Fine-Grained Complexity Analysis of Dependency Quantified Boolean Formulas

CHE CHENG, National Taiwan University, Taiwan
LONG-HIN FUNG, National Taiwan University, Taiwan
JIE-HONG ROLAND JIANG, National Taiwan University, Taiwan
FRIEDRICH SLIVOVSKY, University of Liverpool, UK
TONY TAN, University of Liverpool, UK

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.

JAIR Associate Editor: Insert JAIR AE Name

JAIR Reference Format:
Che Cheng, Long-Hin Fung, Jie-Hong Roland Jiang, Friedrich Slivovsky, and Tony Tan. 2025. Fine-Grained Complexity Analysis of Dependency Quantified Boolean Formulas. *Journal of Artificial Intelligence Research* 4, Article 6 (August 2025), 20 pages. DOI: 10.1613/jair.1.xxxxx

1 Introduction

Propositional satisfiability (SAT) solving has made significant progress over the past 30 years (Biere, Fleury, et al. 2023; Fichte et al. 2023). 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 (Biere, Heule, et al. 2009). 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) (Peterson et al. 2001). 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

Authors' Contact Information: Che Cheng, ORCID: 0009-0009-9126-3239, Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan, f11943097@ntu.edu.tw; Long-Hin Fung, ORCID: 0009-0004-0972-9188, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, r12922017@csie.ntu.edu.tw; Jie-Hong Roland Jiang, ORCID: 0000-0002-2279-4732, Department of Electrical Engineering, Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan, jhjiang@ntu.edu.tw; Friedrich Slivovsky, ORCID: 0000-0003-1784-2346, School of Computer Science and Informatics, University of Liverpool, Liverpool, UK, F.Slivovsky@liverpool.ac.uk; Tony Tan, ORCID: 0009-0005-8341-2004, School of Computer Science and Informatics, University of Liverpool, Liverpool, UK, Tony.Tan@liverpool.ac.uk.

DOI: 10.1613/jair.1.xxxxx

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 (Bubeck 2010; Bubeck and Büning 2006, 2010; Ganian et al. 2020; Scholl et al. 2019). 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 (Scholl et al. 2019).

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, henceforth, denoted by k-DQBF (Fung and Tan 2023). For DNF matrices, this restriction has no effect, since the proofs by Fung and Tan (2023) 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 Scholl et al. (2019) 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, Fung, Cheng, et al. (2024) 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 $\mathrm{var}(v) = \mathrm{var}(\neg v) = v$ for the variable of a literal and $\mathrm{sgn}(v) = \top$ and $\mathrm{sgn}(\neg v) = \bot$ to denote its sign. We also write $v \oplus \bot$ and $v \oplus \top$ to denote the literals v and $\neg v$, respectively.

A clause is a disjunction of literals, and a cube is a conjunction of literals. For a clause/cube C, we write $\mathrm{vars}(C) = \{\mathrm{var}(\ell) \mid \ell \in C\}$ for the set of variables appearing in C. 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 $Q \rightarrow C$, where Q is a cube and C is a clause; and a DNF formula in the form of $\varphi \rightarrow \psi$, where φ is in CNF and ψ is in DNF.

Table 1. Summary of the complexity results.

k	k-DQBF$^{\mathrm{d}}_{\mathrm{cnf}}$	k-DQBF$^{\mathrm{de}}_{\mathrm{cnf}}$	k-DQBF$^{\mathrm{dec}}_{\mathrm{cnf}}$, k-DQBF$^{\mathrm{ds}}_{\mathrm{cnf}}$	k-DQBF$_{\mathrm{cnf}}$	k-DQBF$^{\mathrm{d}}_{\mathrm{dnf}}$
1	-	-	-	L (Theorem 6.1)	coNP-c (Theorem 3.1)
2	NL-c (Theorem 5.3)	NL-c (Corollary 5.12)	NL-c (Corollary 5.12)	coNP-c (Theorem 6.2)	PSPACE-c (Theorem 3.1)
3	NP-c (Theorem 5.9)	NP-c (Corollary 5.12)	NP-c (Corollary 5.12)	Π^{P}_2-h (Theorem 6.3)	NEXP-c (Theorem 3.1)
4				Π^{P}_4-h (Theorem 6.5)	
5				PSPACE-h (Theorem 6.6)	
6+				NEXP-c (Theorem 6.6)	
Non-const.		Σ^{P}_3-c (Scholl et al. 2019)	NEXP-c (Scholl et al. 2019)		

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

We say that two sets of clauses A and B are *variable-disjoint* if for any clause $C_1 \in A$ and $C_2 \in B$, $\mathrm{vars}(C_1) \cap \mathrm{vars}(C_2) = \emptyset$. 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 $\varphi[\bar{a}^{\bar{v}}]$ 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.

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.

[1]To avoid clutter, we always assume a vector of variables $\bar{v} = (v_1, \ldots, v_n)$ does 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.

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

For the dependency set annotation α, we define:

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

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

DQBF$^{\mathbf{dec}}$ 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$^{\mathbf{ds}}$ 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 DQBF$^{\mathrm{ds}}$ formula form a *laminar set family*. The classification of different dependency structures is inspired by Scholl et al. (2019), but we specify the condition that the formula is in CNF explicitly in our notation. That is, DQBF$^{\mathrm{de}}$ and DQBF$^{\mathrm{dec}}$ defined by Scholl et al. (2019) correspond to DQBF$^{\mathrm{de}}_{\mathrm{cnf}}$ and DQBF$^{\mathrm{dec}}_{\mathrm{cnf}}$ in our notation, respectively.

Note that DQBF$^{\mathrm{d}}$ $\subseteq$ DQBF$^{\mathrm{de}}$ $\subseteq$ DQBF$^{\mathrm{dec}}$ $\subseteq$ DQBF$^{\mathrm{ds}}$. 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-DQBF$_{\mathrm{dnf}}$ denotes the class of DQBF with 3 existential variables, arbitrary dependency structure, and matrix in DNF, while DQBF$^{\mathrm{d}}$ denotes the class of DQBF with the dependency structure specified by d and an arbitrary Boolean formula as the matrix. We denote by $\mathrm{sat}(k\text{-DQBF}^{\alpha}_{\beta})$ the satisfiability problem for the class k-DQBF$^{\alpha}_{\beta}$.

