# Model Counting of Query Expressions: Limitations of Propositional Methods[*]

Paul Beame
University of Washington
beame@cs.washington.edu

Jerry Li[*]
MIT
jerryzli@csail.mit.edu

Sudeepa Roy
University of Washington
sudeepa@cs.washington.edu

Dan Suciu
University of Washington
suciu@cs.washington.edu

## ABSTRACT

Query evaluation in tuple-independent probabilistic databases is the problem of computing the probability of an answer to a query given independent probabilities of the individual tuples in a database instance. There are two main approaches to this problem: (1) in *grounded inference* one first obtains the lineage for the query and database instance as a Boolean formula, then performs weighted model counting on the lineage (i.e., computes the probability of the lineage given probabilities of its independent Boolean variables); (2) in methods known as *lifted inference* or *extensional query evaluation*, one exploits the high-level structure of the query as a first-order formula. Although it is widely believed that lifted inference is strictly more powerful than grounded inference on the lineage alone, no formal separation has previously been shown for query evaluation. In this paper we show such a formal separation for the first time.

We exhibit a class of queries for which model counting can be done in polynomial time using extensional query evaluation, whereas the algorithms used in state-of-the-art exact model counters on their lineages provably require exponential time. Our lower bounds on the running times of these exact model counters follow from new exponential size lower bounds on the kinds of $d$-DNNF representations of the lineages that these model counters (either explicitly or implicitly) produce. Though some of these queries have been studied before, no non-trivial lower bounds on the sizes of these representations for these queries were previously known.

[*]This research was partially supported by NSF Awards CCF-1217099, IIS-1115188, IIS-0911036, and IIS-0915054.
[*]This work was done while the author was at the University of Washington.

(c) 2014, Copyright is with the authors. Published in Proc. 17th International Conference on Database Theory (ICDT), March 24-28, 2014, Athens, Greece: ISBN 978-3-89318066-1, on OpenProceedings.org. Distribution of this paper is permitted under the terms of the Creative Commons license CC-by-nc-nd 4.0.

## Categories and Subject Descriptors

H.2.3 [**Database Management**]: Languages—*Query languages*; F.1.3 [**Computation by Abstract Devices**]: Complexity Measures and Classes—*Relations among complexity measures*; I.2.4 [**Artificial Intelligence**]: Knowledge Representation and Methods—*Representation Languages*

## General Terms

Algorithms, Theory

## Keywords

Model counting, Probabilistic databases, Knowledge compilation, FBDD, Read-once branching programs, DNNF, Lower bounds

## 1. INTRODUCTION

Model counting is the problem of computing the number, $\#\Phi$, of satisfying assignments of a Boolean formula $\Phi$. In this paper we are concerned with the weighted version of model counting, which is the same as the probability computation problem on independent random variables. Although model counting is #P-hard in general (even for formulas where satisfiability is easy to check) [25], there have been major advances in practical algorithms that compute exact, weighted model counts for many relatively complex formulas. Exact model counting for propositional formulas (see [12] for a survey) are based on extensions of backtracking search using the *DPLL* family of algorithms [11, 10] that were originally designed for satisfiability search.

We are motivated by probabilistic databases [23], where the query evaluation problem is the following: given a (fixed) Boolean query $Q$, and a database $D$ where each tuple is an independent random variable (the tuple may be present or not, with a known probability), compute $\Pr[Q(D)]$, the probability that $Q$ is true on a random instance of $D$. This is still the weighted model counting problem, with the only difference that the propositional formula $\Phi$ is obtained as the grounding of a first-order formula $Q$ on a database $D$. $\Phi$ is called the *lineage* or the *grounding* of $Q$. An obvious way to compute $\Pr[Q(D)]$ is to first compute the lineage $\Phi$, then perform model counting on $\Phi$. We call this the *grounded inference* approach. In general, however, grounded inference is inefficient, because $\Phi$ is a large propositional formula that depends on many tuples in the database, while the first-order query $Q$ is much smaller.

The mismatch between the high level representation as a first-order formula and the low level of propositional inference was noted early on, and has given rise to various techniques that operate at the first-order level, which are collectively called *lifted inference* in statistical relational models (see [15] for an extensive discussion), or *extensional query evaluation* in probabilistic databases [23]. These methods exploit the high level structure of the first-order formula in order to guide the probabilistic inference.

It is widely believed that lifted inference, or extensional query evaluation, is strictly more powerful than grounded inference. While there have been examples in other contexts where provable separations have been shown (e.g., [20]), no formal separation has previously been shown in the context of query evaluation. We show such a formal separation for the first time.

We describe a family of queries for which we prove formally that grounded inference by current propositional model counting algorithms requires exponential time on any member in the class. We also show that model counting for each query in this class can be done in polynomial time in the size of the domain using a form of lifted inference. This proves that current propositional model counting techniques are strictly weaker than their lifted counterparts, for every query in this family.

Our result on the limitation of inference at the propositional level assumes certain properties of the inference algorithm. To explain these, we first review the state of the art for exact propositional model counting algorithms. These algorithms are based on the DPLL family of algorithms [11, 10], and include several extensions: caching the results of solved sub-problems [18], dynamically decomposing residual formulas into components (Relsat [2]) and caching their counts ([1]), and applying dynamic component caching together with conflict-directed clause learning (CDCL) to further prune the search (Cachet [21] and sharpSAT [24]). In addition to DPLL-style algorithms that compute the counts on the fly, model counting has been addressed through a complementary approach, known as *knowledge compilation*, which converts the input formula into a representation of the Boolean function that the formula defines and from which the model count can be computed efficiently in the size of the representation [7, 8, 14, 19]. Efficiency for knowledge compilation depends both on the size of the representation and the time required to construct it. These two approaches are quite related. As noted in `c2d` [14] (based on component caching) and Dsharp [19] (based on sharpSAT), the traces of all the DPLL-style methods yield knowledge compilation algorithms that can produce what are known as *decision-DNNF* representations [13, 14], a syntactic subclass of *d*-DNNF representations [8, 9];

Thus, the key property we assume about propositional model counting algorithms is that they can be converted to produce a decision-DNNF representation of the propositional formula. Indeed, all the methods for exact model counting surveyed in [12] (and all others of which we are aware) can be converted to knowledge compilation algorithms that produce decision-DNNF representations, without any significant increase in their running time. A decision-DNNF is a rooted DAG where each node either tests a Boolean variable $Z$ and has two outgoing edges corresponding to $Z = 0$ or $Z = 1$, or is an AND-node with two sub-DAGs that do not test any variable in common. These naturally correspond to the two types of operations in any modern DPLL-style algorithm: Shannon expansion on a variable[1] $Z$, or partitioning the formula into two disconnected components[2]. We will refer to the DPLL-style algorithms and knowledge compilation methods used in the state-of-the-art model counters as *decision-DNNF-based model counting algorithms*.

The lower bounds that we prove in this paper are on the size of the decision-DNNF; based on our discussion, these lower bounds also apply to the running time of all decision-DNNF-based algorithm, which includes all modern exact algorithms. Specifically, we prove that, for every query in the class we define, any decision-DNNF for the lineage of that query is exponentially large in the size of the domain. We explain next our lower bounds on the size of the decision-DNNF.

**Our Contributions.** Our first lower bounds are for a family of queries, called $h_k$, $k \geq 1$, [23], for which weighted model counting is not in polynomial time: in fact, probabilistic inference for $h_k$ was shown to be #P-hard [6]. These queries have a very simple lineage, which is a 2-DNF formula. We prove exponential lower bounds of the form $2^{\Omega(\sqrt{n})}$ on the sizes of decision-DNNF representations of these lineages, which is the first non-trivial decision-DNNF lower bound for $h_k$ (Theorem 3.1).

We obtain these lower bounds by first proving lower bounds on the size of FBDDs: an FBDD, or *Free Binary Decision Diagram*, also known as *Read-Once Branching Program*, is a restricted subclass of decision-DNNFs without any AND-nodes (see Section 2). We prove that any FBDD for the Boolean formula representing the lineage of $h_k$ requires at least $2^{n-1}/n$ size for a domain of size $n$. We have shown recently [3] that every decision-DNNF of size $N$ can be converted into an FBDD of size at most $N2^{\log^2 N}$. Together, these two results imply our lower bound of $2^{\Omega(\sqrt{n})}$ on the sizes of decision-DNNF. We note that a lower bound on the size of the FBDD for $h_k$ was known previously [17], but that bound, $2^{\Omega(\log^2 n)}$, is insufficient to yield any decision-DNNF lower bound using our translation.

Our lower bounds for $h_k$ in Theorem 3.1 do not yet prove the separation between lifted and grounded probabilistic inference, because weighted model counting for $h_k$ is #P-hard. To obtain that separation, we extend substantially the class of queries for which the same lower bounds on the size of FBDDs holds, and, therefore, the same bounds on the size of the decision-DNNFs. Each query $h_k$ is a disjunction of $k+1$ queries, $h_k = h_{k0} \vee h_{k1} \vee \ldots \vee h_{kk}$. We prove that for *any* Boolean combination of these $k+1$ queries, the lower bound on the size of the FBDDs continues to apply. Thus, one may take the disjunction of these $k+1$ queries (and obtain $h_k$), or their conjunction, or any other combination: for any such query the lower bound on the size of the FBDD continues to hold. The only restriction on the Boolean combination is that it has to depend on all $k+1$ queries: this is necessary, otherwise the lineage of the query is known to admit an OBDD[3] of linear size in the active domain [16].

[1]$\Pr[\Phi] = \Pr[\Phi[Z=0]] \cdot (1 - \Pr[Z]) + \Pr[\Phi[Z=1]] \cdot \Pr[Z]$.
[2]$\Pr[\Phi_1 \wedge \Phi_2] = \Pr[\Phi_1] \cdot \Pr[\Phi_2]$.
[3]An OBDD is an FBDD where every path from the root to

Our lower bound is based on showing that, every FBDD for such a Boolean combination can, with a small increase in size, be converted into an FBDD that simultaneously represents the lineages of all of its constituent queries $h_{k0}, \ldots, h_{kk}$; from here one immediately obtains an FBDD for their disjunction, which is $h_k$, and for which the previous lower bounds hold.

What makes our result surprising is the fact that, for some Boolean combinations of $h_{k0}, \ldots, h_{kk}$, weighted model counting can be done in polynomial time, by using the inclusion/exclusion formula. This is a form of lifted inference, because the inclusion/exclusion formula is applied to the First Order expression, namely to the Boolean combination of the FO expressions $h_{k0}, \ldots, h_{kk}$. Inclusion/exclusion may be exponential in $k$, but this depends only on the query, not the data, and lifted inference takes only polynomial time in the size of the active domain. In contrast, if one were to apply naively inclusion/exclusion to the grounding of the query, the complexity would be exponential in the size of the active domain, and for that reason, current model counting algorithm on propositional formulas do not use inclusion/exclusion. Our lower bounds prove that this limitation is significant, by showing that their runtime is exponential in the active domain. In other words, lifted inference gains extra power by using the First Order expression to guide the application of inclusion/exclusion, and this ability cannot be recovered at the propositional level by any decision-DNNF-based algorithm.

