A Resolution-style Proof System for DQBF

Markus N. Rabe

University of California, Berkeley
rabe@berkeley.edu

Abstract. This paper presents a sound and complete proof system for Dependency Quantified Boolean Formulas (DQBF) using resolution, universal reduction, and a new proof rule that we call fork extension.

1 Introduction

Dependency quantified Boolean formulas (DQBF) extend quantified Boolean formulas (QBF) by *Henkin quantifiers* $\exists x : Y.\ \varphi$, which bind a variable x and also specify a *dependency set* Y of universal variables that x may depend on [17, 22]. Henkin quantifiers allow us to succinctly express the existence of functions satisfying quantified constraints (see Section 3). For example, we can formulate the existence of a function $add : \mathbb{B}^{32} \times \mathbb{B}^{32} \to \mathbb{B}^{32}$ implementing the axioms of addition in 32-bit addition: $\forall x.\ add(x, 0) = x \ \wedge\ S(x) = x+1 \ \wedge\ \forall x, y.\ add(x, S(y)) = S(add(x, y))$, where $x + 1$ stands for the increment-by-one operation encoded as constraints. To show that the formula is true, we have to find an implementation for add that only depends on two inputs

DQBF enables elegant encodings for interesting applications such as bounded reactive synthesis [9] and partial equivalence checking of circuits [14]. Its elegance and expressiveness make DQBF a potential candidate logic to serve as the interface between algorithms and applications in synthesis, verification, and artificial intelligence that require impractically large encodings in other logics such as QBF and propositional Boolean logic (SAT). Recently, there has been a surge of interest in practical algorithms for DQBF [11–15, 32], but their performance is still unsatisfactory [8]. We argue that this is due to the lack of suitable proof systems.

Resolution is one of the fundamental proof rules for first-order logic [7]. It says that given two clauses $(x \vee a_1 \vee \cdots \vee a_n)$ and $(\neg x \vee b_1 \vee \cdots \vee b_m)$ we can infer the clause $(a_1 \vee \cdots \vee a_n \vee b_1 \vee \cdots \vee b_m)$, which we call the *resolvent*. The best known application of resolution is the conflict analysis step of the conflict-driven clause learning (CDCL) algorithm, which is the basis for modern SAT solvers [26].

While resolution is sufficient to prove or disprove any propositional Boolean formula, we need a second proof rule for QBF. The *universal reduction* rule says that we can delete a literal of a universal variable from a clause, if no variable in the clause may depend on it, i.e. no existential variable in this clause is bound in the scope of the universal variable. Resolution and universal reduction form the

Q-resolution proof system, which plays a similar role for QBF as resolution plays for SAT [6]. Some of the recently popular algorithms for QBF, QCDCL [20, 33], CEGAR [18, 19, 24, 27], and Incremental Determinization [23], can be phrased as variants of Q-Resolution [28].

While resolution is complete for propositional Boolean logic, and Q-resolution is complete for QBF [6], resolution and Q-resolution were shown to be incomplete for DQBF [1, 2]. The only known complete proof systems for DQBF resort to expansion or (similarly) annotating literals with partial instantiations of the universal variables [2]. While expansion is widely used in preprocessors for QBF [4, 32], using expansion as a solving technique is less popular due to its often excessive memory requirements. Instead, most of the popular solvers for SAT and QBF rely on resolution in their reasoning. A resolution-style proof system for DQBF might therefore help to lift efficient algorithms to DQBF.

In this paper, we present a resolution-style proof system for DQBF, called *Fork-Resolution*. It uses resolution, universal reduction, and, to achieve completeness, introduces a new proof rule called *fork extension*. Fork extension is derived from the extension rule from extended resolution [29] and says that, given a clause $(a_1 \vee \cdots \vee a_n \vee b_1 \vee \cdots \vee b_m)$, we can infer the clauses $(x \vee a_1 \vee \cdots \vee a_n)$ and $(\neg x \vee b_1 \vee \cdots \vee b_m)$, where x is a fresh variable. The key insight in this paper is that it is sufficient to apply extension to a specific type of clauses that we call *information forks.*[1]

An information fork is a clause that contains variables with incomparable dependency sets. Consider a DQBF with a single clause $\exists x_1 : \{y_1\}. \exists x_2 : \{y_2\}. (x_1 \vee x_2)$. To satisfy the clause with a Skolem function, we have to decide in which cases the clause is satisfied by x_1 and in which cases x_2 is responsible for satisfying the clause. As x_1 and x_2 have disjoint dependency sets, they cannot depend on the value of the other variable. So the only way to make sure the clause is always satisfied is to require that one of x_1 and x_2 is *always* responsible for satisfying the clause. In other words, there must be a constant that indicates which side of the clause is responsible for satisfying the clause. With the fork extension rule, we introduce a variable x that represents this constant. We split the clause into two clauses $(x_1 \vee x)$ and $(\neg x \vee x_2)$ and introduce x with the empty dependency set $(\exists x : \emptyset)$. In general, the decision of which side of the clause has to satisfy the clause must be decided based only on the information that is available to both literals, i.e. the intersection of their dependency sets.

