Preprint from 9th International Conference on Theory and Applications of Satisfiability Testing (SAT), July 2016, Bordeaux, France

Dependency Schemes for DQBF

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

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

Abstract. Dependency schemes allow to identify variable independencies in QBFs or DQBFs. For QBF, several dependency schemes have been proposed, which differ in the number of independencies they are able to identify. In this paper, we analyze the spectrum of dependency schemes that were proposed for QBF. It turns out that only some of them are sound for DQBF. For the sound ones, we provide a correctness proof, for the others counter examples. Experiments show that a significant number of dependencies can either be added to or removed from a formula without changing its truth value, but with significantly increasing the flexibility for modifying the representation.

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 formal verification of hard- and software systems [1,5,10], automatic test pattern generation [11,12], or planning [22]. Motivated by the success of SAT solvers, more general formalisms like quantified Boolean formulas (QBFs) have been studied. However, for applications like the verification of partial circuits [17,24], the synthesis of safe controllers [6], and the analysis of games with incomplete information [21], even QBF is not expressive enough to provide a compact and natural formulation. The reason is that the dependencies of existential variables on universal ones are restricted in QBF: Each existential variable implicitly depends on all universal variables in whose scope it is, i.e., in a model for a QBF in prenex normal form the value of an existential variable can be chosen depending on the values of the universal variables to the left. Consequently, the dependency sets of the existential variables (i.e., the sets of universal variables they may depend on) are linearly ordered w.r.t. set inclusion in QBF. In so-called *dependency quantified Boolean formulas (DQBFs)* this restriction is removed and for each existential variable a dependency set is explicitly specified. The more general DQBF formulations can be tremendously more compact than equivalent QBF formulations; on the other hand the decision problem is NEXPTIME-complete for DQBF [21] instead of

This work was partly supported by the German Research Council (DFG) as part of the project "Solving Dependency Quantified Boolean Formulas".

PSPACE-complete for QBF. Driven by the needs of the applications mentioned above, research on DQBF solving has started during the last few years, leading to first solvers like iDQ and HQS [14,15,18,28].

Although the dependency sets of existential variables are fixed by the syntax of DQBFs, it might be the case that those dependency sets can be reduced or extended without changing the DQBF's truth value. Manipulating dependency sets can be beneficial in different ways: In search-based DQBF solvers [14] extending the DPLL algorithm, manipulation of the dependencies may be used in a similar way as in QBF solvers [20], e.g., for detecting unit literals and conflicts earlier (due to possible universal reductions), for enabling decisions earlier, etc. In DQBF solvers relying on universal expansions like HQS [18], minimizing the number of dependencies to be considered leads to fewer copies of existential variables and thus to faster solving times with lower memory consumption. Manipulating the number of dependencies might lead to similar advantages for iDQ [15]. iDQ uses instantiation-based solving, i.e., it reduces deciding a DQBF to deciding a series of SAT problems which correspond to partial universal expansions. HQS [18] processes a DQBF by several methods until the resulting formula is a QBF, which then can be solved by an arbitrary QBF solver. Therefore also adding dependencies can be beneficial for HQS in order to make the formula more QBF-like.

An existential variable y, which contains a universal variable x in its syntactic dependency set, is called independent of x iff removing x from y's dependency set preserves the truth value of the DQBF. Using the same proof idea as described in [23] for QBF, it can be shown that deciding whether an existential variable is independent of a universal one has the same complexity as deciding the DQBF itself. Therefore one resorts to sufficient criteria to show independencies. We mainly look into generalizations of so-called dependency schemes, which were devised for QBF [16,23,25,26] and can be computed efficiently. A dependency scheme $\mathrm{ds}(\psi)$ for a DQBF ψ gives pairs of universal variables x and existential variables y such that 'y potentially depends on x'. If $(x, y) \notin \mathrm{ds}(\psi)$, then y is definitely independent of x.

The contributions of this paper are as follows:

- For DQBF, the paper provides generalizations of dependency schemes known from QBF and provides for the first time a comprehensive characterization of the dependency schemes which are sound for proving independencies in DQBFs. For the dependency schemes which are not sound for DQBF counterexamples are given.
- The paper proves for all DQBF dependency schemes that both adding and removing dependencies has a unique fixed point.
- Dependency schemes and an orthogonal method based on detection of functional definitions are seamlessly integrated in order to profit from each other.
- We present first experimental results showing an enormous amount of flexibility w.r.t. adding and removing dependencies in numerous benchmark instances.

Related work. Several dependency schemes have been introduced for QBF. Based on earlier ideas in [4,9,23] defined the so-called standard dependency scheme and the more precise triangle dependency scheme for QBF. In [16,25,26] these ideas have been refined further, leading to the even more precise resolution path dependency scheme. In [19,20] applications of dependency schemes to expansion-based and search-based QBF solvers have been intensively discussed. The correctness of using dependency schemes in search-based QBF solvers like DepQBF is studied in [27], showing that using quadrangle and triangle dependencies in that context is unsound. Instead (sound) reflexive variants of them are proposed. In [28] dependency schemes for DQBF has been considered first. [28] contains a generalization of the simple standard dependency scheme to DQBF, but neither a comprehensive characterization of the dependency schemes which are sound for DQBF nor any deeper analysis.

Structure of this paper. In the next section, we introduce the necessary foundations on DQBFs, Sect. 3 contains the main part of the paper on dependency schemes for DQBF. Section 4 shows first experimental results evaluating the flexibility provided by the different methods. Finally, Sect. 5 concludes the paper with a summary and directions for future research.

2 Foundations

Let φ, κ be quantifier-free Boolean formulas over the set V of variables and $v \in V$. We denote by $\varphi[\kappa/v]$ the Boolean formula which results from φ by replacing all occurrences of v (simultaneously) by κ. For a set $V' \subseteq V$, we denote by $\mathcal{A}(V')$ the set of Boolean assignments for V', i.e., $\mathcal{A}(V') = \{\nu \mid \nu : V' \to \{0,1\}\}$. For each formula φ over V, a variable assignment ν to the variables in V induces a truth value 0 or 1 of φ, which we call $\nu(\varphi)$.

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

$V_\psi^\forall = \{x_1, \dots, x_n\}$ is the set of universal and $V_\psi^\exists = \{y_1, \dots, y_m\}$ the set of existential variables of ψ. We often write $\psi = Q : \varphi$ with the quantifier prefix Q and the matrix φ. $Q \setminus \{v\}$ denotes the prefix that results from removing a variable $v \in V$ from Q together with its quantifier. If v is existential, then its dependency set is removed as well; if v is universal, then all occurrences of v in the dependency sets of existential variables are removed. Similarly we use $Q \cup \{\exists y(D_y)\}$ to add existential variables to the prefix. In this paper, we always assume that a DQBF $\psi = Q : \varphi$ as in Definition 1 with φ in conjunctive normal form (CNF) is given. A formula is in CNF if it is a conjunction of (non-tautological) *clauses*; a clause is a disjunction of *literals*, and a literal is either a variable v or its negation $\neg v$. As usual, we identify a formula in CNF with its set of clauses and a clause with its set of literals. For a formula φ (resp. clause C, literal l), $\mathrm{var}(\varphi)$ (resp. $\mathrm{var}(C)$, $\mathrm{var}(l)$) means the set of variables occurring in φ (resp. C, l); $\mathrm{lit}(\varphi)$ ($\mathrm{lit}(C)$) denotes the set of literals occurring in φ (C).

A QBF (in prenex normal form) is a DQBF such that $D_y \subseteq D_{y'}$ or $D_{y'} \subseteq D_y$ holds for any two existential variables $y, y' \in V_\psi^\exists$. Then the variables in V can be ordered resulting in a linear quantifier prefix, such that for each $y \in V_\psi^\exists$, D_y equals the set of universal variables which are to the left of y.