Remark 2.1. For every $\alpha \in \{\mathrm{d}, \mathrm{de}, \mathrm{dec}, \mathrm{ds}\}$ and $\beta \in \{\mathrm{cnf}, \mathrm{dnf}\}$, checking whether a DQBF formula Φ is in the class 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, and each index requires only logarithmic space.

2.2 Tseitin Transformation

Tseitin transformation is a standard technique to turn an arbitrary Boolean satisfiability problem into an equi-satisfiable one in 3-CNF form (Tseitin 1968). It can be directly lifted to QBF and DQBF by allowing the Tseitin variables to depend on every universal variable. We recall the DQBF version here.

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 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 the $\mathrm{DQBF}_{\mathrm{cnf}}$

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

with matrix in 3-CNF.

To transform it to DNF form, as noted in (Chen et al. 2022), Φ is equisatisfiable to

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

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 (Zhang 2006). We refer to this as the *DNF version* of Tseitin transformation.

2.3 Manipulation of $\mathrm{DQBF}_{\mathrm{cnf}}$

We recall two operations for manipulating $\mathrm{DQBF}_{\mathrm{cnf}}$ formulas, namely *universal reduction* (Balabanov, Chiang, et al. 2014; Fröhlich et al. 2014) and *resolution-based variable elimination* (Wimmer et al. 2015).

Lemma 2.2 (Universal reduction (Balabanov, Chiang, et al. 2014; Fröhlich et al. 2014)). *Let* $\Phi = \forall \bar{x}, \exists y_1(D_1), \ldots, \exists y_k(D_k).\, \varphi$ *be a* $\mathrm{DQBF}_{\mathrm{cnf}}$ *formula,* $C \in \varphi$ *be a clause,* $\ell \in C$ *be a universal literal, and let* $C' := C \setminus \{\ell\}$*. If* $\ell \notin \mathrm{dep}(C')$*, then* Φ *is equisatisfiable to*

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

Using universal reduction, we assume that $\bigcup_{i\in[k]} D_i = \bar{x}$ for every $\mathrm{DQBF}_{\mathrm{cnf}}$ 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 2.3 (Variable elimination by resolution (Wimmer et al. 2015)). *Let* $\Phi = \forall \bar{x}, \exists y_1(D_1), \ldots, \exists y_k(D_k).\, \varphi$ *be a* $\mathrm{DQBF}_{\mathrm{cnf}}$ *formula. We partition* φ *into three sets:*

- $\varphi^{y_1} := \{C \in \varphi \mid y_1 \in C\}$,
- $\varphi^{\neg y_1} := \{C \in \varphi \mid \neg y_1 \in C\}$, *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*

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

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 $|\varphi|^2$.

2.4 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 $Y_{i,\bar{a}_i}$ (Balabanov and Jiang 2015; Bubeck 2010; Fröhlich et al. 2014). Formally,

$$\exp(\Phi) \coloneqq \bigwedge_{(\bar{a},\bar{b}):\varphi[\bar{a},\bar{b}]=\bot} C_{\bar{a},\bar{b}},$$

where $C_{\bar{a},\bar{b}} \coloneqq \bigvee_{i\in[k]} Y_{i,\bar{a}_i} \oplus b_i$. 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 $\mathrm{sat}(k\text{-}\mathrm{DQBF}^{\mathrm{d}}_{\mathrm{dnf}})$

Having defined various subclasses of DQBF, we will 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 (Chen et al. 2022, Proposition 1) and the results by Fung and Tan (2023), we can show that restricting to DNF matrix and pairwise-disjoint dependency sets does not affect the complexity of $\mathrm{sat}(k\text{-DQBF})$.

Theorem 3.1. *$\mathrm{sat}(k\text{-}\mathrm{DQBF}^{\mathrm{d}}_{\mathrm{dnf}})$ is* coNP-, PSPACE-, *and* NEXP-*complete when $k = 1$, $k = 2$, and $k \geqslant 3$, respectively.*

Proof. 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 2.2) preserves both the number of existential variables and the dependency structure. Therefore, we have that $\mathrm{sat}(k\text{-}\mathrm{DQBF}^{\alpha}_{\mathrm{dnf}})$ is as hard as $\mathrm{sat}(k\text{-}\mathrm{DQBF}^{\alpha})$ for every combination of $\alpha \in \{\mathrm{d}, \mathrm{de}, \mathrm{dec}, \mathrm{ds}\}$ and $k \geqslant 1$. In addition, observe that the formula constructed to show the PSPACE- and NEXP-hardness of $\mathrm{sat}(2\text{-DQBF})$ and $\mathrm{sat}(3\text{-DQBF})$ in (Fung and Tan 2023, Theorems 4 and 5) are in fact $2\text{-}\mathrm{DQBF}^{\mathrm{d}}$ and $3\text{-}\mathrm{DQBF}^{\mathrm{d}}$, respectively. Thus, we have $\mathrm{sat}(k\text{-}\mathrm{DQBF}^{\mathrm{d}}_{\mathrm{dnf}})$ is coNP-, PSPACE-, and NEXP-complete for $k = 1$, $k = 2$, and $k \geqslant 3$, respectively. □

Since $k\text{-}\mathrm{DQBF}^{\mathrm{d}}_{\mathrm{dnf}} \subseteq k\text{-}\mathrm{DQBF}^{\mathrm{de}}_{\mathrm{dnf}} \subseteq k\text{-}\mathrm{DQBF}^{\mathrm{dec}}_{\mathrm{dnf}} \subseteq k\text{-}\mathrm{DQBF}^{\mathrm{ds}}_{\mathrm{dnf}} \subseteq k\text{-}\mathrm{DQBF}_{\mathrm{dnf}} \subseteq k\text{-DQBF}$, we have the following corollary:

Corollary 3.2. *$\mathrm{sat}(k\text{-}\mathrm{DQBF}^{\alpha}_{\mathrm{dnf}})$ and $\mathrm{sat}(k\text{-}\mathrm{DQBF}_{\mathrm{dnf}})$ is* coNP-, PSPACE-, *and* NEXP-*complete when $k = 1$, $k = 2$, and $k \geqslant 3$, respectively, for every $\alpha \in \{\mathrm{de}, \mathrm{dec}, \mathrm{ds}\}$.*

4 A Useful Lemma

In this section, we prove a lemma that will be useful for proving hardness results for several subclasses of $\mathrm{DQBF}_{\mathrm{cnf}}$.