This paper is structured as follows. We introduce basic notation and definitions in Section 2 and provide an encoding of functions in DQBF in Section 3. In Section 4, we introduce the Fork-Resolution proof system and prove its soundness and completeness. Section 5 demonstrates that Fork-Resolution can disprove a formula for which Q-resolution is incomplete. We briefly discuss how Fork-Resolution is separated exponentially from other proof systems for DQBF in Section 6. Section 7 discusses related work.

[1] The concept is loosely connected to information forks in reactive synthesis of distributed systems [10].

2 Dependency Quantified Boolean Formulas

Dependency quantified Boolean formulas (DQBFs) over a set of variables V are generated by the following grammar with starting symbol ψ:

$$\begin{aligned}\psi &\coloneqq \exists\, v : \{v, \ldots, v\}.\, \psi \ \mid\ \varphi \\ \varphi &\coloneqq 0 \ \mid\ 1 \ \mid\ v \ \mid\ \neg\varphi \ \mid\ \varphi \vee \varphi \ \mid\ \varphi \wedge \varphi \ ,\end{aligned}$$

where $v \in V$, and $\{v, \ldots, v\}$ stands for a finite subset of V. Quantifiers in DQBF $\exists x : \{y_1, \ldots, y_n\}$ specify a single *existential variable* x and also a finite set of *universal variables* $\{y_1, \ldots, y_n\}$ that x may depend on. We call $\{y_1, \ldots, y_n\}$ also the *dependency set* of x, denoted $dep(x)$. The variables that occur in dependency sets are the *universal variables*. All variables that are not universal are called *existential variables*. We say that a quantifier $\exists\, x : Y.\, \varphi$ *binds* the existential variable x in its subformula φ. To simplify the discussion, we assume that existential variables are bound at most once and that universal variables are never bound. A DQBF is *closed* if it only contains universal variables and bound existential variables.

In the following we define the semantics of DQBF for which we assume that the reader is familiar with the natural semantics of propositional Boolean formulas. An *assignment* $\mathbf{y}$ to a set of variables Y is a function $\mathbf{y} : Y \to \mathbb{B}$ that maps each variable $y \in Y$ to either 1 or 0. Given a set of variables Y, we denote the set of assignments to Y with 2^Y. Given a propositional formula φ over variables X and an assignment $\mathbf{x}'$ for $X' \subseteq X$, we define $\varphi(\mathbf{x}')$ to be the formula obtained by replacing the variables X' by their truth value in $\mathbf{x}'$. By $\varphi(\mathbf{x}', \mathbf{x}'')$ we denote the replacement by multiple assignments for disjoint sets $X', X'' \subseteq X$. A *Skolem function* f_x maps assignments to $dep(x)$ to $\mathbb{B}$, interpreted as assignments to x. We define the truth of a DQBF φ with existential variables $X = \{x_1, \ldots, x_n\}$ and universal variables Y as the existence of Skolem functions $f_X = \{f_{x_1}, \ldots, f_{x_n}\}$, such that $\varphi(\mathbf{y}, f_X(\mathbf{y}))$ is satisfiable for every assignment $\mathbf{y}$ to Y.

A *literal* l is either a variable $x \in X$, or its negation $\neg x$. We use $\bar{l}$ to denote the *complement* operation that gives the literal with the negated value of l and we define $var(l)$ to be the variable that the literal contains. Given a set of literals $\{l_1, \ldots, l_n\}$, their disjunction $(l_1 \vee \ldots \vee l_n)$ is called a *clause*. As usual in the community, we use set notation for the manipulation of clauses. A clause C is called *tautological* if it contains a literal l and its complement $\bar{l}$. We use $vars(C)$ to denote the variables occurring in C.

A propositional formula is in conjunctive normal form (CNF), if it is a conjunction of clauses. A closed DQBF is in CNF if its propositional subformula is in CNF. Every DQBF φ can be transformed into a closed DQBF in CNF with size $O(|\varphi|)$, preserving satisfiability. In the rest of this work we assume DQBFs to be in CNF.

We extend the notation of dependency sets to negated variables (and thus literals), $dep(\neg v) = dep(v)$. Applied to clauses, $dep(C)$ denotes the union of the universal variables and the dependencies of the existential variables in C.