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

Definition 2 (Semantics of DQBF). *Let ψ be a DQBF as above. It is* satisfiable *iff there are functions $s_y : \mathcal{A}(D_y) \to \{0,1\}$ for $y \in V_\psi^\exists$ such that replacing each $y \in V_\psi^\exists$ by (a Boolean expression for) s_y turns ϕ into a tautology. The functions $(s_y)_{y\in V_\psi^\exists}$ are called* Skolem functions *for ψ.*

The elimination of universal variables in solvers like HQS [18] is done by *universal expansion* [2,7,8,17]:

Definition 3 (Universal expansion). *For a DQBF $\psi = \forall x_1 \dots \forall x_n \exists y_1(D_{y_1}) \dots \exists y_m(D_{y_m}) : \varphi$ with $Z_{x_i} = \{y_j \in V_\psi^\exists \,|\, x_i \in D_{y_j})\}$, the* universal expansion *w.r.t. variable $x_i \in V_\psi^\forall$, is defined by*

$$(Q\backslash\{x_i\})\cup\{\exists y_j'(D_{y_j}\backslash\{x_i\}) \,|\, y_j \in Z_{x_i}\} \,:\, \varphi[1/x_i]\wedge\varphi[0/x_i][y_j'/y_j \text{ for all } y_j \in Z_{x_i}].$$

ψ and its universal expansion have the same truth value, they are called 'equisatisfiable'. It can be seen from Definition 3 that the universal expansion w.r.t. a universal variable requires to double all existential variables which depend on it. This shows that reducing the dependency sets can be beneficial (as long as an equisatisfiable DQBF results). Increasing dependency sets may be beneficial as well, if adding dependencies makes the DQBF more QBF-like (see Sect. 3.2).

Manipulating the dependency sets is done using so-called dependency schemes in QBF. In the following section, we investigate which of the QBF dependency schemes can be generalized to DQBF and which cannot.

3 Dependency Schemes

Let $\psi = \forall x_1 \dots \forall x_n \exists y_1(D_{y_1}) \dots \exists y_m(D_{y_m}) : \varphi$ be a DQBF. For $x \in V_\psi^\forall$ and $y \in V_\psi^\exists$, we denote by $\psi \ominus (x,y)$ the formula which results from ψ by removing the dependency of y on x, i.e., by replacing D_y with $D_y \setminus \{x\}$. Accordingly, $\psi \oplus (x,y)$ results from ψ by replacing D_y with $D_y \cup \{x\}$.

Definition 4. *An existential variable $y \in V_\psi^\exists$ is* independent *of a universal variable $x \in V_\psi^\forall$ in ψ, iff $\psi \oplus (x,y)$ and $\psi \ominus (x,y)$ have the same truth value. In this case, the pair (x,y) is called a* pseudo-dependency.

Dependency schemes provide sufficient criteria to show variable independencies.

Definition 5 (Dependency scheme). *A* dependency scheme *for a DQBF ψ is a relation* $\mathrm{ds}(\psi) \subseteq V_\psi^\forall \times V_\psi^\exists$ *such that* $(x, y) \notin \mathrm{ds}(\psi)$ *implies that y is independent of x in ψ.*

Dependency schemes for QBF have been considered in several papers like [16,19,20,23,25–27]. They encompass the standard, strict standard, (reflexive) triangle, (reflexive) quadrangle and the resolution path dependency schemes. In the following, we generalize them to DQBF.

Most dependency schemes are based on 'connections' between clauses:

Definition 6 (Connected). *Let $\psi = Q : \varphi$ be a DQBF. A Z-*path *for $Z \subseteq V$ between two clauses $C, C' \in \varphi$ is a sequence $C_1, \ldots, C_n$ of clauses with $C = C_1$, $C' = C_n$ such that* $\mathrm{var}(C_i) \cap \mathrm{var}(C_{i+1}) \cap Z \neq \emptyset$ *for all $1 \leq i < n$. A sequence $v_1, \ldots, v_{n-1}$ of variables with* $v_i \in \mathrm{var}(C_i) \cap \mathrm{var}(C_{i+1}) \cap Z$ *for all $1 \leq i < n$ is called a* connecting sequence *of the Z-path. Two clauses $C, C' \in \varphi$ are* connected *w.r.t. Z (written $C \overset{Z}{\longleftrightarrow} C'$) if there is a Z-path between C and C'.*

Using the notion of connected clauses, Samer et al. defined the standard (sdep) and triangle (tdep) dependency schemes [23]. Van Gelder [16] generalized standard dependencies to strict standard dependencies (ssdep) and triangle dependencies to quadrangle dependencies (qdep), using connected clauses as well.

For the definition of stronger dependency schemes (i.e., of dependency schemes leading to *smaller* relations $\mathrm{ds}(\psi)$, and thus detecting more independencies), in [16,25,26] a more restricted notion of connected clauses has been introduced which leads to so-called resolution path dependencies:

Definition 7 (Resolution path connected). *Let $\psi = Q : \varphi$ be a DQBF. A* resolution Z-path *for $Z \subseteq V$ between two clauses $C, C' \in \varphi$ is a sequence $C_1, \ldots, C_n$ of clauses with $C = C_1$, $C' = C_n$ such that for all $1 \leq i < n$ there is a literal l_i with* $\mathrm{var}(l_i) \in Z$*, $l_i \in C_i$, $\neg l_i \in C_{i+1}$, and for all $1 \leq i < n-1$ we have* $\mathrm{var}(l_i) \neq \mathrm{var}(l_{i+1})$.[1] *The sequence* $\mathrm{var}(l_1), \ldots, \mathrm{var}(l_{n-1})$ *is called a* connecting sequence *of the resolution Z-path. Two clauses $C, C' \in \varphi$ are* resolution path connected *w.r.t. Z (written $C \underset{\mathrm{rp}}{\overset{Z}{\longleftrightarrow}} C'$) if there is a resolution Z-path between C and C'.*

Figure 1 gives an overview of the dependency schemes that have been proposed for QBF. The figure also shows the relation between the different dependency schemes. An arrow from a dependency scheme ds_1 to a scheme ds_2 means that $\mathrm{ds}_1(\psi) \subseteq \mathrm{ds}_2(\psi)$ holds for all DQBFs ψ and that there is at least one DQBF for which the subset relation is strict. ds_1 is more precise than ds_2 in the sense that every independence identified by ds_2 is also identified by ds_1, but not necessarily vice versa. Arrows that can be derived by transitivity have been omitted. Because the resolution path dependency scheme [16,25,26] is the quadrangle dependency scheme

[1] In [16] there is an additional constraint 'the resolvent of C_i and C_{i+1} w.r.t. l_i is non-tautologous'. This constraint has to be removed according to [25,26]. If it is not removed, resolution path dependencies are not sound.

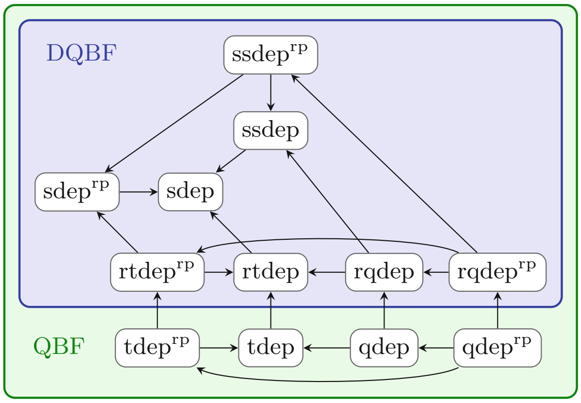


Fig. 1. The different dependency schemes for QBF and DQBF. Only those in the blue box are sound for DQBF, while all in the green box are sound for QBF. (Color figure online)

using 'resolution path connectivity' (Definition 7) instead of simple connectivity (Definition 6), we renamed it into *quadrangle resolution path dependency scheme* ($\mathrm{qdep}^{\mathrm{rp}}$). For each dependency scheme in $\{\mathrm{tdep}, \mathrm{tdep}^{\mathrm{rp}}, \mathrm{qdep}, \mathrm{qdep}^{\mathrm{rp}}\}$ there is a so-called 'reflexive' counterpart (called rtdep, $\mathrm{rtdep}^{\mathrm{rp}}$, rqdep, $\mathrm{rqdep}^{\mathrm{rp}}$, resp.) which is weaker than the original scheme. Reflexivity has been introduced in [27]. Exact definitions for the different dependency schemes for (D)QBF follow in Definition 8.

