This is a preprint. The final version of this paper appeared in Journal of Computer and System Sciences, 70(4):435–470, 2005.

Exact learning of DNF formulas using DNF hypotheses ⋆

Lisa Hellerstein [*,1]

Department of Computer and Information Science, Polytechnic University, Brooklyn NY 11201

Vijay Raghavan [2]

Electrical Engineering and Computer Science Department, Vanderbilt University, Nashville TN 37235

Abstract

We show the following:

(a) For any $\epsilon > 0$, $\log^{(3+\epsilon)} n$-term DNF cannot be polynomial-query learned with membership and strongly proper equivalence queries. **(b)** For sufficiently large t, t-term DNF formulas cannot be polynomial-query learned with membership and equivalence queries that use $t^{1+\epsilon}$-term DNF formulas as hypotheses, for some $\epsilon < 1$ **(c)** Read-thrice DNF formulas are not polynomial-query learnable with membership and proper equivalence queries. **(d)** $\log n$-term DNF formulas can be polynomial-query learned with membership and proper equivalence queries. (This complements a result of Bshouty, Goldman, Hancock, and Matar that $\sqrt{\log n}$-term DNF can be so learned in polynomial *time*.)

Versions of (a), (b), and (c) were known previously, but the previous versions applied to polynomial-*time* learning and used complexity theoretic assumptions. In contrast, (a), (b), and (c) apply to polynomial-*query* learning, imply the results for polynomial-time learning, and do not use any complexity-theoretic assumptions.

Key words: Computational Learning Theory, Disjunctive Normal Form, DNF, Boolean Functions, Certificates, Algorithms, Complexity Theory.
1991 MSC: 06E30, 68Q15, 68Q20, 68Q25, 68T05.

1 Introduction

The learnability of DNF formulas is a central problem in computational learning theory, and has been studied in a variety of learning models ([2–10]). Here we present new results on learning DNF formulas using DNF hypotheses. Our results are for the membership and equivalence query model of Angluin [11], sometimes called the *exact* learning model.

1.1 Proper and strongly proper learning

We consider the following question: at what point does the class of k-term DNF formulas cease to be exactly learnable with membership and proper equivalence queries, when the value of k is increased as a function of n?

Previously, Bshouty, Goldman, Hancock, and Matar showed that $O(\sqrt{\log n})$-term DNF can be learned in polynomial time with membership and proper equivalence queries [5]. In fact, their algorithm can easily be modified to use *strongly* proper equivalence queries, meaning that the hypotheses used in the equivalence queries (and the output) have no more terms than the target. On the negative side, Pillaipakkamnatt and Raghavan showed that if a strong complexity-theoretic assumption holds (that there exists some constant $\epsilon > 0$ such that NP is not contained in DTIME(2^{n^ϵ})) then there exists some constant $c > 0$ such that DNF formulas with more than $\log^c n$ terms cannot be learned in polynomial time using membership and strongly proper equivalence queries [9].

We extend both these results.

First, we show that $O(\log n)$-term DNF can be learned with a polynomial *number* of membership and strongly proper equivalence queries. This proves that there is no information-theoretic barrier to proper learning of this class; however, we leave open the question of whether $O(\log n)$-term DNF can be properly learned in polynomial time. [3]

⋆ A preliminary version of this paper appeared in STOC 2002 [1]

* Corresponding author.
Email addresses: hstein@cis.poly.edu (Lisa Hellerstein),
raghavan@vuse.vanderbilt.edu (Vijay Raghavan).

[1] Research partially funded by NSF Grants CCR-9877122 and EIA-9806207

[2] Research partially funded by NSF Grant CCR-9820840

[3] Unfortunately, a direct implementation of our algorithm would need to solve the problem of finding a Boolean function that computes the majority of all $O(\log n)$-term DNF formulas that are consistent with information known up to a certain point; using the technique in [12], it can be shown that this can be done in polyno-

k	**Learnability of k-term DNF**
$O(\sqrt{\log n})$	Poly-time learnable with membership and strongly proper equivalence queries [5]
$O(\log n)$	Poly-query learnable with membership and strongly proper equivalence queries. (not known to be poly-time learnable)
$\Omega(\log^{3+\epsilon} n)$	Not poly-query learnable with membership and strongly proper equivalence queries (for any $\epsilon > 0$)

Fig. 1. Learnability of k-term DNF with increasing k

Second, we show that for any $\epsilon > 0$, $\Omega(\log^{(3+\epsilon)} n)$-term DNF formulas cannot be learned with a polynomial number of membership and strongly proper equivalence queries, and hence cannot be learned in polynomial time in this model.

It is worth emphasizing that these and other such results in this paper are purely information-theoretic and do not depend on any unproven complexity-theoretic assumptions.

We summarize the current state of knowledge in Figure 1.

It remains an open question where in the range between $\sqrt{\log n}$ and $\log^{(3+\epsilon)} n$ the transition from polynomial time learnability to non-learnability occurs, and where in the range between $\log n$ and $\log^{(3+\epsilon)} n$ the transition from polynomial query learnability to non-learnability occurs.

We note that although the first two results in Figure 1 also clearly apply to the model of learning with membership and proper (as opposed to strongly proper) equivalence queries, it does not immediately follow that the same is true for the third result. This is because, in contrast to the case of *strongly* proper equivalence queries, in the case of proper equivalence queries non-learnability of k-term DNF for a particular k does not immediately imply non-learnability

mial time using an oracle for Σ_2^p, but nothing better is currently known.

of larger DNF formulas (because expanding the class of formulas to be learned also expands the set of formulas that can be used as hypotheses in learning). However, we do show that for any k that is $\Omega(\log^{(3+\epsilon)} n)$ and bounded above by a polynomial in n, k-term DNF cannot be learned with a polynomial number of membership and proper equivalence queries, and hence cannot be learned in polynomial time.

1.2 Proper poly-time learning of $\log n$*-term DNF: Partial results*

It is known that polynomial-time learnability with membership and proper equivalence queries implies polynomial-time learnability in the proper PAC + membership query model [11]. Given this fact and the results of the previous section, it is natural to wonder whether $\log n$-term DNF formulas can at least be learned in polynomial time in the proper PAC + membership query model. We show that if this is true, then $\log n$-term DNF formulas can also be learned in polynomial time (with high probability) with membership and proper equivalence queries. This implication is interesting because, in general, polynomial-time learnability in the proper PAC + membership query model does not imply polynomial-time learning with membership and proper equivalence queries (assuming the existence of one-way functions) [13].

We also show that if $\log n$-term DNF is learnable in polynomial-time in the membership and proper equivalence query model, then a restricted version of a classical DNF truth table minimization problem is solvable in polynomial time.

1.3 Improper exact learning of $\log n$*-term DNF*

Although it is an open question whether there exists a *proper* algorithm for learning $\log n$-term DNF in polynomial time, there are many known *improper* algorithms accomplishing this task [14–18,5]. Although some of the improper algorithms use hypotheses that are not DNF formulas [14,16,17], others use hypotheses that are DNF formulas of length greater than $\log n$ [15,18,5]. The algorithm of Blum and Rudich [18], improved by Bshouty et al. [5], is of the second type. It makes extensive use of universal sets. In Section 3.1 we present a new, simple algorithm that is also based on universal sets. This algorithm is an easy consequence of the fact (proved by Chandra and Markowsky [19]) that a k-term DNF formula has at most 2^k prime implicants. The hypotheses used by the algorithm are DNF formulas with at most n terms.

We do not claim that ours is the first simple algorithm for learning $\log n$-term DNF formulas in polynomial time using DNF hypotheses; we believe that distinction belongs to a divide-and-conquer algorithm of Bshouty [15]. Bshouty's algorithm does not use universal sets, and is very different from ours. Nevertheless, it can be shown that the DNF hypotheses used by his algorithm, like those used in ours, have at most n terms [20].

To our knowledge, there is no improper algorithm that achieves poly-time learnability of DNF with k terms, for any $k = \omega(\log n)$.

1.4 A gap theorem and its consequence

Hellerstein et al. [21] and Hegedüs [22] showed independently that there are certain combinatorial objects called *certificates* that give a precise characterization of polynomial-query learnability in the exact model of membership and proper equivalence queries. Most of the results in this paper are based on proofs of the existence or non-existence of certificates of a given size.

In Section 3.4, we digress briefly to prove a gap theorem on the size of certificates of non-membership for classes of functions closed under identification minors (defined below). We show that if a class C is closed under identification minors, then either C has constant-size certificates of non-membership, or C does not have certificates of non-membership of size $t(n)$ for any $t(n)$ that is $o(\log n)$. For any constant k, the class of k-term DNF is closed under identification minors, and hence the gap theorem applies to this class.

In Section 3.2, we give an initial upper bound on the size of certificates of non-membership for k-term DNF that depends on both n and k. Given the gap theorem, a natural question is whether this can be improved to show the existence of constant-size (i.e., a function only of k) certificates. In Section 3.5, we prove precisely this. Specifically, we show that k-term DNF has certificates of non-membership of size $O(k^5 2^{4k})$, thus establishing it on the "constant side" of the gap.

A consequence is that for fixed k, k-term DNF can be learned using a number of proper equivalence queries and membership queries *linear* in n, although not necessarily in polynomial time.[4] This result was actually known from

[4] Another consequence is that for any constant k, k-term DNF can be characterized by a finite set of combinatorial objects called *constraints* (as defined by Pippenger [23]) or equivalently by a finite set of *DNF identities* (as defined by Ekin et al. [24]). This follows from the fact that if a class of functions is closed under identification minors, then it has constant-size certificates of non-membership if and only if it can be characterized by a finite set of constraints [25]. A class can be characterized by a finite set of constraints if and only if it can be characterized by a finite set of DNF identities [23].

Berggren's membership and proper equivalence query algorithm for learning k-term DNF [26], which for fixed k uses a number of queries linear in n, and time linear in n. However, Berggren's algorithm gives an upper bound of $O(nk^{k^{k+2}})$ on the number of queries whereas our result reduces the number of queries to $O(nk^4 2^{(1+2\log_2 3)k})$.

1.5 *Exact learning of general DNF*

Can our results be used to say anything about the exact learnability of general DNF formulas using membership and proper equivalence queries?

Proving that DNF cannot be polynomial-query learned is equivalent to proving the following result: there is no polynomial $p(t,n)$ such that DNF formulas can be learned by a polynomial-query membership and equivalence query algorithm that uses $t \cdot p(t,n)$-term DNF formulas as hypotheses, where t is the number of terms in the target formula. In Section 4, we make an admittedly modest start towards proving such a result. We show the result holds for restricted values of $p(t,n)$, namely for $p(t,n) = o(\frac{\sqrt{n}}{\log n})$, and for $p(t,n) = \theta(\frac{t^{1/2}}{\log^{3/2+\epsilon} t})$ for any $\epsilon > 0$. Bshouty and Burroughs [27] showed, based on a result of Umans [28], that if $\Sigma_2^P \neq P^{NP}$, then there exists an $\epsilon < 1$ such that DNF formulas cannot be polynomial-*time* learned by a membership and equivalence query algorithm that uses $t^{1+\epsilon}$-term DNF formulas as hypotheses, where t is the number of terms in the target formula. Our result implies theirs without using any complexity-theoretic assumptions. We use techniques of Umans [28] in our proof.

1.6 *Read-thrice DNF etc.*

In some contexts, it is natural to consider the size of a DNF formula to be the number of occurences of literals, rather than the number of terms. In Section 4.2, we prove negative results for this notion of size and then use those results to prove that the class of read-thrice DNF formulas does not have polynomial-size certificates of non-membership. It follows that read-thrice DNF cannot be learned in polynomial time with membership and proper equivalence queries.

This improves on earlier results. Non-learnability of read-thrice DNF formulas in polynomial time was first shown under the assumption NP≠co-NP in [29]; later the weaker assumption P≠NP was shown to suffice in [9]. Our result here simply removes all complexity theoretic assumptions—even if the polynomial hierarchy collapses, read-thrice DNF would still not be exactly learnable in polynomial time.

1.7 Combinatorial consequences of complexity theoretic assumptions

Our negative results on learning k-term DNF and k-literal occurence DNF combine techniques of Umans [28] with other ad hoc techniques invented here. By directly combining *results* (not techniques) of Umans in [30,28] with an observation of Bshouty mentioned in [21], one can immediately obtain weaker versions of some of our negative results.

Bshouty's observation was this: if $\mathcal{R}$ is an honest [5] representation class, and can be learned using a polynomial number of membership and strongly proper equivalence queries (with no restrictions on computational time), then the problem of deciding if a given representation r in $\mathcal{R}$ has an equivalent representation of size k is in Π_2^P. Now consider, for example, the following DNF term minimization problem: *Given a DNF formula g and a number k, determine whether there exists a DNF formula equivalent to g with at most k terms.* Umans showed that this problem is Σ_2^P complete [28]. Since the class of DNF formulas is honest, it immediately follows that if $\Sigma_2^P \neq \Pi_2^P$, then general DNF formulas cannot be learned with a polynomial number of membership and strongly proper equivalence queries. The characterization of polynomial-query learnability with certificates then gives the following equivalent statement: if $\Sigma_2^P \neq \Pi_2^P$, there are Boolean functions f that need more than m terms to represent them as a DNF formula, and yet a certificate proving that f has this property must have more than a polynomial (in m and n) number of assignments. Similar statements follow by combining a generalization of Bshouty's observation with Umans's results on the Σ_2^P completeness of a DNF occurence-of-literals minimization problem, and of approximation versions of either of these minimization problems. In other words, purely *combinatorial* consequences on superpolynomial lower bounds on certificate sizes ensue from complexity-theoretic considerations.

The negative results in this paper are basically about direct, assumption-free proofs of these consequences; indeed, we were inspired initially to try to prove these negative results in part because we believed that the assumption $\Sigma_2^P \neq \Pi_2^P$ was plausible, and therefore that such superpolynomial lower bounds must exist.

[5] For a class of Boolean formulas with an associated size function, the class is honest if it is well-behaved as defined in Section 2.5.

1.8 Organization of the paper

The rest of the paper is structured as follows. Section 2 provides the background for the remaining sections: essential information about Boolean functions, the definition of the learning models used, propositions on DNF formulas, the relationship of certificates to learning, etc. Section 3 has the gap theorem and the positive results on proper learning of k-term DNF formulas and Section 4 has all the negative results.

2 Preliminaries

2.1 Boolean functions

Let $V_n = \{v_1, v_2, \ldots, v_n\}$ be a set of n Boolean variables. An *assignment* a to V_n is an n-vector in $\{0,1\}^{V_n}$. For convenience, we also view a as a function $a : V_n \to \{0,1\}$ and thus $a(v_i)$ denotes the value assigned to v_i by a. For $b \in \{0,1\}$, $a_{v_i \leftarrow b}$ is defined to be the assignment to V_n such that $a_{v_i \leftarrow b}(v_j) = a(v_j)$ for $j \neq i$, and $a_{v_i \leftarrow b}(v_i) = b$. Similarly, $a_{\neg v_i}$ is the assignment to V_n such that $a_{\neg v_i}(v_j) = a(v_j)$ for $j \neq i$, and $a_{\neg v_i}(v_i) = \neg a(v_i)$.

$\mathcal{F}_n$ denotes the set of all Boolean functions defined on the variables in V_n, i.e., each $f \in \mathcal{F}_n$ is a mapping $f : \{0,1\}^{V_n} \to \{0,1\}$. Any functions defined in this paper that are not used in specifying bounds should be assumed to be Boolean.