3 On Quantification over Functions

Our interest in DQBF is rooted in the fact that we can encode existential quantification over functions. In the following, we present a transformation that allows us to eliminate functions from a formula. For the (first-order) theory of equality and uninterpreted functions, there is a similar transformation, called *Ackermannization* [5]. Unlike Ackermannization, the transformation below is linear in the size of the formula.

Consider a formula $\varphi = \exists f : \mathbb{B}^N \to \mathbb{B}.\ \psi$. We can turn that into an equivalent formula φ' that does not use f. Let f have k function applications $f(e_{1,1}, \dots, e_{1,N}), \dots, f(e_{k,1}, \dots, e_{k,N})$ in ψ, where $e_{i,j}$ are terms. We introduce fresh variables $f_1, \dots, f_k$ and dependency sets $X_1, \dots, X_k$, each $X_i = \{x_{i,1}, \dots, x_{i,N}\}$ consisting of N fresh variables, for the function applications. We then define φ' as

$$\begin{aligned}
&\exists f_1 : X_1.\ \dots\ \exists f_k : X_k.\\
&\qquad \textstyle\bigwedge_{i=2}^{k} \Big(\big(\bigwedge_{j=1}^{N} x_{1,j} = x_{i,j} \big) \implies f_1 = f_i \Big) \land \\
&\qquad \Big[\big(\textstyle\bigwedge_{i=1}^{k} \bigwedge_{j=1}^{N} e_{i,j} = x_{i,j} \big) \implies \psi[\{ f(e_{i,1}, \dots, e_{i,N}) / f_i \mid 1 \le i \le k \}] \Big]
\end{aligned}$$

The conjunct $\bigwedge_{i=2}^{k} \Big(\big(\bigwedge_{j=1}^{N} x_{1,j} = x_{i,j} \big) \implies f_1 = f_i \Big)$ requires that variables $f_1, \dots, f_k$ represent the same function. If f_1 and f_i have different values for some "input" x^*, we set $x_1 = x_i = x^*$ and see that $f_1 = f_i$ is violated. Transitivity of equality allows us to avoid encoding the pairwise equality explicitly.

The second conjunct of the formula states that if $f_1, \dots, f_k$ represent the outputs of this function for the k function applications, the formula where we replace the function applications by the f_i has to hold.

The constraints can be transformed into a CNF of linear size with the Tseitin transformation [29].

If φ contains multiple functions they can be replaced independently, by first introducing new variables such that no argument of a function contains a function application. Then replacing one function does not affect the function applications of other functions. So, the linear increase in size does not multiply over multiple applications of the function elimination and we only get a linear increase in the size of the formula overall.

4 The Fork-Resolution Proof System

The Fork-Resolution proof system consists of the following three rules:

Resolution [7]:

$$\frac{C_1 \cup \{l\} \qquad C_2 \cup \{\bar{l}\}}{C_1 \cup C_2} \quad (\mathsf{Res})$$

We call the clause $(C_1 \setminus \{l\}) \cup (C_2 \setminus \{\bar{l}\})$ that is obtained by the resolution rule the *resolvent* of C_1 and C_2 and we call $var(l)$ the *pivot*. The resolution rule is only applied when the resolvent is not a tautology, i.e. does not contain two complementary literals.

Universal Reduction [6]:

$$\frac{C \quad var(l) \notin dep(C \setminus \{l\}) \quad \bar{l} \notin C \quad var(l) \text{ is universal}}{C \setminus \{l\}} \quad (\forall\mathsf{Red})$$

Fork Extension:

$$\frac{C_1 \cup C_2 \qquad dep(C_1) \not\subseteq dep(C_2) \qquad dep(C_1) \not\supseteq dep(C_2) \qquad x \text{ is fresh}}{\exists x : dep(C_1) \cap dep(C_2).\ C_1 \cup \{x\} \ \wedge \ C_2 \cup \{\neg x\}} (\mathsf{FEx})$$

The fork extension rule introduces a new quantified variable x to split a clause into two parts. The dependency set of x is defined as the intersection of $dep(C_1)$ and $dep(C_2)$. We only apply fork extension to clauses $C_1 \cup C_2$ that consist of two parts have *incomparable dependency sets* ($dep(C_1) \not\subseteq dep(C_2)$ and $dep(C_1) \not\supseteq dep(C_2)$). We call such clauses *information forks*.

The idea is that for each assignment to the universal variables one of the two parts C_1 and C_2 is “responsible” for satisfying the original clause $C_1 \cup C_2$. The variables in C_1 and C_2, however, may have different dependency sets, and so they must coordinate their responsibility only based on the information that is common to them.

