Fine-Grained Complexity Analysis of Dependency Quantified Boolean Formulas

Che Cheng
Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan

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

Jie-Hong Roland Jiang
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan

Friedrich Slivovsky
Department of Computer Science, University of Liverpool, UK

Tony Tan
Department of Computer Science, University of Liverpool, UK

Abstract

Dependency Quantified Boolean Formulas (DQBF) extend Quantified Boolean Formulas by allowing each existential variable to depend on an explicitly specified subset of the universal variables. The satisfiability problem for DQBF is NEXP-complete in general, with only a few tractable fragments known to date. We investigate the complexity of DQBF with k existential variables (k-DQBF) under structural restrictions on the matrix – specifically, when it is in Conjunctive Normal Form (CNF) or Disjunctive Normal Form (DNF) – as well as under constraints on the dependency sets. For DNF matrices, we obtain a clear classification: 2-DQBF is PSPACE-complete, while 3-DQBF is NEXP-hard, even with disjoint dependencies. For CNF matrices, the picture is more nuanced: we show that the complexity of k-DQBF ranges from NL-complete for 2-DQBF with disjoint dependencies to NEXP-complete for 6-DQBF with arbitrary dependencies.

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

Keywords and phrases Dependency quantified Boolean formulas, complexity, completeness, conjunctive normal form, disjunctive normal form

Digital Object Identifier 10.4230/LIPIcs.SAT.2025.10

Funding We would like to acknowledge the generous support of Royal Society International Exchange Grant no. EC\R3\233183, the National Science and Technology Council of Taiwan under grant NSTC 111-2923-E-002-013-MY3, and the NTU Center of Data Intelligence: Technologies, Applications, and Systems under grant NTU-113L900903.

1 Introduction

Propositional satisfiability (SAT) solving has made significant progress over the past 30 years [4, 10]. Thanks to clever algorithms and highly optimised solvers, SAT has become a powerful tool for solving hard combinatorial problems in many areas, including verification, planning, and artificial intelligence [5]. Modern solvers can handle very large formulas efficiently, making SAT a practical choice in many settings.

However, for problems beyond NP, such as variants of reactive synthesis, direct encodings in propositional logic often grow exponentially with the input and quickly become too large to fit in memory. This has led to growing interest in more expressive logics, such as Quantified Boolean Formulas (QBF) and Dependency Quantified Boolean Formulas (DQBF) [17]. DQBF extends QBF by allowing explicit control over the dependency sets: each existential variable can be assigned its own set of universal variables it depends on. A model of a DQBF assigns to each existential variable a Skolem function that maps assignments of its dependency set to truth values. From a game-theoretic point of view, a DQBF model is a collection of sets of local strategies – one set for each existential variable – that may observe only part of the universal assignment. This makes DQBF more succinct than QBF and particularly well-suited for applications such as synthesis and verification, where components often make decisions based on partial information. Unfortunately, this added expressiveness comes at a cost: DQBF satisfiability is NEXP-complete, and only a few tractable fragments are known [7, 8, 6, 18, 15]. One notable tractable case involves CNF matrices with dependency sets that are either pairwise disjoint or identical; such formulas can be rewritten into satisfiability-equivalent Σ_3-QBFs [18].

Building on these ideas, we apply similar restrictions on the dependency sets to refine a recent classification of the complexity of DQBF with k existential variables (k-DQBF) [14]. For DNF matrices, this restriction has no effect, since the proofs in [14] for the PSPACE-hardness of 2-DQBF and NEXP-hardness of 3-DQBF can be carried over to formulas with pairwise disjoint dependency sets.

For CNF matrices, the situation is more subtle. For $k \geqslant 3$ and even non-constant k with disjoint dependencies, we extend the strategy of [18] to split clauses containing variables with incomparable dependency sets, but instead of reducing it to a QBF, we directly construct an NP algorithm to establish the NP membership. This technique can be extended to the case where any two dependency sets are either disjoint or comparable, and the size blow-up remains polynomial for constant k. The resulting DQBF only has existential variables with empty dependency sets, and its satisfiability can be checked in NP.

When arbitrary dependencies are allowed in CNF matrices, we prove that 3-DQBF is Π_2^{P}-hard. Further, a variant of Tseitin transformation lets us convert a k-DQBF with an arbitrary matrix into a $(k+3)$-DQBF with CNF matrix, yielding PSPACE-hardness of 5-DQBF and NEXP-hardness of 6-DQBF with CNF matrices.

As for the satisfiability problem of 2-DQBF, [13] shows that it reduces to detecting contradicting cycles in a succinctly represented implication graph, making it PSPACE-complete. For CNF matrices and disjoint dependencies, we show that the fully expanded graph has a simple structure, allowing satisfiability tests in NL. Consequently, the satisfiability of 2-DQBF with CNF matrices and unrestricted dependencies is in coNP – one can guess an assignment to the shared universal variables and solve the resulting instance with disjoint dependencies in NL. We also prove the NL- and coNP-hardness of the two problems via a reduction from 2-SAT and 3-DNF tautology, respectively.

Our results, summarised in Table 1, help map out the complexity of natural fragments of DQBF and show how both the formula structure and dependency restrictions play a key role in determining tractability.

2 Preliminaries

In this section, we define the notation used throughout this paper and recall the necessary technical background. All logarithms have base 2. For a positive integer m, $[m]$ denotes the set of integers $\{1, \ldots, m\}$.

Boolean values TRUE and FALSE are denoted by $\top$ and $\bot$, respectively. Boolean connectives $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$, and $\oplus$ are interpreted as usual. A literal ℓ is a Boolean variable v or its negation $\neg v$. We write $\operatorname{var}(v) = \operatorname{var}(\neg v) = v$ for the variable of a literal and $\operatorname{sgn}(v) = \top$ and $\operatorname{sgn}(\neg v) = \bot$ to denote its sign. We also write $v \oplus \bot$ and $v \oplus \top$ to denote the literals v and $\neg v$, respectively.

Table 1 Summary of the complexity results.

k	k-DQBF$^{\mathsf{d}}_{\mathsf{cnf}}$	k-DQBF$^{\mathsf{de}}_{\mathsf{cnf}}$	k-DQBF$^{\mathsf{dec}}_{\mathsf{cnf}}$, k-DQBF$^{\mathsf{ds}}_{\mathsf{cnf}}$	k-DQBF$_{\mathsf{cnf}}$	k-DQBF$^{\mathsf{d}}_{\mathsf{dnf}}$
1	–	–	–	L (Theorem 19)	coNP-c (Theorem 4)
2	NL-c (Theorem 8)	NL-c (Corollary 17)	NL-c (Corollary 17)	coNP-c (Theorem 20)	PSPACE-c (Theorem 4)
$3, 4$	NP-c (Theorem 14)	NP-c (Corollary 17)	NP-c (Corollary 17)	Π_2^{P}-h (Theorem 21)	NEXP-c (Theorem 4)
5				PSPACE-h (Theorem 23)	
6+				NEXP-c (Theorem 23)	
Non-const.		Σ_3^{P}-c [18]	NEXP-c [18]		

Note: "-c" denotes "-complete", "-h" denotes "-hard", and "non-const." denotes "non-constant."

A clause is a disjunction of literals, and a cube is a conjunction of literals. A Boolean formula φ is in conjunctive normal form (CNF) if it is a conjunction of clauses and in disjunctive normal form (DNF) if it is a disjunction of cubes. We view a clause or a cube as a set of literals and a formula in CNF (respectively, DNF) as a set of clauses (respectively, cubes) whenever appropriate. We sometimes write a clause in the form of $Q \to C$, where Q is a cube and C is a clause, and similarly a DNF formula in the form of $\varphi \to \psi$, where φ is in CNF and ψ is in DNF.

We say that two sets of clauses A and B are *variable-disjoint* if for any clause $C_1 \in A$ and $C_2 \in B$, C_1 and C_2 do not share a common variable. For variable-disjoint sets A and B, we write $A \times B$ to denote the set of clauses $\{(C_1 \vee C_2) \mid C_1 \in A, C_2 \in B\}$. We generalise this notion to $A_1 \times A_2 \times \cdots \times A_n$ for pairwise variable-disjoint sets $A_1, \ldots, A_n$.

We write $\bar{v} = (v_1, \ldots, v_n)$ to denote a vector of n Boolean variables with $|\bar{v}| := n$ denoting its length.[1] An *assignment* on $\bar{v}$ is a function from $\bar{v}$ to $\{\top, \bot\}$. We often identify an assignment on $\bar{v}$ with a vector $\bar{a} = (a_1, \ldots, a_n) \in \{\top, \bot\}^n$, denoted $\bar{a}^{\bar{v}}$, which maps each v_i to a_i. When $\bar{v} \subseteq \bar{u}$, we write $\bar{a}^{\bar{u}}(\bar{v})$ to denote the vector of Boolean values $(\bar{a}^{\bar{u}}(v))_{v \in \bar{v}}$. When $\bar{v}$ is clear from the context, we will simply write $\bar{a}$ instead of $\bar{a}^{\bar{v}}$.

Two assignments $\bar{a}^{\bar{u}}$ and $\bar{b}^{\bar{v}}$ are *consistent*, denoted by $\bar{a}^{\bar{u}} \simeq \bar{b}^{\bar{v}}$, if $\bar{a}^{\bar{u}}(v) = \bar{b}^{\bar{v}}(v)$ for every $v \in \bar{u} \cap \bar{v}$. When $\bar{a}^{\bar{u}}$ and $\bar{b}^{\bar{v}}$ are consistent, we write $(\bar{a}^{\bar{u}}, \bar{b}^{\bar{v}})$ to denote the union $\bar{a}^{\bar{u}} \cup \bar{b}^{\bar{v}}$. Given a Boolean formula ϕ over the variables $\bar{u}, \bar{v}$ and an assignment $\bar{a}^{\bar{v}}$, we denote by $\phi[\bar{a}^{\bar{v}}]$ the induced formula over the variables $\bar{u}$ obtained by assigning the variables in $\bar{v}$ with Boolean values according to the assignment $\bar{a}^{\bar{v}}$.

For a positive integer m and a vector of variables $\bar{u}$ of length $n > \log m$, by abuse of notation, we write $\bar{u} = m$ to denote the cube $\bigwedge_{i \in [n]} u_i \leftrightarrow a_i$, where $(a_1, \ldots, a_n)$ is the n-bit binary representation of m.

[1] To avoid clutter, we always assume a vector of variables $\bar{v} = (v_1, \ldots, v_n)$ do not contain duplicate entries, which can be viewed as a set $\{v_1, \ldots, v_n\}$. We will thus use set-theoretic operations on such vectors as on sets.

2.1 DQBF and Its Subclasses

We consider Dependency Quantified Boolean Formulas (DQBF) of the form

$$\Phi = \forall \bar{x}, \exists y_1(D_1), \ldots, y_k(D_k).\, \varphi\,, \tag{1}$$

where $\bar{x} = (x_1, \ldots, x_n)$, $D_i \subseteq \bar{x}$ is the *dependency set* of the existential variable y_i for every $i \in [k]$, and φ is a quantifier-free Boolean formula over the variables $\bar{x} \cup \bar{y}$ called the *matrix* of Φ. We write $\mathrm{dep}(v) \coloneqq D_i$ if $v = y_i$ and $\mathrm{dep}(v) \coloneqq \{x_i\}$ if $v = x_i$. We extend this notation to literals and clauses by letting $\mathrm{dep}(\ell) \coloneqq \mathrm{dep}(\mathrm{var}(\ell))$ for a literal ℓ and $\mathrm{dep}(C) \coloneqq \bigcup_{\ell \in C} \mathrm{dep}(\ell)$ for a clause C.

We say that Φ is satisfiable if for every $i \in [k]$ there is a Boolean formula f_i using only variables in D_i such that by replacing each y_i with f_i, the formula φ becomes a tautology. In this case, we call the sequence $f_1, \ldots, f_k$ a model of Φ and refer to each individual f_i as a Skolem function for y_i.

