On the Complexity of k-DQBF

Long-Hin Fung
Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan

Tony Tan
Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan

Abstract

Recently *Dependency Quantified Boolean Formula* (DQBF) has attracted a lot of attention in the SAT community. Intuitively, a DQBF is a natural extension of quantified boolean formula where for each existential variable, one can specify the set of universal variables it depends on. It has been observed that a DQBF with k existential variables – henceforth denoted by k-DQBF – is essentially a k-CNF formula in succinct representation. However, beside this and the fact that the satisfiability problem is NEXP-complete, not much is known about DQBF.

In this paper we take a closer look at k-DQBF and show that a number of well known classical results on k-SAT can indeed be lifted to k-DQBF, which shows a strong resemblance between k-SAT and k-DQBF. More precisely, we show the following.

(a) The satisfiability problem for 2- and 3-DQBF is PSPACE- and NEXP-complete, respectively.
(b) There is a parsimonious polynomial time reduction from arbitrary DQBF to 3-DQBF.
(c) Many polynomial time projections from SAT to languages in NP can be lifted to polynomial time reductions from the satisfiability of DQBF to languages in NEXP.
(d) Languages in the class $\mathsf{NSPACE}[s(n)]$ can be reduced to the satisfiability of 2-DQBF with $O(s(n))$ universal variables.
(e) Languages in the class $\mathsf{NTIME}[t(n)]$ can be reduced to the satisfiability of 3-DQBF with $O(\log\ t(n))$ universal variables.

The first result parallels the well known classical results that 2-SAT and 3-SAT are NL- and NP-complete, respectively.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Dependency quantified boolean formulas, existential variables, complexity

Digital Object Identifier 10.4230/LIPIcs.SAT.2023.10

Funding We acknowledge the generous financial support of Taiwan National Science and Technology Council under grant no. 109-2221-E-002-143-MY3.

Acknowledgements We would like to thank Roland Jiang Jie-Hong for many useful and insightful discussions as well as the anonymous referees for their constructive feedback.

1 Introduction

The last few decades have seen a tremendous development of boolean SAT solvers and their applications in many areas of computing [4]. Motivated by applications in hardware verification and synthesis [19, 3, 29, 17, 5, 7, 22, 16], there have been attempts to build efficient solvers for even higher complexity class such as NEXP. One NEXP-complete logic that recently has attracted a lot of attention is *Dependency Quantified Boolean Formulas* (DQBF). Intuitively, DQBF is a natural extension of Quantified Boolean Formula (QBF) where for each existential variable, one can specify the set of universal variables that it depends on.

It has been observed by various researchers that a DQBF is essentially a succinctly represented boolean formula in conjunctive normal form (CNF), where the information about each clause is encoded inside the matrix of the DQBF and that the number of existential variables in a DQBF corresponds precisely to the width of the clauses of the CNF formula it represents. Such succinctness makes the the satisfiability of DQBF jumps to NEXP-complete [26, 8], compared to "only" NP-complete for SAT. However, beside these facts, not much is known about DQBF and it is natural to ask whether there are more resemblances between DQBF and CNF formulas.

In this paper we show that some well known classical results on SAT can indeed be lifted to DQBF – only with exponential blow-up in complexity due to the succinctness of DQBF. To be more precise, for an integer $k \geqslant 1$, let k-DQBF denote a DQBF with k existentially quantified variables. We establish the following.

(a) The satisfiability of k-DQBF where $k = 1, 2, 3$ is coNP-complete, PSPACE-complete and NEXP-complete, respectively.

(b) There is a parsimonious polynomial time reduction from arbitrary DQBF to 3-DQBF.

Note that (a) parallels the well known classical complexity results on 2-SAT and 3-SAT, i.e., NL-complete and NP-complete and (b) parallels the well known parsimonious polynomial time reduction from an arbitrary boolean formula to a 3-CNF formula.

The fact that DQBF is a succinct representation of CNF formulas actually has the same flavour as the succinct representation of graphs with boolean circuits [14]. In such representation, instead of being given the list of edges in a graph, we are given a boolean circuit $C(\bar{x}, \bar{y})$ where $\bar{x}, \bar{y}$ are vectors of boolean variables with length n. The circuit C represents a graph with $\{0,1\}^n$ being the set of vertices and two vertices $\bar{u}$ and $\bar{v}$ are adjacent if and only if $C(\bar{u}, \bar{v}) = 1$. It is shown in [25] that many natural NP-complete graph problems become NEXP-complete when succinctly represented. We observe that the proof in [25] can be modified to obtain reductions from DQBF in the following sense.

(c) If there is a projection (in the sense of [33]) from SAT to a graph problem Π, then there is a polynomial time (Karp) reduction from DQBF to the succinctly represented Π.

Briefly, a projection is a special kind of polynomial time reductions first introduced in [33] and it is known that many reductions from SAT to various NP-complete problems are in fact projections [25]. Intuitively, we can view (c) as lifting the reductions from SAT in the class NP to the reductions from DQBF in the class NEXP.

We also observe that DQBF can be used to describe the languages in $\mathsf{NTIME}[t(n)]$ and $\mathsf{NSPACE}[s(n)]$. More precisely, we show the following.

(d) Every language in the class $\mathsf{NTIME}[t(n)]$ can be reduced to 3-DQBF instances with $O(\log t(n))$ universal variables.

(e) Every language in the class $\mathsf{NSPACE}[s(n)]$ can be reduced to 2-DQBF instances with $O(s(n))$ universal variables.

Note that (d) parallels the reductions from the languages in $\mathsf{NTIME}[t(n)]$ to SAT instances with $O(t(n) \log t(n))$ variables and (e) parallels the reductions from the languages in $\mathsf{NSPACE}[s(n)]$ to QBF instances with $O(s(n)^2)$ variables.

Finally, it is open whether there is a natural *bona fide* problem in the class NEXP [25]. In addition to DQBF being a natural extension of QBF and SAT, results (a)–(c) exhibit a strong resemblance between DQBF and CNF formula. Moreover, (d) and (e) show that DQBF can be used to describe both the classes $\mathsf{NTIME}[t(n)]$ and $\mathsf{NSPACE}[s(n)]$, as opposed to the classical results where we need two different logics QBF and SAT to describe them. Combined with the work in [8], we hope they can be convincing evidences for DQBF to be the *bona fide* problem in NEXP, just like SAT in NP and QBF in PSPACE.

Related works. That DQBF is NEXP-complete is proved in [26] and the hardness proof uses an unbounded number of existential variables. It is recently improved in [8] where it is shown that 4 existential variables are sufficient to achieve NEXP-hardness.

Many powerful and interesting techniques have been developed in the last decade for solving DQBF. See, e.g., [2, 12, 15, 23, 39, 36, 38, 21, 31] and the references within. Some recent solvers include iDQ [13], dCAQE [35], HQS [18, 37], DQBDD [32] and Pedant [28, 27]. Recently there is also a DQBF track in the annual SAT competitions [1].

As mentioned earlier, various researchers have observed that a DQBF is essentially a succinctly represented CNF formula, which can be established by *the universal expansion* [6, 13, 3]. Briefly, it removes the universal variables one by one by considering both its values 0 and 1 separately resulting in an exponentially long boolean formula [6]. Based on this insight, some DQBF benchmarks can be constructed by encoding succinctly graph reachability instances and SAT instances [3]. The solver iDQ in [13] is based on universal expansion with additional refinement strategies that remove unnecessary clauses.

A natural extension of QBF to second-order logic that captures the exponential hierarchy is studied in [9, 20]. Naturally a DQBF can be viewed as an existential second-order boolean formula. However, [9, 20] do not study the precise complexity and expressiveness of DQBF itself. In [30] a characterization of a PSPACE subclass of DQBF is introduced. It requires that the dependency sets of all the existential variables are either the same or disjoint and the matrix is in CNF. A close examination shows that it is a Σ_3^p subclass of DQBF, though the precise complexity is still unknown. This subclass is orthogonal to the one in this paper which is based on the number existential variables.