After giving the definition, we will have a closer look into the different dependency schemes. The main result of the paper will be the fact that for DQBF only those dependency schemes in the blue box are sound and the others are not.

Definition 8 (Dependency schemes). *Let ψ be a DQBF as in Def. 1. Furthermore, let $x^* \in V_\psi^\forall$ and $y^* \in V_\psi^\exists$ such that $x^* \in D_{y^*}$. We set $Z_{x^*} := \{z \in V_\psi^\exists \mid x^* \in D_z\}$.*

For the following dependency schemes, $(x^*, y^*) \in \mathrm{ds}(\psi)$ holds iff there are clauses $C_1, C_2, C_3, C_4 \in \varphi$ with $x^* \in C_1$, $\neg x^* \in C_2$, $y^* \in C_3$, and $\neg y^* \in C_4$ such that the following requirements hold:

1. standard dependency scheme sdep [23]:

$$C_1 \xleftrightarrow{Z_{x^*}} C_3 \vee C_2 \xleftrightarrow{Z_{x^*}} C_3 \vee C_1 \xleftrightarrow{Z_{x^*}} C_4 \vee C_2 \xleftrightarrow{Z_{x^*}} C_4$$

2. strict standard dependency scheme ssdep [16]:

$$(C_1 \xleftrightarrow{Z_{x^*}} C_3 \vee C_1 \xleftrightarrow{Z_{x^*}} C_4) \wedge (C_2 \xleftrightarrow{Z_{x^*}} C_3 \vee C_2 \xleftrightarrow{Z_{x^*}} C_4)$$

3. reflexive triangle dependency scheme rtdep [27]:

$$(C_1 \xleftrightarrow{Z_{x^*}} C_3 \vee C_2 \xleftrightarrow{Z_{x^*}} C_3) \wedge (C_1 \xleftrightarrow{Z_{x^*}} C_4 \vee C_2 \xleftrightarrow{Z_{x^*}} C_4)$$

4. triangle dependency scheme tdep [23]:

$$(C_1 \xleftrightarrow{Z_{x^*}\setminus\{y^*\}} C_3 \vee C_2 \xleftrightarrow{Z_{x^*}\setminus\{y^*\}} C_3) \wedge (C_1 \xleftrightarrow{Z_{x^*}\setminus\{y^*\}} C_4 \vee C_2 \xleftrightarrow{Z_{x^*}\setminus\{y^*\}} C_4)$$

5. reflexive quadrangle dependency scheme rqdep [27]:

$$(C_1 \xleftrightarrow{Z_{x^*}} C_3 \wedge C_2 \xleftrightarrow{Z_{x^*}} C_4) \vee (C_1 \xleftrightarrow{Z_{x^*}} C_4 \wedge C_2 \xleftrightarrow{Z_{x^*}} C_3)$$

6. quadrangle dependency scheme qdep [16]:

$$(C_1 \xleftrightarrow{Z_{x^*}\setminus\{y^*\}} C_3 \wedge C_2 \xleftrightarrow{Z_{x^*}\setminus\{y^*\}} C_4) \vee (C_1 \xleftrightarrow{Z_{x^*}\setminus\{y^*\}} C_4 \wedge C_2 \xleftrightarrow{Z_{x^*}\setminus\{y^*\}} C_3)$$

The references given for each of these dependency schemes refer to the original definition for QBF. Only the standard dependency scheme has been considered for DQBF so far [28]. Its correctness is proven in [29].

For each of these dependency schemes sdep, ssdep, rtdep, tdep, rqdep, qdep, a stronger variant is obtained by replacing connectedness $C_i \xleftrightarrow{Z_{x^*}} C_j$ ($C_i \xleftrightarrow{Z_{x^*}\setminus\{y^*\}} C_j$) with resolution path connectedness $C_i \xleftrightarrow[\text{rp}]{Z_{x^*}} C_j$ ($C_i \xleftrightarrow[\text{rp}]{Z_{x^*}\setminus\{y^*\}} C_j$, resp.). This yields the standard resolution path dependency scheme, the strict standard resolution path dependency scheme, etc. [16,25–27]. They are denoted with sdep^{rp}, ssdep^{rp}, rtdep^{rp}, tdep^{rp}, rqdep^{rp}, and qdep^{rp}, respectively. We use the following sets of dependency schemes:

$$\begin{aligned}\Delta^{\text{dqbf}} &= \{\text{sdep}, \text{sdep}^{\text{rp}}, \text{ssdep}, \text{ssdep}^{\text{rp}}, \text{rtdep}, \text{rtdep}^{\text{rp}}, \text{rqdep}, \text{rqdep}^{\text{rp}}\},\\ \Delta^{\text{qbf}} &= \Delta^{\text{dqbf}} \cup \{\text{tdep}, \text{tdep}^{\text{rp}}, \text{qdep}, \text{qdep}^{\text{rp}}\},\end{aligned}$$

The quadrangle resolution path dependency scheme has been introduced as ‘resolution path dependency scheme’ in [16,25,26]. For a clear categorization, however, we prefer to call it the ‘quadrangle resolution path dependency scheme’ qdep^{rp}.

The relations between the dependency schemes shown in Fig. 1 are the immediate consequences of the different dependency schemes’ definition. As each resolution path connection according to Definition 7 is also a simple connection according to Definition 6, but not vice versa, each variant using resolution paths is stronger than its counterpart that uses simple paths.

For all of the defined dependency schemes, it is known that the following result holds for QBF:

Theorem 1 **([16,23,25–27]).** *Let $\psi = Q : \varphi$ be a **QBF**. Let $x^* \in V_\psi^\forall$ and $y^* \in V_\psi^\exists$ such that $x^* \in D_{y^*}$. If, for some dependency scheme $\text{ds} \in \Delta^{\text{qbf}}$, $(x^*, y^*) \notin \text{ds}(\psi)$ and $\psi \ominus (x^*, y^*)$ is a **QBF** as well, then y^* is independent of x^*.*

In the following, we will prove (1) that the reflexive resolution path dependency scheme (and all weaker schemes) are sound for DQBF as well, and (2) that the triangle dependency scheme (and all stronger schemes) are unsound for DQBF.

We start by showing that the triangle dependency scheme is unsound for arbitrary DQBFs:

Theorem 2. *There is a DQBF $\psi = Q : \varphi$ with $x^* \in V_\psi^\forall$, $y^* \in V_\psi^\exists$ such that $x^* \in D_{y^*}$, with the following property: y^* is* not *independent of x^*, but $(x^*, y^*) \notin \text{tdep}(\psi)$, $(x^*, y^*) \notin \text{tdep}^{\text{rp}}(\psi)$, $(x^*, y^*) \notin \text{qdep}(\psi)$, and $(x^*, y^*) \notin \text{qdep}^{\text{rp}}(\psi)$.*

Proof. Let

$$\begin{aligned} D_1 &= (x_1 \vee x_2 \vee \neg y_1), \quad D_2 = (x_2 \vee y_1 \vee y_2) \\ D_3 &= (\neg x_2 \vee y_1 \vee \neg y_2), \; D_4 = (\neg x_1 \vee \neg x_2 \vee \neg y_1), \text{ and} \\ &\qquad \varphi = D_1 \wedge D_2 \wedge D_3 \wedge D_4. \end{aligned}$$

Consider the DQBF $\psi = \forall x_1 \forall x_2 \exists y_1(x_1, x_2) \exists y_2(x_1) : \varphi$. First we show for ψ that y_1 is *not* independent of x_1, i.e., we have to prove that ψ is satisfiable, but $\psi' = \psi \ominus (x_1, y_1) = \forall x_1 \forall x_2 \exists y_1(x_2) \exists y_2(x_1) : \varphi$ is unsatisfiable. We prove this by considering the full universal expansions of ψ and ψ' w.r.t. variables x_1 and x_2:

– ψ is equisatisfiable to

$$\big[(\neg y_1^{00}) \wedge (y_1^{00} \vee y_2^0)\big] \wedge \big[(y_1^{01} \vee \neg y_2^0)\big] \wedge \big[(y_1^{10} \vee y_2^1)\big] \wedge \big[(y_1^{11} \vee \neg y_2^1) \wedge (\neg y_1^{11})\big]$$

A satisfying assignment is given by $y_2^0 = y_1^{01} = y_1^{10} = 1$, $y_1^{11} = y_2^1 = y_1^{00} = 0$.