Let $f \in \mathcal{F}_n$, and a be an assignment to V_n. If $f(a) = 1$ then a is called a *positive assignment* of f, otherwise it is called a *negative assignment* of f. An assignment a is *justifying* for v_i (w.r.t. f) if $f(a) \neq f(a_{\neg v_i})$. The assignment a is a *positive* justifying assignment for v_i if $a(v_i) = 1$, $f(a) = 1$, and $f(a_{\neg v_i}) = 0$. A *negative* justifying assignment is one that has $a(v_i) = 0$, $f(a) = 1$, and $f(a_{\neg v_i}) = 0$. A variable v_i is *relevant* for f if there is a justifying assignment for v_i.

Let $f \in \mathcal{F}_n$ and let $f' \in \mathcal{F}_{n'}$. We say that f' is an *identification minor (ID minor)* of f if f' can be produced from f by a (possibly empty) sequence of the following three operations:

(1) *Identification of variables*, i.e., $n \geq 2, n' = n - 1$, and $f'(v_1, \ldots, v_{n-1}) = f(v_1, \ldots, v_{n-1}, v_{n-1})$ is said to be obtained from f by *identification of variables.*
(2) *Permutation of variables*, i.e., $n' = n \geq 1$, and for some bijection π :

$\{1,\ldots,n\} \to \{1,\ldots,n\}$, $f'(v_1,\ldots,v_n) = f(v_{\pi(1)},\ldots,v_{\pi(n)})$.

(3) *Addition of irrelevant variables*, i.e., $n' = n+1$ and $f'(v_1,\ldots,v_n,v_{n+1}) = f(v_1,\ldots,v_n)$ (here v_{n+1} is the irrelevant variable).

Let $1 \le i < j \le n$. We say that the function

$$f'(v_1,\ldots,v_n) = f(v_1,\ldots,v_{i-2},v_{i-1},v_j,v_{i+1},v_{i+2},\ldots,v_n)$$

is produced from f by *dp-identifying* v_i *and* v_j. (We say "dp-identifying" here to emphasize that the operation is "*d*omain *p*reserving," that is, the domain of f' is the same as the domain of f.) Note that v_i is an irrelevant variable of f' and that f' is an ID minor of f.

Let $f \in \mathcal{F}_n$, $\{v_{i_1},\ldots,v_{i_m}\} \subseteq V_n$, and $b_1,\ldots,b_m \in \{0,1\}$. We define the function $f_{v_{i_1}\leftarrow b_1,\ldots,v_{i_m}\leftarrow b_m}$ in $\mathcal{F}_n$ to be such that for all assignments a to V_n, $f_{v_{i_1}\leftarrow b_1,\ldots,v_{i_m}\leftarrow b_m}(a) = f(a_{v_{i_1}\leftarrow b_1,\ldots,v_{i_m}\leftarrow b_m})$. We call the function $f_{v_{i_1}\leftarrow b_1,\ldots,v_{i_m}\leftarrow b_m}$ a *dp-projection* of f.

2.2 DNF formulas

A *literal* over a variable v_i is either the variable v_i or its complement $\overline{v_i}$. An *implicant* of a function $f(v_1,\ldots,v_n)$ is a conjunction t of literals over a subset of the variables in V_n such that for any assignment a to V_n, $t(a) = 1 \Rightarrow f(a) = 1$. An implicant t of f is *prime* if deleting any literal from t causes t not to be an implicant of f. We use the notation $|t|_{lit}$ to denote the number of literals in t.

A fundamental proposition in Boolean algebra is that every Boolean function f can be represented as a disjunction of implicants of f; such a representation is called a disjunctive normal form (DNF) formula for f and the implicants are called the *terms* in the representation. The *DNF size* of f, which we denote by $|f|$, is the smallest number of terms in a DNF formula representing f. (If f is a constant function, then $|f| = 0$.)

A function f is a *k-term DNF function* if $|f| \le k$. A *k-term DNF formula* is a DNF formula with *at most* k terms. We will sometimes be imprecise and use *k-term DNF* to refer either to k-term DNF formulas or k-term DNF functions.

If $\bigvee_{i=1}^m t_i$ is a DNF formula, where each t_i is a term, then the number of *occurences of literals* in this formula is $\Sigma_{i=1}^m |t_i|_{lit}$. For f a Boolean function, $|f|_{lit}$ denotes the minimum number of occurences of literals in any DNF formula representing f.

We say that a DNF formula h representing f is *irredundant* if either deleting a term of h or deleting a literal from a term of h results in a formula that does not represent f. We say that a DNF formula h representing f is *minimum* if it is irredundant and has $|f|$ terms.

Note that a positive justifying assignment for a variable v_i vindicates the use of the uncomplemented literal v_i in an irredundant DNF formula representing f, and a negative justifying assignment for v_i does the same for the complemented literal $\overline{v_i}$.

A *read-thrice* DNF formula is one in which each variable appears (with or without negation) at most three times in the formula.

For any fixed constant k, the class of functions that are representable by k-term DNF formulas is clearly closed under ID minors. However, if k is allowed to vary as a function of n then this property may not hold. For example, applying the operation of identification of variables to a $\log n$-term DNF may yield a function that is not a $\log(n-1)$-term DNF.

It is easy to see that if ϕ is a DNF formula representing f, then a DNF formula to represent the function resulting from dp-identification of v_i and v_j may be obtained as follows: in ϕ, replace each occurrence of v_i by v_j; delete any repeated occurrence of literal v_j or $\overline{v_j}$ within a term, and delete every term that contains both literals v_j and $\overline{v_j}$.

2.3 Certificates, universal sets

Let $A \subseteq \{0,1\}^{V_n}$ be a set of assignments. Two functions f and g in $\mathcal{F}_n$ are *consistent on* A if for all $a \in A$, $f(a) = g(a)$.

For any property P of functions, a *certificate* that P holds for a function $f \in \mathcal{F}_n$ is a set $A \subseteq \{0,1\}^{V_n}$ of assignments such that for all functions $g \in \mathcal{F}_n$, if g does not have property P, then f and g are not consistent on A. The *size of a certificate* A is the number of assignments in A.

Let $C = \cup_{n>0} C_n$ be a class of functions, where C_n denotes the functions in C that are defined on V_n. Let $t : \mathcal{N} \to \mathcal{N}$. The class C has *certificates of non-membership of size* $t(n)$ if for all $n > 0$, and for all $g \in \mathcal{F}_n - C_n$, g has a certificate of size at most $t(n)$ proving that $g \notin C$. If $t(n)$ is a constant function, then we say that C has *constant-size certificates of non-membership*, and if $t(n)$ is bounded by a polynomial in n, we say that C has *polynomial-size certificates of non-membership.*

An (n,k)-universal set is a set $U \subseteq \{0,1\}^{V_n}$ of assignments such that for all subsets $\{v_{i_1}, \ldots, v_{i_k}\}$ of V_n of size k, and for all $\sigma \in \{0,1\}^k$, there exists an assignment $a \in U$ such that $\sigma = a(v_{i_1})a(v_{i_2})\ldots a(v_{i_k})$. (This definition assumes $n \geq k$. If not, we define an (n, k) universal set to be the set of all $2^n \leq 2^k$ assignments to V_n.) It's easy to show (non-constructively) that there exists an (n, k)-universal set of size at most $k2^k \log(2n)$.

2.4 Learning models

We first describe the model of learning with membership and equivalence queries. In this model, the task of the learning algorithm is to "learn" an unknown *target concept* f that is a member of a known set of representations C. Each element of C represents a single function. (Note, however, that a single function may have multiple representations in C.) We assume that each element $f \in C$ has associated with it a natural number $s(f)$, called the *size* of f. For our purposes, C will be a set of DNF formulas, each one of which represents a Boolean function in the standard way. For C a class of DNF formulas, and $f \in C$, we define $s(f)$ to be the number of terms of f.

The learning algorithm is allowed to make two types of queries to gather information about f. In a *membership query*, the learning algorithm asks "What is the value of f on a?" where a is an arbitrary element in the domain of f. In response, the learning algorithm receives the answer "$f(a)$." In an *equivalence query*, the learning algorithm asks "Is h equivalent to f?" where h is a representation of a function with the same domain and range as f. If h represents the same function as f, the response is "yes", otherwise the response is an element a in the domain of f such that $f(a) \neq h(a)$. The element a is called a *positive counterexample* if $f(a) = 1$, and a *negative counterexample* if $f(a) = 0$.

If $h \in C$, the equivalence query is *proper*. If, in addition, $s(h) \leq s(f)$, the query is *strongly proper*.

The learning algorithm is required to *exactly identify* f, that is, to output a hypothesis h that represents exactly the same function as f.

If a membership and equivalence query algorithm exactly identifies every possible target concept from C, we say that the algorithm *learns* C.

An algorithm that learns with membership and *proper* equivalence queries is required not only to use proper equivalence queries, but also to output a hypothesis that is in C. The analogous requirement holds for *strongly proper* equivalence queries.

For the remainder of this section, we assume that C is a class of Boolean formulas. We denote by C_n the representations in C of functions on n variables. If $f \in C_n$, we say that f is *defined on n variables* and we let $|f|$ denote the size of the smallest representation in C_n that is functionally equivalent to f, i.e. $|f| = \min_{g \in C_n}\{s(g) | g \equiv f\}$.

A membership and equivalence query algorithm for learning C is *polynomial-time* if there exists a polynomial $p(x, y)$, such that for all $n > 0$, and for all target concepts $f \in C_n$, the algorithm runs in time at most $p(|f|, n)$. (Queries are assumed to be answered in constant time.)

A membership and proper equivalence query algorithm for learning C is *polynomial-query* if there exists a polynomial $p(x, y)$ such that for all $n > 0$, and for all target concepts $f \in C_n$, the number of queries made by the algorithm is at most $p(|f|, n)$, and the hypotheses used in the equivalence queries (and the output hypothesis) are in C and have size at most $p(|f|, n)$.

A class C is *polynomial-time learnable* (*polynomial-query learnable*) with membership and proper equivalence queries if there exists a polynomial-time algorithm (polynomial-query algorithm) for learning C with membership and proper equivalence queries.

Where it does not cause confusion, we sometimes blur the distinction between a class C of representations and the class of functions represented by the members of C.

In Section 3.3, we refer to the PAC + membership query model of learning. In this model, the algorithm is given access to two oracles, an oracle that answers membership queries, and an *example oracle*. The example oracle provides, on request, a pair $(a, f(a))$, where a is an element drawn at random from an unknown distribution D on the domain of f. The algorithm takes two values, ϵ and δ, as input. The algorithm is said to learn a concept class C if for all $f \in C$, for all $\epsilon > 0$, for all $\delta > 0$, and for all possible distributions D on the domain of f (governing the generation of elements by the example oracle), with probability at least $1 - \delta$, the algorithm outputs a concept h defined on the same domain as f, such that $\mathrm{Prob}_{a \in D}[f(a) \neq h(a)] < \epsilon$. Here $\mathrm{Prob}_{a \in D}$ denotes the probability with respect to a random a chosen according to distribution D. A polynomial-time PAC + membership query algorithm runs in time polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, the size of the target f, and the number of variables on which f is defined.

2.5 The connection between certificates and learning

Most of the results in this paper exploit known relationships between certificates and learning. In Propositions 1 and 2, we present these relationships in a form that is relevant to the problems in this paper. Propositions 1 and 2 apply to a class C of Boolean formulas with an associated size function s.

The original results applied to more general of classes of representations, and some of them assumed the size of a hypothesis to be equal to the number of bits in its encoding.

If g is a Boolean function that can be represented by a formula in a class of Boolean formulas C, then we define the size of the function g (with respect to C) to be the size of the smallest formula in C representing g. We denote the size of g by $|g|$. That is, $|g| = \min_{f \in C}\{s(f) | f \text{ represents } g\}$. If g cannot be represented by any formula in C, we define $|g|$ to be infinity.

We denote by $C_{m,n}$ the set of formulas $f \in C_n$ such that $s(f) \leq m$.

In the propositions, when we talk about certificates of non-membership for a class of formulas, we mean certificates of non-membership for the class of functions represented by those formulas.

Proposition 1 *[22,21] Let C be a class of Boolean formulas with associated size function s such that the problem of generating all elements in C_n, given n, is computable. If C has certificates of non-membership of size $q(n)$, then C can be learned by a membership and proper equivalence query algorithm that makes $O(q(n)\log|C_n|)$ queries in learning a target in C_n. Conversely, if C can be learned by a membership and proper equivalence query algorithm that makes $z(n)$ queries in learning a target in C_n, then C has certificates of non-membership of size $z(n)+1$.*

In particular, if $\log|C_n|$ is bounded by a polynomial in n, and for $f \in C$, $|f|$ is bounded by a polynomial in n (where $f \in C_n$), then C is polynomial-query learnable using membership and proper equivalence queries iff C has polynomial-size certificates of non-membership.

Proposition 2 *[21] Let C be a class of Boolean formulas with associated size function s such that the problem of generating all elements in $C_{m,n}$, given m and n, is computable. Suppose there exist functions $p(m,n)$ and $q(m,n)$, increasing in m, such that for all $m,n > 0$, and for all Boolean functions f defined on V_n, if $|f| > p(m,n)$, then there exists a certificate of size at most $q(m,n)$ proving that $|f| > m$. Then C can be learned by a membership and proper equivalence query algorithm that makes at most $O(mq(m,n)\log|C_{m,n}|)$ queries using hypotheses of size at most $p(m,n)$ in learning any target $f \in C_n$ such that $|f| = m$.* [6]

[6] The factor of m in the bound on the number of queries can be eliminated if $|f|$ is given as input to the algorithm. The factor of m comes from repeatedly running an algorithm to learn $C_{j,n}$ for $j = 1, 2, \ldots, |f|$. If doubling of j is used, rather than incrementing by 1, then the bound on the number of queries becomes $O((\log m)q(m,n)\log|C_{m,n}|)$ and the bound on the size of the queries becomes $p(2m,n)$.

Conversely, suppose C can be learned by a membership and proper equivalence query algorithm that makes $z(m,n)$ queries and uses hypotheses of size at most $t(m,n)$ to learn a target formula $f \in C_{m,n}$. Then for all $m,n > 0$, if a Boolean function f defined on V_n is such that $|f| > t(m,n)$, then there exists a certificate of size at most $z(m,n)+1$ proving that $|f| > m$.

In particular, C is polynomial-query learnable with membership and strongly proper equivalence queries iff there exists a polynomial $q(m,n)$, increasing in m, such that for all $m,n > 0$, if f is a function defined on V_n such that $|f| > m$, then there exists a certificate of size at most $q(m,n)$ proving that $|f| > m$.

Note that the above propositions make no reference to the running time of the learning algorithms. The propositions are solely concerned with the "query complexity" of the algorithms, and not with the time complexity. The proofs of the propositions are constructive. However, even when the constructed algorithms are polynomial-query, they are not polynomial-time, in general.

The constructed algorithms can be modified to produce algorithms that run in polynomial time with the aid of a Σ_2^p oracle [12], provided that the class C is *well-behaved*, which we define as meaning that C obeys the following properties: (1) testing whether a given Boolean formula g is in C is computable in polynomial time, (2) the function s defining the size of formulas in C is polynomial-time computable, and (3) there exists a polynomial $p(x,y)$ such that number of bits in the encoding of any Boolean formula $f \in C_n$ is at most $p(s(f),n)$. [7]

2.6 Basic propositions on k-term DNF

Chandra and Markowsky proved the following useful bound on the number of prime implicants of k-term DNF formulas.

