A PSPACE Subclass of Dependency Quantified Boolean Formulas and Its Effective Solving *

Christoph Scholl[1], Jie-Hong Roland Jiang[2], Ralf Wimmer[1,3], Aile Ge-Ernst[1]
[1]Albert-Ludwigs-Universität Freiburg, Freiburg im Breisgau, Germany
[2]National Taiwan University, Taipei, Taiwan
[3]Concept Engineering GmbH, Freiburg im Breisgau, Germany
{scholl, wimmer, geernsta}@informatik.uni-freiburg.de, jhjiang@ntu.edu.tw

Abstract

Dependency quantified Boolean formulas (DQBFs) are a powerful formalism, which subsumes quantified Boolean formulas (QBFs) and allows an explicit specification of dependencies of existential variables on universal variables. This enables a succinct encoding of decision problems in the NEXPTIME complexity class. As solving general DQBFs is NEXPTIME complete, in contrast to the PSPACE completeness of QBF solving, characterizing DQBF subclasses of lower computational complexity allows their effective solving and is of practical importance.

Recently a DQBF proof calculus based on a notion of fork extension, in addition to resolution and universal reduction, was proposed by Rabe in 2017. We show that this calculus is in fact incomplete for general DQBFs, but complete for a subclass of DQBFs, where any two existential variables have either identical or disjoint dependency sets over the universal variables. We further characterize this DQBF subclass to be Σ_3^{P} complete in the polynomial time hierarchy. Essentially using fork extension, a DQBF in this subclass can be converted to an equisatisfiable 3QBF with only a linear increase in formula size. We exploit this conversion for effective solving of this DQBF subclass and point out its potential as a general strategy for DQBF quantifier localization. Experimental results show that the method outperforms state-of-the-art DQBF solvers on a number of benchmarks, including the 2018 DQBF evaluation benchmarks.

1 Introduction

During the last two decades an enormous progress in the solution of quantifier-free Boolean formulas (SAT) has been observed. Nowadays, SAT solving is successfully used in many application areas, e. g., in planning (Rintanen, Heljanko, and Niemelä 2006), automatic test pattern generation (Eggersglüß and Drechsler 2012; Czutro et al. 2010), and formal verification of hard- and software systems (Biere et al. 2003; Clarke et al. 2001; Ivancic et al. 2008). Motivated by the success of SAT solvers, efforts, e. g., (Lonsing and Biere 2010; Janota et al. 2012; Janota and Marques-Silva 2015; Tentrup and Rabe 2015), have been made to consider the more general formalism of quantified Boolean formulas (QBFs).

Although QBFs are capable of encoding decision problems in the PSPACE complexity class, they are not powerful enough to succinctly encode many natural and practical problems that involve decisions under partial information. For example, the analysis of games with incomplete information (Peterson, Reif, and Azhar 2001), topologically constrained synthesis of logic circuits (Balabanov, Chiang, and Jiang 2014), synthesis of safe controllers (Bloem, Könighofer, and Seidl 2014), synthesis of fragments of linear-time temporal logic (Chatterjee et al. 2013), and verification of partial designs (Scholl and Becker 2001; Gitina et al. 2013) fall into this category and require an even more general formalism known as *dependency quantified Boolean formulas (DQBFs)* (Peterson, Reif, and Azhar 2001).

Unlike QBFs, where an existential variable implicitly depends on all the universal variables preceding its quantification level, DQBFs admit the dependency sets being explicitly specified. Essentially the dependency specifiable quantifications correspond to Henkin quantifiers (Henkin 1961). The semantics of a DQBF can be interpreted from a game-theoretic viewpoint as a game played by one universal player and multiple non-cooperative existential players with incomplete information, each partially observing the moves of the universal player as specified by his/her own dependency set. A DQBF is true if and only if the existential players have winning strategies. This specificity of dependencies allows DQBF encodings to be exponentially more compact than their equivalent QBF counterparts. In contrast to the PSPACE-completeness of QBF, the decision problem of DQBF is NEXPTIME-complete (Peterson, Reif, and Azhar 2001).

Driven by the needs of the applications mentioned above, research on DQBF solving has emerged in the past few years, leading to solvers such as iDQ (Fröhlich et al. 2014) and HQS (Gitina et al. 2015; Wimmer et al. 2015; 2017). Due to the high worst-case complexity of DQBF solving, identifying special DQBF structures, as we do in this paper, may provide insights for solver improvement.

As an example for a DQBF, consider the formula

$$\forall x_1 \forall x_2 \exists y_1(x_1) \exists y_2(x_2) \,:\, (x_1 \wedge x_2) \equiv (y_1 \equiv y_2)$$

from (Rabe 2017), which we will use as a running example

*This work was partly supported by the German Research Foundation (DFG) within the project 'Solving Dependency Quantified Boolean Formulas' and the Ministry of Science and Technology of Taiwan. JHJ was supported in part by the Alexander von Humboldt Foundation during this work.
Copyright © 2019, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

throughout this paper. The DQBF asks whether there are choices for y_1 only depending on the value of x_1, denoted $y_1(x_1)$, and for y_2 only depending on x_2, denoted $y_2(x_2)$, such that the Boolean formula after the quantifier prefix evaluates to true for all assignments to x_1 and x_2.[1] The Boolean formula in turn states that the existential variables y_1 and y_2 have to be equal iff x_1 and x_2 are true. Since y_1 can only 'see' x_1 and y_2 only x_2, y_1 and y_2 'cannot coordinate' to satisfy the constraint. Thus, the formula is false. However, a straightforward translation of this DQBF into a QBF with only implicit dependency sets will not work. Changing the quantifier prefix into a QBF quantifier prefix $\forall x_1 \exists y_1 \forall x_2 \exists y_2$ means that y_1 may depend on x_1, but y_2 may depend on x_1 and x_2. In that case the formula would be true. Changing the prefix into $\forall x_2 \exists y_2 \forall x_1 \exists y_1$ has a similar effect.

In this paper, we look into a technique for DQBF solving called fork extension, which has been presented in (Rabe 2017). Rabe uses fork extension in an attempt to turn resolution and universal reduction (Kleine Büning, Karpinski, and Flögel 1995) into a complete proof system for DQBF, which he calls fork resolution. Here we show that (in contrast to the statement in (Rabe 2017)) fork resolution is incomplete.

However, we use the technique of fork extension to show that a certain non-trivial subclass of DQBF (DQBF with pairwise disjoint or identical dependency sets) is in PSPACE (or more precisely: it is Σ_3^{P} complete). In fact, we show that formulas from this subclass can be converted into equisatisfiable QBFs with three quantifier levels, needing only a linear increase in formula size. Note that the above example falls into this DQBF subclass. In addition, for practical application we point out the potential usage of fork extension as a general strategy for DQBF quantifier localization.

To obtain a better characterization of subclasses of DQBF which are in PSPACE and others which are already NEXPTIME-complete, we consider a 'slightly extended' subclass (allowing dependency sets with all universal variables in addition) and show that this subclass is already NEXPTIME-complete. In addition, we prove that for *both* considered subclasses of DQBF the proof system with fork extension is complete.

Finally, we exploit the formula conversion technique to improve DQBF solvers by solving formulas efficiently in case that an initial test reveals that they belong to the special subclasses. Experimental results show that our method outperforms state-of-the-art DQBF solvers on a number of benchmarks, including the 2018 DQBF evaluation benchmarks (Pulina and Seidl 2018). The contributions of this work are thus twofold: providing a better understanding of DQBF complexity on the theoretical side and advancing DQBF solver technology on the practical side.