– ψ' is equisatisfiable to

$$\big[(\neg y_1^0) \wedge (y_1^0 \vee y_2^0)\big] \wedge \big[(y_1^1 \vee \neg y_2^0)\big] \wedge \big[(y_1^0 \vee y_2^1)\big] \wedge \big[(y_1^1 \vee \neg y_2^1) \wedge (\neg y_1^1)\big]$$

Due to the unit clause $(\neg y_1^0)$, y_1^0 has to be 0. Therefore y_2^0 has to be 1 and thus y_1^1 has to be 1. This contradicts the unit clause $(\neg y_1^1)$.

On the other hand, it is easy to see that $(x_1, y_1) \notin \text{tdep}(\psi)$. For $(x_1, y_1) \in \text{tdep}(\psi)$ we would need clauses $C_1, C_2, C_3, C_4 \in \varphi$ with $x_1 \in \text{var}(C_1)$, $x_1 \in \text{var}(C_2)$, $y_1 \in C_3$, $\neg y_1 \in C_4$, such that $C_1 \overset{\{y_2\}}{\longleftrightarrow} C_3$ and $C_2 \overset{\{y_2\}}{\longleftrightarrow} C_4$. The only clauses in φ containing $\neg x_1$ or x_1 are D_1 and D_4. Since $\neg y_1$ is the only existential literal in D_1 and D_4, the only $\{y_2\}$-path starting with D_1 (D_4) is the sequence D_1 (D_4) of length 1. Therefore a suitable clause C_3 cannot be found in φ.

$(x_2, y_1) \notin \text{tdep}(\psi)$ implies $(x_2, y_1) \notin \text{tdep}^{\text{rp}}(\psi)$, $(x_2, y_1) \notin \text{qdep}(\psi)$, and $(x_2, y_1) \notin \text{qdep}^{\text{rp}}(\psi)$. □

The next step is to prove that rqdep^{rp} is sound for DQBF.

Theorem 3. *Let $x^* \in V_\psi^\forall$ and $y^* \in V_\psi^\exists$ such that $x^* \in D_{y^*}$. If $(x^*, y^*) \notin \text{rqdep}^{\text{rp}}(\psi)$, then y^* is independent of x^*.*

For the proof, we assume that y^* is *not* independent of x^* and show that then $(x^*, y^*) \in \text{rqdep}^{\text{rp}}(\psi)$. The main idea of the proof consists of using universal expansion on the DQBF until known results for QBFs become applicable. Then these results imply $(x^*, y^*) \in \text{rqdep}^{\text{rp}}(\psi)$. Before we come to the proof, we have to consider the following technical lemma on DQBFs resulting from universal expansion:

Lemma 1. *Let $\psi = Q : \varphi$ be a DQBF and $\psi' = Q' : \varphi'$ a DQBF derived from ψ by universally expanding some variables in ψ. If there exists a resolution Z-path from $C_1 \in \varphi'$ to $C_2 \in \varphi'$, then the connecting sequence $y_1, \ldots, y_n$ (see Definition 7) does not include a pair (y_i, y_{i+1}) where y_i and y_{i+1} are two copies of the same existential variable in ψ.*

Proof. Assume that the resolution Z-path has a connecting sequence with a pair (y_i, y_{i+1}) where y_i and y_{i+1} are two copies of the same existential variable in ψ. According to the definition of resolution Z-paths, $y_i \neq y_{i+1}$, i.e., y_i and y_{i+1} are two *different* copies of the same existential variable in ψ. Then there exists a clause $C \in \varphi'$ with $y_i, y_{i+1} \in \text{var}(C)$, $y_i \neq y_{i+1}$. This contradicts the definition of universal expansion (see Definition 3), since clauses in a universal expansion can contain at most one copy of the same original variable. $\square$

Now we come to the proof of Theorem 3. After the proof, its construction is illustrated by Example 1.

Proof. Let $\psi = \forall x_1 \ldots \forall x_n \exists y_1(D_{y_1}) \ldots \exists y_m(D_{y_m}) : \varphi$ be a DQBF. W.l.o.g. assume that $x_1 \in D_{y_1}$ and y_1 is not independent of x_1. We have to prove that $(x_1, y_1) \in \text{rqdep}^{\text{rp}}(\psi)$.

Since, in ψ, y_1 is not independent of x_1, ψ has to be satisfiable and $\psi \ominus (x_1, y_1)$ is unsatisfiable. In ψ, we universally expand on the remaining universal variables $x_2, \ldots, x_n$. Let $\boldsymbol{y}_{\textbf{indep}}$ be the copies of existential variables y_i $(i \in \{2, \ldots, m\})$ with $x_1 \notin D_{y_i}$, $\boldsymbol{y}_{\textbf{dep}}$ be the copies of existential variables y_i $(i \in \{2, \ldots, m\})$ with $x_1 \in D_{y_i}$, and $y_1^1, \ldots, y_1^k$ the copies of y_1 with $k = 2^{|D_{y_1}|-1}$. Thus, universal expansion results in the following **QBF** ψ':

$$\psi' = \exists \boldsymbol{y}_{\textbf{indep}} \forall x_1 \exists y_1^1 \ldots \exists y_1^k \exists \boldsymbol{y}_{\textbf{dep}} : \varphi'.$$

Now we start to move copies y_1^i from the right of x_1 to the left of x_1. Since $\psi \ominus (x_1, y_1)$ is unsatisfiable, ψ' will get unsatisfiable when *all* $y_1^1 \ldots y_1^k$ have been moved to the left of x_1. So there exists a maximal subset $\boldsymbol{y}_{\mathbf{1}}^{\textbf{mov}} \subsetneq \{y_1^1, \ldots, y_1^k\}$ (with $\boldsymbol{y}_{\mathbf{1}}^{\textbf{stay}} = \{y_1^1, \ldots y_1^k\} \setminus \boldsymbol{y}_{\mathbf{1}}^{\textbf{mov}}$ containing the remaining variables from $\{y_1^1 \ldots y_1^k\}$) with the property that

$$\psi'' = \exists \boldsymbol{y}_{\textbf{indep}} \exists \boldsymbol{y}_{\mathbf{1}}^{\textbf{mov}} \forall x_1 \exists \boldsymbol{y}_{\mathbf{1}}^{\textbf{stay}} \exists \boldsymbol{y}_{\textbf{dep}} : \varphi'$$

is satisfied and, in ψ'', each variable from $\boldsymbol{y}_{\mathbf{1}}^{\textbf{stay}}$ is *not* independent of x_1. That means that moving an arbitrary variable from $\boldsymbol{y}_{\mathbf{1}}^{\textbf{stay}}$ to the left of x_1 will turn ψ'' from satisfiable to unsatisfiable. The set $\boldsymbol{y}_{\mathbf{1}}^{\textbf{stay}}$ is not empty, since otherwise y_1 would be independent of x_1 in ψ. Choose an arbitrary existential variable y_1^j in $\boldsymbol{y}_{\mathbf{1}}^{\textbf{stay}}$. Since ψ'' is a QBF where y_1^j is not independent of x_1, we have $(x_1, y_1^j) \in \text{qdep}^{\text{rp}}(\psi'')$ due to Theorem 1, i.e., $\exists C_1, C_2, C_3, C_4 \in \varphi'$ with $x_1 \in C_1$, $\neg x_1 \in C_2$, $y_1^j \in C_3$, $\neg y_1^j \in C_4$, and – w.l.o.g. – $C_1 \xleftrightarrow[\text{rp}]{Z_{x_1} \setminus \{y_1^j\}} C_3$ and $C_2 \xleftrightarrow[\text{rp}]{Z_{x_1} \setminus \{y_1^j\}} C_4$ with $Z_{x_1} = \boldsymbol{y}_{\mathbf{1}}^{\textbf{stay}} \cup \boldsymbol{y}_{\textbf{dep}}$.

