AVACS – Automatic Verification and Analysis of Complex Systems

REPORTS
of SFB/TR 14 AVACS

Editors: Board of SFB/TR 14 AVACS

Fast DQBF Refutation

by
Bernd Finkbeiner Leander Tentrup

AVACS Technical Report No. 97
August 2014
ISSN: 1860-9821

Publisher: Sonderforschungsbereich/Transregio 14 AVACS
(Automatic Verification and Analysis of Complex Systems)
Editors: Bernd Becker, Werner Damm, Bernd Finkbeiner, Martin Fränzle,
Ernst-Rüdiger Olderog, Andreas Podelski
ATRs (AVACS Technical Reports) are freely downloadable from `www.avacs.org`

Author(s) contact: Bernd Finkbeiner (`bf@avacs.org`).

Fast DQBF Refutation

Bernd Finkbeiner and Leander Tentrup

Saarland University

Abstract. Dependency Quantified Boolean Formulas (DQBF) extend QBF with Henkin quantifiers, which allow for non-linear dependencies between the quantified variables. This extension is useful in verification problems for incomplete designs, such as the partial equivalence checking (PEC) problem, where a partial circuit, with some parts left open as “black boxes”, is compared against a full circuit. The PEC problem is to decide whether the black boxes in the partial circuit can be filled in such a way that the two circuits become equivalent, while respecting that each black box only observes the subset of the signals that are designated as its input. We present a new algorithm that efficiently refutes unsatisfiable DQBF formulas. The algorithm detects situations in which already a subset of the possible assignments of the universally quantified variables suffices to rule out a satisfying assignment of the existentially quantified variables. Our experimental evaluation on PEC benchmarks shows that the new algorithm is a significant improvement both over approximative QBF-based methods, where our results are much more accurate, and over precise methods based on variable elimination, where the new algorithm scales better in the number of Henkin quantifiers. This is an extended version of [8].

1 Introduction

Dependency Quantified Boolean Formulas (DQBF) are an extension of QBF which allows for non-linear dependencies between quantified variables. Non-linear dependencies occur naturally in verification problems for incomplete designs, such as the *partial equivalence checking* (PEC) problem [10], where a partial circuit, with some parts left open as “black boxes”, is compared against a full circuit. The inputs to the circuit are modeled as universally quantified variables and the outputs of the black boxes as existentially quantified variables. Since the output of a black box should only depend on the inputs that are actually visible to the black box, we need to restrict the dependencies of the existentially quantified variables to subsets of the universally quantified variables.

There has been some success in extending standard techniques of QBF solving to DQBF [3, 9, 10], but, generally, it has proven very difficult to scale the classic algorithms to larger DQBF problems. Fröhlich et al. conclude, based on experiments with various techniques from DPLL-based SAT/QBF-solving, including unit propagation, pure literal reduction, clause learning, selection heuristics and watched literal schemes, that “it does not perform very well” [9].

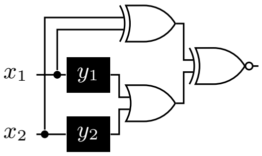


Fig. 1. Example of a partial equivalence checking (PEC) problem. A partial design, consisting here of the two black boxes and the OR gate, is compared to the reference circuit above, here consisting of a single XOR gate. The output of the complete circuit is 1 iff the completion of the partial design and the reference circuit compute the same result.

A much faster alternative to such precise methods is to *approximate* the dependencies, such that all dependencies become linear and DQBF thus simplifies to QBF. For the PEC problem, an overapproximation of the dependencies is still useful to find errors (if the black box cannot be implemented with additional inputs, then it can, for sure, not be implemented according to the original design), but it significantly decreases the accuracy, because errors that result precisely from the incomparable dependencies of the black boxes are no longer detected. Consider, for example, the toy PEC problem shown in Fig. 1, where we ask whether it is possible to implement the XOR gate at the top as an OR of the two black boxes below, which *each* only see *one* of the two inputs x_1 and x_2. This is obviously not possible; however, the three overapproximating linearizations $\forall x_1 \forall x_2 \exists y_1 \exists y_2$, $\forall x_1 \exists y_1 \forall x_2 \exists y_2$ and $\forall x_1 \exists y_2 \forall x_2 \exists y_1$ all result in a positive answer, because an output that depends on both x_1 and x_2 can compute $x_1 \oplus x_2$, which gives the correct result, assuming that the other black box simply outputs constant 0.

