# Skolem Functions for DQBF*

Karina Wimmer[1], Ralf Wimmer[1,2], Christoph Scholl[1], and Bernd Becker[1]

[1] Albert-Ludwigs-Universität Freiburg im Breisgau, Germany
{wimmerka,wimmer,scholl,becker}@informatik.uni-freiburg.de
[2] Dependable Systems and Software, Saarland University, Saarbrücken, Germany

**Abstract.** We consider the problem of computing Skolem functions for satisfied dependency quantified Boolean formulas (DQBFs). We show how Skolem functions can be obtained from an elimination-based DQBF solver and how to take preprocessing steps into account. The size of the Skolem functions is optimized by don't-care minimization using Craig interpolants and rewriting techniques. Experiments with our DQBF solver HQS show that we are able to effectively compute Skolem functions with very little overhead compared to the mere solution of the formula.

## 1 Introduction

Solver-based techniques have proven successful in many areas, ranging from formal verification of hard- and software systems [6,1] over automatic test pattern generation [14,12] to planning [36]. While research on solving quantifier-free Boolean formulas (the famous SAT-problem [10]) has reached a certain level of maturity, designing and improving algorithms for quantified Boolean formulas (QBFs) is in the focus of active research. However, there are applications like the verification of partial circuits [37,18], the synthesis of safe controllers [7], and the analysis of games with incomplete information [32] for which QBF is not expressive enough to provide a compact and natural formulation. The reason is that QBF requires linearly ordered dependencies of the existential variables on the universal ones: Each existential variable implicitly depends on all universal variables in whose scope it is. Relaxing this condition yields so-called *dependency quantified Boolean formulas (DQBFs)*. DQBFs are strictly more expressive than QBFs in the sense that an equivalent QBF formulation can be exponentially larger than a DQBF formulation. This comes at the price of a higher complexity of the decision problem: DQBF is NEXPTIME-complete [32], compared to QBF, which is "only" PSPACE-complete. Encouraged by the success of SAT and QBF solvers and driven by the mentioned applications, research on solving DQBFs has started during the last few years [16,17,19,41], yielding first prototypic solvers like iDQ [17] and HQS [19].

All currently available DQBF solvers are restricted to a pure yes/no answer regarding the satisfiability of the formula, allowing to decide whether an incomplete circuit is realizable, whether a controller with certain properties can be

* This work was partly supported by the German Research Council (DFG) as part of the project "Solving Dependency Quantified Boolean Formulas" and by the Sino-German Center for Research Promotion as part of the project CAP (GZ 1023).

synthesized, and whether a player has a winning strategy in a game. But typically a pure yes/no answer is not satisfactory: In case a circuit is realizable, one wants to have an implementation; if a controller is synthesizable, one wants to get a realization of it; and in a game, where a player has a winning strategy, one wants to know such a winning strategy. These implementations, realizations, and strategies all correspond to so-called *Skolem functions* for the existential variables in a DQBF. While for different paradigms to solve QBFs, Skolem functions can be computed (see below for an overview of related work), we are not aware of any paper that considers the computation of Skolem functions for DQBF.

So, this is the first paper that shows how Skolem functions can be obtained from elimination-based QBF or DQBF solvers like AIGsolve [33,34] or HQS [19]. We do not only take into account the core operations for eliminating variables [19], but also the preprocessing steps [41], which are essential for an efficient solution of the formula. We propose to apply don't-care minimization to reduce the representation size of the computed Skolem functions. We have implemented the described techniques in our DQBF solver HQS; preliminary experiments show not only that we have found a feasible approach to Skolem function computation, but also that the overhead during the solution of the formulas is small.

Due to space restrictions we are only able to give short proof sketches for the main theorems. Detailed proofs are available in a technical report [40].

*Related work.* Computing Skolem functions has not been studied for DQBF so far. Therefore we concentrate on related work in QBF solving.

sKizzo [5] and Squolem [25] are QBF solvers which are based on Skolemization: The existential variables are replaced by an encoding of the Skolem functions' unknown truth tables. In case of sKizzo, the entries of the truth tables are variables, resulting in an (exponentially larger) SAT problem. This SAT problem is represented compactly using OBDDs [39] and solved by an adapted SAT solver. A satisfying assignment corresponds to Skolem functions for the QBF. Squolem is based on eliminating variables $v$ in the QBF prefix from right to left by considering clauses containing $v$ which describe the function table of $v$'s Skolem function.

Balabanov and Jiang [3] and Goultiaeva et al. [20] laid the foundations for extracting Skolem functions from SAT/UNSAT proofs for QBFs in form of term/clause resolution trees. Such proofs can be obtained from search-based QBF solvers like DepQBF [29,31]. However, this approach is not applicable to DQBF: resolution is – in contrast to QBF – not a complete decision procedure for DQBF [2]; in general it is not possible to decide a DQBF using resolution.

Heule et al. [23] consider the extraction of Skolem functions when preprocessing is applied before the actual solving process. They represent the different preprocessing steps in a unified framework, called QRAT. Such QRAT logs can be used to derive Skolem functions for the original formula.

CAQE [38] is a very recent QBF solver, which is based on decomposing the QBF into a sequence of simpler propositional formulas. CAQE also supports the computation of Skolem functions.

*Structure of this paper.* In the next section, we introduce the necessary foundations on DQBFs, Skolem functions, and don't-care minimization of Boolean functions.

In Section 3 we consider the main elimination operations, which are used in the solver core of HQS. The following section shows how preprocessing steps can be taken into account. We present experimental results in Section 5 and conclude the paper in Section 6.

## 2 Foundations

The Boolean values are denoted by $\mathbb{B} = \{0, 1\}$. For a set $V$ of Boolean variables, the set of all variable assignments of $V$ is $\mathbf{A}(V) = \{\nu : V \to \mathbb{B}\}$. We extend variable assignments $\nu \in \mathbf{A}(V)$ to quantifier-free Boolean formulas $\phi$: $\nu(\phi)$ is the value obtained by replacing all variables $v$ occurring in $\phi$ with their value $\nu(v)$ and applying the usual rules of Boolean algebra.

A literal $\ell$ is either a Boolean variable $v \in V$ or its negation $\neg v$. The sign of a literal is given by $\mathrm{sign}(v) = 1$ and $\mathrm{sign}(\neg v) = 0$ for $v \in V$. A clause is a disjunction of literals, and a formula is in conjunctive normal form (CNF) if it is a conjunction of (non-tautological) clauses. We often identify a clause with its set of literals, and a CNF with its set of clauses. For quantifier-free Boolean formulas $\phi$ and $\psi$ over variables $V$ and a variable $v \in V$, the notation $\phi[\psi/v]$ denotes the formula which results from replacing all occurrences of $v$ in $\phi$ simultaneously by $\psi$. $\mathrm{var}(\phi)$ is the set of variables occurring in $\phi$. We treat $\mathrm{var}(\phi)$ as a variable if it is a singleton. We sometimes identify $\phi$ with its represented function $f_\phi$: for $\nu \in \mathbf{A}(V)$, we set $f_\phi(\nu) := \nu(\phi)$. A formula $\phi$ is a representation of a Boolean function $g$ iff $g = f_\phi$. Each Boolean function can be represented as a formula. By ITE we denote the if-then-else function, i. e., $\mathrm{ITE}(a, b, c) = (a \wedge b) \vee (\neg a \wedge c)$.