Proposition 3 *[19] If a function f is representable by a k-term DNF formula, then f has at most 2^k prime implicants.*

[7] Properties (1) and (2) allow an algorithm to test, in polynomial time, whether a given hypothesis can be used in a *proper* equivalence query, and whether it is of a suitable size. Property (3) relates the size of a hypothesis h, as measured by s, to the number of bits needed to specify h. It ensures that an algorithm specifying h does not require time superpolynomial in $s(h)$ just to write down the bits in the encoding of h. These properties imply that listing all formulas in $C_{m,n}$, given m and n, is computable.

The following two propositions are used by Bshouty et al. [5]. The second is implied by the first, based on the result of Naor et al. that an (n, k) universal set of size $2^k k^{O(\log k)} \log n$ can be constructed in time polynomial in n and 2^k [31].

Proposition 4 *[5] Let f be a k-term DNF formula defined on n variables. Let t be a conjunction of literals over a subset of the variables in V_n. Let A_t be the set of assignments produced by setting the variables in t so that t is satisfied, and setting the other variables in V_n according to the assignments of an $(n - |t|_{lit}, k)$-universal set. More precisely, if $\{v_{i_1}, \ldots, v_{i_{n-|t|_{lit}}}\}$ are the literals not appearing in t, then let U be an $(n - |t|_{lit}, k)$-universal set and let $A_t = \{b \in \{0,1\}^{V_n} | t(b) = 1$ and for some $u \in U$, $u(v_j) = b(v_{i_j})$ for all $j \in \{1, \ldots, n - |t|_{lit}\}\}$. Then t is an implicant of f iff $f(a) = 1$ for all $a \in A_t$.*

Proposition 5 *[5] Let f be a k-term DNF formula defined on n variables. Given a membership oracle for f and a conjunction of literals t over variables in V_n, testing whether t is an implicant of f can be done in time polynomial in n and 2^k.*

The next proposition follows easily from the previous one.

Proposition 6 *Let f be a k-term DNF formula defined on n variables. Given a membership oracle for f and an implicant t of f, a prime implicant of f made up of a subset of the literals of t can be generated in time polynomial in n and 2^k. Given a membership oracle for f, and a positive assignment α of f, a prime implicant of f that is satisfied by α can be generated in time polynomial in n and 2^k.*

PROOF. Suppose t is an implicant of f. The following greedy procedure generates an appropriate prime implicant of f: For each literal l of t, delete the literal from t and test, using the method in Proposition 5, whether t is still an implicant of f. If not, put l back into t. Proceed to the next literal.

Given a positive assignment α of f, running the greedy procedure on the conjunction of the n literals satisfied by α produces a prime implicant of f satisfied by α. $\square$

3 Proper Learning of k-term DNF Formulas

We motivate the results in this section by presenting a new improper algorithm for learning $\log n$-term DNF formulas. Like the more complicated algorithms of Blum and Rudich [18] and Bshouty et al. [5], it is based on universal sets.

3.1 A simple polynomial time improper algorithm for learning $\log n$*-term DNF with DNF hypotheses*

We present an algorithm that learns $\log n$-term DNF formulas with membership and equivalence queries, using n-term DNF formulas as hypotheses. The basic idea of the algorithm is the same as the standard algorithm for learning *monotone* DNF [11]: repeatedly collect prime implicants and add them to the hypothesis until a formula equivalent to the target is obtained.

The algorithm is as follows. It works in *phases*, maintaining at all times a current set S (initially empty) of prime implicants of the target formula f. Each phase starts with an equivalence query consisting of the hypothesis h obtained from the disjunction of all the prime implicants in S. If the answer to the equivalence query is "yes", the algorithm stops and outputs h. Otherwise, a counterexample a is received; since the terms of h are prime implicants of f, $f(a) = 1$. The algorithm then augments S by a prime implicant p of f satisfied by a (and thus not already in S), using the procedure described in the proof of Proposition 6 and goes to the next phase.

By Proposition 3, the target formula f has at most n prime implicants, and thus each generated hypothesis has at most n terms. By Proposition 6, the running time of the algorithm is polynomial in n.

The simplicity of the algorithm masks the difficulty of obtaining a polynomial-time *proper* learning algorithm for $\log n$-term DNF. In the next subsection, we show that it is possible to at least get a polynomial-query proper learning algorithm for $\log n$-term DNF. We do this by showing that there exist certificates of non-membership for the class of k-term DNF formulas of size polynomial in n and 2^k.

3.2 Certificates for k*-term DNF whose size depends on* n

We use the following lemmas. The first is implicit in [21]. We present a proof here for completeness.

Lemma 7 *Let $k > 0$ be an integer, and let $f(v_1, \ldots, v_n)$ be such that $|f| > k$. If $|f| > 2k$, then there exists a dp-projection f' of f, and a variable $v_l \in V_n$ such that $k < |f'| \leq 2k$, $|f'_{v_l \leftarrow 0}| \leq k$, and $|f'_{v_l \leftarrow 1}| \leq k$.*

If there is a certificate of size t proving that $|f'| > k$, then there is a certificate of size t proving that $|f| > k$.

PROOF. Let $f = f(v_1, \ldots, v_n)$ and let $|f| > k$. The following procedure produces a dp-projection f' of f having the desired properties.

To begin, let $f' = f$ and let $l = n$.

Repeat the following until termination. If $|f'_{v_l \leftarrow 0}| \leq k$ and $|f'_{v_l \leftarrow 1}| \leq k$, then terminate. Otherwise, let c be such that $|f'_{v_l \leftarrow c}| > k$, let $f' = f'_{v_l \leftarrow c}$, and decrement l.

The procedure terminates, because setting all the variables of f' to constant values would produce a dp-projection f'' of f' such that $|f''| = 0$.

Upon termination, clearly $|f'| > k$. Also, $|f'_{v_l \leftarrow 0}| \leq k$ and $|f'_{v_l \leftarrow 1}| \leq k$. Because $f' = v_l f'_{v_l \leftarrow 1} \vee \overline{v_l} f'_{v_l \leftarrow 0}$, $|f'| \leq 2k$.

Also upon termination, $f' = f_{v_{l+1} \leftarrow b_{l+1}, \ldots, v_n \leftarrow b_n}$ for some $b_{l+1}, \ldots, b_n \in \{0, 1\}$. If A is a certificate that $|f'| > k$, then $\{a_{v_{l+1} \leftarrow b_{l+1}, \ldots, v_n \leftarrow b_n} | a \in A\}$ is a certificate that $|f| > k$. $\square$

The next lemma is due to Blum and Rudich.

Lemma 8 *[18] If a is an assignment satisfying exactly one term s of a k-term DNF formula, then a is a justifying assignment (with respect to that formula) for all but at most k of the variables that appear in s.*

We can now prove the following theorem.

Theorem 9 *The class of k-term DNF formulas has certificates of non-membership of size $O(nk^2 2^k \log n + k^3 2^{3k} \log^2 n)$.*

PROOF. Let $f(v_1, \ldots, v_n)$ be such that $|f| > k$. By Lemma 7, we assume without loss of generality that $|f| \leq 2k$ and there exists $v_l \in V_n$ such that $|f_{v_l \leftarrow 0}| \leq k$, and $|f_{v_l \leftarrow 1}| \leq k$.

For concreteness, let F be a minimum DNF formula representing f.

We describe a certificate proving that $|f| > k$. The certificate will be the union of three sets we will call A, W, and Y. The set A is used to distinguish f from all functions h such that $|h| \leq k$, and f is satisfied by some assignment that falsifies h. The sets W and Y are used to distinguish f from all other functions h such that $|h| \leq k$.

For each term t in F, let A_t be the set of assignments produced by setting the variables in t so that t is satisfied, and setting the other variables F according to the assignments of an $(n - |t|_{lit}, k)$-universal set (as in Proposition 4). Let $A = \cup_{t \in F} A_t$. Clearly $f(a) = 1$ for all $a \in A$.

For each assignment $a \in A$, let $W(a) = \{a_{\neg v_i} | v_i \in V_n\}$ and let $W = \cup_{a \in A} W(a)$. Let $P(a)$ denote the set of variables v_i such that $f(a_{\neg v_i}) = 0$ (and hence $f(a) \neq f(a_{\neg v_i})$, since $f(a) = 1$).

For each assignment $a \in A$, define $X(a)$ as follows. Let the variables not in $P(a) \cup \{v_l\}$ be $\{v_{i_1}, \ldots, v_{i_{n-|P(a) \cup \{v_l\}|}}\}$ where $i_1 < i_2 < \ldots < i_{n-|P(a) \bigcup \{v_l\}|}$, and let U be an $(n - |P(a) \cup \{v_l\}|, 2k)$-universal set. Then let $X(a) = \{b \in \{0,1\}^{V_n} | b(v_i) = a(v_i)$ for all $v_i \in P(a)$ and for some $u \in U$, $u(v_j) = b(v_{i_j})$ for all $j \in \{1, \ldots, n - |P(a) \bigcup \{v_l\}|\}\}$. Note that if $v_l \notin P(a)$, then for each $b \in X(a)$, $X(a)$ also contains $b_{\neg v_l}$, and $|X(a)| = 2|U|$. Otherwise, $|X(a)| = |U|$.

Let $Y(a)$ be the set of negative assignments of f in $X(a)$ and let $Y = \cup_{a \in A} Y(a)$.

We claim that $C = A \cup W \cup Y$ is a certificate that $|f| > k$. The theorem follows from the claim after some arithmetic, using the fact (stated in Section 2.3) that there exists an (n, k) universal set of size at most $k2^k \log(2n)$.

To prove the claim, let h be a function with $|h| \leq k$. We will show that C contains an assignment on which h and f differ. Fix H to be a minimum DNF formula representing h.

Case 1: There exists an assignment that satisfies f and falsifies h

In this case, some term of F is not an implicant of h. In this case, by Proposition 4, f and h are not consistent on A.

Case 2: Every assignment satisfying f also satisfies h.

In this case, since $f(a) = 1$ for all $a \in A$, h is consistent with f on A. Let H' be a DNF consisting of a minimal set of terms from H such that H' is satisfied by all assignments in A. The function h' defined by H' is consistent with f on A, and H' has at most k terms. By Proposition 4, every assignment satisfying f also satisfies h'.

Since $|h'| \leq k < |f|$, $h' \neq f$ and so the DNF H' must contain at least one term s which is not an implicant of f. Furthermore, there must be an assignment a in A which satisfies s but falsifies all other terms in H'. (If not, deleting s from H' results in a smaller DNF which still accepts all the assignments in A.)

Either $h'(b) \neq f(b)$ for some $b \in W(a)$ or $h'(b) = f(b)$ for all $b \in W(a)$. If $h'(b) \neq f(b)$ for some $b \in W(a)$, then $h'(b) = 1$ and $f(b) = 0$, because all assignments satisfying f also satisfy h'. Since H' contains a subset of the terms of H, $h(b) = 1$ also. It follows that f and h are not consistent on W, and we are done.

Suppose, therefore, that $h'(b) = f(b)$ for all $b \in W(a)$. Since a satisfies only term s in H', by Lemma 8 and the consistency of h' and f on $W(a)$, $P(a)$ must include all but at most k literals in s. Let S be the set of these at most k literals. Since s is not an implicant of f, there exists an assignment d satisfying s but falsifying f and hence also falsifying $f_{v_l \leftarrow d(v_l)}$. Let F_l be a minimum DNF representing $f_{v_l \leftarrow d(v_l)}$. For each term of F_l, choose a literal falsified by d. Let T be the set of chosen literals. Since $|f_{v_l \leftarrow d(v_l)}| \leq k$ and F_l is minimum, $|T| \leq k$. Every assignment that, like d, satisfies the literals in $P(a)$, S, and T, and sets v_l to $d(v_l)$, will satisfy h' and falsify f. Because $|S| \leq k$ and $|T| \leq k$, such an assignment exists in $Y(a)$. Further, since h is satisfied by all assignments satisfying h', this assignment also satisfies h. Hence f is not consistent with h on $Y(a)$. □

By combining the above theorem with Proposition 1, and using the fact that a k-term DNF learning algorithm parametrized by k can be made strongly proper by repeatedly running it for increasing values of k, we obtain the following corollary.

Corollary 1 *The class of* $\log n$*-term DNF formulas is polynomial-query learnable with membership and strongly proper equivalence queries.*

3.3 Proper learning of $\log n$*-term DNF*

The question of whether $\log n$-term DNF formulas can be learned in polynomial *time* with membership and proper equivalence queries remains open. We begin by showing that a possible approach to answering the question affirmatively would be to develop a polynomial time algorithm in the weaker proper PAC + membership query model. We then show that the question is related to a truth table minimization problem.

3.3.1 PAC + membership query learning

We say that a class C is learnable in polynomial time *with high probability* in the membership and proper equivalence query model if there exists a membership and proper equivalence query algorithm that, given an input parameter δ where $0 < \delta < 1$, successfully learns any target in C with probability at least $1 - \delta$, and runs in time polynomial in $\frac{1}{\delta}$ and in the usual parameters of the membership and equivalence query model.

Suppose there exists a polynomial-time algorithm that learns $\log n$-term DNF in the proper PAC + membership query model. Let PMEAlg(ϵ, δ) denote such an algorithm. In Figure 2, we present an algorithm that, using PMEAlg as a subroutine, learns $\log n$-term DNF in polynomial time with high probability in the membership and proper equivalence query model.

In Figure 2, EQ(h) denotes the answer to the equivalence query with hypothesis h. Generating a prime implicant satisfied by a (Step 4) is done using the procedure described in the proof of Proposition 6. The target $\log n$-term DNF formula is called f.

We begin by sketching the main ideas behind the algorithm in Figure 2. The approach taken by the algorithm is essentially the same as the approach taken by the simple improper algorithm presented in Section 3.1. Both algorithms work by collecting prime implicants of the target formula f. Each time the algorithms ask an equivalence query that yields a counterexample, the counterexample is used to generate a new prime implicant of the target. However, the algorithm in Figure 2 is somewhat complex because it is limited to using only *proper* equivalence queries, that is, equivalence queries whose hypotheses are $\log n$-term DNF formulas.

Because the number of prime implicants of the $\log n$-term DNF formula f can exceed $\log n$, the algorithm in Figure 2 cannot, in general, ask an equivalence query using a hypothesis h containing all prime implicants of f collected so far. Instead, the algorithm uses h to construct a special set of assignments $C(h)$ (defined below). It then uses the proper PAC+membership query subroutine, with examples drawn randomly from $C(h)$, to produce (with high probability) a $\log n$-term DNF formula g that is consistent with the target formula f on $C(h)$. The formula g is then "minimized" – terms are deleted from g until removing any additional term from g would cause g to be inconsistent with the target f on $C(h)$. The formula g is used in place of h in an equivalence query. The correctness of the algorithm depends on the fact that, because of the way $C(h)$ is constructed and the way g is minimized, making an equivalence query with g rather than h still yields a counterexample from which a new prime implicant of the target can be obtained (unless, of course, g is equivalent to the target, in which case the algorithm is done).

This completes our sketch of the algorithm. We now define the set $C(h)$, which is similar to the certificate set defined in Section 3.2. Note that $C(h)$ is well defined for any DNF formula h defined on n variables.

For each term t of h, let A_t be the set of assignments produced by setting the variables in t so that t is satisfied, and setting the other variables of h according to the assignments of an $(n - |t|_{lit}, \log n)$-universal set (as in Proposition 4).

Let A be the union of all such A_t's.

Input: δ such that $0 < \delta \leq 1$

01. $h \leftarrow$ False

02. **while** EQ(h) $\neq$ "Yes" **do begin**

03. Let a be the counterexample returned by the query.

04. Generate a prime implicant p of the target formula f that is satisfied by a
Comment: a is a positive counterexample

05. $h \leftarrow h \vee p$

06. Construct "certificate" set $C(h)$

07. Using the membership oracle for f, check whether f and h are consistent on $C(h)$. If not, let a be such that $f(a) \neq h(a)$ and goto Step 04.

