

# On Learning Counting Functions With Queries

Steven Homer\*  
Boston University

Zhixiang Chen†  
Boston University

December, 1993

## Abstract

We investigate the problem of learning disjunctions of counting functions, which are general cases of parity and modulo functions, with equivalence and membership queries. We prove that, for any prime number  $p$ , the class of disjunctions of integer-weighted counting functions with modulus  $p$  over the domain  $Z_q^n$  (or  $Z^n$ ) for any given integer  $q \geq 2$  is polynomial time learnable using at most  $n + 1$  equivalence queries, where the hypotheses issued by the learner are disjunctions of at most  $n$  counting functions with weights from  $Z_p$ . The result is obtained through learning linear systems over an arbitrary field. In general a counting function may have a composite modulus. We prove that, for any given integer  $q \geq 2$ , over the domain  $Z_2^n$ , the class of read-once disjunctions of Boolean-weighted counting functions with modulus  $q$  is polynomial time learnable with only one equivalence query, and the class of disjunctions of  $\log \log n$  Boolean-weighted counting functions with modulus  $q$  is polynomial time learnable. Finally, we present an algorithm for learning graph-based counting functions.

## 1 Introduction

Recently, symmetric Boolean functions, especially parity functions and modulo functions, have received much attention in computational learning theory. It is known that the class of single parity functions (see Helmbold, Sloan and Warmuth [HRS]) and the class of single modulo functions with modulus  $p$  for any given prime number  $p$  (see, Blum, Chalasani and Jackson [BCJ]) are pac-learnable. Fisher and Simon [FS] proved

---

\*Department of Computer Science, Boston University, Boston, MA 02215; homer@cs.bu.edu. The author was supported by NSF grant CCR-9103055.

†Department of Computer Science, Boston University, Boston, MA 02215; zchen@cs.bu.edu. The author was supported by NSF grant CCR-9103055 and by a Boston University Presidential Graduate Fellowship.

that parity functions of monomials with at most  $k$  literals are pac-learnable, while given the assumption that  $RP \neq NP$  parity functions of  $k$  monomials are not pac-learnable with the same type of functions as hypotheses for any fixed  $k \geq 2$ . Meanwhile, Blum and Singh [BS] showed that, for any constant  $k$ , Boolean functions of  $k$  monomials are pac-learnable by the more expressive hypothesis class of general DNF formulas. They also proved that, for any  $k \geq 2$ , for any fixed symmetric function  $f$  on  $k$  inputs,  $f$  consisting of  $k$  monomials is not pac-learnable with the same type of functions as hypothesis under the assumption that  $RP \neq NP$ .

In the on-line model with queries, Angluin, Hellerstein, and Karpinski [AHK] have shown that read-once Boolean functions over the basis  $(AND, OR, NOT)$  are polynomial time learnable with equivalence and membership queries. Hancock and Hellerstein [HH] extended this result for Boolean functions to a larger basis including arbitrary threshold functions and parity functions. Further, Bshouty, Hancock, and Hellerstein [BHH] showed that read-once functions over the basis of arbitrary symmetric functions are polynomial time learnable with equivalence and membership queries. However, they also proved that read-twice functions over the same basis are not under standard cryptographic assumptions.

Our goal in this paper is to obtain further positive results for on-line learning of counting functions, which include parity and modulo functions, with equivalence and membership queries. Bshouty, Hancock and Hellerstein's negative result for read-twice Boolean functions over the basis of arbitrary symmetric functions is very strong. However, a key condition in their theorem is that they require the basis to include the three-input consensus function, i.e., a function outputs 1 if and only if all its inputs get the same value. However, for many specific symmetric functions, e.g., modulo functions, counting functions, and threshold functions, this condition does not hold, i.e., no one of those functions is equivalent to a consensus function.

We observe that a disjunction of integer-weighted counting functions over a field  $Z_p$  for a given prime number  $p$  corresponds to a linear system over the field  $Z_p$ . We prove that (1) the class of homogeneous linear systems over an arbitrary field is polynomial time learnable with at most  $n$  equivalence queries, and (2) the class of linear systems over an arbitrary field is polynomial time learnable with at most  $n + 1$  equivalence queries. Here  $n$  is the number of input variables, the hypotheses issued in (1) by the learner are homogeneous linear systems of no more than  $n$  equations, and the hypotheses issued in (2) by the learner are also linear systems of no more than  $n$  equations. The first result implies that, for any prime number  $p$ , the class of disjunctions of integer-weighted modulo functions with modulus  $p$  over the field  $Z_p$  is polynomial time learnable with at most  $n$  queries, where the hypotheses issued by the learner are disjunctions of modulo functions with modulus  $p$  and weights from  $Z_p$ . The second result implies that, for any prime number  $p$ , the class of disjunctions of integer-weighted counting functions with modulus  $p$  is polynomial time learnable with at most  $n + 1$  equivalence queries, where the hypotheses issued by the learner are disjunctions of counting functions with modulus  $p$  and weights from  $Z_p$ . We also extend the above results to disjunctions of integer-weighted modulo functions (or in general integer-weighted counting functions) with different prime moduli.

The above results rely on the facts that  $Z_p$  is a field for any prime number  $p$ . When  $p$  is a composite number, however, this is not true. Nevertheless, we prove that, given any integer  $q \geq 2$ , the class of read-once disjunctions of Boolean-weighted counting functions over the domain  $Z_2^n$  is polynomial time learnable with only *one* equivalence query and  $O(n^q)$  membership queries, where  $n$  is the number of input variables. This result cannot be subsumed by Bshouty, Hancock and Hellerstein’s result [BHH] on learning read-once functions over the basis of arbitrary symmetric functions in the sense of the equivalence query complexity, since their result requires at most  $n^3$  equivalence queries. In general, based on analyzing the “modulo-structure” of a disjunction of Boolean-weighted counting functions, we prove that, for any constant  $c$ , over the domain  $Z_2^n$ , the class of disjunctions of no more than  $\log \log n^c$  many Boolean-weighted counting functions with modulus  $q$  for a given integer  $q \geq 2$  is polynomial time learnable with equivalence and membership queries.

Finally we study the problem of learning graph-based counting functions. Graph-based functions, which were initially introduced by Tseitin [T], have played a key role in the study of the complexity of the resolution method. Those functions are in a one-to-one correspondence with graphs, Tseitin proved that the conjunctive normal form representations of the functions based on square grids are hard for the regular resolution method. Since then, these functions have been closely studied by Galil[G], Ben-Ari[B], and Huang etc.[HLC], etc. However, it took almost twenty years to prove that they are hard for the resolution method (see Urquhart[U]). We consider a more general case in which, we not only label edges with independent Boolean variables and assign an index to each vertex as Tseitin originally did, but we also label each vertex with independent Boolean variables and associate a counting function with each vertex. Another extension of Tseitin’s initial definition to hypergraphs was done by Chvátal and Szemerédi [CS]. We prove that functions based on the graphs in which each vertex is labeled with at least  $q$  independent Boolean variables are polynomial time learnable with one equivalence query.

## 2 Preliminaries

We assume that  $Z$  is the set of all integers. For any integer  $n \geq 1$ , let  $V_n$  be the set of variables  $x_1, \dots, x_n$ . Let  $Z_q = \{0, \dots, q-1\}$  for any integer  $q \geq 2$ ,  $Z_q^n = \{0, \dots, q-1\}^n$ . Elements in  $Z_q^n$  are thought of here as  $n \times 1$  vectors. We consider counting functions that consist of variables in  $V_n$ . Our example space will be  $Z^n$  and  $Z_q^n$  for  $q \geq 2$ . When  $q = 2$ ,  $Z_q^n$  is the  $n$ -bit Boolean space. For any positive integer  $q$ , any  $k \in Z_q$ , and any integer vector  $\vec{a} = (a_1, \dots, a_n)^T \in Z^n$ , an integer-weighted counting function  $C_{q,k,\vec{a}}$  with modulus  $q$  is defined as

$$C_{q,k,\vec{a}}(x_1, \dots, x_n) = \begin{cases} 0 & \text{if } \sum_{i=1}^n a_i x_i \equiv k \pmod{q}, \\ 1 & \text{otherwise.} \end{cases}$$

Here we say that  $\vec{a}$  is the integer-weight vector (or weight for short) of  $C_{q,k,\vec{a}}$ . When  $k = 0$ , we say that  $C_{q,0,\vec{a}}$  is an integer-weighted modulo function, and denote it by  $M_{q,\vec{a}}$ .