### 2.1 Dependency quantified Boolean formulas

Dependency quantified Boolean formulas are obtained by prefixing quantifier-free Boolean formulas with so-called Henkin quantifiers [21].

**Definition 1 (Syntax of DQBF).** *Let $V = \{x_1, \ldots, x_n, y_1, \ldots, y_m\}$ be a finite set of Boolean variables. A* dependency quantified Boolean formula (DQBF) *$\Psi$ over $V$ has the form $\Psi := \forall x_1 \ldots \forall x_n \exists y_1(D_{y_1}) \ldots \exists y_m(D_{y_m}) : \phi$, where $D_{y_i} \subseteq \{x_1, \ldots, x_n\}$ is the* dependency set *of $y_i$ for $i = 1, \ldots, m$, and $\phi$ is a quantifier-free Boolean formula over $V$, called the* matrix *of $\Psi$.*

$U_\Psi = \{x_1, \ldots, x_n\}$ is the set of universal and $E_\Psi = \{y_1, \ldots, y_m\}$ the set of existential variables. A literal $\ell$ is existential (universal, resp.) iff $\mathrm{var}(\ell) \in E_\Psi$ ($\mathrm{var}(\ell) \in U_\Psi$). Sometimes we assume that $\phi$ is given in CNF.

A QBF (in prenex normal form) is a DQBF such that $D_y \subseteq D_{y'}$ or $D_{y'} \subseteq D_y$ holds for any two existential variables $y, y' \in E_\Psi$.

To simplify notation, we define a dependency function $\mathrm{dep}_\Psi : V \to 2^{U_\Psi}$ as follows: $\mathrm{dep}_\Psi(v) = \{v\}$ if $v$ is universal and $\mathrm{dep}_\Psi(v) = D_v$ if $v$ is existential.

The semantics of a DQBF is typically defined by so-called Skolem functions.

**Definition 2 (Semantics of DQBF).** *Let $\Psi$ be a DQBF as above. It is* satisfiable *if there are functions $s_y : \mathbf{A}(D_y) \to \mathbb{B}$ for $y \in E_\Psi$ such that replacing each $y \in E_\Psi$ by (a Boolean expression representing) $s_y$ turns $\phi$ into a tautology. Such functions $(s_y)_{y \in E_\Psi}$ are called* Skolem functions *for $\Psi$.*

Deciding whether a given DQBF is satisfiable is NEXPTIME-complete [32].

**Definition 3 (Equisatisfiability, equivalence of DQBFs).** *Let $\Psi_i = Q_i : \phi_i$ for $i = 1, 2$ be two DQBFs over variables $V$. $\Psi_1$ and $\Psi_2$ are* equisatisfiable *($\Psi_1 \triangleq \Psi_2$), if $\Psi_1$ is satisfiable iff $\Psi_2$ is. $\Psi_1$ and $\Psi_2$ are* logically equivalent *($\Psi_1 \equiv \Psi_2$) if $Q_1 = Q_2$ and $\nu(\phi_1) = \nu(\phi_2)$ for all $\nu \in \mathbf{A}(V)$.*

Logically equivalent formulas that are satisfiable have the same Skolem functions.

The main operations used by elimination-based solvers like HQS [19] to solve DQBFs are variants of variable elimination. For standard Boolean logic, elimination of variables can be performed in different ways, resulting in logically equivalent formulas of typically different sizes and structures:

**Lemma 1 ([24]).** *Let $\phi$ be a Boolean formula and $x$ a variable of $\phi$. We have:*

$$\exists x : \phi \quad \triangleq \quad \phi[0/x] \vee \phi[1/x] \quad \equiv \quad \phi\Big[\phi[1/x]/x\Big] \quad \equiv \quad \phi\Big[\neg\phi[0/x]/x\Big] .$$

This lemma will be used later to obtain formulas for the Skolem functions of existential variables in DQBFs. As we will see later, don't-care minimization can be applied to these Skolem functions to obtain some with a small representation.

### 2.2 Don't-care minimization of Boolean functions

**Definition 4 (Incompletely specified Boolean function).** *Let $V$ be a set of Boolean variables. An* incompletely specified Boolean function *$f$ is given by a don't-care set $\mathrm{DC}(f) \subseteq \mathbf{A}(V)$ and an on-set $\mathrm{ON}(f) \subseteq \mathbf{A}(V)$ such that $\mathrm{DC}(f) \cap \mathrm{ON}(f) = \emptyset$. We additionally define the off-set $\mathrm{OFF}(f) := \mathbf{A}(V) \setminus (\mathrm{DC}(f) \cup \mathrm{ON}(f))$.*

Of course it suffices to specify any two sets of $\mathrm{ON}(f)$, $\mathrm{OFF}(f)$, and $\mathrm{DC}(f)$.

**Definition 5 (Complete extension).** *Let $f$ be an incompletely specified Boolean function. A function $f^* : \mathbf{A}(V) \to \mathbb{B}$ is a* complete extension *of $f$ iff $f^*(\nu) = 1$ for all $\nu \in \mathrm{ON}(f)$ and $f^*(\nu) = 0$ for all $\nu \in \mathrm{OFF}(f)$.*

The goal of don't-care minimization is: Given an incompletely specified Boolean function $f$, find a complete extension $f^*$ of $f$ with a small representation by a circuit or an and-inverter graph (AIG) [27]. This can be done, e. g., by using Craig interpolants.

**Definition 6 (Craig interpolant, [11]).** *Let $\phi = \phi_A \wedge \phi_B$ be a (quantifier-free) Boolean formula that is unsatisfiable. A* Craig interpolant *for $(\phi_A, \phi_B)$ is a Boolean formula $\phi_I$ such that: (a) $\phi_I$ contains only variables which appear in both $\phi_A$ and $\phi_B$, (b) $\phi_A \Rightarrow \phi_I$ is a tautology, and (c) $\phi_I \wedge \phi_B$ is unsatisfiable.*

**Lemma 2.** *Let $f$ be an incompletely specified function over $V$ and $\varphi_{\mathrm{ON}(f)}$, $\varphi_{\mathrm{OFF}(f)}$ be Boolean formulas for $\mathrm{ON}(f)$ and $\mathrm{OFF}(f)$, respectively, i. e., for every assignment $\nu \in \mathbf{A}(V)$, we have $\nu(\varphi_{\mathrm{OFF}(f)}) = 1$ iff $\nu \in \mathrm{OFF}(F)$ and $\nu(\varphi_{\mathrm{ON}(f)}) = 1$ iff $\nu \in \mathrm{ON}(f)$.*

*Then every Craig interpolant for $(\varphi_{\mathrm{ON}(f)}, \varphi_{\mathrm{OFF}(f)})$ represents a complete extension of $f$.*

This lemma will be exploited for don't-care minimization when eliminating existential variables.

**Lemma 3 ([24]).** *Let $\phi$ be a Boolean formula and $x$ a variable in $\phi$. Then each Craig interpolant $\phi_I$ w. r. t. $\phi_A := \neg\phi[0/x] \wedge \phi[1/x]$ and $\phi_B := \neg\phi[1/x] \wedge \phi[0/x]$ satisfies $\exists x : \phi \triangleq \phi[\phi_I/x]$.*

Craig interpolants can be derived from a resolution tree which shows the unsatisfiability of the formula [35]. They find numerous applications in system design, see e. g., [30].