The paper is structured as follows: Sect. 2 presents the foundations of DQBF and Rabe's fork resolution calculus. In Sect. 3 we prove that the fork resolution calculus is incomplete. We present and analyze two sub-classes of DQBF regarding their complexity in Sect. 4. Sect. 5 shows that fork resolution *is* complete for these classes. Experimental results are presented in Sect. 6. Sect. 7 concludes this paper and outlines future work.

[1] We can interpret this as a *game* played by y_1 and y_2 against x_1 and x_2, where y_1 and y_2 only have incomplete information on actions of x_1, x_2, respectively.

2 Foundations

2.1 DQBF and QBF

Dependency quantified Boolean formulas are obtained by prefixing Boolean formulas with the so-called Henkin quantifiers (Henkin 1961).

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) *ψ over V has the form $\psi := Q : \phi$, where $Q = \forall x_1 \ldots \forall x_n \exists y_1(D_{y_1}) \ldots \exists y_m(D_{y_m})$, with $D_{y_i} \subseteq \{x_1, \ldots, x_n\}$ being the* dependency set *of y_i for $i = 1, \ldots, m$, is called the* (quantifier) prefix*, and ϕ, a quantifier-free Boolean formula over V, is called the* matrix *of ψ.*

We denote the set of universal variables by $V_\psi^\forall = \{x_1, \ldots, x_n\}$ and the set of existential variables by $V_\psi^\exists = \{y_1, \ldots, y_m\}$. We define the dependency function $\mathrm{dep} : V \to 2^V$ to be $\mathrm{dep}(v) = D_v$ if $v \in V_\psi^\exists$, and $\mathrm{dep}(v) = \{v\}$ if $v \in V_\psi^\forall$. Moreover, as the order of the variables in the prefix Q does not matter due to the explicit specification of dependency sets, we regard Q as a set. Thereby, the notation $Q \setminus \{v\}$ with a variable $v \in V$ denotes the prefix resulting from Q by removing the variable v together with its quantifier (as well as its dependency set in case v is existential, and all its occurrences in dependency sets if it is universal). Similarly, the notation $Q \cup \{\exists y(D_y)\}$ denotes adding an existential variable y (and universal quantifiers for variables that are in D_y but not in $V_\psi^\forall$) to the prefix.

The semantics of a DQBF is typically defined in terms of so-called Skolem functions. Let $[\![V]\!]$ denote the set of truth assignments to variables V.

Definition 2 (Semantics of DQBF)**.** *A DQBF ψ defined as above is* satisfiable *if and only if there exist functions $s_y : [\![D_y]\!] \to \mathbb{B}$ for $y \in V_\psi^\exists$, called* Skolem functions *of ψ, such that replacing in ϕ each existential variable y with (a Boolean expression over D_y for) s_y turns ϕ into a tautology.*

Deciding whether a given DQBF is satisfiable is NEXPTIME-complete (Peterson, Reif, and Azhar 2001).

Definition 3 (Equisatisfiability of DQBFs)**.** *Two DQBFs ψ and ψ' are called* equisatisfiable *if they are either both satisfiable or both unsatisfiable.*

An important special case of DQBFs is known as *quantified Boolean formulas*. They exhibit a linearly ordered quantifier prefix, where each existential variable y depends on all universal variables in whose scope it is:

Definition 4 (Syntax of QBF, Equivalent DQBF)**.** *Let $V = \{x_1, \ldots, x_n, y_1, \ldots, y_m\}$ be a finite, non-empty set of Boolean variables, $X_1, \ldots, X_k \subseteq \{x_1, \ldots, x_n\}$ a partition of $\{x_1, \ldots, x_n\}$ such that $X_i \neq \emptyset$ for $i = 2, \ldots, k$, and $Y_1, \ldots, Y_k \subseteq \{y_1, \ldots, y_m\}$ a partition of $\{y_1, \ldots, y_m\}$ such that $Y_i \neq \emptyset$ for $i = 1, \ldots, k-1$. Additionally let*

ϕ be a quantifier-free Boolean formula over V. A quantified Boolean formula (QBF) *Ψ (in prenex form) is given by $\Psi := \forall X_1 \exists Y_1 \forall X_2 \exists Y_2 \dots \forall X_k \exists Y_k : \phi$. The QBF Ψ is equivalent to the DQBF $\psi := \forall x_1 \dots \forall x_n \exists y_1(D_{y_1}) \dots \exists y_m(D_{y_m}) : \phi$, with $D_{y_i} = \bigcup_{\ell=1}^{L} X_\ell$ such that L is the unique index with $y_i \in Y_L$. In this case we say that the DQBF ψ 'can be written as a QBF Ψ' or the DQBF ψ 'has an equivalent QBF prefix'.*

Lemma 1 ((Gitina et al. 2015))**.** *A DQBF ψ has an equivalent QBF prefix if $D_y \subseteq D_{y'}$ or $D_{y'} \subseteq D_y$ holds for all $y, y' \in V_\psi^\exists$.*

QBFs, with PSPACE-complete complexity (Meyer and Stockmeyer 1973), can be solved more efficiently than general DQBFs. There are well-known QBF solvers, such as DepQBF (Lonsing and Biere 2010; Lonsing and Egly 2014), AIGSolve (Pigorsch and Scholl 2009; 2010), RAReQS (Janota et al. 2012), Qesto (Janota and Marques-Silva 2015), and CAQE (Tentrup and Rabe 2015).

In the following we assume that the matrix ϕ is given in *conjunctive normal form* (CNF). A formula is in CNF if it is a conjunction of *clauses*; a clause is a disjunction of *literals*, and a literal is either a variable v or its negation $\neg v$. We view a formula in CNF as a set of clauses and a clause as a set of literals, e. g., we write $\big\{\{x_1, \neg x_2\}, \{x_2, \neg x_3\}\big\}$ for the formula $(x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3)$. We assume that none of the clauses of the CNF ϕ under consideration is tautological, i. e., there is no variable v such that $\{v, \neg v\} \subseteq C$ for all $C \in \phi$. For a literal ℓ, $\mathrm{var}(\ell)$ denotes the corresponding variable, i. e., $\mathrm{var}(v) = \mathrm{var}(\neg v) = v$ and $\mathrm{dep}(\ell) = \mathrm{dep}\big(\mathrm{var}(\ell)\big)$. Moreover, for a clause $C = \{\ell_1, \dots, \ell_k\}$ we define $\mathrm{dep}(C) = \cup_{i=1}^{k} \mathrm{dep}(\ell_i)$ and for a CNF $\phi = \{C_1, \dots, C_n\}$ we define $\mathrm{dep}(\phi) = \cup_{i=1}^{n} \mathrm{dep}(C_i)$.

Note that a DQBF can be transformed such that its matrix is in CNF. While transforming the matrix directly into CNF without introducing new variables can cause an exponential blow-up in formula size, Tseitin transformation (Tseitin 1970) provides an alternative with only a linear increase in size at the cost of having additional existential variables. The idea is to introduce auxiliary existential variables that store the truth values of sub-expressions. Because the values of these variables are uniquely determined by the sub-expressions, they may depend on all universal variables.

2.2 Universal Expansion, Resolution, Universal Reduction, and Fork Extension

We give an overview on relevant techniques that are used in the literature in the context of DQBF solving.

Universal expansion, which has been defined for QBF, can be easily generalized to DQBF as observed, e. g., in (Bubeck and Kleine Büning 2006; Bubeck 2010; Balabanov, Chiang, and Jiang 2014; Gitina et al. 2013).