Lemma 4.1. *Let $l \geqslant 0$ be some constant, and $\Phi \coloneqq \forall\bar{z}, \exists\bar{x}(D), \exists y_1(D_1), \ldots, \exists y_k(D_k).\ \bigwedge_{j\in[m]}(C_j^{\bar{x}} \vee C_j^{-\bar{x}})$ be a $(n+k)$-$\mathrm{DQBF}_{\mathrm{cnf}}$, where*

- *every variable in $\bar{x} = (x_1, \ldots, x_n)$ has the dependency set D,*
- *$\mathrm{vars}(C_j^{-\bar{x}}) \cap \bar{x} = \emptyset$,*
- *$\mathrm{vars}(C_j^{\bar{x}}) \subseteq \bar{x}$, and*
- *$C_j^{\bar{x}} = \bigvee_{s\in[n_j]} \ell_{j,s}$, with $n_j \leqslant l$.*

Then, we can construct in logspace an equisatisfiable $(k+l)$-$\mathrm{DQBF}_{\mathrm{cnf}}$ formula.

Proof. We construct

$$\Phi' = \forall\bar{z}, \forall\bar{u}_1, \ldots, \forall\bar{u}_l, \exists y_1(D_1), \ldots, \exists y_k(D_k), \exists t_1(D \cup \bar{u}_1), \ldots, \exists t_l(D \cup \bar{u}_l).\ \varphi',$$

where each $\bar{u}_i$ is of length $\lceil \log_2 n \rceil + 1$, and φ' consists of clauses encoding

- $((\bar{u}_1 = i) \wedge (\bar{u}_{s+1} = i)) \to (t_1 \leftrightarrow t_{s+1})$ for $i \in [n]$ and $s \in [l-1]$, and
- $(\bigwedge_{s \in [n_j]} (\bar{u}_s = \mathrm{ind}(\ell_{j,s})) \to (C_j^{-\bar{x}} \vee \bigvee_{s \in [n_j]} (t_s \leftrightarrow \mathrm{sgn}(\ell_{j,s})))$ for $j \in [m]$,

where $\mathrm{ind}(\ell)$ denotes the index i where $z_i = \mathrm{var}(\ell)$ for a literal ℓ.

The fact that Φ' is a $(k+l)$-DQBF$_{\mathrm{cnf}}$ formula is easy to verify. Note that each constraint in φ' is of the form $Q \to C \vee \psi$, where Q is a DNF, C is a CNF, and ψ involves a constant number of variables. Thus, it can be transformed into the conjunction of a constant number of clauses.

We prove the equisatisfiability by transforming a model of Φ to a model of Φ' and vice versa. Let $f_1, \dots, f_n$, $g_1, \dots, g_k$ be a model of Φ, where each f_i is the Skolem function for x_i and g_i is the Skolem function for y_i. We construct the Skolem function

$$h_s = \bigwedge_{i \in [n]} ((\bar{u}_s = i) \to f_i)$$

for t_s for each $s \in [l]$,

We now show that $g_1, \dots, g_k, h_1, \dots, h_l$ is a model for Φ'. First note that h_s depends only on variables in $D \cup \bar{u}_s$, thus it is a valid Skolem function. Consider an arbitrary assignment $(\bar{a}, \bar{b})$ over $\bar{z}$ and $\bar{u}_1, \dots, \bar{u}_l$. For any $s \in [l-1]$, if $\bar{u}_1 = i$ and $\bar{u}_{s+1} = i$ both hold for some $i \in [n]$, we have $h_1 = h_{s+1} = f_i$ by construction, so $t_1 \leftrightarrow t_{s+1}$ must evaluate to true under $(\bar{a}, \bar{b})$. For any $j \in [m]$, if $(\bigwedge_{s \in [n_j]} (\bar{u}_s = \mathrm{ind}(\ell_{j,s}))$ holds, we consider the corresponding clause C_j in Φ. Since $f_1, \dots, f_n, g_1, \dots, g_k$ is a model of Φ, either $C_j^{-\bar{x}}$ is satisfied by $g_1, \dots, g_k$ and $\bar{a}$ or at least one of $\ell_{j_1}, \dots, \ell_{j,n_j}$ is satisfied by some f_i. In the former case, $C_j^{-\bar{x}}$ will satisfy the corresponding constraint in Φ'. In the latter case, we have $t_s = f_{\mathrm{ind}(\ell_{j,s})}$, and thus the disjunction $\bigvee_{s \in [n_j]} (y_s \leftrightarrow \mathrm{sgn}(\ell_{j,s}))$ must be satisfied. We conclude that $g_1, \dots, g_k, h_1, \dots, h_l$ is a model for Φ'.

We now prove the other direction. To ease notation, we write $\bar{a}_i^{\bar{u}_s}$ to denote the assignment satisfying $\bar{u}_s = i$ for any $i \in [n]$ and $s \in [l]$. Let $g_1, \dots, g_k, h_1, \dots, h_l$ be a model for Φ'. We construct the Skolem function

$$f_i = h_1[\bar{a}_i^{\bar{u}_1}]$$

for x_i for each $i \in [n]$. We now show that $f_1, \dots, f_n, g_1, \dots, g_k$ is a model for Φ. First, note that f_i depends only on variables in D, so it is a valid Skolem function for x_i. Next, observe that the constraint $((\bar{u}_1 = i) \wedge (\bar{u}_{s+1} = i)) \to (t_1 \leftrightarrow t_{s+1})$ guarantees that $h_1[\bar{a}^{\bar{u}_1}] = h_{s+1}[\bar{a}^{\bar{u}_{s+1}}]$ for any $s \in [l-1]$. We can thus substitute all occurrences of h_{s+1} with h_1. Finally, consider an assignment $\bar{a}^{\bar{z}}$ and a clause C_j. Note that $g_1, \dots, g_k, h_1, \dots, h_l$ satisfies the constraint