08. Run PMEAlg$(1/(|C(h)|+1), \delta/n)$, simulating the example oracle by choosing a random assignment a in $C(h)$ independently with uniform probability $1/C(h)$, and generating the value of $f(a)$ using the membership oracle for f.

09. Let g be the hypothesis produced by the PAC + membership query algorithm. (If no hypothesis is produced, output "FAILURE" and exit.)

10. **if** g is not consistent with f on $C(h)$, or g is not a $\log n$-term DNF **then** output "FAILURE" and exit.

11. Reduce g to a minimal set of terms such that each positive example of f in $C(h)$ is satisfied by some term of g, by greedily deleting redundant terms from g

12. $h \leftarrow g$

13. **end**

Fig. 2. $\log n$-term DNF learning algorithm

For each assignment $a \in A$, let $W(a) = \{a_{\neg v_i} | v_i \in V_n\}$. Let W be the union of all $W(a)$'s formed from all assignments in A. Let $P(a)$ denote the set of variables v_i such that $h(a_{\neg v_i}) = 0$ (and hence $h(a) \neq h(a_{\neg v_i})$, because $h(a) = 1$).

Also, for each assignment $a \in A$, let $X(a)$ denote the set of assignments produced by setting all variables in $P(a)$ according to a, and setting the remaining variables on which h is defined according to the assignments in an $(n - |P(a)|, 2\log n)$-universal set. Let $Y(a)$ be the set of negative assignments of h in $X(a)$. Let Y denote the set of all $Y(a)$'s formed from all assignments in A.

Then $C(h) = A \bigcup W \bigcup Y$.

The correctness of the algorithm is based on the following technical lemma.

Lemma 10 *Let g, h, and f be DNF formulas defined on n variables such that g and f have at most $\log n$ terms, and g, h, and f are consistent on $C(h)$. Further, let g be such that no term can be deleted from g without violating the consistency of g, f, and h on $C(h)$.*

Then for all $a \in \{0,1\}^{V_n}$, $h(a) = 1 \Rightarrow g(a) = 1$ and $g(a) = 1 \Rightarrow f(a) = 1$.

PROOF. $h(a) = 1 \Rightarrow g(a) = 1$: Since $h(a) = 1$, a satisfies some term t of h. Because h is consistent with g on A, by Proposition 4 t is an implicant of g also. Finally, since a satisfies t, $g(a) = 1$.

$g(a) = 1 \Rightarrow f(a) = 1$: Suppose not. Then g contains a term s that is not an implicant of f. Consider the formula g' produced by deleting s from g. Since no term of g can be deleted without violating the consistency of g, f, and h on $C(h)$, and every implicant of g' is an implicant of g, there exists an assignment $a \in C(h)$ such that $g'(a) = 0$ but $g(a) = f(a) = h(a) = 1$.

By Proposition 4, if g' and h were consistent on A, every term of h would be an implicant of g', contradicting that $h(a) = 1$ and $g'(a) = 0$. Therefore, g' and h are not consistent on A. Hence there exists $c \in A$ such that $g'(c) = 0$ but $h(c) = g(c) = f(c) = 1$. Clearly s is the only term of g satisfied by c. By the consistency of g and h on $W(c)$ and Lemma 8, the literals in $P(c)$ constitute all but at most $\log n - 1$ of the literals in s. The term s is not an implicant of f, so it must be possible to add at most $2\log n - 1$ literals to $P(c)$ (at most $\log n - 1$ to complete s, and at most $\log n$ to falsify terms of f) to produce a set of literals Q such that any assignment satisfying Q satisfies s but not f.

By the construction of $X(c)$, there exists an assignment $d \in X(c)$ such that d satisfies the literals in Q. Clearly $g(d) = 1$ and $f(d) = 0$. By the consistency of h and f on A and Proposition 4, the terms of h are implicants of f. Therefore, since $f(d) = 0$, it follows that $h(d) = 0$ also. Thus $d \in Y(c)$, contradicting the consistency of g, h, and f on $C(h)$. $\square$

We are now ready to prove the following theorem.

Theorem 11 *If* $\log n$*-term DNF formulas are polynomial-time learnable in the proper PAC + membership query model, then they are polynomial-time learnable with high probability in the membership and proper equivalence query model.*

PROOF. Suppose there exists a polynomial-time algorithm for learning the class of $\log n$-term DNF formulas in the proper PAC + membership query model. Let PMEAlg(ϵ, δ) denote such an algorithm. We prove that the algorithm in Figure 2 learns $\log n$-term DNF in polynomial time with high probability in the membership and proper equivalence query model.

Let h_i denote the value of h in the ith execution of Step 2. Let P_i denote the set of prime implicants of f that are also implicants of h_i. We can prove by induction that for all executions i of Step 2 the following hold:

(1) For all assignments b to the variables on which f is defined, $h_i(b) = 1 \Rightarrow f(b) = 1$
(2) If $i \geq 1$, $P_{i-1} \subset P_i$.

Since, by Lemma 3, f has at most n prime implicants, it follows that after at most n executions of Step 2, the algorithm will either terminate with FAILURE, or it will terminate in Step 2 after asking an equivalence query with a hypothesis equivalent to f.

To prove the above two properties, first consider the case $i = 1$. Since h_1 is identically false, the properties hold. Now consider $i = j$, for some $j > 1$. Assume that at the jth execution of Step 2, the current hypothesis h satisfies the first of the above two properties, and h is not equal to the target. Consider what happens following the jth execution of Step 2. Since h satisfies the first of the above two properties, the counterexample a obtained from the equivalence query is indeed positive, as claimed in the comment. Therefore, in Step 4, a prime implicant p of f, not contained in P_i, is generated and added to h. Steps 4 through 7 may be repeated multiple times before Step 8 is reached. The effect of these multiple repetitions is simply to add additional prime implicants of f to h. If Steps 8 through 10 do not result in the failure of the algorithm, a hypothesis g is generated and minimized. It is easy to verify that immediately before Step 12, f, g, and h satisfy the conditions of Lemma 10, and at least one new prime implicant of f was added to h since the jth execution of Step 2. It follows that the above two properties hold for $i = j + 1$, and hence for all applicable i.

The total running time of the algorithm in Figure 2 is polynomial in n and $1/\delta$ if the method of Naor et al. [31] is used to construct the universal sets.

Each run of PMEAlg, the PAC+membership query algorithm, produces a $\log n$-term DNF hypothesis g consistent with f on $C(h)$ with probability at least $1 - \delta/n$. Therefore, with probability at least δ the algorithm in Figure 2 returns a DNF formula with at most $\log n$ terms that is equivalent to f. $\square$

3.3.2 Truth table DNF minimization and $\log n$*-term DNF learning*

Consider the following problem:

n-term DNF representability given a truth table: Given the truth table of a Boolean function f defined on n variables, find a DNF formula representing f with at most n terms, or determine that such a formula does not exist.

Note that a polynomial-time algorithm for this problem runs in time polynomial in 2^n since a truth table has that many entries. A classic result of Masek is that general minimization of DNF given a truth table is NP-hard [32]. We do not know if the restricted version given above can be solved in polynomial time but we can show the following.

Theorem 12 *If* $\log n$*-term DNF can be learned in polynomial time in the membership and proper equivalence query model, then the problem of* n*-term DNF representability given a truth table can be solved in polynomial time.*

PROOF. Suppose there exists a polynomial-time membership and proper equivalence query algorithm for learning $\log n$-term DNF. We use that algorithm to solve the problem of n-term DNF representability given a truth table, as follows.

Let R be the truth table that is input to the representability problem, and let $f(v_1, \ldots, v_n)$ be the function represented by R.

Let $f'(v_1, \ldots, v_n, v_{n+1}, \ldots, v_{2^n}) = f(v_1, \ldots, v_n)$, that is, f' is produced from f by adding $2^n - n$ irrelevant variables. Clearly $|f| = |f'|$.

Let $N = 2^n$. Let $p()$ be a polynomial that is an upper bound on the running time of the learning algorithm. Without loss of generality, we may assume that the last thing the algorithm does before outputing a hypothesis h is to ask an equivalence query with hypothesis h that is answered "yes".

We use the algorithm for learning $\log N$-term DNF to try to learn f', simulating the membership and equivalence queries as follows. When a membership query asks for the value of f' on an assignment $(a_1, \ldots, a_{2^n})$, we consult R to determine the value of $f(a_1, \ldots, a_n)$ and give that value as an answer.

When an equivalence query presents a hypothesis h, we first check whether h is an n-term DNF formula. If not, there is no n-term DNF equivalent to f' (because a proper algorithm for learning k-term DNF is guaranteed to use only k-term DNF hypotheses in learning a target k-term DNF formula), and hence no n-term DNF equivalent to f. In this case, we abort the learning algorithm and output "no consistent n-term DNF" as the answer to the representability problem.

If h is an n-term DNF formula, we set the variables $a_{n+1}, \ldots, a_{2^n}$ in h to 1 producing a new DNF formula h' on the variables $a_1, \ldots, a_n$. Using R, we check whether $h'(a) = f(a)$ for all $a \in \{0,1\}^{V_n}$. If so, we abort the learning algorithm and output h' as the solution to the truth table problem. Otherwise, if $h'(a) \neq f(a)$ for some $a = (a_1, \ldots, a_n)$, we return the value $(a_1, \ldots, a_n, 1, 1, \ldots, 1)$ as a counterexample to the equivalence query.

If the learning algorithm runs for $p(n) + 1$ steps, or if it terminates without receiving a "yes" answer to an equivalence query, we immediately output "no consistent n-term DNF" as the answer to the representability problem.

It is easy to verify that the above procedure solves the representability problem in time polynomial in N. $\square$

There are 3^n possible terms that can be constructed from the variables in V_n. Therefore, the problem of n-term DNF representability given a truth table can be solved easily in time $N^{O(\log N)}$, where $N = 2^n$, by just exhaustively testing all possible n-term DNF formulas over V_n for equivalence with the function represented by the truth table.

The representability problem can easily be shown to be in the complexity class logSNP, defined in [33]. If it is also complete for logSNP, then Theorem 12 would imply that $\log n$-term DNF cannot be learned in polynomial time unless P = logSNP. However, some suggestive evidence that it is not complete for logSNP comes from the fact that the problem of $\sqrt{n}$-term DNF representability given a truth table can be solved in polynomial time. This fact can be shown either by simulation of the learning algorithm of Bshouty et al. for $\sqrt{\log n}$-term DNF, or more directly as follows. Let f be the function represented by the truth table. Using the truth table, one can generate all prime implicants of f in time polynomial in 2^n. Since, by Proposition 3, a function represented by a k-term DNF can have at most 2^k prime implicants, if f has more than $2^{\sqrt{n}}$ prime implicants then f cannot be represented by a $\sqrt{n}$-term DNF. Otherwise, one can generate all DNF formulas made up of at most $\sqrt{n}$ of the at most $2^{\sqrt{n}}$ prime implicants of f, and test each formula for equivalence with f. Since there are $O(2^n)$ such formulas, this algorithm runs in time polynomial in 2^n.

3.4 A gap theorem for certificate size

We prove a gap theorem on the size of certificates of non-membership for classes of functions that are closed under ID minors. As mentioned in Section 2, for any fixed constant k, k-term DNF formulas are closed under ID minors.

We use the following definition. Let C be a class of Boolean functions. A Boolean function $g \in \mathcal{F}_n$ is a *minimal forbidden ID minor of* C if $g \notin C$ and there is no function $g' \in \mathcal{F}_{n'}$ such that g' is an ID minor of g, $n' < n$, and $g' \notin C$.

Theorem 13 *Let C be a class of Boolean functions closed under ID minors. Let $t : \mathcal{N} \to \mathcal{N}$ be such that for all $n \in \mathcal{N}$, $t(n)$ is equal to the maximum, over all functions $f \in \mathcal{F}_n - C_n$, of the size of the smallest certificate proving that f is not in C. Then either C has constant-size certificates of non-membership, or for infinitely many values of n, $t(n) \geq \log_2 n$.*

PROOF. Suppose that C does not have constant-size certificates of non-membership. Then there exists an infinite sequence $n_1 < n_2 < n_3 \ldots$ such that for all $k \geq 1$, $t(n_k) = t(n_k + 1) = \ldots = t(n_{k+1} - 1) < t(n_{k+1})$. Fix some $k > 1$, and let $n = n_k$. Let $g \in \mathcal{F}_n - C_n$ be such that the smallest certificate proving that g is not in C has size $t(n)$. We claim that $t(n) \geq \log_2 n$. Suppose not. Let $t = t(n)$. Let A be a certificate of non-membership of g in C such that A has size t. Form a $t \times n$ binary matrix M whose rows are the assignments in A in some arbitrary fixed order. Since $t < \log_2 n$, there are $2^t < n$ distinct binary vectors that can appear as columns in M. Therefore, M must have at least two columns that are identical. Let i, j, where $i < j$, be the indices of these columns. Define the function $g'(v_1, \ldots, v_{n-1}) = g(v_1, \ldots, v_{j-1}, v_i, v_j, v_{j+1}, \ldots, v_{n-1})$. The function g' can be produced from g by identification of variables and permutation of variables, and hence is an ID minor of g. Let M' be the matrix obtained from M by deleting column j, and for any row (assignment) a of M, let a' be the corresponding row (assignment) in M'. Since columns i and j of M are equal, $g'(a') = g(a)$.

Now consider the function $g''(v_1, \ldots, v_n) = g'(v_1, \ldots, v_{j-1}, v_{j+1}, \ldots, v_n)$. The function g'' can be obtained from g' by addition of an irrelevant variable and permutation of variables. Moreover, for all rows a of M, $g'(a') = g''(a)$, which implies that $g''(a) = g(a)$. Since A is a certificate of non-membership of g in C, and $g(a) = g''(a)$ for all $a \in A$, it follows that g'' is not in C. Since g'' is an ID minor of g', and C is closed under ID minors, it follows that g' is also not in C.

Since $g' \in F_{n-1} - C_{n-1}$, $n = n_k$, and $t(n-1) < t(n)$, g' has a certificate B' of size $t' < t = t(n)$ proving it is not in C. Form a $t' \times (n-1)$ binary matrix Q' whose rows are the assignments in B' in some arbitrary fixed order. Then form a $t' \times n$ matrix Q whose first $j-1$ columns are equal to the first $j-1$ columns of Q', whose jth column is a copy of column i of Q', and whose $j+1$st through nth columns are equal to columns j through $n-1$ of Q'. Let B be the set of rows (assignments) of Q.

We claim that B is a certificate of non-membership of g in C. Consider any $f \in C_n$. We will show that there is an assignment $b \in B$ such that $f(b) \neq g(b)$. Let $f'(v_1, \ldots, v_{n-1}) = f(v_1, \ldots, v_{j-1}, v_i, v_j, v_{j+1}, \ldots, v_{n-1})$. Since f' is an ID minor of f, $f' \in C$. For each row (assignment) b' of Q', and its corresponding row b of Q, $f'(b') = f(b)$, and $g'(b') = g(b)$. Since B' is a certificate of non-membership of g' in C, there exists a $b' \in B'$ (a row of Q') such that $f'(b') \neq g'(b')$ and hence $f(b) \neq g(b)$. This proves the claim.

It follows from the claim that g has a certificate of non-membership in C of size $t' < t = t(n)$. Contradiction. $\square$

3.5 Certificates whose size is independent of n

We prove an upper bound depending only on k, and not on n, on certificates of non-membership for the class of k-term DNF formulas. It follows that for constant k, k-term DNF has constant-size certificates of non-membership.