We define the subclasses k-$\mathrm{DQBF}^{\alpha}_{\beta}$ of DQBF, where $k \geqslant 1$ indicates the number of existential variables, $\alpha \in \{\mathsf{d}, \mathsf{de}, \mathsf{dec}, \mathsf{ds}\}$ indicates the condition on the dependency sets, and $\beta \in \{\mathsf{cnf}, \mathsf{dnf}\}$ indicates the form of the matrix.

For the dependency set annotation α, we define:

DQBF$^{\mathsf{d}}$ For every $i \neq j$, $D_i \cap D_j = \emptyset$,

DQBF$^{\mathsf{de}}$ For every $i \neq j$, $D_i \cap D_j = \emptyset$ or $D_i = D_j$,

DQBF$^{\mathsf{dec}}$ For every $i \neq j$ with $|D_i| \leqslant |D_j|$, $D_i \cap D_j = \emptyset$, $D_i = D_j$, or $D_j = \bar{x}$, and

DQBF$^{\mathsf{ds}}$ For every $i \neq j$ with $|D_i| \leqslant |D_j|$, $D_i \cap D_j = \emptyset$ or $D_i \subseteq D_j$.

The letters d, e, c, and s denote *disjoint*, *equal*, *complete*, and *subset*, respectively. Note that the dependency sets of a $\mathrm{DQBF}^{\mathsf{ds}}$ formula form a *laminar set family.* The classification of different dependency structures is inspired by [18], but we specify the condition that the formula is in CNF explicitly in our notation. That is, $\mathrm{DQBF}^{\mathrm{de}}$ and $\mathrm{DQBF}^{\mathrm{dec}}$ in [18] correspond to $\mathrm{DQBF}^{\mathsf{de}}_{\mathsf{cnf}}$ and $\mathrm{DQBF}^{\mathsf{dec}}_{\mathsf{cnf}}$ in our notation, respectively.

Note that $\mathrm{DQBF}^{\mathsf{d}} \subseteq \mathrm{DQBF}^{\mathsf{de}} \subseteq \mathrm{DQBF}^{\mathsf{dec}} \subseteq \mathrm{DQBF}^{\mathsf{ds}}$. The first two inclusions are trivial, and the last one comes from the observation that both $D_i = D_j$ and $D_j = \bar{x}$ are special cases of $D_i \subseteq D_j$.

When k, α, or β is missing, it means that the corresponding restriction is dropped. For instance, 3-$\mathrm{DQBF}_{\mathsf{dnf}}$ denotes the class of DQBF with 3 existential variables, arbitrary dependency structure, and matrix in DNF, while $\mathrm{DQBF}^{\mathsf{d}}$ denotes the class of DQBF with the dependency structure specified by d and an arbitrary Boolean formula as the matrix. We denote by $\mathsf{sat}(k\text{-}\mathrm{DQBF}^{\alpha}_{\beta})$ the satisfiability problem for the class k-$\mathrm{DQBF}^{\alpha}_{\beta}$.

► Remark 1. For every $\alpha \in \{\mathsf{d}, \mathsf{de}, \mathsf{dec}, \mathsf{ds}\}$ and $\beta \in \{\mathsf{cnf}, \mathsf{dnf}\}$, checking whether a DQBF formula Φ is in the class $\mathrm{DQBF}^{\alpha}_{\beta}$ can be done deterministically in space logarithmic in the length of Φ. To do so, we iterate through all the variables to check whether it satisfies the conditions set by α. In each iteration, it suffices to store $O(1)$ number of indices of the variables, which requires only logarithmic space.

2.2 Manipulation of DQBF$_{\mathsf{cnf}}$

We recall two operations for manipulating $\mathrm{DQBF}_{\mathsf{cnf}}$ formulas, namely *universal reduction* [3, 12] and *resolution-based variable elimination* [20].

► **Lemma 2** (Universal reduction [3, 12]). *Let $\Phi = \forall \bar{x}, \exists y_1(D_1), \ldots, \exists y_k(D_k).\, \varphi$ be a $\mathrm{DQBF}_{\mathsf{cnf}}$ formula, $C \in \varphi$ be a clause, $\ell \in C$ be a universal literal, and let $C' \coloneqq C \setminus \{\ell\}$. If $\ell \notin \mathrm{dep}(C')$, then Φ is equisatisfiable to*

$$\Phi' \coloneqq \forall \bar{x}, \exists y_1(D_1), \ldots, \exists y_k(D_k).\, \varphi \cup \{C'\} \setminus \{C\}\,.$$

Using universal reduction, we assume that $\bigcup_{i\in[k]} D_i = \bar{x}$ for every $\mathrm{DQBF}_{\mathsf{cnf}}$ formula, since any universal variable not in $\bigcup_{i\in[k]} D_i$ can be universally-reduced from every clause.

For variable elimination by resolution, we only need a weaker version, which is sufficient for our purpose.

▶ **Lemma 3** (Variable elimination by resolution [20]). *Let $\Phi = \forall\bar{x}, \exists y_1(D_1), \ldots, \exists y_k(D_k).\, \varphi$ be a $\mathrm{DQBF}_{\mathsf{cnf}}$ formula. We partition φ into three sets:*

- $\varphi^{y_1} := \{C \in \varphi \,|\, y_1 \in C\}$,
- $\varphi^{\neg y_1} := \{C \in \varphi \,|\, \neg y_1 \in C\}$, *and*
- $\varphi^{\emptyset} := \varphi \setminus (\varphi^{y_1} \cup \varphi^{\neg y_1})$.

If for every $C \in \varphi^{y_1}$ we have $\mathrm{dep}(C) \subseteq \mathrm{dep}(y_1)$, or for every $C \in \varphi^{\neg y_1}$ we have $\mathrm{dep}(C) \subseteq \mathrm{dep}(y_1)$, then Φ is equisatisfiable to

$$\forall\bar{x}, \exists y_2(D_2), \ldots, \exists y_k(D_k).\, \varphi^{\emptyset} \cup \{C \otimes_{y_1} C' \,|\, C \in \varphi^{y_1}, C' \in \varphi^{\neg y_1}\},$$

where $C \otimes_v C'$ denotes the resolution of C and C' w.r.t. the pivot v, i.e., $C \otimes_v C' = (C \setminus \{v\}) \cup (C' \setminus \{\neg v\})$.

The intuition is that y_1 can "see" every assignment that may force it to be assigned to $\top$ (respectively, $\bot$), and thus if all resolvents are satisfied, then there must be a Skolem function for y_1 that satisfies the clauses in $\varphi^{y_1} \cup \varphi^{\neg y_1}$. Note that the number of clauses after removing y_1 is at most $|\varphi|^2$.

2.3 Universal Expansion of k-DQBF

Consider a k-DQBF formula $\Phi := \forall\bar{x}, \exists y_1(D_1), \ldots, \exists y_k(D_k).\, \varphi$. Let $\bar{y} = (y_1, \ldots, y_k)$. Given an assignment $\bar{a}$ on $\bar{x}$ and $\bar{b}$ on $\bar{y}$, for every $i \in [k]$, let $\bar{a}_i$ be the restriction of $\bar{a}$ to D_i and b_i be the restriction of $\bar{b}$ to y_i. We can expand Φ into an equisatisfiable k-CNF formula $\exp(\Phi)$ by instantiating each y_i into exponentially many *instantiated variables* of the form $Y_{i,\bar{a}_i}$ [2, 6, 12]. Formally,

$$\exp(\Phi) := \bigwedge_{(\bar{a},\bar{b}):\varphi[\bar{a},\bar{b}]=\bot} \mathcal{C}_{\bar{a},\bar{b}},$$

where $\mathcal{C}_{\bar{a},\bar{b}} := \bigvee_{i\in[k]} Y_{i,\bar{a}_i} \oplus b_i$. Intuitively, in the expansion $\exp(\Phi)$, the Boolean variable $Y_{i,\bar{a}_i}$ represents the value of a candidate Skolem function $f_i(\bar{a}_i)$ for y_i. The universal expansion shows that the satisfiability of Φ can be reduced to a Boolean satisfiability problem (with exponential blow-up). Moreover, if the assignment $(\bar{a}, \bar{b})$ falsifies the matrix φ, then a satisfying assignment of $\exp(\Phi)$ must assign $Y_{i,\bar{a}_i}$ to $\neg b_i$ for some $i \in [k]$.

3 Complexity of $\mathsf{sat}(k\text{-}\mathrm{DQBF}^{\mathsf{d}}_{\mathsf{dnf}})$

Having defined the various subclasses of DQBF, we can refine previous results by stating them more precisely. In this section, we consider the case where the matrix is in DNF.

By combining the DNF version of Tseitin transformation [9, Proposition 1] and the results in [14], we can show that restricting to DNF matrix and pairwise-disjoint dependency sets does not affect the complexity of $\mathsf{sat}(k\text{-DQBF})$.

▶ **Theorem 4.** $\mathsf{sat}(k\text{-}\mathrm{DQBF}^{\mathsf{d}}_{\mathsf{dnf}})$ *is* coNP-, PSPACE-, *and* NEXP-*complete when $k = 1$, $k = 2$, and $k \geqslant 3$, respectively.*

See the appendix for the detailed proof. Since $k\text{-DQBF}^{\mathsf{d}}_{\mathsf{dnf}} \subseteq k\text{-DQBF}^{\mathsf{de}}_{\mathsf{dnf}} \subseteq k\text{-DQBF}^{\mathsf{dec}}_{\mathsf{dnf}} \subseteq k\text{-DQBF}^{\mathsf{ds}}_{\mathsf{dnf}} \subseteq k\text{-DQBF}_{\mathsf{dnf}} \subseteq k\text{-DQBF}$, we have the following corollary:

▶ **Corollary 5.** $\mathsf{sat}(k\text{-DQBF}^{\alpha}_{\mathsf{dnf}})$ *and* $\mathsf{sat}(k\text{-DQBF}_{\mathsf{dnf}})$ *is* coNP-*,* PSPACE-*, and* NEXP-*complete when* $k = 1$*,* $k = 2$*, and* $k \geqslant 3$*, respectively, for every* $\alpha \in \{\mathsf{de}, \mathsf{dec}, \mathsf{ds}\}$.

4 Complexity of $\mathsf{sat}(k\text{-DQBF}^{\alpha}_{\mathsf{cnf}})$

In this section, we consider the complexity of $\mathsf{sat}(k\text{-DQBF}^{\alpha}_{\mathsf{cnf}})$ and $\mathsf{sat}(\text{DQBF}^{\alpha}_{\mathsf{cnf}})$, with a focus on the case where $\alpha = \mathsf{d}$. We first prove an important property of the expansion of $\text{DQBF}^{\mathsf{d}}_{\mathsf{cnf}}$ formulas in Section 4.1 Then, in Sections 4.2 and 4.3 we show that $\mathsf{sat}(k\text{-DQBF}^{\mathsf{d}}_{\mathsf{cnf}})$ is of the same complexity as k-SAT for $k \geqslant 2$,[2] and that $\mathsf{sat}(\text{DQBF}^{\mathsf{d}}_{\mathsf{cnf}})$ is of the same complexity as SAT. This shows that, in stark contrast to the DNF case in the previous section, with pairwise disjoint dependency sets and with CNF matrix, the exponential gap between SAT and DQBF disappears. Finally, we discuss other dependency structures in Section 4.4.

4.1 Universal Expansion of $\text{DQBF}^{\mathsf{d}}_{\mathsf{cnf}}$

In this section, we show a useful property of the expansion of $\text{DQBF}^{\mathsf{d}}_{\mathsf{cnf}}$ formulas. We fix a $k\text{-DQBF}^{\mathsf{d}}_{\mathsf{cnf}}$ formula:

$$\Phi = \forall \bar{x}, \exists y_1(D_1), \ldots, \exists y_k(D_k).\ \bigwedge_{j \in [m]} C_j. \tag{2}$$

Let $\bar{y} = (y_1, \ldots, y_k)$. Given an assignment $\bar{a}$ on $\bar{x}$ and $\bar{b}$ on $\bar{y}$, for every $i \in [k]$, let $\bar{a}_i$ be the restriction of $\bar{a}$ to D_i and b_i be the restriction of $\bar{b}$ to y_i.