## 3 Undoing elimination steps

In the following we assume that a DQBF of the form:

$$\Psi^0 = \forall x_1 \ldots \forall x_n \exists y_1(D^0_{y_1}) \ldots \exists y_m(D^0_{y_m}) : \phi^0$$

is given with dependency sets $D^0_{y_i} \subseteq \{x_1, \ldots, x_n\}$ for $i = 1, \ldots, m$. We abbreviate the quantifier prefix by $Q^0$ and write $\Psi^0 = Q^0 : \phi^0$.

DQBF preprocessors and elimination-based DQBF solvers execute a sequence of transformation steps on the formula until a pure SAT problem is obtained. Thereby we obtain a sequence of equisatisfiable formulas $\Psi^i = Q^i : \phi^i$ for $i = 1, \ldots, k^*$ such that $\Psi^i$ results from $\Psi^{i-1}$ by applying one transformation step and $\Psi^{k^*}$ is an existential formula that can be solved using a SAT solver.

For $\Psi^{k^*}$, Skolem functions are simply given by a satisfying assignment. The main idea of the paper is to show how Skolem functions for $\Psi^{i-1}$ can be derived from Skolem functions for $\Psi^i$, finally resulting in Skolem functions for the original formula $\Psi^0$.

In the following, we consider the quantifier prefix as a set of tuples formed by quantifiers, variables, and – for existential variables – their dependency sets. The set of universal variables in $\Psi^i$ is denoted by $U^i$ and the set of existential variables by $E^i$. For a variable $y \in E^i$, $D^i_y$ is its dependency set in $\Psi^i$, i. e.,

$$Q^i = \{\forall x \,|\, x \in U^i\} \cup \{\exists y(D^i_y) \,|\, y \in E^i\}.$$

### 3.1 Universal expansion

Universal expansion [9,18] eliminates a universal variable $x^*$ from a DQBF. If any existential variables depend upon $x^*$, they have to be copied to allow them taking different values for $x^* = 0$ and $x^* = 1$.

**Lemma 4 (Universal expansion).** *Let $(s^k_y)_{y \in E^k}$ be Skolem functions for the existential variables in $\Psi^k$ and assume that $\Psi^k$ was obtained from $\Psi^{k-1}$ by expanding the universal variable $x^* \in U^{k-1}$ such that, for $y \in E^{k-1}$ with $x^* \in D^{k-1}_y$, $y'$ is the copy of $y$ appearing in the 1-cofactor w. r. t. $x$. In detail:*

$$\begin{aligned} Q^k : \phi^k = \Big(Q^{k-1} \setminus \big(\{\forall x^*\} \cup \{\exists y(D^{k-1}_y) \,|\, y \in E^{k-1} \wedge x^* \in D^{k-1}_y\}\big)\Big) \\ \cup \big\{\exists y(D^{k-1}_y \setminus \{x^*\}), \exists y'(D^{k-1}_y \setminus \{x^*\}) \,\big|\, y \in E^{k-1} \wedge x^* \in D^{k-1}_y\big\} : \\ \big(\phi^{k-1}[0/x^*] \wedge \phi^{k-1}[1/x^*][y'/y \text{ for all } y \in E^{k-1} \text{ with } x^* \in D^{k-1}_y]\big). \end{aligned}$$

*Then* $(s_y^{k-1})_{y\in E^{k-1}}$ *with* $s_y^{k-1} = s_y^k$ *if* $x^* \notin D_y^{k-1}$, *and* $s_y^{k-1} = \mathrm{ITE}(x^*, s_{y'}^k, s_y^k)$ *if* $x^* \in D_y^{k-1}$ *are Skolem functions for the existential variables in* $\Psi^{k-1}$.

*Proof sketch.* We replace the existential variables with their Skolem functions and show that the resulting formula, which only contains universal variables, is a tautology for both $x^* = 0$ and $x^* = 1$. To do so, one can exploit the fact that $(s_y^k)_{y\in E^k}$ are Skolem functions for the formula after elimination. For a detailed proof see [40]. □

### 3.2 Elimination of existential variables

Elimination of existential variables is done like in QBF. It is applicable for variables which depend upon all universal variables [19].

**Lemma 5.** *Let* $(s_y^k)_{y\in E^k}$ *be Skolem functions for the existential variables in* $\Psi^k$ *and assume that* $\Psi^k$ *was obtained from* $\Psi^{k-1}$ *by eliminating the existential variable* $y^* \in E^{k-1}$ *(which requires* $D_{y^*}^{k-1} = U^{k-1}$*). In detail:*

$$Q^k : \phi^k = Q^{k-1} \setminus \left\{\exists y^*(D_{y^*}^{k-1})\right\} : \left(\phi^{k-1}[0/y^*] \vee \phi^{k-1}[1/y^*]\right)$$

*Then* $(s_y^{k-1})_{y\in E^{k-1}}$ *with* $s_y^{k-1} = s_y^k$ *if* $y \neq y^*$, *and* $s_{y^*}^{k-1} = \phi^{k-1}\big[1/y^*\big]\big[s_z^k/z \text{ for } z \in E^k\big]$ *are Skolem functions for the existential variables in* $\Psi^{k-1}$.

*Proof sketch.* We replace the existential variables by their Skolem functions and show that the resulting formula, which contains only universal variables, is a tautology. For this we assume an arbitrary assignment $\nu$ of the universal variables and distinguish the cases where $\nu(s_{y^*}^{k-1}) = 0$ and where $\nu(s_{y^*}^{k-1}) = 1$. In both cases simple equivalence transformations show that $\nu$ satisfies the formula. □

*Remark 1 (Blockwise elimination).* For improving efficiency of variable elimination, typically sets of variables are eliminated en bloc without creating intermediate results. For existential variable sets, these intermediate results, however, are required for Skolem function computation. A possible way to deal with this is to redo the quantification variable by variable if in the end the formula is satisfied and Skolem functions are to be computed.

*Remark 2 (Alternative Skolem function).* As a Skolem function for $y^*$, we could also use $s_{y^*}^{k-1} := \neg\phi^{k-1}\big[0/y^*\big]\big[s_y^k/y \text{ for } y \in E^k\big]$. The proof is analogous to the proof of Lemma 5.

*Remark 3 (Don't-care minimization of Skolem functions).* Any complete extension of the following incompletely specified Boolean function can be used as a Skolem function for $y^*$ in Lemma 5:

$$\mathrm{ON}(s_{y^*}^{k-1}) = \Big(\ \ \phi^{k-1}\big[1/y^*\big] \wedge \neg\phi^{k-1}\big[0/y^*\big]\Big)\big[s_y^k/y \text{ for } y \in E^k\big],$$
$$\mathrm{OFF}(s_{y^*}^{k-1}) = \Big(\neg\phi^{k-1}\big[1/y^*\big] \wedge \ \ \phi^{k-1}\big[0/y^*\big]\Big)\big[s_y^k/y \text{ for } y \in E^k\big],$$
$$\mathrm{DC}(s_{y^*}^{k-1}) = \Big(\ \ \phi^{k-1}\big[1/y^*\big] \wedge \ \ \phi^{k-1}\big[0/y^*\big]\Big)\big[s_y^k/y \text{ for } y \in E^k\big].$$

The don't-care set $\mathrm{DC}(s_{y^*}^{k-1})$ can be exploited to minimize the size of $s_{y^*}^{k-1}$'s representation, e. g., by using Craig interpolation, cf. Lemma 3.