Definition 5 (Universal Expansion for DQBF)**.** *For a DQBF $\psi = \forall x_1 \dots \forall x_n \exists y_1(D_{y_1}) \dots \exists y_m(D_{y_m}) : \phi$ with $Z_{x_i} = \big\{y_j \in V_\psi^\exists \,\big|\, x_i \in D_{y_j}\big\}$, performing* universal expansion *on variable $x_i \in V_\psi^\forall$ in ψ yields a new prefix $\Big(Q \setminus \Big(\{x_i\} \cup \bigcup_{y_j \in Z_{x_i}} \{y_j\}\Big)\Big) \cup \big\{\exists y_j^b(D_{y_j} \setminus \{x_i\}) \,\big|\, y_j \in Z_{x_i}, b \in \{\overline{x_i}, x_i\}\big\}$ and a new matrix $\phi[0/x_i][y_j^{\overline{x_i}}/y_j] \wedge \phi[1/x_i][y_j^{x_i}/y_j]$ for all $y_j \in Z_{x_i}$, where $\phi[\kappa/v]$ denotes the Boolean formula resulting from ϕ by replacing all occurrences of variable v by Boolean expression κ.*

Definition 6 (Resolution)**.** *Let ϕ be a formula in CNF, ℓ a literal, and $C, C' \in \phi$ clauses such that $\ell \in C$ and $\neg\ell \in C'$. The* resolvent *of C and C' w. r. t. to the* pivot literal *ℓ is given by $C \otimes_\ell C' := \big(C \setminus \{\ell\}\big) \cup \big(C' \setminus \{\neg\ell\}\big)$.*

Resolvents are implied by the formula, i. e., if R is a resolvent of two clauses in ϕ, then ϕ and $\phi \cup \{R\}$ are logically equivalent (Biere et al. 2008, Sect. 3.2.1).

In (Wimmer et al. 2015) it has been proven that resolution can be used to eliminate existential variables from a DQBF under certain conditions (we mention only a subset here):

Theorem 1 (Variable elimination by resolution)**.** *Let $y \in V_\psi^\exists$ be an existential variable of ψ. We partition the matrix ϕ into the clause sets $\phi^y = \{C \in \phi \,|\, y \in C\}$, $\phi^{\neg y} = \{C \in \phi \,|\, \neg y \in C\}$, and $\phi^\emptyset = \phi \setminus (\phi^y \cup \phi^{\neg y})$.*
If one of the following conditions is satisfied
1. *for all $C \in \phi^y$ and all $k \in C$ we have $\mathrm{dep}(k) \subseteq \mathrm{dep}(y)$,*
2. *for all $C' \in \phi^{\neg y}$ and all $k \in C'$ we have $\mathrm{dep}(k) \subseteq \mathrm{dep}(y)$,*

then ψ is equisatisfiable with $\psi' := Q \setminus \{y\} : \phi^\emptyset \wedge \bigwedge_{C \in \phi^y} \bigwedge_{C' \in \phi^{\neg y}} C \otimes_y C'$.

Universal reduction removes a universal variable from a clause if the clause does not contain any existential variable which depends upon it. This technique has been generalized to DQBF in (Balabanov, Chiang, and Jiang 2014; Fröhlich et al. 2014).

Lemma 2 (Universal reduction)**.** *Let $Q : \phi \wedge C$ be a DQBF and $\ell \in C$ a universal literal such that for all $k \in C$ with $k \neq \ell$ we have $\mathrm{var}(\ell) \notin \mathrm{dep}(k)$. Then $Q : \phi \wedge C$ and $Q : \phi \wedge \big(C \setminus \{\ell\}\big)$ are equisatisfiable.*

It has been shown that resolution together with universal reduction (which is called Q-Resolution or QRes for short) provides a complete proof system for QBF (Kleine Büning, Karpinski, and Flögel 1995). This result means that a QBF Ψ is unsatisfiable if and only if QRes can derive the empty clause from Ψ. This statement is *not* true for DQBF as has been shown in (Balabanov, Chiang, and Jiang 2014). Rabe made the attempt to extend resolution with universal reduction to a complete proof system by adding a new operation called *fork extension*. Fork extension is based on the following lemma (Rabe 2017):

Lemma 3 (Splitting)**.** *Let $\psi = Q : \phi \wedge (C_1 \cup C_2)$ be a DQBF containing a clause $C_1 \cup C_2$ and let y be a new variable not occurring in ψ. Then ψ and $Q \cup \big\{\exists y(\mathrm{dep}(C_1) \cap \mathrm{dep}(C_2))\big\} : \phi \wedge (C_1 \cup \{y\}) \wedge (C_2 \cup \{\neg y\})$ are equisatisfiable.*

Rabe applies the above splitting rule on clauses consisting of two parts C_1 and C_2 with incomparable dependency sets, i. e., $\mathrm{dep}(C_1) \not\subseteq \mathrm{dep}(C_2)$ and $\mathrm{dep}(C_2) \not\subseteq \mathrm{dep}(C_1)$. We denote the proof system proposed by Rabe as QResSplit.

3 Incompleteness of QResSplit

Theorem 1 in (Rabe 2017) claims that QResSplit is a sound and complete proof calculus for DQBF. Unfortunately, the completeness part is incorrect.

Theorem 2. *QResSplit is incomplete for DQBF.*

Proof. We show that QResSplit cannot deduce the unsatisfiability of the following DQBF:

$$\forall x_1 \forall x_2 \forall x_3 \exists y_1(x_1,x_2) \exists y_2(x_2,x_3) \exists y_3(x_1,x_3) \; : \\ (x_1 \wedge x_2 \wedge x_3) \equiv (y_1 \oplus y_2 \oplus y_3)\,.$$

where the symbol '$\oplus$' denotes the exclusive OR (XOR) operation. The unsatisfiability of the DQBF can be seen as follows. We first rewrite the matrix as the conjunction of

$$(x_1 \wedge x_2 \wedge x_3) \rightarrow (y_1 \oplus y_2 \oplus y_3) \tag{1a}$$
$$(x_1 \wedge x_2 \wedge x_3) \leftarrow (y_1 \oplus y_2 \oplus y_3) \tag{1b}$$

Eq. (1b) can be written as the conjunction of

$$(\neg x_1) \rightarrow \neg(y_1 \oplus y_2 \oplus y_3) \tag{2a}$$
$$(\neg x_2) \rightarrow \neg(y_1 \oplus y_2 \oplus y_3) \tag{2b}$$
$$(\neg x_3) \rightarrow \neg(y_1 \oplus y_2 \oplus y_3) \tag{2c}$$

By universally expanding on all universal variables the DQBF with its matrix re-expressed by Eq. (1a), (2a), (2b), (2c) can be rewritten as the conjunction of the following linear equations over GF(2) (Han and Jiang 2012).

$$y_1^{x_1 x_2} \oplus y_2^{x_2 x_3} \oplus y_3^{x_1 x_3} = 1 \tag{3a}$$
$$y_1^{\overline{x_1}\,\overline{x_2}} \oplus y_2^{\overline{x_2}\,\overline{x_3}} \oplus y_3^{\overline{x_1}\,\overline{x_3}} = 0 \tag{3b}$$
$$y_1^{\overline{x_1}\,\overline{x_2}} \oplus y_2^{\overline{x_2} x_3} \oplus y_3^{\overline{x_1} x_3} = 0 \tag{3c}$$
$$y_1^{\overline{x_1} x_2} \oplus y_2^{x_2 \overline{x_3}} \oplus y_3^{\overline{x_1}\,\overline{x_3}} = 0 \tag{3d}$$
$$y_1^{\overline{x_1} x_2} \oplus y_2^{x_2 x_3} \oplus y_3^{\overline{x_1} x_3} = 0 \tag{3e}$$
$$y_1^{x_1 \overline{x_2}} \oplus y_2^{\overline{x_2}\,\overline{x_3}} \oplus y_3^{x_1 \overline{x_3}} = 0 \tag{3f}$$
$$y_1^{x_1 \overline{x_2}} \oplus y_2^{\overline{x_2} x_3} \oplus y_3^{x_1 x_3} = 0 \tag{3g}$$
$$y_1^{x_1 x_2} \oplus y_2^{x_2 \overline{x_3}} \oplus y_3^{x_1 \overline{x_3}} = 0 \tag{3h}$$

It can be verified through Gaussian elimination that the system of linear equations has no solution, and thus the original DQBF is unsatisfiable.