Recall that for a DQBF formula Φ, each instantiated clause in $\exp(\Phi)$ corresponds to a falsifying assignment of the matrix of Φ. For a formula in CNF, the set of falsifying assignments can be represented by the union of the set of falsifying assignments of each individual clause. This allows us to represent the instantiated clauses in $\exp(\Phi)$ as the union of polynomially many sets when Φ is a $\text{DQBF}^{\mathsf{d}}_{\mathsf{cnf}}$ formula. Moreover, the disjoint dependency structure allows us to further represent each of these sets as the Cartesian product of variable-disjoint sets of instantiated literals. To formally state the property, we first define some notation.

For a clause C_j in Φ, we write $C^i_j(\Phi)$ to denote the subset of C_j within y_i's dependency set, $\mathcal{L}_{i,j}(\Phi)$ the set of instantiated literals $Y_{i,\bar{a}_i} \oplus b_i$ where the assignment $(\bar{a}_i, b_i)$ falsifies C^i_j, and $\mathfrak{C}_j(\Phi)$ the set of instantiated clauses $\mathcal{C}_{\bar{a},\bar{b}}$ where $(\bar{a}, \bar{b})$ falsifies $\neg C_j$. We now formally define these sets.

▶ **Definition 6.** *Let* Φ *be a* $k\text{-DQBF}^{\mathsf{d}}_{\mathsf{cnf}}$ *formula as in* (2)*. For every* $j \in [m]$ *and* $i \in [k]$*, we define the sets* $C^i_j(\Phi)$*,* $\mathcal{L}_{i,j}(\Phi)$ *and* $\mathfrak{C}_j(\Phi)$*:*

- $C^i_j(\Phi) := \{\ell \in C_j \mid \mathrm{var}(\ell) \in D_i \cup \{y_i\}\}$.
- $\mathcal{L}_{i,j}(\Phi) := \{Y_{i,\bar{a}_i} \oplus b_i \mid (\bar{a}_i, b_i) \simeq \neg C^i_j\}$.
- $\mathfrak{C}_j(\Phi) := \{\mathcal{C}_{\bar{a},\bar{b}} \mid (\bar{a}, \bar{b}) \simeq \neg C_j\}$.

When Φ *is clear from the context, we simply write* C^i_j*,* $\mathcal{L}_{i,j}$ *and* $\mathfrak{C}_j$*.*

We remark that $(\bar{a}, \bar{b}) \simeq \neg C_j$ if and only if $(\bar{a}, \bar{b})$ falsifies C_j, and similarly $(\bar{a}_i, b_i) \simeq \neg C^i_j$ if and only if $(\bar{a}_i, b_i)$ falsifies C^i_j. Note also that $\exp(\Phi) = \bigwedge_{j \in [m]} \bigwedge_{C \in \mathfrak{C}_j} C$ and that the sets $\mathcal{L}_{1,j}, \ldots, \mathcal{L}_{k,j}$ are pairwise variable-disjoint.

[2] There is no dependency structure for $k = 1$.

We now state the property formally.

▶ **Lemma 7.** *Let Φ be as in Equation (2). For every $j \in [m]$, $\mathfrak{C}_j = \mathcal{L}_{1,j} \times \cdots \times \mathcal{L}_{k,j}$.*

Proof. We fix an arbitrary $j \in [m]$. We first prove the "$\subseteq$" direction. Let $\mathcal{C}_{\bar{a},\bar{b}}$ be a clause in $\mathfrak{C}_j$. That is, $(\bar{a}, \bar{b})$ is an assignment that falsifies C_j. Let $\bar{a}_i$ be the restriction of $\bar{a}$ on D_i and b_i be the restriction of $\bar{b}$ on y_i, for every $i \in [k]$. By definition, $\mathcal{C}_{\bar{a},\bar{b}} = \bigvee_{i\in[k]} Y_{i,\bar{a}_i} \oplus b_i$. Since $(\bar{a}, \bar{b})$ falsifies C_j, it is consistent with the cube $\neg C_j$. Hence, for every $i \in [k]$, each $\bar{a}_i, b_i$ is consistent with the cube $\neg C^i_j$. By definition, the literal $Y_{i,\bar{a}_i} \oplus b_i$ belongs to $\mathcal{L}_{i,j}$, for every $i \in [k]$.

Next, we prove the "$\supseteq$" direction. Let $C := (L_1 \vee \cdots \vee L_k) \in \mathcal{L}_{1,j} \times \cdots \times \mathcal{L}_{k,j}$. By definition, for every $i \in [k]$, there is assignment $(\bar{a}_i, b_i)$ such that L_i is the literal $Y_{i,\bar{a}_i} \oplus b_i$ and $(\bar{a}_i, b_i)$ is consistent with the cube $\neg C^i_j$. Due to the disjointness of the dependency sets, all the assignments $(\bar{a}_i, b_i)$'s are pairwise consistent. Let $(\bar{a}, \bar{b})$ be their union $\bigcup_{i\in[k]}(\bar{a}_i, b_i)$.[3] Since each $(\bar{a}_i, b_i)$ is consistent with $\neg C^i_j$, $(\bar{a}, \bar{b})$ is consistent with all of $\neg C^1_j, \dots, \neg C^k_j$. Therefore, $(\bar{a}, \bar{b})$ is a falsifying assignment of C_j. By definition, the clause $\mathcal{C}_{\bar{a},\bar{b}} = \bigvee_{i\in[k]} Y_{i,\bar{a}_i} \oplus b_i$ is in $\mathfrak{C}_j$. ◀

4.2 2-DQBF$^{\mathsf{d}}_{\mathsf{cnf}}$

In this section we will show that sat(2-DQBF$^{\mathsf{d}}_{\mathsf{cnf}}$) is NL-complete, as stated formally in Theorem 8.

▶ **Theorem 8.** sat(2-DQBF$^{\mathsf{d}}_{\mathsf{cnf}}$) *is* NL-*complete.*

Before we proceed to the formal proof, we first review some notation and terminology. Recall that the expansion of a 2-DQBF formula (even when the matrix is in an arbitrary form) is a 2-CNF formula, which can be viewed as a directed graph, called the *implication graph* (of the 2-CNF formula) [1]. The vertices in the implication graph are the literals, and for every clause $(\ell \vee \ell')$ in the formula, there are two edges, $(\neg\ell \to \ell')$ and $(\neg\ell' \to \ell)$.

The following notion of a disimplex will be useful.

▶ **Definition 9** (Disimplex [11]). *Given two sets of vertices $\mathcal{A}, \mathcal{B}$, the* disimplex *from $\mathcal{A}$ to $\mathcal{B}$ is the directed graph $K(\mathcal{A}, \mathcal{B}) := (\mathcal{A} \cup \mathcal{B}, \mathcal{A} \times \mathcal{B})$.*

In other words, a disimplex $K(\mathcal{A}, \mathcal{B})$ is a complete directed bipartite graph where all the edges are oriented from $\mathcal{A}$ to $\mathcal{B}$.

The rest of this subsection is devoted to the proof of Theorem 8. For the rest of this subsection, we fix a 2-DQBF$^{\mathsf{d}}_{\mathsf{cnf}}$ formula $\Phi = \forall \bar{z}_1, \bar{z}_2, \exists y_1(\bar{z}_1), \exists y_2(\bar{z}_2).\ \bigwedge_{j\in[m]} C_j$. We will simply write C^i_j, $\mathcal{L}_{i,j}$ and $\mathfrak{C}_j$ to denote the sets $C^i_j(\Phi)$, $\mathcal{L}_{i,j}(\Phi)$ and $\mathfrak{C}_j(\Phi)$ defined in Definition 6. For a set $\mathcal{L}$ of literals, we denote by $\widehat{\mathcal{L}}$ the set of negated literals in $\mathcal{L}$, i.e., $\widehat{\mathcal{L}} := \{\neg L \mid L \in \mathcal{L}\}$.

We first show that the implication graph of $\exp(\Phi)$ is a finite union of disimplices, and that the length of any shortest path between two vertices is bounded above by $2m$.

▶ **Lemma 10.** *Let $G = (\mathcal{V}, \mathcal{E})$ be the implication graph of* $\exp(\Phi)$. *The set of edges $\mathcal{E}$ can be represented as*

$$\mathcal{E} = \bigcup_{j\in[m]} (\widehat{\mathcal{L}}_{1,j} \times \mathcal{L}_{2,j}) \cup (\widehat{\mathcal{L}}_{2,j} \times \mathcal{L}_{1,j}),$$

[3] Note that, as stated in Section 2.2, we assume that $\bigcup_{i\in[k]} D_i = \bar{x}$.

which is the union of the edge sets of m pairs of disimplices. Moreover, for every two vertices $L, L' \in \mathcal{V}$, if L' is reachable from L, then there exists a path from L to L' of length at most $2m$.

Proof. By definition,

$$\mathcal{E} = \left\{ (\neg Y_{1,\bar{z}_1} \oplus b_1, Y_{2,\bar{z}_2} \oplus b_2), (\neg Y_{2,\bar{z}_2} \oplus b_2, Y_{1,\bar{z}_1} \oplus b_1) \,\middle|\, \varphi[\bar{a}_1^{\bar{z}_1}, \bar{a}_2^{\bar{z}_2}, b_1^{y_1}, b_2^{y_2}] = \bot \right\}.$$

Since any assignment that falsifies φ must falsify some clause C_j in φ, we have

$$\mathcal{E} = \bigcup_{j \in [m]} \bigcup_{\mathcal{C}_{\bar{a},\bar{b}} \in \mathfrak{C}_j} \{ (\neg Y_{1,\bar{z}_1} \oplus b_1, Y_{2,\bar{z}_2} \oplus b_2), (\neg Y_{2,\bar{z}_2} \oplus b_2, Y_{1,\bar{z}_1} \oplus b_1) \}.$$

By Lemma 7, we have $\mathfrak{C}_j = \{ (L_1 \vee L_2) \mid L_1 \in \mathcal{L}_{1,j}, L_2 \in \mathcal{L}_{2,j} \}$ for every $j \in [m]$. Therefore,

$$\mathcal{E} = \bigcup_{j \in [m]} \left(\widehat{\mathcal{L}}_{1,j} \times \mathcal{L}_{2,j} \right) \cup \left(\widehat{\mathcal{L}}_{2,j} \times \mathcal{L}_{1,j} \right).$$

For the second part of the proof, assume, for the sake of contradiction, that $P = (L_0, \ldots, L_n)$ is a shortest path from L to L' with $n > 2m$. Then, by the pigeonhole principle, there must be some $0 \leqslant i_1 < i_2 < n$ such that (L_{i_1}, L_{i_1+1}) and (L_{i_2}, L_{i_2+1}) belongs to the same disimplex $K \subseteq \mathcal{E}$, and thus $(L_{i_1}, L_{i_2+1}) \in K \subseteq \mathcal{E}$. We can then construct a shorter path $P' = (L_0, \ldots, L_{i_1}, L_{i_2+1}, \ldots, L_n)$ from L to L', which contradicts with the assumption that P is a shortest path. ◀

Proof of Theorem 8. For the NL membership, we devise an algorithm by checking the unsatisfiability of $\exp(\Phi)$ directly on these disimplices. We present an NL algorithm that checks the unsatisfiability of $\exp(\Phi)$ by looking for cycles containing both an instantiated literal and its negation in the implication graph $G = (\mathcal{V}, \mathcal{E})$ of $\exp(\Phi)$.[4]

A naïve idea is to first non-deterministically guess a literal L and the paths P from L to $\neg L$ and P' from $\neg L$ to L. However, since $|\mathcal{V}|$ is exponential in $|\bar{x}|$, representing a literal $L \in \mathcal{V}$ takes linear space. We instead make use of Lemma 10 and guess the disimplex each edge of P, P' belongs in, denoted by the sequences $(K(\mathcal{A}_1, \mathcal{B}_1), \ldots, K(\mathcal{A}_n, \mathcal{B}_n))$ and $(K(\mathcal{A}'_1, \mathcal{B}'_1), \ldots, K(\mathcal{A}'_{n'}, \mathcal{B}'_{n'}))$ with $n, n' \in [2m]$, where each $\mathcal{A}, \mathcal{B}$ is of the form $\mathcal{L}_{i,j}$ or $\widehat{\mathcal{L}}_{i,j}$. We then check if

- for every step $j \in [n-1]$, whether there exists some $L_j \in \mathcal{B}_j \cap \mathcal{A}_{j+1}$,
- for every step $j' \in [n'-1]$, whether there exists some $L'_{j'} \in \mathcal{B}'_{j'} \cap \mathcal{A}'_{j'+1}$, and
- whether there exists some $L_0 \in \mathcal{A}_1 \cap \widehat{\mathcal{B}}_n \cap \widehat{\mathcal{A}}'_1 \cap \mathcal{B}'_{n'}$.

