The Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI-23)

Second-Order Quantified Boolean Logic

Jie-Hong R. Jiang

Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan
jhjiang@ntu.edu.tw

Abstract

Second-order quantified Boolean formulas (SOQBFs) generalize quantified Boolean formulas (QBFs) by admitting second-order quantifiers on function variables in addition to first-order quantifiers on atomic variables. Recent endeavors establish that the complexity of SOQBF satisfiability corresponds to the exponential-time hierarchy (EXPH), similar to that of QBF satisfiability corresponding to the polynomial-time hierarchy (PH). This fact reveals the succinct expression power of SOQBFs in encoding decision problems not efficiently doable by QBFs. In this paper, we investigate the second-order quantified Boolean logic with the following main results: First, we present a procedure of quantifier elimination converting SOQBFs to QBFs and a game interpretation of SOQBF semantics. Second, we devise a sound and complete refutation-proof system for SOQBF. Third, we develop an algorithm for countermodel extraction from a refutation proof. Finally, we show potential applications of SOQBFs in system design and multi-agent planning. With these advances, we anticipate practical tools for development.

1 Introduction

Theoretical and practical advancements in Boolean satisfiability (SAT) (Biere et al. 2021) have made it an important technology enabling various complex computation tasks in artificial intelligence, hardware design, software engineering, among others. The frontiers of satisfiability research are continuously extended to (non-)Boolean queries involving different types of quantifiers. Notably, the quantified Boolean formula (QBF) allows existential and universal quantifiers to appear in an expression. The dependency quantified Boolean formula (DQBF) (Peterson and Reif 1979) further extends QBF by allowing Henkin quantifiers (Henkin 1961) to explicitly specify the dependencies of existential variables on universal variables. These extensions make the underlying computational complexity lifted from the NP-completeness of SAT (Cook 1971), to the PSPACE-completeness of QBF (Stockmeyer and Meyer 1973), and to the NEXP-completeness of DQBF (Peterson and Reif 1979). To date, practical decision procedures for checking the QBF and DQBF satisfiability are under active research, and various solvers and preprocessors are available (Biere et al. 2021; Scholl and Wimmer 2018).

The second-order quantified Boolean formula (SOQBF) extends QBF in the allowance of function variables and their quantification. Its satisfiability checking has complexity corresponding to the exponential-time hierarchy (EXPH) (Dawar, Gottlob, and Hella 1998), as was established in (Lohrey 2012; Lück 2016). The connection between SOQBF and EXPH is similar to that between QBF and the polynomial-time hierarchy (PH) (Stockmeyer 1976). Essentially, DQBFs are an elementary class of SOQBFs with only one level of existential quantifiers over function variables. SOQBFs can be powerful to succinctly encode decision problems that have no succinct QBF encoding. There are recent efforts studying variants (Hannula et al. 2016) and special fragments (Hannula et al. 2020) of SOQBFs.

In this work, we study second-order quantified Boolean logic in the aspects of representation and interpretation, proof system, and countermodel extraction. The main results include

- presenting a procedure of quantifier elimination converting SOQBFs to QBFs and a game interpretation of SOQBF semantics (Section 3),
- devising a sound and complete refutation proof system (Section 4),
- developing an algorithm of extracting a countermodel, namely, a winning strategy under the game-theoretic interpretation of SOQBF, from a refutation proof (Section 5), and
- showing potential applications of SOQBFs in system design and multi-agent planning (Section 6).

With these advances, we anticipate practical tools whose development may open new avenues for tackling computation problems too challenging to be formulated previously.

2 Preliminaries

In this work, we are concerned with two types of variables, including atomic variables, represented with lower-case letters, e.g., x, y, and function variables, represented with, e.g., f, g. In the sequel, we do not specify a variable type when it is clear from the context. A variable set is represented with a upper-case letter, e.g., $X = \{x_1, \ldots, x_n\}, F = \{f_1, \ldots, f_n\}$. The cardinality of a set X is denoted as $|X|$.

For notational convention, Boolean connectives are denoted with "$\wedge$" (sometimes omitted) for conjunction, "$\vee$" for disjunction, "$\rightarrow$" for implication, "$\leftrightarrow$" for biconditional, and "$\neg$" for negation. A *literal* is a variable (positive polarity) or the negation of a variable (negative polarity). A *clause* is a disjunction of literals; a *cube* is a conjunction of literals. A *conjunctive normal form* (CNF) formula is a conjunction of clauses. Alternatively, a clause or cube is denoted with a set of literals, and a CNF formula is denoted with a set of clauses. In the sequel, we assume that a CNF formula is free of tautological clauses, namely, clauses containing literals of the same variable of opposite polarities are removed.

When an atomic variable x is valuated, denoted $[\![x]\!]$, it takes on a value from the Boolean domain $\mathbb{B} = \{0, 1\}$. A function variable f is associated with a set of atomic variables, called the *support set* of f, denoted $\mathsf{S}(f)$. When f is valuated, denoted $[\![f]\!]$, it takes on a Boolean function $[\![f]\!] : \mathbb{B}^{|\mathsf{S}(f)|} \rightarrow \mathbb{B}$ over variables $\mathsf{S}(f)$. We remark that the support set of a function variable may, in general, contain function variables. However, in the context of second order quantified Boolean logic, the dependency imposed by $g \in \mathsf{S}(f)$, for some function variable g, can be rewritten by replacing g in $\mathsf{S}(f)$ with a fresh new atomic variable x_g and asserting the equivalence $x_g \leftrightarrow g$. Hence, without loss of generality, we assume the support set of a function variable consists of only atomic variables.

An assignment α over atomic variables X is a mapping $\alpha : X' \rightarrow \mathbb{B}$ for $X' \subseteq X$ that valuates each $x \in X'$. We alternatively represent an assignment on atomic variables as a cube. E.g., the assignment α over $X = \{x_1, x_2\}$ with $\alpha(x_1) \mapsto 1, \alpha(x_2) \mapsto 0$ is represented as $\alpha = x_1 \neg x_2$; in the special case when $X' = \emptyset$, the cube representation of the corresponding assignment is $\alpha = 1$, i.e., a cube without any literals. An assignment α is called *full* with respect to an atomic variable set X if every variable $x_i \in X$ appears in the cube representation of α. Otherwise, α is called *partial* with respect to X. Given an assignment α over variables X, its projection on variables X', denoted $\alpha_{\downarrow X'}$, is a submapping of α on variables $X \cap X'$. E.g., the projection of assignment $\alpha = x_1 \neg x_2$ on $X' = \{x_2, x_3\}$ is $\alpha_{\downarrow X'} = \neg x_2$. Similarly, an assignment α over function variables F maps each $f \in F$ to a function $\alpha(f) : \mathbb{B}^k \rightarrow \mathbb{B}$ for $k = |\mathsf{S}(f)|$. In the sequel, we do not specify the domain type (atomic or function variables) of an assignment when it is clear from the context.

Given a formula φ, we define the *cofactor* of φ with respect to an assignment α over atomic variables X, denoted $\varphi|_\alpha$, as the induced formula of φ after substituting variables X with their respective mapped values under α, and replacing every function variable f, for $\mathsf{S}(f) \cap X \neq \emptyset$, with a new function variable $f^{\alpha_{\downarrow \mathsf{S}(f)}}$ with support set $\mathsf{S}(f) \setminus X$, which denotes an instantiated version of f under assignment $\alpha_{\downarrow \mathsf{S}(f)}$. E.g., for formula $\varphi = (\neg f_1 \vee f_2 \vee x_1 \vee x_3)(f_1 \vee \neg f_2 \vee x_2 \vee \neg x_3)$ with $\mathsf{S}(f_1) = \{x_1, x_3\}$ and $\mathsf{S}(f_2) = \{x_2, x_3\}$ and assignment $\alpha = x_1 \neg x_2$, we have $\varphi|_\alpha = (f_1^{x_1} \vee \neg f_2^{\neg x_2} \vee \neg x_3)$. Note that the valuation $[\![f^{\alpha_{\downarrow \mathsf{S}(f)}}]\!]$ is a function referring only to variables $\mathsf{S}(f) \setminus X$ and it equals the function $[\![f]\!]$ induced under assignment α. That is, $[\![f^{\alpha_{\downarrow \mathsf{S}(f)}}]\!]$ is a projected function valuation of f with respect to assignment $\alpha_{\downarrow \mathsf{S}(f)}$.