When  $\vec{a} \in Z_2^n$ , we say that  $C_{q,k,\vec{a}}$  (or  $M_{q,\vec{a}}$ ) is a Boolean-weighted counting (or modulo) function.

For a integer-weighted counting function  $C_{q,k,\vec{a}}$ , let  $\text{vars}(C_{q,k,\vec{a}})$  denote the set of all relevant variables  $x_i$  of  $C_{q,k,\vec{a}}$ , i.e., variables  $x_i$  such that  $a_i \neq 0$ . A disjunction  $F$  of integer-weighted counting functions  $C_{q_1,k_1,\vec{a}_1}, \dots, C_{q_t,k_t,\vec{a}_t}$  is  $C_{q_1,k_1,\vec{a}_1} \vee \dots \vee C_{q_t,k_t,\vec{a}_t}$ . Let  $\text{vars}(F)$  be the set of all relevant variables of  $F$ , i.e., the set  $\text{vars}(C_{q_1,k_1,\vec{a}_1}) \cup \dots \cup \text{vars}(C_{q_t,k_t,\vec{a}_t})$ . If for any  $i, j \in \{1, \dots, t\}$ ,  $i \neq j$  implies that  $\text{vars}(C_{q_i,k_i,\vec{a}_i}) \cap \text{vars}(C_{q_j,k_j,\vec{a}_j}) = \phi$ , then we say that  $F$  is read-once, i.e., each relevant variable of  $F$  occurs in exactly one counting function in  $F$ .

For  $X \in \{Z_q, Z\}$ , an example  $\alpha \in X^n$  satisfies a counting function  $C$  if and only if  $C(\alpha) = 1$ .  $\alpha$  is a positive example for a disjunction  $F$  of counting functions if it satisfies at least one counting function in  $F$  (we write  $F(\alpha) = 1$ ) and a negative example otherwise (we write  $F(\alpha) = 0$ ). For an example  $\alpha \in Z_2^n$ , let  $\alpha[i]$  denote the  $i$ -th bit value of  $\alpha$ , i.e., the value of the variable  $x_i$  in  $\alpha$ . In general, for any literal  $y$ ,  $\alpha[y]$  denotes the value of  $y$  in  $\alpha$ . For  $i \in \{1, \dots, n\}$ ,  $\text{flip}(\alpha, i)$  stands for the example obtained from  $\alpha$  by flipping exactly the  $i$ -th bit value in  $\alpha$ . More generally, for a set  $I \subseteq \{1, \dots, n\}$ , let  $\text{flip}(\alpha, I)$  be the example obtained from  $\alpha$  by flipping the  $i$ -th bit value in  $\alpha$  for every  $i \in I$ . For convenience, we also extend  $\text{flip}$  to act on literals or sets of literals in the following way, when  $l \in \{x_i, \bar{x}_i\}$ , let  $\text{flip}(\alpha, l) = \text{flip}(\alpha, i)$ , and similarly define  $\text{flip}(\alpha, S)$  for a set  $S$  of literals.

Our learning model is the standard model for on-line learning with equivalence and membership queries (see, [Aa]). A learning process for a class  $C$  of Boolean-valued functions over the domain  $X^n$  with the variable set  $V_n$  is viewed as a dialogue between a learner  $A$  and the environment. The goal of the learner is to learn an unknown *target* function  $f \in C$  that has been fixed by the environment. In order to gain information about  $f$  the learner proposes hypothesis function  $h$  from a fixed hypothesis space  $H$  with  $C \subseteq H$ . Whenever  $h \neq f$  for the proposed hypothesis  $h$ , the environment responds with a counterexample  $\alpha \in X^n$  such that  $h(\alpha) \neq f(\alpha)$ . The learner may also ask membership queries for some examples  $\alpha \in X_n$ , to which the environment responds with “yes” if  $f(\alpha) = 1$  or “no” if otherwise. The learner succeeds when he receives “yes” for an equivalence query from the environment, or he can conclude that the current hypothesis is logically equivalent to the target function  $f$ . We assume that the time complexity of asking a membership query for an example is the cost to write it down, and the time complexity of asking an equivalence query for hypothesis  $h$  is the cost to write  $h$  down. We say that  $C$  is polynomial time learnable with equivalence and membership queries, if there is an algorithm for learning any target function  $f \in C$ , using polynomially in  $n$  and the size of  $f$  many equivalence and membership queries, while the time complexity of the algorithm is polynomial in  $n$ , the size of  $f$ , and the size of the largest example that occurred during the learning process.

### 3 Counting Functions via Linear Systems

In this section, we assume that,  $K$  is an arbitrary field; addition and multiplication of two elements in  $K$ , and inversion of a nonzero element in  $K$ , are all of polynomial time complexity. For any positive integer  $n$ ,  $K^n$  is a vector space of dimension  $n$  over the field  $K$ . Every  $\alpha \in K^n$  denotes an  $n \times 1$  vector, and  $\alpha^T$  is the  $1 \times n$  transposition of  $\alpha$ . Let  $\vec{0}_{m,1}$  be an  $m \times 1$  zero-vector,  $\vec{X}_{n,1}$  be an  $n \times 1$  vector of  $n$  variables  $x_1, \dots, x_n$ , where  $x_i$  takes values from  $K$ . For any  $m \times n$  matrix  $A_{m,n}$  and any  $m \times 1$  vector  $\vec{b}_{m,1}$  over  $K$ , a linear system  $L(A_{m,n}, \vec{b}_{m,1})$  of  $m$  linear equations over  $K$  is given as follows,

$$A_{m,n} \vec{X}_{n,1} = \vec{b}_{m,1}.$$

$\alpha \in K^n$  is a solution of the linear system  $L(A_{m,n}, \vec{b}_{m,1})$ , if

$$A_{m,n} \alpha = \vec{b}_{m,1}.$$

When  $\vec{b}_{m,1} = \vec{0}_{m,1}$ , we say that  $L(A_{m,n}, \vec{b}_{m,1})$  is a homogeneous linear system, or homogeneous system for short. For convenience, we write  $L(A_{m,n}) = L(A_{m,n}, \vec{0}_{m,1})$ . The following two general theorems are established.

**Theorem A.** *The class of homogeneous systems over the domain  $K^n$  for any given field  $K$  is polynomial time learnable with at most  $n$  equivalence queries. Moreover, the hypotheses issued by the learner are also homogeneous systems over  $K$  with no more than  $n$  linear equations.*

**Theorem B.** *The class of all linear systems over the domain  $K^n$  for any given field  $K$  is polynomial time learnable with at most  $n + 1$  equivalence queries. Moreover, the hypotheses issued by the learner are also linear systems over the field  $K$  with no more than  $n$  equations.*

From now on in this section we assume that,  $p$  is a given prime number, and  $q \geq 2$  is a given integer. We know that  $Z_p$  is a field with modulo  $p$  addition and multiplication. Note that addition and multiplication of any two numbers in  $Z_p$ , and inversion of any non-zero number in  $Z_p$ , are of  $poly(\log p)$  complexity. Where the length of any number in  $Z_p$  is no more than  $\log p$ . Before we prove the above two general theorems, we first give the following corollaries.

**Corollary A.1.** *Assume  $q \leq p$ . The class of disjunctions of modulo functions  $M_{p,\vec{a}}$  with integer-weights  $\vec{a} \in Z^n$  over the domain  $Z_q^n$  is polynomial time learnable with at most  $n$  equivalence queries, while the hypotheses issued by the learner are disjunctions of at most  $n$  modulo functions  $M_{p,\vec{a}}$  with weights  $\vec{a} \in Z_p^n$ .*

**Corollary A.2.** *Assume  $q > p$ . Given  $X \in \{Z_q, Z\}$ , the class of all disjunctions of modulo functions  $M_{p,\vec{a}}$  with integer-weights  $\vec{a} \in Z^n$  over the domain  $X^n$  is polynomial time learnable with at most  $n$  equivalence queries, while the hypotheses issued by the learner are disjunctions of at most  $n$  modulo functions  $M_{p,\vec{a}}$  with weights  $\vec{a} \in Z_p^n$ .*

**Corollary A.3.** *Given  $X \in \{Z_q, Z\}$  with  $q \geq 2$ . Let  $P = \{p_1, \dots, p_k\}$  be a set of prime numbers. Then, the class of disjunctions of modulo functions  $M_{p,\vec{a}}$  with integer-weights*

$\vec{a} \in Z^n$  and  $p \in P$  over the domain  $X^n$  is polynomial time learnable with at most  $kn$  equivalence queries, while the hypotheses issued by the learner are disjunctions of at most  $kn$  modulo functions  $M_{p,\vec{a}}$  with weights  $\vec{a} \in Z_p^n$  and  $p \in P$ .