*Remark 4 (Skolem functions for SAT problems).* If the result of the elimination process is a pure SAT problem with only existential quantifiers, we can solve it using a SAT solver. In case the formula is satisfiable, any satisfying assignment corresponds to (constant) Skolem functions for the existential variables.

*Example 1.* Consider the DQBF $\Psi^0 = \forall x_1 \forall x_2 \exists y_1(x_1) \exists y_2(x_2) : \phi^0(x_1, x_2, y_1, y_2)$. Elimination yields the following sequence of matrices:

$$\phi^0(x_1, x_2, y_1, y_2) \xrightarrow{\forall x_1} \phi^1(x_2, y_1, y_1', y_2) \xrightarrow{\exists y_2} \phi^2(x_2, y_1, y_1') \xrightarrow{\forall x_2} \phi^3(y_1, y_1').$$

$\phi^3(y_1, y_1')$ is a SAT-Problem. Assume that the SAT-solver returns $y_1 = a$ and $y_1' = b$ as a satisfying assignment. We need to compute Skolem functions for $y_1$ and $y_2$ in $\phi^0$. The following table shows the Skolem functions for the individual formulas:

| | Skolem function for | | |
|---|---|---|---|
| **Formula** | $y_1$ | $y_1'$ | $y_2$ |
| $\Psi^3$ | $a$ | $b$ | n/a |
| $\Psi^2$ | $a$ | $b$ | n/a |
| $\Psi^1$ | $a$ | $b$ | $\phi^1(x_2, a, b, 1)$ |
| $\Psi^0$ | $(\neg x_1 \wedge a) \vee (x_1 \wedge b)$ | n/a | $\phi^1(x_2, a, b, 1)$ |

## 4 Handling pre- and inprocessing steps

Typically preprocessing is used to simplify the formula before the actual solution process starts. It is well known that preprocessing can reduce the computation times for solving the formula by orders of magnitude [41]. Thereby the set of variables occurring in the formula as well as its set of clauses change. For details about the preprocessing steps for DQBF, we refer the reader to [41].

The procedure for taking preprocessing steps into account is the same as for the elimination steps: We assume that Skolem functions $(s_y^k)_{y \in U^k}$ for $\Psi^k$ are given and show how to obtain Skolem functions $(s_y^{k-1})_{y \in U^{k-1}}$ for the formula $\Psi^{k-1}$ before applying a preprocessing operation.

### 4.1 Equivalence transformations and universal reduction

All operations which replace the formula $\Psi^{k-1}$ by a logically equivalent formula $\Psi^k$ (see Definition 3) preserve Skolem functions and can essentially be ignored for the computation of Skolem functions. This applies (among others) to the following preprocessing techniques: addition of resolvents, deletion of subsumed clauses, and hidden literal addition [41,22].

Universal reduction removes a variable $x^* \in U^{k-1}$ from a clause $C \in \phi^{k-1}$ if $C$ does not contain an existential variable which depends on $x^*$. In general, the resulting matrix $\phi^k$ is not logically equivalent to $\phi^{k-1}$. However, universal reduction changes neither the set of existential variables nor their Skolem functions. Therefore universal reduction steps can be ignored when computing Skolem functions.

### 4.2 Replacing variables by constants

Different techniques identify variables in the formula which must or may be replaced by constants: unit and failed literals, contradicting implication chains, backbones (variables which have the same value in all satisfying assignments of the matrix) [26], pure literals, or more generally, monotonic literals. For these techniques, Skolem functions can be derived using the following lemma.

**Lemma 6 (Replacement by constants).** *Assume that $\Psi^k$ is created from $\Psi^{k-1}$ by replacing an existential variable $y^* \in E^{k-1}$ by a constant value $c \in \mathbb{B}$, i. e., $Q^k : \phi^k = Q^{k-1} \setminus \left\{\exists y^*(D^{k-1}_{y^*})\right\} : \phi^{k-1}[c/y^*]$.*
*If $(s^k_y)_{y\in E^k}$ are Skolem functions for $\Psi^k$, then $(s^{k-1}_y)_{y\in E^{k-1}}$ are Skolem functions for $\Psi^{k-1}$, where $s^{k-1}_y = s^k_y$ for $y \neq y^*$, and $s^{k-1}_{y^*} = c$.*

While for backbones, the constant Skolem function is the only possibility, for monotonic variables other Skolem functions might be available. However, a constant function has a representation of minimum size and is therefore preferred.

### 4.3 Equivalent variables

If the preprocessor detects that the existential variable $y^* \in E^{k-1}$ is equivalent to the literal $\ell$, then either the whole formula is unsatisfied if $\ell$ is universal and $y^*$ does not depend on $\ell$. Otherwise all occurrences of $y^*$ can be replaced by $\ell$. For the Skolem function of $y^*$ the following lemma holds:

**Lemma 7 (Equivalent literals).** *Let $\Psi^k$ result from $\Psi^{k-1}$ by replacing the existential variable $y^* \in E^{k-1}$ by the literal $\ell$, i. e.,*

$$Q^k : \phi^k = Q^{k-1} \setminus \left\{\exists y^*(D^{k-1}_{y^*})\right\} : \phi^{k-1}[\ell/y^*].$$

*If $(s^k_y)_{y\in E^k}$ are Skolem functions for $\Psi^k$, then $(s^{k-1}_y)_{y\in E^{k-1}}$ are Skolem functions for $\Psi^{k-1}$, where*

$$s^{k-1}_y = \begin{cases} s^k_y, & \text{if } y \neq y^*, \\ \ell, & \text{if } y = y^* \text{ and } \mathrm{var}(\ell) \in U^{k-1}, \\ s^k_{\mathrm{var}(\ell)}, & \text{if } y = y^* \text{ and } \mathrm{var}(\ell) \in E^{k-1} \text{ and } \mathrm{sign}(\ell) = 1, \\ \neg s^k_{\mathrm{var}(\ell)}, & \text{if } y = y^* \text{ and } \mathrm{var}(\ell) \in E^{k-1} \text{ and } \mathrm{sign}(\ell) = 0. \end{cases}$$

### 4.4 Structure extraction

For solvers which do not rely on a CNF-representation of the formula, the reconstruction of the Boolean expression from which the CNF was generated is often beneficial. This is particularly the case if Tseitin transformation was applied to a circuit. Thereby clauses are detected which represent the equivalence $y^* \equiv \eta$ where $\eta$ is the function computed by a logical gate and $y^*$ the existential variable introduced by Tseitin transformation for the output of the gate. In the resulting representation $y^*$ is replaced by $\eta$. Accordingly, a Skolem function for $y^*$ can be obtained from the Skolem functions of the existential variables $\mathrm{var}(\eta)$:

**Lemma 8 (Structure extraction).** *Let $\Psi^k$ result from $\Psi^{k-1}$ by replacing $y^* \in E^{k-1}$ by the expression $\eta$ such that $y^* \notin \mathrm{var}(\eta)$ and*

$$\bigcup_{y \in \mathrm{var}(\eta) \cap E^{k-1}} D_y^{k-1} \subseteq D_{y^*}^{k-1}.$$