3 Second-Order Quantified Boolean Formula

Syntax

A *second-order quantified Boolean formula* (SOQBF) Φ can be inductively defined in the Backus-Naur form with the following rules.

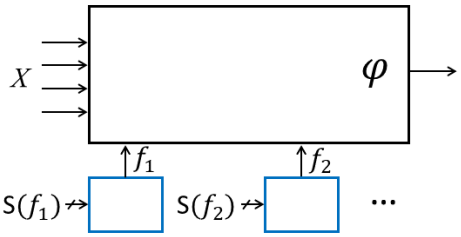
$$\Phi ::= 0 \mid 1 \mid x \mid f \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid \exists x.\Phi \mid \exists f.\Phi \qquad (1)$$

for x and f being atomic and function variables, respectively. A quantifier $\forall$ or $\exists$ is called a *first-order* (resp. *second-order*) *quantifier* if the variable bound by it is an atomic (resp. a function) variable. I.e., in the above construction rules, the quantifiers for $\exists x$ and $\exists f$ are first-order and second-order quantifiers, respectively. For brevity, the construction rules with quantifier $\forall$ and Boolean connective $\vee$ are not shown as they are similar to those of quantifier $\exists$ and Boolean connective $\wedge$, and also can be constructed by the $\neg$ and $\exists$ rules and the $\neg$ and $\wedge$ rules, respectively. A function variable in an SOQBF is specified with a fixed arity. There are two possible choices to describe the arguments of an n-ary function variable. One is to list all its arguments explicitly, and in this case a function variable may have multiple appearances in a formula with different arguments. E.g., in expression $f(f(x, y), z)$, the 2-ary function variable f appears twice with different arguments. The other is to assume that an n-ary function variable always appear in a formula with fixed arguments. E.g., in expression $(x \vee f \vee g)(z \vee \neg f \vee \neg g)$, assume f and g are 2-ary and 3-ary function variables with fixed arguments (x, y) and (x, y, z), respectively. We note that the former can always be converted to the latter through Ackermann's expansion (Ackermann 1954). E.g., $f(f(x, y), z)$ is converted to $\exists w.f_1(w, z) \wedge (w \leftrightarrow f_2(x, y))$ along with the constraint $\forall x, y, z, w.((x \leftrightarrow w) \wedge (y \leftrightarrow z)) \rightarrow (f_1(w, z) \leftrightarrow f_2(x, y))$ for w being a fresh new atomic variable. The formulas before and after the conversion are equisatisfiable. Therefore, without loss of generality, we assume a function variable has fixed arguments.

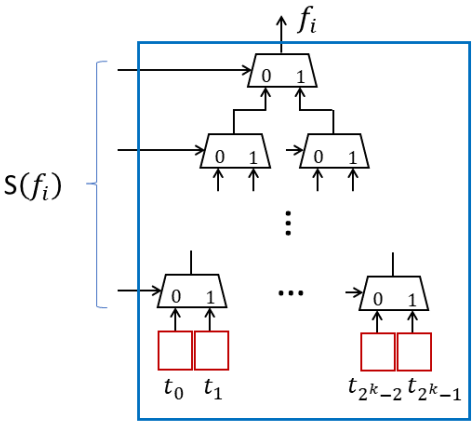
Any SOQBF can be converted to the following prenex form

$$\mathcal{Q}_1 F_1, \mathcal{Q}_2 F_2, \ldots, \mathcal{Q}_n F_n, Q_1 X_1, \ldots, Q_m X_m.\varphi, \qquad (2)$$

where $\mathcal{Q}_i, Q_j \in \{\forall, \exists\}$, for $\mathcal{Q}_i \neq \mathcal{Q}_{i+1}$ and $Q_j \neq Q_{j+1}$, F_i and X_j are nonempty sets of function and atomic variables, respectively, with $\mathsf{S}(f) \subseteq X$ for any $f \in F_i$, and φ is a quantifier-free formula expressed in terms of variables $F_1 \cup \ldots \cup F_n \cup X_1 \cup \ldots \cup X_m$. In Eq. (2), the quantifier part on the left is called the *prefix*, and φ is called the *matrix*. A function variable $f \in F_i$ bound by quantifier $\mathcal{Q}_i$ is said to be of (second-order) *quantification level* i, denoted $\mathsf{lev}(f) = i$. Similarly, an atomic variable $x \in X_i$ bound by quantifier Q_i is said to be of first-order *quantification level* i. Also, we use the notation $\mathsf{qnt}(f)$ to refer to the quantifier type of variable f. In this work, we assume a formula is totally quantified, that is, every variable is bound by some quantifier.



(a) Block diagram of matrix circuit.



(b) Detailed circuit for the function-variable block in (a).

Figure 1: Matrix circuit of an SOQBF.

As to be shown, Eq. (2) can always be rewritten into an SOQBF with a single first-order quantification level, i.e., $m = 1$, of the form

$$\mathcal{Q}_1 F_1, \mathcal{Q}_2 F_2, \ldots, \mathcal{Q}_n F_n, \forall X.\varphi, \text{ or} \quad (3)$$
$$\mathcal{Q}_1 F_1, \mathcal{Q}_2 F_2, \ldots, \mathcal{Q}_n F_n, \exists X.\varphi. \quad (4)$$

As Eq. (3) and Eq. (4) are dual to each other, any property obtained for Eq. (3) can be carried to Eq. (4) by duality. Hence, in the sequel, we shall primarily consider Eq. (3).

Semantics

Consider the SOQBF Φ of Eq. (2). The matrix φ over function variables $f_1, f_2, \ldots$ and atomic variables $X = \bigcup_{i=1}^{m} X_i$ can be viewed as a circuit as illustrated in Fig. 1, where each box f_i of the circuit in Fig. 1(a) corresponds to a multiplexer tree (or a binary decision diagram) controlled by the support set $\mathsf{S}(f_i) \subseteq X$ of f_i, $|\mathsf{S}(f_i)| = k$ as shown in Fig. 1(b). An assignment to function variable f_i corresponds to determining the truth-table values $t_0, \ldots, t_{2^k-1}$, for $k = |\mathsf{S}(f_i)|$, the red boxes in Fig. 1(b). For an assignment α to the function variables, the truth or falsity of the SOQBF Φ is determined by the QBF $Q_1 X_1, \ldots, Q_m X_m.\varphi'$, where φ' is the circuit induced by substituting all the truth-table values with constant 0 or 1 according to α. Therefore, the SOQBF Φ can be evaluated by a series of QBF evaluations with respect to function variable assignments following the second-order quantifiers $\mathcal{Q}_1 F_1, \mathcal{Q}_2 F_2, \ldots, \mathcal{Q}_n F_n$.

Consequently, the SOQBF of Eq. (2) is true if there exists a *model*, namely, a set of *Skolem functionals* for the existential function variables such that substituting each existential function variable with its corresponding Skolem functional makes the induced formula true. The Skolem functional of variable $f \in F_i$ for $\mathcal{Q}_i = \exists$ is a higher-order function that maps every assignment to variables $\bigcup_{j<i, \mathcal{Q}_j = \forall} F_j$ to an assignment to variable f. On the contrary, an SOQBF is false if there exists a *countermodel*, namely, a set of *Herbrand functionals* for the universal function variables such that substituting each universal function variable with its corresponding Herbrand functional makes the induced formula false. The Herbrand functional of variable $f \in F_i$ for $\mathcal{Q}_i = \forall$ is a higher-order function that maps every assignment to variables $\bigcup_{j<i, \mathcal{Q}_j = \exists} F_j$ to an assignment to variable f.