**Corollary B.1.** Assume  $q \leq p$ . The class of all disjunctions of counting functions  $C_{p,k,\vec{a}}$  with integer-weights  $\vec{a} \in Z^n$  over the domain  $Z_q^n$  is polynomial time learnable with at most  $n + 1$  equivalence queries, while the hypotheses issued by the learner are disjunctions of at most  $n$  counting functions  $C_{p,k,\vec{a}}$  with integer-weights  $\vec{a} \in Z_p^n$ .

**Corollary B.2.** Given  $X \in \{Z_q, Z\}$  with  $q > p$ . The class of all disjunctions of counting functions  $C_{p,k,\vec{a}}$  with integer-weights  $\vec{a} \in Z^n$  over the domain  $X^n$  is polynomial time learnable with at most  $n + 1$  equivalence queries, while the hypotheses issued by the learner are disjunctions of at most  $n$  counting functions  $C_{p,k,\vec{a}}$  with weights  $\vec{a} \in Z_p^n$ .

**Corollary B.3.** Given  $X \in \{Z_q, Z\}$  with  $q \geq 2$ . Let  $P = \{p_1, \dots, p_k\}$  be a set of prime numbers. Then, the class of disjunctions of counting functions  $C_{p,k,\vec{a}}$  with integer-weights  $\vec{a} \in Z^n$  and  $p \in P$  over the domain  $X^n$  is polynomial time learnable with at most  $k(n + 1)$  equivalence queries, while the hypotheses issued by the learner are disjunctions of at most  $kn$  counting functions  $C_{p,k,\vec{a}}$  with weights  $\vec{a} \in Z_p^n$  and  $p \in P$ .

We now prove our theorems and corollaries.

**Proof of Theorem A.** Assume that  $L(A_{m,n})$  is the target system. Let  $I_{l,l}$  be the  $l \times l$  identity matrix over  $K$ . Let  $S_r$  be the set of all solutions received during the first  $r$  stages, the learning algorithm Learn-HS (where ‘‘HS’’ stands for ‘‘homogeneous system’’) is given as follows.

**Learn-HS:**

Stage 1. Set the first hypothesis  $H_1 = L(I_{n,n})$ . Ask an equivalence query for  $H_1$ . If the learner receives ‘‘yes’’ then stop, otherwise he receives a non-zero solution  $\vec{\alpha}_1 \in K^n$  to  $L(A_{m,n})$ . Let  $S_1 = \{\vec{\alpha}_1\}$ .

Stage  $r \geq 2$ . Let  $S_{r-1} = \{\vec{\alpha}_1, \dots, \vec{\alpha}_{r-1}\}$ . Construct from vectors in  $S_{r-1}$  a matrix  $B_{n-(r-1),n}$  such that the set of all solutions of the homogeneous system  $L(B_{n-(r-1),n})$  is  $\text{span}(S_{r-1}) = \{t_1\vec{\alpha}_1 + \dots + t_{r-1}\vec{\alpha}_{r-1} \mid t_i \in K, 1 \leq i \leq r-1\}$ . Set the  $r$ -th hypothesis  $H_r = L(B_{n-(r-1),n})$ . If  $r = n + 1$ , the learner concludes that  $H_r$  is equivalent to  $L(A_{m,n})$  so stop. When  $r \leq n$ , ask an equivalence query for  $H_r$ , if ‘‘yes’’ then stop, otherwise the learner receives a solution  $\vec{\alpha}_r$  which is outside  $\text{span}(S_{r-1})$ . Set  $S_r = S_{r-1} \cup \{\vec{\alpha}_r\}$ .

**End of Learn-HS.**

**Claim 3.1.** At any stage  $r$  with  $1 < r \leq n + 1$ , the following holds: (1) vectors in  $S_{r-1}$  are linearly independent; (2) the matrix  $B_{n-(r-1),n}$  exists; (3)  $\text{span}(S_{r-1})$  is the set of all solutions of  $H_r$ ; (4) every vector in  $\text{span}(S_{r-1})$  is a solution of the target system.

**Proof of Claim 3.1.** By induction on  $r$ . When  $r = 2$ ,  $S_1$  contains exactly one nonzero solution  $\vec{\alpha}_1$  of the target system  $L(A_{m,n})$ , so it is trivial that vectors in  $S_1$  are

linearly independent, and every vector in  $\text{span}(S_1)$  is a solution of  $L(A_{m,n})$ . Since  $\vec{\alpha}_1$  is nonzero, we may assume without loss of generality that the first element in it is not 0. Let  $\vec{\alpha}_1 = (a_{11}, a_{21}, \dots, a_{n1})^T$ . Since  $K$  is a field,  $a_{11} \neq 0$  implies the inverse  $a_{11}^{-1}$  exists. Let  $D_{n-1,1} = (a_{21}, \dots, a_{n1})^T$ , define the matrix

$$B_{n-1,n} = \left( -D_{n-1,1} \begin{pmatrix} a_{11}^{-1} \end{pmatrix}, E_{n-1,n-1} \right).$$

Then,  $B_{n-1,n}$  has rank  $n-1$ . By simple calculation,  $B_{n-1,n} \vec{\alpha}_1 = \vec{0}_{n-1,1}$ . Thus,  $\text{span}(S_1)$  is exactly the set of all solutions of the system  $L(B_{n-1,n})$ . Hence, our claim holds for  $r = 2$ .

Assume our claim is true for any  $r$  with  $1 < r \leq k < n + 1$ . At stage  $k + 1$ , by the induction assumption, we know that, vectors in  $S_{k-1}$  are linearly independent, vectors in  $\text{span}(S_{k-1})$  are solutions of the target system, and  $\text{span}(S_{k-1})$  is the set of all solutions of the hypothesis  $H_k$ . Thus, when the learner receives a counterexample  $\vec{\alpha}_k$  for  $H_k$ , then  $\vec{\alpha}_k$  is a solution of the target system outside  $\text{span}(S_{k-1})$ , this implies that  $\vec{\alpha}_k$  is linearly independent from vectors in  $S_{k-1}$ . Hence, vectors in  $S_k = S_{k-1} \cup \{\vec{\alpha}_k\}$  are linearly independent and vectors in  $\text{span}(S_k)$  are solutions of the target system. Let the matrix  $Q_{n,k} = (\vec{\alpha}_1, \dots, \vec{\alpha}_k)$ , since  $K$  is a field, we may assume without loss of generality that the submatrix  $G_{k,k}$  consisting of elements on the first  $k$  rows in  $Q_{n,k}$  has an inverse  $G_{k,k}^{-1}$ . Let  $N_{n-k,k}$  be the submatrix consisting of elements on the last  $n - k$  rows in  $Q_{n,k}$ . Define the matrix

$$B_{n-k,n} = \left( -N_{n-k,k} G_{k,k}^{-1}, E_{n-k,n-k} \right).$$

Then,  $B_{n-k,n}$  has rank  $n - k$ , and  $B_{n-k,n} Q_{n,k} = \vec{0}_{n-k,k}$ . Thus,  $\text{span}(S_k)$  is the set of all solutions of the system  $L(B_{n-k,n})$ . Combing the above analysis, our claim holds.  $\square$

By the above claim, at any stage  $r$  with  $2 \leq r \leq n$ , either the learner learns the target system, or receives a solution of the target system which is linearly independent from the solutions in  $S_{r-1}$ . Since the target system has at most  $n$  linearly independent solutions, the learner learns it with at most  $n$  equivalence queries.

Let  $N$  be the size of the longest element in any counterexamples received by the learner during the learning process. By the assumption that, addition and multiplication of any two elements in  $K$ , and inversion of any element in  $K$ , are of polynomial time complexity, one can find at stage  $r$  the matrix  $B_{n-(r-1),n}$  in time polynomial in  $n$  and  $N$ . So, the total time complexity of the algorithm Learn-HS is polynomial in  $n$  and  $N$ .  $\square$

**Proof of Theorem B.** Assume that  $L(A_{m,n}, \vec{b}_{m,1})$  is the target system. Let  $I_{l,l}$  be the  $l \times l$  identity matrix over  $K$ . The learning algorithm Learn-IHS works in stages. Learn-IHS is given on the next page.