Now we make use of a simple property of the process of universal expansion in order to turn the two constructed resolution $(Z_{x_1} \setminus \{y_1^j\})$-paths for ψ'' into suitable resolution paths for the original DQBF ψ proving that $(x_1, y_1) \in \text{rqdep}^{\text{rp}}(\psi)$: Each clause C in a universal expansion can be mapped back to a clause C^{orig} of the original formula 'it results from' by (1) adding universal literals which have been removed by the universal expansion, since they are unsatisfied in the copy of the clause at hand, and by (2) replacing copies of existential literals ℓ by the existential literals ℓ^{orig} in the original formula. Now consider one of the constructed resolution $(Z_{x_1} \setminus \{y_1^j\})$-paths $D_1, \ldots, D_n$ with $D_1 = C_1$ and $D_n = C_3$ (or $D_1 = C_2$ and $D_n = C_4$). For all $1 \leq i < n$ there is a literal ℓ_i with $\text{var}(\ell_i)$ from $Z_{x_1} \setminus \{y_1^j\} = \boldsymbol{y}_{\mathbf{dep}} \cup (\boldsymbol{y}_{\mathbf{1}}^{\mathbf{stay}} \setminus \{y_1^j\})$, $\ell_i \in C_i$, $\neg\ell_i \in C_{i+1}$. It is easy to see that $D_1^{\text{orig}}, \ldots, D_n^{\text{orig}}$ is a resolution Z'_{x_1}-path for ψ with $Z'_{x_1} = \{y \in V_\psi^\exists \mid x_1 \in D_y\}$: For all $1 \leq i < n$ there is a literal ℓ_i^{orig} with $\text{var}(\ell_i^{\text{orig}}) \in Z'_{x_1}$, $\ell_i^{\text{orig}} \in C_i$, $\neg\ell_i^{\text{orig}} \in C_{i+1}$. (Note that ℓ_i^{orig} may also be y_1 or $\neg y_1$, since the connecting sequence of $D_1, \ldots, D_n$ may contain a copy of y_1 which is different from y_1^j.) Due to Lemma 1, $\text{var}(\ell_i)$ and $\text{var}(\ell_{i+1})$ are copies of *different* existential variables and thus $\text{var}(\ell_i^{\text{orig}}) \neq \text{var}(\ell_{i+1}^{\text{orig}})$ for all $1 \leq i < n$. Altogether we have proven that there are two resolution Z'_{x_1}-paths for ψ with $Z'_{x_1} = \{y \in V_\psi^\exists \mid x_1 \in D_y\}$ leading from $C_1^{\text{orig}} \in \varphi$ with $x_1 \in C_1^{\text{orig}}$ to $C_3^{\text{orig}} \in \varphi$ with $y_1 \in C_3^{\text{orig}}$ and from $C_2^{\text{orig}} \in \varphi$ with $\neg x_1 \in C_2^{\text{orig}}$ to $C_4^{\text{orig}} \in \varphi$ with $\neg y_1 \in C_4^{\text{orig}}$. Thus $(x_1, y_1) \in \text{rqdep}^{\text{rp}}(\psi)$.[2] $\square$

Example 1. We illustrate the construction of the proof for Theorem 3 by means of an example. Here we use the same example as in the proof of Theorem 2, i.e.,

$$\begin{aligned} D_1 &= (x_1 \vee x_2 \vee \neg y_1), \quad D_2 = (x_2 \vee y_1 \vee y_2) \\ D_3 &= (\neg x_2 \vee y_1 \vee \neg y_2), \; D_4 = (\neg x_1 \vee \neg x_2 \vee \neg y_1), \\ &\qquad \varphi = D_1 \wedge D_2 \wedge D_3 \wedge D_4, \end{aligned}$$

and the DQBF $\psi = \forall x_1 \forall x_2 \exists y_1(x_1, x_2) \exists y_2(x_1) : \varphi$. For ψ, y_1 is *not* independent of x_1. In ψ, we universally expand on the universal variables different from x_1, i.e., we expand on x_2. This results in the DQBF

$$\psi' = \forall x_1 \exists y_1^0(x_1) \exists y_1^1(x_1) \exists y_2(x_1) : \\ \big[(x_1 \vee \neg y_1^0) \wedge (y_1^0 \vee y_2)\big] \wedge \big[(y_1^1 \vee \neg y_2) \wedge (\neg x_1 \vee \neg y_1^1)\big] .$$

Since only one universal variable is left, the DQBF is a QBF and can be written (in QBF notation) as

$$\psi' = \forall x_1 \exists y_1^0 \exists y_1^1 \exists y_2 : \; \big[(x_1 \vee \neg y_1^0) \wedge (y_1^0 \vee y_2)\big] \wedge \big[(y_1^1 \vee \neg y_2) \wedge (\neg x_1 \vee \neg y_1^1)\big] .$$

There are no existential variables which do not have x_1 in their dependency set, i.e., in the notions used in the proof, $\boldsymbol{y}_{\mathbf{indep}} = \emptyset$, $\boldsymbol{y}_{\mathbf{dep}} = \{y_2\}$, and there are two copies (y_1^0 and y_1^1) of y_1. Moving y_1^0 or y_1^1 to the left of $\forall x_1$ turns the QBF from satisfiable to unsatisfiable, thus, in the notions used in the proof, $\boldsymbol{y}_{\mathbf{1}}^{\mathbf{mov}} = \emptyset$ and $\boldsymbol{y}_{\mathbf{1}}^{\mathbf{stay}} = \{y_1^0, y_1^1\}$, and each variable from $\boldsymbol{y}_{\mathbf{1}}^{\mathbf{stay}}$ is not independent of x_1. Choose $y_1^0 \in \boldsymbol{y}_{\mathbf{1}}^{\mathbf{stay}}$. Due to Theorem 1, $(x_1, y_1^0) \in \text{qdep}^{\text{rp}}(\psi')$. This can be seen by choosing $C_1 = C_4 = (x_1 \vee \neg y_1^0)$, $C_2 = (\neg x_1 \vee \neg y_1^1)$, and $C_3 = (y_1^0 \vee y_2)$.

[2] The construction does *not* lead to $(Z'_{x_1} \setminus \{y_1\})$-paths and thus $(x_1, y_1) \in \text{qdep}^{\text{rp}}(\psi)$ cannot be proven (which is not surprising due to Theorem 2).

(1) $(x_1 \lor \neg y_1^0)$ is a (trivial) resolution $\{y_1^1, y_2\}$-path with empty connecting sequence showing $C_1 \xleftrightarrow[\text{rp}]{\{y_1^1,y_2\}} C_4$ and
(2) $(\neg x_1 \lor \neg y_1^1), (y_1^1 \lor \neg y_2), (y_1^0 \lor y_2)$ is a resolution $\{y_1^1, y_2\}$-path with connecting sequence y_1^1, y_2 showing $C_2 \xleftrightarrow[\text{rp}]{\{y_1^1,y_2\}} C_3$.

Mapping the clauses of the universal expansion ψ' back to the corresponding original clauses from ψ turns (1) the first path into $(x_1 \lor x_2 \lor \neg y_1)$ (again with empty connecting sequence) and (2) the second path into $(\neg x_1 \lor \neg x_2 \lor \neg y_1)$, $(\neg x_2 \lor y_1 \lor \neg y_2)$, $(x_2 \lor y_1 \lor y_2)$ with connecting sequence y_1, y_2.