The truth and falsity of SOQBFs can be interpreted from a game-theoretic viewpoint. An SOQBF game is played by the exist-player, who controls the assignments to the existential function variables, and the forall-player, who controls the universal function variables. They take turns to assign function variables in ascending order of the quantification levels in the prefix. The former aims to satisfy the formula, whereas the latter aims to falsify the formula. The SOQBF is true if the exist-player has a winning strategy and false if the forall-player has a winning strategy. The winning strategy of the exist-player is effectively the model of Skolem functionals, and that of the forall-player is effectively the countermodel of Herbrand functionals. As an SOQBF is either true or false, the SOQBF game is determined, that is, either the exist-player or the forall-player has a winning strategy. This determinacy contrasts with the indeterminacy of the DQBF game, where the Henkin quantifier results in the asymmetric roles between the forall-player and the (multiple) exist-players (Balabanov, Chiang, and Jiang 2014). Note that when a DQBF is converted to an SOQBF by Skolemization, the converted SOQBF is of one level of function-variable quantification, in the form $\exists F_1, \forall X.\varphi$.

Quantifier Elimination and QBF Conversion

Two formulas are called *model-equivalent* if there is a bijection between the models of the two formulas. That is, a model of one formula uniquely maps to a model of the other formula, and vice versa. A formula-rewriting transformation is *model-preserving* if the formulas before and after the transformation are model-equivalent.

An SOQBF can be converted to a model-equivalent QBF via *ground instantiation*, where every function variable is instantiated with respect to a full assignment over its support set. The conversion can be done by iteratively eliminating the innermost atomic variable through formula expansion until no more atomic variable is left. Specifically, for the SOQBF

$$\mathcal{Q}_1 F_1, \ldots, \mathcal{Q}_n F_n, QX, \forall y.\varphi, \quad (5)$$

by eliminating variable y, it can be rewritten into a model-equivalent SOQBF as

$$\mathcal{Q}_1 F_1^y \cup F_1^{\neg y}, \ldots, \mathcal{Q}_n F_n^y \cup F_n^{\neg y}, QX.\varphi|_y \wedge \varphi|_{\neg y}, \quad (6)$$

where $F_i^y = \{f^{\alpha \wedge y} \mid f^\alpha \in F_i, y \in \mathsf{S}(f^\alpha)\} \cup \{f^\alpha \mid f^\alpha \in F_i, y \notin \mathsf{S}(f^\alpha)\}$ and $F_i^{\neg y} = \{f^{\alpha \wedge \neg y} \mid f^\alpha \in F_i, y \in \mathsf{S}(f^\alpha)\} \cup \{f^\alpha \mid f^\alpha \in F_i, y \notin \mathsf{S}(f^\alpha)\}$. Note that without instantiation, a function variable f is considered as f^α for $\alpha = 1$. Also, for

$$\mathcal{Q}_1 F_1, \ldots, \mathcal{Q}_n F_n, QX, \exists y.\varphi, \tag{7}$$

by eliminating variable y, it can be rewritten as

$$\mathcal{Q}_1 F_1^y \cup F_1^{\neg y}, \ldots, \mathcal{Q}_n F_n^y \cup F_n^{\neg y}, QX.\varphi|_y \vee \varphi|_{\neg y}. \tag{8}$$

Note that after all atomic variables are eliminated, any instantiated function variable, say f^α, is of an empty support set. That is, its valuation corresponds to a 0-ary function that represents a truth-table entry under assignment α of function $[\![f]\!]$. As a 0-ary function variable can be seen as an effective atomic variable, the final formula is effectively a QBF.

Example 1. *Consider the SOQBF*

$$\forall g, \exists f, \forall x_1, \exists x_2, \forall x_3.\varphi,$$

with the support sets $\mathsf{S}(g) = \{x_1, x_2\}$ *and* $\mathsf{S}(f) = \{x_1, x_3\}$*, and the matrix*

$$\varphi = (g \vee f \vee \neg x_1 \vee \neg x_2 \vee x_3)(g \vee \neg f).$$

Expanding on x_3 *yields the SOQBF*

$$\forall g, \exists F, \forall x_1, \exists x_2.\varphi',$$

where $F = \{f^{x_3}, f^{\neg x_3}\}$ *with* $\mathsf{S}(f^{x_3}) = \mathsf{S}(f^{\neg x_3}) = \{x_1\}$*, and*

$$\varphi' = (g \vee f^{\neg x_3} \vee \neg x_1 \vee \neg x_2)(g \vee \neg f^{\neg x_3})(g \vee \neg f^{x_3}).$$

Expanding further on x_2 *yields the SOQBF*

$$\forall G, \exists F, \forall x_1.\varphi'',$$

where $G = \{g^{x_2}, g^{\neg x_2}\}$ *with* $\mathsf{S}(g^{x_2}) = \mathsf{S}(g^{\neg x_2}) = \{x_1\}$*, and*

$$\varphi'' = (g^{x_2} \vee f^{\neg x_3} \vee \neg x_1)(g^{x_2} \vee \neg f^{\neg x_3})(g^{x_2} \vee \neg f^{x_3}) \vee (g^{\neg x_2} \vee \neg f^{\neg x_3})(g^{\neg x_2} \vee \neg f^{x_3}).$$

Finally, expanding on x_1 *yields the SOQBF,*

$$\forall G', \exists F'.\varphi''',$$

where $G' = \{g^{x_1 x_2}, g^{x_1 \neg x_2}, g^{\neg x_1 x_2}, g^{\neg x_1 \neg x_2}\}$ *with* $\mathsf{S}(g^{x_1 x_2}) = \mathsf{S}(g^{x_1 \neg x_2}) = \mathsf{S}(g^{\neg x_1 x_2}) = \mathsf{S}(g^{\neg x_1 \neg x_2}) = \{\}$*,* $F' = \{f^{x_1 x_3}, f^{x_1 \neg x_3}, f^{\neg x_1 x_3}, f^{\neg x_1 \neg x_3}\}$ *with* $\mathsf{S}(f^{x_1 x_3}) = \mathsf{S}(f^{x_1 \neg x_3}) = \mathsf{S}(f^{\neg x_1 x_3}) = \mathsf{S}(f^{\neg x_1 \neg x_3}) = \{\}$*, and*

$$\begin{aligned}\varphi''' = {} & ((g^{x_1 x_2} \vee f^{x_1 \neg x_3})(g^{x_1 x_2} \vee \neg f^{x_1 \neg x_3}) \\ & (g^{x_1 x_2} \vee \neg f^{x_1 x_3}) \vee \\ & (g^{x_1 \neg x_2} \vee \neg f^{x_1 \neg x_3})(g^{x_1 \neg x_2} \vee \neg f^{x_1 x_3})) \wedge \\ & ((g^{\neg x_1 x_2} \vee \neg f^{\neg x_1 \neg x_3})(g^{\neg x_1 x_2} \vee \neg f^{\neg x_1 x_3}) \vee \\ & (g^{\neg x_1 \neg x_2} \vee \neg f^{\neg x_1 \neg x_3})(g^{\neg x_1 \neg x_2} \vee \neg f^{\neg x_1 x_3})).\end{aligned}$$

After x_1, x_2, x_3 *are expanded, the final SOQBF is effectively a QBF as all the function variables are of zero arity.*

The model-equivalence between Eq. (5) and Eq. (6) and that between Eq. (7) and Eq. (8) can be verified by the semantics of SOQBF as defined in Section 3.