A recurring theme in the proofs to follow is that we consider how variables can be defined only in terms of the variables that they share clauses with. The following definitions help us to zoom in on the neighborhood of a variable.

Definition 1 (Projection of assignments). *Given two sets of variables X and X' with $X' \subseteq X$ and an assignment $\mathbf{x}$ to X. We call an assignment $\mathbf{x}'$ to X' the* projection *of $\mathbf{x}$ to X', if $\mathbf{x}'(x) = \mathbf{x}(x)$ for all $x \in X'$. We denote the projection of $\mathbf{x}$ to X' with $\mathbf{x}|_{X'}$.*

Definition 2 (Projection of Skolem functions). *Let φ be a true DQBF in PCNF, let C be a clause in φ, and let $f : 2^Y \to 2^X$ be Skolem function. We call a function $f_C : 2^{dep(C)} \to 2^{vars(C)}$ the* projection *of f to C, if for all assignments $\mathbf{y}$ to Y and variables x in C it holds $f(\mathbf{y})(x) = f_C(\mathbf{d})(x)$, where $\mathbf{d}$ is the projection $\mathbf{d}$ of $\mathbf{y}$ to $dep(C)$. We denote the projection of f to C with $f|_C$.*

In the following lemma we recall the soundness of resolution and universal reduction and we prove the soundness of the fork extension rule.

Lemma 1. *The fork-resolution proof system for DQBF is sound.*

Proof. We show soundness for each proof rule individually.

Res. The resolution rule is as usual, with the exception that it is possible to apply resolution to tautological clauses, in which case the resolvent is subsumed by of $C_1 \cup \{l\}$ or $C_2 \cup \{\bar{l}\}$. For all other clauses soundness follows from the soundness of QU-resolution for DQBF [2,30].

∀Red. (Similarly in [1]) Let l be a literal of a universal variable x and let C be a clause with $l \in C$ and $\bar{l} \notin C$ and let $x \notin dep(C)$. If the DQBF is true, there is a function f_C mapping assignments $\boldsymbol{d}$ to $dep(C)$ to values $f_C(\boldsymbol{d})$ for the existential variables in C, such that clause C is satisfied. In particular, C is satisfied for each assignment to the universals setting l to 0 because $\bar{l} \notin C$. Since f_C is independent of x, we know that after removing l from C, the clause is still satisfied.

FEx. Let $\mathcal{Q}.\,\varphi$ be a true DQBF in PCNF with quantifier prefix $\mathcal{Q}$, universal variables Y, and existential variables X, let $C_1 \cup C_2$ be a clause in φ, and let x be a fresh variable. We show that $\mathcal{Q}.\,\exists x : dep(C_1) \cap dep(C_2).\ \varphi \wedge C_1 \cup \{x\} \wedge C_2 \cup \{\neg x\}$ is true by constructing a Skolem function $f_x : dep(C_1) \cap dep(C_2) \to 2^{\{x\}}$ for x that together with f_C, satisfies the (new) constraints.

Consider an arbitrary Skolem function f for $\mathcal{Q}.\,\varphi$ and let $\mathbf{y}$ be an assignment to Y. We fix $f_x(\mathbf{y}|_{dep(x)})$ to be 1, if C_1 is not satisfied by $\mathbf{y}$ or $f(\mathbf{y})$, and we fix $f_x(\mathbf{y}|_{dep(x)})$ to be 0, if C_2 is not satisfied by $\mathbf{y}$ or $f(\mathbf{y})$. In case both C_1 and C_2 are satisfied, we (arbitrarily) fix x to be 1. The case that neither C_1 and C_2 are satisfied contradicts the fact that f is a Skolem function.

Let us assume that for the given $\mathbf{y}$ and Skolem function f one part of the clause is violated, which we assume w.l.o.g. to be C_1. To prove the correctness of f_x we have to show that there cannot exist a second assignment $\mathbf{y}'$ to Y that agrees with $\mathbf{y}$ on the variables $dep(C_1) \cap dep(C_2)$ but violates C_2 instead. Assuming that there is such an assignment $\mathbf{y}'$, we can construct an assignment $\mathbf{y}''$ to Y that agrees with $\mathbf{y}$ on *all but* the variables of $dep(C_2)$, and agrees with $\mathbf{y}'$ on the variables of $dep(C_2)$. Since $\mathbf{y}$ and $\mathbf{y}''$ agree on $dep(C_1) \cap dep(C_2)$, assignment $\mathbf{y}''$ violates both C_1 and C_2, which contradicts the satisfaction of $C_1 \cup C_2$. The soundness of the FEx rule follows by way of contradiction. □

The central insight of this paper is that when we eliminate information forks from a DQBF, we can eliminate existential variables using resolution like in QBF. Eliminating variables may introduce new information forks, so in general we have to alternate fork extension and resolution.