**Claim 3.2.** *At any stage  $r$  with  $1 < r \leq n + 1$ , the following holds: (1) vectors in  $S_{r-1}$  are linearly independent; (2) the matrix  $B_{n-(r-1),n}$  exists; (3)  $\text{span}(S_{r-1})$  is the set of all solutions of the homogeneous system  $L(B_{n-(r-1),n})$ , and every vector  $\vec{a} \in \text{span}(S_{r-1})$  is a solution of the homogeneous system  $L(A_{m,n})$ ; (4) finally,  $\text{span}(S_{r-1}) = \{\vec{a} + \vec{\alpha}_0\}$*

**Learn-IHS:**

*Stage 0.* Choose a matrix  $B_{n,n}$  and a vector  $\vec{d}_{n,1}$  over  $K$  such that the rank of  $B_{n,n}$  is different from that of the matrix  $(B_{n,n}, \vec{d}_{n,1})$ . Ask an equivalence query for the hypothesis  $H_0 = L(B_{n,n}, \vec{d}_{n,1})$ . Note that  $H_0$  has no solutions. If the learner receives “yes” then stop, otherwise he receives a solution  $\vec{\alpha}_0$  for the target system. Set  $S_0 = \phi$ .

*Stage 1.* Set the hypothesis  $H_1 = L(I_{n,n}, \vec{\alpha}_0)$ . Ask an equivalence query for  $H_1$ . If “yes” then stop, otherwise the learner receives a solution  $\vec{\alpha}_1 \in K^n$  to  $L(A_{m,n}, \vec{b}_{m,1})$  other than  $\vec{\alpha}_0$ . Let  $S_1 = \{\vec{\alpha}_1 - \vec{\alpha}_0\}$ .

*Stage  $r \geq 2$ .* Let  $S_{r-1} = \{\vec{\alpha}_1 - \vec{\alpha}_0, \dots, \vec{\alpha}_{r-1} - \vec{\alpha}_0\}$ . Construct from vectors in  $S_{r-1}$  a matrix  $B_{n-(r-1),n}$  such that the set of all solutions of the homogeneous system  $L(B_{n-(r-1),n})$  is  $\text{span}(S_{r-1}) = \{t_1(\vec{\alpha}_1 - \vec{\alpha}_0) + \dots + t_{r-1}(\vec{\alpha}_{r-1} - \vec{\alpha}_0) | t_i \in K, 1 \leq i \leq r-1\}$ . Set the  $r$ -th hypothesis  $H_r = L(B_{n-(r-1),n}, B_{n-(r-1),n}, \vec{\alpha}_0)$ . If  $r = n+1$ , the learner concludes that  $H_r$  is equivalent to  $L(A_{m,n})$ , so stop. If  $r \leq n$ , ask an equivalence query for  $H_r$ , if “yes” then stop, otherwise the learner receives a solution  $\vec{\alpha}_r$ . Set  $S_r = S_{r-1} \cup \{\vec{\alpha}_r - \vec{\alpha}_0\}$ .

**End of Learn-IHS.**

$\vec{\alpha} \in \text{span}(S_{r-1})\}$  is the set of all solutions of the hypothesis  $H_r = L(B_{n-(r-1),n}, B_{n-(r-1),n}, \vec{\alpha}_0)$ , and any  $\vec{\alpha} \in \text{Ispan}(S_r)$  is a solution of the target system  $L(A_{m,n}, \vec{b}_{m,1})$ .

**Proof of Claim 3.2.** By induction on  $r$ . When  $r = 2$ ,  $S_1$  contains exactly one nonzero solution  $\vec{\alpha}_1 - \vec{\alpha}_0$  of the homogeneous system  $L(A_{m,n})$ , since both  $\vec{\alpha}_1$  and  $\vec{\alpha}_0$  are solutions to the target system. So it is trivial that vectors in  $S_1$  are linearly independent, i.e., (1) is true.

Since by (1)  $\vec{\alpha}_1 - \vec{\alpha}_0$  is linearly independent, we may assume without loss of generality that the first element in it is not 0. Let  $\vec{\alpha}_1 - \vec{\alpha}_0 = (a_{11}, a_{21}, \dots, a_{n1})^T$ . Since  $K$  is a field,  $a_{11} \neq 0$  implies the inverse  $a_{11}^{-1}$  exists. Let  $D_{n-1,1} = (a_{21}, \dots, a_{n1})^T$ , we can choose the matrix  $B_{n-1,n}$  as follows

$$B_{n-1,n} = \left( -D_{n-1,1} \left( a_{11}^{-1} \right), I_{n-1,n-1} \right).$$

This implies (2).

Note that  $B_{n-1,n}$  has rank  $n-1$ . By simple calculation,  $B_{n-1,n}(\vec{\alpha}_1 - \vec{\alpha}_0) = \vec{0}_{n-1,1}$ . Thus,  $\text{span}(S_1)$  is exactly the set of all solutions of the system  $L(B_{n-1,n})$ . Since each vector in  $S_{r-1}$  is a solution to  $L(A_{m,n})$ , so are vectors in  $\text{span}(S_{r-1})$ . Thus, (3) is true.

Note that  $\vec{\alpha}_0$  is a solution to  $L(B_{n-1,n}, B_{n-1,n}, \vec{\alpha}_0)$ . By (3),  $\text{Ispan}(S_1)$  is the set of all solutions of  $H_2 = L(B_{n-1,n}, B_{n-1,n}, \vec{\alpha}_0)$ , and every vector in  $\text{Ispan}(S_1)$  is a solution to the target system. Hence, (4) is true.

Assume our claim is true for any  $r$  with  $1 < r \leq k < n+1$ . At stage  $k+1$ , by the induction assumption, we know that, vectors in  $S_{k-1}$  are linearly independent,

$\text{span}(S_{k-1})$  is the set of all solutions of the hypothesis  $H_k$ , and vectors in  $\text{Ispan}(S_{k-1})$  are solutions of the target system. Thus, when the learner receives a counterexample  $\vec{\alpha}_k$  for  $H_k$ , then  $\vec{\alpha}_k$  is a solution of the target system outside  $\text{Ispan}(S_{k-1})$ , this implies that  $\vec{\alpha}_k - \vec{\alpha}_0$  is linearly independent from vectors in  $S_{k-1}$ . Hence, vectors in  $S_k = S_{k-1} \cup \{\vec{\alpha}_k - \vec{\alpha}_0\}$  are linearly independent, i.e., (1) is true.

Let the matrix  $Q_{n,k} = ((\vec{\alpha}_1 - \vec{\alpha}_0), \dots, (\vec{\alpha}_k - \vec{\alpha}_0))$ . Since  $K$  is a field, we may assume without loss of generality that the submatrix  $G_{k,k}$  consisting of elements in the first  $k$  rows in  $Q_{n,k}$  has an inverse  $G_{k,k}^{-1}$ . Let  $N_{n-k,k}$  be the submatrix consisting of elements on the last  $n-k$  rows in  $Q_{n,k}$ . The matrix  $B_{n-k,n}$  exists, actually we can choose it as

$$B_{n-k,n} = \begin{pmatrix} -N_{n-k,k}G_{k,k}^{-1} & I_{n-k,n-k} \end{pmatrix}.$$

Hence, (2) is true.

$B_{n-k,n}$  has rank  $n-k$ , and  $B_{n-k,n}Q_{n,k} = \vec{0}_{n-k,k}$ . Thus, by (1),  $\text{span}(S_k)$  is the set of all solutions of the homogeneous system  $L(B_{n-k,n})$ , and each vector in  $\text{span}(S_k)$  is a solution to  $L(A_{m,n})$ . This implies that (3) is true.