This proves our separation result between grounded and lifted probabilistic inference. More precisely, we use the general characterization given in [6] for Union of Conjunctive Queries (UCQ), which gives a specific property (entirely) based on the structure of $Q$ that allows exact model counting in time polynomial in the size of the database. This yields a $2^{\Omega(\sqrt{n})}$ versus $n^{O(1)}$ separation between these propositional and lifted methods for weighted model counting for a wide variety of such queries $Q$ (Theorem 3.7).

As we explained, our lower bounds on the running time of weighted model counting algorithm apply to decision-DNNF-based model counting algorithms. Their input is a CNF, and their component rule writes a residual formula as $\Phi = \Phi_1 \wedge \Phi_2$ where $\Phi_1$, $\Phi_2$ are sets of clauses with no common variables. On the other hand, queries in probabilistic databases have lineage expressions that are DNF formulas, were a more natural decomposition would be $\Phi = \Phi_1 \vee \Phi_2$, where $\Phi_1, \Phi_2$ are formulas with no common variables. A natural question is whether a simple extension of a model counting algorithm with this kind of decomposition could significantly improve their power. We answer this in the negative. More precisely, we strengthen (Theorem 2.1) the conversion from decision-DNNF to FBDD in [3] to one with the same complexity that applies to a new, more general class of representations than decision-DNNFs, which we call *decomposable logic decision diagrams* (DLDDs). A DLDD is a DAG where every node either tests a Boolean variable $Z$ (like in a decision-DNNF), or applies a binary Boolean operator to its two children, $f(\Phi_1, \Phi_2)$, where the two children $\Phi_1$ and $\Phi_2$ have no common Boolean variables. We prove that every decision-DNNF with $N$ nodes can be converted into an equivalent FBDD with at most $N2^{\log^2 N}$ nodes, thus matching the result in [3] for decision-DNNF. Therefore, our lower bounds extend to algorithms that use more general decompositions, with any unary or binary operators, including independent AND, independent OR, and negation.

a leaf tests the Boolean variables in the same order.

**Roadmap.** We discuss some useful knowledge compilation representations in Section 2. In Section 3, we describe our main results which are proved in the following Sections 4 and 5. We discuss related issues in Section 6.

## 2. BACKGROUND

In this section we review the knowledge compilation representations used in the rest of the paper.

**FBDDs.** An FBDD [4] is a rooted directed acyclic graph (DAG) $\mathcal{F}$ that computes $m$ Boolean functions $\mathbf{\Phi} = (\Phi_1, \ldots, \Phi_m)$. $\mathcal{F}$ has two kinds of nodes: *decision nodes*, which are labeled by a Boolean variable $X$ and have two outgoing edges labeled 0 and 1; and *sink nodes* labeled with an element from $\{0,1\}^m$. Every path from the root to some sink node may test a Boolean variable $X$ at most once. For each assignment $\theta$ on all the Boolean variables, $\mathbf{\Phi}[\theta] = (\Phi_1[\theta], \ldots, \Phi_m[\theta]) = L$, where $L$ is the label of the unique sink node reachable by following the path defined by $\theta$. The *size* of the FBDD $\mathcal{F}$ is the number nodes in $\mathcal{F}$. Typically $m = 1$, but we will also consider FBDDs $\mathcal{F}$ with $m > 1$ and call $\mathcal{F}$ a *multi-output FBDD*.

For every node $u$, the sub-DAG of $\mathcal{F}$ rooted at $u$, denoted $\mathcal{F}_u$, computes $m$ Boolean functions $\mathbf{\Phi}_u$ defined as follows. If $u$ is a decision node labeled with $X$ and has children $u_0, u_1$ for 0- and 1-edge respectively, then $\mathbf{\Phi}_u = (\neg X)\mathbf{\Phi}_{u_0} \vee X\mathbf{\Phi}_{u_1}$; if $u$ is a sink node labeled $L \in \{0,1\}^m$, then $\mathbf{\Phi}_u = L$. $\mathcal{F}$ computes $\mathbf{\Phi} = \mathbf{\Phi}_r$ where $r$ is the root. The probability of each of the $m$ functions can be computed in time linear in the size of the FBDD using a simple dynamic program: $\Pr[\mathbf{\Phi}_u] = (1 - p(X))\Pr[\mathbf{\Phi}_{u_0}] + p(X)\Pr[\mathbf{\Phi}_{u_1}]$.

For our purposes, it will also be useful to consider FBDDs with *no-op nodes*. A no-op node is not labeled by any variable, and has a single child; the meaning is that we do not test any variable, but simply continue to its unique child. Every FBDD with no-op nodes can be transformed into an equivalent FBDD without no-op nodes, by simply skipping over the no-op node.

**Decision-DNNFs.** A decision-DNNF[5] $D$ generalizes an FBDD allowing *decomposable AND-nodes* in addition to decision-nodes, *i.e.*, any AND-node $u$ must satisfy the restriction that, for its two children $u_1, u_2$, the sub-DAGS $\mathcal{D}_{u_1}$ and $\mathcal{D}_{u_2}$ do not mention any common Boolean variables. The function $\mathbf{\Phi_u}$ is defined as $\mathbf{\Phi_u} = \mathbf{\Phi_{u_1}} \wedge \mathbf{\Phi_{u_2}}$, and its probability is computed as $\Pr[\mathbf{\Phi_u}] = \Pr[\mathbf{\Phi_{u_1}}] \cdot \Pr[\mathbf{\Phi_{u_2}}]$. In a decision-DNNF, similar to FBDDs, any Boolean variable can be tested at most once along any path from the root to any sink.

**DLDDs.** In this paper we introduce *Decomposable*

[4]FBDDs are also known as a *Read Once Branching Programs*.

[5]A decision-DNNFis a special case of both an AND-FBDD (which has no restriction on AND nodes) [26] and a d-DNNF [8], which is a restricted kind of circuit used for knowledge compilation; see [3] for a discussion.

*Decision Logic Diagrams* or DLDDs by further generalizing decision-DNNFs. A DLDD can also have NOT-nodes $u$ having a unique child $u_1$, and decomposable OR-, XOR-, and EQUIV-nodes similar to decomposable AND-nodes[6]: (i) for a NOT-node, $\mathbf{\Phi_u} = \neg\mathbf{\Phi_{u_1}}$, and $\Pr[\mathbf{\Phi_u}] = 1-\Pr[\mathbf{\Phi_{u_1}}]$; (ii) for an OR-node, $\mathbf{\Phi_u} = \mathbf{\Phi_{u_1}} \vee \mathbf{\Phi_{u_2}}$, and $\Pr[\mathbf{\Phi_u}] = 1-(1-\Pr[\mathbf{\Phi_{u_1}}])\cdot(1-\Pr[\mathbf{\Phi_{u_2}}])$; (iii) for an XOR-node, $\mathbf{\Phi_u} = \mathbf{\Phi_{u_1}}\cdot\neg\mathbf{\Phi_{u_2}} \vee \neg\mathbf{\Phi_{u_1}}\cdot\mathbf{\Phi_{u_2}}$, and (iv) for an EQUIV-node, $\mathbf{\Phi_u} = \mathbf{\Phi_{u_1}}\cdot\mathbf{\Phi_{u_2}} \vee \neg\mathbf{\Phi_{u_1}}\cdot\neg\mathbf{\Phi_{u_2}}$ (again $\Pr[\mathbf{\Phi_u}]$ can easily be computed from $\Pr[\mathbf{\Phi_{u_1}}], \Pr[\mathbf{\Phi_{u_2}}]$). Hence the probability of the formula can still be computed in time linear in $\mathcal{D}$.

**Conversion of a DLDD into an equivalent FBDD.** The trace of any DPLL-based algorithm with caching and components is a decision-DNNF. Therefore any lower bound on the size of decision-DNNFs represents a lower bound on the running time of modern model counting algorithms. We have proven recently the first lower bounds on decision-DNNFs [3]. However, model counting algorithms were designed for CNF expressions: for example, the component analysis partitions the clauses into two disconnected components (without common variables), then computes the probability as $\Pr[\Phi_1 \wedge \Phi_2] = \Pr[\Phi_1]\Pr[\Phi_2]$. In order to run such an algorithm on a DNF expression (which are more related to lineages in databases) one would naturally first apply a negation, which transforms the formulas into CNF. This suggest a simple extension of such algorithms: allow the application of the negation operator at any step. The trace now also has NOT-nodes and therefore is a special case of DLDDs. But we prove our first result in the paper for general DLDDs:

THEOREM 2.1. *For any DLDD $\mathcal{D}$ with $N$ nodes there exists an equivalent FBDD $\mathcal{F}$ computing the same formula as $\mathcal{D}$, with at most $N2^{\log^2 N}$ nodes (at most quasi-polynomial increase in size).*

In [3] we have proven a similar result with the same bound for decision-DNNFs; now we strengthen it to DLDDs; the extension is rather simple and appears in the full version of the paper [4].

## 3. MAIN RESULTS

Here we formally state our main results and discuss their implications, and defer the proofs to the following sections. We start by introducing some elementary queries that work as building blocks for the class of queries considered in these results [6, 17]:

Let $[n]$ denote the set $\{1,\ldots,n\}$. Fix $k > 0$ and consider the following set of $k+1$ Boolean queries $\mathbf{h_k} = (h_{k0},\cdots,h_{kk})$, where

$$\begin{aligned} h_{k0} &= \exists x_0 \exists y_0\; R(x_0) \wedge S_1(x_0,y_0) \\ h_{k\ell} &= \exists x_\ell \exists y_\ell\; S_\ell(x_\ell,y_\ell) \wedge S_{\ell+1}(x_\ell,y_\ell) \qquad \forall \ell \in [k-1] \\ h_{kk} &= \exists x_k \exists y_k\; S_k(x_k,y_k) \wedge T(y_k) \end{aligned}$$

Fix a domain size $n > 0$; for each $i,j \in [n]$, let $R(i)$, $S_1(i,j),\ldots,S_k(i,j)$, $T(j)$ be Boolean variables representing potential tuples in the database. Then the corresponding *lineages*, the associated Boolean expressions for these queries are [7]:

$$H_{k0} = \bigvee_{i,j\in[n]} R(i)S_1(i,j), \qquad H_{kk} = \bigvee_{i,j\in[n]} S_k(i,j)T(j),$$
$$H_{k\ell} = \bigvee_{i,j\in[n]} S_\ell(i,j)S_{\ell+1}(i,j) \qquad \forall \ell \in [k-1]$$

We define $\mathbf{H_k} = (H_{k0},\ldots,H_{kk})$. Two well-studied queries [6] that we will consider in this section are given below:

**Query $h_k$:** $h_k$ is a disjunction on the queries in $\mathbf{h_k}$: $h_k = h_{k0} \vee h_{k1} \vee \cdots \vee h_{kk}$. The lineage $H_k$ of $h_k$ is given by $H_k = H_{k0} \vee H_{k1} \vee \cdots \vee H_{kk}$.