The celebrated Cook-Levin reductions [10, 24] show that every language in $\mathsf{NTIME}[t(n)]$ can be reduced to CNF formulas with $O(t(n)^2)$ variables. The bound on the number of variables was later improved in [11] to $O(t(n)\log t(n))$. The reduction from languages in $\mathsf{NSPACE}[s(n)]$ to QBF with $O(s(n)^2)$ variables is from [34]. In [8] the notion called succinct projection is introduced as a general method to reduce various natural NEXP-complete problems to DQBF. It is the analogue of the Cook-Levin reductions for the class NEXP. Our result (c) shows that the reductions in [25] can be viewed as the "converse" reductions of the ones in [8].

The succinct representation of graphs with boolean circuits was first introduced in [14]. As mentioned earlier, under such representation, many NP-complete problems become NEXP-complete and NL-complete problems become PSPACE-complete [25], using the notion of projections introduced in [33]. To the best of our knowledge, this is the only known technique to lift the results from the class NL and NP to the class PSPACE and NEXP.

Organisation. We first introduce some useful notations and the formal definition of DQBF in Section 2. In Section 3 we study the complexity of DQBF based on the number of existential variables. We discuss how to lift the reductions for NP-complete problems to the class NEXP in Section 4. Then, in Section 5 we show the relations between the class $\mathsf{NTIME}[t(n)]$ and $\mathsf{NSPACE}[s(n)]$ and DQBF. Finally, we conclude with Section 6.

2 Preliminaries

Notations. In this paper we let $\Sigma = \{0, 1\}$. We use 0 and 1 to represent the boolean values *false* and *true*, respectively. We will use the symbol a, b, c (possibly indexed) to denote an element in Σ and $\bar{a}, \bar{b}, \bar{c}$ (possibly indexed) to denote a string in Σ^* with $|\bar{a}|$ denoting the length of $\bar{a}$. To avoid clutter, tuples of values from Σ will be written as strings. For example, instead of $(1, 0, 1, 1)$, we will simply write 1011.

We use x, y, z, u, v (possibly indexed) to denote boolean variables and the bar version $\bar{x}, \bar{y}, \bar{z}, \bar{u}, \bar{v}$ (possibly indexed) to denote vectors of boolean variables with $|\bar{x}|$ denoting the length of $\bar{x}$. We implicitly assume that in a vector $\bar{x}$ there is no variable occurring more than once. We write $\bar{z} \subseteq \bar{x}$ when every variable in $\bar{z}$ also occurs in $\bar{x}$.

We write $\varphi(\bar{x})$ to denote a (boolean) formula/circuit with variables/input gates $\bar{x}$. When the variables/input gates are not relevant or clear from the context, we simply write φ.

Let $\varphi(\bar{x})$ be a formula/circuit where $\bar{x} = (x_1, \ldots, x_n)$. Let $\bar{z} = (z_1, \ldots, z_n)$. We write $\varphi[\bar{x}/\bar{z}]$ to denote the formula obtained by substituting each x_i with z_i simultaneously for each $1 \leqslant i \leqslant n$. For a string $\bar{a} = (a_1, \ldots, a_n) \in \Sigma^n$, we write $\varphi[\bar{x} \mapsto \bar{a}]$ to denote the evaluation value of φ when we assign each x_i with a_i.

For $\bar{z} \subseteq \bar{x}$ and $\bar{a} \in \Sigma^{|\bar{x}|}$, we write $\mathsf{prj}_{\bar{x}}(\bar{z}, \bar{a})$ to denote the projection of $\bar{a}$ to the components in $\bar{z}$ according to the order of the variables in $\bar{x}$. For example, if $\bar{x} = (x_1, \ldots, x_5)$ and $\bar{z} = (x_2, x_4, x_5)$, then $\mathsf{prj}_{\bar{x}}(\bar{z}, 00101)$ is 001, i.e., the projection of 00101 to its 2^{nd}, 4^{th} and 5^{th} bits. Note that if $\bar{a} = (a_1, \ldots, a_n)$ and $\bar{x} = (x_1, \ldots, x_n)$, $\mathsf{prj}_{\bar{x}}(x_i, \bar{a})$ is the value a_i.

Dependency quantified boolean formula. A *dependency quantified boolean formula* (DQBF) in prenex normal form is a formula of the form:

$$\Phi := \forall x_1 \cdots \forall x_n \; \exists y_1(\bar{z}_1) \cdots \exists y_k(\bar{z}_k) \quad \phi \tag{1}$$

where each $\bar{z}_i \subseteq (x_1, \ldots, x_n)$ and ϕ, called *the matrix*, is a quantifier-free boolean formula using variables $x_1, \ldots, x_n, y_1, \ldots, y_k$. The variables $x_1, \ldots, x_n$ are called *the universal variables*, $y_1, \ldots, y_k$ *the existential variables* and each $\bar{z}_i$ *the dependency set* of y_i. We call Φ a k-DQBF, where k is the number of existential variables in Φ. To avoid clutter, sometimes we write Φ as: $\Phi := \forall \bar{x} \; \exists y_1(\bar{z}_1) \cdots \exists y_k(\bar{z}_k) \; \phi$, where $\bar{x} = (x_1, \ldots, x_n)$.

A DQBF Φ in the form (1) is *satisfiable*, if there is a tuple $(f_1, \ldots, f_k)$ of functions where $f_i : \Sigma^{|\bar{z}_i|} \to \Sigma$ for every $1 \leqslant i \leqslant k$, and by replacing each y_i with $f_i(\bar{z}_i)$, the formula ϕ becomes a tautology. The tuple $(f_1, \ldots, f_k)$ is called the *satisfying Skolem functions* for Φ. In this case, we also say that Φ is satisfiable by the Skolem functions $(f_1, \ldots, f_k)$.

The problem sat(DQBF) is defined as follows. On input DQBF Φ in the form (1), decide if it is satisfiable. For $k \geqslant 1$, we denote by sat(k-DQBF) the restriction of sat(DQBF) on k-DQBF. As a language, sat(DQBF) := $\{\Phi \mid \Phi \text{ is a satisfiable DQBF}\}$ and sat(k-DQBF) := $\{\Phi \mid \Phi \text{ is a satisfiable } k\text{-DQBF}\}$.

▶ Remark 1. We may allow the matrix ϕ to be in a *circuit form*, i.e., it is given as a (boolean) circuit with input gates $x_1, \ldots, x_n, y_1, \ldots, y_k$. Such form does not effect the generality of our result since it can be converted to a standard formula form with additional universal variables, but without additional existential variables. See [8, Proposition 1].

▶ Remark 2. A DQBF can be seen a natural generalization of SAT and QBF. Indeed, a boolean formula with variables $y_1, \ldots, y_k$ can be seen as a DQBF without any universal variable and $y_1, \ldots, y_k$ are existential variables with empty dependency set. It is also easy to see that a QBF is just a DQBF where the dependency set form an ordering w.r.t. inclusion, i.e., $\bar{z}_1 \subseteq \bar{z}_2 \subseteq \cdots \subseteq \bar{z}_k$. Indeed, a QBF $\forall x_1 \exists y_1 \forall x_2 \exists y_2 \cdots \forall x_n \exists y_n \; \phi$ can be viewed as a DQBF $\forall x_1 \forall x_2 \cdots \forall x_n \exists y_1(x_1) \exists y_2(x_1, x_2) \cdots \exists y_n(x_1, \ldots, x_n) \; \phi$. Conversely, suppose we have a DQBF Φ as in (1), where $\bar{z}_1 \subseteq \bar{z}_2 \subseteq \cdots \subseteq \bar{z}_k$. Reordering the universal variables, we may assume that $\bar{z}_i = (x_1, \ldots, x_{j_i})$ for each $1 \leqslant i \leqslant k$, where $j_1 \leqslant j_2 \leqslant \cdots \leqslant j_k \leqslant n$. Thus, Φ can be rewritten as the QBF: $\forall x_1 \cdots \forall x_{j_1} \exists y_1 \forall x_{j_1+1} \cdots \forall x_{j_2} \exists y_2 \forall x_{j_2+1} \cdots \forall x_{j_k} \exists y_k \forall x_{j_k+1} \cdots \forall x_n \; \phi$.