In this paper, we present a new algorithm for DQBF that combines the efficiency of the QBF abstraction with the accuracy of the classic methods [3,9,10]. We focus on the *refutation* of DQBF, because this corresponds to the identification of errors in the PEC problem. Our algorithm identifies situations in which already a subset of the possible assignments of the universally quantified variables suffices to rule out a satisfying assignment of the existentially quantified variables. We call assignments to the universal variables *paths*. In the PEC example from Fig. 1, there are 4 possible paths ($x_1x_2, x_1\overline{x_2}, \overline{x_1}x_2, \overline{x_1x_2}$). However, already 3 paths[1], $x_1x_2, x_1\overline{x_2}$, and $\overline{x_1x_2}$, suffice to rule out a satisfying assignment for the existential variables: Since y_1 does not depend on x_2, its value must be the same for x_1x_2 and for $x_1\overline{x_2}$; likewise, the value of y_2 must be the same for $x_1\overline{x_2}$ and for $\overline{x_1x_2}$. For x_1x_2, both y_1 and y_2 must be 0, because $1 \oplus 1 = 0$. However, if $y_1 = 0$ for x_1x_2, then $y_1 = 0$ also for $x_1\overline{x_2}$, which leads to a contradiction, because y_2 must be equal to 1 for $x_1\overline{x_2}$ because $1 \oplus 0 = 1$ and, at the same time, equal to 0, because $0 \oplus 0 = 0$. In our algorithm, we specify the existence of such a set of paths as a QBF formula. We iteratively increase the number of paths to be considered and terminate as soon as a satisfying assignment is ruled out.

Related Work. DQBF was first defined by Peterson and Reif [15] and gained more attention recently [1, 10]. The first investigation of practical methods for

[1] Later in the paper, we present an optimization that further reduces the number of paths required in this example to just two.

DQBF solving is a DPLL-based approach due to Fröhlich et al. [9]. In addition, there is an expansion-based solver [10]. It is also possible to reduce DQBF to SAT, using skolemization methods similar to those originally developed for QBF [3]. To the best of our knowledge, there is, however, so far no publicly available DQBF solver. The idea of partial expansions was used in previous work, e.g., there is a two-phase proof system for QBF that expands certain paths and than refutes the formula by propositional resolution [11]. The verification of incompletely specified circuits has received significant attention (cf. [14, 16]); the connection between DQBF and the PEC problem was first pointed out by Gitina et al. [10]. On a more general level, the verification of partial designs is related to the synthesis problems for reactive systems with incomplete information and for distributed systems (cf. [6, 12]). In previous work, we have proposed an efficient method for disproving the existence of distributed realizations of specifications given in linear-time temporal logic (LTL) [7] that bounds, similar to the approach of this paper, the number of paths under consideration.

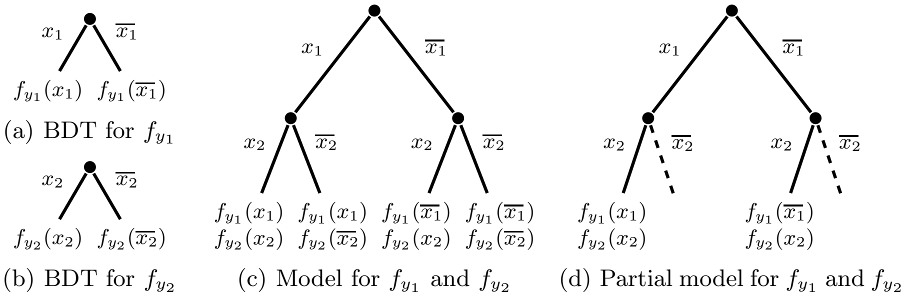
2 DQBF

Let $\mathcal{V}$ be a finite set of propositional variables. We use the convention to denote the set of all universal variables $\mathcal{X}$, an element $x \in \mathcal{X}$, and a subset $X \subseteq \mathcal{X}$ ($y \in Y \subseteq \mathcal{Y}$ for existential variables, respectively). The standard form of DQBF [9] is

$$\forall x_1 . \forall x_2 . \forall x_3 \ldots \exists_{H_1} y_1 . \exists_{H_2} y_2 . \exists_{H_3} y_3 \ldots \varphi \ , \tag{1}$$

that is, formulas beginning with universal quantified variables followed by the existentially quantified *Henkin quantifier* and the quantifier-free *matrix* φ. A Henkin quantifier $\exists_H y$ explicitly states the dependency for variable y by its support set $H \subseteq \mathcal{X}$, which is the difference to QBF, where the preceding universal quantification determines the dependency of an existential variable. For the matrix φ we allow negation $\neg$, disjunction $\vee$, conjunction $\wedge$, implication $\rightarrow$, equivalence $\leftrightarrow$, exclusive or $\oplus$, and the abbreviations *true* $\top$ and *false* $\bot$.