$$\left(\bigwedge_{s \in [n_j]} (\bar{u}_s = \mathrm{ind}(\ell_{j,s}) \right) \to \left(C_j^{-\bar{x}} \vee \bigvee_{s \in [n_j]} (t_s \leftrightarrow \mathrm{sgn}(\ell_{j,s})) \right)$$

over all assignments on the universal variables. In particular, by instantiating each $\bar{u}_s$ with $\mathrm{ind}(\ell_{j,s})$, we have

$$C_j^{-\bar{x}} \vee \bigvee_{s \in [n_j]} (t_s^{\mathrm{ind}(\ell_{j,s})} \leftrightarrow \mathrm{sgn}(\ell_{j,s}))$$

must always be satisfied. That is, if $C_j^{-\bar{x}}$ is not satisfied by $g_1, \dots, g_k$, then at least one of $h_1[\bar{a}^{\bar{u}_1}_{\mathrm{ind}(\ell_{j,1})}], \dots,$ $h_1[\bar{a}^{\bar{u}_1}_{\mathrm{ind}(\ell_{j,n_j})}]$ must be assigned to $\mathrm{sgn}(\ell_{j,s})$. It follows by construction of the f_i's that $f_1, \dots, f_n, g_1, \dots, g_k$ must satisfies C_j. □

Intuitively, Lemma 4.1 says that for DQBF$_{\mathrm{cnf}}$ formulas, existential variables sharing the same dependency set can be "compressed", as long as each clause contains only a small number of such variables. In addition, we make the following remark.

Remark 4.2. If $k = 0$ and $D = \emptyset$, the constructed formula becomes a l-DQBF$^{\mathrm{d}}_{\mathrm{cnf}}$ formula.

We will use Lemma 4.1 and Remark 4.2 to reduce different SAT and QBF formulas to corresponding DQBF subclasses in Sections 5 and 6 to obtain the desired hardness results.

5 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 5.1. Then, in Sections 5.2 and 5.3 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 5.4.

5.1 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 $C_{\bar{a},\bar{b}}$ where $(\bar{a}, \bar{b})$ falsifies $\neg C_j$. We now formally define these sets.

Definition 5.1. 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 $\mathfrak{C}_j(\Phi)$:

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

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.

We now state the property formally.

Lemma 5.2. *Let Φ be as in Eq. (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 $C_{\bar{a},\bar{b}}$ 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, $C_{\bar{a},\bar{b}} = \bigvee_{i \in [k]} Y_{i,\bar{a}_i} \oplus b_i$. Since $(\bar{a}, \bar{b})$ falsifies C_j, it is consistent with the cube $\neg C_j$. Hence,

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

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 $\neg C^1_j, \ldots, \neg C^k_j$. Therefore, $(\bar{a}, \bar{b})$ is a falsifying assignment of C_j. By definition, the clause $C_{\bar{a},\bar{b}} = \bigvee_{i\in[k]} Y_{i,\bar{a}_i} \oplus b_i$ is in $\mathfrak{C}_j$. □

5.2 2-DQBF$^{\mathrm{d}}_{\mathrm{cnf}}$

In this section we will show that sat(2-DQBF$^{\mathrm{d}}_{\mathrm{cnf}}$) is NL-complete.

Theorem 5.3. *sat(2-DQBF$^{\mathrm{d}}_{\mathrm{cnf}}$) 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) (Aspvall et al. 1979). 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 5.4 (Disimplex (Figueroa and Llano 2010)). 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 5.3. For the rest of this subsection, we fix a 2-DQBF$^{\mathrm{d}}_{\mathrm{cnf}}$ 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 5.1. 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 5.5. *Let $G = (\mathcal{V}, \mathcal{E})$ be the implication graph of $\exp(\Phi)$. The set of edges $\mathcal{E}$ can be represented as*

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

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

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

[3]Note that, as stated in Section 2.3, we assume that $\bigcup_{i\in[k]} D_i = \bar{x}$.

By Lemma 5.2, 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 5.3. 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 5.5 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^i_j and $C^{i'}_{j'}$ 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^i_j with the clause $\hat{C}^i_j$ with the sign of y_i flipped if a literal of y_i is present, i.e.,

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

For the hardness proof, note that a 2-SAT formula is essentially a $\mathrm{DQBF_{cnf}}$ where all variables share the common dependency set $\emptyset$ and every clause contains exactly two literals. By Lemma 4.1 and Remark 4.2, it is equisatisfiable to a 2-$\mathrm{DQBF^d_{cnf}}$. □

5.3 k-$\mathbf{DQBF^d_{cnf}}$: $k \geqslant 3$ and Non-Constant k

For $k \geqslant 3$ and even arbitrary $\mathrm{DQBF^d_{cnf}}$, we show that it is NP-complete. Let Φ be as in Eq. (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 5.6. *Let Φ be as in Eq. (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 $\bigwedge_{C \in \mathfrak{C}_j} C$ 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 $C \in \mathfrak{C}_j$, by Lemma 5.2, there exists some $L \in \mathcal{L}_{i,j} \cap C$ that is satisfied by $\bar{a}$. Thus, C is satisfied by L.

[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 "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 $\bigwedge_{C\in\mathfrak{C}_j} C$. □

Remark 5.7. Recall that $\exp(\Phi) = \bigwedge_{j\in[m]} \bigwedge_{C\in\mathfrak{C}_j} C$. Thus, Lemma 5.6 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 5.6. This formulation will be useful later on.

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

Lemma 5.8. $\mathrm{sat}(\mathrm{DQBF}^{\mathrm{d}}_{\mathrm{cnf}})$ *is in* NP.

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

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

with k existential variables and m clauses.

By the reformulation of Lemma 5.6 in Remark 5.7, 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 2.3.[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 5.9. $\mathrm{sat}(k\text{-}\mathrm{DQBF}^{\mathrm{d}}_{\mathrm{cnf}})$ *for every* $k \geqslant 3$ *and* $\mathrm{sat}(\mathrm{DQBF}^{\mathrm{d}}_{\mathrm{cnf}})$ *are* NP*-complete.*

Proof. By Lemma 5.8, $\mathrm{sat}(\mathrm{DQBF}^{\mathrm{d}}_{\mathrm{cnf}})$ is in NP. Since $k\text{-}\mathrm{DQBF}^{\mathrm{d}}_{\mathrm{cnf}} \subseteq \mathrm{DQBF}^{\mathrm{d}}_{\mathrm{cnf}}$, $\mathrm{sat}(k\text{-}\mathrm{DQBF}^{\mathrm{d}}_{\mathrm{cnf}})$ is also in NP for every constant k.