The following lemmas show that after fork extension and variable elimination the clauses to which they were applied become obsolete and can be removed. Theorem 1 then shows how the lemmas can be put into action to obtain a sound and complete algorithm.

Lemma 2. *Let $\mathcal{Q}\,.\,\varphi \wedge C_1 \cup C_2$ be a DQBF in PCNF with quantifier prefix $\mathcal{Q}$ and let x be a fresh variable. Then $\mathcal{Q}.\ \varphi \wedge C_1 \cup C_2$ and $\mathcal{Q}.\,\exists y : dep(C_1) \cap dep(C_2).\ \varphi \wedge C_1 \cup \{x\} \wedge C_2 \cup \{\neg x\}$ are equivalent.*

Proof. Soundness of the FEx rule was proven in Lemma 1. The other direction follows from the soundness of resolution and the soundness of removing constraints and unused variables. □

Variable elimination by resolution is a well known technique for SAT and QBF [3, 6]. Given a propositional formula φ in CNF over variables X we can eliminate a variable $x \in X$ by adding all resolvents for pivot x and removing all clauses with literals of x. We denote the formula obtained through variable elimination $\mathsf{elim}(x, \varphi)$.

Proposition 1 (Variable elimination [3,6]). *Given a propositional formula φ over variables X and a variable $x \in X$. Then for all assignments $\mathbf{x}$ to $X \setminus \{x\}$ we have that $\mathsf{elim}(x, \varphi)(\mathbf{x})$ if, and only if, $\exists x.\, \varphi(\mathbf{x})$.*

To lift variable elimination to DQBF we need a stronger property of resolution. Instead of expressing the notion of equivalence between φ and $\mathsf{elim}(x, \varphi)$ via existential quantification, we provide a function with minimal dependencies resolving the quantification.

Lemma 3 (Variable elimination with dependencies). *Given a propositional formula φ in CNF over variables X and a variable $x \in X$. Let $X' \subseteq X$ be the set of variables occurring together with x in some clause in φ but not x itself. Then there is a function f_x mapping assignments to X' to assignments to x such that for all assignments $\mathbf{x}$ to $X \setminus \{x\}$ we have that $\mathsf{elim}(x, \varphi)(\mathbf{x})$ if, and only if, $\varphi(\mathbf{x}, f_x(\mathbf{x}|_{X'}))$.*

Proof. Let φ' be the clauses that contain a literal of x. For every assignment $\mathbf{x}'$ to X', we define $f_x(\mathbf{x}')$ as the value that satisfies φ', if one exists. If there is no such value, we arbitrarily fix $f_x(\mathbf{x}')$ to be 1.

"$\Longleftarrow$": Whenever φ is true, also $\mathsf{elim}(x, \varphi)$ must be true, because of the soundness of the elimination rule and because removing clauses only makes it easier to satisfy the formula.

"$\Longrightarrow$": Let $\mathbf{x}$ be an assignment to $X \setminus \{x\}$ such that $\varphi(\mathbf{x}, f_x(\mathbf{x}|_{X'}))$ is false. If a clause that does not contain x is violated, then also $\mathsf{elim}(x, \varphi)(\mathbf{x})$ is false, as it contains the same clause. Otherwise, a clause C containing x is violated. By choice of f_x we know that x is chosen to be 1, but also for value 0 not all constraints could be satisfied. Thus there must be a clause C' that is violated if we changed the assignment of x. The resolvent of C and C' is violated for $\mathbf{x}$. □

Next we show that leaf-existentials in DQBFs without information forks can be eliminated through resolution. We call an existential variable x in a DQBF φ a *leaf-existential* if all existentials x' in φ have a smaller or incomparable dependency set than x ($dep(x) \not\subset dep(x')$).

Lemma 4 (Similarly in [31], Thm. 4). *Let $\psi = \exists x_1 : Y_1. \dots \exists x_n : Y_n.\, \varphi$ be a DQBF without information forks. Further let x_1 be a leaf-existential. Then there is an equivalent DQBF $\psi' = \exists x_2 : Y_2. \dots \exists x_n : Y_n.\, \mathsf{elim}(x_1, \varphi)$.*

Proof. The soundness of resolution immediately gives us that $\psi \implies \psi'$. We prove $\psi' \implies \psi$ in the following:

Let φ'' be the conjunction of all resolvents with pivot x_1. Let ψ' be true and let $f_{x_2}, \dots, f_{x_k}$ be the Skolem functions proving ψ' to be true. Further, let X'

be the variables occurring in φ'. By Lemma 3 we obtain a function f_{x_1} for x_1 mapping the assignments to $X' \setminus \{x_1\}$ to $\mathbb{B}$ such that $\varphi'[x_1/f_{x_1}(Y')]$ and φ'' are equivalent. We can transform f_{x_1} to a Skolem function for x_1 with domain Y_1 by inlining the definitions for the other Skolem functions, as $\mathit{dep}(X' \setminus \{x_1\}) \subseteq \mathit{dep}(x_1)$ due to the lack of information-forks and x_1 being a leaf-existential. □

Theorem 1. *A DQBF is false if, and only if, we can derive an empty clause in the Fork-Resolution proof system.*

Proof. We provide an round-based algorithm based on the proof rules. Each round consists of two steps. The first step is to eliminate all information forks using Lemma 2. The second step is to eliminate a leaf-existential with a dependency set of maximal size using Lemma 4. The proof system does not allow us to remove clauses as suggested in Lemmas 2 and 4, but additional clauses can never impede the derivation of the empty clause.

The termination guarantee for the (Q-)resolution proof systems for SAT and QBF can be easily established, since variable elimination reduces the number of variables in the formula. However, Fork-Resolution introduces variables with the fork extension rule, so its termination guarantee is based on the dependency sets that occur in the formula: Each round reduces either the number of existentials with a maximally sized dependency set or the size of the maximally sized dependency set. The lexicographic ordering of the two provides the termination relation. □

5 Example

We demonstrate the proof system along an example. We use a shortened version of the example used to show incompleteness of Q-resolution for DQBF [1]. The formula states $\exists y_1 : \{x_1\}. \exists y_2 : \{x_2\}. (x_1 \wedge x_2) \leftrightarrow (y_1 = y_2)$. The formula states that the existential variables y_1 and y_2 have to be equal iff x_1 and x_2 are true. But y_1 and y_2 can each only see one of x_1 and x_2 and thus cannot coordinate to satisfy the constraint. That is, the formula is false. The propositional part of the formula can be represented in CNF as follows:

$$y_1 \vee y_2 \vee x_1 \tag{1}$$
$$\neg y_1 \vee \neg y_2 \vee x_1 \tag{2}$$
$$y_1 \vee y_2 \vee x_2 \tag{3}$$
$$\neg y_1 \vee \neg y_2 \vee x_2 \tag{4}$$
$$y_1 \vee \neg y_2 \vee \neg x_1 \vee \neg x_2 \tag{5}$$
$$\neg y_1 \vee y_2 \vee \neg x_1 \vee \neg x_2 \tag{6}$$

Despite the formula being false, we can see with a little effort that all resolvents of the clauses above are tautologies. This demonstrates that Q-resolution for DQBF is incomplete [1].

The Fork-Resolution proof system can disprove the formula as we show in the following. Variables y_1 and y_2 are leaf-existentials, and all six clauses are information forks. Applying FEx to clauses (1) to (6) introduces variables t_1 to t_6 with empty dependencies ($\exists t_1 : \emptyset. \ldots \exists t_6 : \emptyset.$) and yields the following clauses:

$t_1 \vee y_1 \vee x_1$	(1a)	$\neg t_1 \vee y_2$	(1b)
$t_2 \vee \neg y_1 \vee x_1$	(2a)	$\neg t_2 \vee \neg y_2$	(2b)
$t_3 \vee y_1$	(3a)	$\neg t_3 \vee y_2 \vee x_2$	(3b)
$t_4 \vee \neg y_1$	(4a)	$\neg t_4 \vee \neg y_2 \vee x_2$	(4b)
$t_5 \vee y_1 \vee \neg x_1$	(5a)	$\neg t_5 \vee \neg y_2 \vee \neg x_2$	(5b)
$t_6 \vee \neg y_1 \vee \neg x_1$	(6a)	$\neg t_6 \vee y_2 \vee \neg x_2$	(6b)

Next, we derive six resolvents as listed below. Their names indicate the pair of clauses they originate from. (We drop universal variables by universal reduction.)

$t_1 \vee t_4$	$(1a4a)$	$\neg t_1 \vee \neg t_5$	$(1b5b)$
$t_3 \vee t_6$	$(3a6a)$	$\neg t_2 \vee \neg t_6$	$(2b6b)$
$t_4 \vee t_5$	$(4a5a)$	$\neg t_3 \vee \neg t_4$	$(3b4b)$

Resolving clauses $(4a5a)$ and $(1b5b)$ gives us $(\neg t_1 \vee t_4)$, which we resolve with $(1a4a)$ to get t_4. Similarly, we resolve clauses $(3a6a)$ and $(2b6b)$ to get $(\neg t_2 \vee t_3)$, which we resolve with $(t_2 \vee t_3)$ to get t_3. Resolving t_3 and t_4 with $(3b4b)$ derives the empty clause.