Note that  $\vec{\alpha}_0$  is a solution to  $L(B_{n-k,n}, B_{n-k,n}\vec{\alpha}_0)$ . By (3),  $\text{Ispan}(S_1)$  is the set of all solutions of  $H_{k+1} = L(B_{n-k,n}, B_{n-k,n}\vec{\alpha}_0)$ , and every vector in  $\text{Ispan}(S_k)$  is a solution to the target system. Hence, (4) is true.  $\square$

By Claim 3.2, at any stage  $r$  with  $1 < r \leq n$ , either the learner learns the target system, or receives a solution  $\vec{\alpha}_r$  of the target system such that  $\vec{\alpha}_r - \vec{\alpha}_0$  is linearly independent from the solutions in  $S_r$ . Since the homogeneous system  $L(A_{m,n})$  of the target system has at most  $n$  linearly independent solutions, so the learner learns  $L(A_{m,n})$  (and hence  $L(A_{m,n}, \vec{b}_{m,1})$ ) with at most  $n+1$  equivalence queries. Since addition and multiplication of any two elements in  $K$ , and inversion of any nonzero element in  $K$ , are of polynomial time complexity, at any stage  $r$ , one can find the matrix  $B_{n-(r-1),n}$  in time polynomially in  $n$  and the size of the longest element in any vectors received during the first  $r$  stages. Thus, the time complexity of the algorithm Learn-IHS is polynomial in  $n$  and the size of the longest element in vectors received during the learning process.  $\square$

**Proof of Corollary A.1.** Assume  $F = M_{p,\vec{a}_1} \vee \dots \vee M_{p,\vec{a}_t}$  is the target function. For the integer-weight  $\vec{a}_i = (a_{i1}, \dots, a_{in})^T$  of  $M_{p,\vec{a}_i}$ , let  $\vec{b}_i = (a_{i1} \bmod p, \dots, a_{in} \bmod p)^T$ . Then,  $\vec{b}_i \in Z_p^n$ . It is easy to see that  $F$  is equivalent to the function

$$F^* = M_{p,\vec{b}_1} \vee \dots \vee M_{p,\vec{b}_t}.$$

Hence, in order to learn  $F$ , one only need to learn  $F^*$ . Define a matrix

$$A_{t,n} = \begin{pmatrix} (\vec{b}_1)^T \\ \vdots \\ (\vec{b}_t)^T \end{pmatrix}.$$

Then,  $F^*$  (and hence  $F$ ) is equivalent to the homogeneous system over the domain  $Z_q^n$

$$A_{t,n}\vec{X}_{n,1} = \vec{0}_{t,1}$$

in the sense that, for any vector  $\vec{x} \in Z_q^n$ ,  $F^*(\vec{x}) = 0$  if and only if  $\vec{x}$  is a solution of the above system. Note also that for a vector  $\vec{d} \in Z_p^n$ , a linear equation  $(\vec{d})^T \cdot \vec{X}_{n,1} \equiv 0 \pmod{p}$  is equivalent to the modulo function  $M_{p,\vec{d}}$ . Therefore, our corollary follows from Theorem A and Lemma 3.3.  $\square$

**Lemma 3.3.** *Assume  $q \leq p$ . Let  $L(B_{s,n})$  be a given homogeneous system over the domain  $Z_p^n$  with modulo  $p$  addition and multiplication. Assume that  $S = \{\vec{\alpha}_1, \dots, \vec{\alpha}_r\}$  is a set of linearly independent vectors in  $Z_p^n$  such that  $\text{span}(S) = \{k_1\vec{\alpha}_1 + \dots + k_r\vec{\alpha}_r \mid k_i \in Z_p, 1 \leq i \leq r\}$  is the set of all solutions of  $L(B_{s,n})$  over the domain  $Z_p^n$ . Then, the set of all solutions of  $L(B_{s,n})$  over the domain  $Z_q^n$  is  $R\text{span}(S) = \text{span}(S) \cap Z_q^n$ .*

**Proof of Lemma 3.3.** It is obvious that any vector in  $R\text{span}(S)$  is a solution of  $L(B_{s,n})$  over  $Z_q^n$ . Suppose that  $\vec{b}$  is a solution of  $L(B_{s,n})$  over  $Z_q^n$ , then it is also a solution of  $L(B_{s,n})$  over  $Z_p^n$ , since  $Z_q^n \subseteq Z_p^n$ . Hence,  $\vec{b} \in R\text{span}(S)$ .  $\square$

**Proof of Corollary A.2.** In a similar way as one did in the proof of corollary A.1, this corollary follows from Theorem A and Lemma 3.4.  $\square$

**Lemma 3.4.** *Given  $X \in \{Z_q, Z\}$  with  $q > p$ . Let  $L(B_{s,n})$  be a homogeneous system over the domain  $X^n$  with modulo  $p$  addition and multiplication. Assume that  $S = \{\vec{\alpha}_1, \dots, \vec{\alpha}_r\}$  is a set of linearly independent vectors in  $Z_p^n$  such that  $\text{span}(S) = \{k_1\vec{\alpha}_1 + \dots + k_r\vec{\alpha}_r \mid k_i \in Z_p, 1 \leq i \leq r\}$  is the set of all solutions of  $L(B_{s,n})$  over the domain  $Z_p^n$ . Then, the set of all solutions of  $L(B_{s,n})$  over the domain  $X^n$  is  $E\text{span}(S) = \{\vec{\alpha} + (q_1p, \dots, q_np)^T \mid \vec{\alpha} \in \text{span}(S); q_j \in X, 1 \leq j \leq n\} \cap X^n$ .*

**Proof of Lemma 3.4.** It is obvious that any vector in  $E\text{span}(S)$  is a solution of  $L(B_{s,n})$  over  $X^n$ . Suppose that  $\vec{b} = (b_1, \dots, b_n)^T \in X^n$  is a solution of  $L(B_{s,n})$ . Let  $b_j = d_j + q_jp$ , where  $d_j \in Z_p, q_j \in X$ . Let  $\vec{d} = (d_1, \dots, d_n)^T$ , then  $\vec{d} \in Z_p^n$ , and  $\vec{d}$  is a solution of  $L(B_{s,n})$  because  $\vec{b}$  is. Since  $\text{span}(S)$  is the set of all solutions of  $L(B_{s,n})$  over  $Z_p^n$ , there exist  $k_i \in Z_p$  for  $1 \leq i \leq r$  such that  $\vec{d} = k_1\vec{\alpha}_1 + \dots + k_r\vec{\alpha}_r$ . Thus,  $\vec{b} = k_1\vec{\alpha}_1 + \dots + k_r\vec{\alpha}_r + (q_1p, \dots, q_np)^T \in E\text{span}(S)$ .  $\square$

**Proof of Corollary A.3.** Assume that

$$F = M_{p_1, \vec{a}_{11}} \vee \dots \vee M_{p_1, \vec{a}_{1t_1}} \vee \dots \vee M_{p_k, \vec{a}_{k1}} \vee \dots \vee M_{p_k, \vec{a}_{kt_k}}$$

is a target function. For the integer-weight  $\vec{a}_{ij} = (b_{ij1}, \dots, b_{ijn})^T$  of  $M_{p_i, \vec{a}_{ij}}$ , let

$$\vec{a}_{ij}^* = (b_{ij1} \bmod p_i, \dots, b_{ijn} \bmod p_i)^T.$$

Then,  $\vec{a}_{ij}^* \in Z_{p_i}^n$ . It is easy to see that  $F$  is equivalent to the function

$$F^* = M_{p_1, \vec{a}_{11}^*} \vee \dots \vee M_{p_1, \vec{a}_{1t_1}^*} \vee \dots \vee M_{p_k, \vec{a}_{k1}^*} \vee \dots \vee M_{p_k, \vec{a}_{kt_k}^*}.$$

Hence, in order to learn  $F$ , one only needs to learn  $F^*$ . Define the matrices,

$$A_{i,n}^i = \begin{pmatrix} (\vec{a}_{i1}^*)^T \\ \vdots \\ (\vec{a}_{it_i}^*)^T \end{pmatrix}, \quad i = 1, \dots, k.$$

Then,  $F^*$  (and hence  $F$ ) is equivalent to the “conjunction” of the homogeneous systems

$$A_{t_i, n}^i \vec{X}_{n,1} = \vec{0}_{t_i,1}, \quad i = 1, \dots, k,$$

over the domain  $X^n$  with modulo  $p_i$  addition and multiplication in the sense that, for any vector  $\vec{u} \in X^n$ ,  $F^*(\vec{u}) = 0$  if and only if  $\vec{u}$  is a solution for each of the above systems. Note also that for a vector  $\vec{u} \in Z^n$ , a linear equation  $(\vec{u})^T \cdot \vec{X}_{n,1} \equiv 0 \pmod{p_i}$  is equivalent to the modulo function  $M_{p_i, \vec{u}}$  with the integer-weight  $\vec{u}$ .