**Query $h_0$:** Also we define $h_0$ that uses a single relation symbol $S$ in addition to $R$ and $T$: $h_0 = \exists x \exists y\; R(x) \wedge S(x,y) \wedge T(y)$. $S$ is defined on Boolean variables $S(i,j)$, $i,j \in [n]$, and therefore the lineage $H_0$ of $h_0$ is $H_0 = \bigvee_{i,j\in[n]} R(i)S(i,j)T(j)$.

## Lower bounds on FBDDs for queries $h_0$, $h_k$

Jha and Suciu [17] previously showed that every FBDD for the lineage $H_1$ of $h_1$ has size $2^{\Omega(\log^2 n)}$. Our first result improves this to an exponential lower bound, not just for $H_1$ but also for $H_0$ and all $H_k$ for $k > 1$:

THEOREM 3.1. *For every $n > 0$, any FBDD for $H_0$ or $H_k$ for $k \geq 1$ has $\geq 2^{(n-1)}/n$ nodes.*

It is known that weighted model counting for both $H_0$ and $H_k$ is #P-hard [6]. However, the lower bounds we show on these FBDD sizes are absolute (independent of any complexity theoretic assumption) and do not rely on the #P-hardness of the associated weighted model counting problems. We give the proof in Section 4. This improved bound is critical for proving the overall lower bound result in this paper (Theorem 3.4).

While we do not need $h_0$ and $H_0$ in the rest of the paper, we include it in Theorem 3.1 because it is obtained in a fashion similar to that for $H_k$ and substantially improves on a $2^{\Omega(\sqrt{n})}$ lower bound for $H_0$ from our previous work [3] which was based on a result by of Bollig and Wegener [5][8]. Our new lower bound improves this to the nearly optimal $2^{n-1}/n$.

We also note that our stronger lower bounds for $H_1$ give instances of bipartite 2-DNF formulas that are simpler to describe than those of [5] but yield as good a lower bound on FBDD sizes in terms of their number of variables and even better bounds as a function of their number of terms[9].

[6]These four nodes along with NOT-nodes can capture all possible non-constant functions on two Boolean variables

[7]For simplicity, conjunctions in Boolean formulas are represented as products.

[8]Bollig and Wegener defined a set $E_n \subseteq [n]\times[n]$ for which any FBDD for the formula $\bigvee_{(i,j)\in E_n} R(i)T(j)$ requires size $2^{\Omega(\sqrt{n})}$ which obviously implies the same lower bound for $H_0$. The set $E_n$ is given as follows: Assume that $n = p^2$ where $p$ is a prime number. Each number $0 \leq i < n$ can be uniquely written as $i = a + bp$ where $0 \leq a,b < p$. Then: $E_n = \{(i+1,j+1) \mid i = a+bp, j = c+dp, c \equiv (a+bd) \mod p\}$.

[9]In the formulas of [5], $p$ is analogous to $n$ in our formulas and theirs have $p^3$ terms, versus only $2n^2$ for our formulas.

## Lower bounds for FBDDs for queries over $\mathbf{h_k}$

Theorem 3.1 gives a lower bound on $h_k$, which is simply the logical OR of the queries in $\mathbf{h_k}$. Theorem 3.3 below generalizes this result by allowing queries that are *arbitrary functions* of queries in $\mathbf{h_k}$.

Let $f(\mathbf{X}) = f(X_0, X_1, \cdots, X_k)$ be an arbitrary Boolean function on $k+1$ Boolean variables $\mathbf{X} = (X_0, \cdots, X_k)$, and $Q$ the Boolean query $Q = f(h_{k0}, h_{k1}, \cdots, h_{kk})$. Clearly, the lineage of $Q$ is $f(\mathbf{H_k}) = f(H_{k0}, H_{k1}, \cdots, H_{kk})$.

Example 3.2. *If $f(X_0, X_1, \cdots, X_k) = \bigvee_{\ell=0}^{k} X_\ell$, we get query $h_k = \bigvee_{\ell=0}^{k} h_{k\ell}$; its lineage is $H_k = \bigvee_{\ell=0}^{k} H_{k\ell}$.*

The function $f$ *depends on* a variable $X_\ell$, $\ell \in \{0, \ldots, k\}$, if there is an assignment $\mu_\ell$ on the rest of the variables $\mathbf{X} \backslash \{X_\ell\}$ such that $f[\mu_\ell] = X_\ell$ or $\neg X_\ell$.

Theorem 3.3. *If $f$ depends on all $k+1$ variables $X_0, \cdots, X_k$, then any FBDD $\mathcal{F}$ with $N$ nodes for the lineage of $Q = f(h_{k0}, \cdots, h_{kk})$ can be converted into a multi-output FBDD for $(H_{k0}, \cdots, H_{kk})$ with $O(k2^k n^3 N)$ nodes. In particular, for $k \leq \alpha n$ for any constant $\alpha < 1$, $\mathcal{F}$ requires at least $2^{\Omega(n)}$ nodes.*

We prove the theorem in Section 5. The condition that $f$ depends on all variables is necessary (see Sections 4 and 5): if $Q$ does not depend on any one of the queries in $\mathbf{h_k}$, then its lineage has an FBDD of size linear in the number of Boolean variables.

Theorem 3.3 extends prior work in several ways. First, it is the first result showing exponential lower bounds on FBDDs for a large class of queries. Prior to Theorem 3.3 the only known lower bound was the quasipolynomial lower bound for $h_1$ [17]. Second, although a conversion of an FBDD for a specific query $Q_W$ (described later in this section) into one for $h_1$ was given in [17], this conversion did not extend to other queries. While we were inspired by that proof, the techniques we use in Theorem 3.3 are considerably more powerful, and use new ideas which can be of independent interest to show lower bounds on the size of FBDDs in general.

We also extend the lower bound in Theorem 3.3 by proving a dichotomy theorem for a slightly more general class of queries: any query in this class either has a polynomial-time model counting algorithm, or all existing decision-DNNF-based model counting algorithms require exponential time. The details of the dichotomy theorem appear in the full version of the paper [4].

## Lower Bounds for Model Counting Algorithms for Queries over $\mathbf{h_k}$

Theorems 2.1, 3.1, and 3.3 together prove the following lower bound result:

Theorem 3.4. *If $Q$ is a Boolean combination of the queries in $\mathbf{h_k}$ that depends on all $k+1$ queries in $\mathbf{h_k}$, then any DLDD (and therefore any decision-DNNF) for the lineage $\Theta$ of $Q$ has size $2^{\Omega(\sqrt{n})}$. Further, this size is $2^{\Omega(n/k)}$ if $Q$ is monotone in the queries in $\mathbf{h_k}$.*

Proof. Let $N$ be the size of a DLDD for $Q$. By Theorem 2.1, $Q$ has an FBDD of size $N2^{\log^2 N}$. By Theorem 3.3, $H_1$ has an FBDD for size $2^{O(\log^2 N)}$, which has to be $2^{\Omega(n)}$

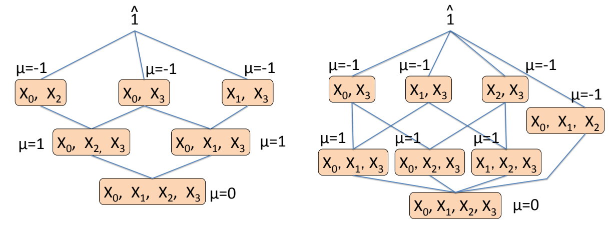

Figure 1: The lattices for (a) $f_W$, (b) $f_9$.

by Theorem 3.1, implying that $N$ is $2^{\Omega(\sqrt{n})}$. Further, by Theorem 3.1, the lower bound is $2^{\Omega(n/k)}$ if $Q$ is monotone in the queries in $\mathbf{h_k}$ since a monotone function on $k+1$ variables is a monotone $(k+1)$-DNF and thus the lineage of $Q$ is given by a monotone $2(k+1)$-DNF in its inputs. □

Since, as we discussed previously, current propositional exact weighted model counting algorithms (extended with negation to handle DNFs) without loss of generality yield DLDDs of size at most their running time, we immediately obtain:

Corollary 3.5. *All current propositional exact model counting algorithms require running time $2^{\Omega(\sqrt{n})}$ to perform weighted model counting for any query $Q$ that is a Boolean combination of the queries in $\mathbf{h_k}$ and depends on all $k+1$ queries in $\mathbf{h_k}$.*

## Propositional versus Lifted Model Counting

Theorem 3.4 when applied to query $h_k$, $k \geq 1$, is not surprising: #P-hardness of $h_k$ makes it unlikely to have an efficient model counting algorithm. However, there are many other query combinations over $\mathbf{h_k}$ for which lifted methods taking advantage of the high-level structure yield polynomial-time model counting and therefore outperform current propositional techniques.

Consider the case when $Q = f(\mathbf{h_k})$ and $f$ is a monotone Boolean formula $f(X_0, \cdots, X_k)$, and thus $Q$ is a UCQ query. Here the cases when weighted model counting for $Q$ can be done in polynomial time are entirely determined by the structure of the query expression[10] $f$, and we review it here briefly following [23].

To check if weighted model counting for $Q$ is computable in polynomial time, write $f$ as a CNF formula, $f = \bigwedge_i C_i$, where each (positive) clause $C_i$ is a set of propositional variables $C_i \subseteq \{X_0, \cdots, X_k\}$. Define the lattice $(L, \leq)$, where $L$ contains all subsets $u \subseteq \mathbf{X}$ that are a union of clauses $C_i$, and the order relation is given by $u \leq v$ if $u \supseteq v$. The maximal element of the lattice is $\emptyset$, (we denote it $\hat{1}$), while the minimal element is $\mathbf{X}$ (we denote it $\hat{0}$). The Möbius function on the lattice $L$, $\mu : L \times L \rightarrow \mathbf{R}$, is defined as $\mu(u,u) = 1$ and $\mu(u,v) = -\sum_{u<w\leq v} \mu(w,v)$ [22]. The following holds [23]: if $\mu(\hat{0}, \hat{1}) = 0$, then weighted model counting for $Q$ can be done in time polynomial in $n$ (using in the inclusion/exclusion formula on the CNF); if $\mu(\hat{0}, \hat{1}) \neq 0$, then the weighted model counting problem for $Q$ is #P-hard.

[10]The propositional formula $f$ describes the query expression $Q$, and should not be confused with the propositional grounding of $Q$ on the instance $R(i), S_1(i,j), \cdots, S_k(i,j), T(j)$; $\ell \in [1, k-1]$, $i, j \in [n]$.