For the hardness proof, note that a 3-SAT formula is essentially a $\mathrm{DQBF}_{\mathrm{cnf}}$ where all variables share the common dependency set $\emptyset$ and every clause contains exactly three literals. By Lemma 4.1 and Remark 4.2, it is equisatisfiable to a $3\text{-}\mathrm{DQBF}^{\mathrm{d}}_{\mathrm{cnf}}$. Since adding more existential variables only increases the complexity, $\mathrm{sat}(k\text{-}\mathrm{DQBF}^{\mathrm{d}}_{\mathrm{cnf}})$ for every $k \geqslant 3$ and $\mathrm{sat}(\mathrm{DQBF}^{\mathrm{d}}_{\mathrm{cnf}})$ are also NP-hard. □

5.4 k-$\mathrm{DQBF}^{\alpha}_{\mathrm{cnf}}$: Different Dependency Structure

It has been shown by Scholl et al. (2019) that $\mathrm{sat}(\mathrm{DQBF}^{\mathrm{de}}_{\mathrm{cnf}})$ is Σ^{P}_3-complete and $\mathrm{sat}(\mathrm{DQBF}^{\mathrm{dec}}_{\mathrm{cnf}})$ is NEXP-complete. Since $\mathrm{DQBF}^{\mathrm{dec}}_{\mathrm{cnf}} \subseteq \mathrm{DQBF}^{\mathrm{ds}}_{\mathrm{cnf}} \subseteq \mathrm{DQBF}$ and $\mathrm{sat}(\mathrm{DQBF})$ is also NEXP-complete, we know $\mathrm{sat}(\mathrm{DQBF}^{\mathrm{ds}}_{\mathrm{cnf}})$ is NEXP-complete. In this section, we show a surprising result that, when k is a constant, $\mathrm{sat}(k\text{-}\mathrm{DQBF}^{\alpha}_{\mathrm{cnf}})$ has the same complexity as k-SAT and $\mathrm{sat}(k\text{-}\mathrm{DQBF}^{\mathrm{d}}_{\mathrm{cnf}})$ for every $\alpha \in \{\mathrm{de}, \mathrm{dec}, \mathrm{ds}\}$. Since $k\text{-}\mathrm{DQBF}^{\mathrm{d}}_{\mathrm{cnf}} \subseteq k\text{-}\mathrm{DQBF}^{\mathrm{de}}_{\mathrm{cnf}} \subseteq k\text{-}\mathrm{DQBF}^{\mathrm{dec}}_{\mathrm{cnf}} \subseteq k\text{-}\mathrm{DQBF}^{\mathrm{ds}}_{\mathrm{cnf}}$, it suffices to show the results for $\mathrm{sat}(k\text{-}\mathrm{DQBF}^{\mathrm{ds}}_{\mathrm{cnf}})$.

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

Theorem 5.10. $\mathrm{sat}(2\text{-}\mathrm{DQBF}^{\mathrm{ds}}_{\mathrm{cnf}})$ *is* NL*-complete.*

[5] In fact, it is in L, as shown later in Theorem 6.1.

Proof. Since $2\text{-DQBF}^{\mathrm{ds}}_{\mathrm{cnf}} \supseteq 2\text{-DQBF}^{\mathrm{d}}_{\mathrm{cnf}}$, the hardness follows from Theorem 5.3. For NL membership, consider a $2\text{-DQBF}^{\mathrm{ds}}_{\mathrm{cnf}}$ formula $\Phi := \forall \bar{x}, \exists y_1(D_1), \exists y_2(D_2).\, \varphi$. First, we check whether D_1 and D_2 are disjoint using only logarithmic space. (See Remark 2.1.) If D_1 and D_2 are disjoint, we use the algorithm from Theorem 5.3 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 (Janota and Marques-Silva 2015). 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 $\bar{z}_1 := D_1$, $\bar{z}_2 := D_2 \setminus D_1$, and $\bar{z}_3 := \bar{x} \setminus D_2$. Note that Φ is equivalent to a QBF

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

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 iterate through all 4-tuples of clause indices and check whether the properties hold. □

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

Theorem 5.11. *For every constant $k \geqslant 3$, $\mathrm{sat}(k\text{-DQBF}^{\mathrm{ds}}_{\mathrm{cnf}})$ is NP-complete.*

Before we present the proof of Theorem 5.11, we note that since $k\text{-DQBF}^{\mathrm{de}}_{\mathrm{cnf}} \subseteq k\text{-DQBF}^{\mathrm{dec}}_{\mathrm{cnf}} \subseteq k\text{-DQBF}^{\mathrm{ds}}_{\mathrm{cnf}}$, we obtain the following results as a corollary of Theorems 5.3 and 5.9 to 5.11.

Corollary 5.12. *$\mathrm{sat}(k\text{-DQBF}^{\mathrm{de}}_{\mathrm{cnf}})$ and $\mathrm{sat}(k\text{-DQBF}^{\mathrm{dec}}_{\mathrm{cnf}})$ 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 5.11.

Proof of Theorem 5.11. We will consider the membership proof. The hardness follows from Theorem 5.9. We fix a $k\text{-DQBF}^{\mathrm{ds}}_{\mathrm{cnf}}$ 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 2.2, 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 6.1.

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

Case 1: $D_t = \bar{x}$. We apply Lemma 2.3 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 5.8.

Let $\{i_1, \dots, i_p\} = \{i \mid D_i \subseteq D_t\}$ and $\{i'_1, \dots, i'_q\} = \{i' \mid D_{i'} \cap D_t = \emptyset\}$. 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:

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

We have the following lemma.