On the other hand, to see the limitation of QResSplit, the matrix of the DQBF should be expressed in CNF with the following clauses

$$(y_1 \vee \overline{y_2} \vee \overline{y_3} \vee \overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$$
$$(\overline{y_1} \vee y_2 \vee \overline{y_3} \vee \overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$$
$$(\overline{y_1} \vee \overline{y_2} \vee y_3 \vee \overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$$
$$(y_1 \vee y_2 \vee y_3 \vee \overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$$

$$(\overline{y_1} \vee \overline{y_2} \vee \overline{y_3} \vee x_1) \quad (\overline{y_1} \vee \overline{y_2} \vee \overline{y_3} \vee x_2) \quad (\overline{y_1} \vee \overline{y_2} \vee \overline{y_3} \vee x_3)$$
$$(y_1 \vee y_2 \vee \overline{y_3} \vee x_1) \quad (y_1 \vee y_2 \vee \overline{y_3} \vee x_2) \quad (y_1 \vee y_2 \vee \overline{y_3} \vee x_3)$$
$$(y_1 \vee \overline{y_2} \vee y_3 \vee x_1) \quad (y_1 \vee \overline{y_2} \vee y_3 \vee x_2) \quad (y_1 \vee \overline{y_2} \vee y_3 \vee x_3)$$
$$(\overline{y_1} \vee y_2 \vee y_3 \vee x_1) \quad (\overline{y_1} \vee y_2 \vee y_3 \vee x_2) \quad (\overline{y_1} \vee y_2 \vee y_3 \vee x_3)$$

The following observations can be made.

1. Universal reductions are not applicable to any clause, since in each clause all universal variables occur in the dependency sets of the existential variables.
2. All resolvents for pairs of clauses are tautologies.
3. Fork extension is inapplicable, because any partition of a clause C into two parts C_1 and C_2 does not yield incomparable dependency sets, that is, either $\mathrm{dep}(C_1) \subseteq \mathrm{dep}(C_2)$ or $\mathrm{dep}(C_2) \subseteq \mathrm{dep}(C_1)$.

Consequently, it is not possible to derive the empty clause by QResSplit from this DQBF. □

4 Classes DQBF$^{\mathrm{de}}$, DQBF$^{\mathrm{de+}}$, and DQBF$^{\mathrm{dec}}$

In this section, we define two main non-trivial sub-classes of DQBFs which we call DQBF$^{\mathrm{de}}$ and DQBF$^{\mathrm{dec}}$.[2] DQBF$^{\mathrm{de}}$ contains all DQBFs where the dependency sets of the existential variables are either pairwise equal or disjoint. DQBF$^{\mathrm{dec}}$ extends DQBF$^{\mathrm{de}}$ by additionally allowing dependency sets containing *all* universal variables. Formally,

Definition 7 (DQBF$^{\mathrm{de}}$ and DQBF$^{\mathrm{dec}}$)**.** *A DQBF $\psi := \forall x_1 \ldots \forall x_n \exists y_1(D_{y_1}) \ldots \exists y_m(D_{y_m}) : \phi$ with ϕ in CNF belongs to DQBF^{de} iff for all pairs of existential variables y_i and y_j ($1 \le i,j \le m$) $D_{y_i} = D_{y_j}$ or $D_{y_i} \cap D_{y_j} = \emptyset$. ψ belongs to DQBF^{dec} iff for all pairs of existential variables y_i and y_j ($1 \le i,j \le m$) $D_{y_i} = D_{y_j}$ or $D_{y_i} \cap D_{y_j} = \emptyset$ or $D_{y_i} = \{x_1,\ldots,x_n\}$ or $D_{y_j} = \{x_1,\ldots,x_n\}$.*

In the following, we prove that DQBF$^{\mathrm{de}}$ can be reduced to QBF and is thus in PSPACE. In contrast, the 'slightly extended' class DQBF$^{\mathrm{dec}}$ turns out to be an NEXPTIME complete subclass of DQBF (i. e., it is equivalent to DQBF in terms of complexity).

Theorem 3. *DQBF^{de} is in PSPACE.*

The proof is constructive and transforms a DQBF in DQBF$^{\mathrm{de}}$ into an equisatisfiable QBF. The conversion proceeds according to the following Alg. 1.

input : DQBF $\psi := Q : \phi$, $\psi \in$ DQBF$^{\mathrm{de}}$,
$Q := \forall x_1 \ldots \forall x_n \exists y_1(D_{y_1}) \ldots \exists y_m(D_{y_m})$,
ϕ in CNF, clauses in ϕ universally reduced
output : Equisatisfiable 3QBF

1 **while** $\exists C \in \phi, \ell_1 \in C \cap V_\psi^\exists$ *with* $\mathrm{dep}(\ell_1) \neq \emptyset$ *and* $\mathrm{dep}(C) \setminus \mathrm{dep}(\ell_1) \neq \emptyset$ **do**
2 $\quad C_1 := \{\ell \in C \mid \mathrm{dep}(\ell) \subseteq \mathrm{dep}(\ell_1)\}$; $C_2 := C \setminus C_1$;
3 $\quad t :=$ new variable with $\mathrm{dep}(t) = \emptyset$;
4 $\quad \phi := (\phi \setminus C) \cup \{C_1 \cup \{t\}\} \cup \{C_2 \cup \{\neg t\}\}$;
5 $\quad Q := Q \cup \{\exists t(\emptyset)\}$;
6 $\{t_1, \ldots t_k\} := \{v \in Q \mid v \in V_\psi^\exists, \mathrm{dep}(v) = \emptyset\}$;
7 $\{w_1, \ldots w_{m'}\} := \{v \in Q \mid v \in V_\psi^\exists, \mathrm{dep}(v) \neq \emptyset\}$;
8 **return** $\exists t_1, \ldots \exists t_k \forall x_1 \ldots \forall x_n \exists w_1 \ldots \exists w_{m'} : \phi$;

Algorithm 1: Convert DQBF$^{\mathrm{de}}$ into 3QBF.

Proof. We consider a DQBF in DQBF$^{\mathrm{de}}$ as given in Alg. 1. W. l. o. g. we assume that the clauses in ϕ are already universally reduced according to Lemma 2. The dependency sets of $y_1, \ldots, y_m$ form k non-empty disjoint sets $D^1, \ldots, D^k$ or k non-empty disjoint sets $D^1, \ldots, D^k$ together with $\emptyset$.