6 Separation from other Proof Systems

Our definition of resolution admits universal variables as pivots and thus includes QU-resolution [30]. This already gives us the result that Fork-Resolution has exponentially smaller proofs for some formulas than D-IR-calc and ∀Exp-Res [2]. For the other direction consider a QBF, where the FEx rule cannot be applied and Fork-Resolution coincides with QU-resolution. IR-calc and ∀Exp-Res are exponentially more succinct on some formulas than QU-resolution [2].

The question whether without resolution on universal variables Fork-Resolution proofs can be exponentially more succinct than proofs by D-IR-calc is open.

7 Related Work

Previous (sound and complete) proof systems for DQBF rely on universal expansion or instantiation [2], which is also the basis for DQBF solvers, such as iDQ [13]. Earlier DQBF solvers relied on a search procedure generalizing DPLL [12], on generating refutation proofs of increasing size [11], or on expansion in symbolic datastructures [15].

In QBF, *clausal abstraction* is a technique to split clauses with variables in order to separate the quantifiers [18, 24]. Clausal abstraction is applied when the dependency sets of two parts of a clause are different but ordered, while FEx is applied when they are incomparable.

The Bernays-Schönfinkel class of first-order logic, also called the effectively propositional fragment (EPR), and DQBF are related in that they share the same complexity class NEXPTIME [21]. Translations between the two logics are known. Like iDQ, EPR solvers rely on instantiation and a number of proof rules that are quite different from SAT solvers and QBF solvers. A recent attempt to run an EPR solver in the QBF competition suggested that EPR is not competitive on the type of problems in the QBF libraries [16]. Hence, by lifting the resolution-based solver technologies from SAT and QBF to DQBF, Fork-Resolution may enable solving new classes of problems.

8 Conclusion

The beauty of DQBF lies it its unified representation of propositional variables, quantified variables, and functions. It offers incredible succinctness but presented a big challenge for practical solver development. In a recent work solvers for SAT, QBF, and DQBF were compared on encodings of the same problem [9]. The experiment showed that current DQBF solvers on their compact encodings fall far behind QBF solvers on the longer QBF encoding, and even SAT solver on the much longer propositional encoding of the same problem. This suggests that the current solving technologies for DQBF are *unable to leverage the succinctness* of DQBF encodings. Arguably, instantiation and expansion—the currently used techniques for solving DQBF—are aimed at making a DQBF more QBF-like. So it is maybe not that surprising that directly encoding the problem in QBF outperforms the DQBF approach.

In this paper, we presented an alternative approach to DQBF. The Fork-Resolution proof system is sound and complete and lifts resolution to *reasoning about functions* without instantiating their inputs. This opens new paths in the development of practical algorithms for DQBF. It remains to show that efficient algorithms can be built based on Fork-Resolution.

Acknowledgements. The author expresses his gratitude to Armin Biere, Daniel J. Fremont, Martina Seidl, Sanjit A. Seshia, Martin Suda, Leander Tentrup, and Ralf Wimmer for supportive comments and detailed discussions on this work.

This work was supported in part by NSF grants CCF-1139138, CNS-1528108, and CNS-1646208, and by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

References

1. Valeriy Balabanov, Hui-Ju Katherine Chiang, and Jie-Hong R Jiang. Henkin quantifiers and Boolean formulae: A certification perspective of DQBF. *Theoretical Computer Science*, 523:86–100, 2014.
2. Olaf Beyersdorff, Leroy Chew, Renate A Schmidt, and Martin Suda. Lifting QBF resolution calculi to DQBF. In *Proceedings of SAT*, pages 490–499. Springer, 2016.
3. Armin Biere. Resolve and expand. In *Proceedings of SAT*, 2004.