The idea behind our upper bound proof is, roughly speaking, as follows. Consider a function f such that $|f| > k$. By Lemma 7, we can assume without loss of generality that $|f| \leq 2k$. For k a constant, then, we can assume that f can be expressed by a DNF formula with a constant number of terms. The heart of our proof is showing that we can also assume that f is defined on a constant number of variables. To do this, we introduce a new relation on variables, which we call the *twin* relation. The twin relation partitions the variables of f into equivalence classes. We show that, because f can be expressed by a DNF formula with a constant number of terms, the number of such equivalence classes is constant. By repeatedly combining variables of f that are in the same equivalence class (in an appropriate way), we transform f into a new function g defined on only a constant number of variables. There is a constant-size certificate proving that $|g| > k$, and we show that this certificate yields a constant-size certificate that $|f| > k$.

We now present our upper bound proof. We begin by defining the twin relation.

We say that two variables v_i and v_j are *twins* in a function f if either

(a) v_i and v_j have exactly the same sets of positive and negative justifying assignments, or
(b) the set of positive justifying assignments of v_i is the set of negative justifying assignments of v_j and the set of positive justifying assignments of v_j is the set of negative justifying assignments of v_i.

In case (a), we say that v_i and v_j are *identical* twins and in case (b) that they are *fraternal* twins.

Lemma 14 *Let h be an irredundant DNF formula representing a function f.*

Variables v_i and v_j of f are identical twins with respect to f iff every term of h satisfies the following:

(1) it contains v_i iff it contains v_j
(2) it contains $\overline{v_i}$ iff it contains $\overline{v_j}$

Similarly, variables v_i and v_j of f are fraternal twins with respect to f iff every term of h satisfies the following:

(1) it contains v_i iff it contains $\overline{v_j}$
(2) it contains $\overline{v_i}$ iff it contains v_j

PROOF. We present the proof for identical twins. The proof for fraternal twins is analogous.

Suppose that v_i and v_j are identical twins and t is some term of h that contains the literal v_i. Since h is irredundant, there must exist an assignment α such that $\alpha(v_i) = 1$ and $1 = t(\alpha) = f(\alpha) \neq f(\alpha_{\neg v_i})$. Since v_i and v_j are identical twins, $\alpha(v_j) = 1$ and $f(\alpha) \neq f(\alpha_{\neg v_j})$. Consequently, t must contain v_j lest it be satisfied by $\alpha_{\neg v_j}$. A symmetric argument shows that if t contains $\overline{v_i}$, then t must also contain $\overline{v_j}$.

Conversely, suppose each term of h contains v_i iff it contains v_j, and contains $\overline{v_i}$ iff it contains $\overline{v_j}$. Let α be a positive justifying assignment for v_i. Since α is a positive justifying assignment for v_i, α must satisfy a term s of h containing the literal v_i. The term s must also contain v_j. Therefore, $\alpha(v_i) = \alpha(v_j) = 1$.

Let t be a term of h. Since $f(\alpha_{\neg v_i}) = 0$, $\alpha_{\neg v_i}$ falsifies t. We will show that $\alpha_{\neg v_j}$ also falsifies t.

If term t contains v_i and v_j, or it contains $\overline{v_i}$ and $\overline{v_j}$, then clearly $\alpha_{\neg v_j}$ falsifies t. The only other possibility is that the variables v_i and v_j do not appear in t in either negated or unnegated form. In this case, since $\alpha_{\neg v_i}$ falsifies t, so does

$\alpha_{\neg v_j}$. Hence $\alpha_{\neg v_j}$ falsifies all terms t of h, implying that $h(\alpha_{\neg v_j}) = 0$ and thus α is a positive justifying assignment for v_j.

A symmetric argument shows that if α is a negative justifying assignment for v_i, then α is a negative justifying assignment for v_j. $\square$

For any variable v_i, let $\mathrm{Twins}(v_i, f)$ denote the set of all variables that are twins of v_i with respect to function f. For fixed f, $\mathrm{Twins}(v_i, f)$ is clearly an equivalence class. In what follows, we are concerned with the study of minimum DNF representations of f. The following simple proposition enables us to assume in many situations, without incurring any loss of generality, that f has only identical twins.

Proposition 15 *Let $f(v_1, \ldots, v_n)$ be a function. Let $f_{\overline{v_i}}$ denote the function obtained by replacing v_i with $\overline{v_i}$ in f (formally, $f_{\overline{v_i}}(v_1, \ldots, v_n) = f(v_1, \ldots, v_{i-1}, \overline{v_i}, v_{i+1}, \ldots, v_n)$). If h is any DNF formula representing f, then the formula obtained by replacing every occurrence of v_i with $\overline{v_i}$ represents the function $f_{\overline{v_i}}$. Moreover, for any pair of variables v_j and v_m of f, v_j and v_m are twins in f iff they are twins in $f_{\overline{v_i}}$.*

Lemma 16 *Let h be a minimum DNF formula representing f. Let v_i and v_j be identical twins such that $|\mathrm{Twins}(v_i, f)| \geq 3$ and $i < j$. Then the formula h' obtained by deleting every occurrence of v_i and $\overline{v_i}$ from h (and doing no simplification to the resulting formula) is an irredundant DNF formula representing the function f' produced by dp-identifying v_i and v_j. Furthermore, if $v_m \in \mathrm{Twins}(v_i, f)$ and $m \neq i$, then $v_m \in \mathrm{Twins}(v_j, f')$. If v_m is an identical twin of v_i and v_j in f, then v_m is an identical twin of v_j in f'.*

PROOF. Let f, h, v_i and v_j be as in the statement of the lemma. Since $|\mathrm{Twins}(v_i, f)| \geq 3$, there exists a variable $v_m \in \mathrm{Twins}(v_i, f)$ distinct from v_i and v_j. By Proposition 15, we can assume without loss of generality that v_m is an identical twin of v_i, and v_j.

Since by Lemma 14 every term of h contains v_i (respectively $\overline{v_i}$) iff it contains v_j (respectively $\overline{v_j}$), it is clear that h' is a DNF representation of f'. We now show that h' is irredundant.

For any assignment α to V_n, define α' as follows: If $\alpha(v_i) = \alpha(v_j) = \alpha(v_m)$, then $\alpha' = \alpha$, else $\alpha' = \alpha_{v_j \leftarrow 0, v_m \leftarrow 1}$. By Lemma 14, every term of h containing v_i (respectively, $\overline{v_i}$) contains v_m and v_j (respectively, $\overline{v_j}$ and $\overline{v_m}$). It follows that an assignment α satisfies a term of h iff α' satisfies that term, and α' satisfies a term X of h iff α' satisfies the corresponding term X' of h' (where X' is the term X with any occurrence of v_i or $\overline{v_i}$ deleted). Therefore, for all assignments α, $h(\alpha) = h'(\alpha')$.

Consider a term X' of h' and the corresponding term X of h. Since h is irredundant, there exists an assignment α that satisfies X but does not satisfy any other term of h. The assignment α' therefore satisfies X' but no other term of h'. Thus removing X' from h does not produce an equivalent DNF.

Now consider any literal l appearing in a term X' of h', and the corresponding term X of h. Without loss of generality, assume that l is an unnegated variable v_c. Since h is irredundant, there exists an assignment α that satisfies X and such that $h(\alpha_{\neg v_c}) = 0$. Clearly α' also satisfies X and X', and hence $h'(\alpha') = 1$. Since v_c is a literal in X', clearly v_c is not equal to v_i. Suppose v_c is not equal to v_j or v_m. Then $(\alpha_{\neg v_c})' = (\alpha')_{\neg v_c}$. Because $h(\alpha_{\neg v_c}) = 0$, it follows that $h'((\alpha_{\neg v_c})') = 0$ and hence $h'((\alpha')_{\neg v_c}) = 0$ also. Since α' satisfies X' and $h'((\alpha')_{\neg v_c}) = 0$, removing v_c from X' in h' does not produce an equivalent DNF.

Now suppose v_c is equal to v_j or v_m. Since X' (and hence X) contains v_c and α satisfies X, $\alpha(v_i) = \alpha(v_j) = \alpha(v_m)$, and by definition, $\alpha' = \alpha$. Thus $\alpha_{\neg v_c} = \alpha'_{\neg v_c}$, and hence $h(\alpha'_{\neg v_c}) = 0$. Again, α' satisfies X' and $h'(\alpha'_{\neg v_c}) = 0$, proving that removing v_c from X' in h' does not produce an equivalent DNF.

This completes the proof that h' is irredundant. Clearly a term of h' contains the literal v_j (respectively $\overline{v_j}$) iff it contains the literal v_m (respectively $\overline{v_m}$). Therefore, by Lemma 14, $v_m \in \mathrm{Twins}(v_j, f')$, and v_m is an identical twin of v_j and v_i. $\square$

Lemma 17 *Let h be a minimum DNF formula representing a function f. Let v_i and v_j be identical twins such that $|\mathrm{Twins}(v_i, h)| \geq 3$ and $i < j$. Let f' be the function obtained by dp-identifying v_i and v_j in f. Then $|f'| = |f|$.*

PROOF. Let $v_m \in \mathrm{Twins}(v_i, h)$ such that v_m is distinct from v_i and v_j. By Proposition 15, we assume without loss of generality that v_m is an identical twin of v_i and v_j in h.

Let h' be the DNF formula obtained from h by deleting every occurence of v_i and $\overline{v_i}$ from h (and doing no simplification). By Lemma 16, h' is a irredundant DNF for f'. The formula h' has the same number of terms as h.

We claim that h' is a minimum DNF representing f' by showing that any minimum DNF for f' has at least as many terms as h. The lemma follows immediately from this claim.

To prove the claim, let g' be a minimum DNF representing f'. Consider the formula g produced from g' by replacing every occurence of the literal v_j (respectively, $\overline{v_j}$) in g by the conjunction of literals $v_i v_j$ (respectively, $\overline{v_i}\,\overline{v_j}$).

We will show that g represents f, and thus g, and hence g', has at least as many terms as h.

Note that because g' and h' are irredundant, by Lemmas 14 and 16, each term of g' and h' either contains v_j and v_m, $\overline{v_j}$ and $\overline{v_m}$ or neither v_j, v_m, $\overline{v_j}$, nor $\overline{v_m}$.

Consider an assignment α to V_n. We must show that $f(\alpha) = g(\alpha)$.

- Case 1: $\alpha(v_i) = \alpha(v_j)$
  In this case it is clear that $g'(\alpha) = g(\alpha)$ and $h'(\alpha) = h(\alpha)$. Since h represents f and h' and g' are equivalent, $f(\alpha) = g(\alpha)$.
- Case 2: $\alpha(v_i) \neq \alpha(v_j)$ and $f(\alpha) = 1$
  Since $f(\alpha) = 1$, by Lemma 14, α must satisfy a term X of h that does not contain v_i, v_j, v_m, or the negation of any of these variables. Term X also appears in h'. Let $\beta = \alpha_{v_m \leftarrow \alpha(v_i)}$. The assignment β also satisfies X. Since h' and g' are equivalent, $g'(\beta) = h'(\beta) = 1$. Since $\beta(v_m) = \alpha(v_i) \neq \alpha(v_j) = \beta(v_j)$, β satisfies a term Y of g' that does not contain v_j, v_m, or the negation of either of those variables. The assignment α also satisfies Y, and therefore $g(\alpha) = 1$, and $g(\alpha) = f(\alpha)$.
- Case 3: $\alpha(v_i) \neq \alpha(v_j)$, $f(\alpha) = 0$, and $f'(\alpha) = 0$
  Here $g'(\alpha) = 0$ since g' represents f'. Since g is produced from g' by adding literals to terms of g, $g(\alpha) = 0$ also, and $g(\alpha) = f(\alpha)$.
- Case 4: $\alpha(v_i) \neq \alpha(v_j)$, $f(\alpha) = 0$, and $f'(\alpha) = 1$
  The assignment α can only satisfy terms of h' that contain v_j and v_m or their negations, lest α satisfy a corresponding term of h. Let $\beta = \alpha_{v_m \leftarrow \alpha(v_i)}$. Since $\beta(v_j) \neq \beta(v_m)$, and β falsifies all terms of h' not containing v_j and v_m or their negations (since α falsifies them), it follows that $h'(\beta) = 0$. Since g' is equivalent to h', $g'(\beta) = 0$. Since g is produced from g' by adding literals to some terms of g, $g(\beta) = 0$ also. It follows that α falsifies all terms of g that do not contain v_i, v_j, v_m, or the negation of any of those variables. Finally, since any term of g containing v_m (respectively, $\overline{v_m}$) also contains v_j and hence v_i (respectively, $\overline{v_j}$ and hence $\overline{v_i}$), α falsifies any such term. Therefore, $g(\alpha) = 0$ and $g(\alpha) = f(\alpha)$.

□

Lemma 18 *Let h be a minimum DNF formula representing a function f. Let $i < j$ and let v_i and v_j be identical twins with respect to f. Let f' be the function obtained by dp-identifying v_i and v_j in f and let X' be a certificate that $|f'| > k$. Then the set X of assignments obtained from X' by copying the value v_j to v_i (i.e. $X = \{\alpha_{v_i \leftarrow \alpha(v_j)} | \alpha \in X'\}$) is a certificate that $|f| > k$.*

PROOF. Suppose for contradiction that there exists a function g such that $|g| \leq k$ and $g(\alpha_{v_i \leftarrow \alpha(v_j)}) = f(\alpha_{v_i \leftarrow \alpha(v_j)})$ for all $\alpha \in X'$. Let h be a minimum DNF formula representing g.

Let h' be the DNF formula produced from h by replacing every occurence of the literal v_i (respectively, $\overline{v_i}$) by v_j (respectively $\overline{v_j}$). Since $\alpha_{v_i \leftarrow \alpha(v_j)}(v_i) = \alpha_{v_i \leftarrow \alpha(v_j)}(v_j)$ for all $\alpha \in X'$ and v_i is an irrelevant variable of h', $h(\alpha_{v_i \leftarrow \alpha(v_j)}) = h'(\alpha_{v_i \leftarrow \alpha(v_j)}) = h'(\alpha)$ for all $\alpha \in X'$.

Similarly, since f' is produced from f by dp-identifying v_i and v_j, $f(\alpha_{v_i \leftarrow \alpha(v_j)}) = f'(\alpha_{v_i \leftarrow \alpha(v_j)}) = f'(\alpha)$ for all $\alpha \in X'$.

Since h represents g, $h(\alpha_{v_i \leftarrow \alpha(v_j)}) = f(\alpha_{v_i \leftarrow \alpha(v_j)})$ for all $\alpha \in X'$. Thus $h'(\alpha) = f'(\alpha)$ for all $\alpha \in X'$. Since h' has at most k terms, this contradicts that X' is a certificate that $|f'| > k$. $\square$

Theorem 19 *The class of k-term DNF formulas has certificates of non-membership of size $O(k^3 2^{(1+2\log_2 3)k})$ (which is approximately $O(k^3 2^{4.17k})$). Thus for any integer constant $k \geq 1$, the class of k-term DNF formulas has constant-size certificates of non-membership.*

PROOF. Let k be a constant, and let $f(v_1, \ldots, v_n)$ be such that $|f| > k$. By Lemma 7, we can assume without loss of generality that $|f| \leq 2k$.