In lines 1–5 we apply fork extension to ψ as long as ϕ contains a clause C with an existential literal $\ell_1 \in C$, $\mathrm{dep}(\ell_1) \neq \emptyset$, and $\mathrm{dep}(C) \setminus \mathrm{dep}(\ell_1) \neq \emptyset$. In that case (due to universal reduction) C has to contain at least a second existential literal ℓ_2 with $\mathrm{dep}(\ell_2) \neq \emptyset$ and $\mathrm{dep}(\ell_2) \neq \mathrm{dep}(\ell_1)$ (and thus $\mathrm{dep}(\ell_2) \cap \mathrm{dep}(\ell_1) = \emptyset$, since ψ is in DQBF$^{\mathrm{de}}$). By fork extension we split C into $C_1 \cup \{t\}$ and $C_2 \cup \{\neg t\}$ (t is a new existential variable). C_1 contains all existential literals

[2] 'de' stands for 'disjoint or equal dependency sets' and 'dec' for 'disjoint, equal, or complete dependency sets'.

from C with the same dependency set as ℓ_1, all existential literals from C with empty dependency sets, and all universal literals from C which are in $\mathrm{dep}(\ell_1)$. By construction, $\mathrm{dep}(C_1) \cap \mathrm{dep}(C_2) = \emptyset$ and therefore $\mathrm{dep}(t) = \emptyset$. Fork extension is performed until the condition mentioned above does not hold for any clause. We call the result after all fork extensions ψ' and the corresponding matrix ϕ'.

The clauses of ϕ' can now be partitioned into $k+1$ sets $P_0, P_1, \ldots, P_k$. P_0 is possibly empty and contains only clauses C with $\mathrm{dep}(C) = \emptyset$. Each P_i with $1 \leq i \leq k$ is non-empty and contains only clauses C_i with $\mathrm{dep}(C_i) = D^i$, i. e., for all existential literals $\ell \in C_i$ we have $\mathrm{dep}(\ell) = D^i$ or $\mathrm{dep}(\ell) = \emptyset$, for all universal literals $\mu \in C_i$ we have $\mathrm{var}(\mu) \in D^i$. Now it is easy to see that all existential variables y occurring in P_i with $1 \leq i \leq k$ and $\mathrm{dep}(y) = D^i$ can be eliminated by resolution according to Thrm. 1. Clauses C_i with y or $\neg y$ only occur in P_i. As shown above, for all literals $k \in C_i$ we have $\mathrm{dep}(k) \subseteq D^i = \mathrm{dep}(y)$ and thus, y satisfies conditions 1) *and* 2) of Thrm. 1; 1) or 2) would already be sufficient according to Thrm. 1. Let $P_i^y = \{C \in P_i \mid y \in C\}$, $P_i^{\neg y} = \{C \in P_i \mid \neg y \in C\}$, and $P_i^\emptyset = P_i \setminus (P_i^y \cup P_i^{\neg y})$. It is important to note that elimination of y by resolution replaces P_i by $P_i' = P_i^\emptyset \wedge \bigwedge_{C \in P_i^y} \bigwedge_{C' \in P_i^{\neg y}} C \otimes_y C'$ and therefore after elimination of y we still have the property that all literals k in P_i' have dependency sets $\mathrm{dep}(k) \subseteq D^i$. Thus, variable elimination of existential variables can be performed in any arbitrary order for all existential variables with non-empty dependency sets D^i.

The consideration above implies that replacing all dependency sets $D^i \neq \emptyset$ in ψ' by the set $\{x_1, \ldots, x_n\}$ of all universal variables leads to an equisatisfiable DQBF. This simply follows from the fact that variable elimination by resolution for y depending on D^i and for y depending on $\{x_1, \ldots, x_n\}$ would lead to *exactly the same results*. After this transformation ψ' has three groups of variables: Existential variables depending on $\emptyset$, including the new variables introduced by fork extension and called $t_1, \ldots t_k$ in line 6 of Alg. 1, universal variables $x_1, \ldots, x_n$, and existential variables depending on $\{x_1, \ldots, x_n\}$, called $w_1, \ldots w_{m'}$ in line 7. Now it is easy to see that this DQBF can be written as a QBF ψ'' with only three quantifier levels, see line 8.

The size of ψ'' is linear in the size of the original DQBF ψ, since the number of needed fork extensions is limited by the number of occurrences of existential literals with non-empty dependency sets in ψ. Since the decision problem for QBF is in PSPACE, DQBF$^{\mathrm{de}}$ is in PSPACE as well. □

Example 1. *We consider the DQBF*

$$\forall x_1 \forall x_2 \exists y_1(x_1) \exists y_2(x_2) \,:\, (x_1 \wedge x_2) \equiv (y_1 \equiv y_2).$$

which is in DQBF^{de}. Translating the matrix ϕ into CNF results in

$$(y_1 \vee \overline{y_2} \vee \overline{x_1} \vee \overline{x_2}) \wedge (\overline{y_1} \vee y_2 \vee \overline{x_1} \vee \overline{x_2}) \wedge (\overline{y_1} \vee \overline{y_2} \vee x_1) \wedge (\overline{y_1} \vee \overline{y_2} \vee x_2) \wedge (y_1 \vee y_2 \vee x_1) \wedge (y_1 \vee y_2 \vee x_2).$$

Because the first clause $C = y_1 \vee \overline{y_2} \vee \overline{x_1} \vee \overline{x_2}$ contains variables with incomparable dependency sets, we have to split it into $C_1 = y_1 \vee \overline{x_1}$ and $C_2 = \overline{y_2} \vee \overline{x_2}$ and replace it by $t_1 \vee y_1 \vee \overline{x_1}$ and $\overline{t_1} \vee \overline{y_2} \vee \overline{x_2}$ with a new variable t_1 depending on $\mathrm{dep}(C_1) \cap \mathrm{dep}(C_2) = \emptyset$. Applying fork extension to all 6 clauses results in 12 clauses with 6 new variables $t_1, \ldots, t_6$, all with empty dependency sets:

$(t_1 \vee y_1 \vee \overline{x_1})$	$(\overline{t_1} \vee \overline{y_2} \vee \overline{x_2})$
$(t_2 \vee \overline{y_1} \vee \overline{x_1})$	$(\overline{t_2} \vee y_2 \vee \overline{x_2})$
$(t_3 \vee \overline{y_1} \vee x_1)$	$(\overline{t_3} \vee \overline{y_2})$
$(t_4 \vee \overline{y_1})$	$(\overline{t_4} \vee \overline{y_2} \vee x_2)$
$(t_5 \vee y_1 \vee x_1)$	$(\overline{t_5} \vee y_2)$
$(t_6 \vee y_1)$	$(\overline{t_6} \vee y_2 \vee x_2)$

The new matrix ϕ' can be partitioned into two sets P_1 and P_2. P_1 corresponds to the left block of clauses above; P_2 corresponds to the right one. All clauses in P_1 have dependency set $\{x_1\}$; all clauses in P_2 have dependency set $\{x_2\}$.

According to Thrm. 1 we can remove y_1 by resolution from P_1 (and thus from ϕ') and we can remove y_2 from P_2 (ϕ'). This means that we can replace the dependency sets of y_1 and y_2 by $\{x_1, x_2\}$ and we can write the DQBF as a QBF $\exists t_1 \ldots \exists t_6 \forall x_1 \forall x_2 \exists y_1 \exists y_2 : \phi'$.