Universal expansion. We briefly review the universal expansion method, a useful and well known method for showing that a k-DQBF essentially represents an exponentially large k-CNF formula [6, 13, 3]. Let Φ be k-DQBF as in (1). For each $\bar{a} \in \Sigma^n$, let $\varphi_{\bar{a}}$ be the following boolean formula/circuit.

$$\varphi_{\bar{a}} \;:=\; \phi\,[\bar{x}/\bar{a},\ \bar{y}/(f_1(\mathsf{prj}_{\bar{x}}(\bar{z}_1,\bar{a})),\ldots,f_k(\mathsf{prj}_{\bar{x}}(\bar{z}_k,\bar{a})))]$$

That is, the variables in $\bar{x}$ in the matrix ϕ are substituted with the values in $\bar{a}$ and each y_i with $f_i(\mathsf{prj}_{\bar{x}}(\bar{z}_i,\bar{a}))$. Treating each $f_i(\mathsf{prj}_{\bar{x}}(\bar{z}_i,\bar{a}))$ as a boolean variable, $\varphi_{\bar{a}}$ is a standard boolean formula/circuit with k variables and can be rewritten as a k-CNF formula, say, by building its truth table where each row (in the truth table) with 0 value is represented by one clause. By expanding the universal quantifiers, the DQBF Φ can be easily seen to be equivalent to:

$$\bigwedge_{\bar{a}\in\Sigma^n} \varphi_{\bar{a}},$$

where we assume each $\varphi_{\bar{a}}$ is already rewritten in k-CNF.

For our purpose in this paper it is not necessary to convert the entire $\varphi_{\bar{a}}$ into a k-CNF formula. We usually only need to extract one clause at a time from the formula $\varphi_{\bar{a}}$, which is facilitated by the following notation. For each $(\bar{a},\bar{b}) \in \Sigma^n \times \Sigma^k$, where $\bar{a} = (a_1,\ldots,a_n)$ and $\bar{b} = (b_1,\ldots,b_k)$, we define the clause $C_{\bar{a},\bar{b}}$ as $\ell_1 \vee \cdots \vee \ell_k$, where each literal ℓ_i is as follows.

$$\ell_i \;:=\; \begin{cases} f_i(\mathsf{prj}_{\bar{x}}(\bar{z}_i,\bar{a})) & \text{if } b_i = 0 \\ \neg f_i(\mathsf{prj}_{\bar{x}}(\bar{z}_i,\bar{a})) & \text{if } b_i = 1 \end{cases}$$

We call $C_{\bar{a},\bar{b}}$ *the clause associated with* $(\bar{a},\bar{b})$ and *the universal expansion of* Φ is defined as:

$$\exp(\Phi) \quad := \bigwedge_{(\bar{a},\bar{b})\in\Sigma^n\times\Sigma^k \text{ s.t. } \phi[(\bar{x},\bar{y})\mapsto(\bar{a},\bar{b})]=0} C_{\bar{a},\bar{b}}$$

Intuitively, $\phi[(\bar{x},\bar{y}) \mapsto (\bar{a},\bar{b})] = 0$ means that we are only interested in the row $\bar{a},\bar{b}$ that yields 0 in the truth table of ϕ. Since the clause $C_{\bar{a},\bar{b}}$ is defined precisely to represent such row 0, it is straightforward to see that $\exp(\Phi)$ is indeed equivalent to $\bigwedge_{\bar{a}\in\Sigma^n}\varphi_{\bar{a}}$, and hence, to Φ.

Alternatively, we can also define the expansion $\exp(\Phi)$ by the following simple rewriting rule. Let $\bar{x} = (x_1,\ldots,x_n)$ and $\bar{y} = (y_1,\ldots,y_k)$. With additional "fresh" k universal variables $\bar{v} = (v_1,\ldots,v_k)$, Φ is equivalent to the following DQBF Φ'.

$$\Phi' \;:=\; \forall\bar{x}\ \forall\bar{v}\ \exists y_1(\bar{z}_1)\cdots\ \exists y_k(\bar{z}_k) \qquad \Big(\bigwedge_{i=1}^{k} v_i = y_i\Big) \;\to\; \phi'$$

where ϕ' is $\phi[\bar{y}/\bar{v}]$, i.e., the formula obtained by simultaneously substituting each y_i with v_i in ϕ. Note that ϕ' no longer uses existentially quantified variables. By simple rewriting rule and the expansion on the universal quantifiers, we obtain:

$$\bigwedge_{(\bar{a},\bar{b})\in\Sigma^n\times\Sigma^k} \Big(\bigvee_{i=1}^{k}(\neg b_i \wedge f_i(\mathsf{prj}_{\bar{x}}(\bar{z}_i,\bar{a}))) \vee (b_i \wedge \neg f_i(\mathsf{prj}_{\bar{x}}(\bar{z}_i,\bar{a})))\Big) \;\vee\; \phi'[\bar{x},\bar{v}\mapsto\bar{a},\bar{b}] \qquad (2)$$

For each $(\bar{a}, \bar{b}) \in \Sigma^n \times \Sigma^k$, when $\phi'[\bar{x}, \bar{v} \mapsto \bar{a}, \bar{b}] = 1$, the conjunct already yields 1. Thus, (2) is equivalent to:

$$\bigwedge_{(\bar{a},\bar{b})\in\Sigma^n\times\Sigma^k \text{ s.t. } \phi'[\bar{x},\bar{v}\mapsto\bar{a},\bar{b}]=0} \Big(\bigvee_{i=1}^{k} (\neg b_i \wedge f_i(\mathsf{prj}_{\bar{x}}(\bar{z}_i, \bar{a}))) \vee (b_i \wedge \neg f_i(\mathsf{prj}_{\bar{x}}(\bar{z}_i, \bar{a}))) \Big)$$

Since b_i is either 0 or 1, exactly one of $\neg b_i \wedge f_i(\mathsf{prj}_{\bar{x}}(\bar{z}_i, \bar{a}))$ and $b_i \wedge \neg f_i(\mathsf{prj}_{\bar{x}}(\bar{z}_i, \bar{a}))$ evaluates to 0. Thus, it evaluates to either $f_i(\mathsf{prj}_{\bar{x}}(\bar{z}_i, \bar{a}))$ or $\neg f_i(\mathsf{prj}_{\bar{x}}(\bar{z}_i, \bar{a}))$. Therefore, the disjunction $\bigvee_{i=1}^{k}(\neg b_i \wedge f_i(\mathsf{prj}_{\bar{x}}(\bar{z}_i, \bar{a}))) \vee (b_i \wedge \neg f_i(\mathsf{prj}_{\bar{x}}(\bar{z}_i, \bar{a})))$ is equivalent to $C_{\bar{a},\bar{b}}$.

3 The complexity of sat(k-DQBF)

In this section we will study the complexity of $\mathsf{sat}(k\text{-DQBF})$ for $k = 1, 2, 3$. We will start with the case when $k = 1$ and 2 in Subsection 3.1. The case when $k = 3$ is presented in Subsection 3.2.

3.1 On sat(1-DQBF) and sat(2-DQBF)

We first establish that $\mathsf{sat}(1\text{-DQBF})$ is coNP-complete.

▶ **Theorem 3.** *$\mathsf{sat}(1\text{-DQBF})$ is coNP-complete.*

Proof. We note that checking whether a boolean formula is tautology is just a special case of $\mathsf{sat}(1\text{-DQBF})$ where the existential variable is not used. Since checking tautology is already coNP-hard, the same hardness for $\mathsf{sat}(1\text{-DQBF})$ follows immediately. To establish the coNP membership, note that a 1-CNF formula is not satisfiable if and only if it contains two contradicting literals. We will use the same idea for 1-DQBF.

