0$. On the other hand, if $l.phase|_\alpha = 1$, then $Node_1|_\alpha = Node_2|_\alpha = 0$. Therefore $F[x]|_\alpha = 0$ (determined by the RFAO clause $Node_1|_\alpha = 0$) and $l.elit|_\alpha = 0$ similarly.

By the above analysis, it holds that $l.elit|_\alpha = 0$ for any $l \in C_{u' \setminus v'}$. Therefore $u'.shadcls|_\alpha = 0$, and taking into account that $u'.shadcls$ is α-inherited we establish $\Phi_{\text{mtx}}|_\alpha = 0$. $\square$

Algorithm *CountermodelExtractLQ* has a linear time complexity because each vertex is processed once by traversing its literals once. When a clause or cube is pushed into an RFAO array, only its ID needs to be stored.

Experimental Results

We implemented the countermodel extraction algorithm of Figure 1 and its dual model extraction algorithm in the C++ language in the tool ResQu[2], and named the new implementation as ResQu-Lp. 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 KBKF family of formulas (Kleine Büning, Karpinski, and Flögel 1995) and the benchmark formulas of QBFEVAL'10 (QBFEVAL 2010). With the test cases, we evaluated the performance of ResQu-Lp in terms of runtime, memory consumption, and the quality of the produced countermodels (i.e., the circuit size and depth of the constructed Herbrand functions).

We applied the QBF solver DepQBF (Lonsing and Biere 2010) without and with long-distance resolution (command line parameter "`--long-dist-res`" (Egly, Lonsing, and Widl 2013) to produce Q- and LQ-resolution proofs, respectively. Then we compared the model and countermodel extraction in three settings: ResQu (Balabanov and Jiang 2012) for Q-resolution proofs, and ResQu-LQU (Balabanov, Widl, and Jiang 2014) and ResQu-Lp for LQ-resolution proofs. Further, to validate the constructed models and countermodels, the SAT solver MiniSat (Eén

[2]http://alcom.ee.ntu.edu.tw/resqu/

Table 2: Time statistics (in seconds) for KBKF instances.

t	DepQBF	ResQu		DepQBF-L	ResQu-LQU		ResQu-Lp	
	slv	ext	vld	slv	ext	vld	ext	vld
10	0.1	0.1	0.1	0.0	0	0.1	0.0	0.1
11	0.2	0.1	0.3	0.0	0.1	0.1	0.0	0.1
12	0.5	0.3	0.7	0.0	0.1	0.1	0.0	0.1
13	1.2	0.6	2.3	0.0	0.3	0.1	0.0	0.1
14	2.8	1.4	7.6	0.0	0.7	0.1	0.0	0.1
15	6.8	3.0	30.5	0.0	1.8	0.1	0.0	0.1
16	16.6	6.7	-1	0.0	3.9	0.8	0.0	0.1
17	41.0	15.1	-1	0.0	9.4	5.4	0.0	0.1
18	102.8	33.6	-1	0.0	20.5	40.4	0.0	0.1
19	261.5	74.1	-1	0.0	48.8	-1	0.0	0.1
20	674.2	175.7	-1	0.0	95.1	-1	0.0	0.1
30	-1	-	-	0.0	-1	-	0.0	0.1
40	-1	-	-	0.0	-1	-	0.1	0.1
50	-1	-	-	0.0	-1	-	0.1	0.1
60	-1	-	-	0.0	-1	-	0.1	0.1
70	-1	-	-	0.0	-1	-	0.2	0.1
80	-1	-	-	0.0	-1	-	0.3	0.1
90	-1	-	-	0.0	-1	-	0.3	0.1
100	-1	-	-	0.0	-1	-	0.4	0.1

and Sörensson 2003) embedded in ABC (Brayton and Mishchenko 2010) was applied.

Table 2 shows the runtime statistics for QBF solving, countermodel extraction, and countermodel validation for the KBKF family of QBF instances, which are all false and hard for Q-resolution, but easy for LQ-resolution, to refute (Egly, Lonsing, and Widl 2013; Kleine Büning, Karpinski, and Flögel 1995). Column 1 lists the instances indexed by parameter t, which reflects the formula size (there are $3t+2$ variables and $4t+1$ clauses in the t^{th} member of the KBKF family). Columns denoted by DepQBF and DepQBF-L indicate QBF solver settings without and with long distance resolution, respectively. The columns "slv," "ext," and "vld" report the CPU runtime in seconds for QBF solving, certificate extraction, and certificate validation, respectively. An entry containing "-1" indicates that the computation was either out of the time limit of 1,000 seconds, or out of the memory limit of 25 GB. An entry containing "-" indicates that the data is not available. As evident from Table 2, DepQBF required runtime (and, in fact, yielded proof size) exponential in t, whereas DepQBF-L required runtime (and, in fact, yielded proof size) linear in t. ResQu was able to extract countermodels from all the proofs produced by DepQBF for $t \leq 20$, but the cases with $t \geq 16$ could not be validated within the time limit. On the other hand, ResQu-LQU was only able to extract countermodels for $t \leq 20$ from the proofs produced by DepQBF-L within the time limit, while the cases with $t \geq 18$ could not be validated within the time limit. In comparison, ResQu-Lp easily accomplished every extraction task within 0.4 seconds under 6 MB memory consumption; its produced countermodels were all validated within 0.1 seconds.