We reject if one of the checks fails, and accept if all checks succeed. In the latter case, there are paths $P = (L_0, L_1, \ldots, L_{n-1}, \neg L_0)$ and $P' = (\neg L_0, L'_1, L'_2, \ldots, L'_{n'-1}, L_0)$.

In particular, $\mathcal{L}_{i,j} \cap \mathcal{L}_{i',j'}$ is non-empty if and only if $i = i'$ and C_j^i and $C_{j'}^{i'}$ are consistent. The consistency check can be done by keeping two pointers to the position in the clause using $\log(|\bar{x}| + 2)$ bits per pointer. This can easily be generalised to check the intersection of any constant number of $\mathcal{L}_{i,j}$'s. For $\widehat{\mathcal{L}}_{i,j}$, simply replace C_j^i with the clause $\hat{C}_j^i$ with the sign of y_i flipped if a literal of y_i is present, i.e.,

$$\hat{C}_j^i \coloneqq \left(C_j^i \setminus \{y_i, \neg y_i\} \right) \cup \left(\neg C_j^i \cap \{y_i, \neg y_i\} \right).$$

[4] Recall that a 2-SAT formula φ is unsatisfiable if and only if there is a cycle containing both a literal and its negation in the implication graph of φ.

For the hardness proof, we provide a reduction from 2-SAT to 2-DQBF$^{\mathsf{d}}_{\mathsf{cnf}}$. Let $\varphi = \bigwedge_{j\in[m]}(\ell_{j,1} \vee \ell_{j,2})$ be a 2-CNF formula over the variables $\bar{x} = (x_1, \dots, x_n)$. The idea is to encode the assignment of $\bar{x}$ with a function $f : [n] \to \{\bot, \top\}$ where $f(i)$ represents the assignment of x_i.

In the following, for a literal ℓ, let $\mathrm{ind}(\ell)$ denote the index i where $x_i = \mathrm{var}(\ell)$. We construct the 2-DQBF$^{\mathsf{d}}_{\mathsf{cnf}}$ formula

$$\Psi := \forall \bar{u}_1, \forall \bar{u}_2, \exists y_1(\bar{u}_1), \exists y_2(\bar{u}_2).\, \psi\,,$$

where $\bar{u}_1$ and $\bar{u}_2$ have length $O(\log n)$ for representing the variables in $\bar{x}$ and ψ is a CNF formula that states the following.

- $((\bar{u}_1 = i) \wedge (\bar{u}_2 = i)) \to (y_1 \leftrightarrow y_2)$ for every $i \in [n]$.
- $((\bar{u}_1 = \mathrm{ind}(\ell_{j,1})) \wedge (\bar{u}_2 = \mathrm{ind}(\ell_{j,2}))) \to ((y_1 \leftrightarrow \mathrm{sgn}(\ell_{j,1})) \vee (y_2 \leftrightarrow \mathrm{sgn}(\ell_{j,2})))$ for every $j \in [m]$.

The first item states that the Skolem functions for y_1 and y_2 must be the same. The second item implies that $\bar{a}^{\bar{x}}$ is a satisfying assignment of φ if and only if the function $f_{\bar{a}}$ is a Skolem function for Ψ by encoding the assignment $\bar{a}^{\bar{x}}$ as a function $f_{\bar{a}} \colon [n] \to \{\top, \bot\}$, where $f_{\bar{a}}(i) = \bar{a}^{\bar{x}}(x_i)$. ◀

4.3 k-DQBF$^{\mathsf{d}}_{\mathsf{cnf}}$: $k \geqslant 3$ and Non-Constant k

For $k \geqslant 3$ and even arbitrary DQBF$^{\mathsf{d}}_{\mathsf{cnf}}$, we show that it is NP-complete. Let Φ be as in Equation (2). To show the NP membership, we first show that for every $j \in [m]$, some y_i is responsible for satisfying all the clauses in $\mathfrak{C}_j$.

▶ **Lemma 11.** *Let Φ be as in Equation (2) and let $\bar{Y}$ be the vector of variables in $\exp(\Phi)$. For every $j \in [m]$ and every assignment $\bar{a}$ on $\bar{Y}$, $\bar{a}$ satisfies the CNF formula $\bigwedge_{\mathcal{C}\in\mathfrak{C}_j} \mathcal{C}$ if and only if $\bar{a}$ satisfies the cube $\bigwedge_{L\in\mathcal{L}_{i,j}} L$ for some $i \in [k]$.*

Proof. We first prove the "if" direction. Let $\bar{a}$ be an assignment on $\bar{Y}$. If $\bar{a}$ satisfies the cube $\bigwedge_{L\in\mathcal{L}_{i,j}} L$, then, for every clause $\mathcal{C} \in \mathfrak{C}_\mathrm{j}$, by Lemma 7, there exists some $L \in \mathcal{L}_{i,j} \cap \mathcal{C}$ that is satisfied by $\bar{a}$. Thus, $\mathcal{C}$ is satisfied by L.

For the "only if" direction, assume that $\bar{a}$ does not satisfy the cube $\bigwedge_{L\in\mathcal{L}_{i,j}} L$ for every $i \in [k]$. That is, for every $i \in [k]$, there exists some $L_i \in \mathcal{L}_{i,j}$ such that $\bar{a}(\mathrm{var}(L_i)) \neq \mathrm{sgn}(L_i)$. It follows that the clause $\left(\bigvee_{i\in[k]} L_i\right) \in \mathfrak{C}_j$ is falsified by $\bar{a}$, and thus $\bar{a}$ does not satisfy $\bigwedge_{\mathcal{C}\in\mathfrak{C}_j} \mathcal{C}$. ◀

▶ Remark 12. Recall that $\exp(\Phi) = \bigwedge_{j\in[m]} \bigwedge_{\mathcal{C}\in\mathfrak{C}_j} \mathcal{C}$. Thus, Lemma 11 can be reformulated as follows. For every assignment $\bar{a}^{\bar{Y}}$, $\bar{a}^{\bar{Y}}$ satisfies $\exp(\Phi)$ if and only if there is a function $\xi : [m] \to [k]$ such that for every $j \in [m]$, $\bar{a}^{\bar{Y}}$ satisfies the cube $\bigwedge_{L\in\mathcal{L}_{\xi(j),j}} L$. Intuitively, the function ξ is the mapping that maps index j to index i in the statement in Lemma 11. This formulation will be useful later on.

The next lemma shows the NP membership of sat(DQBF$^{\mathsf{d}}_{\mathsf{cnf}}$).

▶ **Lemma 13.** sat(DQBF$^{\mathsf{d}}_{\mathsf{cnf}}$) *is in* NP.

Proof. Consider a DQBF$^{\mathsf{d}}_{\mathsf{cnf}}$ formula:

$$\Phi = \forall \bar{z}_1, \dots, \forall \bar{z}_k, \exists y_1(\bar{z}_1), \dots, \exists y_k(\bar{z}_k).\ \bigwedge_{j\in[m]} C_j$$

with k existential variables and m clauses.

By the reformulation of Lemma 11 in Remark 12, an assignment $\bar{a}$ on $\bar{Y}$ satisfies $\exp(\Phi)$ if and only if there exists a mapping $\xi\colon [m] \to [k]$ such that $\bar{a}^{\bar{Y}}$ satisfies $\bigwedge_{j\in[m]} \bigwedge_{L\in\mathcal{L}_{\xi(j),j}} L$, or equivalently, if there exists a partition $\{S_i\}_{i\in[k]}$ of $[m]$ such that for each $i \in [k]$, the following QBF Φ_i is satisfiable:

$$\Phi_i = \forall \bar{z}_i, \exists y_i.\ \bigwedge_{j\in S_i} C^i_j\ .$$

Note that since Φ_i contains only one existential variable and it depends on all universal variables, checking the satisfiability of Φ_i is in P using Lemma 3.[5] An NP algorithm guesses the partition $\{S_i\}_{i\in[k]}$ and verifies that Φ_i is satisfiable for every $i \in [k]$. ◀

▶ **Theorem 14.** $\mathsf{sat}(k\text{-DQBF}^{\mathsf{d}}_{\mathsf{cnf}})$ *for every* $k \geqslant 3$ *and* $\mathsf{sat}(\text{DQBF}^{\mathsf{d}}_{\mathsf{cnf}})$ *are* NP*-complete.*

Proof. By Lemma 13, $\mathsf{sat}(\text{DQBF}^{\mathsf{d}}_{\mathsf{cnf}})$ is in NP. Since $k\text{-DQBF}^{\mathsf{d}}_{\mathsf{cnf}} \subseteq \text{DQBF}^{\mathsf{d}}_{\mathsf{cnf}}$, $\mathsf{sat}(k\text{-DQBF}^{\mathsf{d}}_{\mathsf{cnf}})$ is also in NP for every constant k.

For the NP-hardness, a reduction from 3-SAT to $3\text{-DQBF}^{\mathsf{d}}_{\mathsf{cnf}}$ can be done analogous to that of Theorem 8. A complete proof can be found in Section B.2. Since adding more existential variables only increases the complexity, $\mathsf{sat}(k\text{-DQBF}^{\mathsf{d}}_{\mathsf{cnf}})$ for every $k \geqslant 3$ and $\mathsf{sat}(\text{DQBF}^{\mathsf{d}}_{\mathsf{cnf}})$ are also NP-hard. ◀

4.4 k-DQBF$^{\alpha}_{\mathsf{cnf}}$: Different Dependency Structure

It has been shown in [18] that $\mathsf{sat}(\text{DQBF}^{\mathsf{de}}_{\mathsf{cnf}})$ is Σ^{P}_3-complete and $\mathsf{sat}(\text{DQBF}^{\mathsf{dec}}_{\mathsf{cnf}})$ is NEXP-complete. Since $\text{DQBF}^{\mathsf{dec}}_{\mathsf{cnf}} \subseteq \text{DQBF}^{\mathsf{ds}}_{\mathsf{cnf}} \subseteq \text{DQBF}$ and $\mathsf{sat}(\text{DQBF})$ is also NEXP-complete, we know $\mathsf{sat}(\text{DQBF}^{\mathsf{ds}}_{\mathsf{cnf}})$ is NEXP-complete. In this section, we show a surprising result that, when k is a constant, $\mathsf{sat}(k\text{-DQBF}^{\alpha}_{\mathsf{cnf}})$ has the same complexity as k-SAT and $\mathsf{sat}(k\text{-DQBF}^{\mathsf{d}}_{\mathsf{cnf}})$ for every $\alpha \in \{\mathsf{de}, \mathsf{dec}, \mathsf{ds}\}$. Since $k\text{-DQBF}^{\mathsf{d}}_{\mathsf{cnf}} \subseteq k\text{-DQBF}^{\mathsf{de}}_{\mathsf{cnf}} \subseteq k\text{-DQBF}^{\mathsf{dec}}_{\mathsf{cnf}} \subseteq k\text{-DQBF}^{\mathsf{ds}}_{\mathsf{cnf}}$, it suffices to show the results for $\mathsf{sat}(k\text{-DQBF}^{\mathsf{ds}}_{\mathsf{cnf}})$.