EXAMPLE 3.6. *Here we give examples of easy and hard queries:*

- *For a trivial example, $h_k = h_{k0} \vee \cdots \vee h_{kk}$ has a single clause, hence its lattice has exactly two elements $\hat{0}$ and $\hat{1}$, and $\mu(\hat{0}, \hat{1}) = -1$, hence $h_k$ is #P-hard.*
- *Two more interesting examples for $k = 3$:*

$$\begin{aligned} f_W &= (X_0 \vee X_2) \wedge (X_0 \vee X_3) \wedge (X_1 \vee X_3) \\ f_9 &= (X_0 \vee X_3) \wedge (X_1 \vee X_3) \wedge (X_2 \vee X_3) \\ &\qquad \wedge (X_0 \wedge X_1 \wedge X_2) \end{aligned}$$

*Their lattices, shown in Figure 1, satisfy $\mu(\hat{0}, \hat{1}) = 0$. Therefore, for the queries $Q_W = f_W(h_{30}, h_{31}, h_{32}, h_{33})$ and $Q_9 = f_9(h_{30}, h_{31}, h_{32}, h_{33})$, weighted model counting can be done in polynomial time. For example, to compute the probability of $Q_W$ we apply the inclusion/exclusion formula on the query expression and get $\Pr[Q_W] =$*

$$\begin{aligned} &\Pr[h_{30} \vee h_{32}] + \Pr[h_{30} \vee h_{33}] + \Pr[h_{31} \vee h_{33}] \\ &- \Pr[h_{30} \vee h_{32} \vee h_{33}] - \Pr[h_{30} \vee h_{31} \vee h_{33}] \\ &- \Pr[h_{30} \vee h_{31} \vee h_{32} \vee h_{33}] + \Pr[h_{30} \vee h_{31} \vee h_{32} \vee h_{33}] \end{aligned}$$

*While computing $\Pr[h_{30} \vee h_{31} \vee h_{32} \vee h_{33}]$ is #P-hard (because this query is $h_3$), the two occurrences of this term cancel out, and for all remaining terms one can compute the probability in polynomial time in $n$ (since each misses at least one term $h_{30}, h_{31}, h_{32}, h_{33}$). Thus, weighted model counting can be done in polynomial time for $Q_W$ (similarly for $Q_9$), at the query expression level.*

On the other hand, Theorem 3.4 proves that, if we ground $Q_W$ or $Q_9$ first, then any decision-DNNF-based model counting algorithm will take exponential time on the lineage. This leads to the main separation result of this paper:

THEOREM 3.7 (MAIN RESULT). *Let $Q$ be any monotone, Boolean combination of the queries in $\mathbf{h_k}$ that depends on all $k + 1$ queries in $\mathbf{h_k}$ such that $\mu(\hat{0}, \hat{1}) = 0$. Then weighted model counting for $Q$ can be done in time polynomial in $n$, whereas all existing decision-DNNF-based propositional algorithms for model counting require exponential time on the lineage.*

## 4. EXPONENTIAL LOWER BOUNDS FOR ALL $H_K$

In this section we prove Theorem 3.1 which gives lower bounds on the sizes of FBDDs computing all $H_k$. We find it convenient to prove these bounds assuming a natural property of FBDDs. We show that we can ensure this property with only minimal change in FBDD size, yielding our claimed lower bounds.

Let $\Phi$ be a Boolean formula. A *prime implicant* (or *minterm*) of $\Phi$ is a term $T$ such that $T \Rightarrow \Phi$ and no proper subterm of $T$ implies $\Phi$. If $T$ involves $k$ variables then we call it a *k-prime implicant.* 1-prime implicants are also known as *unit variables.* For example, $X$ and $W$ are unit in $X \vee YZ \vee YU \vee W$.

The following definition is motivated by the *unit clause rule* in DPLL algorithms which are primarily designed for satisfiability of CNF formulas. If there is any clause consisting of a single variable or its negation (a unit clause), then DPLL immediately sets such variables, one after another, since their value is forced.

DEFINITION 4.1. *Let $\mathcal{F}$ be an FBDD for a Boolean function $\Phi$. Call a node $u$ in $\mathcal{F}$ a* unit node *if $\Phi_u$ has a unit, and a* decision node *otherwise. We say that $\mathcal{F}$ follows the* unit rule *if for every unit node $u$ the variable tested at $u$ is a unit.*

In the special case that $\Phi$ is a monotone formula, we can apply a transformation in order to convert any FBDD $\mathcal{F}$ for $\Phi$ into one that follows the unit rule and is not much larger than $\mathcal{F}$.

For a variable $X$ of $\Phi$, define the *degree* of $X$ in $\Phi$ to be the maximum over all partial assignments $\theta$ of the number of unit of $\Phi[\theta \cup \{X{=}1\}]$ that are not units of $\Phi[\theta]$. (If $\Phi$ is a DNF formula then the degree of $X$ is at most the number of distinct variables to co-occur in terms with $X$.) Write $\Delta(\Phi)$ for the maximum degree of any variable in $\Phi$. In section 4.1 we prove the following:

LEMMA 4.2. *If $\Phi$ is a monotone formula with FBDD $\mathcal{F}$ of size $N$, then $\Phi$ has an FBDD of size at most $\Delta(\Phi) \cdot N$ that follows the unit rule.*

Since $H_k$ obviously has degree at most $n$ (for variables $R(i)$ and $T(j)$), we obtain the following corollary.

COROLLARY 4.3. *If $H_k$ has an FBDD of size $N$, then $H_k$ has an FBDD of size at most $nN$ that follows the unit rule.*

Now Theorem 3.1 is an immediate consequence of Corollary 4.3 together with the following lemma.

LEMMA 4.4. *Every FBDD $\mathcal{F}$ for $H_k$ that follows the unit rule has size $\geq 2^{(n-1)}$.*

The proof of Lemma 4.4 follows using a general technique in which one defines a notion of *admissible* paths in $\mathcal{F}$. We will give such a definition and show that no two admissible paths in $\mathcal{F}$ can lead to the same node of $\mathcal{F}$ since they must correspond to different subfunctions of $H_k$. We will further show that every admissible path branches off from other admissible paths at least $n - 1$ times, guaranteeing that $\mathcal{F}$ must contain a complete binary tree of distinct nodes of depth $n - 1$ (in which edges may have been stretched to partial paths).

For the remainder of this section we fix some FBDD $\mathcal{F}$ for $H_k$ that follows the unit rule. Given a path $P$ in $\mathcal{F}$, let $Row(P)$ be the set of $i \in [n]$ so that $P$ tests $R(i)$ at a decision node or there are $\ell$ and $j$ so that $P$ tests $S_\ell(i, j)$ at a decision node; similarly, let $Col(P)$ be the set of $j \in [n]$ which $P$ tests $T(j)$ at a decision node or there is some $\ell$ and $i$ so that $P$ tests $S_\ell(i, j)$ at a decision node. Let $\mathcal{P}$ be the set of partial paths $P$ starting at the root and ending at a (non-leaf) decision node so that both $|Row(P)| < n$ and $|Col(P)| < n$ but any extension of $P$ has either $|Row(P)| = n$ or $|Col(P)| = n$.

The following is an easy observation.

LEMMA 4.5. *For all $k \geq 0$, if $P_1, P_2 \in \mathcal{P}$ with $H_k[P_1] = H_k[P_2]$ then the two paths test the same set of $R$ and $T$ variables and must assign those tested the same values.*

| $R(i)$ | $S_1(i,j)$ | $S_2(i,j)$ | $S_3(i,j)$ | $S_4(i,j)$ | $S_5(i,j)$ | $T(j)$ |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |

Table 1: The patterns for admissible paths for $k = 5$.

PROOF. Suppose that there is some $R(i)$ such that $P_1$ assigns $R(i)$ value $b \in \{0, 1\}$ that $P_2$ does not. $H_k[P_2] = H_k[P_1]$ does not depend on $R(i)$ so we can assume without loss of generality that $P_2$ assigns $R(i)$ value $1 - b$. Suppose without loss of generality that $P_1$ sets $R(i)$ to 1 and $P_2$ sets $R(i)$ to 0.

First consider the case $k = 0$. Let $j_1 \in [n] - Col(P_1)$. Since $P_2$ sets $R(i)$ to 0, $H_0[P_2]$ does not depend on $S(i, j_1)$ but $H_0[P_1]$ sets neither $T(j_1)$ nor $S(i', j_1)$ for any $i'$, so it does depend on $S(i, j_1)$, a contradiction.

Now suppose that $k \geq 1$. Let $j_2 \in [n] - Col(P_2)$. As $\mathcal{F}$ follows the unit rule, this implies that $P_1$ sets $S_1(i, j) = 0$ for all $j$, which in particular implies that $H_k[P_1]$, and thus $H_k[P_2]$, does not depend on $S_1(i, j_2)$. However, since $j_2 \notin Col(P_2)$, all terms of $H_{k\ell}$ for $\ell \in [k]$ involving indices $(i', j_2)$ are unset for every $i'$, which implies that $H_k[P_2]$ depends on $S_1(i, j_2)$, a contradiction. The case when the difference is $T(j)$ is analogous. □

We will first prove Lemma 4.4 for $k = 2m + 1$ odd. The cases when $k > 0$ is even as well as when $k = 0$ use almost identical techniques and their proofs appear in the full version of this paper [4].

DEFINITION 4.6. *Let $P$ be a partial path through $\mathcal{F}$ starting at the root. It is* admissible *if for all $i, j$, it is consistent with one of the four following assignments:*

1. *$R(i) = T(j) = 0$ and $S_\ell(i, j) = 0$ for all $\ell$ odd and $S_\ell(i, j) = 1$ for all $\ell$ even,*
2. *$R(i) = T(j) = 1$ and $S_\ell(i, j) = 1$ for all $\ell$ even and $S_\ell(i, j) = 0$ for all $\ell$ odd,*
3. *$R(i) = 0, T(j) = 1$ and $S_\ell(i, j) = 1$ for all $\ell$ even and $S_\ell(i, j) = 0$ for all $\ell$ odd, or*
4. *$R(i) = 1, T(j) = 0$ and $S_\ell(i, j) = 1$ for all $\ell$ even and $S_\ell(i, j) = 0$ for all $\ell$ odd.*

*$P$ is* forbidden *if it is not admissible. Let $\mathcal{A} \subset \mathcal{P}$ be the set of admissible paths in $\mathcal{P}$. (See Table 1 for the case $k = 5$).*

LEMMA 4.7. *If $P_1, P_2 \in \mathcal{A}$ are distinct then $H_k[P_1] \neq H_k[P_2]$.*