*That means $Q^k : \phi^k = Q^{k-1} \setminus \{\exists y^*(D_{y^*}^{k-1})\} : \phi^{k-1}[\eta/y^*]$. If $(s_y^k)_{y \in E^k}$ are Skolem functions for $\Psi^k$, then $(s_y^{k-1})_{y \in E^{k-1}}$ are Skolem functions for $\Psi^{k-1}$ where $s_y^{k-1} = s_y^k$ if $y \neq y^*$, and $s_{y^*}^{k-1} = \eta[s_z^k/z,\ \text{for } z \in \mathrm{var}(\eta) \cap E^k]$.*

### 4.5 Variable elimination by resolution

In QBF, an existential variable $y^*$ can be eliminated by resolution if it belongs to the inner-most quantifier block[3]. Thereby all clauses containing $y^*$ or $\neg y^*$ are replaced by all possible resolvents w. r. t. $y^*$. Having a closer look at how a Skolem function can be obtained for $y^*$, we can see that the condition of $y^*$ being in the inner-most quantifier block can be strengthend such that it is also applicable to DQBF, where there is in general no linear order on the variables.

Let $y^* \in E^{k-1}$ be an existential variable. We partition the set $\phi^{k-1}$ of clauses into $\phi_{y^*}^{k-1} = \{C \in \phi^{k-1} \mid y^* \in C\}$, $\phi_{\neg y^*}^{k-1} = \{C \in \phi^{k-1} \mid \neg y^* \in C\}$, and $\phi_{\emptyset}^{k-1} = \phi^{k-1} \setminus (\phi_{y^*}^{k-1} \cup \phi_{\neg y^*}^{k-1})$.

**Lemma 9 (Resolution).** *Assume that $\Psi^k$ results from $\Psi^{k-1}$ by eliminating variable $y^*$ using resolution, i. e.,*

$$Q^k : \phi^k = Q^{k-1} \setminus \{\exists y^*(D_{y^*}^{k-1})\} : \phi_{\emptyset}^{k-1} \cup \{C \otimes_{y^*} C' \mid C \in \phi_{y^*}^{k-1} \wedge C' \in \phi_{\neg y^*}^{k-1}\},$$

*where $C \otimes_{y^*} C'$ denotes the resolvent of $C$ and $C'$ w. r. t. $y^*$. This can be done if one of the following conditions holds:*

- *Case 1:* $\mathrm{dep}_{\Psi^{k-1}}(y^*) \supseteq \bigcup_{C \in \phi_{y^*}^{k-1}} \bigcup_{\ell \in C \setminus \{y^*\}} \mathrm{dep}_{\Psi^{k-1}}(\ell)$,
- *Case 2:* $\mathrm{dep}_{\Psi^{k-1}}(y^*) \supseteq \bigcup_{C' \in \phi_{\neg y^*}^{k-1}} \bigcup_{\ell \in C' \setminus \{\neg y^*\}} \mathrm{dep}_{\Psi^{k-1}}(\ell)$.

*If $(s_y^k)_{y \in E^k}$ are Skolem functions for $\Psi^k$, then $(s_y^{k-1})_{y \in E^{k-1}}$ are Skolem functions for $\Psi^{k-1}$ where, for the two cases, $s_y^{k-1}$ is defined as follows:*

$$s_y^{k-1} = \begin{cases} s_y^k, & \text{if } y \neq y^*, \\ \neg\phi_{y^*}^{k-1}[0/y^*][s_z^k/z \text{ for } z \in E^k], & \text{if } y = y^* \text{ and Case 1 applies,} \\ \neg\phi_{\neg y^*}^{k-1}[1/y^*][s_z^k/z \text{ for } z \in E^k], & \text{if } y = y^* \text{ and Case 2 applies.} \end{cases}$$

For a proof see [41]. If both cases apply, we can use don't-care minimization to reduce the AIG size of the Skolem function $s_{y^*}^{k-1}$ for the eliminated variable $y^*$.

Variable elimination by resolution is sound for DQBF also in a third case when an existential variable $y^* \in E^{k-1}$ fulfills the conditions for structure extraction [41]. (Depending on the solver back-end, one might prefer elimination by resolution instead of structure extraction in order to preserve the CNF structure of the matrix.) It is easy to see that in this case the Skolem function $s_{y^*}^{k-1}$ can simply computed from $(s_y^k)_{y \in E^k}$ as in Sect. 4.4.

[3] If the inner-most quantifier block is universal, it can be removed by universal reduction.

### 4.6 Blocked clause elimination (BCE)

BCE [22] allows to delete certain clauses $C$ from a formula without changing its truth value. This is the case if all resolvents of $C$ w. r. t. one of its existential literals $\ell \in C$ are tautologies and if the dependency set of the variable that makes the resolvent a tautology is a subset of $\mathrm{var}(\ell)$'s dependency set [41].

**Definition 7 (Outer clause, outer formula).** *Let $\psi = Q : \phi$ be a DQBF, $C \in \phi$ a clause, and $\ell \in C$ a literal of $C$. The* outer clause *of $C$ on $\ell$ is given by*

$$\mathcal{OC}(\psi, C, \ell) = \big\{\kappa \in C \,\big|\, \kappa \neq \ell \wedge \mathrm{dep}_\psi(\kappa) \subseteq \mathrm{dep}_\psi(\ell)\big\}.$$

*Let $\ell$ be a literal in a DQBF $\psi$. The* outer formula *of $\psi$ on $\ell$ is given by*

$$\mathcal{OF}(\psi, \ell) = \big\{\mathcal{OC}(\psi, D, \neg\ell) \,\big|\, D \in \phi \wedge \neg\ell \in D\big\}.$$

Now we can define blocked clauses for DQBF [41]:

**Definition 8 (Blocked clause).** *Let $\psi = Q : \phi$ be a DQBF, $C \in \phi$ a clause. The clause $C$ is* blocked *if there is an existential literal $\ell \in C$ such that $\mathcal{OC}(\psi, C, \ell) \cup \mathcal{OC}(\psi, D, \neg\ell)$ is a tautology for all $D \in \phi$ with $\neg\ell \in D$.*

It is known that blocked clauses can be deleted from a DQBF without changing its truth value [41].

Similar to the QBF case, which is described in [23], we can derive Skolem functions in case of blocked clause elimination using the following lemma:

**Lemma 10 (Blocked clause elimination).** *Let $\Psi^k$ be created from $\Psi^{k-1}$ by deleting the clause $C$, which is blocked in $\Psi^{k-1}$ w. r. t. the existential literal $\ell \in C$, i. e., $Q^k : \phi^k = Q^{k-1} : \phi^{k-1} \setminus \{C\}$. If $(s_y^k)_{y \in E^k}$ are Skolem functions for $\Psi^k$, then $(s_y^{k-1})_{y \in E^{k-1}}$ are Skolem functions for $\Psi^{k-1}$ where*

$$s_y^{k-1} = \begin{cases} s_y^k, & \text{if } y \neq \mathrm{var}(\ell), \\ \mathrm{ITE}\big(\mathcal{OF}(\Psi^{k-1}, \ell)[s_z^k/z \text{ for } z \in E^k],\ \mathrm{sign}(\ell),\ s_y^k\big), & \text{if } y = \mathrm{var}(\ell). \end{cases}$$

*Proof sketch.* Let $y^* := \mathrm{var}(\ell)$. First, $s_{y^*}^{k-1}$ depends only on variables in $D_{y^*}^{k-1}$ because $\mathcal{OF}(\Psi^{k-1}, \ell)$ contains only variables $v$ with $\mathrm{dep}_{\Psi^{k-1}}(v) \subseteq \mathrm{dep}_{\Psi^{k-1}}(y^*)$.