A DQBF formula Φ is satisfiable, if there exists a *Skolem function* f_y for each existential variable $y \in \mathcal{Y}$, such that for all possible assignments of the universal variables $\mathcal{X}$, the Skolem functions evaluated on these assignments satisfy the matrix. An assignment is a function $\alpha : \mathcal{V} \rightarrow \{0, 1\}$ and a Skolem function $f_y : (H_y \rightarrow \{0, 1\}) \rightarrow (\{y\} \rightarrow \{0, 1\})$ maps assignments of the dependencies to an assignment of y. We identify an assignment α by a set $\{v \in \mathcal{V} \mid \alpha(v) = 1\} \subseteq 2^{\mathcal{V}}$ and f_y by a function $2^{H_y} \rightarrow 2^{\{y\}}$. We represent a function f_y as a binary decision tree (BDT), where the branching of the tree represents the assignment of the dependencies and f_y serves as the labeling function for the leaves, see Fig. 2(a)–(c) for examples of BDTs. As a notation for paths, we use sequences of (possibly negated) variables $x \in \mathcal{X}$, e.g., the path $x_1\overline{x_2}$ is a shorthand for the assignment $\{x_1\}$. A path $X \subseteq \mathcal{X}$ of a BDT satisfies a propositional formula φ, if the assignment $X \cup f_y(X)$, i.e., the joint assignment of this path and the respective labeling, satisfies φ. A *model* $\mathcal{M}$ of a satisfiable formula Φ is a binary decision tree over $\mathcal{X}$ such that (1) every path in the tree satisfies the matrix



x_1
$\overline{x_1}$
$f_{y_1}(x_1)$
$f_{y_1}(\overline{x_1})$
(a) BDT for f_{y_1}
x_2
$\overline{x_2}$
$f_{y_2}(x_2)$
$f_{y_2}(\overline{x_2})$
(b) BDT for f_{y_2}
$f_{y_1}(x_1)$ $f_{y_2}(x_2)$
$f_{y_1}(x_1)$ $f_{y_2}(\overline{x_2})$
$f_{y_1}(\overline{x_1})$ $f_{y_2}(x_2)$
$f_{y_1}(\overline{x_1})$ $f_{y_2}(\overline{x_2})$
(c) Model for f_{y_1} and f_{y_2}
$f_{y_1}(x_1)$ $f_{y_2}(x_2)$
$f_{y_1}(\overline{x_1})$ $f_{y_2}(x_2)$
(d) Partial model for f_{y_1} and f_{y_2}

Fig. 2. The figure shows the binary decision trees for f_{y_1} (a) and f_{y_2} (b), their composition (c), and a partial model (d) for the PEC problem in Fig. 1.

and (2) the labels of the leaves are *consistent* according to the dependencies of the existential variables, i.e., there exists a decomposition of the decision tree into individual Skolem functions f_y for each $y \in \mathcal{Y}$. For example, the Skolem functions of a satisfiable DQBF formula

$$\forall x_1, x_2.\, \exists_{\{x_1\}}\, y_1.\, \exists_{\{x_2\}}\, y_2.\, \varphi \tag{2}$$

are the unary functions f_{y_1} and f_{y_2}, depicted in Fig. 2(a) and (b), respectively. Figure 2(c) shows the corresponding model, that is the composition of f_{y_1} and f_{y_2}. In this representation, the incomparable dependencies become visible: Despite the branching of the tree by both variables x_1 and x_2, the results of the Skolem functions f_{y_1} and f_{y_2} must be equal on paths that cannot be distinguished according to the dependencies, e.g., as y_2 does not depend on x_1, the paths $x_1 x_2$ and $\overline{x_1} x_2$ are indistinguishable for f_{y_2} and the result on both paths is $f_{y_2}(x_2)$. A DQBF formula is *unsatisfiable* if there does not exist a model, i.e., for all *candidate models* always at least one path violates the matrix φ.