PROOF. Suppose $P_1, P_2 \in \mathcal{A}$ are distinct with $H_k[P_1] = H_k[P_2] = F$. Let $u$ be the first node at which $P_1$ and $P_2$ diverge, and assume without loss of generality that $P_1$ takes the 0-edge and $P_2$ takes the 1-edge. Notice that $u$ must be a decision node. By Lemma 4.5, the node $u$ cannot test a $R(i)$ or $T(j)$ variable so it must test $S_\ell(i, j)$ for some $i, j$. Assume that $\ell$ is even (the case when $\ell$ is odd is symmetrical; switch the roles of $P_1$ and $P_2$). Then $F$ does not contain the prime implicant $S_\ell(i, j)S_{\ell+1}(i, j)$ and does not contain any units, but along $P_2$ the variable $S_\ell(i, j) = 1$ , so $S_{\ell+1}(i, j) = 0$ along $P_2$. This implies that $F$ does not contain the prime implicant $S_{\ell+1}(i, j)S_{\ell+2}(i, j)$ but since $P_1$ cannot set $S_{\ell+1}(i, j) = 0$ as otherwise it would be forbidden, this implies that $P_1$ sets $S_{\ell+2}(i, j) = 0$. Inductively, we conclude that $S_{\ell+2p}(i, j)$ is set to zero on $P_1$ and $S_{\ell+1+2p}(i, j)$ is set to zero on $P_2$ for all non-negative integers $p \leq (k - \ell - 1)/2$. In particular, $S_k(i, j) = 0$ along $P_2$, so the prime implicant $S_k(i, j)T(j)$ does not appear in $F$; as $F$ has no units, $T(j)$ must be set to zero in $P_1$, as otherwise it would be forbidden. Doing the same procedure but inducting downwards, we also conclude that $R(i) = 0$ in $P_1$ and $S_1(i, j) = 0$ in $P_2$. However, by Lemma 4.5, this implies that $R(i) = T(j) = 0$ in $P_2$, and since $S_1(i, j) = S_k(i, j) = 0$ we conclude that $P_2$ is forbidden, which is a contradiction. □

PROOF OF LEMMA 4.4. By Lemma 4.7, it suffices to count how many paths are in $\mathcal{A}$, because each such path must correspond to a unique node in the FBDD. We show that there are at least $2^{n-1}$ such paths. For any path $P \in \mathcal{A}$, call an assignment at a decision node $u$ along $P$ *forced* if taking the opposite assignment would have resulted in a forbidden path, and call the assignment *unforced* otherwise. We claim that there are at least $n - 1$ unforced assignments along any path $P \in \mathcal{P}$. Since some extension of $P$ either sets some variable in all rows or in all columns, $P$ itself must have either $|Row(P)| = n - 1$ or $|Col(P)| = n - 1$. Without loss of generality assume that $|Row(P)| = n - 1$. Then the patterns of admissible paths ensure that, for each $i \in Row(P)$, the first decision node $u$ along $P$ testing a variable either of the form $R(i)$ or $S_\ell(i, j)$ for some $\ell$ and $j$ must be unforced (see Table 1). So there must be at least $n - 1$ unforced assignments.

We now define an injection from $\{0, 1\}^{n-1}$ to $\mathcal{A}$, as follows: map each sequence of bits $(a_1, \ldots, a_{n-1})$ to the unique path $P \in \mathcal{A}$ that at its $i$-th unforced decision takes the $a_i$-edge for $i \leq n - 1$, takes the 1-edge at all unforced decisions after its first $n - 1$ unforced decisions, makes all forced decisions as required, and at each unit node takes the 0 branch. The existence of such an injection implies that $|\mathcal{A}| \geq 2^{n-1}$, as claimed. □

## 4.1 Proof of Lemma 4.2

We begin with a simple property of monotone functions. For a formula $\Phi$ let $U(\Phi)$ denote the set of units in $\Phi$. The following proposition will be useful because it implies that for monotone formulas, setting units cannot create additional units.

PROPOSITION 4.8. *If $\Phi$ is a monotone function and $W$ is a variable in $\Phi$, then $U(\Phi[W{=}0]) \subseteq U(\Phi)$.*

Let $\mathcal{F} = (V, E)$ be an FBDD for a monotone formula $\Phi$, where $V$ and $E$, respectively, denote the nodes and edges of $\mathcal{F}$. For every edge $e = (u, v) \in E$, define $U(e) = U(\Phi_v) - U(\Phi_u)$. Observe that by Proposition 4.8, any edge $e$ for which $U(e)$ is non-empty must be labeled 1 in $\mathcal{F}$.

Fix some canonical ordering $\pi$ on the variables of $\Phi$. Define the following transformation on $\mathcal{F}$ to produce an FBDD $\mathcal{F}'$ for $\Phi$ that follows the unit rule: The set of nodes $V'$ of $\mathcal{F}'$ is given by:

$$V' = V \cup \{(e, i) \mid e = (u, v) \in E, u \in V,\ 1 \leq i \leq |U(e)|\}$$

The other details of $\mathcal{F}'$ are given as follows:

- For $e = (u, v) \in E$, the new vertices $(e, 1), \ldots, (e, |U(e)|)$ will appear in sequence on a path from $u$ to $v$ that replaces the edge $e$. (If $U(e)$ is empty then the original edge $e$ remains.)

- Edge $(u,(e,1))$ in $\mathcal{F}'$ will have label 1, which is the label that $e$ has in $\mathcal{F}$.
- The variable labeling each new vertex $(e,i)$ in $V'$ will be the $i$-th element of $U(e)$ under the ordering $\pi$; we denote this variable by $Z_{e,i}$.
- The 1-edge out of each new vertex $(e,i)$ will lead to the 1-sink. The 0-edge will lead to the next vertex on the newly created path.
- For a vertex $w \in V$ labeled by a variable $W$, if $W$ appears in $U(e)$ for any edge $e=(u,v)$ such that there is a path in $\mathcal{F}$ from $v$ to $w$ then the node $w$ becomes a no-op node in $\mathcal{F}'$, namely its labeling variable $W$ is removed, its 1-outedge is removed, and its 0-outedge is retained with no label. Otherwise, $w$ keeps the variable label $W$ as in $\mathcal{F}$ and its outedges remain the same in $\mathcal{F}'$.

The size bound required for Lemma 4.2 is immediate by construction since the degree of a variable upper-bounds the number of new units that setting it can create. However, in order for this construction to be well-defined we need to ensure that the conversion to no-op nodes does not conflict with the conversion of edges to paths of units.

Proposition 4.9. *If the variable $W$ labeling $w$ is in $U(e)$ for some edge $e=(u,v)$ for which there is a path from $v$ to $w$, then the outedges $e'$ of $w$ have $U(e')=\emptyset$.*

Proof. The assumption implies that $W$ is a unit of some $\Phi_v$. Therefore $\Phi_v = W \vee \Phi'_v$ for some $\Phi'_v$. Since $\mathcal{F}$ is an FBDD and $W$ labels $w$, $W$ is not set on the path from $v$ to $w$, hence $\Phi_w = W \vee \Phi''$ for some formula $\Phi''$. A 0-outedge $e_0$ from $w$ always has $U(e_0)=\emptyset$ and the 1-outedge $e_1=(w,w')$ of $w$ sets $W$ to 1 and hence $\Phi_{w'}=1$, which implies that $U(e_1)$ is also empty. $\square$

The following simple proposition is useful in reasoning about the correctness of our construction.

Proposition 4.10. *If there is a path from $u$ to $v$ in $\mathcal{F}$ and $X \in U(\Phi_u)$ then either $X \in U(\Phi_v)$, or $\Phi_v = 1$, or $X$ is queried on the path from $u$ to $v$ and hence $\Phi_v$ does not depend on $X$.*

Proof. $X \in U(\Phi_u)$ implies that $\Phi_u = X \vee F$ for some monotone formula $F$. If $X$ is set on the path from $u$ to $v$ then $\Phi_v$ does not depend on $X$; otherwise $\Phi_v = X \vee F'$ for some monotone formula $F'$ and either $X$ is a prime implicant or $F'$ is the constant 1 and hence $\Phi_v = 1$. $\square$

Taken together with the size bound for our construction, the following lemma immediately implies Lemma 4.2.

Lemma 4.11. *Let $\Phi$ be monotone and computed by FBDD $\mathcal{F}$. Then $\mathcal{F}'$ is an FBDD for $\Phi$ that follows the unit rule.*

Proof. We first show that $\mathcal{F}'$ is an FBDD, namely, every root-leaf path $P$ in $\mathcal{F}'$ queries each variable at most once. $P$ contains old nodes $u \in V$ and new nodes $(e,i)$. Suppose that a variable $X$ is tested twice along a path. Clearly the two tests cannot be done by old nodes since $\mathcal{F}$ is an FBDD. It cannot be tested by an old node $u$ and later by a new node $(e,i)$, because once tested by $u$, for any descendent node $v$, the formula $\Phi_v$ no longer depends on $X$, hence $X \notin U(e)$. It cannot be first tested by a new node $(e,i)$ and then later by an old node $u$ since the test at the old node would have been removed and converted to a no-op by the last item in the construction of $\mathcal{F}'$. Finally, suppose that the two tests are done by two new nodes $(e_1,i)$, and $(e_2,j)$ on $P$, where we write $e_1=(u_1,v_1)$ and $e_2=(u_2,v_2)$. then we must have $X \in U(v_1)$ and $X \notin U(u_2)$ where there is a path from $v_1$ to $u_2$ in $\mathcal{F}$. By Proposition 4.10, this implies that $\Phi_{u_2}$ does not depend on $X$ which contradicts the requirement that $X \in U(v_2)$ since $v_2$ is a child of $u_2$.