4. Armin Biere, Florian Lonsing, and Martina Seidl. Blocked clause elimination for QBF. In *Proceedings of CADE*, pages 101–115, 2011.
5. Roberto Bruttomesso, Alessandro Cimatti, Anders Franzén, Alberto Griggio, Alessandro Santuari, and Roberto Sebastiani. To Ackermann-ize or not to Ackermann-ize? On efficiently handling uninterpreted function symbols in $SMT(\mathcal{EUF} \cup \mathcal{T})$. In *Proceedings of LPAR*, pages 557–571. Springer, 2006.
6. H.K. Buning, M. Karpinski, and A. Flogel. Resolution for quantified boolean formulas. *Information and Computation*, 117(1):12 – 18, 1995.
7. Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM (JACM)*, 7(3):201–215, 1960.
8. Peter Faymonville, Bernd Finkbeiner, Markus N Rabe, and Leander Tentrup. 3 encodings of reactive synthesis. In *Proceedings of QUANTIFY*, pages 20–22, 2015.
9. Peter Faymonville, Bernd Finkbeiner, Markus N. Rabe, and Leander Tentrup. Encodings of bounded synthesis. In *Proceedings of TACAS*, 2017.
10. Bernd Finkbeiner and Sven Schewe. Uniform distributed synthesis. In *Proceedings of LICS*, pages 321–330, Washington, DC, USA, 2005. IEEE Computer Society.
11. Bernd Finkbeiner and Leander Tentrup. Fast DQBF refutation. In *Proceedings of SAT*, pages 243–251, 2014.
12. Andreas Fröhlich, Gergely Kovásznai, and Armin Biere. A DPLL algorithm for solving DQBF. *Proceedings of Pragmatics of SAT*, 2012:2012, 2012.
13. Andreas Fröhlich, Gergely Kovásznai, Armin Biere, and Helmut Veith. iDQ: Instantiation-based DQBF solving. In *Proceedings of Pragmatics of SAT*, pages 103–116, 2014.
14. K. Gitina, S. Reimer, M. Sauer, R. Wimmer, C. Scholl, and B. Becker. Equivalence checking of partial designs using dependency quantified boolean formulae. In *Proceedings of ICCD*, pages 396–403, Oct 2013.
15. Karina Gitina, Ralf Wimmer, Sven Reimer, Matthias Sauer, Christoph Scholl, and Bernd Becker. Solving DQBF through quantifier elimination. In *Proceedings of DATE*, 2015.
16. E. Giunchiglia, M. Narizzano, L. Pulina, and A. Tacchella. Quantified Boolean Formulas satisfiability library (QBFLIB), 2005. `www.qbflib.org`.
17. Leon Henkin. Some remarks on infinitely long formulas. *Journal of Symbolic Logic*, 30:167–183, 1961.
18. Mikolás Janota and Joao Marques-Silva. Solving QBF by clause selection. In *Proceedings of IJCAI*, pages 325–331. AAAI Press, 2015.
19. Mikolás Janota and João P. Marques Silva. Abstraction-based algorithm for 2QBF. In *Proceedings of SAT*, pages 230–244, 2011.
20. Florian Lonsing and Armin Biere. DepQBF: A dependency-aware QBF solver. *JSAT*, 7(2-3):71–76, 2010.
21. G. Peterson, J. Reif, and S. Azhar. Lower bounds for multiplayer noncooperative games of incomplete information. *Computers & Mathematics with Applications*, 41(7):957 – 992, 2001.
22. Gary L Peterson and John H Reif. Multiple-person alternation. In *Proceedings of FOCS*, pages 348–363. IEEE, 1979.
23. Markus N. Rabe and Sanjit A. Seshia. Incremental determinization. In *Proceedings of SAT*, Berlin, Heidelberg, 2016. Springer-Verlag.
24. Markus N Rabe and Leander Tentrup. CAQE: A certifying QBF solver. In *Proceedings of FMCAD*, pages 136–143, 2015.
25. Jörg Siekmann and Graham Wrightson. *Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970.* Springer Science & Business Media, 1983.

26. João P Marques Silva and Karem A Sakallah. GRASP - A new search algorithm for satisfiability. In *Proceedings of CAD*, pages 220–227. IEEE, 1997.
27. Leander Tentrup. Non-prenex QBF solving using abstraction. In *Proceedings of SAT*, volume 9710 of *LNCS*, pages 393–401. Springer, 2016.
28. Leander Tentrup. On expansion and resolution in CEGAR based QBF solving. In *CAV (to appear)*, 2017.
29. Grigori S Tseitin. On the complexity of derivation in propositional calculus. *Studies in constructive mathematics and mathematical logic, Reprinted in [25]*, 2(115-125):10–13, 1968.
30. Allen Van Gelder. Contributions to the theory of practical quantified Boolean formula solving. In *Principles and Practice of Constraint Programming*, pages 647–663. Springer, 2012.
31. Ralf Wimmer, Karina Gitina, Jennifer Nist, Christoph Scholl, and Bernd Becker. Preprocessing for DQBF. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 173–190. Springer, 2015.
32. Ralf Wimmer, Sven Reimer, Paolo Marin, and Bernd Becker. HQSpre–An effective preprocessor for QBF and DQBF. In *Proceedings of TACAS*, 2017.
33. Lintao Zhang and S. Malik. Conflict driven learning in a quantified Boolean satisfiability solver. In *Proceedings of ICCAD*, pages 442–449, Nov 2002.