Let h be a minimum DNF formula representing f. Consider the set of equivalence classes $\{\text{Twins}(v_j, f) | v_j \in V_n\}$. For each equivalence class in this set, by Proposition 15 we may assume that all pairs of variables in the set are identical twins. Hence by Lemma 14, in any term of h, the members of an equivalence class all have the same *type* of occurence within the term: either all members appear without negation, all appear with negation, or none appear. Furthermore, if E_i and E_j are two distinct equivalence classes, then there must be some term in which the type of occurence of members of E_i in that term differs from the type of occurence of members of E_j. It therefore follows from Lemma 14 that the number of equivalence classes is at most 3^{2k}.

For each equivalence class $E = \{v_{j_1}, \ldots, v_{j_m}\}$ in $\{\text{Twins}(v_j, f) | v_j \in V_n\}$ such that $j_1 < j_2 < \ldots < j_m$, if $m \geq 3$, dp-identify j_{m-1} and j_m, then j_{m-2} and j_m, then j_{m-3} and j_m, and so on down to j_2 and j_m. The resulting function g has at most two relevant variables from each of the equivalence classes of f, and thus g has at most $2 * 3^{2k}$ relevant variables. By Lemma 17, $|g| = |f|$. Let g' be the function derived from g by deleting all irrelevant variables from g (formally, if $v_{j_1}, \ldots, v_{j_d}$ are the relevant variables of g and $j_1 < j_2 < \ldots < j_d$, then g' is defined on V_d, and for all $v_1, \ldots, v_d \in \{0, 1\}$, $g'(v_1, \ldots, v_d) = g(x_1, \ldots, x_n)$, where $x_{j_i} = v_i$ for $1 \leq i \leq d$, and x_l equals an arbitrary constant if $l \notin \{j_1, \ldots, j_d\}$).

Clearly $|g'| = |g|$. The number of variables on which g' is defined is at most 3^{2k}, and thus by Theorem 9 and Lemma 18, there is a certificate of size $O(k^3 2^{(1+2\log_2 3)k})$ that $|g| > k$ and a certificate of the same size that $|f| > k$. $\square$

For $n > 3^{2k}$, the bound in Theorem 19 is at least as good, or better, than the bound in Theorem 9.

We also have the following learning result.

Theorem 20 *There is a membership and proper equivalence query algorithm that learns the class of k-term DNF formulas using $O(nk^4 2^{(1+2\log_2 3)k})$ queries.*

PROOF. Follows immediately from Theorem 9, Proposition 1, and the fact that the number of k-term DNF formulas on n variables is at most 3^{nk}. $\square$

4 Negative results for learning DNF

We first prove lower bounds on the size of certificates for k-term DNF. The details of the construction are then used to prove a series of non-learnability results involving DNF formulas.

4.1 Non-learnability of DNF using minimum number of terms

We use the following construction of Umans [28]. Let ϕ be a DNF formula. Let $X = a_1 \dots a_N$ be an implicant of ϕ such that X is not equivalent to ϕ.

Define

$$\phi' = \phi \wedge (\overline{a_1} \vee \overline{a_2} \vee \dots \vee \overline{a_N})$$

Multiplying out the terms of ϕ' (deleting any terms that become identically false, but preserving any other terms, even duplicates), let $s_1, \dots, s_m$ be the terms of the resulting DNF.

Define new variables $w_1, w_2, \cdots, w_m$ and $d_1, d_2, \cdots, d_m$ and let

$$\gamma' = \bigvee_{i=1}^{m} s'_i \vee \bigvee_{i=1}^{N} (\bigvee_{j=1}^{m} u_i^j)$$

where $s'_i = w_1 \dots w_{i-1} s_i w_{i+1} \dots w_m$, $u_i^j = \overline{a_i} d_j w_1 \dots w_m$.

The following lemma is due to Umans.

Lemma 21 *[28] Let f be the function represented by γ'. Let k be the minimum value of $|X'|_{lit}$, over all implicants X' of ϕ such that the literals of X' are a subset of $\{a_1, \ldots, a_N\}$. Then $|f| = m(k+1)$.*

We include a proof of a more general version of Umans's lemma in the appendix. Using the lemma, we get the following result on certificate size.

Lemma 22 *For any positive integer k, if γ' is not representable as a DNF formula with at most $m(k+1)$ terms, then a certificate of this fact must include, for every conjunction X' consisting of k literals in $\{a_1, \ldots, a_N\}$, an assignment that falsifies ϕ and satisfies X'.*

PROOF. Let $X' = a_{l_1} a_{l_2} \ldots a_{l_k}$. Consider

$$\gamma'' = \bigvee_{i=1}^{m} s'_i \vee \bigvee_{i=1}^{k} \bigvee_{j=1}^{m} u^j_{l_i}$$

where $u^j_{l_i} = \overline{a_{l_i}} d_j w_1 \ldots w_m$.

Since γ'' has $m(k+1)$ terms, and those terms are a subset of the terms of γ', the certificate must contain an assignment A such that $\gamma'(A) = 1$ and $\gamma''(A) = 0$. The assignment A must therefore set $\overline{a_{i^*}} d_{j^*} w_1 \ldots w_m$ to true for some $i = i^*$ such that a_i is not in X' and some $j = j^*$. It follows that it must set all the w_i's to 1 and d_{j^*} to 1. The assignment A must also falsify every term in γ''. Therefore, it must satisfy X', lest A satisfy term $u^{j^*}_{l_i}$ of γ'' for some i. Finally, A must falsify $\phi \wedge (\overline{a_1} \vee \ldots \vee \overline{a_N})$ in order to falsify the s'_i's. Since A satisfies $\overline{a_{i^*}}$, it must therefore falsify ϕ. □

We now define a specific DNF formula ϕ and apply Umans's construction to it. Let $N > 1$. Let C be a circuit (over AND, OR, NOT gates, where the AND and OR gates have unbounded fan-in) of size $O(N)$ computing the majority function over the variables $a_1, \ldots, a_N$. It is well known that such a majority circuit exists (see, for example, [34]).

Let t be the number of gates of C. Let $c_1, \ldots, c_t$ be Boolean variables denoting the output of the t gates, with c_t being the output of the entire circuit C.

Define ϕ to be the DNF formula made up of disjunction of the following terms, collected by iterating i from 1 through t:

- If the i^{th} gate is an AND over the variables $v_1, \ldots, v_l$, then include terms $c_i\overline{v_j}$ for all v_j, and the term $\overline{c_i}v_1 \ldots v_l$.
- If the i^{th} gate is an OR over the variables $v_1, \ldots, v_l$, then include terms $\overline{c_i}v_j$ for each v_j, and the term $c_i\overline{v_1} \ldots \overline{v_l}$
- If the i^{th} gate is a NOT of the variable v, then include the terms vc_i and $\overline{vc_i}$.
- In addition to using the appropriate rule above, if i is t, include the term c_t.

This is a standard construction for creating a DNF formula from a circuit (See e.g. [35], p. 163. Umans uses this construction for a different purpose in [28].) Since C computes the majority function, it is easy to show the following proposition:

Proposition 23 *Let $S = \{a_1, \ldots, a_N\}$. Then*

(1) If S' is a subset of S of size at least $N/2$, then the conjunction of the variables in S is an implicant of ϕ.
(2) If S' is a subset of S of size less than $N/2$, then the conjunction of the variables in S is not an implicant of ϕ.
(3) If an assignment A to the variables of ϕ falsifies ϕ, then at most $N/2 - 1$ of the variables in S are set to 1 by A.

PROOF. The proof follows from the fact that for any assignment to the variables of ϕ, if $c_1, \ldots, c_t$ all equal the values that appear on the corresponding wires of C, when C is given input values $a_1, \ldots, a_N$, then the value of ϕ is precisely the output of C on inputs $a_1, \ldots, a_N$. If $c_1, \ldots, c_t$ do not all equal the values that appear on the corresponding wires of C, when C is given input values $a_1, \ldots, a_N$, then ϕ is satisfied. □

Using ϕ in Umans's construction, we get a new DNF formula γ'. Let f be the function computed by γ' (f is defined on the variables of γ').

We are now ready to prove the following certificate size lower bound.

Lemma 24 *Let $k < \frac{N}{2}$. Then $|f| > m(k+1)$, and any certificate proving that $|f| > m(k+1)$ contains at least $\frac{\binom{N}{k}}{\binom{\frac{N}{2}-1}{k}}$ assignments.*

PROOF. Let $S = \{a_1, \ldots, a_N\}$ and let $X = a_1 \ldots a_N$. By Proposition 23, X is an implicant of ϕ, but for any conjunction X' consisting of fewer than $N/2$ variables of the variables appearing in X, X' is not an implicant of ϕ. Therefore, by Lemma 21, γ' cannot be represented by a DNF of size $m(k+1)$

and hence $|f| > m(k+1)$. By Lemma 22, any certificate that γ' cannot be represented by a DNF of size $m(k+1)$ must contain, for every conjunction X' consisting of k variables in S, an assignment that falsifies ϕ and satisfies X'. There are $\binom{N}{k}$ such conjunctions X'. Every assignment that falsifies ϕ sets at least $N/2+1$ variables in S to 0, and hence sets at most $N/2-1$ variables to 1. It follows that every assignment that falsifies ϕ satisfies at most $\binom{\frac{N}{2}-1}{k}$ conjunctions X' consisting of k variables in S. Therefore, the number of assignments in the desired certificate is at least $\frac{\binom{N}{k}}{\binom{\frac{N}{2}-1}{k}}$. □

Note that $m = \theta(N^2)$, and that γ', and hence f, is defined on $\theta(N^2)$ variables. Thus $|f| = m(\frac{N}{2}+1) = \theta(N^3)$.

We now prove our main non-learnability results.

Theorem 25 *The following hold:*

(1) For any $\epsilon > 0$, $\Omega(\log^{(3+\epsilon)} n)$-term DNF is not polynomial-query learnable with membership and strongly proper equivalence queries.
(2) DNF is not polynomial-query learnable with membership and strongly proper equivalence queries.

PROOF. The second statement follows immediately from the first. We prove the first.

Let f be as constructed above. Let n be the number of variables on which f is defined. Then $n = \theta(N^2)$.

Setting k to $\frac{N}{2}-1$ in Lemma 24 shows that it takes a certificate of size at least $\binom{N}{\frac{N}{2}-1}$ to prove that $|f| > m(\frac{N}{2})$. Since $\binom{N}{\frac{N}{2}-1} = O(\frac{2^N}{\sqrt{N}})$, it is not polynomial in n and in $|f|$.

Since $|f| = \theta(N^3)$, it follows from Proposition 2 that for some $p(n) = \theta(n^{\frac{3}{2}})$, the class of $p(n)$-term DNF formulas is not polynomial-query learnable with membership and strongly proper equivalence queries.

We now use a padding trick to improve the above non-learnability result to $\log^{3+\epsilon} n$-term DNF, where ϵ is an arbitrary constant greater than 0. Let $d > 1$ be a constant such that $|f| \le dN^3$. Let $n' = 2^{dN^{(3/(3+\epsilon))}}$. Without loss of generality, assume $n' > N$. Add irrelevant [8] variables to f so that it is defined on n' variables, and call the new function f'. Then $|f'| = |f| = m(\frac{N}{2}+1) \le$

[8] By using the idea of twins from section 3.5, this part of the proof can be modified to use a padding consisting of relevant variables, if desired.

$dN^3 \leq \log^{(3+\epsilon)} n'$ and it takes a certificate of size at least $\binom{N}{\frac{N}{2}-1}$ to prove that $|f'| > m(\frac{N}{2})$. Since $\binom{N}{\frac{N}{2}-1}$ is still superpolynomial in n', the second statement of the lemma then follows immediately from Proposition 2 and the fact that $|f'| \leq \log^{(3+\epsilon)} n'$. $\square$

The previous theorem implies that a polynomial-query algorithm for learning DNF with membership and proper equivalence queries would sometimes have to use hypotheses with more terms than the target formula. But how much larger would the hypotheses have to be? The following theorem gives a lower bound.

Theorem 26 *Let $p(t,n)$ be such that $p(t,n) = o(\frac{\sqrt{n}}{\log n})$ or $p(t,n) = \theta(\frac{t^{1/2}}{\log^{3/2+\epsilon} t})$ for some $\epsilon > 0$. Then DNF formulas cannot be polynomial-query learned by a membership and equivalence query algorithm that uses $t \cdot p(t,n)$-term DNF formulas as hypotheses, where t is the number of terms in the target formula.*

PROOF. Let f be as defined above. Let n be such that f is defined on n variables. By Lemma 24, it takes a certificate of size at least $\frac{\binom{N}{k}}{\binom{\frac{N}{2}-1}{k}}$ to prove that $|f| > m(k+1)$. Note that $n = \theta(N^2)$ and $m = \theta(n)$.

To show the result for $p(t,n) = o(\frac{\sqrt{n}}{\log n})$, let $k = \omega(\log N)$ and $k \leq \frac{N}{2} - 1$. Then $\binom{N}{k}/\binom{\frac{N}{2}-1}{k}$ is superpolynomial in n. Recall that $|f| = m(\frac{N}{2})$. Hence by Proposition 2, any membership and equivalence query algorithm for learning $m(k+1)$-term DNF using $m(\frac{N}{2})$-term DNF formulas as hypotheses must make a superpolynomial number of queries. The ratio of $|f| = m(\frac{N}{2})$, to $m(k+1)$ equals $\theta(\frac{N}{k})$. Since $n = \theta(N^2)$ and $k = \omega(\log N)$, $\theta(\frac{N}{k}) = o(\frac{\sqrt{n}}{\log n})$.

To show the result for $p(t,n) = \theta(\frac{t^{1/2}}{\log^{3/2+\epsilon} t})$, let $k = \log^{1+\epsilon/2} n$. The value $\binom{N}{k}/\binom{\frac{N}{2}-1}{k}$ is again superpolynomial in n, and any membership and equivalence query algorithm for learning $m(k+1)$-term DNF using $m(\frac{N}{2})$-term DNF formulas as hypotheses must make a superpolynomial number of queries. Note that $m(\frac{N}{2}) = \theta(n^{3/2})$. Setting $t = m(k+1)$, we get that $t = \theta(n \log^{1+\epsilon/2} n)$, and thus $t \cdot \frac{t^{1/2}}{\log^{3/2+\epsilon} t} = O(n^{3/2})$. Since a superpolynomial number of queries is required for hypotheses with at most $\theta(n^{3/2})$ terms, a superpolynomial number is also required for hypotheses with at most $t \cdot \frac{t^{1/2}}{\log^{3/2+\epsilon} t}$ terms. $\square$

Theorems 25 and 26 are about learning general DNF. The following corollary is about proper learning of k-term DNF, for particular k.

Corollary 2 *Let $z(n) = \omega(\log n)$. Let $\epsilon > 0$, $c > 0$. If k is $\Omega(\log^{(3+\epsilon)} n)$ and $O(n^c)$, then k-term DNF is not poly-query learnable with membership and proper equivalence queries.*

PROOF. It follows from the proof of Theorem 25 that for any $\epsilon > 0$, there exists a particular function $g(n)$ (equal to $|f'|$) such that $g(n) = \theta(\log^{(3+\epsilon)} n)$ and $g(n)$-term DNF is not polynomial-query learnable with membership and proper equivalence queries. By increasing the number of irrelevant variables added to f in producing f', it is easy to show that this non-learnability result holds for any $g(n)$ that is $\theta(\log^{(3+\epsilon)} n)$.

To show the non-learnability result for larger values of g, we modify the construction of f'.

The proof of Theorem 25 is based on showing that it takes a certificate of size exponential in N to distinguish f, which has size $m(N/2 + 1)$, from all functions represented by formulas of the form γ'' as defined in the proof of Lemma 22, for $k = N/2 - 1$.