QBF Approximation. Given a DQBF formula Φ, a QBF formula Ψ with the same matrix is an approximation of Φ, written $\Phi \preceq \Psi$ if for all existential variables $y \in \mathcal{Y}$ it holds that $H_y \subseteq X_y$, where H_y is the support set of y and $X_y \subseteq \mathcal{X}$ is the dependency set of y in the QBF formula. Given two QBF approximations Ψ and Ψ', we call Ψ *stronger* than Ψ', written $\Psi \preceq \Psi'$, if for all $y \in \mathcal{Y}$ it holds that $X_y \subseteq X'_y$ [10]. In (2), y_1 and y_2 have *incomparable* dependencies as neither $\{x_1\} \subseteq \{x_2\}$ nor $\{x_2\} \subseteq \{x_1\}$. Hence, in all strongest QBF approximations, that is $\forall x_1 \exists y_1 \forall x_2 \exists y_2$ and $\forall x_2 \exists y_2 \forall x_1 \exists y_1$, at least one existential variable has more dependencies than before. The resulting inaccuracy was already highlighted in the introduction on the PEC problem from Fig. 1, which corresponds to formula (2) with matrix $\varphi = (y_1 \vee y_2) \leftrightarrow (x_1 \oplus x_2)$. All QBF abstractions of (2) are satisfiable despite the DQBF formula being unsatisfiable.

Variable Elimination. The expansion based method for converting a DQBF formula Φ into a logically equivalent QBF formula Ψ [1, 5] uses the idea of unrolling the binary decision tree, e.g., expanding (2) by x_1 gives us:

$$\forall x_2 .\, \exists_{\{x_2\}}\, y_2 .\, \exists_{\emptyset}\, y_1, y_1' .\, \varphi|_{x_1=\bot} \wedge \varphi'|_{x_1=\top} \quad , \tag{3}$$

where $\varphi|_{x=b}$ denotes the formula φ where all occurrences of x are substituted by b and φ' is the formula obtained from φ by replacing all occurrences of y_1 by y_1'. In the expansion of x_1, only variable y_1 is duplicated to represent the different choices of the Skolem function f_{y_1} on the paths that differ in the assignment of x_1. Likewise, variable y_2 is not duplicated. After the expansion of all universal variables, the resulting existential QBF formula can be solved by a SAT solver.

3 Bounded Unsatisfiability

Instead of expanding the whole binary decision tree it is often possible to determine unsatisfiability with only a subset of the assignments to the universal variables. The restricted choice of a Skolem function based on its dependencies can be also defined as a *consistency condition* between paths: Given two paths $P_1, P_2 \in 2^{\mathcal{X}}$, different assignments of an existential variable $y \in \mathcal{Y}$ on P_1 and P_2 are *consistent* if the assignments of H_y on P_1 and P_2 are different. We introduce the notion of partial models that are decision trees which contain only a subset of the original paths. Formally, a *partial model* $\mathcal{P}$ of a DQBF formula Φ is a decision tree over $\mathcal{X}$ consisting of paths $P \subseteq 2^{\mathcal{X}}$ such that (1) every path in the tree satisfies the matrix and (2) the labels of the leaves are *consistent* according to the dependencies of the existential variables and the selected paths P. As partial models are weaker than models, the existence of a partial model does not imply the existence of a model, but from the non-existence of a partial model follows the non-existence of a model.

Lemma 1. *Given a DQBF formula Φ and a set of paths $P \subseteq 2^{\mathcal{X}}$. Φ is unsatisfiable if there does not exist a partial model over P.*

Proof. We only consider the case $P \neq 2^{\mathcal{X}}$, as for $P = 2^{\mathcal{X}}$ the definition of partial models and models coincide. Assume for contradiction that there does not exist a partial model over P while Φ is satisfiable, i.e., there exists a model $\mathcal{M}$ of Φ. We define a candidate partial model $\mathcal{P}$ by restricting $\mathcal{M}$ to the paths P. From the definition of models, it follows that all paths in $\mathcal{P}$ satisfy φ and the labeling of the leaves are consistent on all paths. Hence, $\mathcal{P}$ is a partial model of Φ, violating our assumption. □

Bounded Unsatisfiability. We turn the idea of non-existing partial models into the bounded unsatisfiability problem that limits the number of paths under consideration in order to show that no partial model exists. For a $k \geq 1$, a DQBF formula Φ is *k-bounded unsatisfiable* if there exists a set of paths $P \subseteq 2^{\mathcal{X}}$ with $|P| \leq k$ such that there does not exist a partial model over P.