We start with $\mathsf{sat}(2\text{-DQBF}^{\mathsf{ds}}_{\mathsf{cnf}})$.

▶ **Theorem 15.** $\mathsf{sat}(2\text{-DQBF}^{\mathsf{ds}}_{\mathsf{cnf}})$ *is* NL*-complete.*

Proof. Since $2\text{-DQBF}^{\mathsf{ds}}_{\mathsf{cnf}} \supseteq 2\text{-DQBF}^{\mathsf{d}}_{\mathsf{cnf}}$, the hardness follows from Theorem 8. For NL membership, consider a $2\text{-DQBF}^{\mathsf{ds}}_{\mathsf{cnf}}$ formula $\Phi \coloneqq \forall \bar{x}, \exists y_1(D_1), \exists y_2(D_2).\, \varphi$. First, we check whether D_1 and D_2 are disjoint using only logarithmic space. (See Remark 1.) If D_1 and D_2 are disjoint, we use the algorithm from Theorem 8 to determine its satisfiability. Otherwise, without loss of generality, we may assume that $D_1 \subseteq D_2$. We will show that this case can be decided in deterministic logarithmic space. Indeed, in this case Φ is a standard QBF and we can perform a level-ordered Q-resolution proof [16]. Since there are only two existential variables, any proof uses at most four clauses, and we can simply iterate through all 4-tuples of clause indices and check whether Q-resolution can be performed.

In the following, we give an alternative proof that works directly on the semantics of QBF. To ease notation, we write $\bar{z}_1 \coloneqq D_1$, $\bar{z}_2 \coloneqq D_2 \setminus D_1$, and $\bar{z}_3 \coloneqq \bar{x} \setminus D_2$. Note that Φ is equivalent to a QBF

$$\begin{aligned}
\Psi &= \forall \bar{z}_1, \exists y_1, \forall \bar{z}_2, \exists y_2, \forall \bar{z}_3.\, \varphi \\
&= \forall \bar{z}_1, \exists y_1, \forall \bar{z}_2.\, (\forall \bar{z}_3.\, \varphi[\bot^{y_2}]) \lor (\forall \bar{z}_3.\, \varphi[\top^{y_2}]) \\
&= \forall \bar{z}_1.\, (\forall \bar{z}_2.\, (\forall \bar{z}_3.\, \varphi[(\bot^{y_1}, \bot^{y_2})]) \lor (\forall \bar{z}_3.\, \varphi[(\bot^{y_1}, \top^{y_2})])) \\
&\qquad \lor (\forall \bar{z}_2.\, (\forall \bar{z}_3.\, \varphi[(\top^{y_1}, \bot^{y_2})]) \lor (\forall \bar{z}_3.\, \varphi[(\top^{y_1}, \top^{y_2})]))\ ,
\end{aligned}$$

[5] In fact, it is in L, as shown later in Theorem 19.

which is false if and only if there are assignments $\bar{a}^{\bar{z}_1}$, $\bar{b}^{\bar{z}_2}$, and $\bar{c}^{\bar{z}_2}$ such that

$$\forall \bar{z}_3.\, \varphi[(\bot^{y_1}, \bot^{y_2}, \bar{a}^{\bar{z}_1}, \bar{b}^{\bar{z}_2})] \vee \forall \bar{z}_3.\, \varphi[(\bot^{y_1}, \top^{y_2}, \bar{a}^{\bar{z}_1}, \bar{b}^{\bar{z}_2})] \vee \forall \bar{z}_3.\, \varphi[(\top^{y_1}, \bot^{y_2}, \bar{a}^{\bar{z}_1}, \bar{c}^{\bar{z}_2})]$$
$$\vee \forall \bar{z}_3.\, \varphi[(\top^{y_1}, \top^{y_2}, \bar{a}^{\bar{z}_1}, \bar{c}^{\bar{z}_2})]$$

is false. Since each of the four disjuncts is still in CNF, the formula is false if and only if each disjunct has a falsified clause. This is equivalent to finding four clauses $C_1, C_2, C_3, C_4 \in \varphi$ such that

- the clauses C_1, C_2, C_3, C_4 are consistent on the variables in $\bar{z}_1$,
- the clauses C_1, C_2 are consistent on the variables in $\bar{z}_2$,
- the clauses C_3, C_4 are consistent on the variables in $\bar{z}_2$, and
- $\neg C_1$, $\neg C_2$, $\neg C_3$, and $\neg C_4$ are consistent with $\neg y_1 \wedge \neg y_2$, $\neg y_1 \wedge y_2$, $y_1 \wedge \neg y_2$, and $y_1 \wedge y_2$, respectively.

To find such clauses, we can iterate through all 4-tuples of clause indices and check whether the properties hold. ◀

Next, we show that for every $k \geqslant 3$, $\mathsf{sat}(k\text{-DQBF}^{\mathsf{ds}}_{\mathsf{cnf}})$ is NP-complete, just like k-SAT.

▶ **Theorem 16.** *For every constant $k \geqslant 3$, $\mathsf{sat}(k\text{-DQBF}^{\mathsf{ds}}_{\mathsf{cnf}})$ is* NP*-complete.*

Before we present the proof of Theorem 16, we note that since $k\text{-DQBF}^{\mathsf{de}}_{\mathsf{cnf}} \subseteq k\text{-DQBF}^{\mathsf{dec}}_{\mathsf{cnf}} \subseteq k\text{-DQBF}^{\mathsf{ds}}_{\mathsf{cnf}}$, we obtain the following results as a corollary of Theorems 8 and 14–16.

▶ **Corollary 17.** $\mathsf{sat}(k\text{-DQBF}^{\mathsf{de}}_{\mathsf{cnf}})$ *and* $\mathsf{sat}(k\text{-DQBF}^{\mathsf{dec}}_{\mathsf{cnf}})$ *are* NL*-complete when $k = 2$ and* NP*-complete when $k \geqslant 3$.*

The rest of this section is devoted to the proof of Theorem 16.

Proof of Theorem 16. We will consider the membership proof. The hardness follows from Theorem 14. We fix a $k\text{-DQBF}^{\mathsf{ds}}_{\mathsf{cnf}}$ formula:

$$\Phi := \forall \bar{x}, \exists y_1(D_1), \ldots, \exists y_k(D_k).\ \bigwedge_{j \in [m]} C_j. \tag{3}$$

Without loss of generality, we may assume that no existential variable has an empty dependency set, since our NP algorithm can guess an assignment to such variables at the outset. By Lemma 2, we may also assume that every universal variable appears in some dependency set. We say that a dependency set D_i is maximal if there is no j where $D_i \subsetneq D_j$. An existential variable y_i is maximal if its dependency set is maximal.

To decide the satisfiability of Φ, our algorithm works by recursion on the number of existential variables. The base case is when there is only one existential variable. This case can be decided in polynomial time and, in fact, in deterministic logspace. See, e.g., Theorem 19.

For the induction step, we pick a maximal variable y_t. There are two cases.

Case 1: $D_t = \bar{x}$. We apply Lemma 3 and eliminate y_t, resulting in a formula with one less existential variable and $O(m^2)$ clauses. We then proceed recursively.

Case 2: $D_t \neq \bar{x}$. We deal with this case by generalising the technique in Lemma 13.

Let $\{i_1, \ldots, i_p\} = \{i \mid D_i \subseteq D_t\}$ and $\{i'_1, \ldots, i'_q\} = \{i' \mid D_{i'} \cap D_t = \emptyset\}$. For each $j \in [m]$, we partition C_j into two clauses:

$$C_j^{+t} := \{\ell \mid \mathrm{dep}(\ell) \subseteq D_t\}$$
$$C_j^{-t} := C_j \setminus C_j^{+t}$$

Intuitively, C_j^{+t} is the subclause of C_j that includes all the literals with dependency sets inside D_t. On the other hand, C_j^{-t} is the subclause that contains the rest of the literals. Due to the laminar structure of the dependency sets and that y_t is a maximal variable, $C_j^{-t} = \{\ell \mid \mathrm{dep}(\ell) \cap D_t = \emptyset\}$.

For a function $\xi : [m] \to \{+t, -t\}$, we define two formulas:

$$\Phi_{+t,\xi} := \forall \bar{x}, \exists y_{i_1}(D_{i_1}), \ldots, \exists y_{i_p}(D_{i_p}). \bigwedge_{j:\xi(j)=+t} C_j^{+t}$$

$$\Phi_{-t,\xi} := \forall \bar{x}, \exists y_{i'_1}(D_{i'_1}), \ldots, \exists y_{i'_q}(D_{i'_q}). \bigwedge_{j:\xi(j)=-t} C_j^{-t}$$

We have the following lemma.

▶ **Lemma 18.** *Φ is satisfiable if and only if there is a function $\xi : [m] \to \{+t, -t\}$ such that $\Phi_{+t,\xi}$ and $\Phi_{-t,\xi}$ are both satisfiable.*

Note that guessing ξ requires m bits. The algorithm guesses the function ξ and verifies recursively that both $\Phi_{+t,\xi}$ and $\Phi_{-t,\xi}$ are satisfiable. Since the algorithm terminates after k steps, and k is a constant, and the number of clauses constructed in each recursive step is at most quadratically many, each step can be done in polynomial time. ◀

The proof of Lemma 18 is a generalisation of Lemma 11. Let $\bar{Y}$ be the vector of variables in the expansion $\exp(\Phi)$. We can show that an assignment $\bar{a}$ on $\bar{Y}$ satisfies $\exp(\Phi)$ if and only if it satisfies $\exp(\Phi_{+t,\xi})$ and $\exp(\Phi_{-t,\xi})$ for some function ξ. A detailed proof can be found in the appendix.

5 Complexity of sat(k-DQBF$_{\mathsf{cnf}}$)

In this section, we remove the constraint on the dependency structure and consider k-DQBF$_{\mathsf{cnf}}$. The case $k = 1$ can be solved very efficiently.

▶ **Theorem 19.** sat(1-DQBF$_{\mathsf{cnf}}$) *is in* L.

Proof. Let $\Phi = \forall \bar{z}_1, \bar{z}_2, \exists y(\bar{z}_1). \bigwedge_{j\in[m]} C_j$. Similar to the proof of Theorem 15, to show unsatisfiability, it suffices to find $C_1, C_2 \in \varphi$ such that

- C_1, C_2 are consistent on the variables in $\bar{z}_1$, and
- $\neg C_1$ and $\neg C_2$ are consistent with $\neg y$ and y, respectively.

The correctness follows from the same reasoning. ◀

Next, we consider the case where $k = 2$.

▶ **Theorem 20.** sat(2-DQBF$_{\mathsf{cnf}}$) *is* coNP*-complete.*

Proof. For membership, we give an NP algorithm for checking unsatisfiability. Let $\Phi := \forall \bar{x}, \exists y_1(D_1), \exists y_2(D_2). \varphi$ be be a 2-DQBF$_{\mathsf{cnf}}$ formula. Let $\bar{z} = D_1 \cap D_2$. Note that for every assignment $\bar{a}$ on $\bar{z}$, the induced formula $\Phi[\bar{a}]$ is a 2-DQBF$^{\mathsf{d}}_{\mathsf{cnf}}$ formula, the satisfiability of which can be decided in polynomial time by Theorem 8. Therefore, to decide whether Φ is unsatisfiable, we can guess an assignment $\bar{a}$ on $\bar{z}$ and accept if and only if $\Phi[\bar{a}]$ is not satisfiable.