By construction, $\mathcal{F}'$ obviously follows the unit rule. It remains to prove that $\mathcal{F}'$ computes $\Phi$. We show something slightly stronger: For any function $F$, define $F^-$ to be $F[U(F)=0]$, in which all variables in $U(F)$ are set to 0. We claim by induction that for all nodes of $v \in V$, if $\theta'$ labels a path in $\mathcal{F}'$ from the root to $v$, then $\Psi[\theta'] = \Phi_v^-$. and $\theta' = \theta \cup \{U(\Phi_v)=0\}$ for some $\theta$ that labels a path in $\mathcal{F}$ from the root to $v$. This trivially is true for the root. If it is true for the output nodes, then $\mathcal{F}'$ correctly computes $\Psi$ since constant functions have no units. Let $v \in V$ and suppose that this is true for all vertices $u$ such that there is some path $\theta'$ from the root to $v$ in $\mathcal{F}'$ for which $u$ is the last vertex in $V$ on $\theta'$. By the construction, for each such $u$ there must be an edge $e=(u,v) \in E$. Suppose that the variable tested at $u$ in $\mathcal{F}$ is $W$. We have 3 cases: If $e=(u,v) \in E$ is a 1-edge then $\Phi_v = \Phi_u[W{=}1]$. Every path $\theta'$ from the root to $v$ through $u$ is of the form $\theta' = \theta \cup \{W{=}1\} \cup \{U(e){=}0\}$ for some $\theta$ that labels a path from the root to $u$ in $\mathcal{F}'$. (This is true even if $U(e)$ is empty.) By induction, $\Psi[\theta] = \Phi_u^- = \Phi_u[U(\Phi_u){=}0]$ and by definition $U(\Phi_v) = U(\Phi_u) \cup U(e)$ so $\Psi[\theta'] = \Phi_u[W{=}1 \cup \{U(\Phi_v){=}0\}] = \Phi_v[U(\Phi_v){=}0]$ as required. If $e=(u,v) \in E$ is a 0-edge of $\mathcal{F}$ then $\Phi_v = \Phi_u[W{=}0]$. If $u$ became a no-op vertex in $\mathcal{F}'$ then there was some ancestor $w$ of $u$ at which $W$ became a unit of $\Phi_w$. Since $\mathcal{F}$ is an FBDD, it does not query $W$ between that ancestor and $u$. By Proposition 4.10, either $W \in U(\Phi_u)$ or $\Phi_u = 1$. In the latter subcase, $\Phi_v = \Phi_u = 1$ and the correctness for $u$ implies that for $v$. In the former case, $U(\Phi_u) = U(\Phi_v) \cup \{W\}$ and $\Phi_u^- = \Phi_v^-$ and again the correctness for $\Phi_u^-$ implies that for $\Phi_v^-$. In the case that $u$ does not become a no-op vertex, $W$ is not a unit of $\Phi_u$ so $U(\Phi_u) = U(\Phi_v)$ and the fact that all paths to $u$ yield $\Phi_u[U(\Phi_u){=}0] = \Phi_u[U(\Phi_v){=}0]$ for all paths to $v$ that pass through $u$ as the previous vertex in $V$, The last edge to $v$ adds the extra $W{=}0$ constraint. Adding this constraint to $\Phi_u[U(\Phi_v){=}0]$ yields $\Phi_v[U(\Phi_v){=}0] = \Phi_v^-$ as required. Therefore the statements holds for all possible paths from the root to $v$. $\square$

## 5. LOWER BOUNDS FOR BOOLEAN COMBINATIONS OVER $\mathbf{H_k}$

In this section we prove Theorem 3.3. Throughout this section we fix $f(\mathbf{X}) = f(X_0,\ldots,X_k)$, a Boolean function that depends on all variables $\mathbf{X}$, and a domain size $n > 0$. To prove Theorem 3.3, we first prove that any FBDD for the lineage of the query $Q = f(h_{k0},\ldots,h_{kk})$ can be converted into a multi-output FBDD for all of $\mathbf{H_k} = (H_{k0}, H_{k1}, \ldots, H_{kk})$ with at most an $O(k2^k n^3)$ increase in size. The proof is constructive. Theorem 3.3 then follows immediately using Theorem 3.1 since any FBDD for $\mathbf{H_k}$ yields an FBDD for $H_k$ of the same size.

Recall that $H_{k\ell}$ denotes the lineage of $h_{k\ell}$ and let $\Psi = f(\mathbf{H_k}) = f(H_{k0},\ldots,H_{kk})$ be the lineage of $Q$.

If $\mathcal{F}$ is an FBDD for $\Psi = f(\mathbf{H_k})$, we will let $\Phi_u$ denote the Boolean function computed at the node $u$; thus $\Psi = \Phi_r$, where $r$ is the root node of $\mathcal{F}$. By the correctness of $\mathcal{F}$, all paths $P$ leading to $u$ have the property that $\Psi[P] = \Phi_u$.

In order to produce the FBDD $\mathcal{F}'$ for $\mathbf{H_k}$ from $\mathcal{F}$ computing $\Psi = f(\mathbf{H_k})$, we would like to ensure that every internal node $v$ of $\mathcal{F}'$ has the property that all paths $P$ leading to $v$ not only are consistent with the same residual function $\Phi_v = \Psi[P]$, but they also all agree on the residual values of $H_{k\ell}(v) \overset{\text{def}}{=} H_{k\ell}[P]$ for all $\ell$. Since we will not easily be able to characterize its paths we find it convenient to define this property not only with respect to paths of $\mathcal{F}'$ but for formulas $\Phi_v$ with respect to arbitrary partial assignments $\theta$. We use the term *transparent* to describe the property that the value of $\Phi_v$ automatically also reveals the values for all $H_{k\ell}(v)$. Call a formula $\Phi$ a *restriction* of $\Psi$ if $\Phi = \Psi[\theta]$ for some partial assignment $\theta$.

Definition 5.1. *Fix $\Psi = f(H_{k0}, \dots, H_{kk})$. A formula $\Phi$ that is a restriction of $\Psi$ is called* transparent *if there exist $k+1$ formulas $\varphi_0, \dots, \varphi_k$ such that, for every partial assignment $\theta$, if $\Phi = \Psi[\theta]$, then $H_{k0}[\theta] = \varphi_0$, …, $H_{kk}[\theta] = \varphi_k$. We say that $\Phi$* defines *$\varphi_0, \dots, \varphi_k$.*

In other words, assuming that $\Phi$ is a restriction of $\Psi$, $\Phi = \Psi[\theta]$ for some partial assignment $\theta$, then $\Phi$ is transparent if the formulas $H_{k0}[\theta], \dots, H_{kk}[\theta]$ are uniquely defined; i.e., are independent of $\theta$. Equivalently, for any two assignments $\theta, \theta'$, if $\Psi[\theta] = \Psi[\theta'] = \Phi$, then for all $0 \le \ell \le k$, $H_{k\ell}[\theta] = H_{k\ell}[\theta']$.

Example 5.2. *Let $k = 3$ and $f = X_0 \vee X_1 \vee X_2 \vee X_3$. Given a domain size $n > 0$, the formula $\Psi$ is:*

$$\Psi = \bigvee_{i,j} R(i)S_1(i,j) \vee \bigvee_{i,j} S_1(i,j)S_2(i,j)$$
$$\bigvee_{i,j} S_2(i,j)S_3(i,j) \vee \bigvee_{i,j} S_3(i,j)T(j)$$

*$H_{30}, \dots, H_{33}$ denote each of the four disjunctions above. Let $\Phi = R(3)S_1(3,7) \vee S_1(3,7)S_2(3,7)$. There are many partial substitutions $\theta$ for which $\Phi = \Psi[\theta]$: for example, $\theta$ may set to 0 all variables with index $\neq (3,7)$, and also set $S_3(3,7) = T(7) = 0$; or, it could set $S_3(3,7) = 0$, $T(7) = 1$; there are many more choices for variables with index $\neq (3,7)$. However, one can check that, for any $\theta$ such that $\Phi = \Psi[\theta]$, we have:*

$$H_{30}[\theta] = R(3)S_1(3,7) \qquad H_{31}[\theta] = S_1(3,7)S_2(3,7)$$
$$H_{32}[\theta] = 0 \qquad H_{33}[\theta] = 0$$

*Therefore, $\Phi$ is* transparent. *On the other hand, consider $\Phi' = S_1(3,7)$. This formula is no longer transparent, because it can be obtained by extending any $\theta$ that produces $\Phi$ with either $R(3) = 0, S_2(3,7) = 1$, or $R(3) = 1, S_2(3,7) = 0$, or $R(3) = S_2(3,7) = 1$, and these lead to different residual formulas for $H_{30}$ and $H_{31}$ (namely $0$ and $S_1(3,7)$, or $S_1(3,7)$ and $0$, or $S_1(3,7)$ and $S_1(3,7)$).*

In order to convert an FBDD $\mathcal{F}$ for $\Psi = f(\mathbf{H_k})$ into a multi-output FBDD for $\mathbf{H_k} = (H_{k0}, \dots, H_{kk})$, we will try to modify it so that the formulas defined by the restrictions reaching its nodes become transparent without much of an increase in the FBDD size. To do this we will add new intermediate nodes at which the formulas may not be transparent but we will be able to reason about its computations based on the nodes where the formulas are transparent.

Observe that if we know that $\Phi_v = \Psi[\theta]$ is transparent and we have a small multi-output FBDD $\mathcal{F}_\theta$ for $\mathbf{H_k}[\theta]$ then we can simply append that small FBDD at node $v$ to finish the job and ignore what the original FBDD did below $v$. Intuitively, the reason that $\mathbf{H_k}$ and $H_k$ might not have such small FBDDs is the tension between the $R(i)S_1(i,j)$ terms, which give a preference for reading entries in row-major order and the $S_k(i,j)T(j)$ terms, which suggest column-major order, together with the intermediate $S_\ell(i,j)S_{\ell+1}(i,j)$ terms that link these two conflicting preferences. If all of those links are broken, then it turns out that there is no conflict in the variable order and the difficulty disappears. This motivates the following definition which we will use to make this intuitive idea precise.

Definition 5.3. *Let $\theta$ be a partial assignment to $Var(\mathbf{H_k})$.*

- *A* transversal *in $\theta$ is a pair of indices $(i,j)$ such that $R(i)S_1(i,j)$ is a prime implicant of $H_{k0}[\theta]$, $S_k(i,j)T(j)$ is a prime implicant of $H_{kk}[\theta]$, and $S_\ell(i,j)S_{\ell+1}(i,j)$ is a prime implicant of $H_{k\ell}[\theta]$ for all $\ell \in [k-1]$.*
- *Call two pairs of indices (or transversals) $(i_1,j_1),(i_2,j_2)$* independent *if $i_1 \neq i_2$ and $j_1 \neq j_2$.*
- *A Boolean formula is called* transversal-free *if there* exists *a $\theta$ such that $\Phi = \Psi[\theta]$ and $\theta$ has no transversals.*

We now see that assignments without transversals, or even those with few independent transversals, yield small FBDDs.

Lemma 5.4. *Let $\theta$ be a partial assignment to $Var(\mathbf{H_k})$. If $\theta$ has at most $t$ independent transversals then there exists a multi-output FBDD for $(H_{k0}[\theta], \dots, H_{kk}[\theta])$ of size $O(k2^{k+t}n^2)$.*

Proof. We first show that if $t = 0$ ($\theta$ has no transversals) then there exists a small OBDD that computes $\mathbf{H_k}[\theta]$.

Let $G_\theta$ be the following undirected graph. The nodes are the variables $Var(\mathbf{H_k})$, and the edges are pairs of variables $(Z, Z')$ such that $ZZ'$ is a 2-prime implicant in $H_{k\ell}[\theta]$ for some $\ell$. Since $\theta$ has no transversals, all nodes $R(i)$ are disconnected from all nodes $T(j)$. In particular, there exists a partition $Var(\mathbf{H_k}) = \mathbf{Z}' \cup \mathbf{Z}''$ such that all $R(i)$'s are in $\mathbf{Z}'$, all $T(j)$'s are in $\mathbf{Z}''$ and every $H_{k\ell}[\theta]$ can written as $\varphi'_\ell \vee \varphi''_\ell$ where $Var(\varphi'_\ell) \subset \mathbf{Z}'$ and $Var(\varphi''_\ell) \subset \mathbf{Z}''$; in particular, $\varphi''_0 = \varphi'_k = 0$.