Theorem 2. *A DQBF formula Φ is unsatisfiable iff it is k-bounded unsatisfiable for some $k \geq 1$.*

Proof. If a DQBF formula Φ is unsatisfiable, it follows immediately that it is $2^{|\mathcal{X}|}$-bounded unsatisfiable as $2^{|\mathcal{X}|}$ is the total number of paths. Assume that a DQBF formula Φ is k-bounded unsatisfiable, then there exists a set of paths P with $|P| \leq k$ such that Φ is unsatisfiable by Theorem 1. □

4 Encoding of Bounded Unsatisfiability in QBF

We give an encoding of the k-bounded unsatisfiability problem to QBF for a fixed bound k. Before presenting the general encoding, we show the basic steps on formula (2) $\forall x_1, x_2.\, \exists_{\{x_1\}}\, y_1.\, \exists_{\{x_2\}}\, y_2.\, \varphi$. The formula is unsatisfiable iff for all candidate models, there exists a path that violates the matrix φ. Instead of expanding all four paths, we restrict the binary decision tree on two paths (but do not choose which one) and encode the search for the paths as QBF formula

$$\exists x_1^1, x_1^2, x_2^1, x_2^2.\, \forall y_1^1, y_1^2.\, \forall y_2^1, y_2^2.\, \neg\varphi^1 \vee \neg\varphi^2 \ , \tag{4}$$

that asserts that either path violates φ. This, however, does not accurately represent the incomparable dependencies of y_1 and y_2. For the assignment depicted in Fig. 2(d), where only x_1 has a different assignment on the two paths, y_2^1 and y_2^2 can have different assignment as well, despite the fact that y_2 does not depend on x_1. To fix this inaccuracy, we introduce a *consistency condition* that ensures the restricted choices across multiple paths. For example, the consistency condition for y_2 in (4) is $(y_2^1 \leftrightarrow y_2^2) \vee (x_2^1 \not\leftrightarrow x_2^2)$, i.e., either the assignment of y_2 is equal on both paths, or the assignment of the dependency x_2 is different on both paths. In the following, we describe the general encoding.

We build a QBF formula Ψ that encodes the k-bounded unsatisfiability problem, i.e., for a given bound k, the satisfaction of Ψ implies that Φ is unsatisfiable. In the encoding, we introduce k copies of the existential and universal variables in the DQBF formula Φ. Moreover, we specify a consistency condition that enforces that the universal variables can only act according to the assignment of the dependencies given by the support sets.

$$\begin{aligned} \mathrm{bunsat}(\Phi, k) \coloneqq\ & \exists x_1^1, \ldots, x_m^1, x_1^2, \ldots, x_m^2, \ldots, x_m^k. \\ & \forall y_1^1, \ldots, y_n^1, y_1^2, \ldots, y_n^2, \ldots, y_n^k. \\ & \mathrm{consistent}(\{y_1, \ldots, y_n\}, k) \rightarrow \bigvee_{1 \leq i \leq k} \neg\varphi^k \ , \end{aligned} \tag{5}$$

where φ^k denotes the formula φ for which every variable v is replaced by v^k. The consistency condition is given by the formula

$$\mathrm{consistent}(Y, k) \coloneqq \bigwedge_{y \in Y} \ \bigwedge_{(i,j) \in \{1, \ldots, k\}^2} \Big((y^i \leftrightarrow y^j) \vee (\bigvee_{x \in H_y} x^i \not\leftrightarrow x^j) \Big) \ . \tag{6}$$

Lemma 3. *Given a DQBF formula Φ. For a bound $k \leq 2^{|\mathcal{X}|}$ and a set of paths $P \subseteq 2^{\mathcal{X}}$ with $|P| = k$. A binary decision tree $\mathcal{P}$ over paths P is a candidate partial model iff $\mathcal{P}$ satisfies consistent$(\mathcal{Y})$.*

Proof. Assume that $\mathcal{P}$ is a candidate partial model. Pick an arbitrary existential variable $y \in \mathcal{Y}$ with support set $H_y \subseteq \mathcal{X}$. Pick two different paths P_i and P_j from P (if $|P| = 1$, consistent$(\mathcal{Y})$ is immediately satisfied). To be consistent, the valuation of y on both paths must be equal, or $P_i \cap H_y \neq P_j \cap H_y$. Hence consistent$(\mathcal{Y})$ is satisfied.