By having a closer look at the proof of Thrm. 3 we immediately observe that the theorem holds for an extended class DQBF$^{\mathrm{de+}}$:

Definition 8 (DQBF$^{\mathrm{de+}}$)**.** *A DQBF $\psi := \forall x_1 \ldots \forall x_n \exists y_1(D_{y_1}) \ldots \exists y_m(D_{y_m}) : \phi$ with ϕ in CNF belongs to DQBF^{de+} iff for all pairs of existential variables y_i and y_j ($1 \leq i, j \leq m$)* occurring together in some clause of ϕ *we have $D_{y_i} = D_{y_j}$ or $D_{y_i} \cap D_{y_j} = \emptyset$.*

Corollary 1. *DQBF^{de+} is in PSPACE and each DQBF ψ in DQBF^{de+} can be transformed into an equisatisfiable 3QBF (a QBF with 3 quantifier levels).*

The proof of Thrm. 3 actually gives us a precise characterization of DQBF$^{\mathrm{de+}}$ (or DQBF$^{\mathrm{de}}$) within the polynomial time hierarchy (Stockmeyer 1976): On the one hand, each DQBF ψ in DQBF$^{\mathrm{de+}}$ can be transformed in polynomial time into an equisatisfiable 3QBF (a QBF with 3 quantifier levels). Therefore DQBF$^{\mathrm{de+}}$ is in Σ_3^{P}. On the other hand, each 3QBF can be viewed as a DQBF in DQBF$^{\mathrm{de+}}$, showing the Σ_3^{P}-hardness of DQBF$^{\mathrm{de+}}$. Together we have:

Corollary 2. *DQBF^{de+} is Σ_3^{P} complete.*

In contrast, Thrm. 3 does not hold anymore, if we replace DQBF$^{\mathrm{de}}$ by the 'slightly extended' class DQBF$^{\mathrm{dec}}$. In fact, we have:

Theorem 4. *DQBF^{dec} is NEXPTIME complete.*

Proof. DQBF$^{\mathrm{dec}}$ is in NEXPTIME, since DQBF$^{\mathrm{dec}}$ is a subclass of DQBF and DQBF is in NEXPTIME. Now we show that DQBF$^{\mathrm{dec}}$ is NEXPTIME hard by showing that each DQBF formula can be transformed into an equisatisfiable formula in DQBF$^{\mathrm{dec}}$ by a polynomial transformation. The proof can be led by considering that – according to (Gitina et al. 2013) – each DQBF formula can be regarded as a Partial Equivalence Checking Problem (PEC) and each PEC can be backtranslated into a DQBF. The backtranslation from (Gitina et al. 2013) in fact produces a formula from DQBF$^{\mathrm{dec}}$.

More specifically, we give a direct transformation. Let $\psi := \forall x_1 \ldots \forall x_n \exists y_1(D_{y_1}) \ldots \exists y_m(D_{y_m}) : \phi$ be a DQBF. Let $D_{y_j} = \{x_{j_1}, \ldots, x_{j_{k_j}}\}$. Introduce for each such variable

x_{j_i} a new copy $x^j_{j_i}$, leading to a new dependency set $D'_{y_j} = \{x^j_{j_1}, \ldots, x^j_{j_{k_j}}\}$. Since the dependency sets contain only new variable copies, it is clear that they are disjoint. It is easy to see that the non-CNF DQBF

$$\psi' := \forall x_1 \ldots \forall x_n \forall x^1_{1_1} \ldots \forall x^1_{1_{k_1}} \ldots \forall x^m_{m_1} \ldots \forall x^m_{m_{k_m}}$$
$$\exists y_1(D'_{y_1}) \ldots \exists y_m(D'_{y_m}) : \Big(\bigwedge_{j=1}^{m} \bigwedge_{l=1}^{k_j} (x^j_{j_l} \equiv x_{j_l}) \Big) \rightarrow \phi \quad (4)$$

and ψ are equisatisfiable. (The idea is that we require ϕ only to hold for assignments to the universal variables, where the copied variables have the same values as the original variables.) The matrix of ψ' can be transformed into CNF either using the laws of distributivity (with a potential exponential blow-up in size) or using Tseitin transformation with new Tseitin variables depending on all universal variables. The second option leads to an equisatisfiable formula in DQBF$^{\text{dec}}$ which is polynomial in the size of the original formula ψ. □

The proof shows that in Def. 7 it was essential to restrict DQBF$^{\text{de}}$ to formulas with the matrix in CNF. The described transformation in Eq. (4) allows us to make the dependency sets of all existential variables disjoint with a polynomial blowup of the formula – but it does not preserve the CNF structure of the matrix. If it did, Thrm. 3 would imply that NEXPTIME = PSPACE.

Quantifier Localization The 3QBFs resulting from fork extension in the proof of Thrm. 3 have a special structure which allows to 'localize quantifiers', i. e., to restrict the scope of quantifiers to *subformulas* of the matrix, leading to a non-prenex formula. QBF and DQBF solvers relying on formula rewriting like Q-Resolution and universal expansion profit from quantifier localization to a great extent, since the rewriting has to be done only for smaller subformulas.

For each set P_i from the partition of clauses mentioned in the proof of Thrm. 3, the existential variables depending on D_i as well as the universal variables from D_i can be quantified *locally* to P_i. E. g. for the DQBF from Ex. 1, after fork extension the clauses are partitioned into two sets P_1 and P_2 with $\text{dep}(P_1) = \{x_1\}$ and $\text{dep}(P_2) = \{x_2\}$. With quantifiers localized to P_1 and P_2, the DQBF can be rewritten into $\exists t_1 \ldots \exists t_6 : \big([\forall x_1 \exists y_1 : P_1] \wedge [\forall x_2 \exists y_2 : P_2]\big)$.

Skolem Functions For many applications not only DQBF solving, but also the computation of Skolem functions is needed. Despite the equisatisfiability between the converted 3QBFs and their original DQBFs in DQBF$^{\text{de+}}$, Skolem functions derived by QBF solving may not necessarily meet the dependency set requirements due to the relaxed QBF dependency conditions. Fortunately, using the special structure of the converted 3QBFs allowing partitioning the matrix into sets P_i, it is easy to prove that Skolem functions with the originally intended dependency sets can be recovered by simply substituting the extra dependency variables in the 3QBF Skolem functions with arbitrary Boolean constants.

5 QResSplit for DQBF$^{\text{de}}$ and DQBF$^{\text{dec}}$

In Sect. 3 we have shown that the proof system QResSplit is not complete for DQBF. Having a closer look at the proof of Thrm. 3, we immediately see that QResSplit is complete both for DQBF$^{\text{de}}$ / DQBF$^{\text{de+}}$ and for DQBF$^{\text{dec}}$. The proof is straightforward and much less involved than the (incorrect) proof attempt for Theorem 1 in (Rabe 2017).

Theorem 5. *QResSplit is complete for DQBF$^{\text{de+}}$ and for DQBF$^{\text{dec}}$.*

Proof. Let us consider a DQBF $\psi := \forall x_1 \ldots \forall x_n \exists y_1(D_{y_1}) \ldots \exists y_m(D_{y_m}) : \phi$ in DQBF$^{\text{dec}}$. First, the existential variables depending on all universal variables can be eliminated by resolution according to Thrm. 1.[3]

The result is a DQBF ψ' in DQBF$^{\text{de}} \subseteq$ DQBF$^{\text{de+}}$ (which may be exponentially larger than ψ). For each DQBF ψ' in DQBF$^{\text{de+}}$ fork extension exactly as in the proof of Thrm. 3 leads to a DQBF ψ''. It has been shown that all existential variables with non-empty dependency sets can be eliminated by resolution from ψ''. The resulting DQBF ψ''' contains only universal variables and existential variables with empty dependency sets. Then the universal variables can be removed by universal reduction, see Lemma 2. Finally, the remaining clauses to be considered form a SAT problem. There is a resolution proof deriving an empty clause from those clauses iff the remaining SAT problem is unsatisfiable. □

6 Experiments

For our experiments we have used one core of an Intel Xeon CPU E5-2650v2 with 2.6 GHz. The runtime of all experiments was limited to 1800 s and the memory to 4 GB.