Proposition 1. *The SOQBFs of Eq.* (5) *and Eq.* (6) *are model-equivalent.*

Proposition 2. *The SOQBFs of Eq.* (7) *and Eq.* (8) *are model-equivalent.*

Normal Form Conversion

We note that any totally-quantified SOQBF can be converted to the normal form of Eq. (3) through a sequence of model-preserving transformation steps. Specifically, a non-prenex SOQBF can be converted to a prenex SOQBF by variable renaming; the second-order quantifiers can be placed on the left of the first-order quantifiers because the support-set information is explicitly specified for all function variables; the first-order quantifiers can be homogenized to all universal with a modified Skolemization procedure detailed in the following. Without loss of generality, the SOQBF

$$\mathcal{Q}_1 F_1, \ldots, \mathcal{Q}_n F_n, \forall X_1, \exists y, \forall X_2.\varphi, \tag{9}$$

with $\mathsf{S}(f) \subseteq X_1 \cup \{y\} \cup X_2$ for $f \in F_i$, can be converted to

$$\mathcal{Q}_1 F_1, \ldots, \mathcal{Q}_n F_n, \exists f_y, \forall X_1, \forall y, \forall X_2.(y \leftrightarrow f_y) \rightarrow \varphi, \tag{10}$$

where f_y is a fresh new function variable with $\mathsf{S}(f_y) = X_1$.

Proposition 3. *The SOQBFs of Eq.* (9) *and Eq.* (10) *are equisatisfiable, and are model-equivalent with respect to the models of function variables* $F_1 \cup \ldots \cup F_n$*.*

Proof. Their relation can be shown by QBF conversion. □

When the variable y is not in any support set of the function variables of Eq. (9), then the standard Skolemization applies, and Eq. (10) reduces to

$$\mathcal{Q}_1 F_1, \ldots, \mathcal{Q}_n F_n, \exists f_y, \forall X_1, \forall X_2.\varphi', \tag{11}$$

where φ' is obtained from φ by replacing every appearance of variable y in φ with variable f_y.

In Eq. (3), when $\mathcal{Q}_n = \forall$, it can be simplified to an equisatisfiable SOQBF of $n-1$ levels of second-order quantifiers as the following proposition states.

Proposition 4. *The SOQBF of Eq.* (3) *with* $\mathcal{Q}_n = \forall$ *is equisatisfiable to*

$$\mathcal{Q}_1 F_1, \mathcal{Q}_2 F_2, \ldots, \mathcal{Q}_{n-1} F_{n-1}, \forall X', \forall X.\varphi', \tag{12}$$

where X' *is derived from* F_n *with* $|X'| = |F_n|$ *by creating a fresh atomic variable* $x_f \in X'$ *for every function variable* $f \in F_n$*, and* φ' *is derived from* φ *by replacing function variable* f *with its corresponding atomic variable* x_f *in* φ*. Furthermore, the SOQBF of Eq.* (3) *with* $\mathcal{Q}_n = \forall$ *and* φ *in CNF can be simplified to an equisatisfiable formula with all appearances of variables* F_n *being removed from the prefix and matrix of Eq.* (3)*.*

Proof. Expanding variables X in Eq. (3) yields a QBF formula with its matrix being a conjunction of formulas. Because universal quantification and conjunction commute, we can reuse the same name for the instantiated variables of $f \in F_n$ in the universal quantification. Further when φ is in CNF, the variables of F_n can be removed by QBF $\forall$-reduction. □

That is, the SOQBF of Eq. (3) with $\mathcal{Q}_n = \forall$ is equisatisfiable to the same formula but changing the arity of every function $f \in F_n$ to zero. In this case, the number of effective second-order quantification levels is reduced by one. Similarly, the SOQBF of Eq. (4) with $\mathcal{Q}_n = \exists$ is equisatisfiable

to the same formula but changing the arity of every function $f \in F_n$ to zero, thus reducing one second-order quantification level.

We note that an SOQBF with one function variable depending on another function variable can be converted to the normal form where function variables depend only on atomic variables. Specifically, let f be a function variable with support set $\mathsf{S}(f)$ containing a function variable g. The SOQBF

$$..., \mathcal{Q}_i f, ..., \mathcal{Q}_j g, ..., \forall X.\varphi$$

can be re-expressed as

$$..., \mathcal{Q}_i f, ..., \mathcal{Q}_j g, ..., \forall X, \forall y.(y \leftrightarrow g) \rightarrow \varphi$$

for $g \in \mathsf{S}(f)$ being replaced with a fresh atomic variable y. We note that the above rewriting also works if the quantification level of g is smaller than f. A similar discussion is made in (Hannula et al. 2016).

Complexity

The exponential-time hierarchy (EXPH) is to the computational complexity of checking SOQBF satisfiability (SOQSAT) as the polynomial-time hierarchy (PH) is to that of checking QBF satisfiability. Recall the complexity classes Σ_k^{EXP}, for $\Sigma_k^{\mathsf{EXP}} = \mathsf{NEXP}^{\Sigma_{k-1}^{\mathsf{P}}}$. For $k = 1$, $\Sigma_1^{\mathsf{EXP}} = \mathsf{NEXP}$, which is the complexity of satisfiability checking for the S-form DQBF (Balabanov, Chiang, and Jiang 2014). By duality, complexity classes Π_k^{EXP}, for $\Pi_k^{\mathsf{EXP}} = \mathsf{coNEXP}^{\Sigma_{k-1}^{\mathsf{P}}}$. For $k = 1$, $\Pi_1^{\mathsf{EXP}} = \mathsf{coNEXP}$, which is the complexity of satisfiability checking for the H-form DQBF (Balabanov, Chiang, and Jiang 2014).

Let SOQSAT_k be the satisfiability of SOQBFs with k levels of second-order quantifiers and $\mathcal{Q}_1 = \exists$. The following statement, also in (Lück 2016), can be established by directly encoding the execution of an alternating Turing machine (ATM) (Chandra, Kozen, and Stockmeyer 1981) with $k-1$ alternations in an SOQBF of k levels of second-order quantification. (We omit the proof due to the page limit.)

Theorem 1. *For any $k \geq 1$, SOQSAT_k is Σ_k^{EXP}-complete.*

Consider SOQSAT without bounding the quantification levels of an SOQBF. We note that as long as the number of alternations of the ATM is polynomial in the input size, the computation table of an exponential-time execution of the ATM can be efficiently encoded in an SOQBF of polynomial size. Let AEXP(poly) be the class of problems that can be decided by an ATM with k alternations for k polynomial in the input size. Consequently, the following corollary, also in (Lück 2016), is immediate.

Corollary 1. *SOQSAT is complete in AEXP(poly).*

4 Refutation Proof System

In the sequel, we assume that an SOQBF is in the prenex normal form with all atomic variables being universally quantified as in Eq. (3), and the matrix is in CNF. As shown previously, an SOQBF can be converted to a model-equivalent QBF by ground instantiation. The conversion ensures the matrix of the resultant QBF remains in CNF as all atomic variables are universally quantified. Then QBF resolution, Q-res, a sound and complete refutation proof system for QBF, can be applied to decide the truth or falsity of a QBF (Kleine Büning, Karpinski, and Flögel 1995). Essentially, Q-res consists of two rules: 1) resolution of two clauses on an existential pivot variable and 2) reduction on universal variables in a clause whose quantification levels are greater than those of any other existential variables in the clause. However, the conversion from an SOQBF to a QBF by ground instantiation mostly suffers an immediate exponential growth of formula size. To alleviate this problem, we apply a lazy instantiation strategy similar to the resolution principle for the first-order logic (FOL) (Robinson 1965), as we detail below.

For a clause $C = \bigvee_i \ell(f_i) \vee \bigvee_j \ell(x_j)$ in the matrix of an SOQBF, where $\ell(v)$ denotes the literal appearance of variable v in C, its function-form is derived by the rule:

$$\frac{\bigvee_i \ell(f_i) \vee \bigvee_j \ell(x_j)}{\bigvee_i \ell(f_i^{\{\neg\ell(x_j) \mid x_j \in \mathsf{S}(f_i)\}})} \text{ Functionize}$$

where literal $\ell(f_i^{\{\neg\ell(x_j) \mid x_j \in \mathsf{S}(f_i)\}})$ inherits the polarity of literal $\ell(f_i)$. That is, the function variables of C are replaced with their respective partially instantiated versions with respect to the literal set (i.e., assignment) $\{\neg\ell(x_j)\}$ whereas all literals of atomic variables are being removed from C. Thereby, the derived set of clauses is in a function-form, namely, having no literals of atomic variables.