Let Ψ be the following DQBF.

$$\Psi \;:=\; \forall\bar{x}\exists y(\bar{z})\; \psi, \qquad \text{where } \bar{x} = (x_1, \ldots, x_n) \tag{3}$$

It is straightforward to show that there are two contradicting literals in $\exp(\Psi)$ if and only if there is $\bar{a}, \bar{b} \in \Sigma^{n+1}$ such that:

1. $\mathsf{prj}_{(\bar{x},y)}(y, \bar{a}) \;\neq\; \mathsf{prj}_{(\bar{x},y)}(y, \bar{b})$.
2. $\mathsf{prj}_{(\bar{x},y)}(\bar{z}, \bar{a}) \;=\; \mathsf{prj}_{(\bar{x},y)}(\bar{z}, \bar{b})$.
3. $\psi[(\bar{x}, y) \mapsto \bar{a}] = \psi[(\bar{x}, y) \mapsto \bar{b}] = 0$.

The algorithm for accepting unsatisfiable 1-DQBF works as follows. On input Ψ as in (3), guess two assignments $\bar{a}, \bar{b}$ and accept if conditions 1–3 hold. ◀

Next we establish that $\mathsf{sat}(2\text{-DQBF})$ is PSPACE-complete.

▶ **Theorem 4.** *$\mathsf{sat}(2\text{-DQBF})$ is PSPACE-complete.*

Proof. The PSPACE-hardness is established by reduction from succinct 2-colorability, which is known to be PSPACE-complete [25]. The problem *succinct 2-colorability* is defined as: On input circuit C, decide if the graph represented by C is 2-colorable, i.e., there is coloring of the vertices with 3 colors such that no two adjacent vertices have the same color.

Let $C(\bar{u}, \bar{v})$ be the input circuit, where $|\bar{u}| = |\bar{v}| = n$. We may assume that $\{0, 1\}$ is the set of colors and view a coloring on the vertices as a function $f : \{0, 1\}^n \times \{0, 1\}$. Consider the following 2-DQBF.

$$\Psi \;:=\; \forall \bar{x}_1 \forall \bar{x}_2\; \exists y_1(\bar{x}_1) \exists y_2(\bar{x}_2) \quad \big(\bar{x}_1 = \bar{x}_2 \;\to\; y_1 = y_2\big) \quad \wedge \quad \big(C(\bar{x}_1, \bar{x}_2) \;\to\; y_1 \neq y_2\big)$$

where $|\bar{x}_1| = |\bar{x}_2| = n$. Intuitively, it states that y_1 and y_2 must represent the same function and that two adjacent vertices have different colors. It is routine to verify that Ψ is satisfiable if and only if the graph represented by the circuit C is 2-colorable.

To establish the PSPACE-membership, we will use the same idea that 2-SAT is in NL. Note that 2-CNF formula can be rewritten as a conjunction of implications:

$$(\ell_{1,1} \to \ell_{1,2}) \;\wedge\; (\ell_{2,1} \to \ell_{2,2}) \;\wedge\; \cdots \;\wedge\; (\ell_{n,1} \to \ell_{n,2})$$

where each $\ell_{i,j}$ is a literal. In turn, it can be viewed as a directed graph where the literals are the vertices and the implications are the edges. It is not satisfiable if and only if there is a cycle in the graph that contains a literal ℓ and its negation. To check the existence of such a cycle, it suffices to use $O(t)$ space, where t is the number of bits required to remember a literal.

We will use a similar idea to establish the PSPACE-membership of sat(2-DQBF). The detail is as follows. Let Ψ be the input 2-DQBF.

$$\Psi \;:=\; \forall \bar{x}\; \exists y_1(\bar{z}_1) \exists y_2(\bar{z}_2) \quad \psi \qquad \text{where } \bar{x} = (x_1, \ldots, x_n).$$

The algorithm works as follows.

- Guess $\bar{a} \in \Sigma^{n+2}$ where $\psi[(\bar{x}, y_1, y_2) \mapsto \bar{a}] = 0$ and a variable $f_i(\mathsf{prj}_{\bar{x},y_1,y_2}(\bar{z}_i, \bar{a}))$ in the clause $C_{\bar{a}}$.
- Guess a series of implications from $f_i(\mathsf{prj}_{\bar{x},y_1,y_2}(\bar{z}_i, \bar{a}))$ to its negation in $\exp(\Psi)$ and vice versa.

To guess a series of implications from $f_i(\mathsf{prj}_{\bar{x},y_1,y_2}(\bar{z}_i, \bar{a}))$ to its negation, it suffices to remember only one literal at a time which requires only $O(n)$ bits. This establishes the PSPACE-membership of sat(2-DQBF). ◀

3.2 On sat(k-DQBF) where $k \geqslant 3$

In this subsection we will consider sat(k-DQBF) where $k \geqslant 3$. We will first establish that sat(3-DQBF) is NEXP-complete. Then, we will show how transform arbitrary DQBF to an equisatisfiable 3-DQBF.

▶ **Theorem 5.** *sat(3-DQBF) is NEXP-complete.*

Proof. The membership is straightforward. The proof for hardness is by reduction from *succinct 3-colorability* which is known to be NEXP-complete [25]. The idea is quite similar to the one in [8] which reduces it to sat(4-DQBF). By a more careful book-keeping, we show that 3 existential variables is enough to encode succinct 3-colorability.

Let $C(\bar{u}, \bar{v})$ be a circuit where $|\bar{u}| = |\bar{v}| = n$ and let $G_C = (V_C, E_C)$ be the graph represented by C. The main idea is simple. We assume that $\{01, 10, 11\}$ is the set of colors and represent a 3-coloring of the graph G_C with a function $f : \{0,1\}^n \times \{0,1\}^2 \to \{0,1\}$ where for every color $\bar{c} \in \{01, 10, 11\}$, for every vertex $\bar{a} \in \{0,1\}^n$, $f(\bar{a}, \bar{c}) = 1$ if and only if vertex $\bar{a}$ is colored with color $\bar{c}$. We will show that we can construct a 3-DQBF which states that "the coloring must be proper."

The details are as follows. Let $C(\bar{u}, \bar{v})$ be the input circuit (to succinct 3-colorability) where $|\bar{u}| = |\bar{v}| = n$. Consider the following 3-DQBF Ψ.

$$\begin{aligned}\Psi \;:=\; &\forall \bar{x}_1 \forall u_1 \forall v_1\; \forall \bar{x}_2 \forall u_2 \forall v_2\; \forall \bar{x}_3 \forall u_3 \forall v_3 \\ &\exists y_1(\bar{x}_1, u_1, v_1)\; \exists y_2(\bar{x}_2, u_2, v_2)\; \exists y_3(\bar{x}_3, u_3, v_3) \qquad \psi\end{aligned}$$

where $|\bar{x}_1| = |\bar{x}_2| = |\bar{x}_3| = n$ and the formula ψ states the following.

- All y_1, y_2, y_3 are the same function. Formally:

$$\bigwedge_{1 \leqslant i,j \leqslant 3} \Big(\bar{x}_i = \bar{x}_j \ \wedge \ (u_i, v_i) = (u_j, v_j) \Big) \ \rightarrow \ y_i = y_j$$

- No vertex is assigned with the color 00. Formally:

$$(u_1, v_1) = 00 \qquad \rightarrow \qquad y_1 = 0$$

- Every vertex is assigned with exactly one color from $\{01, 10, 11\}$. Formally:

$$\left(\bar{x}_1 = \bar{x}_2 = \bar{x}_3 \ \wedge \ \left(\begin{array}{l} (u_1, v_1), (u_2, v_2), (u_3, v_3) \neq 00 \\ \text{and are pairwise different} \end{array} \right) \right) \ \rightarrow \ \left(\begin{array}{l} \text{exactly} \\ \text{one of} \\ y_1, y_2, y_3 \\ \text{has value 1} \end{array} \right)$$

- Adjacent vertices have different colors. Formally:

$$\Big(C(\bar{x}_1, \bar{x}_2) = 1 \ \wedge \ (u_1, v_1) = (u_2, v_2) \Big) \ \rightarrow \ \Big(y_1 = 0 \ \vee \ y_2 = 0 \Big)$$

It is routine to verify that G_C is 3-colorable if and only if Ψ is satisfiable. Moreover, Ψ can be constructed in polynomial time. ◀

Next, we present a parsimonious polynomial time transformation from arbitrary DQBF to 3-DQBF. It is the DQBF analogue of the well known transformation from SAT to 3-SAT.