For hardness, we provide a reduction from the 3-DNF tautology problem to sat(2-DQBF$_{\mathsf{cnf}}$). Let $\varphi = \bigvee_{j\in[m]} Q_j$ be a 3-DNF formula over the variables $\bar{x} = (x_1, \ldots, x_n)$, where each $Q_j = (\ell_{j,1} \wedge \ell_{j,2} \wedge \ell_{j,3})$ is a 3-literal cube. We construct the following 2-DQBF$_{\mathsf{cnf}}$ formula:

$$\Psi = \forall \bar{x}, \forall \bar{u}_1, \forall \bar{u}_2, \exists y_1(\bar{x}, \bar{u}_1), \exists y_2(\bar{x}, \bar{u}_2). \psi_1 \wedge \psi_2 \wedge \psi_3 \wedge \psi_4\,,$$

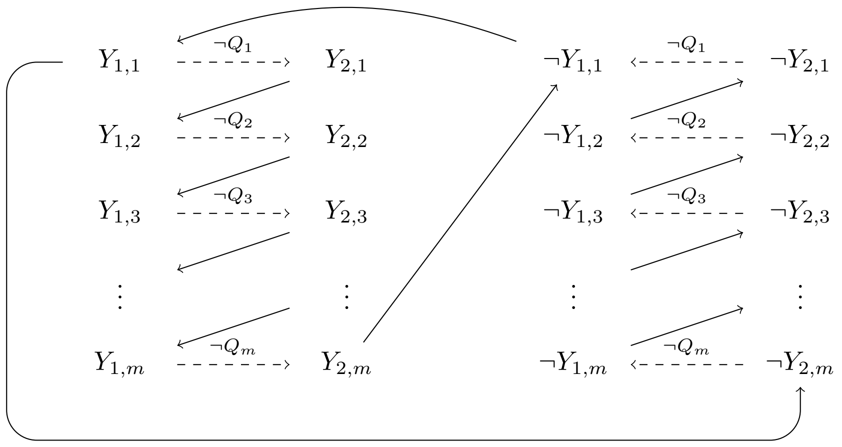

Figure 1 The implication graph $G_{\bar{a}}$. Each dashed edge $\overset{\neg Q_i}{\dashrightarrow}$ is present if and only if $\bar{a}^{\bar{x}}$ falsifies Q_i.

where $\bar{u}_1, \bar{u}_2$ have length $O(\log m)$ for representing the numbers in $[m]$ and $\psi_1, \ldots, \psi_4$ are as follows.

$$\begin{aligned}
\psi_1 &:= (\bar{u}_1 = 1) \rightarrow y_1 \\
\psi_2 &:= \bigwedge_{j \in [m-1]} \Big(\big((\bar{u}_1 = j+1) \wedge (\bar{u}_2 = j) \big) \rightarrow (y_2 \rightarrow y_1) \Big) \\
\psi_3 &:= \big((\bar{u}_1 = 1) \wedge (\bar{u}_2 = m) \big) \rightarrow (y_2 \rightarrow \neg y_1) \\
\psi_4 &:= \bigwedge_{j \in [m]} \bigwedge_{i \in [3]} \Big(\big((\bar{u}_1 = j) \wedge (\bar{u}_2 = j) \wedge \neg \ell_{j,i} \big) \rightarrow (y_1 \rightarrow y_2) \Big)
\end{aligned}$$

We claim that φ is a tautology if and only if Ψ is satisfiable. To see this, we fix an arbitrary assignment $\bar{a}$ on $\bar{x}$ and consider the induced formula $\Psi[\bar{a}]$. Note that $\Psi[\bar{a}]$ is a 2-DQBF$^{\mathsf{d}}_{\mathsf{cnf}}$ formula with universal variables $\bar{u}_1, \bar{u}_2$. Since $|\bar{u}_1| = |\bar{u}_2| = \log m$, the expansion $\exp(\Psi[\bar{a}])$ is a 2-CNF formula with $2m$ variables $Y_{1,1}, \ldots, Y_{1,m}, Y_{2,1}, \ldots, Y_{2,m}$. Here we abuse the notation and write $Y_{i,j}$ instead of $Y_{i,\bar{a}}$ where $\bar{a}$ is the binary representation of j.

It can be easily verified that the implication graph $G_{\bar{a}}$ of the expansion $\exp(\Psi[\bar{a}])$ is as shown in Figure 1, where a dashed edge $\overset{\neg Q_j}{\dashrightarrow}$ is present if and only if $\bar{a}$ falsifies the cube Q_j. Indeed, ψ_1 states that the edge $\neg Y_{1,1} \rightarrow Y_{1,1}$ is present. ψ_2 states that the edges $Y_{2,i} \rightarrow Y_{1,i+1}$ and $\neg Y_{1,i+1} \rightarrow \neg Y_{2,i}$ are present for every $i \in [m-1]$. ψ_3 states that the edges $Y_{2,m} \rightarrow \neg Y_{1,1}$ and $Y_{1,1} \rightarrow \neg Y_{2,m}$ are present. Finally, ψ_4 states that the dashed edges $Y_{1,j} \overset{\neg Q_j}{\dashrightarrow} Y_{2,j}$ and $Y_{1,j} \overset{\neg Q_j}{\dashrightarrow} Y_{2,j}$ are present if $\bar{a}^{\bar{x}}$ falsifies Q_j, for every $j \in [m]$. This implies that $\bar{a}^{\bar{x}}$ falsifies all cubes in φ if and only if there exists a cycle in $G_{\bar{a}}$. Since a cycle in $G_{\bar{a}}$ (if exists) contains contradicting literals, $\bar{a}^{\bar{x}}$ falsifies all cubes in φ if and only if $\Psi[\bar{a}]$ is not satisfiable. Since the assignment $\bar{a}$ is arbitrary, φ is a tautology if and only if Ψ is satisfiable. ◀

Next, we consider k-DQBF$_{\mathsf{cnf}}$. Note that 3-DQBF$_{\mathsf{cnf}}$ subsumes both 3-DQBF$^{\mathsf{d}}_{\mathsf{cnf}}$ and 2-DQBF$_{\mathsf{cnf}}$. Thus, sat(3-DQBF$_{\mathsf{cnf}}$) is is both NP-hard and coNP-hard. We improve these results by showing that it is Π_2^{P}-hard.

▶ **Theorem 21.** sat(3-DQBF$_{\mathsf{cnf}}$) *is* Π_2^{P}*-hard.*

Proof. We show this by reduction from Π_2-QBF in 3-CNF. The reduction is similar to the one in Theorem 14, except that the existential variables are allowed to depend on common variables, thereby enabling the encoding of a two-level QBF.

Let $\varphi = \forall \bar{z}, \exists \bar{x}. \bigwedge_{j\in[m]}(\ell_{j,1} \vee \ell_{j,2} \vee \ell_{j,3})$ be a Π_2-QBF in 3-CNF. Let $\bar{x} = (x_1, \dots, x_n)$ and for a literal ℓ, let $\mathrm{ind}(\ell)$ denote the index i where $x_i = \mathrm{var}(\ell)$.

We construct the following 3-DQBF$_{\mathsf{cnf}}$ formula:

$$\Psi \coloneqq \forall \bar{z}, \forall \bar{u}_1, \forall \bar{u}_2, \forall \bar{u}_3, \exists y_1(\bar{z}, \bar{u}_1), \exists y_2(\bar{z}, \bar{u}_2), \exists y_3(\bar{z}, \bar{u}_3).\, \psi_1 \wedge \psi_2 \wedge \psi_3\,,$$

where $\bar{u}_1, \bar{u}_2, \bar{u}_3$ have length $\log n$ for representing the variables in $\bar{x}$ and each ψ_1, ψ_2, ψ_3 is the following CNF formula:

$$\psi_1 \coloneqq \bigwedge_{i\in[n]} \Big(\big((\bar{u}_1 = i) \wedge (\bar{u}_2 = i)\big) \to (y_1 \leftrightarrow y_2)\Big)$$
$$\psi_2 \coloneqq \bigwedge_{i\in[n]} \Big(\big((\bar{u}_1 = i) \wedge (\bar{u}_3 = i)\big) \to (y_1 \leftrightarrow y_3)\Big)$$
$$\psi_3 \coloneqq \bigwedge_{j\in[m]} \Big((\sigma_{j,1} \wedge \sigma_{j,2} \wedge \sigma_{j,3}) \to (\tau_{j,1} \vee \tau_{j,2} \vee \tau_{j,3})\Big)\,,$$

where for every $i \in [3]$, $\sigma_{j,i}$ and $\tau_{j,i}$ are defined as

$$(\sigma_{j,i}, \tau_{j,i}) = \begin{cases} (\bar{u}_i = \mathrm{ind}(\ell_{j,i}), y_i \leftrightarrow \mathrm{sgn}(\ell_{j,i})) & \text{if } \mathrm{var}(\ell_{j,i}) \text{ is existential in } \varphi \\ (\top, \ell_{j,i}) & \text{otherwise.} \end{cases}$$

The intuition is that a vector of Skolem functions $f_1, \dots, f_n$ for $\bar{x}$ in φ can be encoded as a single function $g : \{\top, \bot\}^{|\bar{z}|} \times [n] \to \{\top, \bot\}$, which then corresponds to a Skolem function for each y_i in Ψ. The formulas ψ_1 and ψ_2 state that for every Skolem functions (g_1, g_2, g_3) for Ψ, all functions g_1, g_2, g_3 must be the same. Then, ψ_3 states that the Skolem functions $f_1, \dots, f_n$ for φ (if exists) must satisfies all the clauses $C_1, \dots, C_m$. Therefore, φ is a true QBF if and only if Ψ is satisfiable. ◀

We next show a Tseitin-like transformation that requires only three additional existential variables. This is in contrast with the standard Tseitin transformation that requires as many existential variables as the length of the input formula. Our new Tseitin-like transformation leads to new hardness results for sat(5-DQBF$_{\mathsf{cnf}}$) and sat(6-DQBF$_{\mathsf{cnf}}$).

▶ **Lemma 22** (Reformulated Tseitin Transformation)**.** *Given a k-DQBF formula, we can construct in polynomial time an equisatisfiable $(k+3)$-DQBF$_{\mathsf{cnf}}$ formula.*

Proof. Let $\Phi = \forall \bar{x}, \exists y_1(\bar{z}_1), \dots, \exists y_k(\bar{z}_k).\, \varphi$ be a k-DQBF formula where φ is a circuit with gates $g_1, \dots, g_m$. We assume, without loss of generality, that

$$g_i = \begin{cases} x_i & \text{for every } 1 \leqslant i \leqslant n \\ y_{i-n} & \text{for every } n+1 \leqslant i \leqslant n+k \\ f_i(g_{l_i}, g_{r_i}) & \text{for every } n+k+1 \leqslant i \leqslant m, \end{cases}$$

where $l_i, r_i \in [i-1]$ are the indices of the two fan-ins of the gate g_i implementing the Boolean function f_i, and that g_m corresponds to the primary output of the circuit.

We construct the $(k+3)$-DQBF$_{\mathsf{cnf}}$ formula

$$\Psi \coloneqq \forall \bar{x}, \forall \bar{u}_1, \forall \bar{u}_2, \forall \bar{u}_3, \exists y_1(\bar{z}_1), \dots, \exists y_k(\bar{z}_k), \exists t_1(\bar{x}, \bar{u}_1), \exists t_2(\bar{x}, \bar{u}_2), \exists t_3(\bar{x}, \bar{u}_3).\, \psi\,,$$

where $\bar{u}_1, \bar{u}_2, \bar{u}_3$ have length $\log m$ for representing the numbers in $[m]$ and ψ is a CNF formula with clauses encoding

- $((\bar{u}_1 = i) \wedge (\bar{u}_2 = i)) \rightarrow (t_1 \leftrightarrow t_2)$ for every $i \in [m]$,
- $((\bar{u}_1 = i) \wedge (\bar{u}_3 = i)) \rightarrow (t_1 \leftrightarrow t_3)$ for every $i \in [m]$,
- $(\bar{u}_1 = i) \rightarrow (t_1 \leftrightarrow x_i)$ for every $i \in [n]$,
- $(\bar{u}_1 = n + i) \rightarrow (t_1 \leftrightarrow y_i)$ for every $i \in [k]$,
- $((\bar{u}_1 = i) \wedge (\bar{u}_2 = l_i) \wedge (\bar{u}_3 = r_i)) \rightarrow (t_1 \leftrightarrow f_i(t_2, t_3))$ for every $n + k + 1 \leqslant i \leqslant m$,
- $(\bar{u}_1 = m) \rightarrow t_1$.