Essentially, a literal f^α (resp. $\neg f^\alpha$) asserts the condition that the function value of f under any full assignment β over variables $\mathsf{S}(f)$ consistent with α, i.e., $\beta \wedge \alpha \neq 0$, is true (resp. false). Let Φ' be the SOQBF obtained from Φ such that its matrix consists of clauses functionized from those of Φ and its prefix quantifies the new variables for the quantifier type and quantification level of $f_i^{\{\neg\ell(x_j) \mid x_j \in \mathsf{S}(f_i)\}}$ being the same as those of f_i in Φ. We remark that unlike ground instantiation, the valuations of two partially instantiated function variables may not be independent. E.g., for a function variable f with $\mathsf{S}(f) = \{x_1, x_2\}$, its two instantiated versions f^{x_1} and $f^{\neg x_2}$ are not independent because their valuations have to agree on $f^{x_1 \neg x_2}$. For simplicity, we shall assume that all the partially instantiated versions of variable f_i are bundled together in the prefix and referred to as a *quantifier bundle*. We sometimes write $\mathcal{Q}f_i$, for $\mathcal{Q} \in \{\exists, \forall\}$, in the prefix to represent quantification over all instantiated versions of f_i.

Lemma 1. *An SOQBF Φ and its functionized version Φ' are model-equivalent.*

Proof. Let φ' be the matrix of Φ'. Expanding φ' with $\bigwedge_\alpha \varphi'|_\alpha$ for assignments α over all atomic variables X of Φ (recall that X are universally quantified) results in an effective QBF, which is the same as the QBF obtained through ground instantiation on Φ. □

The following two rules form a refutation proof system SOQ-res for functionized SOQBFs.

$$\frac{C_1 \vee f^{\alpha_1} \qquad C_2 \vee \neg f^{\alpha_2} \qquad \mathsf{qnt}(f) = \exists,\ \alpha_1 \wedge \alpha_2 \neq 0}{\mathrm{DInst}(C_1, \alpha_2, \alpha_1) \vee \mathrm{DInst}(C_2, \alpha_1, \alpha_2)}\ \exists_F\text{-Resolve}$$

$$\frac{C \vee \ell(f^{\alpha}) \qquad \mathsf{qnt}(f) = \forall,\ \mathsf{lev}(f) > \mathsf{lev}(g) \text{ for any } g \text{ in } C,\ \mathsf{qnt}(g) = \exists}{C}\ \forall_F\text{-Reduce}$$

In rule $\exists_F$-resolve, given a clause C and two ordered assignments α and β, the *differential instantiation* $\mathrm{DInst}(C, \alpha, \beta)$ operation produces a new clause C' with

$$\bigvee_{\ell(f_i^{\gamma}) \in C} \ell(f_i^{\gamma \wedge \{\ell(x_j) \in \alpha \setminus \beta \ \mid\ x_j \in \mathsf{S}(f_i), x_j \notin \gamma, \neg x_j \notin \gamma\}}),$$

where $\alpha \setminus \beta$ is in the set notation of assignments, denoting removing β literals from α. Note that the two differential instantiations in the resolvent effectively achieve *unification* in the resolution principle of FOL (Robinson 1965). We remark that the above rule provides a stronger unification than that of DQBF calculus D-IR-calc (Beyersdorff et al. 2016), where the resolvent generated effectively corresponds to $\mathrm{DInst}(C_1, \alpha_1 \wedge \alpha_2, 1) \vee \mathrm{DInst}(C_2, \alpha_1 \wedge \alpha_2, 1)$[1].

Recall the assumption that a clause is non-tautological. We note that $\exists_F$-resolve on two non-tautological clauses may yield a partially tautological clause, e.g., $(C \vee f^{\alpha} \vee \neg f^{\beta})$ for $\alpha \wedge \beta \neq 0$. In such occasions, we shall remove the tautological part of the clause. Before presenting the method for partial-tautology removal, we first generalize the notation f^{α} of function variable f instantiated with respect to an assignment α over a subset of $\mathsf{S}(f)$ to f^{χ}, where χ is a Boolean expression over variables $\mathsf{S}(f)$. A literal f^{χ} (resp. $\neg f^{\chi}$) indicates the condition that f must be true (resp. false) for any assignment α that satisfies χ. By this definition, the equality $f^{\alpha_1 \vee \alpha_2} = f^{\alpha_1} \wedge f^{\alpha_2}$ holds. E.g., for $\mathsf{S}(f) = \{x_1, x_2\}$, the partial instantiation f^{x_1} is equivalent to $f^{x_1 x_2 \vee x_1 \neg x_2} = f^{x_1 x_2} \wedge f^{x_1 \neg x_2}$. Also, the literal $\neg f^{x_1}$, meaning $(\neg f)^{x_1}$, is equivalent to $\neg f^{x_1 x_2 \vee x_1 \neg x_2} = \neg f^{x_1 x_2} \wedge \neg f^{x_1 \neg x_2}$, but not $\neg(f^{x_1 x_2} \wedge f^{x_1 \neg x_2})$. Note that, in special cases, literal f^1 indicates f being a constant-1 function, and literal f^0 equals 1 (a tautology), imposing no constraint on f.

Now the partially tautological clause $(C \vee f^{\alpha} \vee \neg f^{\beta})$ with $\alpha \wedge \beta \neq 0$ can be rewritten as $(C \vee f^{\alpha \wedge (\beta \vee \neg \beta)} \vee \neg f^{(\alpha \vee \neg \alpha) \wedge \beta})$, which in turn can be factored into four clauses $(C \vee f^{\alpha \wedge \beta} \vee \neg f^{\alpha \wedge \beta})$ $(C \vee f^{\alpha \wedge \beta} \vee \neg f^{\neg \alpha \wedge \beta})$ $(C \vee f^{\alpha \wedge \neg \beta} \vee \neg f^{\alpha \wedge \beta})$ $(C \vee f^{\alpha \wedge \neg \beta} \vee \neg f^{\neg \alpha \wedge \beta})$. By removing the first tautological clause, we obtain the non-tautological part of $(C \vee f^{\alpha} \vee \neg f^{\beta})$ as $(C \vee f^{\alpha \wedge \beta} \vee \neg f^{\neg \alpha \wedge \beta})$ $(C \vee f^{\alpha \wedge \neg \beta} \vee \neg f^{\alpha \wedge \beta})$ $(C \vee f^{\alpha \wedge \neg \beta} \vee \neg f^{\neg \alpha \wedge \beta})$. In the sequel, we assume that a partially tautological clause is rewritten with non-tautological ones by the following rule.

$$\frac{(C \vee f^{\alpha} \vee \neg f^{\beta}) \qquad \alpha \wedge \beta \neq 0}{\begin{array}{l}(C \vee f^{\alpha \wedge \beta} \vee \neg f^{\neg \alpha \wedge \beta}) \wedge \\ (C \vee f^{\alpha \wedge \neg \beta} \vee \neg f^{\alpha \wedge \beta}) \wedge \\ (C \vee f^{\alpha \wedge \neg \beta} \vee \neg f^{\neg \alpha \wedge \beta})\end{array}}\ \text{Detautologize}$$

[1]Note that Boolean constant 1 here corresponds to an assignment whose literal set is empty.

Note that the expression deduced by detautologization is not in CNF and has to be further factored into $|\alpha| + |\beta| + |\alpha||\beta|$ clauses.

Note that without removing tautological clauses may result in unsound $\forall_F$-reduce operation. To see the problem, consider the SOQBF $\forall f.(f \vee \neg f)$, which equals $\forall f.1$ and is true. However, without removing the tautological clause $(f \vee \neg f)$, by $\forall_F$-reduction, the matrix reduces to an empty clause, making the SOQBF equal $\forall f.0$, which is false. Therefore, it is crucial to make clauses non-tautological.