Let $\mathcal{P}$ be an binary decision tree over P such that the assignments of the existential variables $\mathcal{Y}$ satisfy consistent$(\mathcal{Y})$. Let $y \in \mathcal{Y}$ be an arbitrary existential variable. Pick any combination $(i, j) \in \{1, \ldots, k\}^2$ of paths. By the satisfaction of consistent$(\mathcal{Y})$, it follows that either the valuation of y is equal on both paths, or the valuation of a variable from the support set H_y is different on the two paths. Thus, $\mathcal{P}$ is a candidate partial model.

Theorem 4. *A DQBF formula Φ is unsatisfiable iff there exists a bound $k \geq 1$ such that the QBF formula bunsat(Φ, k) is satisfiable.*

Proof. Assume Φ is unsatisfiable, i.e., for all candidate models, there exists a path that violates the matrix. Choose a bound k of $2^{|\mathcal{X}|}$, i.e., we choose to expand the whole decision tree. In this case, the consistency condition models exactly the dependencies of the existential variables (Theorem 3). Thus, bunsat(Φ, k) considers every candidate model and from the unsatisfiability of Φ follows that for each such candidate, there exists a path that satisfies $\neg\varphi$, hence, bunsat(Φ, k) is satisfiable.

Assume there exists a $k \geq 1$ such that bunsat(Φ, k) is satisfiable. We choose the paths $P \subseteq 2^{\mathcal{X}}$ from the assignment of the existential variables in bunsat(Φ, k). By Theorem 3, the consistency condition allows for all candidate partial models and, for each candidate, one of the paths satisfies $\neg\varphi$. Hence, it follows that Φ is k-bounded unsatisfiable and, by Theorem 2, that Φ is unsatisfiable. □

Proposition 5. *Let Φ be a DQBF formula. For $k \geq 1$, the QBF formula bunsat(Φ, k) has $k \cdot |\mathcal{X}|$ existential and $k \cdot |\mathcal{Y}|$ universal variables, respectively, and the matrix is of size $\mathcal{O}(|\mathcal{Y}| \cdot k^2 \cdot \max_{y \in \mathcal{Y}} |H_y| + k \cdot |\varphi|)$.*

Proof. Omitting symmetric cases in the consistency condition gives us $\binom{k}{2}$ different sets $(i, j) \in \{1, \ldots, k\}^2$. □

Combining with QBF Abstraction. One critical observation in the QBF encoding in (5) is that many paths are not needed for proving the unsatisfiability, but for enforcing consistent labels in the partial model. In (5), the universal variables depend on all existential variables which corresponds to the weakest QBF approximation of Φ. By using a stronger QBF abstraction, the QBF formula itself takes care for a part of the consistency condition, which can decrease the bound needed to refute a DQBF formula.

Example. Consider again the PEC example from Fig. 1. As we have seen, there exist two strongest QBF approximations, but both are satisfiable due to overapproximation. However, we prove unsatisfiability by using a strongest QBF abstraction together with a bound of two: The formula

$$\exists x_1^1, x_1^2.\, \forall y_1^1, y_1^2.\, \exists x_2^1, x_2^2.\, \forall y_2^1, y_2^2.\, ((y_2^1 \leftrightarrow y_2^2) \vee (x_1^1 \not\leftrightarrow x_1^2)) \rightarrow (\neg\varphi^1 \vee \neg\varphi^2) \quad (7)$$

is satisfiable (choose arbitrary assignment α with $\alpha(x_1^1) \neq \alpha(x_1^2)$ and $\alpha(x_2^1) = \alpha(x_2^2)$). As the assignment of y_2 must be the same on both paths (otherwise it violates the consistency condition), it holds that either $\alpha(y_2) \neq \alpha(x_1^1)$ or $\alpha(y_2) \neq \alpha(x_1^2)$, hence the matrix is violated on either path.