Define *row-major order* of the variables in the set $Var(\mathbf{H_k}) - \{\mathrm{T}(1), \cdots, T(n)\}$ by:

$$R(1), S_1(1,1), \dots, S_k(1,1), S_1(1,2) \dots, S_k(1,n),$$
$$R(2), S_1(2,1), \dots, S_k(2,1), S_1(2,2) \dots, S_k(2,n),$$
$$\dots$$
$$R(n), S_1(n,1), \dots, S_k(n,1), S_1(n,2) \dots, S_k(n,n)$$

Let $\pi'$ be the restriction of the row-major order to the variables in $\mathbf{Z}'$. Similarly, let $\pi''$ be the restriction to $\mathbf{Z}''$ of the corresponding column-major order of the variables that omits the $R(i)$'s, and places the $T(j)$ before all variables $S_\ell(i,j)$. We build a multi-output OBDD using the order $\pi = (\pi', \pi'')$ for $(H_{k0}, \dots, H_{kk})$. In the first part using order $\pi'$ it will compute each $\varphi'_\ell$ term in parallel in width

$O(2^k)$ and in the second part it will continue by including the additional terms from $\varphi''_\ell$ using order $\pi''$. Observe that, except for the $R(i)S_1(i,j)$ terms, each of the variables in the 2-prime implicants in $\varphi'_\ell$ appear consecutively in $\pi'$. Each level of the OBDD will have at most $2^{k+3}$ nodes, one for each tuple consisting of a vector of values of the partially computed values for the $k+1$ functions $\varphi'_\ell$, remembered value of $R(i)$, and remembered value of the immediately preceding queried variable. In the part using order $\pi''$, the remembered value of $T(j)$ is used instead of the remembered value of $R(i)$. The size of $\mathcal{F}'$ is $O(k2^kn^2)$ since there are $kn^2+2n$ variables in total in $Var(\mathbf{H_k})$.

For general $t$, let $I$ and $J$ be the sets of rows and columns, respectively, of the transversals $(i,j)$ in $\theta$. Since $\theta$ has at most $t$ independent transversals, the smaller of $I$ and $J$ has size at most $t$. Suppose that this smaller set is $I$; the case when $J$ is smaller is analogous. In this case, every transversal $(i,j)$ of $\theta$ has $i \in I$. Notice that if we set all $R(i)$ variables with $i \in I$ in an assignment $\theta'$ then the assignment $\theta \cup \theta'$ has no transversals, and thus, by the above construction, $\mathbf{H_k}[\theta']$ can be computed efficiently by a multi-output OBDD. Therefore, construct the FBDD which first exhaustively tests all possible settings of these at most $t$ variables in a complete tree of depth $t$, then at each leaf node of the tree, attaches the OBDD constructed above. $\square$

A nice property of a single transversal for $\theta$ is that its existence ensures that each $H_{k\ell}$ is a non-trivial function of its remaining inputs; more transversals will in fact ensure that less about each $H_{k\ell}$ disappears. We will see the following: if there are at least some small number of independent transversals for $\theta$ (three suffice), then we can use the fact that $f$ depends on all inputs to ensure that $\Psi[\theta] = f(\mathbf{H_k})[\theta]$ will be transparent provided one additional condition holds: there is no variable which we can set to kill off all transversals in $\theta$ at once.

If we didn't have this additional condition, then the construction of $\mathcal{F}'$ for $\mathbf{H_k}$ would be simple: We would just use Lemma 5.4 at all nodes $v$ of $\mathcal{F}$ at which all assignments $\theta$ for which $\Phi_v = \Psi[\theta]$ do not have enough transversals to ensure transparency of $\Phi_v$.

Failure of the additional condition is somewhat reminiscent of the situation with setting units in Section 4: This failure means that there is some variable we can set to kill off all transversals in $\theta$ at once, which by Lemma 5.4 means that along the branch corresponding to that setting one can get an easy computation of $\mathbf{H_k}$ (not quite as simple as fixing the value of the formula to 1 by setting units as in Section 4, but still easy). It is not hard to see, and implied by the proposition below, which is easy to verify, that the only way to kill off multiple independent transversals at once is to set such a variable to 1. By analogy we call such variables $\mathbf{H_k}$*-units*.

PROPOSITION 5.5. *Let $\Phi = \Psi[\theta]$ for some $\theta$ with $t$ independent transversals and $\theta' = \theta \cup \{W{=}b\}$ for $b \in \{0,1\}$. The number of independent transversals in $\theta'$ is in $\{t-1,t\}$ if $b=0$ and is in $\{0,t-1,t\}$ if $b=1$.*

DEFINITION 5.6. *We say that a variable $Z$ is an* $\mathbf{H_k}$-unit *for the formula $\Phi$ if $\Phi[Z=1]$ is transversal-free but $\Phi$ is not. We let $U_k(\Phi)$ denote the set of $\mathbf{H_k}$-units of $\Phi$, and we say that $\Phi$ is* $\mathbf{H_k}$-unit-free *if $U_k(\Phi) = \emptyset$.*

The following lemma makes our intuitive claim precise; the proof of Lemma 5.7 appears in the full version of the paper [4] due to space constraints.

LEMMA 5.7. *Let $\Psi = f(\mathbf{H_k})$ where $f$ depends on all its inputs. Suppose that there exists a $\theta$ with at least 3 independent transversals such that $\Psi[\theta] = \Phi$. If $\Phi$ is $\mathbf{H_k}$-unit-free then $\Phi$ is transparent.*

We will still need to deal with the situation when $\Phi$ has any $\mathbf{H_k}$-units along with multiple independent transversals. Our strategy will be simple: whenever we encounter an edge in $\mathcal{F}$ on which an $\mathbf{H_k}$-unit is created (possibly more than one at once) and the resulting formula has sufficiently many transversals then, just as with the unit rule, we immediately test these $\mathbf{H_k}$-units, one at a time, each one after the previous one has been set to 0 (since the branch where it is set to 1 has an easy computation remaining).

In order to analyze this strategy properly, it will be useful to understand how $\mathbf{H_k}$-units can arise. Observe, that if $\Phi = \Psi[\theta]$ and $Z$ is a unit for some $H_{k\ell}[\theta]$, for $0 \le \ell \le k$, then $Z$ is an $\mathbf{H_k}$-unit for $\Psi[\theta]$, because setting $Z=1$ we ensure that $H_{k\ell}[\theta \cup \{Z=1\}] = 1$, wiping out all transversals. The following lemma shows a converse of this statement under the assumption that $\theta$ has at least 4 independent transversals; the proof of Lemma 5.8 appears in the full version of the paper [4].

LEMMA 5.8. *Let $\Psi = f(\mathbf{H_k})$ where $f$ depends on all its inputs. If $\Phi = \Psi[\theta]$ for some partial assignment $\theta$ that has at least 4 independent transversals, then $U_k(\Phi) = \bigcup_{\ell \in \{0,\ldots,k\}} U(H_{k\ell}[\theta])$.*

Since a transversal $(i,j)$ requires that all elements of $\mathbf{H_k}$ have 2-prime implicants rather than units on the terms involving $(i,j)$, Lemma 5.8 immediately implies the following:

COROLLARY 5.9. *If $\Phi = \Psi[\theta]$ for some partial assignment $\theta$, then no $\mathbf{H_k}$-unit of $\Phi$ is in the prime implicants indexed by any transversal of $\theta$.*

Since the formulas in $\mathbf{H_k}$ are monotone, by Lemma 5.8 and Proposition 4.8, units are created by setting a variable to 1. Hence, if $\Phi$ has at least 4 independent transversals, then setting all $\mathbf{H_k}$-units in $\Phi$ to 0 in turn yields a formula that still has at least 4 independent transversals (by Corollary 5.9) and is $\mathbf{H_k}$-unit-free (by Lemma 5.8), and hence transparent (by Lemma 5.7 and Proposition 4.8).

We now describe the procedure for building a multi-output FBDD $\mathcal{F}'$ computing $\mathbf{H_k}$: Start with the FBDD $\mathcal{F}$ for $\Psi$ and let $V$ and $E$ be, respectively, the vertices and edges of $\mathcal{F}$. Let $V_4 \subseteq V$ be the set of nodes $v \in V$ such that $\Phi_v = \Psi[\theta]$ for some assignment $\theta$ that has at least 4 independent transversals. By Proposition 5.5, $V_4$ is closed under predecessors (ancestors) in $\mathcal{F}$; let $E_4$ be the set of edges in $\mathcal{F}$ whose endpoints are both in $V_4$. The following is immediate from Proposition 5.5 and the definition of $V_4$.

PROPOSITION 5.10. *If $v \in V_4$ but some child of $v$ is not in $V_4$ then either or both of the following hold: (i) there is an assignment $\theta$ with precisely 4 independent transversals such that $\Phi_v = \Psi[\theta]$, or (ii) the variable $Z$ tested at $v$ is in $U_k(\Phi_v)$ and the 0-child of $v$ is in $V_4$.*

We will apply a similar construction to that of Section 4.1 to the subgraph of $\mathcal{F}$ on $V_4$. For $e = (u,v)$, define $U_k(e) =$

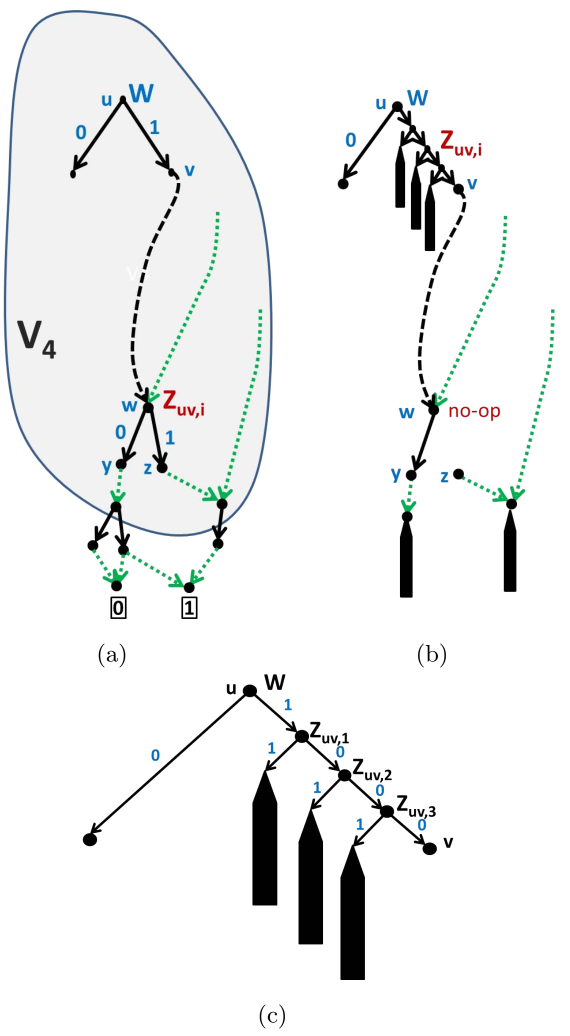

Figure 2: Given an FBDD $\mathcal{F}$ for $\Psi = f(\mathbf{H_k})$ in (a), apply the conversion to produce $\mathcal{F}'$ for $\mathbf{H_k}$ as in (b), with detail for unit propagation in (c) in case that setting $W = 1$ produces new $\mathbf{H_k}$-units.