Second, we have to show that $\phi^{k-1}[s_z^{k-1}/z \text{ for } z \in E^{k-1}]$ is a tautology. We show $\nu\big(\phi^{k-1}[s_z^{k-1}/z \text{ for } z \in E^{k-1}]\big) = 1$ for every assignment $\nu \in \mathbf{A}(U^{k-1})$. We partition the clauses into those which contain $\ell$, those which contain $\neg\ell$, and the remaining ones. We distinguish the cases where $\nu\big(\mathcal{OF}(\Psi^{k-1}, \ell)\big)$ is 0 and where it is 1, and prove that in both cases all clauses are satisfied. Details can be found in [40]. □

The effectiveness of blocked clause elimination is often increased by adding hidden and covered literals before testing whether a clause is blocked. Adding hidden literals yields an equivalent DQBF (see Section 4.1), but the addition of covered literals has to be taken into account when computing Skolem functions.

For detailed information how to handle covered literal addition, we refer the reader to the extended version [40] of this paper.

## 5 Experimental results

We extended our DQBF solver HQS [19][4] by the possibility to compute Skolem functions for satisfied DQBFs. The computation of Skolem functions works in two phases: During the solution process we collect the necessary data and store it on a stack. When the satisfiability of the formula has been determined, we free the other data structures of the solver and extract the Skolem functions from the collected data. During the extraction phase, HQS supports optimizing the Skolem functions of eliminated existential variables according to Remarks 2 and 3. If don't-care optimization using Craig interpolation is enabled, we choose a Skolem function for the eliminated existential variable $y^* \in E^{k-1}$ among $\phi^{k-1}[1/y^*]$, $\neg\phi^{k-1}[0/y^*]$, and the computed interpolant, taking one with minimal AIG size. If interpolation is disabled, we only choose among the first two options.

Additionally, in a post-processing step, we can use the tool ABC [8] to further optimize the Skolem function representations. It supports AIG rewriting based on so-called *internal* don't cares. In contrast to the *external* don't cares that we proposed in Remark 3, they encompass values which cannot appear at internal signals of the AIG.

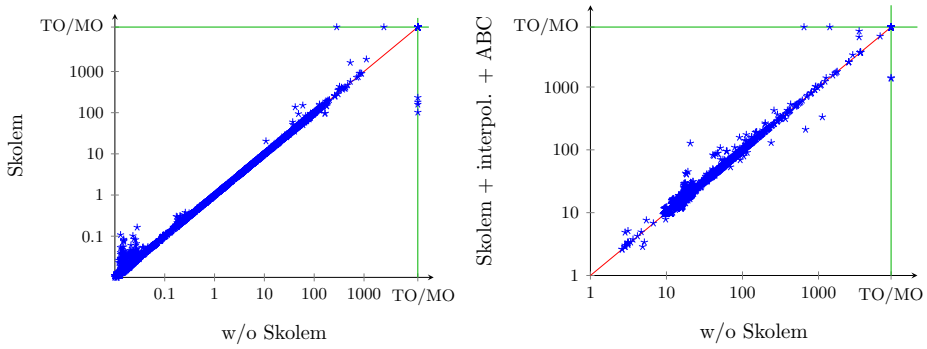
HQS is accompanied by a proof checker, which verifies that the Skolem functions depend only on the allowed variables and that replacing the existential variables in the formula by their Skolem function indeed yields a tautology. Checking whether the Skolem functions depend only on the allowed variables is performed just by traversing the AIGs and computing the *structural* support, since, by construction, the AIGs of the Skolem functions do not structurally depend on more than the allowed variables.[5] Logic optimizations done by ABC could increase the structural support in principle, but since logic optimization does not change the represented Boolean functions semantically, additional variables in the structural support which are introduced by ABC can be removed by replacing them by arbitrary constants.[6] The second and more important part of the check is done by replacing the existential variables by their Skolem functions and by calling a SAT-solver to verify that the resulting formula is a tautology. As a SAT-solver, we have used Minisat 2.2 [13]. We have applied this proof checker to all computed Skolem functions and confirmed their correctness.

All experiments were run on one Intel Xeon E5-2650v2 CPU core at 2.60 GHz clock frequency and 64 GB of main memory under Linux (kernel version 3.13) as operating system, running in 64 bit mode. We aborted all experiments which either took more than 3600 s CPU time or more than 8 GB ( $= 2^{30}$ bytes) of main memory. As benchmarks we used 4811 DQBF instances from different sources: DQBFs resulting from equivalence checking of incomplete circuits [18,15,17], controller synthesis problems [7], and instances obtained from converting SAT instances into DQBFs that depend only on a logarithmic number of variables [4].

[4] A recent binary of HQS, all DQBF benchmarks we used as well as our proof checker are available at `https://projects.informatik.uni-freiburg.de/projects/dqbf`.

[5] Of course, the semantical support could also be checked by a series of SAT calls.

[6] However, this case never occurred in our experiments.




(a) Computation time (in seconds) (b) Memory consumption (in MB)

**Fig. 1.** Influence of data collection on the computation time (left) and memory consumption with and without computation of Skolem functions (right)

First we compare the number of instances which could be solved and for which Skolem functions could be computed in different solver configurations. Table 1 shows the results. Out of the 4811 instances, on average 4010 could be solved; of those instances, 3288 were UNSAT, 722 SAT. Skolem functions were obtained for all of the solved SAT instances, i. e., we could not only decide realizability, but even determine implementations of controllers or the unknown circuit parts. These numbers are independent of whether interpolation and/or ABC were used to optimize the size of the Skolem functions. The small deviations seem to be due to random effects from scheduling and influences from other processes. The numbers change by one or two when we re-run the solver in the same configuration.

**Table 1.** Solved instances

| Variant | solved | unsat | sat | Skolem |
|---|---|---|---|---|
| w/o Skolem functions | 4008 | 3286 | 722 | n/a |
| w/o optimizations | 4010 | 3289 | 721 | 721 |
| + interpolation | 4010 | 3289 | 721 | 721 |
| + ABC | 4009 | 3287 | 722 | 722 |
| + interpolation + ABC | 4008 | 3287 | 721 | 721 |

This is consistent with Fig. 1, which shows the influence of *data collection* on the computation time (left) until the truth value of the formula has been determined and on the peak memory consumption (right) until the computation of Skolem functions has been finished. The memory consumption of ABC is not taken into account, because it runs when HQS has terminated and needed less memory than HQS in all cases. A mark below the diagonal means that the variant on the vertical axis performs better for that instance than the variant on the horizontal axis. The figures show that the solution time only changes by a small amount due to data collection. The memory consumption increases only slightly in most cases. In a few exceptions, the memory consumption even decreases. The reason is that the Skolem functions share AIG nodes with the formula in the solver core. This changes the way how the AIG manager can optimize the AIG representation, which can actually lead to lower memory consumption.