We start our experimental analysis by considering two sets of parameterized benchmarks with disjoint dependency sets. Both sets are generalizations of the DQBF in Ex. 1.

For the first set of parameterized benchmarks we defined a family of DQBF$^{\text{de}}$ formulas $F_0, F_1, \ldots$, with the matrix translated into CNF in a straigthforward manner. F_n is defined as $F_n := Q^F_n : \big(\bigwedge_{i=0}^{n}(x^i_1 \wedge x^i_2)\big) \equiv \big(y^n_1 \equiv y^n_2\big)$. The prefix Q^F_n contains universal variables from the sets $X^n_1 = \{x^0_1, \ldots, x^n_1\}$, $X^n_2 = \{x^0_2, \ldots, x^n_2\}$. The existential variables y^n_1 and y^n_2 depend on X^n_1 and X^n_2, respectively. F_n has $2n+2$ universal and 2 existential variables. The second set $G_0, G_1, \ldots$ is defined similarly, but G_n has n^2+3n+2 universal and $2n+2$ existential variables. G_n is defined as $G_n := Q^G_n : \bigwedge_{i=0}^{n} \varphi_i$ with $\varphi_i = \big(\bigwedge_{j=0}^{i}\big(x^i_{j,1} \wedge x^i_{j,2}\big)\big) \equiv \big(y^i_1 \equiv y^i_2\big)$. For $0 \le i \le n$ the prefix Q^G_n contains universal variables from the sets $X^i_1 = \{x^i_{1,1}, \ldots, x^i_{i,1}\}$, $X^i_2 = \{x^i_{1,2}, \ldots, x^i_{i,2}\}$ and existential variables y^i_1 and y^i_2 depending on X^i_1 and X^i_2, resp. Note that both F_0 and G_0 coincide with our running example, which is a shortened version (Rabe 2017) of the formula used in (Balabanov, Chiang, and Jiang 2014) to prove that QRes is not complete for DQBF.

[3] For this completeness consideration, we actually do not have to remove clauses during 'variable elimination by resolution'. The corresponding theorem just says that we do not need to consider the removed clauses in any attempt to derive the empty clause.

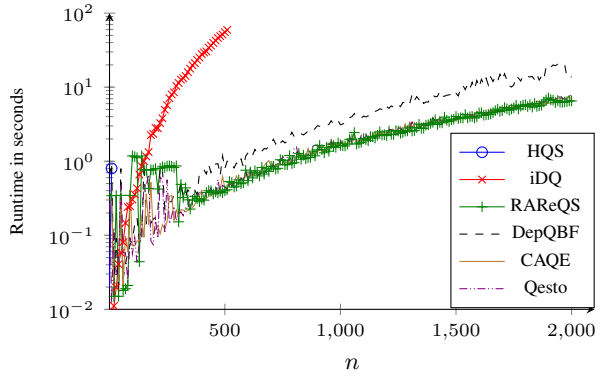

Figure 1: DQBF Solvers vs. QBF Solvers for Family F_n.

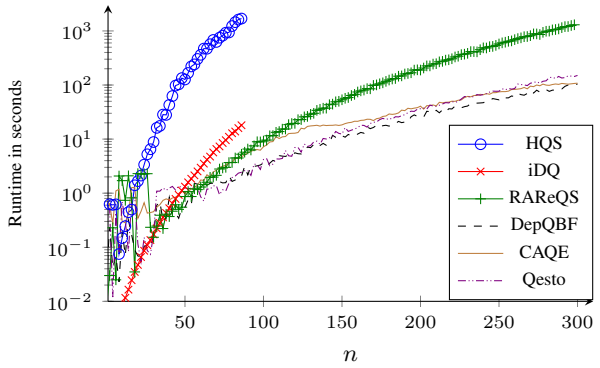

Figure 2: DQBF Solvers vs. QBF Solvers for Family G_n.

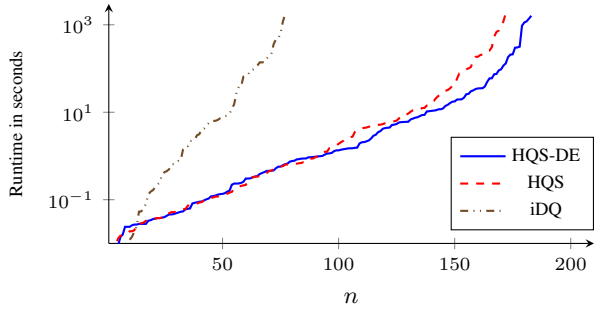

Figure 3: DQBF Solvers on Competition Benchmarks.

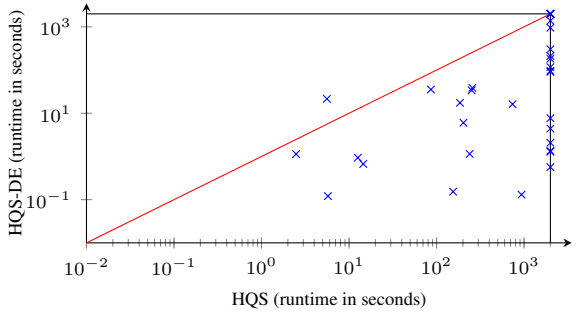

Figure 4: Competition Benchmarks, DQBF$^{\text{de+}}$.

We implemented the translation of DQBF formulas from DQBF$^{\text{de+}}$ into QBF as shown in Sect. 4 and applied it to the families F_n and G_n. For F_n we generated benchmarks with n increasing in steps of 10 from 0 to 2000, for G_n with n increasing from 0 to 300 in steps of 2. We evaluated the runtimes of the DQBF solvers HQS (Gitina et al. 2015) (which was the winner of the DQBF track of the QBFEVAL'18 competition) and iDQ (Fröhlich et al. 2014) applied to the original formulas and compared them to the runtimes of the state-of-the-art QBF solvers DepQBF (Lonsing and Biere 2010), RAReQS (Janota et al. 2012), Qesto (Janota and Marques-Silva 2015), and CAQE (Tentrup and Rabe 2015) applied to the QBFs resulting from translation. The CPU times for the QBF solvers include the times for translating the DQBFs into QBFs using fork extension.

In Fig. 1 we present the results for formulas F_n, in Fig. 2 for formulas G_n. The x-axis represents the value n, the y-axis the CPU time in seconds. HQS can only solve F_0 and F_{10}, but fails already for F_{20}. iDQ solves all instances until F_{510} and exceeds the memory limit starting with F_{520}. In contrast, all QBF solvers work pretty well on all generated instances. All of them can solve the instances up to F_{2000}. The maximum runtimes for solving the instances lie between 20.9 s for DepQBF and 7.2 s for RAReQS. This impressively shows the success of our conversion method for DQBF$^{\text{de}}$ formulas. The results are confirmed by experiments for G_n as well. Here both HQS and iDQ can solve the instances until G_{86} (for G_{88} HQS exceeds the time limit, iDQ the memory limit), but with higher runtimes for HQS (HQS needs 1694.1 s for G_{86}, iDQ 18.0 s). CAQE, DepQBF, and Qesto solve all benchmarks up to G_{300} with a maximum CPU time between 103.3 s for CAQE and 150.4 s for Qesto. RAReQS performs a little bit worse on this benchmark set; it exceeds the memory limit for G_{300} and solves G_{298} within 1300.8 s, but still performs much better than the DQBF solvers.