These two paths are resolution Z_{x_1}-paths for ψ with $Z_{x_1} = \{z \in V_\psi^\exists \mid x_1 \in D_z\} = \{y_1, y_2\}$. They prove $(x_1, y_1) \in \text{rqdep}^{\text{rp}}(\psi)$.

3.1 Monotonicity of Dependency Schemes

Monotonicity of dependency schemes has been considered before for QBF [23,26]. In general, dependency schemes are not monotone [23,26]. Monotone dependency schemes have the advantage that removing pseudo-dependencies identified by that scheme has a unique fixed point, i.e., the order of removal has no influence on the final result.

Definition 9 (Monotone). *Let* ds *be a dependency scheme. It is called* monotone *if for all DQBFs ψ, variables $x^* \in V_\psi^\forall$ and $y^* \in V_\psi^\exists$ such that $(x^*, y^*) \notin \text{ds}(\psi)$ the condition $\text{ds}\big(\psi \ominus (x^*, y^*)\big) \subseteq \text{ds}\big(\psi\big)$ is satisfied.*

That means, a dependency scheme is monotone if removing a pseudo-dependency from a DQBF never turns a pseudo-dependency into a proper one.

All dependency schemes which are sound for DQBF in the sense that they can be used to remove pseudo-dependencies have the nice property of monotonicity:

Theorem 4. *All dependency schemes in Δ^{dqbf} are monotone.*

Monotonicity simply follows from the definition of 'connected' (Definition 6) and 'resolution path connected' (Definition 7): Removing a pseudo-dependency (x^*, y^*) means removing x^* from the dependency set of y^*. Let $\text{ds} \in \Delta^{\text{dqbf}}$. For $(x^{**}, y^{**}) \in \text{ds}\big(\psi \ominus (x^*, y^*)\big)$, the existence of certain (resolution) $Z_{x^{**}}$-paths from clauses containing x^{**} or $\neg x^{**}$ to clauses containing y^{**} or $\neg y^{**}$ is needed with $Z_{x^{**}} = \big\{y \in V_\psi^\exists \;\big|\; x^{**} \in D_y\big\}$. Since the dependency sets of $\psi \ominus (x^*, y^*)$ can only be smaller than or equal to the dependency sets of ψ, the corresponding paths are valid for proving $(x^{**}, y^{**}) \in \text{ds}\big(\psi\big)$ as well.

3.2 Adding Dependencies

For DQBFs, it may not only be beneficial to remove dependencies from the dependency set of an existential variable, but also to add dependencies in order to make the formula more QBF-like. Consider, for instance, the formula

$\forall x_1 \forall x_2 \exists y_1(x_1) \exists y_2(x_2) : \varphi$ with some matrix φ. It can either be turned into a QBF by removing one of the dependencies or by adding one dependency.

Dependency schemes can be used to add pseudo-dependencies. The intuition is that one may add a dependency (x^*, y^*) according to a dependency scheme ds if ds allows to remove the dependency afterwards.

Lemma 2. *Let* $\mathrm{ds} \in \Delta^{\mathrm{dqbf}}$ *be a dependency scheme,* ψ *a DQBF, and* $(x^*, y^*) \in V_\psi^\forall \times V_\psi^\exists$ *such that* $x^* \notin D_{y^*}$. *If* $(x^*, y^*) \notin \mathrm{ds}\big(\psi \oplus (x^*, y^*)\big)$, *then* ψ *and* $\psi \oplus (x^*, y^*)$ *are equisatisfiable.*

Proof. If $(x^*, y^*) \notin \mathrm{ds}\big(\psi \oplus (x^*, y^*)\big)$, then $\psi \oplus (x^*, y^*)$ and $\big(\psi \oplus (x^*, y^*)\big) \ominus (x^*, y^*) = \psi$ are equisatisfiable. □

We can show that even *adding* dependencies has a unique fixed point. The reason for this is not as simple as for removing pseudo-dependencies in Sect. 3.1 and needs a slightly refined analysis. It relies on the following lemma which says that the order in which dependencies are added does not matter.

Lemma 3. *Let* $\mathrm{ds} \in \Delta^{\mathrm{dqbf}}$ *be a dependency scheme. Additionally, for* $i = 1, 2$, *let* $x_i \in V_\psi^\forall$, $y_i \in V_\psi^\exists$ *such that* $x_i \notin D_{y_i}$ *and* $(x_i, y_i) \notin \mathrm{ds}\big(\psi \oplus (x_i, y_i)\big)$. *Then* $(x_2, y_2) \notin \mathrm{ds}\big(\psi \oplus (x_1, y_1) \oplus (x_2, y_2)\big)$.

Proof. W. l. o. g. we assume that ds = sdep and define the following abbreviations:

$$\psi_1 := \psi \oplus (x_1, y_1), \quad \psi_2 := \psi \oplus (x_2, y_2), \quad \psi_{12} := \psi \oplus (x_1, y_1) \oplus (x_2, y_2).$$

Now we distinguish the following two cases:

- **Case 1:** $x_1 \neq x_2$.
  Assume that $(x_2, y_2) \in \mathrm{sdep}(\psi_{12})$. That means, there are clauses $C, C' \in \varphi$ such that $x_2 \in \mathrm{var}(C)$, $y_2 \in \mathrm{var}(C')$ and $C \xleftrightarrow{Z_{\psi_{12}}^{x_2}} C'$ where $Z_{\psi_{12}}^{x_2} = \{y \in V_\psi^\exists \mid x_2 \in D_y^{\psi_{12}}\}$. Since $x_2 \in D_y^{\psi_{12}}$ iff $x_2 \in D_y^{\psi_2}$, we have $C \xleftrightarrow{Z_{\psi_2}^{x_2}} C'$ with $Z_{\psi_2}^{x_2} = \{y \in V_\psi^\exists \mid x_2 \in D_y^{\psi_2}\}$. This in turn means that $(x_2, y_2) \in \mathrm{sdep}(\psi_2)$, which contradicts the assumption of the lemma.
- **Case 2:** $x_1 = x_2$.
  Let $x := x_1 = x_2$. We have $(x, y_i) \notin \mathrm{sdep}(\psi_i)$ for $i = 1, 2$. Assume that $(x, y_2) \in \mathrm{sdep}(\psi_{12})$.
  $(x, y_2) \in \mathrm{sdep}(\psi_{12})$ means there are clauses $C, C' \in \varphi$ such that $x \in \mathrm{var}(C)$, $y_2 \in \mathrm{var}(C')$ and $C \xleftrightarrow{Z_{\psi_{12}}^{x}} C'$, $Z_{\psi_{12}}^{x} = \{y \in V_\psi^\exists \mid x \in D_y^{\psi_{12}}\}$. Say the $Z_{\psi_{12}}^{x}$-path proving $C \xleftrightarrow{Z_{\psi_{12}}^{x}} C'$ is $C = C^{(1)}, C^{(2)}, \ldots, C^{(n)} = C'$ with the connecting sequence $y^{(1)}, y^{(2)}, \ldots, y^{(n-1)}$ and assume w. l. o. g. that there is no $1 \leq i < n$ with $y^{(i)} = y_2$ (otherwise we simply shorten the path). As $(x, y_2) \notin \mathrm{sdep}(\psi_2)$, the connecting sequence has to contain y_1, i. e., $y^{(i)} = y_1$ for some $1 \leq i < n$. That means, $C \xleftrightarrow{Z_{\psi_1}^{x}} C^{(i)}$ with $x \in \mathrm{var}(C)$ and $y_1 \in \mathrm{var}(C^{(i)})$, $Z_{\psi_1}^{x} = \{y \in V_\psi^\exists \mid x \in D_y^{\psi_1}\}$. Therefore $(x, y_1) \in \mathrm{sdep}(\psi_1)$, which is a contradiction.