For rule $\forall_F$-reduce, in the sequel we shall assume that a clause C is maximally reduced by iterative application of the rule until no more literals can be removed from C.

Lemma 2. *The resolvent produced by the $\exists_F$-resolve rule is logically implied by the conjunction of the two parent clauses.*

Proof. Note that $\mathrm{DInst}(C_1 \vee f^{\alpha_1}, \alpha_2, \alpha_1)$ can be rewritten as $\mathrm{DInst}(C_1, \alpha_2, \alpha_1) \vee f^{\alpha_1 \wedge \alpha_2}$ and thus is implied by $C_1 \vee f^{\alpha_1}$. Similarly, $\mathrm{DInst}(C_2 \vee \neg f^{\alpha_2}, \alpha_1, \alpha_2)$ equals $\mathrm{DInst}(C_2, \alpha_1, \alpha_2) \vee \neg f^{\alpha_1 \wedge \alpha_2}$ and is implied by $C_2 \vee \neg f^{\alpha_2}$. In addition, we know that $(\mathrm{DInst}(C_1, \alpha_2, \alpha_1) \vee f^{\alpha_1 \wedge \alpha_2}) \wedge (\mathrm{DInst}(C_2, \alpha_1, \alpha_2) \vee \neg f^{\alpha_1 \wedge \alpha_2})$ implies $\mathrm{DInst}(C_1, \alpha_2, \alpha_1) \vee \mathrm{DInst}(C_2, \alpha_1, \alpha_2)$. Hence the lemma follows. □

Lemma 3. *A functionized SOQBF with one existential quantification level is false if and only if an empty clause can be derived by a sequence of $\exists_F$-resolve operations.*

Proof. A functionized SOQBF with one existential quantification level can be seen as a special case of Skolem normal form of an FOL formula. As the $\exists_F$-resolve rule is an analogy of the resolution principle of FOL (Robinson 1965), the lemma follows from the completeness of the FOL resolution principle using the semantic-tree argument (Chang and Lee 1973). □

The soundness of $\forall_F$-reduction is stated in the following lemma.

Lemma 4. *An SOQBF Φ with prefix π and matrix $\varphi \wedge C$ (for C non-tautological) is true if and only if the SOQBF $\Phi' = \pi.\varphi \wedge C'$ is true, where C' is a clause $\forall_F$-reduced from C.*

Proof. ($\Rightarrow$) If Φ is true, then we can find a model $\mathcal{M}$ of Skolem functionals for the existential function variables. The model $\mathcal{M}$ makes all the clauses of the matrix $\varphi \wedge C$ be satisfied under all possible function assignments to the universal function variables. As C' is $\forall_F$-reduced from C, the reduced universal function variable does not affect the Skolem functionals to satisfy C' besides φ. Therefore, $\mathcal{M}$ is also a model for Φ'.

($\Leftarrow$) Observe that removing literals from C to get C' makes Φ' no easier to satisfy than Φ. Any model for Φ' is also a model for Φ. □

Theorem 2. *An SOQBF Φ is false if and only if an empty clause can be derived by* SOQ-res.

Proof. The proof can be done in a way similar to that of Q-res for QBF (Kleine Büning, Karpinski, and Flögel 1995). ($\Rightarrow$) We show by induction on k, the number of quantifier bundles. For the base case, when $k = 1$ and $\Phi = \exists F_1.\varphi$ with $F_1 = \{f_1^{\alpha_{1,1}}, \ldots, f_1^{\alpha_{1,n_1}}\}$, the empty clause is derivable according to Lemma 3.

When $k = 1$ and $\Phi = \forall F_1.\varphi$ with $F_1 = \{f_1^{\alpha_{1,1}}, \ldots, f_1^{\alpha_{1,n_1}}\}$, the empty clause is clearly derivable by $\forall_F$-reduce. As the induction hypothesis, assume that the empty clause can be derived for any Φ with $k < m$.

For the induction case, when $k = m$ and $\Phi = \exists F_1, \mathcal{Q}_2 F_2, \ldots, \mathcal{Q}_m F_m.\varphi$ with $F_i = \{f_i^{\alpha_{i,1}}, \ldots, f_i^{\alpha_{i,n_i}}\}$, consider all possible valuations on the function variable f_1, that is, all possible assignments to the $2^{|\mathsf{S}(f_1)|}$ ground instantiated function variables of f_1. The formula of Φ induced under a valuation $[\![f_1]\!]$, i.e., an assignment to the $2^{|\mathsf{S}(f_1)|}$ ground-instantiated function variables of f_1, denoted $\Phi|_{[\![f_1]\!]}$, is the formula $\mathcal{Q}_2 F_2, \ldots, \mathcal{Q}_m F_m.\varphi|_{f_1=[\![f_1]\!]}$, where the cofactor $\varphi|_{f_1=[\![f_1]\!]}$ denotes the formula of φ with every appearance of F_1-literals being valuated (to either 0 or 1) with respect to $[\![f_1]\!]$.[2] By the induction hypothesis, an empty clause is derivable via SOQ-res for the formula induced under every function valuation $[\![f_1]\!]$. Let $\Phi' = \exists F_1, \mathcal{Q}_2 F_2, \ldots, \mathcal{Q}_m F_m.\varphi'$ for $\varphi' = \{C \in \varphi \mid C \text{ not satisfied by } [\![f_1]\!]\}$. Then, either an empty clause or a CNF formula of f_1-instantiated literals falsified by $[\![f_1]\!]$ can be deduced for Φ' via SOQ-res. Note that because $\varphi \to \varphi'$ holds, the formulas deducible from Φ' can also be deduced from Φ. Let ψ be the conjunction of these deduced CNF formulas (possibly including an empty clause) for all valuations of f_1. It is effectively a formula after quantifier elimination of variables $F_2, \ldots, F_m$ from Φ. Because Φ is false, so is ψ. An empty clause is derivable from ψ by $\exists_F$-resolve due to its correspondence to the base case.

When $k = m$ and $\Phi = \forall F_1, \mathcal{Q}_2 F_2, \ldots, \mathcal{Q}_m F_m.\varphi$, following the above reasoning, as ψ is a CNF formula and all its variables are universally quantified, an empty clause can be derived by $\forall_F$-reduce.

Therefore, for a false SOQBF Φ with an arbitrary number of quantifier bundles, an empty clause can be derived by SOQ-res.

($\Leftarrow$) By the soundness of $\exists_F$-resolution and $\forall_F$-reduction as stated in Lemmas 2 and 4, the derivation of an empty clause via SOQ-res ensures the falsity of the SOQBF. □

5 Countermodel Extraction

Similar to the countermodel extraction of Herbrand functions from a Q-res-proof of a QBF (Balabanov and Jiang 2012), it is possible to extract a countermodel of Herbrand functionals from a SOQ-res-proof of an SOQBF. Let $\mathsf{ite}(a, b, c)$ be the if-then-else operation on Boolean expressions a, b, and c, denoting "if a, then b, else c," which equals the expression $ab \vee \neg ac$. We represent a decision list (Rivest 1987) as a sequence $(e_1, c_1), (e_2, c_2), \ldots, (e_k, c_k)$ of expression-constant pairs to mean function

$$\mathsf{ite}(e_1, c_1, \mathsf{ite}(e_2, c_2, \mathsf{ite}(\ldots, \mathsf{ite}(e_k, c_k, \neg c_k)\ldots))).$$

The Herbrand functionals are to be built with decision lists (DLs) in Algorithm 1 for countermodel extraction.