Pad each of the functions γ'' with irrelevant variables so that they contain n' variables (like f' does). Let S be the set of these padded γ'' functions. The size of each of these γ'' functions is $m(N/2)$.

Now let $q(n)$ be any function bounded above by a polynomial in n. We modify f' and the functions in S as follows so that they are respectively of size $m(N/2 + 1) + q(n')$ and $m(N/2) + q(n')$.

Consider the set T consisting of the assignments (to the n' variables of f') that set all the w_i's to 0. These are negative assignments of f' and of all the functions in S. The number of assignments in T is exponential in n'. For sufficiently large n', it is therefore possible to select a subset T' consisting of $q(n')$ of these assignments, such that each pair of assignments in T' differs on at least two variables.

Now let $\hat{f}$ be the function produced from f' by making all the assignments in T' be positive assignments. That is, $\hat{f}(a) = 1$ for all $a \in T'$, and $\hat{f}(a) = f(a)$ otherwise. Similarly, for each function $\sigma \in S$, let $\hat{\sigma}$ be the function such that $\hat{\sigma}(a) = 1$ for all $a \in T'$, and $\hat{\sigma}(a) = \sigma(a)$ otherwise. Let $\hat{S} = \{\hat{\sigma} | \sigma \in S\}$. It is straightforward to show that for any $a \in T'$, a is a justifying assignment, with respect to $\hat{f}$ and with respect to the functions in $\hat{S}$, for each variable it assigns. It follows that a DNF for any of these functions must contain one term for each of the assignments in T', and that term must be satisfied only by that assignment. Thus $|\hat{f}| = m(N/2 + 1) + q(n')$ and the size of the functions in $\hat{S}$ is $m(N/2) + q(n')$. A certificate distinguishing $\hat{f}$ from the functions in $\hat{S}$ also distinguishes f' from the functions in S, and hence has size superpolynomial in n'. The corollary then follows from Proposition 2. $\square$

4.2 Non-learnability of DNF using minimum number of literals

Recall that $|f|_{lit}$ is the minimum number of occurences of literals in any DNF representing f. We begin this section by presenting results similar to those in the previous section, but for $|f|_{lit}$ instead of for $|f|$. We then use these results in the next section to prove non-learnability results for read-thrice DNF.

Umans [28] mentions that a lemma similar to Lemma 21 (with different bounds) holds if we consider the number of literal occurences in γ'' rather than the number of terms. He also briefly sketches how the proof of Lemma 21 can be modified to to prove this similar lemma. Lemma 27, below, is an explicit statement of such a lemma, and we give a full proof of it in the appendix.

Let ϕ, ϕ', γ', $\{a_1, \ldots, a_N\}$, and m be as defined before the statement of Lemma 21 (That is, ϕ is an arbitrary DNF formula, and $a_1 \ldots a_N$ is an implicant of ϕ that is not equal to ϕ.) Multiplying out the terms of ϕ' (deleting any terms that become identically false, but preserving any other terms, even duplicates), let q be the number of occurences of literals in the resulting DNF. Recall that m is the number of terms in the resulting DNF.

Lemma 27 *Let f be the function represented by γ'. Let k be the minimum value of $|X'|_{lit}$, over all implicants X' of ϕ such that the literals of X' are a subset of $\{a_1, \ldots, a_N\}$. If $m > k-2$, then $q + m(m-1) + m(k-1)(m+2) < |f|_{lit} \leq q + m(m-1) + mk(m+2)$.*

Now let ϕ be the specific DNF formula based on a majority circuit defined in the previous section. Then $m = \theta(N^2)$, $q = \theta(N^2)$. Again, let f be the function represented by γ'. By Proposition 23 and Lemma 27, $|f|_{lit} > q + m(m-1) + m(\frac{N}{2} - 1)(m+2)$ and $|f|_{lit} = \theta(N^5)$.

Arguments similar to those in the previous section now yield analogous results, presented in Lemma 28 and Theorem 29 below. We omit the proofs. Note that the first of the three results listed in Theorem 29 is for $\log^{(5+\epsilon)} n$-literal occurence DNF, whereas the analogous result in the previous section is for $\log^{3+\epsilon} n$-term DNF. A similar discrepancy is seen in the third result of the theorem. This stems from the fact that $|f|_{lit} = O(N^5)$, whereas $|f| = O(N^3)$.

Lemma 28 *If $k < N/2$ then $|f|_{lit} > q + m(m-1) + mk(m+2)$ and any certificate that $|f|_{lit} > q + m(m-1) + mk(m+2)$ contains at least $\frac{\binom{N}{k}}{\binom{\frac{N}{2}-1}{k}}$ assignments.*

Theorem 29 *The following hold:*

(1) For any $\epsilon > 0$, $\log^{(5+\epsilon)} n$-literal occurence DNF is not polynomial-query learnable with membership and proper equivalence queries, if the hypotheses are restricted to have no more literal occurences than the target.
(2) DNF is not polynomial-query learnable with membership and proper equivalence queries, if the hypotheses are restricted to have no more literal occurences than the target.
(3) Let $p(t, n)$ be such that $p(t, n) = o(\frac{\sqrt{n}}{\log n})$ or $p(t, n) = \theta(\frac{t^{1/4}}{\log^{5/4+\epsilon} t})$ for some $\epsilon > 0$. Then DNF formulas cannot be polynomial-query learned by a membership and equivalence query algorithm that uses $t \cdot p(t, n)$-term DNF formulas as hypotheses, where t is the number of terms in the target formula.

4.3 Non-learnability of read-thrice DNF formulas

Consider the following two decision problems:

DNF literal occurences: Given a DNF formula with s occurences of literals, and a number $s' < s$, does there exist an equivalent DNF formula with only s' occurences of literals?

Reach Thrice DNF Expressibility: Given a Boolean formula, does there exist an equivalent DNF formula that is read-thrice?

We begin by giving a reduction from the first of these problems to the second. We use the construction given in the reduction, together with the results of the previous section, to prove a non-learnability result for read-thrice DNF.

The idea behind the reduction is to construct a new formula η' from ξ, with the property that there exists a DNF formula equivalent to ξ containing at most s' occurences of literals iff there exists a read-thrice DNF formula equivalent to η'.

The construction of η' is as follows. For each $v_i \in V_n$, let $v_{i,1}, v_{i,2} \ldots, v_{i,s'}$ be s' new variables we call the *surrogates* of v_i.

Define DNF formulas $\xi_1, \ldots, \xi_n$ such that each ξ_i is obtained from ξ by replacing, for each variable v_j, all occurences of v_j by $v_{i+j-1,1}$ (considering $n+r$ to be equal to r, for all $0 < r < n$). Note that because $\xi_1, \ldots, \xi_n$ are obtained from ξ by cyclicly permuting the variables of ξ_1, the total number of occurences of each variable $v_{j,1}$ in $\xi_1, \ldots, \xi_n$ (with or without negation) is s, the total number of literal occurrences in ξ.

For $1 \leq i \leq n$, let $\{y_{i,1}, y_{i,2}, \ldots, y_{i,ns'}\}$ be new variables and call them the surrogates of y_i. Let

$$\eta = y_{1,1}\overline{y_{2,1}}, \overline{y_{3,1}} \ldots, \overline{y_{n,1}}\xi_1 \vee \overline{y_{1,1}}y_{2,1}\overline{y_{3,1}} \ldots, \overline{y_{n,1}}\xi_2 \vee \ldots \vee \overline{y_{1,1}}\overline{y_{2,1}} \ldots, \overline{y_{n-1,1}}y_{n,1}\xi_n.$$

Let

$$\eta' = \eta \vee \bigvee_{i=1}^{n} NC(v_{i,1}, \ldots, v_{i,s'}) \vee \bigvee_{i=1}^{n} NC(y_{i,1}, \ldots, y_{i,ns'})$$

where $NC(v_1, \ldots, v_t)$ is defined to be $v_1\overline{v_2} \vee v_2\overline{v_3} \vee \ldots \vee v_t\overline{v_1}$ (NC stands for non-consensus).

We now prove that η' has the desired property.

Lemma 30 *There is a DNF formula ξ' equivalent to ξ that has at most s' total occurences of literals iff there is a read-thrice DNF formula η'' equivalent to η'.*

PROOF. Suppose ξ' is equivalent to ξ, and ξ' has s' total occurences of literals. For $1 \leq i \leq n$, define ξ'_i to be the formula obtained from ξ by replacing, for each variable v_j, all occurences of v_j by $v_{i+j-1,1}$. Let

$$\begin{aligned}\tau = &(y_{1,1}\overline{y_{2,1}}\ \overline{y_{3,1}} \ldots, \overline{y_{n,1}})\xi'_1 \vee (\overline{y_{1,1}}y_{2,1}\overline{y_{3,1}} \ldots \overline{y_{n,1}})\xi'_2 \vee \ldots \vee \\ &(\overline{y_{1,1}}\ \overline{y_{2,1}} \ldots \overline{y_{n-1,1}}y_{n,1})\xi'_n.\end{aligned} \tag{1}$$

Let $\tau' = \tau \vee \bigvee_{i=1}^{n} NC(v_{i,1}, \ldots, v_{i,s'}) \vee \bigvee_{i=1}^{n} NC(y_{i,1}, \ldots, y_{i,ns'})$.

The formula τ' is equivalent to η'. Create a read-thrice DNF η'' equivalent to τ' as follows. Distribute the conjunctions $\overline{y_{1,1}} \ldots \overline{y_{i-1,1}}y_{i,1}\overline{y_{i+1,1}} \ldots \overline{y_{n,1}}$ over the terms of the ξ'_i's to form a DNF. Each variable $y_{i,1}$ appears in the resulting DNF at most ns' times outside the NC formulas (with or without negation), because each ξ'_i has s' occurences of literals, and hence at most s' terms. For $1 \leq i \leq n$, do the following: For each of the occurences of $y_{i,1}$ outside the NC formulas, substitute a distinct surrogate of y_i.

Since the ξ'_i's are formed by cyclicly permuting the variables of ξ'_1, each $v_{i,1}$ now appears exactly s' times outside of the NC formulas. For $1 \leq i \leq n$, do the following: For each of the s' occurences of $v_{i,1}$ outside the NC formulas, substitute a distinct surrogate of v_i. Let η'' be the resulting formula.

Clearly η'' is a read-thrice DNF formula. We show that η'' is equivalent to τ'. Consider any assignment a to the variables of η''. If there exists a v_i such that a does not assign the same value to all surrogates of v_i, then q satisfies an NC formula, and thus $\eta''(a) = \tau'(a) = 1$. Similarly, if there exists a y_i such that a

does not assign the same value to all surrogates of y_i, then $\eta''(a) = \tau'(a) = 1$. Suppose no such v_i or y_i exists. Then a falsifies all the NC formulas. Moreover, since η'' is formed from τ' by performing replacements in which one surrogate of a v_i is replaced by another, or one surrogate of a w_i is replaced by another, $\eta''(a) = \tau'(a)$. Thus η'' and τ' are equivalent. Since τ' and η' are equivalent, so are η'' and η'.

Conversely, suppose there is a read-thrice DNF formula η'' equivalent to η'. Assume without loss of generality that η'' is irredundant.

We begin by showing that each variable $v_{i,j}$ must occur in one term of the form $v_{i,j}\overline{v_{i,k}}$, for some $k \neq j$. Let a be an assignment that sets all the $y_{s,t}$'s to 0, sets all surrogates of v_k to equal values if $k \neq i$, sets all surrogates of v_i except $v_{i,j}$ to 0, and sets $v_{i,j}$ to 1. Since a satisfies η', it must satisfy some term t of η''. The term t contains $v_{i,j}$, because otherwise $t(a_{v_{i,j}\leftarrow 0}) = 1$, contradicting that $\eta'(a_{v_{i,j}\leftarrow 0}) = 0$. The term t also contains $\overline{v_{i,k}}$ for some $k \neq i$, because otherwise $t(a_{v_{i,1}\leftarrow 1,\ldots,v_{i,s'}\leftarrow 1}) = 1$, contradicting that $\eta'(a_{v_{i,1}\leftarrow 1,\ldots,v_{i,s'}\leftarrow 1}) = 0$. Furthermore, since $v_{i,j}\overline{v_{i,k}}$ is an implicant of η'' and η'' is irredundant, $t = v_{i,j}\overline{v_{i,k}}$.

A symmetric argument shows that $v_{i,j}$ appears in a term of the form $\overline{v_{i,j}}v_{i,k}$ for some $k \neq j$. We call each term $v_{i,j}\overline{v_{i,k}}$ or $\overline{v_{i,j}}v_{i,k}$, $i \neq j$, a *non-consensus* term. Thus each variable $v_{i,j}$ appears at least twice in non-consensus terms of η''.

For $1 \leq i \leq n$, let η''_i be obtained from η'' as follows: Replace each surrogate of y_i by 1, each surrogate of y_j by 0 (for $j \neq i$), and each surrogate of v_j by $v_{j,1}$ (for $1 \leq j \leq n$). Delete any terms that become identically false. The resulting formula is equivalent to ξ_i.

Suppose t is a term of η'' that becomes a term $\hat{t}$ in the construction of η''_i (i.e. t does not become identically false). Clearly t is not a non-consensus term. We claim that t becomes identically false in the construction of η''_j, for all $j \neq i$. Suppose not. Then t cannot contain any non-negated surrogate of a y variable, nor any non-consensus term, implying that it is satisfied by some assignment a setting all surrogates of the y variables to 0, and setting the surrogates of each v variable to equal values. But $\eta'(a) = 0$, contradiction. Therefore, the terms of η'' that do not become false in constructing η''_i are disjoint from the terms that do not become false in constructing η''_j, for all $j \neq i$. Since the total number of occurences of surrogates of the v variables in η'' outside the non-consensus terms is at most $s'n$, by the pigeonhole principle there exists an i such that the number of literal occurences in η''_i is at most s'. Since η''_i is equivalent to ξ_i, there is a DNF equivalent to ξ with at most s' literal occurences. □

We now relate certificate size for the number of occurences of literals in ξ to certificate size showing that η' is not read-thrice.

Lemma 31 *If η' is not read-thrice. then a certificate of this fact has size at least that of the minimum size of a certificate proving that ξ does not have an equivalent DNF with at most s' occurences of literals.*

PROOF. Let C be a certificate that η' is not read-thrice. Let d be the size of the smallest certificate proving that ξ does not have an equivalent DNF with at most s' occurences of literals. Suppose $|C| < d$. Define $C' \subseteq C$ to consist of all assignments in C that, for each v_j, set all surrogates of v_j to equal values, and for some i, set all surrogates of y_i to 1 and all surrogates of y_j, where $j \neq i$ to 0. For each $a' \in C'$, let a be the assignment to V_n such that if the surrogates of y'_i are set to 1, then for all j, $a(v_j) = a'(v_{j+i-1,1})$ (which in turn equals $a'(v_{j+i-1,2}), \ldots, a'(v_{j+i-1,n})$). Clearly $\eta'(a') = \xi(a)$. Let C'' be the set of all such assignments a.

Since $|C|'' < d$, there exists a formula τ with at most s' occurences of literals such that for all $a \in C''$, $\tau(a) = \xi(a')$. For each i, let τ_i be produced from τ by replacing, for each variable v_j, all occurences of v_j by $v_{i+j-1,1}$.