Lemma 5.13. *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. □

It remains to prove Lemma 5.13. 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.

we introduce the following 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

$$\mathrm{cl}(\bar{a}^D, \bar{b}^S) := \bigvee_{i \in S} Y_{i,\bar{a}_i} \oplus b_i\,,$$

where each $\bar{a}_i = \bar{a}^D(D_i)$ and$b_i = \bar{b}^S(y_i)$. Similarly, for an assignment $(\bar{a}^{D^c}, \bar{b}^{S^c})$, we define the clause

$$\mathrm{cl}(\bar{a}^{D^c}, \bar{b}^{S^c}) := \bigvee_{y_i \in S^c} Y_{i,\bar{a}_i} \oplus b_i\,,$$

where each $\bar{a}_i = \bar{a}^{D^c}(D_i)$ and $b_i = \bar{b}^{S^c}(y_i)$.

We now generalise Definition 5.1 to the laminar case.

Definition 5.14. Let Φ be as in Eq. (3). For every $j \in [m]$, we define the sets

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

The following lemma is a generalisation of Lemma 5.2 to the laminar case.

Lemma 5.15. *Let Φ be as in Eq. (3). 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 5.2. For the sake of completeness, we present it here.

We fix an arbitrary $j \in [m]$. We first prove the "$\subseteq$" direction. Let $C_{\bar{a},\bar{b}}$ 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, $C_{\bar{a},\bar{b}} = \bigvee_{i\in[k]} Y_{i,\bar{a}_i} \oplus b_i$. Since $(\bar{a}, \bar{b})$ falsifies C_j, it is consistent with the cube $\neg C_j$.

Let $\bar{a}_t = \bar{a}^{\bar{x}}(D)$, $\bar{b}_t = \bar{b}^{\bar{y}}(S)$, $\bar{a}_0 = \bar{a}^{\bar{x}}(D^c)$, 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 $\mathrm{cl}(\bar{a}_0^{D^c}, \bar{b}_0^{S^c})$ is in $\mathcal{L}^*_{-t,j}(\Phi)$. The inclusion follows, since

$$C_{\bar{a},\bar{b}} = \mathrm{cl}(\bar{a}_t^D, \bar{b}_t^S) \ \vee\ \mathrm{cl}(\bar{a}_0^{D^c}, \bar{b}_0^{S^c}).$$

Next, we prove the "$\supseteq$" direction. Let $C \in \mathcal{L}^*_{+t,j}(\Phi) \times \mathcal{L}^*_{-t,j}(\Phi)$. We write $C = B_1 \vee B_2$, 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 from 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 $\neg C_j^{+t} \wedge \neg C_j^{-t}$. It follows that $(\bar{a}^{\bar{x}}, \bar{b}^{\bar{y}})$ is a falsifying assignment of C_j. By definition, the clause $C_{\bar{a},\bar{b}} = B_1 \vee B_2$ and it is in $\mathfrak{C}_j$. □

Now, Lemma 5.13 follows from the following lemma, which is the generalisation of Lemma 5.6.

Lemma 5.16. *Let Φ be as in Eq. (3) 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. We observe that

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

where the second equality follows from Lemma 5.15. Thus, $\exp(\Phi)$ is satisfiable if and only if 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 $\exp(\Phi_{-t,\xi})$. □

6 Complexity of sat(k-DQBF$_{\text{cnf}}$)

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

Theorem 6.1. *sat(1-DQBF$_{\text{cnf}}$) 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 5.10, 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 6.2. *sat(2-DQBF$_{\text{cnf}}$) 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$_{\text{cnf}}$ 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$^{\text{d}}_{\text{cnf}}$ formula, the satisfiability of which can be decided in polynomial time by Theorem 5.3. 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$_{\text{cnf}}$). 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$_{\text{cnf}}$ formula:

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

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$^{\text{d}}_{\text{cnf}}$ 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 Fig. 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. □

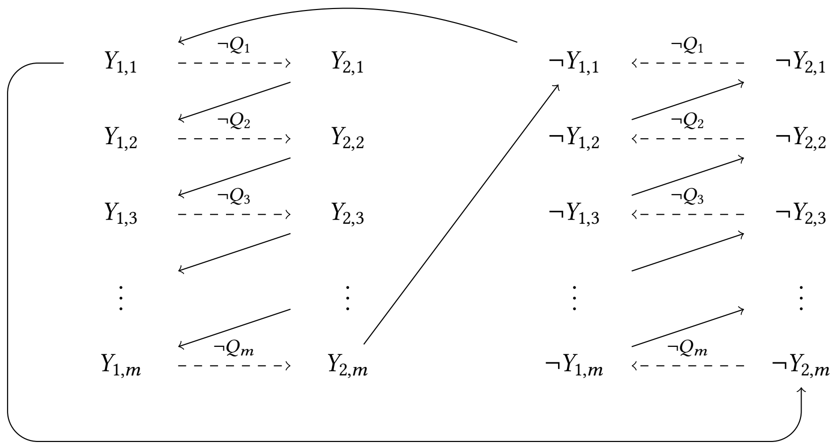

Fig. 1. The implication graph $G_{\bar{a}}$ Each dashed edge $\overset{\neg Q_i}{\dashrightarrow}$ is present if and only if $\bar{a}^{\bar{x}}$ falsifies Q_i.

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

Theorem 6.3. *$\mathrm{sat}(3\text{-DQBF}_{\mathrm{cnf}})$ is Π_2^{P}-hard.*

Proof. Note that a Π_2-QBF formula in 3-CNF is essentially a $\mathrm{DQBF}_{\mathrm{cnf}}$ where all variables share the common dependency set of all universal variables, and every clause contains exactly three literals. By Lemma 4.1, it is equisatisfiable to a $3\text{-DQBF}_{\mathrm{cnf}}$. □

We next show that $\mathrm{sat}(4\text{-DQBF}_{\mathrm{cnf}})$ is Π_4^{P}-hard. To do this, we need a stronger version of Lemma 4.1 that allows the compression of existential variables with different dependencies.