The same argumentation can be applied for all dependency schemes in Δ^{dqbf}, replacing "connected" by "resolution path connected" where appropriate. $\square$

Corollary 1. *Adding as many dependencies as possible according to* $\text{ds} \in \Delta^{\text{dqbf}}$ *leads to a unique result, irrespective of the order.*

3.3 Manipulation of Dependencies Using Functional Definitions

CNFs (especially those derived from circuit representations) may contain so-called functional definitions. These are clauses which are logically equivalent to a formula $y \equiv f(v_1, \dots, v_k)$ with an existential variable y, an arbitrary boolean function f and (universal or existential) variables v_i $(1 \le i \le k)$. Here y is called the *defined variable*, f is the *defn* of y, and the clauses corresponding to the relationship $y \equiv f(v_1, \dots, v_k)$ are also called the *defining clauses*.

As already mentioned in [28], the detection of functional definitions provides a method for manipulating dependency sets which is completely orthogonal to the dependency schemes presented so far: If the CNF contains defining clauses for $y \equiv f(v_1, \dots, v_k)$ and $\bigcup_{i=1}^{k} \text{dep}(v_i) \subsetneq D_y$,[3] then D_y can be replaced by $\bigcup_{i=1}^{k} \text{dep}(v_i)$, by $V_\psi^\forall$, or by any set in between without changing the truth value of the DQBF, since any assignment to the universal variables from $\bigcup_{i=1}^{k} \text{dep}(v_i)$ already fixes the only value of y which is able to satisfy the defining clauses.

The manipulation of dependencies using functional definitions can be combined with the manipulation of dependencies using dependency schemes:

If the goal is to remove as many dependencies as possible, then we first remove dependencies based on functional definitions. Then, the dependency sets are already reduced when we start computing dependency schemes. Since this reduces the number of existing (resolution) Z-paths, this leads to smaller dependency schemes and thus to the detection of more pseudo-dependencies.

If the goal is to add as many dependencies as possible, we proceed the other way round: We make use of dependency schemes first and then add dependencies due to functional definitions. If we started by adding dependencies due to functional definitions, then the potential to add dependencies due to dependency schemes would be reduced with the same argumentation as above.

4 Experiments

We have implemented all of the presented dependency schemes that are sound for DQBF as well as the detection of functional definitions.[4] For each dependency scheme ds, we used an implementation which needs linear run-time in the size of the graphs representing $\overset{Z}{\longleftrightarrow}$ / $\underset{\text{rp}}{\overset{Z}{\longleftrightarrow}}$ for computing whether $(x, y) \in \text{ds}(\psi)$ as in [25]. Our tool supports both adding and removing pseudo-dependencies. We want to compare the effectiveness of the different dependency schemes.

[3] For notational convenience, we use here $\text{dep}(v_i) = D_{v_i}$ if v_i is existential and $\text{dep}(v_i) = \{v_i\}$ if v_i is universal.

[4] Our tool and all benchmarks we used are available at https://projects.informatik.uni-freiburg.de/projects/dqbf.

Table 1. Effectiveness of the different dependency schemes (f. d. = extraction of functional definitions)

Benchmark	f. d.?	Original	sdep	sdep$^{\mathrm{rp}}$	ssdep	ssdep$^{\mathrm{rp}}$	rtdep	rtdep$^{\mathrm{rp}}$	rqdep	rqdep$^{\mathrm{rp}}$
		adding dependencies								
pec_adder_n_bit_9_7	no	19751	123628	123628	123628	123628	123628	123628	123628	123628
	yes	113805	124038	124038	124038	124038	124038	124038	124038	124038
term1_0.50_1.00_3_1	no	48678	48730	48730	48745	48745	48730	48866	48745	48869
	yes	48768	48730	48730	48745	48745	48730	48866	48745	48869
C432_0.20_1.00_5_3	no	32560	32649	32960	32649	32965	32649	32996	32649	33001
	yes	32560	32649	32960	32649	32965	32649	32996	32649	33001
		removing dependencies								
genbuf15f14unrealy	no	198412	198412	198412	198412	198412	198412	198412	198412	198412
	yes	60873	60873	60873	60873	60873	60873	60873	60873	60873
term1_0.50_1.00_3_1	no	48678	32570	32506	29683	29623	32570	32234	29683	29375
	yes	41061	27543	27479	25142	25082	27543	27207	25142	24834
C432_0.20_1.00_5_3	no	32560	28397	22915	28389	22883	28397	21362	28389	21346
	yes	32264	28109	22627	28101	22595	28109	21074	28101	21058

All experiments were run on one Intel Xeon E5-2650v2 CPU core at 2.60 GHz clock frequency and 64 GB of main memory under Ubuntu Linux as operating system. As benchmarks we used 4811 DQBF instances from different sources. They encompass the 4381 instances already used in [28]: DQBFs resulting from equivalence checking of incomplete circuits [13,15,17] and controller synthesis problems [6]. We have added 34 instances which were obtained from converting SAT instances into DQBFs that depend only on a logarithmic number of variables [3]. Additionally we used 396 instances resulting from partial equivalence checking problems [17,24].

We applied detection of functional definitions and then the reflexive quadrangle resolution path dependency scheme rqdep$^{\mathrm{rp}}$ to remove as many dependencies as possible. The same was done for adding dependencies, using first rqdep$^{\mathrm{rp}}$ and then detection of functional definitions. For all instances, our tool terminated within fractions of a second and used less than 50 MB of main memory.

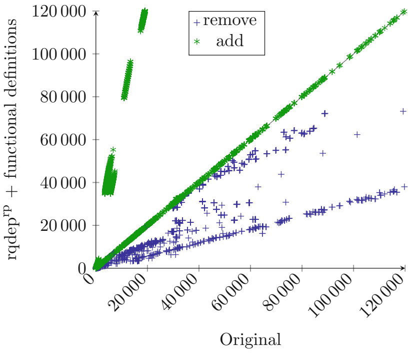
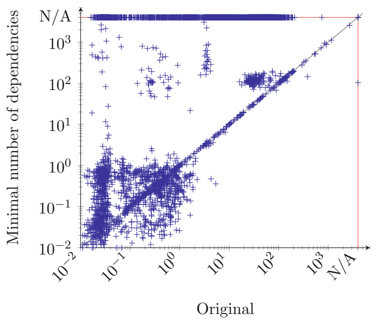


Fig. 2. Effectiveness of dependency manipulation using rqdep$^{\mathrm{rp}}$ and detection of functional definitions.(Color figure online)

Figure 2 shows the results. We compare the original formula with the formula both after the removal (blue crosses) and the addition (green stars) of pseudo-dependencies. For a fixed instance, a mark on the diagonal means that the dependency set could not be modified, above that it could be increased and below that it could be reduced. We can observe that almost all



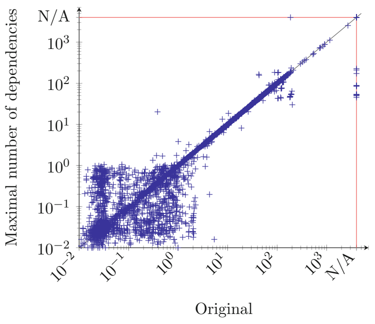


Fig. 3. Running times of HQS [18] compared to the original benchmark after removing (left) and adding (right) dependencies using rqdep$^{\mathrm{rp}}$ and detecting functional definitions. "N/A" means that the benchmark could not be solved within 3600 s computation time and 8 GB of memory.

instances could be modified; in some cases, the size of the maximal dependency set is ten times the size of the minimal one.

Table 1 shows the numbers of dependencies for some selected instances before and after modifying the dependency sets using the different dependency schemes. We can observe that they are not equally powerful and that the detection of functional definitions can amplify the effectiveness of the dependency schemes. Since the computation times for rqdep$^{\mathrm{rp}}$ are in the same range as the computation times for the other dependency schemes, we propose to use rqdep$^{\mathrm{rp}}$ together with the detection of functional definitions.