Let $\beta = (y_{1,1}\overline{y_{2,1}} \ldots \overline{y_{n,1}})\tau_1 \vee (\overline{y_{1,1}} y_{2,1}\overline{y_{3,1}} \ldots \overline{y_{n,1}})\tau_2 \vee \ldots \vee (\overline{y_{1,1}} \ldots \overline{y_{n-1,1}} y_{n,1})\tau_n$ and let $\beta' = \beta \vee \bigvee_{i=1}^{n} NC(v_{i,1}, \ldots, v_{i,s'}) \vee \bigvee_{i=1}^{n} NC(y_{i,1}, \ldots, y_{i,ns'})$. Clearly β' is consistent with η' on the assignments in C. Form a DNF from β' by distributing each conjunction $\overline{y_{1,1}} \ldots y_{i,1}\overline{y_{i+1,1}} \ldots, \overline{y_{n,1}}$ appearing in β' over the terms of τ_i. For each v_i, the total number of occurences of surrogates of v_i in the resulting DNF, excluding occurences in the non-consensus terms, is at most s'. For each y_i, the total number of occurences of surrogates of y_i, excluding occurences in the non-consensus terms, is at most ns'. Replacing each occurence of a $v_{i,1}$ outside the non-consensus terms with a distinct surrogate of v_i, and each occurence of a $y_{i,1}$ outside the non-consensus terms with a distinct surrogate of y_i, yields a read-thrice formula equivalent to β', and therefore consistent with η' on the assignments in C. This contradicts that C is a certificate that η' is not representable by a read-thrice DNF. $\Box$

We can now prove the main theorem of this section.

Theorem 32 *Read-thrice DNF is not polynomial-query learnable using membership and proper equivalence queries.*

PROOF. Let ξ be equal to the DNF formula γ' defined in Section 4 and used in Section 4.2 (γ' is based on the majority circuit).

Let $s' = q + m(m-1) + m(\frac{N}{2}-1)(m+2)$ and let f be the function expressed by γ', where m, q, and N are also defined as in Sections 4 and 4.2.

Recall that $|f|_{lit} > q + m(m-1) + m(\frac{N}{2}-1)(m+2)$ and $m = \theta(N^2)$. By Lemma 28, any certificate that $|f|_{lit} > q + m(m-1) + m(\frac{N}{2}-1)(m+2)$ contains at least $\binom{N}{\frac{N}{2}-1}$ assignments. By Lemma 30, there is no read-thrice DNF formula equivalent to η'. By Lemma 31, any certificate that the function represented by η' cannot be expressed by a read-thrice DNF formula contains at least $\binom{N}{\frac{N}{2}-1}$ assignments. Since the number of variables in η' is polynomial in N, and $\binom{N}{\frac{N}{2}-1}$ is not polynomial in N, it follows that the class of read-thrice DNF formulas does not have certificates of non-membership of size polynomial in n. The theorem then follows immediately from Proposition 1. $\square$

5 Acknowledgments

We thank Christino Tamon for useful discussions, and the referees for a number of helpful suggestions and corrections. We thank Nader Bshouty for the personal communication cited in Section 1.3.

References

[1] L. Hellerstein, V. Raghavan, Exact learning of dnf formulas using dnf hypotheses, in: Proceedings of the 34th Annual Symposium on Theory of Computing (STOC), 2002, pp. 465–473.

[2] D. Angluin, M. Kharitonov, When won't membership queries help?, J. Comput. System Sci 50 (2) (1995) 336–355, doi:10.1006/jcss.1995.1026.

[3] A. Blum, M. Furst, J. Jackson, M. Kearns, Y. Mansour, S. Rudich, Weakly learning DNF and characterizing statistical query learning using Fourier analysis, in: Proc. of Twenty-sixth ACM Symposium on Theory of Computing, 1994.

[4] N. H. Bshouty, A subexponential exact learning algorithm for DNF using equivalence queries, Information Processing Letters 59 (1) (1996) 37–39.

[5] N. H. Bshouty, S. A. Goldman, T. R. Hancock, S. Matar, Asking questions to minimize errors, J. Comput. System Sci 52 (2) (1996) 268–286, doi:10.1006/jcss.1996.0021.

[6] J. C. Jackson, An efficient membership-query algorithm for learning DNF with respect to the uniform distribution, J. Comput. System Sci 55 (3) (1997) 414–440, doi:10.1006/jcss.1997.1533.

[7] A. Klivans, R. Servedio, Learning DNF in time $2^{\tilde{o}(n^{1/3})}$, in: Proceedings of the 33rd Annual Symposium on Theory of Computing (STOC), 2001.

[8] N. Linial, Y. Mansour, N. Nisan, Constant depth circuits, Fourier transform and learnability, J. of the ACM 40 (3) (1993) 607–620.

[9] K. Pillaipakkamnatt, V. Raghavan, On the limits of proper learnability of subclasses of DNF formulas, Machine Learning 25 (2-3) (1996) 237–263.

[10] J. Tarui, T. Tsukiji, Learning DNF by approximating inclusion-exclusion formulae, in: Proc. IEEE Conference on Computational Complexity, 1999, pp. 215–220.

[11] D. Angluin, Queries and concept learning, Machine Learning 2 (4) (1988) 319–342.

[12] J. Kobler, W. Lindner, Oracles in Σ_2^p are sufficient for exact learning, in: Proc. 8th Intl. Workshop on Algorithmic Learning Theory (ALT), 1997, pp. 277–290.

[13] A. Blum, Separating distribution-free and mistake-bound learning models over the Boolean domain, SIAM J. on Comput. 23 (5) (1994) 990–1000.

[14] E. Kushilevitz, A simple algorithm for learning $o(\log n)$-term DNF, Information Processing Letters 61 (6) (1997) 289–292.

[15] N. H. Bshouty, Simple learning algorithms using divide and conquer, Computational Complexity 6 (1997) 174–194.

[16] N. H. Bshouty, Exact learning via the monotone theory, Information and Computation 123 (1) (1995) 146–153.

[17] A. Beimel, F. Bergadano, N. H. Bshouty, E. Kushilevitz, S. Varricchio, On the applications of multiplicity automata in learning, in: IEEE Symposium on Foundations of Computer Science, 1996, pp. 349–358.

[18] A. Blum, S. Rudich, Fast learning of k-term DNF formulas with queries, J. Comput. System Sci 51 (3) (1995) 367–373, doi:10.1006/jcss.1995.1075.

[19] A. Chandra, G. Markowsky, On the number of prime implicants, Discrete Mathematics 24 (1978) 7–11.

[20] N. H. Bshouty, Personal communication (5 November 2003).

[21] L. Hellerstein, K. Pillaipakkamnatt, V. Raghavan, D. Wilkins, How many queries are needed to learn?, J. of the ACM 43 (5) (1996) 840–862.

[22] T. Hegedus, Generalized teaching dimensions and the query complexity of learning, in: Proceedings of the 8th Annual Conference on Computational Learning Theory (COLT'95), ACM Press, 1995, pp. 108–117.

[23] N. Pippenger, Galois theory for minors of finite functions, Discrete Mathematics 254 (2002) 405–419.

[24] O. Ekin, S. Foldes, P. Hammer, L. Hellerstein, Equational theories of Boolean functions, Discrete Mathematics 211 (2000) 27–51.

[25] L. Hellerstein, On generalized constraints and certificates, Discrete Mathematics 226 (1-3) (2001) 211–232.

[26] U. Berggren, Linear time deterministic learning of k-term DNF, in: Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory, ACM Press, 1993, pp. 37–40.

[27] N. H. Bshouty, L. Burroughs, On the proper learning of axis parallel concepts, in: Proceedings of the 15th Annual Conference on Computational Learning Theory (COLT'02), Vol. LNAI 2375, Springer Verlag, 2002, pp. 287–302.

[28] C. Umans, Hardness of approximating Σ_2^p minimization problems, in: Proc. IEEE Symposium on Foundations of Computer Science, 1999, pp. 465–474.

[29] A. Aizenstein, T. Hegedus, L. Hellerstein, L. Pitt, Complexity theoretic hardness results for query learning, Computational Complexity 7 (1) (1998) 19–53.

[30] C. Umans, The minimum equivalent DNF problem and shortest implicants, in: IEEE Symposium on Foundations of Computer Science, 1998, pp. 556–563.

[31] M. Naor, L. J. Schulman, A. Srinivasan, Splitters and near-optimal derandomization, in: IEEE Symposium on Foundations of Computer Science, 1995, pp. 182–191.

[32] W. J. Masek, Some NP-complete set covering problems, unpublished Manuscript (1979).

[33] C. H. Papadimitriou, M. Yannakakis, On limited nondeterminism and the complexity of the VC-dimension, J. Comput. System Sci 53 (2) (1996) 161–170.

[34] D. E. Muller, F. P. Preparata, Bounds to complexities of networks for sorting and for switching, Journal of the Association for Computing Machinery 22 (1975) 195–201.

[35] C. H. Papadimitriou, Computational Complexity, Addison-Wesley, 1994.

6 Appendix

We re-state Lemma 27 and Lemma 21 as a single lemma here, and give a single proof.

Lemma 33 *[28] Let f be the function represented by γ'. Let k be the minimum value of $|X'|_{lit}$, over all implicants X' of ϕ such that the literals of X' are a subset of $\{a_1, \ldots, a_N\}$. Then $|f| = m(k+1)$, and if $m > k-2$, $q + m(m-1) + m(k-1)(m+2) < |f|_{lit} \leq q + m(m-1) + mk(m+2)$.*

PROOF. Without loss of generality, assume that the literals $a_1, \ldots, a_N$ are unnegated.

First, let X' be an implicant of ϕ such that the literals of X' are a subset of $\{a_1, \ldots, a_N\}$ and $|X'|_{lit}$ is minimized. Let $a_{l_1}, \ldots, a_{l_k}$ be the literals in X'. We show that

$$\gamma'' = \bigvee_{i=1}^{m} s'_i \vee \bigvee_{i=1}^{k} \bigvee_{j=1}^{m} u^j_{l_i}$$

where $u^j_{l_i} = \overline{a_{l_i}} d_j w_1 \ldots w_m$, is equivalent to γ'.

Suppose not. Then since γ'' contains a subset of the terms of γ', there exists an assignment α such that $\gamma'(\alpha) = 1$ and $\gamma''(\alpha) = 0$. The assignment α must therefore set $\overline{a_{i^*}} d_{j^*} w_1 \ldots w_m$ to true for some i^* such that a_{i^*} is not in X' and some j^*. It follows that α must set all of $w_1, w_2, \cdots w_m$ to 1 and d_{j^*} to 1. The assignment α must also falsify every term in γ''. Therefore, it must satisfy X', lest α satisfy term $u^{j^*}_{l_i}$ of γ'' for some i. Finally, α must falsify $\phi \wedge (\overline{a_1} \vee \ldots \vee \overline{a_N})$ in order to falsify the s'_i's. Since α satisfies $\overline{a^*_i}$, it must therefore falsify ϕ. This contradicts that X' is an implicant of ϕ. It follows that γ'' is equivalent to γ'. Moreover, γ'' has $q + m(m-1) + mk(m+2)$ occurences of literals and $m(k+1)$ terms. We have thus proven that $|f| \leq m(k+1)$ and $|f|_{lit} \leq q + m(m-1) + mk(m+2)$.

We now prove lower bounds on $|f|$ and $|f|_{lit}$. Consider any irredundant DNF formula γ'' equivalent to γ'. Let k be the minimum value of $|X'|_{lit}$, over all implicants X' of ϕ such that the literals of X' are a subset of $\{a_1, \ldots, a_N\}$.

A prime implicant P of a function $g(v_1, \ldots, v_n)$ is called *essential* if there exists some assignment a to V_n, such that a satisfies P, but a doesn't satisfy any other prime implicant of g. Every irredundant DNF representing g must contain P as a term.

We begin by showing that each s'_i is an essential prime implicant of f. Fix i. Consider an assignment α setting s_i to 1, w_i to 0, all w_j's to 1 ($j \neq i$), and other variables arbitrarily. It is straightforward to verify that for all literals l in s'_i, $f(\alpha_{\neg var(l)}) = 0$, where $var(l)$ denotes the underlying variable of literal l Deleting any literal l from s'_i yields a term that is satisfied by $\alpha_{\neg var(l)}$ and hence is not an implicant of f, since $f(\alpha_{\neg var(l)}) = 0$. Therefore, s'_i is a prime implicant of f. To show that s'_i is also essential, consider any prime implicant I of f that is not equal to s'_i. There must exist a literal l contained in s'_i but not contained in I. The assignment α cannot satisfy I, because if it did, $\alpha_{\neg l}$ would also satisfy I, contradicting that $f(\alpha_{\neg l}) = 0$ and I is an implicant of f. Since s'_i is the only prime implicant of f satisfied by α, s'_i is essential.

It follows that γ'' must contain each s'_i. There are m terms s'_i, and these m terms contain $q + m(m-1)$ occurences of literals.

Let R be the set of terms in γ'' that are not s'_i terms and let $t \in R$ be any such term. We first show that t must contain one unnegated literal d_j, for some j. Suppose not. Consider the set A of assignments satisfying t. Let A' be the set of assignments produced from the assignments in A by setting all the d_j variables to 0. The assignments in A' satisfy t and hence γ'', and since γ' and γ'' are equivalent, the assignments in A' also satisfy γ'. It follows that each assignment in A' must satisfy a term s'_i in γ', and thus each assignment in A' satisfies a term s'_i in γ'' as well. But then each assignment in A satisfies a term s'_i of γ', and t can be removed from γ'' without changing the function represented by γ'', contradicting that γ'' is irredundant. So t must contain at least one unnegated literal d_j.

The term t must also contain at least one literal $\overline{a_i}$, because all assignments satisfying γ' set at least one a_i to 0. Also, t must contain at least $m-1$ of the variables in $w_1, \ldots, w_m$. If not, setting the w_j's appearing in the term to 1, the other w_j's to 0, and the other variables so as to satisfy t, would produce an assignment that was satisfying for γ'' but falsifying for γ', contradicting that γ'' and γ' are equivalent.

We have thus shown that every term in R must contain at least $m+1$ literals. In addition, since each term in R must contain at least one literal $\overline{a_i}$ and one literal d_j, we may label each term t in R with a single pair (i, j) such that t contains literals $\overline{a_i}$ and d_j (label each term only once). By the pigeonhole principle, there exists a j^* such that at most $\frac{|R|}{m}$ of the terms in R are labeled (i, j^*) for some i. Let X' be the conjunction of literals a_i such that a term in R is labeled (i, j^*).

We claim that X' is an implicant of ϕ. If not, there exists some assignment b to the variables in ϕ that satisfies X' but falsifies ϕ. Since X is an implicant of ϕ, b must falsify some variable a_{i^*} in X that is not in X'. Extend b so that it sets all w variables to 1, and all variables d_j to 0, except d_{j^*}, which it sets to 1. The resulting assignment b' satisfies γ', because it satisfies $u_i^{j^*} = \overline{a_{i*}} d_{j^*} w_1 \ldots w_m$. However, b' falsifies ϕ', because it falsifies ϕ. Therefore it falsifies each s'_i. It also falsifies all terms t in R, because if t is labeled (i, j) such that $j \neq j^*$, then t contains d_j which is falsified by b', and if t is labeled (i, j^*), then i is in X' and t contains $\overline{a_i}$ which is falsified by b'. This contradicts that γ' and γ'' are equivalent. Thus X' is an implicant of ϕ. Since the smallest implicant of ϕ contains k literals, it follows that $\frac{|R|}{m} \geq k$. Hence R contains at least mk terms, each of which has at least $m+1$ literals. It follows that γ'' has at least $m(k+1)$ terms and at least $q + m(m-1) + mk(m+1)$ occurences of literals. Thus $|f| \geq m(k+1)$ and for $m > k-2$, $|f| \geq q + m(m-1) + mk(m+1) > q + m(m-1) + m(k-1)(m+2)$. □