Lemma 6.4. *Let $l \geqslant 0$ be some constant, and $\Phi := \forall \bar{z}, \exists x_1(D_1), \dots, \exists x_n(D_n), \exists y_1(E_1), \dots, y_k(E_k). \bigwedge_{j\in[m]}(C_j^{\bar{x}} \vee C_j^{-\bar{x}})$ be a $(n+k)$-$\mathrm{DQBF}_{\mathrm{cnf}}$, where*

- *$\mathrm{vars}(C_j^{-\bar{x}}) \cap \bar{x} = \emptyset$,*
- *$\mathrm{vars}(C_j^{\bar{x}}) \subseteq \bar{x}$, and*
- *$C_j^{\bar{x}} = \bigvee_{s\in[n_j]} \ell_{j,s}$ with $n_j \leqslant l$.*

Moreover, let $S_1, \dots, S_p$ be subsets of $\bar{z}$ such that each D_i can be represented as the intersection of some subset of $\{S_1, \dots, S_p\}$, i.e., for each $i \in [n]$, there exists some $\mathcal{P}_i \subseteq \{S_1, \dots, S_p\}$ such that $D_i = \bigcap_{P\in\mathcal{P}_i} P$, and let $S = \bigcup_{i\in[p]} S_i$.

Then, we can construct in logspace an equisatisfiable $(p+k+l)$-$\mathrm{DQBF}_{\mathrm{cnf}}$ formula. Moreover, if $S = S_q$ for some $q \in [p]$, then we can construct an equisatisfiable $(p+k+l-1)$-$\mathrm{DQBF}_{\mathrm{cnf}}$ formula.

Proof. We construct

$$\Phi' = \forall \bar{z}, \forall \bar{u}_1, \dots, \forall \bar{u}_l, \exists y_1(E_1), \dots, \exists y_k(E_k), \exists t_1(S \cup \bar{u}_1), \dots, \exists t_l(S \cup \bar{u}_l), \exists v_1(S_1 \cup \bar{u}_1), \dots, \exists v_p(S_p \cup \bar{u}_1). \varphi',$$

where each $\bar{u}_i$ is of length $\lceil \log_2 n \rceil + 1$, and φ' consists of clauses encoding

- $((\bar{u}_1 = i) \wedge (\bar{u}_{s+1} = i)) \longrightarrow (t_1 \leftrightarrow t_{s+1})$ for $i \in [n]$ and $s \in [l-1]$,
- $(\bar{u}_1 = i) \longrightarrow (t_1 \leftrightarrow v_q)$ for $i \in [n]$ and $q \in \{q \mid S_q \in \mathcal{P}_i\}$, and
- $\bigwedge_{s\in[n_j]}(\bar{u}_s = \mathrm{ind}(\ell_{j,s}) \longrightarrow (C_j^{-\bar{x}} \vee \bigvee_{s\in[n_j]}(t_s \leftrightarrow \mathrm{sgn}(\ell_{j,s}))$ for $j \in [m]$,

where $\mathrm{ind}(\ell)$ denotes the index i such that $z_i = \mathrm{var}(\ell)$ for a literal ℓ.

The construction is mostly the same as that for Lemma 4.1, except we now have p additional variables $v_1, \ldots, v_p$, and the corresponding constraints $(\bar{u}_1 = i) \longrightarrow (t_1 \leftrightarrow v_q)$ for $i \in [n]$ and $q \in \{q \mid D_i \subseteq S_q\}$. Without these, the variables $\bar{x}$ would be allowed to depend on the entirety of S. Consider an existential variable x_i. If $S_q \in \mathcal{P}_i$, then the variable v_q ensures that the strategy of x_i must be consistent across all assignments that are consistent on S_q. For any two assignments $\bar{a}^{\bar{z}}$ and $\bar{a}'^{\bar{z}}$ that are consistent on D_i, there must be a sequence of assignments $\bar{a}_0^{\bar{z}}, \ldots, \bar{a}_p^{\bar{z}}$ such that

- $\bar{a}_0^{\bar{z}} = \bar{a}^{\bar{z}}$,
- $\bar{a}_p^{\bar{z}} = \bar{a}'^{\bar{z}}$,
- $\bar{a}_{q-1}^{\bar{z}} = \bar{a}_q^{\bar{z}}$ if $S_q \notin \mathcal{P}_i$, and
- $\bar{a}_{q-1}^{\bar{z}}$ and $\bar{a}_q^{\bar{z}}$ are consistent on S_q if $S_q \notin \mathcal{P}_i$.

It follows that the strategy of x_i must be consistent on $\bar{a}^{\bar{z}}$ and $\bar{a}'^{\bar{z}}$ through a series of constraints. Note that if $S_q = S$, then v_q will be an exact copy of t_1, and we can therefore omit it to obtain a $(p + k + l - 1)$-DQBF$_{\mathrm{cnf}}$.

The rest of the proof follows exactly the same as that for Lemma 4.1. □

A naïve extension of Lemma 4.1 will require l copies of existential variables for each dependency, which is too costly to obtain any useful results even when there are only two different dependencies. Instead, we encode different dependency sets efficiently by representing them as intersections of a (small) family of sets.

We can now show the Π_4^{P}-hardness of sat(4-DQBF$_{\mathrm{cnf}}$).

Theorem 6.5. *sat(4-DQBF$_{\mathrm{cnf}}$) is Π_4^{P}-hard.*

Proof. Consider a Π_4-QBF formula in 3-CNF

$$\Phi = \forall \bar{z}_1, \exists \bar{x}_1, \forall \bar{z}_2, \exists \bar{x}_2 . \varphi .$$

Let $S_1 = \bar{z}_1$ and $S_2 = \bar{z}_1 \cup \bar{z}_2$. We can apply Lemma 6.4 with $k = 0$, $l = 3$, $p = 2$, and S_2 is a maximum element in $\{S_1, S_2\}$. Thus, Φ is equisatisfiable to a 4-DQBF$_{\mathrm{cnf}}$. □

Finally, we provide the hardness results for sat(k-DQBF$_{\mathrm{cnf}}$) with $k = 5$ and $k \geqslant 6$.

Theorem 6.6. *sat(k-DQBF$_{\mathrm{cnf}}$) is PSPACE-hard when $k = 5$ and NEXP-complete when $k \geqslant 6$.*

Proof. Using Tseitin transformation (see Section 2.2), we can transform a k-DQBF formula Φ into an equisatisfiable $(k + n)$-DQBF$_{\mathrm{cnf}}$ formula Φ' by introducing $n = O(|\Phi|)$ fresh existential variables, such that

- each freshly introduced existential variable depends on all universal variables, and
- each clause in Φ' has exactly three literals.

By applying Lemma 4.1 on Φ', we can construct an equisatisfiable $(k + 3)$-DQBF$_{\mathrm{cnf}}$ formula Φ''.