▶ **Theorem 6.** *There is a parsimonious polynomial time (Karp) reduction from* $\mathsf{sat}(DQBF)$ *to* $\mathsf{sat}(3\text{-}DQBF)$*. In other words, there is a polynomial time algorithm such that: On input DQBF* Ψ*, it outputs a 3-DQBF* Φ *such that* Ψ *and* Φ *have the same number of satisfying Skolem functions.*

Proof. Before we proceed to give the details, we will first explain the intuition. Consider the following k-DQBF Ψ.

$$\Psi := \forall \bar{x} \ \exists y_1(\bar{z}_1) \cdots \ \exists y_k(\bar{z}_k) \quad \psi \qquad\qquad \text{where } \bar{x} = (x_1, \ldots, x_n)$$

Let $n_i = |\bar{z}_i|$, for each $1 \leqslant i \leqslant k$. We will encode the satisfying Skolem functions $(f_1, \ldots, f_k)$ for Ψ as one function $g : \Sigma^n \times \Sigma^k \rightarrow \Sigma$ as follows. For every $\bar{a} \in \Sigma^n$, for every $\bar{b} = (b_1, \ldots, b_k) \in \Sigma^k$,

$$g(\bar{a}, \bar{b}) \ = \ 1 \quad \text{if and only if} \quad b_i = f_i(\mathsf{prj}_{\bar{x}}(\bar{z}_i, \bar{a})) \qquad\qquad \text{for every } 1 \leqslant i \leqslant k$$

We call such function g *the encoding of* $(f_1, \ldots, f_k)$. Note that for a function $g : \Sigma^n \times \Sigma^k \rightarrow \Sigma$ to properly encode k functions $(f_1, \ldots, f_k)$, it has to satisfy the following "functional property":

For every $\bar{a} \in \Sigma^n$, there is exactly one $\bar{b} \in \Sigma^k$ such that $g(\bar{a}, \bar{b}) = 1$.

Unfortunately DQBF by itself is not strong enough to state such property. For this, we need another type of encoding that can be expressed with 3-DQBF.

We first introduce a few terminology and notations. For $\bar{b}, \bar{c} \in \Sigma^k$, we denote by $\bar{b} \leqslant_{\text{lex}} \bar{c}$, if $\bar{b}$ is "lexicographically" less than or equal to $\bar{c}$.[1] Note that one can easily write a (boolean) formula $\varphi(\bar{x}_1, \bar{x}_2)$, where $|\bar{x}_1| = |\bar{x}_2| = k$ such that $\varphi[(\bar{x}_1, \bar{x}_2) \mapsto (\bar{b}, \bar{c})] = 1$ if and only if $\bar{b} \leqslant_{\text{lex}} \bar{c}$. We denote by $\bar{b} - 1$ and $\bar{b} + 1$ the induced predecessor and successor of $\bar{b}$ in the lexicographic ordering of Σ^k (when $\bar{b} \neq 0^k$ and $\bar{b} \neq 1^k$, respectively).

The monotonic encoding of the functions $(f_1, \ldots, f_k)$ is a function $h : \Sigma^n \times \Sigma^k \rightarrow \Sigma$ such that for every $\bar{a} \in \Sigma^n$, the following holds.

[1] This is the standard lexicographic ordering on Σ^k where 0 is "less than" 1.

$\bar{x} = (x_1, \ldots, x_n)$	$\bar{v} = (v_1, \ldots, v_k)$	$h(\bar{x}, \bar{v})$	
$\bar{a}$	0^k	0	all zeroes
$\vdots$	$\vdots$	$\vdots$	
$\bar{a}$	$\bar{b} - 1$	0	
$\bar{a}$	$\bar{b}$	1	all ones
$\bar{a}$	$\bar{b} + 1$	1	
$\vdots$	$\vdots$	$\vdots$	
$\bar{a}$	1^k	1	

Figure 1 Here $\bar{b} \in \Sigma^k$. The notations $\bar{b} - 1$ and $\bar{b} + 1$ denote the predecessor and the successor of $\bar{b}$, respectively, according to the lexicographic ordering of Σ^k. The function $h : \Sigma^n \times \Sigma^k \to \Sigma$ is monotone (w.r.t. $\bar{v}$), i.e., if $h(\bar{a}, \bar{b}) = 1$, then $h(\bar{a}, \bar{c}) = 1$ for every $\bar{c}$ greater than $\bar{b}$ lexicographically. Such h encodes a function $g : \Sigma^n \times \Sigma^k \to \Sigma$ where $g(\bar{a}, \bar{b}) = 1$ if and only if $h(\bar{a}, \bar{b} - 1) = 0$ and $h(\bar{a}, \bar{b}) = 1$.

- For every $\bar{b}, \bar{c} \in \Sigma^k$, $h(\bar{a}, \bar{b}) \leqslant h(\bar{a}, \bar{c})$ whenever $\bar{b} \leqslant_{\text{lex}} \bar{c}$. That is, it is monotonic w.r.t. to the last k bits.
- For every $\bar{b} = (b_1, \ldots, b_k) \in \Sigma^k$ such that $b_i = f_i(\mathsf{prj}_{\bar{x}}(\bar{z}_i, \bar{a}))$, for every $1 \leqslant i \leqslant k$, the following holds.
  - $h(\bar{a}, \bar{c}) = 0$, for every $\bar{c} \in \Sigma^k$ where $\bar{c} <_{\text{lex}} \bar{b}$.
  - $h(\bar{a}, \bar{c}) = 1$, for every $\bar{c} \in \Sigma^k$ where $\bar{b} \leqslant_{\text{lex}} \bar{c}$.

Intuitively, if h is the monotonic encoding of $(f_1, \ldots, f_k)$, the value $\bar{b} = (b_1, \ldots, b_k)$ where $b_i = f_i(\mathsf{prj}_{\bar{x}}(\bar{z}_i, \bar{a}))$ for every $1 \leqslant i \leqslant k$ can be identified as the lexicographically smallest $\bar{b}$ such that $g(\bar{a}, \bar{b}) = 1$. See Figure 1 for an illustration.

Note that if $h : \Sigma^n \times \Sigma^k \to \Sigma$ is the monotonic encoding of $(f_1, \ldots, f_k)$, we can recover the encoding of $(f_1, \ldots, f_k)$. Indeed, define the function $g : \Sigma^n \times \Sigma^k \to \Sigma$ as follows.

- If $h(\bar{a}, 0^n) = 0$, then $g(\bar{a}, \bar{b}) = 1$ if and only if $h(\bar{a}, \bar{b} - 1) = 0$ and $h(\bar{a}, \bar{b}) = 1$.
  In this case, note that there is exactly one $\bar{b}$ such that $h(\bar{a}, \bar{b} - 1) = 0$ and $h(\bar{a}, \bar{b}) = 1$. Thus, there is exactly one $\bar{b}$ such that $g(\bar{a}, \bar{b}) = 1$.
- If $h(\bar{a}, 0^n) = 1$, then $g(\bar{a}, 0^n) = 1$ and for every $\bar{b} \neq 0^n$, $g(\bar{a}, \bar{b}) = 0$.

Since h is the monotonic encoding of $(f_1, \ldots, f_k)$, it is immediate that g is the encoding of $(f_1, \ldots, f_k)$.