Continuing the above experiments, Table 3 shows the circuit sizes in terms of the number of and-inverter graph (AIG) nodes, denoted "#AIG," and circuit depths, denoted "#LVL," to evaluate the quality of the extracted Herbrand functions. An entry containing "-" indicates that either the data is unavailable or ABC failed to read in the certificate due to its excessive size. Note that the simplified AIGs of the Herbrand functions of KBKF instance t produced by ResQu-Lp have t AIG nodes and are significantly smaller than the corresponding functions produced by other methods.

To evaluate the performance of ResQu-Lp on application benchmark formulas, we conducted the above experiments on the QBFEVAL'10 instances. Out of the 569 formulas, DepQBF-L was able to generate resolution proofs for 177 false QBFs and 98 true QBFs within the limits of 1,000 seconds, 2 GB RAM, and 1 GB proof size. Among the 177 proofs of falsity, 144 involved only Q-resolution proofs, and 33 involved LQ-resolution proofs; on the other hand, the 98 proofs of truth all involved only Q-resolution proofs. Therefore, we focused on the 33 instances where LQ-resolution proofs were available for comparison. To compare the quality of the extracted certificates, Figure 3 plots the results in terms of AIG size and circuit depth. The x axis in the figure

Table 3: Certificate sizes for KBKF instances.

t	ResQu		ResQu-LQU		ResQu-Lp	
	#AIG	#LVL	#AIG	#LVL	#AIG	#LVL
10	3.1k	1.6k	93	10	10	2
11	6.1k	3.1k	231	13	11	2
12	11.9k	6.1k	532	16	12	2
13	24.6k	12.3k	840	17	13	2
14	47.5k	24.5k	1.5k	19	14	2
15	95.1k	49.1k	2.7k	20	15	2
16	253.9k	82.0k	16.4k	29	16	2
17	-	-	61.6k	32	17	2
18	-	-	132.8k	33	18	2
19	-	-	-	-	19	2
20	-	-	-	-	20	2
30	-	-	-	-	30	2
40	-	-	-	-	40	2
50	-	-	-	-	50	2
60	-	-	-	-	60	2
70	-	-	-	-	70	2
80	-	-	-	-	80	2
90	-	-	-	-	90	2
100	-	-	-	-	100	2

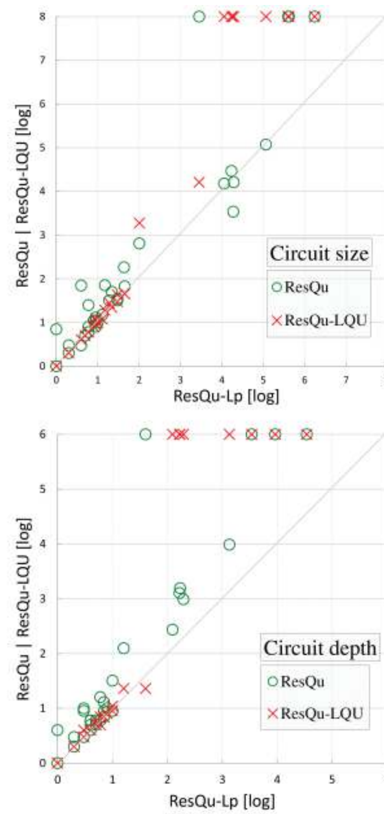

Figure 3: Comparison on certificate quality for application benchmarks.

corresponds to the results obtained by RESQU-LP, and the y axis corresponds to those obtained by RESQU and RESQU-LQU. Both axes are presented in the $\log_{10}$ scale, and an index k indicates 10^k. If a method failed to extract a certificate, then the corresponding result was set to the upper bound of the figure. We note that DEPQBF was not able to produce Q-resolution proofs for three formulas whose LQ-resolution proofs were obtainable. Hence, extracting certificates from LQ-resolution proofs is beneficial for certain applications. As shown in Figure 3, certificates produced by RESQU-LP are consistently smaller than those produced by RESQU-LQU. Furthermore RESQU-LQU was not able to produce certificates for eight instances, whereas RESQU-LP handled them without difficulty. The certificates extracted by RESQU-LQU are on average 16% smaller than those produced by RESQU in both AIG size and depth; in contrast, the certificates produced by RESQU-LP are on average 45% smaller in AIG size and 55% smaller in AIG depth than those extracted by RESQU. These results suggest not only the distinct value of LQ-resolution, but also the effectiveness of our algorithm in extracting high-quality certificates.