Algorithm 1: Countermodel Extraction

Input: A false SOQBF Φ and an SOQ-res proof DAG $G_\Pi(V_\Pi, E_\Pi)$
Output: A countermodel represented with decision lists

```
1: for all vertex v of G_Π in topological order do
2:   if v.clause is ∀_F-reduced from u.clause, i.e., (u, v) ∈ E_Π then
3:     for all universal ℓ(f^α) in u.clause but not in v.clause do
4:       γ := α;
5:       for all non-empty DL_{f^β} of variable f do
6:         if α ∧ β ≠ 0 then
7:           if ℓ(f^α) = f^α then
8:             let DL_{f^{α∧β}} be DL_{f^β} added by (¬v.clause, 0);
9:           else if ℓ(f^α) = ¬f^α then
10:            let DL_{f^{α∧β}} be DL_{f^β} added by (¬v.clause, 1);
11:          rename DL_{f^β} to DL_{f^{¬α∧β}};
12:          γ := ¬β ∧ γ;
13:       if γ ≠ 0 then
14:         if ℓ(f^α) = f^α then
15:           create DL_{f^γ} with entry (¬v.clause, 0);
16:         else if ℓ(f^α) = ¬f^α then
17:           create DL_{f^γ} with entry (¬v.clause, 1);
18:  if v.clause is an empty clause then
19:    return DLs;
```

The SOQ-res refutation proof of a false SOQBF can be represented as a directed acyclic graph (DAG) $G_\Pi(V_\Pi, E_\Pi)$, where a vertex $v \in V_\Pi$ represents a clause $v.clause$ and an edge $(u, v) \in E_\Pi$ signifies $u.clause$ is derived from $v.clause$ by either $\exists_F$-resove or $\forall_F$-reduce. Given a false SOQBF and its SOQ-res refutation DAG, Algorithm 1 computes a countermodel consisting of Herbrand functionals for the universally quantified function variables. It traverses G_Π in a topological order until a vertex of empty clause is reached (Line 1). For a vertex v with $v.clause$ being $\forall_F$-reduced from $u.clause$ (Line 2), it examines the $\forall_F$-reduced literals from $u.clause$ (Line 3). For each such literal, say $\ell(f^\alpha)$, it updates all DLs of instantiated function variables f^β of f for $\alpha \wedge \beta \neq 0$ (Lines 5–11). Specifically, the DL of $f^{\alpha\wedge\beta}$ is created to be the same as f^β but appended with the pair $(\neg v.clause, 0)$ if $\ell(f^\alpha)$ is of positive polarity (Lines 7, 8) and $(\neg v.clause, 1)$ otherwise (Lines 9, 10). Also, the DL of f^β becomes that of $f^{\neg\alpha\wedge\beta}$ (Line 11). If α is not completely covered by the β assignments (Line 13), a new DL is created for the remaining assignments of α with entry $(\neg v.clause, 0)$ if $\ell(f^\alpha)$ is of positive polarity (Lines 14, 15) and $(\neg v.clause, 1)$ otherwise (Lines 16, 17).

[2] Note that literal f_1^α (resp. $\neg f_1^\alpha$) valuates to 1 under $[\![f_1]\!]$ if and only if all ground instantiated variables $\{f_1^\beta \mid \alpha \wedge \beta \neq 0\}$ are assigned 1 (resp. 0) in $[\![f_1]\!]$.

The DLs are returned when a vertex of empty clause is reached (Lines 18, 19).

Note that any two distinct DLs, say, $\mathtt{DL}_{f^{\alpha_1}}$ and $\mathtt{DL}_{f^{\alpha_2}}$, computed by Algorithm 1 satisfy the orthogonality that the instantiated assignments are mutually independent, i.e., $\alpha_1 \wedge \alpha_2 = 0$. To maintain this orthogonality, the number of DLs may grow exponentially in the number of atomic variables.

Note also that a Herbrand functional returned by the algorithm may refer not only to existential function variables but also to universal function variables. To eliminate the dependencies on universal function variables, unwanted universal function variables can be substituted with their corresponding Herbrand functionals. By repeated substitutions in ascending order of quantification levels, all Herbrand functionals can be made to depend only on existential function variables. In the substitutions, an instantiated universal function variable may not have its corresponding DL returned by Algorithm 1. However, the Herbrand functional of an arbitrary instantiated function variable f^{α} can be obtained by forming the conjunction of the Herbrand functionals of variables f^{β}, for $\beta \wedge \alpha \neq 0$, which are derivable from the DLs returned from Algorithm 1.

The correctness of Algorithm 1 is stated as follows.

Theorem 3. *Given a false SOQBF $\Phi = \pi.\varphi$ and a refutation proof, let the instantiated universal functional variables in φ be substituted with the Herbrand functionals extracted by Algorithm 1. Then φ after the substitution is unsatisfiable.*

Proof. Observe that when the instantiated assignments of every universal function variable appearing in the resolution proof are all orthogonal, the constructed DLs form a countermodel, as can be shown in the same way as that of the QBF case (Balabanov and Jiang 2012). Furthermore, it can be verified that Algorithm 1 maintains the orthogonality of the partially instantiated universal function variables in constructing the DLs. □

Example 2. *Consider the SOQBF*

$$\forall g_1, \exists f_1, \forall g_2, \exists f_2, \forall x_1, \forall x_2, \forall x_3. C_1 C_2 C_3,$$

where the support sets $\mathsf{S}(g_1) = \{x_1\}$, $\mathsf{S}(f_1) = \{x_2, x_3\}$, $\mathsf{S}(g_2) = \{x_2, x_3\}$, $\mathsf{S}(f_2) = \{x_1, x_3\}$, *and*

$$\begin{aligned} C_1 &= (g_1^{x_1} \vee f_1^{x_2 \neg x_3} \vee f_2^{x_1 \neg x_3}), \\ C_2 &= (g_2 \vee \neg f_2), \\ C_3 &= (\neg g_1^{\neg x_1} \vee \neg f_1^{x_2} \vee \neg g_2^{x_2} \vee f_2^{\neg x_1}). \end{aligned}$$

Its falsity can be shown by the following refutation steps.

$C_4 = (g_1^{x_1} \vee f_1^{x_2 \neg x_3} \vee g_2^{\neg x_3})$ *by* $\exists_F$*-resolve*(C_1, C_2);
$C_5 = (g_1^{x_1} \vee f_1^{x_2 \neg x_3})$ *by* $\forall_F$*-reduce*(C_4);
$C_6 = (g_2 \vee \neg g_1^{\neg x_1} \vee \neg f_1^{x_2} \vee \neg g_2^{x_2})$ *by* $\exists_F$*-resolve*(C_2, C_3);
$C_6' = (g_2^{\neg x_2} \vee \neg g_1^{\neg x_1} \vee \neg f_1^{x_2} \vee \neg g_2^{x_2})$ *by tautology removal*;
$C_7 = (\neg g_1^{\neg x_1} \vee \neg f_1^{x_2})$ *by* $\forall_F$*-reduce*(C_6');
$C_8 = (g_1^{x_1} \vee \neg g_1^{\neg x_1})$ *by* $\exists_F$*-resolve*(C_5, C_7);
$C_9 = ()$ *by* $\forall_F$*-reduce*(C_8).

Given the proof, Algorithm 1 derives the DLs:

$$\begin{aligned} \mathtt{DL}_{g_2^{\neg x_2 \neg x_3}} &= \mathsf{ite}(\neg g_1^{x_1} \neg f_1^{x_2 \neg x_3}, 0, \mathsf{ite}(g_1^{\neg x_1} f_1^{x_2}, 0, 1)), \\ \mathtt{DL}_{g_2^{\neg x_2 x_3}} &= \mathsf{ite}(g_1^{\neg x_1} f_1^{x_2}, 0, 1), \\ \mathtt{DL}_{g_2^{x_2 \neg x_3}} &= \mathsf{ite}(\neg g_1^{x_1} \neg f_1^{x_2 \neg x_3}, 0, \mathsf{ite}(g_1^{\neg x_1} f_1^{x_2}, 1, 0)), \\ \mathtt{DL}_{g_2^{x_2 x_3}} &= \mathsf{ite}(g_1^{\neg x_1} f_1^{x_2}, 1, 0), \\ \mathtt{DL}_{g_1^{x_1}} &= \mathsf{ite}(1, 0, 1) \\ \mathtt{DL}_{g_1^{\neg x_1}} &= \mathsf{ite}(1, 1, 0). \end{aligned}$$