The intuitive meaning is as follows. The first two items state that the Skolem functions for t_1, t_2, t_3 must be the same. The next three state that the Skolem function g for t_1 encodes the values of the gates, where $g(\bar{x}, \bar{u}_1)$ is the value of gate j when $\bar{u}_1$ is the binary representation of j. The last item encodes that the output of the circuit must be true. The correctness proof is routine and hence omitted. ◀

In particular, by combining Lemma 22 with the fact that sat(2-DQBF) and sat(3-DQBF) are PSPACE-complete and NEXP-complete [14], we establish the hardness of sat(5-DQBF$_{\mathsf{cnf}}$) and sat(6-DQBF$_{\mathsf{cnf}}$).

▶ **Theorem 23.** sat(5-DQBF$_{\mathsf{cnf}}$) *is* PSPACE-*hard and* sat(6-DQBF$_{\mathsf{cnf}}$) *is* NEXP-*complete.*

6 Conclusions and Future Work

While sat(k-DQBF$^{\mathsf{d}}_{\mathsf{dnf}}$) is as hard as sat($k$-DQBF), we observe a range of differing complexity results in the CNF case. For the case of sat(k-DQBF$^{\mathsf{d}}_{\mathsf{cnf}}$), we show that it is in fact as easy as k-SAT – exponentially easier than sat(k-DQBF). Generalising the results in [18], we also show that sat(DQBF$^{\mathsf{d}}_{\mathsf{cnf}}$) is NP-complete and that sat(k-DQBF$^{\alpha}_{\mathsf{cnf}}$) has the same complexity as k-SAT for $\alpha \in \{\mathsf{d}, \mathsf{de}, \mathsf{dec}, \mathsf{ds}\}$. For the case of k-DQBF$_{\mathsf{cnf}}$, we show that it is only coNP-complete when $k = 2$ (whereas sat(2-DQBF) is PSPACE-complete) and of the same NEXP-complete complexity as sat(DQBF) when $k \geqslant 6$. These results show that, when parametrising DQBF with the number of existential variables, it is more natural to consider DNF as the normal form for the matrix, analogous to how CNF is considered the standard form for SAT.

The exact complexity of sat(k-DQBF$_{\mathsf{cnf}}$) is yet to be discovered for $k = 3$, 4, and 5. In particular, the best-known membership result is still that they are in NEXP. We leave this for future work.

References

1 Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Inf. Process. Lett.*, 8(3):121–123, 1979. doi:10.1016/0020-0190(79)90002-4.

2 V. Balabanov and J.-H. R. Jiang. Reducing satisfiability and reachability to DQBF. In *QBF Workshop*, 2015.

3 Valeriy Balabanov, Hui-Ju Katherine Chiang, and Jie-Hong R. Jiang. Henkin quantifiers and Boolean formulae: A certification perspective of DQBF. *Theoretical Computer Science*, 523:86–100, 2014. doi:10.1016/j.tcs.2013.12.020.

4 Armin Biere, Mathias Fleury, Nils Froleyks, and Marijn J. H. Heule. The SAT museum. In Matti Järvisalo and Daniel Le Berre, editors, *Proceedings of the 14th International Workshop on Pragmatics of SAT*, volume 3545, pages 72–87. CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3545/paper6.pdf.

5 Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185. IOS Press, 2009.

6 Uwe Bubeck. *Model-Based Transformations for Quantified Boolean Formulas.* PhD thesis, University of Paderborn, 2010.

7 Uwe Bubeck and Hans Kleine Büning. Dependency quantified Horn formulas: Models and complexity. In Armin Biere and Carla P. Gomes, editors, *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings*, volume 4121, pages 198–211. Springer, 2006. `doi:10.1007/11814948_21`.

8 Uwe Bubeck and Hans Kleine Büning. Rewriting (dependency-)quantified 2-CNF with arbitrary free literals into existential 2-HORN. In Ofer Strichman and Stefan Szeider, editors, *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing - SAT 2010, 13th International Conference*, volume 6175, pages 58–70. Springer, 2010. `doi:10.1007/978-3-642-14186-7_7`.

9 Fa-Hsun Chen, Shen-Chang Huang, Yu-Cheng Lu, and Tony Tan. Reducing NEXP-complete problems to DQBF. In Alberto Griggio and Neha Rungta, editors, *Proceedings of the 22nd Conference on Formal Methods in Computer-Aided Design*, pages 199–204, 2022. `doi:10.34727/2022/isbn.978-3-85448-053-2_26`.

10 Johannes Klaus Fichte, Daniel Le Berre, Markus Hecher, and Stefan Szeider. The silent (r)evolution of SAT. *Commun. ACM*, 66(6):64–72, 2023. `doi:10.1145/3560469`.

11 Ana Paulina Figueroa and Bernardo Llano. An infinite family of self-diclique digraphs. *Appl. Math. Lett.*, 23(5):630–632, 2010. `doi:10.1016/j.aml.2010.01.026`.

12 Andreas Fröhlich, Gergely Kovásznai, Armin Biere, and Helmut Veith. iDQ: Instantiation-based DQBF solving. In *Proceedings of the Fifth Pragmatics of SAT workshop*, volume 27, pages 103–116, 2014. `doi:10.29007/1s5k`.

13 Long-Hin Fung, Che Cheng, Yu-Wei Fan, Tony Tan, and Jie-Hong Roland Jiang. 2-DQBF solving and certification via property-directed reachability analysis. In *Proceedings of the Conference on Formal Methods in Computer-Aided Design*, pages 197–207. IEEE, 2024. `doi:10.34727/2024/isbn.978-3-85448-065-5_25`.

14 Long-Hin Fung and Tony Tan. On the complexity of k-DQBF. In *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing*, volume 271, pages 10:1–10:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. `doi:10.4230/LIPIcs.SAT.2023.10`.

15 Robert Ganian, Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Fixed-parameter tractability of dependency QBF with structural parameters. In Diego Calvanese, Esra Erdem, and Michael Thielscher, editors, *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12-18, 2020*, pages 392–402, 2020. `doi:10.24963/kr.2020/40`.

16 Mikolás Janota and João Marques-Silva. Expansion-based QBF solving versus Q-resolution. *Theor. Comput. Sci.*, 577(C):25–42, 2015. `doi:10.1016/j.tcs.2015.01.048`.

17 G. Peterson, J. Reif, and S. Azhar. Lower bounds for multiplayer noncooperative games of incomplete information. *Computers & Mathematics with Applications*, 41(7):957–992, 2001. `doi:10.1016/S0898-1221(00)00333-3`.

18 Christoph Scholl, Jie-Hong Roland Jiang, Ralf Wimmer, and Aile Ge-Ernst. A PSPACE subclass of dependency quantifed Boolean formulas and its effective solving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1584–1591. AAAI Press, 2019. `doi:10.1609/aaai.v33i01.33011584`.

19 G. Tseitin. On the complexity of derivation in propositional calculus. In *Studies in Constructive Mathematics and Mathematical Logic, Part II*, 1968.

20 Ralf Wimmer, Karina Gitina, Jennifer Nist, Christoph Scholl, and Bernd Becker. Preprocessing for DQBF. In Marijn Heule and Sean A. Weaver, editors, *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing*, volume 9340, pages 173–190, Cham, 2015. Springer International Publishing. `doi:10.1007/978-3-319-24318-4_13`.

21 Lintao Zhang. Solving QBF with combined conjunctive and disjunctive normal form. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 143–150. AAAI Press, 2006. URL: `http://www.aaai.org/Library/AAAI/2006/aaai06-023.php`.

A Additional Background

We provide additional background knowledge in this section.

A.1 Tseitin Transformation

Tseitin transformation is a standard technique to turn an arbitrary Boolean satisfiability problem into an equisatisfiable one in 3-CNF form [19]. It can be directly lifted to QBF and DQBF by allowing the Tseitin variables to depend on every universal variable.

Given a DQBF

$$\Phi = \forall \bar{x}, \exists y_1(\bar{z}_1), \ldots, \exists y_k(\bar{z}_k).\, \varphi\,,$$

where φ is a circuit with gates $g_1, \ldots, g_m$, we assume, without loss of generality, that

$$g_i = \begin{cases} x_i & \text{for every } 1 \leqslant i \leqslant n \\ y_{i-n} & \text{for every } n+1 \leqslant i \leqslant n+k \\ f_i(g_{l_i}, g_{r_i}) & \text{for every } n+k+1 \leqslant i \leqslant m, \end{cases}$$

where $l_i, r_i \in [i-1]$ are the indices of the two fanins of the gate g_i implementing the Boolean function f_i.

The core idea of Tseitin transformation is that we can introduce a fresh variable t_i for every gate g_i and encode locally the relation between the inputs and the output of the gate. The formula ψ_G encoding these constraints is a CNF formula encoding

- $t_i \leftrightarrow x_i$ for every $1 \leqslant i \leqslant n$,
- $t_i \leftrightarrow y_{i-n}$ for every $n+1 \leqslant i \leqslant n+k$, and
- $t_i \leftrightarrow f_i(t_{l_i}, t_{r_i})$ for every $n+k+1 \leqslant i \leqslant m$.

We then have Φ is equisatisfiable to

$$\Psi_1 \coloneqq \forall \bar{x}, \exists y_1(\bar{z}_1), \ldots, \exists y_k(\bar{z}_k), \exists \bar{t}(\bar{x}).\, \psi_G \wedge t_m\,.$$

To transform it to DNF form, as noted in [9], Φ is equisatisfiable to

$$\Psi_2 \coloneqq \forall \bar{x}, \forall \bar{t}, \exists y_1(\bar{z}_1), \ldots, \exists y_k(\bar{z}_k).\, \psi_G \rightarrow t_m\,.$$

Note that the matrix of the formula is in DNF form. In the context of QBF, it can be thought of as applying the Tseitin transformation on $\neg\varphi$ and then negating the resulting existential formula [21]. We refer to this as the *DNF version* of Tseitin transformation.

B Omitted Proofs

We fill in the omitted proofs in the main text in this section.

B.1 Complexity of $\mathsf{sat}(k\text{-DQBF}^{\mathsf{d}}_{\mathsf{dnf}})$

Proof of Theorem 4. Since we are considering subclasses of k-DQBF, it suffices to show the hardness part.

First, observe that the DNF version of the Tseitin transformation (see Section A.1) preserves both the number of existential variables and the dependency structure. Therefore, we have that $\mathsf{sat}(k\text{-DQBF}^{\alpha}_{\mathsf{dnf}})$ is as hard as $\mathsf{sat}(k\text{-DQBF}^{\alpha})$ for every combination of α and k. In addition, observe that the formula constructed to show the PSPACE- and NEXP-hardness of $\mathsf{sat}(2\text{-DQBF})$ and $\mathsf{sat}(3\text{-DQBF})$ in [14, Theorems 4 and 5] are in fact $2\text{-DQBF}^{\mathsf{d}}$ and $3\text{-DQBF}^{\mathsf{d}}$, respectively. Thus, we have $\mathsf{sat}(k\text{-DQBF}^{\mathsf{d}}_{\mathsf{dnf}})$ is coNP-, PSPACE-, and NEXP-complete for $k = 1$, $k = 2$, and $k \geqslant 3$, respectively. ◀

B.2 Hardness of $\mathsf{sat}(3\text{-}\mathsf{DQBF}^{\mathsf{d}}_{\mathsf{cnf}})$

Hardness Proof of Theorem 14. Let $\varphi = \bigwedge_{j\in[m]}(\ell_{j,1} \vee \ell_{j,2} \vee \ell_{j,3})$ be a 3-CNF formula over the variables $\bar{x} = (x_i)_{i\in[n]}$. We again write $\mathrm{ind}(\ell) := i$ if $\mathrm{var}(\ell) = x_i$.