5 Experimental Results

We have implemented our new method as a prototype that uses the bounded unsatisfiability method together with a strongest QBF abstraction. In this section, we report on experiments carried out on a 2.6 GHz Opteron system. The QBF instances generated by our reduction are solved using a combination of the QBF preprocessor Bloqqer [4] in version 031 and the QBF solver DepQBF [13] in version 2.0. As a base of comparison, we have also implemented an expansion-based DQBF solver using the BDD library CUDD in version 2.4.2[2].

Table 1 shows the performance of our solver on a number of PEC benchmarks, including the arithmetic circuits *multiplier* (4-bit) and *adder* (32-bit), a 32-bit *lookahead* arbiter implementation, and a 32-bit *multiplexer* [17]. The PEC instances are created as follows: Starting with a circuit, we exchange a variable number of gates by black boxes and use one copy of the original circuit as the specification. Random faults are inserted by replacing exactly one gate with a different gate. With only one exception (instances with more than 7 black boxes of the *multiplier* instances), more than 94% of the instances were solved correctly with bound two, while the number of correctly solved instances by the QBF abstraction drops as low as 84.4%. The running times in Table 1 are given relative to the running time of the expansion-based solver. Our solver outperforms the expansion-based solver significantly, especially on benchmarks with a large number of black boxes. For example, with 9 black boxes, the difference ranges from 37% faster (*adder*) to more than 5 times faster (*lookahead*).

Table 1 indicates that an increase of the bound from 1, which corresponds to the plain QBF approximation, to 2 already results in a significant improvement of accuracy. Table 2 analyzes the impact of the bound on the approximation quality in more detail. Here, the benchmark is a circuit family from [10], depicted for two black boxes in Fig. 3. The circuit uses the XOR of the inputs as specification and a random Boolean function as implementation, where black boxes with pairwise different dependencies serve as inputs to this function. For two black boxes, we created all $65536 = 2^{16}$ instances of Boolean functions with four inputs. For more than two, we selected a random subset of 50000 instances.

Table 2 shows an interesting correlation between the plain QBF approximation and the bounded-path method: The less effective the plain approximation,

[2] The source code of the benchmarks and tools are available at `http://react.uni-saarland.de/tools/bunsat/`

Table 1. Results of the bounded unsatisfiability method on PEC examples

circuit	BBs	unsat.	bound 1	/ 2	time	circuit	BBs	unsat.	bound 1	/ 2	time
multiplier	1	950	100%	100%	27.5%	multiplexer	1	931	100%	100%	57.1%
	3	927	97.6%	100%	22.2%		3	908	97.9%	99.8%	48.0%
	5	924	87.6%	99.6%	17.9%		5	906	95.7%	98.5%	41.9%
	7	912	67.9%	95.9%	13.8%		7	896	92.5%	97.0%	35.5%
	9	870	30.9%	76.2%	16.2%		9	889	88.9%	94.7%	28.9%
adder	1	962	100%	100%	10.8%	lookahead	1	999	100%	100%	4.5%
	3	959	100%	100%	9.0%		3	997	98.2%	100%	3.4%
	5	959	99.9%	100%	8.9%		5	996	97.1%	100%	3.3%
	7	951	99.5%	100%	7.0%		7	996	94.2%	99.9%	0.7%
	9	957	98.5%	99.9%	6.8%		9	986	84.4%	99.1%	0.8%

The table shows the approx. ratio of the bounded-path prototype using a bound ≤ 2 and the median running time relative to the expansion solver (timeout after 5min per instance). For every number of black boxes, we generated 1000 random instances.

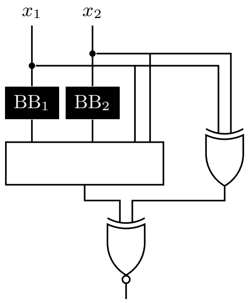


Fig. 3. XOR template

Table 2. Result of the XOR random function example

BBs	total	sat.	unsat.	bounded unsatisfiable			
				1	2	3	> 3
2	65536	32377	33159	22687	10472	0	0
			100%	68.4%	31.6%	0%	0%
3	50000	9273	40727	11257	26169	3115	186
			100%	27.6%	64.3%	7.6%	0.5%
4	50000	190	49810	5002	43781	1015	12
			100%	10.0%	87.9%	2.0%	$< 0.1\%$

The table shows the approximation quality of the bounded-path prototype with respect to the number of black boxes.

the more effective the bounded-path method. With an increasing number of black boxes, the number of solved instances by the plain QBF approximation (bound 1) decreases by more than a half with every black box. At the same time, the relative number of instances that are solved with a bound of at most two is always larger than 90%. With a bound of at most three, nearly all unsatisfiable instances are detected (> 99%). While the QBF approximation alone thus does not lead to satisfactory results, a comparatively small bound suffices to solve almost all instances.