They yield the following Herbrand functionals:

$$\begin{aligned} \mathcal{H}[g_1^{x_1}] &= 0, \\ \mathcal{H}[g_1^{\neg x_1}] &= 1, \\ \mathcal{H}[g_2^{x_2 x_3}] &= f_1^{x_2}, \\ \mathcal{H}[g_2^{x_2 \neg x_3}] &= f_1^{x_2 \neg x_3} f_1^{x_2} = f_1^{x_2}, \\ \mathcal{H}[g_2^{\neg x_2 x_3}] &= \neg f_1^{x_2}, \\ \mathcal{H}[g_2^{\neg x_2 \neg x_3}] &= f_1^{x_2 \neg x_3} \neg f_1^{x_2} = 0. \end{aligned}$$

By combining and substituting the Herbrand functionals to their corresponding function variables in the matrix, we get

$$(f_1^{x_2 \neg x_3} \vee f_2^{x_1 \neg x_3})(\neg f_2)(\neg f_1^{x_2} \vee f_2^{\neg x_1}),$$

which can deduce an empty clause via $\exists_F$*-resolve.*

6 Applications

We briefly mention two potential applications of SOQBF, one in system synthesis and the other in AI planning.

For the first application, consider the system block diagram shown in Fig. 2, where an unknown component F (with inputs Y) is to be synthesized within a known context circuit C (with inputs X), which is to be composed with some third-party design G (with inputs Z). We are asked to design F such that the entire composite system satisfies some desired safety property P regardless of any function G for composition. For simplicity, assume the systems are combinational, i.e., memoryless. Then the synthesis problem can be expressed in an SOQBF of the form

$$\exists F, \forall G, \exists H, \forall X, Y, Z, W. \varphi,$$

where function variables F, G, H are with support sets $\mathsf{S}(F) = Y, \mathsf{S}(G) = Z, \mathsf{S}(H) = X \cup Y \cup Z \cup W$, for W being extra atomic variables for CNF encoding of the matrix constraint φ and H being extra function variables for normal form conversion.

For the second application, consider a planning problem of two opposing agents A_1 and A_2. Assume the two agents take actions by turns sequentially. We are interested in knowing whether, under a bounded planning horizon, for any action strategy of A_1, agent A_2 has a strategy to enforce the environment transition from the initial state to a goal state. Assume for simplicity that the planning horizon is one. Then the planning problem can be expressed in an SOQBF of the form

$$\forall F_1, \exists F_2, \exists E, \forall S, S', S'', X.$$
$$(I(S) \wedge T_1(S, S', F_1) \wedge T_2(S', S'', F_2) \rightarrow G(S'')) \wedge \psi,$$

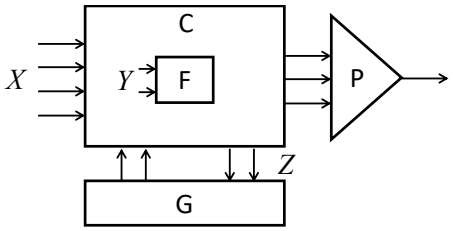


Figure 2: Circuit synthesis for compositional safety.

where function variables F_1 with $\mathsf{S}(F_1) = S$ and F_2 with $\mathsf{S}(F_2) = S'$ correspond to the state updates due to the action choices of A_1 and A_2, respectively; function variables E are extra ones for normal form conversion; atomic variables S, S', and S'' are the state variables of the environment in the beginning, after A_1 taking action, and after A_2 taking action, respectively; atomic variables X are extra ones for CNF encoding; predicates I, T_1, T_2, and G correspond to formulas that constrain the initial states, state-action relation of A_1, state-action relation of A_2, and goal states, respectively; quantifier-free formula ψ is induced from normal form conversion.

7 Conclusions and Future Work

This paper studied the second-order quantified Boolean logic from the aspects of representation and interpretation, complexity, refutation proof system, and countermodel extraction. Specifically, we established the connection between SOQBF and QBF. We extended the FOL resolution principle and QBF resolution calculus to a sound and complete refutation proof system for SOQBF, and extended the countermodel extraction algorithm of QBF to SOQBF. Potential applications of SOQBF were also discussed. For future work, we plan to identify more applications. In particular, checking memory consistency models (Cooksey et al. 2019) could be an interesting direction. Also, we intend to develop SOQBF solvers for potential practical applications.

Acknowledgements

The author is grateful to Christoph Scholl and Tony Tan for their valuable comments on the manuscript. This work was partly supported by the National Science and Technology Council of Taiwan under Grant 111-2221-E-002-182 and Grant 111-2923-E-002-013-MY3.

References

Ackermann, W. 1954. *Solvable cases of the decision problem*. North-Holland Publishing Company.

Balabanov, V.; Chiang, H.-J. K.; and Jiang, J.-H. R. 2014. Henkin quantifiers and Boolean formulae: A certification perspective of DQBF. *Theoretical Computer Science*, 523: 86–100.

Balabanov, V.; and Jiang, J.-H. R. 2012. Unified QBF Certification and Its Applications. *Formal Methods in System Design*, 41: 45–65.

Beyersdorff, O.; Chew, L.; Schmidt, R. A.; and Suda, M. 2016. Lifting QBF Resolution Calculi to DQBF. In *Proceedings of International Conference on Theory and Applications of Satisfiability Testing*, 490–499.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2021. *Handbook of Satisfiability*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

Chandra, A. K.; Kozen, D. C.; and Stockmeyer, L. J. 1981. Alternation. *J. ACM*, 28(1): 114–133.

Chang, C.-L.; and Lee, R. C.-T. 1973. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press.

Cook, S. A. 1971. The Complexity of Theorem-Proving Procedures. In *Proceedings of ACM Symposium on Theory of Computing*, 151–158.

Cooksey, S.; Harris, S.; Batty, M.; Grigore, R.; and Janota, M. 2019. PrideMM: Second Order Model Checking for Memory Consistency Models. In *Formal Methods. FM 2019 International Workshops*, 507–525.

Dawar, A.; Gottlob, G.; and Hella, L. 1998. Capturing Relativized Complexity Classes without Order. *Mathematical Logic Quarterly*, 44(1): 109–122.

Hannula, M.; Kontinen, J.; Lück, M.; and Virtema, J. 2016. On quantified propositional logics and the exponential time hierarchy. In *Proceedings of International Symposium on Games, Automata, Logics and Formal Verification*, 198–212.

Hannula, M.; Kontinen, J.; Lück, M.; and Virtema, J. 2020. On the Complexity of Horn and Krom Fragments of Second-Order Boolean Logic. *CoRR*, abs/2007.03867.

Henkin, L. 1961. Some Remarks on Infinitely Long Formulas. In *Journal of Symbolic Logic*, 167–183.

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

Lohrey, M. 2012. Model-checking hierarchical structures. *Journal of Computer and System Sciences*, 78(2): 461–490.

Lück, M. 2016. Complete Problems of Propositional Logic for the Exponential Hierarchy. *CoRR*, abs/1602.03050.

Peterson, G. L.; and Reif, J. H. 1979. Multiple-person alternation. In *Proceedings of IEEE Symposium on Foundations of Computer Science*, 348–363.

Rivest, R. L. 1987. Learning decision lists. *Machine Learning*, 2(3): 229–246.

Robinson, J. A. 1965. A Machine-Oriented Logic Based on the Resolution Principle. *J. ACM*, 12(1): 23–41.

Scholl, C.; and Wimmer, R. 2018. Dependency Quantified Boolean Formulas: An Overview of Solution Methods and Applications - Extended Abstract. In *Proceedings of International Conference on Theory and Applications of Satisfiability Testing*, 3–16.

Stockmeyer, L. J. 1976. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1): 1–22.

Stockmeyer, L. J.; and Meyer, A. R. 1973. Word Problems Requiring Exponential Time(Preliminary Report). In *Proceedings of the ACM Symposium on Theory of Computing*, 1–9.