One then learns  $F^*$  (hence  $F$ ) through learning  $L(A_{t_i, n}^i)$ , for  $i = 1, \dots, k$ , simultaneously. At each stage, let  $H_i$  be the hypothesis for  $L(A_{t_i, n}^i)$ ,  $i = 1, \dots, k$ . In other word,  $H_i$  is a hypothesis for

$$M_{p_i, \vec{a}_{t_i}^*} \vee \dots \vee M_{p_i, \vec{a}_{t_i}^*}.$$

One sets  $H = H_1 \vee \dots \vee H_k$  to be the hypothesis for  $F^*$ . According to Corollaries A.1 and A.2, one can learn each of the systems  $L(A_{t_i, n}^i)$  with at most  $n$  equivalence queries, and the hypotheses issued by the learner are homogeneous systems with weights from  $Z_{p_i}$ . When one receives a counterexample for the hypothesis  $H$ , one can derive from this counterexample a new linearly independent vector (i.e., solution) for at least one of the systems  $L(A_{t_i, n}^i)$ . Thus, with at most  $kn$  equivalence queries one can learn  $F^*$ . Since by Corollary A.1 and A.2 the time complexity for learning each of the systems  $L(A_{t_i, n}^i)$  is polynomial in  $n$  and the largest size of elements in vectors received by the learner during the learning process, so the time complexity for learning  $F^*$  is  $kP(n, N)$ , where  $P$  is a polynomial and  $N$  is the size of the largest element in any vectors received by the learner.  $\square$

**Proof of Corollary B.1.** Assume  $F = C_{p, k_1, \vec{a}_1} \vee \dots \vee C_{p, k_t, \vec{a}_t}$  is the target function. Our proof is similar to that of Corollary A.1. But, instead of modulo functions  $M_{p, \vec{a}_i}$ , we consider counting functions  $C_{p, k_i, \vec{a}_i}$ ,  $i = 1, \dots, t$ . In the same manner as we did for Corollary A.1, we obtain a matrix  $A_{t, n}$ . Let  $R_{t,1} = (k_1, \dots, k_t)^T$ . Then,  $F$  is equivalent to the linear system over the domain  $Z_q^n$

$$A_{t, n} \vec{X}_{n,1} = \vec{R}_{t,1}.$$

Therefore, our corollary follows from Theorem B and Lemma 3.5.  $\square$

**Lemma 3.5.** *Assume  $q \leq p$ . Let  $L(B_{s, n}, B_{s, n} \vec{b}_{s,1})$  be a given linear system over the domain  $Z_p^n$ . Assume that  $S = \{\vec{\alpha}_1, \dots, \vec{\alpha}_r\}$  is a set of linearly independent vectors in  $Z_p^n$  such that  $\text{span}(S) = \{k_1 \vec{\alpha}_1 + \dots + k_r \vec{\alpha}_r \mid k_i \in Z_p, 1 \leq i \leq r\}$  is the set of all solutions of  $L(B_{s, n})$  over the domain  $Z_p^n$ . Then, the set of all solutions of  $L(B_{s, n}, B_{s, n} \vec{b}_{s,1})$  over the domain  $Z_q^n$  is  $R\text{span}(S) = \text{span}(S) \cap Z_q^n$ , and the set of all solutions of  $L(B_{s, n}, B_{s, n} \vec{b}_{s,1})$  over the domain  $Z_q^n$  is  $IR\text{span}(S) = I\text{span}(S) \cap Z_q^n$ , where  $I\text{span}(S) = \{\vec{\alpha} + \vec{b}_{s,1} \mid \vec{\alpha} \in \text{span}(S)\}$ .*

**Proof of Lemma 3.5.** It is obvious that any vector in  $R\text{span}(S)$  is a solution of  $L(B_{s, n})$  over  $Z_q^n$ , and any vector in  $IR\text{span}(S)$  is a solution of  $L(B_{s, n}, \vec{b}_{s,1})$  over  $Z_q^n$ . Suppose that  $\vec{f}$  is a solution of  $L(B_{s, n})$  over  $Z_q^n$ , then it is also a solution of  $L(B_{s, n})$  over  $Z_p^n$ , since  $Z_q^n \subseteq Z_p^n$ . Hence,  $\vec{b} \in R\text{span}(S)$ . When  $\vec{g}$  is a solution of  $L(B_{s, n}, B_{s, n} \vec{b}_{s,1})$  over

$Z_q^n$ , then  $\vec{g} - \vec{b}_{s,1}$  is a solution of  $L(A_{m,n})$  over the domain  $Z_p^n$ . Thus,  $\vec{g} - \vec{b}_{s,1} \in \text{span}(S)$ , which implies  $\vec{g} \in \text{span}(S)$ . Hence,  $\vec{g} \in \text{IRspan}(S)$ .  $\square$

**Proof of Corollary B.2.** This corollary follows from Theorem B and the following lemma in a manner similar to corollary B.1.  $\square$

**Lemma 3.6.** *Given  $X \in \{Z_q, Z\}$  with  $q > p$ . Let  $L(B_{s,n}, B_{s,n}\vec{b}_{s,1})$  be a linear system over the domain  $X^n$  with modulo  $p$  addition and multiplication. Assume  $S = \{\vec{\alpha}_1, \dots, \vec{\alpha}_r\}$  is a set of linearly independent vectors in  $Z_p^n$  such that  $\text{span}(S) = \{k_1\vec{\alpha}_1 + \dots + k_r\vec{\alpha}_r \mid k_i \in Z_p, 1 \leq i \leq r\}$  is the set of all solutions of  $L(B_{s,n})$  over the domain  $Z_p^n$ . Then, the set of all solutions of  $L(B_{s,n})$  over the domain  $X^n$  is  $\text{Espan}(S) = \{\vec{\alpha} + (q_1p, \dots, q_np)^T \mid \vec{\alpha} \in \text{span}(S); q_j \in X, 1 \leq j \leq n\} \cap X^n$ , and the set of all solutions of  $L(B_{s,n}, B_{s,n}\vec{b}_{s,1})$  is  $\text{IEspan}(S) = \{\vec{f} + \vec{b}_{s,1}^* \mid \vec{f} \in \text{Espan}(S)\}$ . Here,  $b_{s,1} = (b_1, \dots, b_s)^T$ ,  $b_{s,1}^* = (b_1 \bmod p, \dots, b_s \bmod p)^T$ .*

**Proof.** It is obvious that any vector in  $\text{Espan}(S)$  is a solution of  $L(B_{s,n})$  over  $X^n$ , and any vector in  $\text{IEspan}(S)$  is a solution of  $L(B_{s,n}, B_{s,n}\vec{b}_{s,1})$  over  $X^n$ . Suppose that  $\vec{f} = (f_1, \dots, f_n)^T \in X^n$  is a solution of  $L(B_{s,n})$ . Let  $f_j = d_j + q_jp$ , where  $d_j \in Z_p, q_j \in X$ . Let  $\vec{d} = (d_1, \dots, d_n)^T$ , then  $\vec{d} \in Z_p^n$ , and  $\vec{d}$  is a solution of  $L(B_{s,n})$  because  $\vec{f}$  is. Since  $\text{span}(S)$  is the set of all solutions of  $L(B_{s,n})$  over  $Z_p^n$ , there exist  $k_i \in Z_p$  for  $1 \leq i \leq r$  such that  $\vec{d} = k_1\vec{\alpha}_1 + \dots + k_r\vec{\alpha}_r$ . Thus,  $\vec{f} = k_1\vec{\alpha}_1 + \dots + k_r\vec{\alpha}_r + (q_1p, \dots, q_np)^T \in \text{Espan}(S)$ . Similarly, when  $\vec{g} \in X^n$  is a solution of  $L(B_{s,n}, B_{s,n}\vec{b}_{s,1})$ ,  $\vec{g} \in \text{IEspan}(S)$ .  $\square$

**Proof of Corollary B.3.** Assume that

$$F = C_{p_1, k_{11}, \vec{a}_{11}} \vee \dots \vee C_{p_1, k_{1t_1}, \vec{a}_{1t_1}} \vee \dots \vee C_{p_s, k_{s1}, \vec{a}_{s1}} \vee \dots \vee C_{p_s, k_{st_s}, \vec{a}_{st_s}}$$

is a target function. Instead of modulo functions  $M_{p_i, \vec{a}_{ij}}$  in the proof of Corollary A.3, we now consider counting functions  $C_{p_i, k_{ij}, \vec{a}_{ij}}$ . Thus, we obtain matrices  $A_{t_i, n}^i$  in the same manner. Define

$$\vec{R}_{t_i, 1} = (k_{11}, \dots, k_{it_i})^T, i = 1, \dots, s.$$

Then, F is equivalent to the ‘‘conjunction’’ of the linear systems,

$$A_{t_i, n}^i \vec{X}_{n,1} = \vec{R}_{t_i, 1}, i = 1, \dots, s,$$

over the domain  $X^n$  with modulo  $p_i$  addition and multiplication in the sense that, for any vector  $\vec{u} \in X^n$ ,  $F^*(\vec{u}) = 0$  if and only if  $\vec{u}$  is a solution to each of the above systems. Hence, our corollary follows from Corollaries B.1 and B.2, with the similar analysis as we did in the proof of Corollary A.3.  $\square$

## 4 Read-Once Disjunctions of Counting Functions

As argued in [BHH], it is reasonable to believe that an equivalence query is more expensive than a membership query. A practically ideal learning algorithm will use as few equivalence queries as possible. We will design a learning algorithm for the class of

read-once disjunctions of Boolean-weighted counting functions over the domain  $Z_2^n$  that requires only *one* (it is not hard to see that this is also the lower bound) equivalence query. Previous work ([BHH]) shows that this class can be learned using equivalence and membership queries, but the bound on the number of equivalence queries is  $n^3$ . In the following, we assume that  $q \geq 2$  is a given integer,  $F = C_{q,k_1,\vec{a}_1} \vee \cdots \vee C_{q,k_t,\vec{a}_t}$  is a disjunction of counting functions with Boolean-weights  $\vec{a}_i \in Z_2^n$ ,  $i = 1, \dots, t$ . We also assume that  $\alpha$  is a negative counterexample for  $F$ .