Conclusions

In this work we have shown that extracting a Herbrand-function countermodel (Skolem-function model) from any given LQ-resolution proof can be done in time linear with respect to the proof size. Apart from the theoretical advancement, we demonstrated the superiority of the new algorithm through experimental evaluation. Substantial savings on time and space resources were observed in crafted QBF instances as well as in application instances, though not as significant as the crafted instances. Further, we have shown experimentally that the new procedure can yield certificates of high quality suitable for synthesis and other applications.

Acknowledgements

This work was supported in part by the Ministry of Science and Technology (MOST) of Taiwan under grants 101-2923-E-002-015-MY2, 102-2221-E-002-232, and 103-2221-E-002-273; POLARIS (PTDC/EIA-CCO/123051/2010) and INESC-ID's PIDDAC funding PEst-OE/EEI/LA0021/2011; WWTF project ICT10-018 and FWF grant S11409-N23.

References

Balabanov, V., and Jiang, J.-H. R. 2012. Unified QBF certification and its applications. *Formal Methods in System Design* 41:45–65.

Balabanov, V.; Widl, M.; and Jiang, J.-H. R. 2014. QBF resolution systems and their proof complexities. In *International Conference on Theory and Applications of Satisfiability Testing (SAT)*, 154–169. Springer.

Benedetti, M., and Mangassarian, H. 2008. QBF-based formal verification: Experience and perspectives. *Journal on Satisfiability, Boolean Modeling and Computation* 5(1-4):133–191.

Benedetti, M. 2005. sKizzo: A suite to evaluate and certify QBFs. In *International Conference on Automated Deduction (CADE)*. Springer. 369–376.

Brayton, R. K., and Mishchenko, A. 2010. ABC: An academic industrial-strength verification tool. In *International Conference on Computer Aided Verification (CAV)*, 24–40. Springer.

Dershowitz, N.; Hanna, Z.; and Katz, J. 2005. Bounded model checking with QBF. In *International Conference on Theory and Applications of Satisfiability Testing (SAT)*, volume 3569, 408–414. Springer.

Eén, N., and Sörensson, N. 2003. An extensible SAT-solver. In *International Conference on Theory and Applications of Satisfiability Testing (SAT)*, volume 2919, 502–518. Springer.

Egly, U.; Lonsing, F.; and Widl, M. 2013. Long-distance resolution: Proof generation and strategy extraction in search-based QBF solving. In *International Conference on Logic Programming and Automated Reasoning (LPAR)*, 291–308. Springer.

Giunchiglia, E.; Narizzano, M.; and Tacchella, A. 2006. Clause/term resolution and learning in the evaluation of quantified Boolean formulas. *Journal on Artificial Intelligence Research (JAIR)* 26:371–416.

Goultiaeva, A.; Van Gelder, A.; and Bacchus, F. 2011. A uniform approach for generating proofs and strategies for both true and false QBF formulas. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 546–553. AAAI Press.

Jiang, J.-H. R.; Lin, H.-P.; and Hung, W.-L. 2009. Interpolating functions from large Boolean relations. In *International Conference on Computer-Aided Design (ICCAD)*, 779–784. IEEE/ACM.

Kleine Büning, H.; Karpinski, M.; and Flögel, A. 1995. Resolution for quantified Boolean formulas. *Information and Computation* 117(1):12–18.

Kontchakov, R.; Pulina, L.; Sattler, U.; Schneider, T.; Selmer, P.; Wolter, F.; and Zakharyaschev, M. 2009. Minimal module extraction from DL-lite ontologies using QBF solvers. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 836–841. AAAI Press.

Lonsing, F., and Biere, A. 2010. DepQBF: A dependency-aware QBF solver (system description). *Journal on Satisfiability, Boolean Modeling and Computation* 7:71–76.

Marques-Silva, J. P., and Sakallah, K. A. 1996. GRASP – a new search algorithm for satisfiability. In *International Conference on Computer-Aided Design (ICCAD)*, 220–227. IEEE/ACM.

QBFEVAL. 2010. QBF solver evaluation portal. http://www.qbflib.org/qbfeval/.

Rintanen, J. 2007. Asymptotically optimal encodings of conformant planning in QBF. In *National Conference on Artificial Intelligence (AAAI)*, 1045–1050. AAAI Press.

Schaefer, M., and Umans, C. 2002. Completeness in the polynomial-time hierarchy: a compendium. *SIGACT News* 33(3):32–49.

Skolem, T. 1928. Über die mathematische Logik. *Norsk Mat. Tidsskrift* 106:125–142. [Translation in *From Frege to Gödel, A Source Book in Mathematical Logic*, J. van Heijenoort, Harvard Univ. Press, 1967.].

Staber, S., and Bloem, R. 2007. Fault localization and correction with QBF. In *International Conference on Theory and Applications of Satisfiability Testing (SAT)*, 355–368. Springer.

Zhang, L., and Malik, S. 2002. Conflict driven learning in a quantified Boolean satisfiability solver. In *International Conference on Computer-Aided Design (ICCAD)*, 442–449. IEEE/ACM.