We construct the following 3-DQBF$^{\mathsf{d}}_{\mathsf{cnf}}$ formula

$$\Psi := \forall\bar{u}_1, \forall\bar{u}_2, \forall\bar{u}_3, \exists y_1(\bar{u}_1), \exists y_2(\bar{u}_2), \exists y_3(\bar{u}_3).\, \psi\,,$$

where $\bar{u}_1$, $\bar{u}_2$, and $\bar{u}_3$ are of length $O(\log n)$ for representing the variables in $\bar{x}$ and ψ is a CNF formula that states the following.

- $(\bar{u}_1 = i) \wedge (\bar{u}_2 = i) \to (y_1 \leftrightarrow y_2)$ for every $i \in [n]$,
- $(\bar{u}_1 = i) \wedge (\bar{u}_3 = i) \to (y_1 \leftrightarrow y_3)$ for every $i \in [n]$, and
- $(\bar{u}_1 = \mathrm{ind}(\ell_{j,1})) \wedge (\bar{u}_2 = \mathrm{ind}(\ell_{j,2})) \wedge (\bar{u}_3 = \mathrm{ind}(\ell_{j,3})) \to (y_1 \leftrightarrow \mathrm{sgn}(\ell_{j,1})) \vee (y_2 \leftrightarrow \mathrm{sgn}(\ell_{j,2})) \vee (y_3 \leftrightarrow \mathrm{sgn}(\ell_{j,3}))$ for every $j \in [m]$.

The first two items state that the Skolem functions for y_1, y_2, and y_3 must be the same. The third item implies that $\bar{a}^{\bar{x}}$ is a satisfying assignment of φ if and only if the function $f_{\bar{a}}$ is a Skolem function for Ψ by encoding $\bar{a}^{\bar{x}}$ as a function $f_{\bar{a}}\colon [n] \to \{\top, \bot\}$, where $f_{\bar{a}}(i) = \bar{a}^{\bar{x}}(x_i)$. ◀

B.3 Proof of Lemma 18

Recall that we fix a k-DQBF$^{\mathsf{ds}}_{\mathsf{cnf}}$ formula:

$$\Phi := \forall\bar{x}, \exists y_1(D_1), \ldots, \exists y_k(D_k).\ \bigwedge_{j\in[m]} C_j. \tag{4}$$

Let $\bar{y} = (y_1, \ldots, y_k)$. We recall some of the notation used in the main text.

- y_t is a maximal variable where $D_t \neq \bar{x}$.
- $\{i_1, \ldots, i_p\} = \{i \mid D_i \subseteq D_t\}$ and $\{i'_1, \ldots, i'_q\} = \{i' \mid D_{i'} \cap D_t = \emptyset\}$.
- For each $j \in [m]$:

$$C_j^{+t} := \{\ell \mid \mathrm{dep}(\ell) \subseteq D_t\}$$
$$C_j^{-t} := C_j \setminus C_j^{+t}$$

- For a function $\xi : [m] \to \{+t, -t\}$, we define two formulas:

$$\Phi_{+t,\xi} := \forall\bar{x}, \exists y_{i_1}(D_{i_1}), \ldots, \exists y_{i_p}(D_{i_p}).\ \bigwedge_{j \text{ s.t. } \xi(j)=+t} C_j^{+t}$$
$$\Phi_{-t,\xi} := \forall\bar{x}, \exists y_{i'_1}(D_{i'_1}), \ldots, \exists y_{i'_q}(D_{i'_q}).\ \bigwedge_{j \text{ s.t. } \xi(j)=-t} C_j^{-t}$$

Finally, we recall Lemma 18.

▶ **Lemma 18.** *Φ is satisfiable if and only if there is a function $\xi : [m] \to \{+t, -t\}$ such that $\Phi_{+t,\xi}$ and $\Phi_{-t,\xi}$ are both satisfiable.*

To prove Lemma 18, we will need additional notation and terminology. Let $S_t := \{y_i \mid D_i \subseteq D_t\}$. Note that $y_t \in S_t$. To ease notation, we write $D_t^c := \bar{x} \setminus D_t$ and $S_t^c := \bar{y} \setminus S_t$. That is, D_t^c is the complement of D_t w.r.t. $\bar{x}$ and S_t^c is the complement of S_t w.r.t. $\bar{y}$. In the following, we will drop the subscript t in D_t, S_t, D_t^c, S_t^c and simply write D, S, D^c, S^c.

For an assignment $(\bar{a}^D, \bar{b}^S)$, we define the clause $\mathrm{cl}(\bar{a}^D, \bar{b}^S)$:

$$\mathrm{cl}(\bar{a}^D, \bar{b}^S) \;:=\; \bigvee_{i \in S} Y_{i,\bar{a}_i} \oplus b_i, \qquad \text{where each } \bar{a}_i = \bar{a}^D(D_i) \text{ and } b_i = \bar{b}^S(y_i).$$

Similarly, for an assignment $(\bar{a}^{D^c}, \bar{b}^{S^c})$, we define the clause $\mathrm{cl}(\bar{a}^{D^c}, \bar{b}^{S(y)^c})$:

$$\mathrm{cl}(\bar{a}^{D^c}, \bar{b}^{S^c}) \;:=\; \bigvee_{y_i \in S^c} Y_{i,\bar{a}_i} \oplus b_i, \qquad \text{where each } \bar{a}_i = \bar{a}^{D^c}(D_i) \text{ and } b_i = \bar{b}^{S^c}(y_i).$$

We now generalise Definition 6 to the laminar case.

▶ **Definition 24.** *Let Φ be as in Equation (4). For every $j \in [m]$, we define the sets:*

$$\begin{aligned}
\mathcal{L}^*_{+t,j}(\Phi) &:= \{\mathrm{cl}(\bar{a}^D, \bar{b}^S) \mid (\bar{a}^D, \bar{b}^S) \simeq \neg C_j^{+t}\} \\
\mathcal{L}^*_{-t,j}(\Phi) &:= \{\mathrm{cl}(\bar{a}^{D^c}, \bar{b}^{S^c}) \mid (\bar{a}^{D^c}, \bar{b}^{S^c}) \simeq \neg C_j^{-t}\} \\
\mathfrak{C}_j(\Phi) &:= \{\mathcal{C}_{\bar{a},\bar{b}} \mid (\bar{a}^{\bar{x}}, \bar{b}^{\bar{y}}) \simeq \neg C_j\}
\end{aligned}$$

The following lemma is a generalisation of Lemma 7 to the laminar case.

▶ **Lemma 25.** *Let Φ be as in Equation (4). Then, for every $j \in [m]$, $\mathfrak{C}_j = \mathcal{L}^*_{+t,j}(\Phi) \times \mathcal{L}^*_{-t,j}(\Phi)$.*

Proof. The proof is a straightforward generalisation of Lemma 7. For the sake of completeness, we present it here.

We fix an arbitrary $j \in [m]$. We first prove the "$\subseteq$" direction. Let $\mathcal{C}_{\bar{a},\bar{b}}$ be a clause in $\mathfrak{C}_j$. That is, $(\bar{a}^{\bar{x}}, \bar{b}^{\bar{y}})$ is an assignment that falsifies C_j. By definition, $\mathcal{C}_{\bar{a},\bar{b}} = \bigvee_{i \in [k]} Y_{i,\bar{a}_i} \oplus b_i$. Since $(\bar{a}, \bar{b})$ falsifies C_j, it is consistent with the cube $\neg C_j$.

Let $\bar{a}_t = \bar{a}^{\bar{x}}(D)$ and $\bar{b}_t = \bar{b}^{\bar{y}}(S)$. Let $\bar{a}_0 = \bar{a}^{\bar{x}}(D^c)$ and $\bar{b}_0 = \bar{b}^{\bar{y}}(S^c)$. Both are consistent with the cubes $\neg C_j^{+t}$ and $\neg C_j^{-t}$, respectively. By definition, the clause $\mathrm{cl}(\bar{a}_t^D, \bar{b}_t^S)$ is in $\mathcal{L}^*_{+t,j}(\Phi)$ and the clause $\mathrm{cl}(\bar{a}_t^{D^c}, \bar{b}_t^{S^c})$ is in $\mathcal{L}^*_{-t,j}(\Phi)$. The inclusion follows since

$$\mathcal{C}_{\bar{a},\bar{b}} = \mathrm{cl}(\bar{a}_t^D, \bar{b}_t^S) \;\vee\; \mathrm{cl}(\bar{a}_t^{D^c}, \bar{b}_t^{S^c})$$

Next, we prove the "$\supseteq$" direction. Let $C \in \mathcal{L}^*_{+t,j}(\Phi) \times \mathcal{L}^*_{-t,j}(\Phi)$. Let $C := B_1 \vee B_2$, where $B_1 \in \mathcal{L}^*_{+t,j}(\Phi)$ and $B_2 \in \mathcal{L}^*_{-t,j}(\Phi)$. By definition,

- there is assignment $(\bar{a}_1^D, \bar{b}_1^S)$ such that B_1 is the clause $\mathrm{cl}(\bar{a}_1^D, \bar{b}_1^S)$,
- there is assignment $(\bar{a}_2^{D^c}, \bar{b}_2^{S^c})$ such that B_2 is the clause $\mathrm{cl}(\bar{a}_2^{D^c}, \bar{b}_2^{S^c})$.

Since the dependency sets of the variables in S are disjoint with the dependency sets of the variables in S^c , the assignments $(\bar{a}_1^D, \bar{b}_1^S)$ and $(\bar{a}_2^{D^c}, \bar{b}_2^{S^c})$ are consistent. Let $(\bar{a}^{\bar{x}}, \bar{b}^{\bar{y}})$ be their union, which is consistent with $\neg C_j^{+t} \wedge \neg C_j^{-t}$. Therefore, $(\bar{a}^{\bar{x}}, \bar{b}^{\bar{y}})$ is a falsifying assignment of C_j. By definition, the clause $\mathcal{C}_{\bar{a},\bar{b}} = B_1 \vee B_2$ and it is in $\mathfrak{C}_j$. ◀

Now, Lemma 18 follows from the following lemma, which is the generalisation of Lemma 11.

▶ **Lemma 26.** *Let Φ be as in Equation (4) and let $\bar{Y}$ be the vector of variables in* $\exp(\Phi)$*. For every assignment $\bar{a}^{\bar{Y}}$, $\bar{a}^{\bar{Y}}$ satisfies* $\exp(\Phi)$ *if and only if it satisfies* $\exp(\Phi_{+t,\xi})$ *and* $\exp(\Phi_{-t,\xi})$ *for some function $\xi : [m] \to \{+t, -t\}$.*

Proof. The proof is similar to Lemma 26. For completeness, we reprove it here. We observe that:

$$\begin{aligned}
\exp(\Phi) &= \bigwedge_{j \in [m]} \bigwedge_{C \in \mathfrak{C}_j} C \\
&= \bigwedge_{j \in [m]} \bigwedge_{(C_1, C_2) \in \mathcal{L}^*_{+t,j}(\Phi) \times \mathcal{L}^*_{-t,j}(\Phi)} C_1 \vee C_2
\end{aligned}$$

The second equality comes from Lemma 25. Thus, $\exp(\Phi)$ is satisfiable iff there is a function $\xi : [m] \to \{+t, -t\}$ such that

$$\left(\bigwedge_{j:\xi(j)=+t} \bigwedge_{C_1 \in \mathcal{L}^*_{+t,j}(\Phi)} C_1 \right) \wedge \left(\bigwedge_{j:\xi(j)=-t} \bigwedge_{C_2 \in \mathcal{L}^*_{-t,j}(\Phi)} C_2 \right)$$

is satisfiable. The first part of the conjunction is precisely $\exp(\Phi_{+t,\xi})$ and the second part is precisely $\exp(\Phi_{-t,\xi})$. ◀