Encouraged by those results we enhanced the DQBF solver HQS (Wimmer et al. 2017) by a special treatment of formulas from DQBF$^{\text{de+}}$. Whenever we detect that a DQBF belongs to DQBF$^{\text{de+}}$ we translate it into a QBF and process it further by QBF solving. We call the resulting solver HQS-DE.[4] As a benchmark set we used the complete set of 334 instances from the DQBF track of the QBFEVAL'18 competition (Pulina and Seidl 2018). Fig. 3 shows a cactus plot comparing the performance of HQS-DE to HQS and iDQ on the complete set of DQBF benchmarks. It shows that HQS clearly outperforms iDQ on this set of benchmarks. HQS is able to solve 171 benchmarks, whereas iDQ only solves 77. Interestingly, HQS-DE in turn outperforms HQS, solving even 12 benchmarks more than HQS within 1800 s.

For the QBFEVAL'18 benchmarks, there are 34 out of 334 instances falling into the DQBF$^{\text{de+}}$ class. Fig. 4 shows a scat-

[4] Available at `https://abs.informatik.uni-freiburg.de/src/projects_content.php?projectID=21`.

ter plot with runtimes of HQS-DE and HQS for this subset of benchmarks. We see that HQS-DE clearly outperforms HQS on the DQBF^{de+} benchmarks. It is faster than HQS for all but one benchmark, and solves 28 benchmarks in contrast to only 14 solved by HQS. iDQ solves 18 out of 34 benchmarks. Although iDQ is slightly better than HQS on the DQBF^{de+} benchmarks, it is still strongly inferior to HQS-DE.

7 Conclusions and Future Work

Through our theoretical investigation on special classes of DQBFs, we have shown that the insights lead to improvements of state-of-the-art DQBF solvers by providing a specialized solution using fork extension. Experiments on two parameterized benchmark sets and the DQBF competition instances have shown the benefits of the proposed method.

For the future, we plan to look into fork extension for enabling quantifier localization in a more general setting, not only restricted to formulas from DQBF^{de+}. Moreover, we plan to complement the successful approach of the solver HQS which uses partial universal expansion and so-called dependency elimination (Wimmer et al. 2017) for transforming a DQBF into a QBF by a technique that alternatively transforms the DQBF into an equisatisfiable DQBF in DQBF^{de+}, enabling a translation into 3QBF by fork extension.

References

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

Biere, A.; Cimatti, A.; Clarke, E. M.; Strichman, O.; and Zhu, Y. 2003. Bounded model checking. *Advances in Computers* 58:117–148.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2008. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

Bloem, R.; Könighofer, R.; and Seidl, M. 2014. SAT-based synthesis methods for safety specs. In *VMCAI*, 1–20.

Bubeck, U., and Kleine Büning, H. 2006. Dependency quantified Horn formulas: Models and complexity. In *SAT*, 198–211.

Bubeck, U. 2010. *Model-based transformations for quantified Boolean formulas*. Ph.D. Dissertation, University of Paderborn, Germany.

Chatterjee, K.; Henzinger, T. A.; Otop, J.; and Pavlogiannis, A. 2013. Distributed synthesis for LTL fragments. In *FMCAD*, 18–25.

Clarke, E. M.; Biere, A.; Raimi, R.; and Zhu, Y. 2001. Bounded model checking using satisfiability solving. *Formal Methods in System Design* 19(1):7–34.

Czutro, A.; Polian, I.; Lewis, M. D. T.; Engelke, P.; Reddy, S. M.; and Becker, B. 2010. Thread-parallel integrated test pattern generator utilizing satisfiability analysis. *Int'l Journal of Parallel Programming* 38(3-4):185–202.

Eggersglüß, S., and Drechsler, R. 2012. A highly fault-efficient SAT-based ATPG flow. *IEEE Design & Test of Computers* 29(4):63–70.

Fröhlich, A.; Kovásznai, G.; Biere, A.; and Veith, H. 2014. iDQ: Instantiation-based DQBF solving. In *Int'l Workshop on Pragmatics of SAT (POS)*, 103–116.

Gitina, K.; Reimer, S.; Sauer, M.; Wimmer, R.; Scholl, C.; and Becker, B. 2013. Equivalence checking of partial designs using dependency quantified Boolean formulae. In *ICCD*, 396–403.

Gitina, K.; Wimmer, R.; Reimer, S.; Sauer, M.; Scholl, C.; and Becker, B. 2015. Solving DQBF through quantifier elimination. In *DATE*, 1617–1622.

Han, C., and Jiang, J. R. 2012. When boolean satisfiability meets gaussian elimination in a simplex way. In *CAV*, 410–426.

Henkin, L. 1961. Some remarks on infinitely long formulas. In *Infinitistic Methods: Proc. of the 1959 Symp. on Foundations of Mathematics*, 167–183.

Ivancic, F.; Yang, Z.; Ganai, M. K.; Gupta, A.; and Ashar, P. 2008. Efficient SAT-based bounded model checking for software verification. *Theor. Comput. Sci.* 404(3):256–274.

Janota, M., and Marques-Silva, J. 2015. Solving QBF by clause selection. In *IJCAI*, 325–331.

Janota, M.; Klieber, W.; Marques-Silva, J.; and Clarke, E. M. 2012. Solving QBF with counterexample guided refinement. In *SAT*, 114–128.

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

Lonsing, F., and Biere, A. 2010. DepQBF: A dependency-aware QBF solver. *Journal on Satisfiability, Boolean Modelling and Computation* 7(2-3):71–76.

Lonsing, F., and Egly, U. 2014. Incremental QBF solving by DepQBF. In *Int'l Congress on Mathematical Software (ICMS)*, 307–314.

Meyer, A. R., and Stockmeyer, L. J. 1973. Word problems requiring exponential time: Preliminary report. In *STOC*, 1–9.

Peterson, G.; Reif, J.; and Azhar, S. 2001. Lower bounds for multiplayer non-cooperative games of incomplete information. *Computers & Mathematics with Applications* 41(7–8):957–992.

Pigorsch, F., and Scholl, C. 2009. Exploiting structure in an AIG based QBF solver. In *DATE*, 1596–1601.

Pigorsch, F., and Scholl, C. 2010. An AIG-based QBF-solver using SAT for preprocessing. In *DAC*, 170–175.

Pulina, L., and Seidl, M. 2018. QBF-Eval'18 – Competitive evaluation of QBF solvers. `http://www.qbflib.org/qbfeval18.php`.

Rabe, M. N. 2017. A resolution-style proof system for DQBF. In *SAT*, 314–325.

Rintanen, J.; Heljanko, K.; and Niemelä, I. 2006. Planning as satisfiability: parallel plans and algorithms for plan search. *Artificial Intelligence* 170(12-13):1031–1080.

Scholl, C., and Becker, B. 2001. Checking equivalence for partial implementations. In *DAC*, 238–243.

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

Tentrup, L., and Rabe, M. N. 2015. CAQE: A certifying QBF solver. In *FMCAD*, 136–143.

Tseitin, G. S. 1970. On the complexity of derivation in propositional calculus. *Studies in Constructive Mathematics and Mathematical Logic* Part 2:115–125.

Wimmer, R.; Gitina, K.; Nist, J.; Scholl, C.; and Becker, B. 2015. Preprocessing for DQBF. In *SAT*, 173–190.

Wimmer, R.; Karrenbauer, A.; Becker, R.; Scholl, C.; and Becker, B. 2017. From DQBF to QBF by dependency elimination. In *SAT*, 326–343.