The final extraction phase takes a few seconds at most, even if the optimizations are enabled. Since the internal data structures of the solver have already

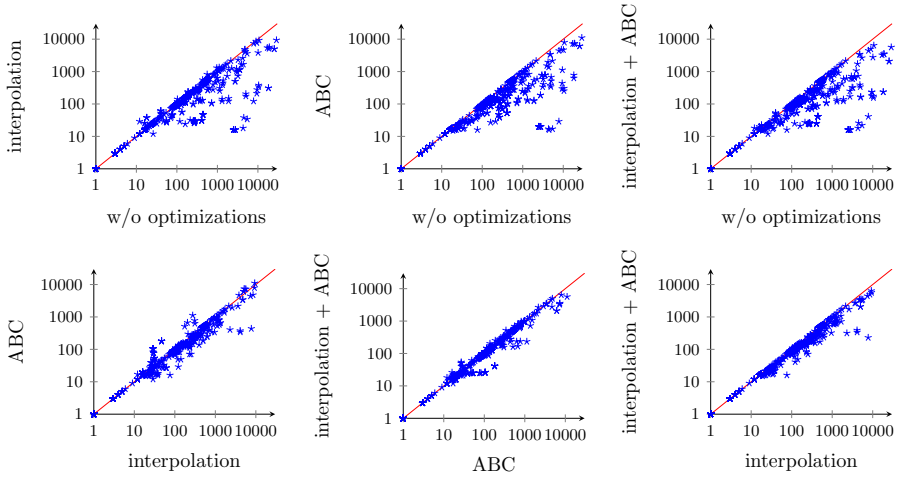



**Fig. 2.** The sizes (number of AND nodes) in the Skolem functions' AIG representations with different optimizations enabled. Note that the axes are logarithmically scaled.

been freed at that point, the peak memory consumption does not occur during Skolem function computation, but during the solution process.

In Fig. 2, we compare the effectiveness of optimizing the Skolem functions by don't-care minimization and rewriting. We ran HQS on the satisfiable instances with five different configurations: (1) without any optimization of the Skolem functions (besides taking the smaller one of $\phi^{k-1}[1/y^*]$ and $\neg\phi^{k-1}[0/y^*]$ for eliminated existential variables, cf. Remark 2); (2) applying ABC to the obtained Skolem functions; (3) using interpolation according to Lemma 2; and (4) using both interpolation and ABC. The diagrams show the values for all (roughly) 722 instances for which we were able to compute Skolem functions.

We can observe that both interpolation (first row, left) and ABC (first row, mid) in isolation have, on average, a positive effect on the sizes of the Skolem functions. Nevertheless, since we perform don't-care optimization using interpolation for each eliminated existential variable individually, it may in a few cases increase the joint size of all Skolem functions, which share some of the AIG nodes. In contrast, ABC never increases the size, because it performs optimization globally for the shared AIGs. The left diagram in the second row compares the effectiveness of ABC and interpolation. While they are similarly effective in many cases, there are instances

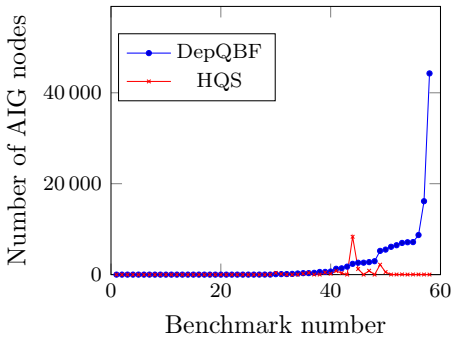



**Fig. 3.** Comparing DepQBF and HQS regarding the size of the computed Skolem functions

for which ABC is superior to interpolation and vice versa. Therefore, adding both optimizations often leads to a further decrease in size (second row, mid and right).

Because QBFs are a special case of DQBFs, we can use HQS to compute Skolem functions for satisfied QBFs. In Fig. 3, we compare the sizes of the Skolem functions generated by HQS with those generated by the state-of-the-art QBF solver DepQBF 5.0 [29,28] for a set of satisfiable QBF instances from the QBF Gallery 2013[7] and from partial equivalence checking [37] (with a single black box). Since HQS (and in particular its preprocessor) is not optimized for solving QBF instances, we abstain from a detailed comparison of the running times of HQS and DepQBF. DepQBF is often (but not always) faster than HQS. In a few cases, the generation of Skolem functions with DepQBF failed because the necessary resolution proof became too large (we aborted DepQBF when the size of the dumped resolution proof exceeded 20 GB).

Fig. 3 shows the sizes of the Skolem functions computed by DepQBF and by HQS (with interpolation and ABC). To enable a fair comparison, we also applied ABC with the same commands to the Skolem functions generated using DepQBF. We can observe that HQS' Skolem functions are in most cases smaller (often significantly) than those obtained from DepQBF.

In summary, we can conclude that the proposed method allows the computation of Skolem function for satisfied DQBFs with very little overhead regarding computation time and memory consumption. Applying interpolation and ABC to decrease the size of the Skolem functions has in general a positive effect. Regarding the sizes of the computed Skolem functions, HQS is at least comparable to the QBF solver DepQBF on small to medium size QBF instances.

## 6 Conclusion

We have shown how Skolem functions can be computed for satisfiable DQBFs. They play a crucial role in many applications from implementations of missing circuit parts or controllers to winning strategies in games. We have shown how don't-care minimization can help reduce the size of the Skolem functions. In a series of experiments we demonstrated that the computation of Skolem functions is not only possible in theory but also feasible in practice: both the overhead during the solution process and the time for extracting the functions from the collected data are small.

An open problem is the certification of unsatisfiability. For QBFs this can be done by negating the formula and then computing Skolem functions (which are here called Herbrand functions). This is not possible for DQBFs [2] because DQBFs are not closed under negation.[8] Finding ways to certify the unsatisfiability of a DQBF is an important task for future work.

[7] see `http://www.kr.tuwien.ac.at/events/qbfgallery2013/`

[8] The DQBF-variant we consider is called S-form DQBF in [2]. Its negation yields a so-called H-form DQBF, which does not support the computation of Skolem functions.