**Lemma 4.1.** *For any variable  $x \in \text{vars}(C_{q,k_i,\vec{a}_i})$ ,  $F(\text{flip}(\alpha, x)) = C_{q,k_i,\vec{a}_i}(\text{flip}(\alpha, x)) = 1$ ,  $i = 1, \dots, t$ .*

**Proof.** Since  $\alpha$  is a negative example for  $F$ ,  $C_{q,k_i,\vec{a}_i}(\alpha) = 0$ . This implies that

$$S = \sum_{x \in \text{vars}(C_{q,k_i,\vec{a}_i})} \alpha[x] \equiv k_i \pmod{q}.$$

Hence, for any  $x \in \text{vars}(C_{q,k_i,\vec{a}_i})$ , after flipping  $x$  in  $\alpha$ , the original sum  $S$  modulo  $q$  then becomes either  $k_i + 1$  or  $k_i - 1$ , so  $F(\text{flip}(\alpha, x)) = C_{q,k_i,\vec{a}_i}(\text{flip}(\alpha, x)) = 1$ .  $\square$

**Lemma 4.2.**  $\text{vars}(F) = \{x \in V_n \mid F(\text{flip}(\alpha, x)) = 1\}$ .

**Proof.** On the one hand, by Lemma 4.1,  $\text{vars}(F) = \bigcup \{\text{vars}(C_{q,k_i,\vec{a}_i}) \mid i = 1, \dots, t\} \subseteq \{x \in V_n \mid F(\text{flip}(\alpha, x)) = 1\}$ . On the other hand, for any variable  $y \in \{x \in V_n \mid F(\text{flip}(\alpha, x)) = 1\}$ , we have  $C_{q,k_j,\vec{a}_j}(\text{flip}(\alpha, y)) = 1$  for some  $j \in \{1, \dots, t\}$ . Note again that  $C_{q,k_j,\vec{a}_j}(\alpha) = 0$ , since  $\alpha$  is a negative example. Thus,  $y \in \text{vars}(C_{q,k_j,\vec{a}_j})$  and,  $\text{vars}(F) = \{x \in V_n \mid F(\text{flip}(\alpha, x)) = 1\}$ .  $\square$

**Lemma 4.3.** *For any two distinct variables  $u, v \in \text{vars}(C_{q,k_i,\vec{a}_i})$ , for any  $w \notin \text{vars}(C_{q,k_i,\vec{a}_i})$ ,  $1 \leq i \leq t$ , we have (1)  $F(\text{flip}(\alpha, \{u, w\})) = 1$  and, (2)  $F(\text{flip}(\alpha, \{u, v\})) = 0$  if  $\alpha[u] \neq \alpha[v]$ .*

**Proof.** It follows from  $F(\alpha) = 0$  that  $C_{q,k_i,\vec{a}_i}(\alpha) = 0$ , i.e.,

$$S = \sum_{x \in \text{vars}(C_{q,k_i,\vec{a}_i})} \alpha[x] \equiv k_i \pmod{q}.$$

For  $u \in \text{vars}(C_{q,k_i,\vec{a}_i})$  and  $w \notin \text{vars}(C_{q,k_i,\vec{a}_i})$ , after flipping  $u$  and  $w$  in  $\alpha$ , the above sum  $S$  is changed to  $k_i - 1 \pmod{q}$  or  $k_i + 1 \pmod{q}$ , thus  $F(\text{flip}(\alpha, \{u, w\})) = C_{q,k_i,\vec{a}_i}(\text{flip}(\alpha, \{u, w\})) = 1$ . For two distinct variables  $u, v \in \text{vars}(C_{q,k_i,\vec{a}_i})$ , if  $\alpha[u] \neq \alpha[v]$ , after flipping  $u$  and  $v$  in  $\alpha$ , the above sum  $S$  is still  $k_i \pmod{q}$ , thus  $F(\text{flip}(\alpha, \{u, v\})) = C_{q,k_i,\vec{a}_i}(\text{flip}(\alpha, \{u, v\})) = 0$ .  $\square$

**Lemma 4.4.** *Assume that  $F$  is read-once. Then, for any set  $S$  of exactly  $p$  variables such that they all have the same value in  $\alpha$ ,  $F(\text{flip}(\alpha, S)) = 0$  if and only if  $S \subseteq \text{vars}(C_{q,k_i,\vec{a}_i})$  for some  $C_{q,k_i,\vec{a}_i}$  in  $F$ .*

**Proof.** The sufficient condition is trivial, since  $F$  is read-once. Assume  $F(\text{flip}(\alpha, S)) = 0$  and suppose by contradiction that  $S \not\subseteq \text{vars}(C_{q,k_i,\vec{a}_i})$  for any  $C_{q,k_i,\vec{a}_i}$  in  $F$ , this implies that there are  $C_{q,k_i,\vec{a}_i}$  and  $C_{q,k_j,\vec{a}_j}$  with  $i \neq j$  such that  $S \cap \text{vars}(C_{q,k_i,\vec{a}_i}) \neq \phi$ , and  $S \cap \text{vars}(C_{q,k_j,\vec{a}_j}) \neq \phi$ . Thus,  $F(\text{flip}(\alpha, S)) = C_{q,k_i,\vec{a}_i}(\text{flip}(\alpha, S)) = C_{q,k_i,\vec{a}_i}(\text{flip}(\alpha, S)) = 1$ , a contradiction. So, there must be some  $C_{q,k_i,\vec{a}_i}$  in  $F$  such that  $S \subseteq \text{vars}(C_{q,k_i,\vec{a}_i})$ .  $\square$

**Lemma 4.5.** *Assume  $\text{vars}(C_{q,k_i,\vec{a}_i}) = \{u_1, \dots, u_m\}$  and  $m < q$ . Then, (1)  $C_{q,k_i,\vec{a}_i}$  is equivalent to  $[C_{q,0,\vec{a}_i}(u_1) \vee \dots \vee C_{q,0,\vec{a}_i}(u_m)]$  if  $[\alpha[u_i] = \dots = \alpha[u_m] = 0]$ ; (2)  $C_{q,k_i,\vec{a}_i}$  is equivalent to  $[C_{q,1,\vec{a}_i}(u_1) \vee \dots \vee C_{q,1,\vec{a}_i}(u_m)]$  if  $[\alpha[u_i] = \dots = \alpha[u_m] = 1]$ .*

**Proof.** Note that  $C_{q,k_i,\vec{a}_i}(\alpha) = 0$ . When  $\alpha[u_i] = \dots = \alpha[u_m] = 0$ ,  $\alpha[u_1] + \dots + \alpha[u_m] = 0 \equiv k_i \pmod{q}$ . When  $\alpha[u_i] = \dots = \alpha[u_m] = 1$ ,  $\alpha[u_1] + \dots + \alpha[u_m] = m \equiv k_i \pmod{q}$ . In the first case, we have  $k_i = 0$ . Since  $m < q$ ,  $C_{q,0,\vec{a}_i}(u_1, \dots, u_m)$  is equivalent to  $C_{q,0,\vec{a}_i}(u_1) \vee \dots \vee C_{q,0,\vec{a}_i}(u_m)$ . In the latter case, we have  $k_i = m < q$ , thus  $C_{q,m,\vec{a}_i}(u_1, \dots, u_m)$  is equivalent to  $C_{q,1,\vec{a}_i}(u_1) \vee \dots \vee C_{q,1,\vec{a}_i}(u_m)$ .  $\square$

**Theorem 4.1.** *The class of all read-once disjunctions of Boolean-weighted counting functions with modulus  $q$  over the domain  $Z_2^n$  is polynomial time learnable using only one equivalence query and  $O(n^q)$  membership queries.*

**Proof.** Assume  $F = C_{q,k_1,\vec{a}_1} \vee \dots \vee C_{q,k_t,\vec{a}_t}$  is the target function. We construct the learning algorithm Learn-RODC (where ‘‘RODC’’ stands for ‘‘read-once disjunctions of counting functions’’) that runs in stages. Learn-RODC is given on the next page.