$U_k(\Phi_v) - U_k(\Phi_u)$ to be the set of new $\mathbf{H_k}$-units created along edge $e$. There are two differences from the argument in Section 4.1: (1) we will only apply the construction to edges in $E_4$ and will build the rest of $\mathcal{F}'$ independently of $\mathcal{F}$, and (2) unlike setting ordinary units to 1, in which the corresponding FBDD edges simply point to the 1-sink, each setting of an $\mathbf{H_k}$-unit to 1 only guarantees that the resulting formula is transversal-free; moreover the transversal-free formulas resulting from different settings may be different. The details are as follows (see Figure 2).

- For every $e = (u, v) \in E_4$ such that $U_k(e)$ is non-empty (and hence the 0-child of $u$ is also in $V_4$), add new vertices $(e, 1), \ldots, (e, |U_k(e)|)$ and replace $e$ with a path from $u$ to $v$ having the new vertices in order as internal vertices.
- Edge $(u, (e, 1))$ in $\mathcal{F}'$ will have label 1, which is the label that $e$ has in $\mathcal{F}$; denote the variable tested at $u$ by $W$.
- The variable labeling each new vertex $(e, i)$ will be the $i$-th element of $U_k(e)$ under some fixed ordering of variables; we denote this variable by $Z_{e,i}$.
- The 0-edge out of each new vertex $(e, i)$ will lead to the next vertex on the newly created path. However, unlike the simple situation with ordinary units, the 1-edge out of each new vertex $(e, i)$ will lead to a distinct new node $(u, i)$ of $\mathcal{F}'$. Since $(u, v) \in E_4$ there is some partial assignment $\theta$ such that $\Phi_u = \Psi[\theta]$, $\Phi_v = \Psi[\theta, W{=}1]$, and $\theta \cup \{W{=}1\}$ has at least 4 transversals; for definiteness we will pick the lexicographically first such assignment. Define the partial assignment
$$\begin{aligned}\theta(u,i) = \theta &\cup \{W{=}1\} \cup \{U_k(\Phi_u){=}0\} \\ &\cup \{Z_{e,1}{=}0, \ldots, Z_{e,i-1}{=}0, Z_{e,i}{=}1\},\end{aligned}$$
to be the assignment that sets all $\mathbf{H_k}$-units in $\Phi_u$ to 0 along with the first $i-1$ of the $\mathbf{H_k}$-units created by setting $W$ to 1. The sub-dag of $\mathcal{F}'$ rooted at $(u, i)$ will be the size $O(k2^k n^2)$ FBDD for $\mathbf{H_k}[\theta(u, i)]$ constructed in Lemma 5.4.
- For any node $w \in V_4$, whose 0-child is in $V_4$, such that $w$ is labeled by a variable $W$ that was an $\mathbf{H_k}$-unit of $\Phi_v$ for some ancestor $v$ of $w$, convert $w$ to a no-op node pointing to its 0-child; that is, remove its variable label and its 1-outedge and retain its 0-outedge with its labeling removed.
- For any node $v \in V_4$ with a child that is not in $V_4$ and to which the previous condition did not apply, let $\theta$ be a partial assignment such that $\Phi_v = \Psi[\theta]$ and $\theta$ has precisely 4 independent transversals, as guaranteed by Proposition 5.10, make $v$ the root of the size $O(k2^k n^2)$ FBDD for $\mathbf{H_k}[\theta']$ constructed in Lemma 5.4 where $\theta' = \theta \cup \{U_k(\Phi_v) = 0\}$.
- All other labeled edges of $\mathcal{F}$ between nodes of $E_4$ are included in $\mathcal{F}'$.

The fact that this is well-defined follows similarly to Proposition 4.9.

Lemma 5.11. *$\mathcal{F}'$ as constructed above is a multi-output FBDD computing $\mathbf{H_k}$ that has size at most $O(k2^k n^3)$ times the size of $\mathcal{F}$.*

Proof. We first analyze the size of $\mathcal{F}'$: As in the analysis for computing $H_k$, some nodes $u$ have one added unit-setting path of length at most $n$ and each node on the path of at the extremities of $V_4$ has a new added FBDD of size $O(k2^k n^2)$ yielding only $O(k2^k n^3)$ new nodes per node of $\mathcal{F}$. Also, the fact that $\mathcal{F}'$ is an FBDD follows similarly to the proof in Lemma 4.11.

If $\Phi_v$ is the function computed in $\mathcal{F}$ at node $v$ for all $v \in V_4$, then we show by induction that for every partial assignment $\theta'$ reaching $v$ in $\mathcal{F}'$, $\Psi[\theta'] = \Phi_v[U_k(\Phi_v){=}0]$ and $\theta' = \theta \cup \{U_k(\Phi_v){=}0\}$ for some partial assignment $\theta$ such that $\Phi_v = \Psi[\theta]$. It is trivially true of the root. The argument is similar to that for Lemma 4.11.

We now see why this is enough. Since $v \in V_4$, $\Phi_v[U_k(\Phi_v){=}0]$ is $\mathbf{H_k}$-unit-free and has at least 4 transversals, and so it is transparent by Lemma 5.7. It remains to observe that (i) each multi-output FBDD attached directly to any node $v \in V_4$ used a restriction $\theta$ of $\Psi$ that would lead to that node in $\mathcal{F}'$, which, because $\Psi[\theta]$ is transparent, implies that its leaves correctly compute the values of $\mathbf{H_k}$, and (ii) the same holds for the restriction leading to node $(u, i)$ with parent $(e, i)$, namely, the restriction used to build the multi-output FBDD consists of a restriction $\theta$ that in $\mathcal{F}'$ would reach node $u \in V_4$ and for which $\Psi[\theta]$ is transparent, together with the assignment $\{W{=}1\} \cup \{Z_{e,1}{=}0, \ldots, Z_{e,i-1}{=}0, Z_{e,i}{=}1\}$ which follows the unique path from $u$ to $(u, i)$. Again this implies that its leaves correctly compute the values of $\mathbf{H_k}$. $\square$

## 6. DISCUSSION

In this paper we proved exponential separations between lifted model counting using extensional query evaluation and state-of-the-art propositional methods for exact model counting. Our results were obtained by proving exponential lower bounds on the sizes of the decision-DNNF representations implied by those proposition methods even for queries that can be evaluated in polynomial time. We also introduced DLDDs, which generalize decision-DNNFs while retaining their good algorithmic properties for model counting. Though our query lower bounds apply equally to their DLDD representations, DLDDs may prove to be better than decision-DNNFs in other scenarios.

In light of our lower bounds, it would be interesting to prove a dichotomy, classifying queries into those for which any decision-DNNF-based model counting algorithm takes exponential time and those for which such algorithms run in polynomial time. In this paper we showed such a dichotomy for a very restricted class of queries. A dichotomy for general model counting is known for the broader query class UCQ [6] which classifies queries as either #P-hard or solvable in polynomial time. Our separation results show that this same dichotomy does not extend to decision-DNNF-based algorithms; is there some other general dichotomy that can be shown for this class of algorithms?

## 7. REFERENCES

[1] F. Bacchus, S. Dalmao, and T. Pitassi. Algorithms and complexity results for #sat and bayesian inference. In *FOCS*, pages 340–351, 2003.

[2] R. J. Bayardo, Jr., and J. D. Pehoushek. Counting models using connected components. In *AAAI*, pages 157–162, 2000.

[3] P. Beame, J. Li, S. Roy, and D. Suciu. Lower bounds for exact model counting and applications in probabilistic databases. In *UAI*, 2013.

[4] P. Beame, J. Li, S. Roy, and D. Suciu. Model counting of query expressions: Limitations of propositional methods. *CoRR*, abs/1312.4125, 2013.

[5] B. Bollig and I. Wegener. A very simple function that requires exponential size read-once branching programs. *Inf. Process. Lett.*, 66(2):53–57, Apr. 1998.

[6] N. N. Dalvi and D. Suciu. The dichotomy of probabilistic inference for unions of conjunctive queries. *J. ACM*, 59(6):30, 2012.

[7] A. Darwiche. Decomposable negation normal form. *J. ACM*, 48(4):608–647, 2001.

[8] A. Darwiche. On the tractable counting of theory models and its application to truth maintenance and belief revision. *Journal of Applied Non-Classical Logics*, 11(1-2):11–34, 2001.

[9] A. Darwiche and P. Marquis. A knowledge compilation map. *J. Artif. Int. Res.*, 17(1):229–264, Sept. 2002.

[10] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962.

[11] M. Davis and H. Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960.

[12] C. P. Gomes, A. Sabharwal, and B. Selman. Model counting. In *Handbook of Satisfiability*, pages 633–654. IOS Press, 2009.

[13] J. Huang and A. Darwiche. Dpll with a trace: From sat to knowledge compilation. In *IJCAI*, pages 156–162, 2005.

[14] J. Huang and A. Darwiche. The language of search. *JAIR*, 29:191–219, 2007.

[15] M. Jaeger and G. V. den Broeck. Liftability of probabilistic inference: Upper and lower bounds. In *Proceedings of the 2nd International Workshop on Statistical Relational AI*, 2012.

[16] A. K. Jha and D. Suciu. Knowledge compilation meets database theory: compiling queries to decision diagrams. In *ICDT*, pages 162–173, 2011.

[17] A. K. Jha and D. Suciu. Knowledge compilation meets database theory: Compiling queries to decision diagrams. *Theory Comput. Syst.*, 52(3):403–440, 2013.

[18] S. M. Majercik and M. L. Littman. Using caching to solve larger probabilistic planning problems. In *AAAI*, pages 954–959, 1998.

[19] C. Muise, S. A. McIlraith, J. C. Beck, and E. I. Hsu. Dsharp: fast d-dnnf compilation with sharpsat. In *Canadian AI*, pages 356–361, 2012.

[20] A. Sabharwal. Symchaff: exploiting symmetry in a structure-aware satisfiability solver. *Constraints*, 14(4):478–505, 2009.

[21] T. Sang, F. Bacchus, P. Beame, H. A. Kautz, and T. Pitassi. Combining component caching and clause learning for effective model counting. In *SAT*, 2004.

[22] R. P. Stanley. *Enumerative Combinatorics.* Cambridge University Press, 1997.

[23] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases.* Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[24] M. Thurley. sharpsat: counting models with advanced component caching and implicit bcp. In *SAT*, pages 424–429, 2006.

[25] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 1979.

[26] I. Wegener. *Branching programs and binary decision diagrams: theory and applications.* SIAM, Philadelphia, PA, USA, 2000.