We now give the details of the reduction from $\mathsf{sat}(\text{DQBF})$ to $\mathsf{sat}(3\text{-DQBF})$. On input $\Psi := \forall \bar{x} \exists y_1(\bar{z}_1) \cdots \exists y_k(\bar{z}_k)\ \psi$, where $\bar{x} = (x_1, \ldots, x_n)$, it outputs the following DQBF:

$$\Phi \ := \ \forall \bar{x}_1 \forall \bar{v}_1\ \forall \bar{x}_2 \forall \bar{v}_2\ \forall \bar{x}_3 \forall \bar{v}_3\ \exists p_1(\bar{x}_1, \bar{v}_1) \exists p_2(\bar{x}_2, \bar{v}_2) \exists p_3(\bar{x}_3, \bar{v}_3) \quad \phi$$

where $|\bar{x}_i| = n$ and $|\bar{v}_i| = k$ for each $1 \leqslant i \leqslant 3$ and ϕ states the following.

- p_1 and p_2 represent the monotonic encoding of the Skolem functions $(f_1, \ldots, f_k)$ for Ψ (if exist).
- p_3 represents the encoding of the Skolem functions $(f_1, \ldots, f_k)$ for Ψ (if exists).

The details of ϕ are as follows. Let $\bar{x}_i = (x_{i,1}, \ldots, x_{i,n})$ and $\bar{v}_i = (v_{i,1}, \ldots, v_{i,k})$ for every $1 \leqslant i \leqslant 3$.

(1) p_1 and p_2 represent the same function. Formally:

$$\big(\bar{x}_1 = \bar{x}_2 \quad \wedge \quad \bar{v}_1 = \bar{v}_2\big) \quad \rightarrow \quad \big(p_1 = p_2\big)$$

(2) The function represented by p_1 (and p_2) is a monotonic w.r.t. the bits in $\bar{v}_1$. Formally:

$$\big(\bar{x}_1 = \bar{x}_2 \quad \wedge \quad \bar{v}_1 \leqslant_{\text{lex}} \bar{v}_2\big) \quad \rightarrow \quad \big(p_1 \leqslant p_2\big)$$

(3) That $\bar{v}_1 = 1^k$ implies $p_1 = 1$. Formally:

$$\bar{v}_1 = 1^k \quad \rightarrow \quad p_1 = 1$$

Note that this condition implies that the fact that if p_1 represents a function $h : \Sigma^n \times \Sigma^k \rightarrow \Sigma$, then for every $\bar{a} \in \Sigma^n$, there is $\bar{b} \in \Sigma^k$ such that the value $h(\bar{a}, \bar{b}) = 1$. This is because p_1 represents a monotonic function w.r.t. $\bar{v}_1$.

(4) The function represented by p_3 is the encoding of the k functions whose monotonic encoding is represented by p_1 (and p_2). Formally:

$$\Big(\big(\bar{x}_1 = \bar{x}_2 = \bar{x}_3\big) \ \wedge \ \big(\bar{v}_1 + 1 = \bar{v}_2 = \bar{v}_3\big)\Big) \quad \rightarrow \quad \Big(\big(p_1 = 0 \ \wedge \ p_2 = 1\big) \leftrightarrow p_3 = 1\Big)$$
$$\wedge \ \Big(\big(\bar{x}_2 = \bar{x}_3\big) \ \wedge \ \big(\bar{v}_2 = \bar{v}_3 = 0^k\big)\Big) \quad \rightarrow \quad \Big(p_2 = 1 \leftrightarrow p_3 = 1\Big)$$

(5) The functions $(f_1, \ldots, f_k)$ encoded by p_1, p_2, p_3 respect the dependency set $\bar{z}_i$ for every $1 \leqslant i \leqslant k$. Formally:

$$\Big(\big(\bar{x}_1 = \bar{x}_2 \ \wedge \ \bar{v}_1 + 1 = \bar{v}_2 \ \wedge \ p_1 = 0 \ \wedge \ p_2 = 1\ \big) \ \wedge \ p_3 = 1\Big)$$
$$\rightarrow \quad \bigwedge_{1 \leqslant i \leqslant k} \mathsf{prj}_{\bar{x}}(\bar{z}_i, \bar{x}_2) = \mathsf{prj}_{\bar{x}}(\bar{z}_i, \bar{x}_3) \ \rightarrow \ v_{2,i} = v_{3,i}$$
$$\wedge \ \Big(\big(\bar{v}_2 = 0^k \ \wedge \ p_2 = 1\big) \ \wedge \ p_3 = 1\Big)$$
$$\rightarrow \quad \bigwedge_{1 \leqslant i \leqslant k} \mathsf{prj}_{\bar{x}}(\bar{z}_i, \bar{x}_2) = \mathsf{prj}_{\bar{x}}(\bar{z}_i, \bar{x}_3) \ \rightarrow \ v_{2,i} = v_{3,i}$$

(6) The functions $(f_1, \ldots, f_k)$ encoded by p_3 is indeed a satisfying Skolem functions for Ψ. Formally:

$$p_3 = 1 \quad \rightarrow \quad \psi[(\bar{x}, y_1, \ldots, y_k)/(\bar{x}_3, \bar{v}_3)]$$

It is routine to verify that Ψ and Φ are equisatisfiable. Note also that every Skolem functions $(f_1, \ldots, f_k)$ for Ψ is uniquely represented by their encoding and monotonic encoding. Conversely, every encoding and monotonic encoding represented by p_3 and p_1, p_2 uniquely represented the Skolem functions $(f_1, \ldots, f_k)$ for Ψ. Thus, Ψ and Φ have the same number of satisfying Skolem functions. By inspection, the 3-DQBF Φ can be constructed in polynomial time. ◀

4 Lifting the projections in the class NP to the class NEXP

In this section we will establish the relations between the so called *projections* (from SAT to NP-complete graph problem Π) and polynomial time reductions (from sat(DQBF) to the succinctly represented Π). We first recall the definition of projections [33, 25]. Let $\xi : \Sigma^* \rightarrow \Sigma^*$ be a reduction from a language L to another language K. We say that ξ is a *projection*, if the following holds.

- There is a polynomial $p(n)$ such that for every $w \in \Sigma^*$, the length of $\xi(w)$ is $p(|w|)$.
- There is a polynomial time (deterministic) algorithm $\mathcal{A}$ such that on input 1^n and $1 \leqslant i \leqslant p(n)$, where i is in the binary representation, the output $\mathcal{A}(1^n, i)$ is one of the following:
  - a value (either 0 or 1);
  - a variable x_j (appropriately encoded) where $1 \leqslant j \leqslant n$;
  - the negation of a variable $\neg x_j$ (appropriately encoded) where $1 \leqslant j \leqslant n$;

  such that if $z_1 \cdots z_{p(n)}$ are the output $\mathcal{A}(1^n, 1), \ldots, \mathcal{A}(1^n, p(n))$, the following holds. For every $w = b_1 \cdots b_n \in \Sigma^n$:

$$\xi(w) \;=\; z_1 \cdots z_{p(n)}[(x_1, \ldots, x_n) \mapsto (b_1, \ldots, b_n)]$$

  where $z_1 \cdots z_{p(n)}[(x_1, \ldots, x_n) \mapsto (b_1, \ldots, b_n)]$ is the 0-1 string obtained by substituting each x_i with b_i.

The intuitive meaning of the algorithm $\mathcal{A}(1^n, i)$ is as follows. The i^{th} bit of the output of the reduction ξ on an input of length n is either 0 or 1 or the j^{th} bit of the input (when $\mathcal{A}(1^n, i) = x_j$) or the complement of the j^{th} bit of the input (when $\mathcal{A}(1^n, i) = \neg x_j$). Note also that if there is a projection ξ from L to K, then there is a polynomial time reduction from L to K, where on input w, we compute each bit in $\xi(w)$ by computing $\mathcal{A}(1^{|w|}, i)$ for every $1 \leqslant i \leqslant p(|w|)$. Almost all know reductions from SAT to graph problems are, in fact, projections. We recall the following result from [25] and briefly review the proof.