## References


1. Ashar, P., Ganai, M.K., Gupta, A., Ivancic, F., Yang, Z.: Efficient SAT-based bounded model checking for software verification. In: Proc. of ISoLA. Technical Report, vol. TR-2004-6, pp. 157–164. University of Cyprus (2004)
2. Balabanov, V., Chiang, H.K., Jiang, J.R.: Henkin quantifiers and boolean formulae: A certification perspective of DQBF. Theor. Comp. Sci. 523, 86–100 (2014)
3. Balabanov, V., Jiang, J.R.: Unified QBF certification and its applications. Formal Methods in System Design 41(1), 45–65 (2012)
4. Balabanov, V., Jiang, J.H.R.: Reducing satisfiability and reachability to DQBF (2015), talk at the Int'l Workshop on Quantified Boolean Formulas (QBF)
5. Benedetti, M.: Evaluating QBFs via symbolic Skolemization. In: Proc. of LPAR. LNCS, vol. 3452, pp. 285–300. Springer (2005)
6. Biere, A., Cimatti, A., Clarke, E.M., Strichman, O., Zhu, Y.: Bounded model checking. Advances in Computers 58, 117–148 (2003)
7. Bloem, R., Könighofer, R., Seidl, M.: SAT-based synthesis methods for safety specs. In: Proc. of VMCAI. LNCS, vol. 8318, pp. 1–20. Springer (2014)
8. Brayton, R.K., Mishchenko, A.: ABC: an academic industrial-strength verification tool. In: Proc. of CAV. LNCS, vol. 6174, pp. 24–40. Springer (2010)
9. Bubeck, U., Kleine Büning, H.: Dependency quantified Horn formulas: Models and complexity. In: Proc. of SAT. LNCS, vol. 4121, pp. 198–211. Springer (2006)
10. Cook, S.A.: The complexity of theorem-proving procedures. In: Proc. of STOC. pp. 151–158. ACM Press (1971)
11. Craig, W.: Linear reasoning. A new form of the Herbrand-Gentzen theorem. Journal of Symbolic Logic 22(3), 250–268 (1957)
12. Czutro, A., Polian, I., Lewis, M.D.T., Engelke, P., Reddy, S.M., Becker, B.: Thread-parallel integrated test pattern generator utilizing satisfiability analysis. Int'l Journal of Parallel Programming 38(3-4), 185–202 (2010)
13. Eén, N., Sörensson, N.: An extensible SAT-solver. In: Proc. of SAT. LNCS, vol. 2919, pp. 502–518. Springer (2003)
14. Eggersglüß, S., Drechsler, R.: A highly fault-efficient SAT-based ATPG flow. IEEE Design & Test of Computers 29(4), 63–70 (2012)
15. Finkbeiner, B., Tentrup, L.: Fast DQBF refutation. In: Proc. of SAT. LNCS, vol. 8561, pp. 243–251. Springer (2014)
16. Fröhlich, A., Kovásznai, G., Biere, A.: A DPLL algorithm for solving DQBF. In: Int'l Workshop on Pragmatics of SAT (POS) (2012)
17. Fröhlich, A., Kovásznai, G., Biere, A., Veith, H.: iDQ: Instantiation-based DQBF solving. In: Int'l Workshop on Pragmatics of SAT (POS). EPiC Series, vol. 27, pp. 103–116. EasyChair (2014)
18. Gitina, K., Reimer, S., Sauer, M., Wimmer, R., Scholl, C., Becker, B.: Equivalence checking of partial designs using dependency quantified Boolean formulae. In: Proc. of ICCD. pp. 396–403. IEEE CS (2013)
19. Gitina, K., Wimmer, R., Reimer, S., Sauer, M., Scholl, C., Becker, B.: Solving DQBF through quantifier elimination. In: Proc. of DATE. IEEE (2015)
20. Goultiaeva, A., Van Gelder, A., Bacchus, F.: A uniform approach for generating proofs and strategies for both true and false QBF formulas. In: Proc. of IJCAI. pp. 546–553. IJCAI/AAAI (2011)
21. Henkin, L.: Some remarks on infinitely long formulas. In: Infinitistic Methods: Proc. of the 1959 Symp. on Foundations of Mathematics. pp. 167–183. Pergamon Press, Warsaw, Panstwowe (1961)
22. Heule, M., Järvisalo, M., Lonsing, F., Seidl, M., Biere, A.: Clause elimination for SAT and QSAT. Journal of Artificial Intelligence Research 53, 127–168 (2015)

23. Heule, M., Seidl, M., Biere, A.: Efficient extraction of Skolem functions from QRAT proofs. In: Proc. of FMCAD. pp. 107–114. IEEE (2014)
24. Jiang, J.R.: Quantifier elimination via functional composition. In: Proc. of CAV. LNCS, vol. 5643, pp. 383–397. Springer (2009)
25. Jussila, T., Biere, A., Sinz, C., Kröning, D., Wintersteiger, C.M.: A first step towards a unified proof checker for QBF. In: Proc. of SAT. LNCS, vol. 4501, pp. 201–214. Springer (2007)
26. Kilby, P., Slaney, J.K., Thiébaux, S., Walsh, T.: Backbones and backdoors in satisfiability. In: Proc. of NAI/IAAI. pp. 1368–1373. AAAI Press / The MIT Press (2005)
27. Kuehlmann, A., Paruthi, V., Krohm, F., Ganai, M.K.: Robust boolean reasoning for equivalence checking and functional property verification. IEEE Trans. on CAD of Integrated Circuits and Systems 21(12), 1377–1394 (2002)
28. Lonsing, F., Bacchus, F., Biere, A., Egly, U., Seidl, M.: Enhancing search-based QBF solving by dynamic blocked clause elimination. In: Proc. of LPAR. LNCS, vol. 9450, pp. 418–433. Springer (2015)
29. Lonsing, F., Biere, A.: DepQBF: A dependency-aware QBF solver. Journal on Satisfiability, Boolean Modelling and Computation 7(2-3), 71–76 (2010)
30. McMillan, K.L.: Applications of Craig interpolants in model checking. In: Proc. of TACAS. LNCS, vol. 3440, pp. 1–12. Springer (2005)
31. Niemetz, A., Preiner, M., Lonsing, F., Seidl, M., Biere, A.: Resolution-based certificate extraction for QBF. In: Proc. of SAT. LNCS, vol. 7317, pp. 430–435. Springer (2012)
32. Peterson, G., Reif, J., Azhar, S.: Lower bounds for multiplayer non-cooperative games of incomplete information. Computers & Mathematics with Applications 41(7–8), 957–992 (2001)
33. Pigorsch, F., Scholl, C.: Exploiting structure in an AIG based QBF solver. In: Proc. of DATE. pp. 1596–1601. IEEE (2009)
34. Pigorsch, F., Scholl, C.: An AIG-based QBF-solver using SAT for preprocessing. In: Proc. of DAC. pp. 170–175. ACM Press (2010)
35. Pudlák, P.: Lower bounds for resolution and cutting planes proofs and monotone computations. Journal of Symbolic Logic 62(3), 981–998 (1997)
36. Rintanen, J., Heljanko, K., Niemelä, I.: Planning as satisfiability: parallel plans and algorithms for plan search. Artificial Intelligence 170(12-13), 1031–1080 (2006)
37. Scholl, C., Becker, B.: Checking equivalence for partial implementations. In: Proc. of DAC. pp. 238–243. ACM Press (2001)
38. Tentrup, L., Rabe, M.N.: CAQE: A certifying QBF solver. In: Proc. of FMCAD. pp. 136–143. IEEE (2015)
39. Wegener, I.: Branching programs and binary decision diagrams. SIAM Monographs on Discrete Mathematics and Applications, SIAM (2000)
40. Wimmer, K., Wimmer, R., Scholl, C., Becker, B.: Skolem functions for DQBF (extended version). Tech. rep., FreiDok plus, Universitätsbibliothek Freiburg, Freiburg im Breisgau, Germany (Jun 2016), available at `https://www.freidok.uni-freiburg.de`
41. Wimmer, R., Gitina, K., Nist, J., Scholl, C., Becker, B.: Preprocessing for DQBF. In: Proc. of SAT. LNCS, vol. 9340, pp. 173–190. Springer (2015)