6 Conclusion

We have presented a method for DQBF refutation that significantly outperforms the previous expansion-based approach on PEC benchmarks. The new method is based on an improved approximation of the DQBF formula within QBF, based on evaluating the formula on multiple paths and expressing dependency constraints as consistency conditions. Our experiments show that considering multiple paths significantly improves the accuracy, especially with an increasing number of black boxes. Compared to an expansion-based solver, the running time of our prototype implementation scales better with the number of black boxes. In the future, we plan to extend the method to sequential circuits and to integrate QBF certification [2] in order to identify faulty components.

Acknowledgements. This work was partially supported by the German Research Foundation (DFG) as part of SFB/TR 14 AVACS. We thank Christoph Scholl for comments on an earlier version of this paper.

References

1. Balabanov, V., Chiang, H.J.K., Jiang, J.H.R.: Henkin quantifiers and Boolean formulae. In: Cimatti, A., Sebastiani, R. (eds.) SAT. LNCS, vol. 7317, pp. 129–142. Springer (2012)
2. Balabanov, V., Jiang, J.H.R.: Unified QBF certification and its applications. Formal Methods in System Design 41(1), 45–65 (2012)
3. Benedetti, M.: Evaluating QBFs via symbolic skolemization. In: Baader, F., Voronkov, A. (eds.) LPAR. LNCS, vol. 3452, pp. 285–300. Springer (2004)
4. Biere, A., Lonsing, F., Seidl, M.: Blocked clause elimination for QBF. In: CADE. LNCS, vol. 6803, pp. 101–115. Springer (2011)
5. Bubeck, U., Büning, H.K.: Dependency quantified horn formulas: Models and complexity. In: Biere, A., Gomes, C.P. (eds.) SAT. LNCS, vol. 4121, pp. 198–211. Springer (2006)
6. Finkbeiner, B., Schewe, S.: Uniform distributed synthesis. In: LICS. pp. 321–330. IEEE Computer Society (2005)
7. Finkbeiner, B., Tentrup, L.: Detecting unrealizable specifications of distributed systems. In: Ábrahám, E., Havelund, K. (eds.) TACAS. LNCS, vol. 8413, pp. 78–92. Springer (2014)
8. Finkbeiner, B., Tentrup, L.: Fast DQBF refutation. In: Sinz, C., Egly, U. (eds.) SAT. LNCS, vol. 8561, pp. 243–251. Springer (2014)
9. Fröhlich, A., Kovásznai, G., Biere, A.: A DPLL algorithm for solving DQBF. Proc. POS 12 (2012)
10. Gitina, K., Reimer, S., Sauer, M., Wimmer, R., Scholl, C., Becker, B.: Equivalence checking of partial designs using dependency quantified Boolean formulae. In: ICCD. pp. 396–403. IEEE (2013)
11. Janota, M., Marques-Silva, J.: On propositional QBF expansions and Q-resolution. In: Järvisalo, M., Gelder, A.V. (eds.) SAT. LNCS, vol. 7962, pp. 67–82. Springer (2013)
12. Kupferman, O., Vardi, M.Y.: Synthesis with incomplete information. In: ICTL (1997)
13. Lonsing, F., Biere, A.: DepQBF: A dependency-aware QBF solver. JSAT 7(2-3), 71–76 (2010)
14. Nopper, T., Scholl, C., Becker, B.: Computation of minimal counterexamples by using black box techniques and symbolic methods. In: Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on. pp. 273–280 (Nov 2007)

15. Peterson, G.L., Reif, J.: Multiple-person alternation. In: Foundations of Computer Science, 1979., 20th Annual Symposium on. pp. 348–363 (Oct 1979)
16. Scholl, C., Becker, B.: Checking equivalence for partial implementations. In: Proceedings of the 38th Annual Design Automation Conference. pp. 238–243. DAC '01, ACM, New York, NY, USA (2001)
17. William J. Dally, R.C.H.: Digital Design, A Systems Approach. Cambridge University Press (2012)