▶ **Theorem 7** ([25]). *If there is a projection from SAT to a graph problem* Π*, then the succinct version of* Π *is NEXP-hard.*

Proof. We assume that a graph $G = (V, E)$ is encoded as 0-1 string of length $|V|^2$ representing the adjacency matrix of G. Let $L \in \text{NEXP}$. Let $\mathcal{M}$ be the NTM that accepts L in time $2^{p(n)}$ for some polynomial $p(n)$. For $w \in \Sigma^*$, let F_w denote the CNF formula obtained by applying the standard Cook-Levin reduction on w (w.r.t. the NTM $\mathcal{M}$). Assuming that F_w is encoded as 0-1 string, the length of $F(w)$ is $2^{q(n)}$ for some polynomial $q(n)$.

We can design a polynomial time deterministic algorithm $\mathcal{A}$ that on input w and two indexes i, j, determine if a literal ℓ_i appears in clause C_j in the formula F_w. Note that i and j can be encoded in binary representation with $p(n)$ bits. We can easily modify $\mathcal{A}$ into another algorithm $\mathcal{A}'$ such that on input w and index $1 \leqslant i \leqslant 2^{q(n)}$, output the bit-$i$ in the formula F_w.

Let ξ be the projection from SAT to a graph problem Π. Suppose for a formula F, the graph $\xi(F)$ has $r(|F|)$ vertices, for some polynomial $r(n)$. Using ξ, we can design a polynomial time algorithm $\mathcal{B}$ that on input w and index $1 \leqslant i, j \leqslant r(2^{q(n)})$ (in binary), output an index i' (in binary) such that the bit-i' in F_w is the same as the (i, j)-entry in the adjacency matrix of $\xi(F_w)$. We can then combine both algorithms $\mathcal{A}'$ and $\mathcal{B}$ to obtain another algorithm $\mathcal{C}$ such that on input w and indexes $1 \leqslant i, j \leqslant r(2^{q(n)})$, it outputs the (i, j)-entry in the adjacency matrix of $\xi(F_w)$. Note that when the length of the input is fixed, we can construct in polynomial time the boolean circuit representing the algorithm $\mathcal{C}$.

Now, the reduction from L to succinct Π works as follows. On input w, it constructs the boolean circuit for $\mathcal{C}$ where the input length is fixed to $|w| + 2 \cdot \log r(2^{q(n)})$ and the first $|w|$ input gates are fed with w. Note that the output circuit represents the graph $\xi(F_w)$. Thus, we obtain the reduction from L to succinct Π. ◀

We show that it can actually be stated as follows.

▶ **Corollary 8.** *If there is a projection from SAT to a graph problem* Π*, then there is a polynomial time (Karp) reduction from* $\mathsf{sat}(DQBF)$ *to the problem* Π *in succinct representation.*

Proof. We actually just follow the proof in [25] with a slight modification on the definition of $\exp(\Psi)$. Let Ψ be a DQBF with matrix ψ. We modify the definition of the clause $C_{\bar{a},\bar{b}}$ as follows.

- If $\psi[\bar{x},\bar{v}\mapsto\bar{a},\bar{b}]=0$, we set $C_{\bar{a},\bar{b}}$ as in Section 2.
- If $\psi[\bar{x},\bar{v}\mapsto\bar{a},\bar{b}]=1$, we set $C_{\bar{a},\bar{b}}$ as a trivial clause such as $(\neg x\vee x\vee\cdots\vee x)$ for some arbitrary variable x.

Under such definition, we can easily design an algorithm $\mathcal{A}'$ that on input DQBF Ψ and index i, output the i^{th} bit in the formula $\exp(\Psi)$. The reduction from sat(DQBF) to succinct Π can be obtained in exactly the same way as in Theorem 7. ◀

5 The relations between NTIME$[t(n)]$, NSPACE$[s(n)]$ and DQBF

In this section we will show how to reduce the languages in the class $\mathsf{NTIME}[t(n)]$ and $\mathsf{NSPACE}[s(n)]$ to DQBF. We implicitly assume that the functions $t(n)$ and $s(n)$ are time/space constructible. We start with the following theorem which has been proved in [8].[2]

▶ **Theorem 9** ([8, Theorem 1])**.** *For every* $L\in NTIME[t(n)]$*, there is a (deterministic) algorithm* $\mathcal{A}$ *that runs in time polynomial in* n *and* $\log t(n)$ *such that on input word* w*, it outputs a DQBF* Ψ *such that* $w\in L$ *if and only if* Ψ *is satisfiable. Moreover, the output DQBF* Ψ *uses* $O(\log t(n))$ *universal variable and* $O(1)$ *existential variables, where* n *is the length of the input* w*.*

The constant hidden in $O(1)$ in Theorem 9 depends on the number of states, tapes and tape symbols of the Turing machine M that decides L. Combining Theorem 6 and 9, we obtain the following corollary.

▶ **Corollary 10** ([8, Theorem 1])**.** *For every* $T(n)\geqslant n$*, for every* $L\in NTIME[t(n)]$*, there is a (deterministic) algorithm* $\mathcal{A}$ *that runs in time polynomial in* n *and* $\log t(n)$ *such that on input word* w*, it outputs a 3-DQBF* Ψ *such that* $w\in L$ *if and only if* Ψ *is satisfiable. Moreover, the output DQBF* Ψ *uses* $O(\log t(n))$ *universal variable where* n *is the length of the input* w*.*

Note that since the reductions in Theorem 9 and 6 are parsimonious, the algorithm $\mathcal{A}$ in Corollary 10 is also parsimonious in the sense that if $\mathcal{M}$ is the NTM that accepts L, then the number of accepting runs of $\mathcal{M}$ on input word w is the precisely the number of satisfying Skolem functions for the output DQBF Ψ.

▶ **Theorem 11.** *For every language* $L\in NSPACE[s(n)]$*, there is a deterministic algorithm* $\mathcal{A}$ *with run time polynomial in* n *and* $s(n)$ *such that: On input* w*, it outputs a 2-DQBF* Ψ *with* $O(s(|w|))$ *universal variables such that* $w\in L$ *if and only if* Ψ *is not satisfiable.*

Proof. Let $L\in\mathsf{NSPACE}[s(n)]$ and let $\mathcal{M}$ be the NTM that accepts L using $s(n)$ space. We may assume that $\mathcal{M}$ halts on every input word. We first present the reduction to 2-CNF formula (with exponential blow-up). On input word w, it construct the following formula, denoted by F_w.

(a) The variables are X_C, where the index C ranges over all the configurations of M on w.

[2] Actually [8, Theorem 1] establishes Theorem 9 for some exponential $t(n)$, i.e., $t(n)=2^{p(n)}$ for some polynomial $p(n)$. However, it can be easily verified that the proof can be used for Theorem 9.

(b) For every two configurations C_1 and C_2 where C_2 is the next configuration of C_1, we have an implication $X_{C_1} \to X_{C_2}$.

(c) For the initial configuration C_0, we have the implication $\neg X_{C_0} \to X_{C_0}$.

(d) For the initial configuration C_0 and the accepting configuration C_{acc}, we have the implication $X_{C_{acc}} \to \neg X_{C_0}$.

It is not difficult to show that M accepts w if and only if F_w is not satisfiable.

We can easily modify the construction above to the case of DQBF by encoding the configuration of M on w with strings from Σ with length $O(s(n))$, where $n = |w|$. That is, on input word w, we can construct in time polynomial in $s(n)$ a DQBF Ψ such that $\exp(\Psi)$ is equivalent to the formula F_w. The details are as follows. On input word w, construct the following DQBF:

$$\Psi \;:=\; \forall \bar{x}_1 \forall \bar{x}_2 \; \exists y_1(\bar{x}_1) \exists y_2(\bar{x}_2) \quad \psi$$

where $|\bar{x}_1| = |\bar{x}_2| = O(s(n))$, i.e., $\bar{x}_1, \bar{x}_2$ are used to represent the 0-1 encoding of the configuration of $\mathcal{M}$ of w and the matrix ψ states the following.