Recall that sat(2-DQBF) and sat(3-DQBF) are PSPACE- and NEXP-complete, respectively (Fung and Tan 2023). Combining the above, we conclude that sat(5-DQBF$_{\mathrm{cnf}}$) is PSPACE-hard and sat(6-DQBF$_{\mathrm{cnf}}$) is NEXP-complete. □

7 Conclusions and Future Work

While sat(k-DQBF$^{\mathrm{d}}_{\mathrm{dnf}}$) 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$^{\mathrm{d}}_{\mathrm{cnf}}$), we show that it is in fact as easy as k-SAT—exponentially easier than sat(k-DQBF). Generalising the results by Scholl et al. (2019), we also show that sat(DQBF$^{\mathrm{d}}_{\mathrm{cnf}}$) is NP-complete and that sat(k-DQBF$^{\alpha}_{\mathrm{cnf}}$) has the same complexity as k-SAT for $\alpha \in \{\mathrm{d}, \mathrm{de}, \mathrm{dec}, \mathrm{ds}\}$. For the case of k-DQBF$_{\mathrm{cnf}}$, 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 $\text{sat}(k\text{-DQBF}_{\text{cnf}})$ 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.

Acknowledgments

This paper elaborates and strengthens the results announced in the conference paper (Cheng et al. 2025). We thank the reviewers of SAT 2025 for their feedback on the initial version.

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.

A Reproducibility Checklist for JAIR

Select the answers that apply to your research – one per item.

All articles:

(1) All claims investigated in this work are clearly stated. [yes]
(2) Clear explanations are given how the work reported substantiates the claims. [yes]
(3) Limitations or technical assumptions are stated clearly and explicitly. [yes]
(4) Conceptual outlines and/or pseudo-code descriptions of the AI methods introduced in this work are provided, and important implementation details are discussed. [yes]
(5) Motivation is provided for all design choices, including algorithms, implementation choices, parameters, data sets and experimental protocols beyond metrics. [yes]

Articles containing theoretical contributions:

Does this paper make theoretical contributions? [yes]

If yes, please complete the list below.

(1) All assumptions and restrictions are stated clearly and formally. [yes]
(2) All novel claims are stated formally (e.g., in theorem statements). [yes]
(3) Proofs of all non-trivial claims are provided in sufficient detail to permit verification by readers with a reasonable degree of expertise (e.g., that expected from a PhD candidate in the same area of AI). [yes]
(4) Complex formalism, such as definitions or proofs, is motivated and explained clearly. [yes]
(5) The use of mathematical notation and formalism serves the purpose of enhancing clarity and precision; gratuitous use of mathematical formalism (i.e., use that does not enhance clarity or precision) is avoided. [yes]
(6) Appropriate citations are given for all non-trivial theoretical tools and techniques. [yes]

Articles reporting on computational experiments:

Does this paper include computational experiments? [no]

If yes, please complete the list below.

(1) All source code required for conducting experiments is included in an online appendix or will be made publicly available upon publication of the paper. The online appendix follows best practices for source code readability and documentation as well as for long-term accessibility. [yes/partially/no]
(2) The source code comes with a license that allows free usage for reproducibility purposes. [yes/partially/no]

(3) The source code comes with a license that allows free usage for research purposes in general. [yes/partially/no]
(4) Raw, unaggregated data from all experiments is included in an online appendix or will be made publicly available upon publication of the paper. The online appendix follows best practices for long-term accessibility. [yes/partially/no]
(5) The unaggregated data comes with a license that allows free usage for reproducibility purposes. [yes/partially/no]
(6) The unaggregated data comes with a license that allows free usage for research purposes in general. [yes/partially/no]
(7) If an algorithm depends on randomness, then the method used for generating random numbers and for setting seeds is described in a way sufficient to allow replication of results. [yes/partially/no/NA]
(8) The execution environment for experiments, the computing infrastructure (hardware and software) used for running them, is described, including GPU/CPU makes and models; amount of memory (cache and RAM); make and version of operating system; names and versions of relevant software libraries and frameworks. [yes/partially/no]
(9) The evaluation metrics used in experiments are clearly explained and their choice is explicitly motivated. [yes/partially/no]
(10) The number of algorithm runs used to compute each result is reported. [yes/no]
(11) Reported results have not been "cherry-picked" by silently ignoring unsuccessful or unsatisfactory experiments. [yes/partially/no]
(12) Analysis of results goes beyond single-dimensional summaries of performance (e.g., average, median) to include measures of variation, confidence, or other distributional information. [yes/no]
(13) All (hyper-) parameter settings for the algorithms/methods used in experiments have been reported, along with the rationale or method for determining them. [yes/partially/no/NA]
(14) The number and range of (hyper-) parameter settings explored prior to conducting final experiments have been indicated, along with the effort spent on (hyper-) parameter optimisation. [yes/partially/no/NA]
(15) Appropriately chosen statistical hypothesis tests are used to establish statistical significance in the presence of noise effects. [yes/partially/no/NA]

Articles using data sets:

Does this work rely on one or more data sets (possibly obtained from a benchmark generator or similar software artifact)? [no]

If yes, please complete the list below.

(1) All newly introduced data sets are included in an online appendix or will be made publicly available upon publication of the paper. The online appendix follows best practices for long-term accessibility with a license that allows free usage for research purposes. [yes/partially/no/NA]
(2) The newly introduced data set comes with a license that allows free usage for reproducibility purposes. [yes/partially/no]
(3) The newly introduced data set comes with a license that allows free usage for research purposes in general. [yes/partially/no]
(4) All data sets drawn from the literature or other public sources (potentially including authors' own previously published work) are accompanied by appropriate citations. [yes/no/NA]
(5) All data sets drawn from the existing literature (potentially including authors' own previously published work) are publicly available. [yes/partially/no/NA]
(6) All new data sets and data sets that are not publicly available are described in detail, including relevant statistics, the data collection process and annotation process if relevant. [yes/partially/no/NA]

(7) All methods used for preprocessing, augmenting, batching or splitting data sets (e.g., in the context of hold-out or cross-validation) are described in detail. [yes/partially/no/NA]

Explanations on any of the answers above (optional):

[Text here; please keep this brief.]