Our work demonstrates the potential of adding/removing dependencies and lays the groundwork for exploiting this flexibility later on. Figure 3 demonstrates that trivial solutions (just adding or removing dependencies) for exploiting the flexibility do not help much. It shows the solution times of HQS [18] on all 4811 instances after applying rqdep$^{\mathrm{rp}}$ and detection of functional definitions compared to the original benchmark. The left image is for removing dependencies, the right one for adding dependencies. We can see that manipulating the dependency sets can have a huge effect on the solution times. They can both increase and decrease. This is not really surprising, since both adding and removing dependencies may increase or decrease the 'distance' of the DQBF from an equisatisfiable QBF. Some of the benchmarks even became solvable or unsolvable within the resource limits of 3600 s computation time and 8 GB of memory. These results show that a clever way for exploiting the flexibility is definitely needed to solve more instances in less time.

5 Conclusion

We have presented a complete characterization of those dependency schemes which can be generalized from QBF to DQBF and those which cannot. The generalizations are suitable to remove and add dependencies in DQBFs. Both adding and removing dependencies lead to a unique fixed point, irrespective of the order of adding/removing and can be combined with an orthogonal method based on functional definitions. First experimental results show that the presented methods give an enormous amount of flexibility for the manipulation of dependency sets and each method has its own contributions to the overall flexibility. A central task for future research is to develop appropriate heuristics for exploiting the flexibility in the dependency sets in order to turn a DQBF into a QBF at lower costs. The groundwork for such a method has been laid by the presented paper.

Acknowledgments. We thank the anonymous reviewers for their really helpful comments.

References

1. Ashar, P., Ganai, M.K., Gupta, A., Ivancic, F., Yang, Z.: Efficient SAT-based bounded model checking for software verification. In: Margaria, T., Steffen, B., Philippou, A., Reitenspieß, M. (eds.) International Symposium on Leveraging Applications of Formal Methods (ISoLA). Technical Report, vol. TR-2004-6, pp. 157–164, Department of Computer Science, University of Cyprus, Paphos, Cyprus, October 2004
2. Balabanov, V., Chiang, H.K., Jiang, J.R.: Henkin quantifiers and boolean formulae: a certification perspective of DQBF. Theor. Comput. Sci. **523**, 86–100 (2014)
3. Balabanov, V., Jiang, J.H.R.: Reducing satisfiability and reachability to DQBF, September 2015. Talk at the International Workshop on Quantified Boolean Formulas (QBF)
4. Biere, A.: Resolve and expand. In: H. Hoos, H., Mitchell, D.G. (eds.) SAT 2004. LNCS, vol. 3542, pp. 59–70. Springer, Heidelberg (2005)
5. Biere, A., Cimatti, A., Clarke, E.M., Strichman, O., Zhu, Y.: Bounded model checking. Adv. Comput. **58**, 117–148 (2003)
6. Bloem, R., Könighofer, R., Seidl, M.: SAT-based synthesis methods for safety specs. In: McMillan, K.L., Rival, X. (eds.) VMCAI 2014. LNCS, vol. 8318, pp. 1–20. Springer, Heidelberg (2014)
7. Bubeck, U.: Model-based transformations for quantified boolean formulas. Ph.D. thesis, University of Paderborn (2010)
8. Bubeck, U., Büning, H.K.: Dependency quantified horn formulas: models and complexity. In: Biere, A., Gomes, C.P. (eds.) SAT 2006. LNCS, vol. 4121, pp. 198–211. Springer, Heidelberg (2006)
9. Bubeck, U., Kleine Büning, H.: Bounded universal expansion for preprocessing QBF. In: Marques-Silva, J., Sakallah, K.A. (eds.) SAT 2007. LNCS, vol. 4501, pp. 244–257. Springer, Heidelberg (2007)
10. Clarke, E.M., Biere, A., Raimi, R., Zhu, Y.: Bounded model checking using satisfiability solving. Formal Methods Syst. Design **19**(1), 7–34 (2001)
11. Czutro, A., Polian, I., Lewis, M.D.T., Engelke, P., Reddy, S.M., Becker, B.: Thread-parallel integrated test pattern generator utilizing satisfiability analysis. Int. J. Parallel Prog. **38**(3–4), 185–202 (2010)
12. Eggersglüß, S., Drechsler, R.: A highly fault-efficient SAT-based ATPG flow. IEEE Des. Test Comput. **29**(4), 63–70 (2012)

13. Finkbeiner, B., Tentrup, L.: Fast DQBF refutation. In: Sinz, C., Egly, U. (eds.) SAT 2014. LNCS, vol. 8561, pp. 243–251. Springer, Heidelberg (2014)
14. Fröhlich, A., Kovásznai, G., Biere, A.: A DPLL algorithm for solving DQBF. In: International Workshop on Pragmatics of SAT (POS), Trento, Italy (2012)
15. Fröhlich, A., Kovásznai, G., Biere, A., Veith, H.: iDQ: instantiation-based DQBF solving. In: Berre, D.L. (ed.) International Workshop on Pragmatics of SAT (POS). EPiC Series, vol. 27, pp. 103–116. EasyChair, Vienna, Austria. http://www.easychair.org/publications/?page=2037484173
16. Van Gelder, A.: Variable independence and resolution paths for quantified boolean formulas. In: Lee, J. (ed.) CP 2011. LNCS, vol. 6876, pp. 789–803. Springer, Heidelberg (2011)
17. Gitina, K., Reimer, S., Sauer, M., Wimmer, R., Scholl, C., Becker, B.: Equivalence checking of partial designs using dependency quantified Boolean formulae. In: Proceedings of ICCD, pp. 396–403. IEEE CS, Asheville, October 2013
18. Gitina, K., Wimmer, R., Reimer, S., Sauer, M., Scholl, C., Becker, B.: Solving DQBF through quantifier elimination. In: Proceedings of DATE. IEEE, Grenoble, March 2015
19. Lonsing, F., Biere, A.: Efficiently representing existential dependency sets for expansion-based QBF solvers. ENTCS **251**, 83–95 (2009)
20. Lonsing, F., Biere, A.: Integrating dependency schemes in search-based QBF solvers. In: Strichman, O., Szeider, S. (eds.) SAT 2010. LNCS, vol. 6175, pp. 158–171. Springer, Heidelberg (2010)
21. Peterson, G., Reif, J., Azhar, S.: Lower bounds for multiplayer non-cooperative games of incomplete information. Comput. Math. Appl. **41**(7–8), 957–992 (2001)
22. Rintanen, J., Heljanko, K., Niemelä, I.: Planning as satisfiability: parallel plans and algorithms for plan search. Artif. Intell. **170**(12–13), 1031–1080 (2006)
23. Samer, M., Szeider, S.: Backdoor sets of quantified Boolean formulas. J. Autom. Reasoning **42**(1), 77–97 (2009)
24. Scholl, C., Becker, B.: Checking equivalence for partial implementations. In: ACM/IEEE Design Automation Conference (DAC), pp. 238–243. ACM Press, Las Vegas, June 2001
25. Slivovsky, F., Szeider, S.: Computing resolution-path dependencies in linear time. In: Cimatti, A., Sebastiani, R. (eds.) SAT 2012. LNCS, vol. 7317, pp. 58–71. Springer, Heidelberg (2012)
26. Slivovsky, F., Szeider, S.: Quantifier reordering for QBF. J. Autom. Reasoning **56**(4), 459–477 (2016)
27. Slivovsky, F., Szeider, S.: Soundness of Q-resolution with dependency schemes. Theor. Comput. Sci. **612**, 83–101 (2016)
28. Wimmer, R., Gitina, K., Nist, J., Scholl, C., Becker, B.: Preprocessing for DQBF. In: Heule, M., et al. (eds.) SAT 2015. LNCS, vol. 9340, pp. 173–190. Springer, Heidelberg (2015). doi:10.1007/978-3-319-24318-4_13
29. Wimmer, R., Gitina, K., Nist, J., Scholl, C., Becker, B.: Preprocessing for DQBF. Reports of SFB/TR 14 AVACS 110. http://www.avacs.org