(1) The functions y_1 and y_2 are the same. Formally:

$$\bar{x}_1 = \bar{x}_2 \;\to\; y_1 = y_2$$

(2) If $\bar{x}_1$ and $\bar{x}_2$ encode configurations and $\bar{x}_2$ is the "next" configuration of $\bar{x}_1$, then y_1 implies y_2. Formally:

$$C(\bar{x}_1, \bar{x}_2) \;\to\; \big(\neg y_1 \;\vee\; y_2\big)$$

where $C(\bar{x}_1, \bar{x}_2)$ is a formula that checks whether $\bar{x}_1$ and $\bar{x}_2$ are configurations and $\bar{x}_2$ is the next configuration of $\bar{x}_1$. Such formula can be easily constructed in time polynomial in $s(n)$.

(3) If $\bar{x}_1$ is the accepting configuration and $\bar{x}_2$ is the initial configuration, then y_1 implies $\neg y_2$. Formally:

$$C'(\bar{x}_1, \bar{x}_2) \;\to\; \big(\neg y_1 \;\vee\; \neg y_2\big)$$

where $C'(\bar{x}_1, \bar{x}_2)$ is a formula that checks whether $\bar{x}_1$ is the accepting configuration and $\bar{x}_2$ is the initial configuration. Such formula can be easily constructed in time polynomial in $\max(n, s(n))$.

(4) If $\bar{x}_1, \bar{x}_2$ are the initial configuration, then $y_1 \;\vee\; y_2$. Formally:

$$C''(\bar{x}_1, \bar{x}_2) \;\to\; \big(y_1 \;\vee\; y_2\big)$$

where $C''(\bar{x}_1, \bar{x}_2)$ is a formula that checks whether $\bar{x}_1, \bar{x}_2$ are the initial configuration. Again, such formula can be easily constructed in time polynomial in $\max(n, s(n))$.

It is not difficult to see that on every input word w, the formula F_w is equivalent to $\exp(\Psi)$. This completes the proof of Theorem 11. ◀

It is not difficult to see that if the TM $\mathcal{M}$ in the proof of Theorem 11 is deterministic, we can easily modify the construction such that $w \in L$ if and only if the output Ψ is satisfiable.

6 Concluding remarks

In this paper we have shown that a number of well known classical results on SAT can be lifted up to DQBF. Previously it has been observed that DQBF formulas indeed represent (exponentially large) CNF formulas and that the satisfiability problem becomes NEXP-complete. In this paper we take one little step forward by presenting a few results that shows a strong resemblance between DQBF and SAT. Together with the work in [8], we hope that it can be a convincing evidence for DQBF to be the *bona fide* problem in NEXP.

We believe there is still a lot of work to do. There are still a plethora of results on SAT that we haven't considered and it would be interesting to investigate which results can be lifted to DQBF and which can't. We leave it for future work.

References

1 SAT competition, 2020. URL: `http://www.satcompetition.org/`.
2 V. Balabanov, H.-J. K. Chiang, and J.-H. R. Jiang. Henkin quantifiers and boolean formulae: A certification perspective of DQBF. *Theor. Comput. Sci.*, 523:86–100, 2014.
3 V. Balabanov and J.-H. R. Jiang. Reducing satisfiability and reachability to DQBF. In *Talk given at QBF*, 2015.
4 A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*. IOS Press, 2009.
5 R. Bloem, R. Könighofer, and M. Seidl. SAT-based synthesis methods for safety specs. In *VMCAI*, 2014.
6 U. Bubeck. *Model-based transformations for quantified boolean formulas*. PhD thesis, University of Paderborn, 2010.
7 K. Chatterjee, T. Henzinger, J. Otop, and A. Pavlogiannis. Distributed synthesis for LTL fragments. In *FMCAD*, 2013.
8 F.-H. Chen, S.-C. Huang, Y.-C. Lu, and T. Tan. Reducing NEXP-complete problems to DQBF. In *FMCAD*, 2022.
9 D. Chistikov, C. Haase, Z. Hadizadeh, and A. Mansutti. Higher-order quantified boolean satisfiability. In *MFCS*, 2022.
10 S. Cook. The complexity of theorem proving procedures. In *STOC*, 1971.
11 S. Cook. Short propositional formulas represent nondeterministic computations. *Inf. Process. Lett.*, 26(5):269–270, 1988.
12 A. Fröhlich, G. Kovásznai, and A. Biere. A DPLL algorithm for solving DQBF. In *POS-12, Third Pragmatics of SAT workshop*, 2012.
13 A. Fröhlich, G. Kovásznai, A. Biere, and H. Veith. iDQ: Instantiation-based DQBF solving. In *POS-14, Fifth Pragmatics of SAT workshop*, 2014.
14 H. Galperin and A. Wigderson. Succinct representations of graphs. *Inf. Control.*, 56(3):183–198, 1983.
15 A. Ge-Ernst, C. Scholl, and R. Wimmer. Localizing quantifiers for DQBF. In *FMCAD*, 2019.
16 Aile Ge-Ernst, Christoph Scholl, Juraj Síc, and Ralf Wimmer. Solving dependency quantified boolean formulas using quantifier localization. *Theor. Comput. Sci.*, 925:1–24, 2022.
17 K. Gitina, S. Reimer, M. Sauer, R. Wimmer, C. Scholl, and B. Becker. Equivalence checking of partial designs using dependency quantified boolean formulae. In *ICCD*, 2013.
18 K. Gitina, R. Wimmer, S. Reimer, M. Sauer, C. Scholl, and B. Becker. Solving DQBF through quantifier elimination. In *DATE*, 2015.
19 J.-H. R. Jiang. Quantifier elimination via functional composition. In *CAV*, 2009.
20 R. Jiang. Second-order quantified boolean logic. In *AAAI*, 2023.
21 G. Kovásznai. What is the state-of-the-art in DQBF solving. In *Join Conference on Mathematics and Computer Science*, 2016.

22 A. Kuehlmann, V. Paruthi, F. Krohm, and M. Ganai. Robust boolean reasoning for equivalence checking and functional property verification. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 21(12):1377–1394, 2002.

23 O. Kullmann and A. Shukla. Autarkies for DQCNF. In *FMCAD*, 2019.

24 L. Levin. Universal search problems (in russian. *Problems of Information Transmission*, 9(3):115–116, 1973.

25 C. Papadimitriou and M. Yannakakis. A note on succinct representations of graphs. *Inf. Control.*, 71(3):181–185, 1986.

26 G. Peterson and J. Reif. Multiple-person alternation. In *FOCS*, 1979.

27 F.-X. Reichl and F. Slivovsky. Pedant: A certifying DQBF solver. In *SAT*, 2022.

28 F.-X. Reichl, F. Slivovsky, and S. Szeider. Certified DQBF solving by definition extraction. In *SAT*, 2021.

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

30 C. Scholl, R. J.-H. Jiang, R. Wimmer, and A. Ge-Ernst. A PSPACE subclass of dependency quantified boolean formulas and its effective solving. In *AAAI*, 2019.

31 C. Scholl and R. Wimmer. Dependency quantified boolean formulas: An overview of solution methods and applications - extended abstract. In *SAT*, 2018.

32 J. Síc and J. Strejcek. DQBDD: an efficient bdd-based DQBF solver. In *SAT*, 2021.

33 S. Skyum and L. Valiant. A complexity theory based on boolean algebra. *J. ACM*, 32(2):484–502, 1985.

34 L. Stockmeyer and A. Meyer. Word problems requiring exponential time: Preliminary report. In *STOC*, 1973.

35 L. Tentrup and M. Rabe. Clausal abstraction for DQBF. In *SAT*, 2019.

36 K. Wimmer, R. Wimmer, C. Scholl, and B. Becker. Skolem functions for DQBF. In *ATVA*, 2016.

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

38 R. Wimmer, S. Reimer, P. Marin, and B. Becker. HQSpre – an effective preprocessor for QBF and DQBF. In *TACAS*, 2017.

39 R. Wimmer, C. Scholl, and B. Becker. The (D)QBF preprocessor hqspre - underlying theory and its implementation. *J. Satisf. Boolean Model. Comput.*, 11(1):3–52, 2019.