We now analyze the algorithm Learn-RODC. We may assume without loss of generality that  $F \not\equiv \text{‘‘TRUE’’}$ . Thus, at stage 0, the learner receives a negative counterexample  $\alpha$  for  $F$ . It follows from Lemma 4.2 that one finds  $\text{vars}(F)$  at stage 1 with  $n$  membership queries. At stage 2, by Lemma 4.3, one finds all those  $\text{vars}(C_{q,k_i,\vec{a}_i})$  such that there are two variables in  $\text{vars}(C_{q,k_i,\vec{a}_i})$  with different values in  $\alpha$ . Thus,  $\bigvee \{C_{q,k(P \cup G),\vec{a}(P \cup G)} \mid (P, G) \in PG\}$  is the disjunction of all those counting functions in  $F$  such that each of them has two relevant variables with different values in  $\alpha$ . The number of membership queries required at this stage is at most  $2n^2$ . At stage 3, by Lemma 4.4, one finds all those  $\text{vars}(C_{q,k_i,\vec{a}_i})$  such that  $\text{vars}(C_{q,k_i,\vec{a}_i})$  consists of at least  $p$  variables that have the same value in  $\alpha$ . Thus,  $\bigvee \{C_{q,k(S),\vec{a}(S)} \mid S \in RS\}$  is the disjunction of all those counting functions in  $F$  such that each of them has at least  $p$  relevant variables with the same value in  $\alpha$ . The number of membership queries required at this stage is at most  $n^q$ . By Lemma 4.5,  $\bigvee \{C_{q,\alpha[x],\vec{a}(\{x\})} \mid x \in E\text{vars}(F)\}$  is equivalent to the disjunction of all those counting functions  $C_{q,k_i,\vec{a}_i}$  in  $F$  such that  $\text{vars}(C_{q,k_i,\vec{a}_i})$  consists of less than  $p$  relevant variables that have the same value in  $\alpha$ . No membership queries are required at this stage. With the above analysis,  $F$  is equivalent to  $H$ . Learn-RODC needs only one equivalence query,  $n + 2n^2 + n^q$  membership queries. The time complexity is  $O(n^2 + 2n^3 + n^{q+1}) = O(n^{q+1})$ .  $\square$

## 5 Disjunctions of a Non-Constant Number of Counting Functions

A typical strategy for learning  $k$ -term DNF formulas with equivalence and membership queries is that at each stage the learner tries to learn only one term in the target formulas while turning all the other terms off. The difficulty involved in this strategy is how the learner can turn off all terms off except one on. When  $k$  is a constant, this difficulty was overcome by Angluin’s discriminant mechanism [Ab]. When  $k = O(\log n)$ , it was overcome by Blum and Rudich’s derandomization technique [BR]. However, unlike a

**Learn-RODC:**

Stage 0. Ask an equivalence query for the “TRUE” function. If “yes” then stop; otherwise the learner receives a negative counterexample  $\alpha$ .

Stage 1. For each  $x \in V_n$ , ask a membership query for  $flip(\alpha, x)$ . Let  $vars(F)$  be the set of all those  $x$  such that the learner receives “yes” for  $flip(\alpha, x)$ .

Stage 2. Fix any  $u \in vars(F)$ . For any  $v \in vars(F) - \{u\}$  such that  $\alpha[u] \neq \alpha[v]$ , ask a membership query for  $flip(\alpha, \{u, v\})$ . Let  $G_u$  be the set of all those  $v$  such that the learner receives “no” for  $flip(\alpha, \{u, v\})$ . Let  $P_u$  be the set of all those  $x$  such that  $G_x = G_u \neq \phi$ , and  $\alpha[x] = \alpha[u]$ . Set  $PG = \{(P_u, G_u) | u \in vars(F), G_u \neq \phi\}$ .

Stage 3. Let  $Rvars(F)$  be the set of all variables in  $vars(F)$  but not in any set in  $PG$ . Fix any  $u \in Rvars(F)$ . For any subset  $S$  of  $Rvars(F) - \{u\}$  with exactly  $p - 1$  variables such that all those variables and  $u$  have the same value in  $\alpha$ , ask a membership query for  $flip(\alpha, \{u\} \cup S)$ . Let  $S_u$  be the union of all those subsets  $S$  and  $\{u\}$  such that the learner receives “no” for  $flip(\alpha, \{u\} \cup S)$ . Set  $RS = \{S_u | u \in Rvars(F), S_u \neq \phi\}$ .

Stage 4. Let  $Evars(F)$  be the set of all variables in  $vars(F)$  but not in any sets in  $PG$  or  $RS$ . For any set  $A \subseteq V_n$ , let  $\vec{a}(A)$  be the characteristic vector of  $A$ , and  $k(A) = \sum_{x \in A} \alpha[x] \bmod q$ . The learner concludes that the target function  $F$  is equivalent to

$$H = \bigvee \{C_{q,k(P \cup G),a(P \cup G)} | (P, G) \in PG\} \vee \bigvee \{C_{q,k(S),\vec{a}(S)} | S \in RS\} \\ \vee \bigvee \{C_{q,\alpha[x],\vec{a}(\{x\})} | x \in Evars(F)\}.$$

**End of Learn-RODC.**

monomial which turns on if and only if all its literals turn on, a counting function depends on the modulo  $p$  value of the sum of its variables. Thus, it is not hard to see that Angluin’s discriminant mechanism and Blum and Rudich’s derandomization technique are not suitable for learning a disjunction of a non-constant number of counting functions. Nevertheless, based on analyzing the “modulo-structure” of counting functions, we prove that for any constant  $c$ , any disjunction with no more than  $\log \log n^c$  many Boolean-weighted counting functions over the domain  $Z_2^n$  is polynomial time learnable.

Assume that  $q \geq 2$  is a given integer number,  $F = C_{q,k_1,\vec{a}_1} \vee \dots \vee C_{q,k_t,\vec{a}_t}$  is a disjunction of counting functions over the domain  $Z_2^n$  with Boolean-weights  $\vec{a}_i \in Z_2^n$ . Assume also that  $\alpha$  is a negative counterexample for  $F$ . For any  $S \subseteq vars(F)$ , define  $C_S = \{C_{q,k_i,\vec{a}_i} | S \subseteq vars(C_{q,k_i,\vec{a}_i}), 1 \leq i \leq t\}$ . We say that  $S \neq \phi$  is a “modulo-block” of  $F$  if,  $S = \bigcap_{C_{q,k_i,\vec{a}_i} \in C_S} vars(C_{q,k_i,\vec{a}_i})$ , and for any  $C_{q,k_j,\vec{a}_j} \notin C_S$ ,  $S \cap vars(C_{q,k_j,\vec{a}_j}) = \phi$ . Let  $MB_F$  (“MB” stands for “modulo-blocks”) denote the set of all modulo-blocks of  $F$ . Note that For any two modulo-blocks  $B, D \in MB_F$ , either  $B = D$ , or  $B \cap D = \phi$ .

**Lemma 5.1.** For any modulo-block  $B \in MB_F$ , for any two distinct variables  $x, y \in B$  and, for any variable  $u \in \text{vars}(F) - B$ , we have (1)  $F(\text{flip}(\alpha, \{x, u\})) = 0$  and, (2)  $F(\text{flip}(\alpha, \{x, y\})) = 0$  if  $\alpha[x] \neq \alpha[y]$ .

**Proof.** By the definition,  $x, y \in B$  implies  $x, y \in \text{vars}(C_{q, k_i, \vec{a}_i})$  for any  $C_{q, k_i, \vec{a}_i} \in C_B$  and  $x, y \notin \text{vars}(C_{q, k_j, \vec{a}_j})$  for any  $C_{q, k_j, \vec{a}_j} \notin C_B$ .  $C_{q, k_i, \vec{a}_i}(\alpha) = 0$  means that

$$\sum_{v \in \text{vars}(C_{q, k_i, \vec{a}_i})} \alpha[v] \equiv k_i \pmod{q}.$$

If  $\alpha[x] \neq \alpha[y]$ , the above sum will not change after flipping both  $x$  and  $y$  in  $\alpha$ . So,  $F(\text{flip}(\alpha, \{x, y\})) = C_{q, k_i, \vec{a}_i}(\text{flip}(\alpha, \{x, y\})) = 0$ . On the other hand, it is easy to see that  $C_{q, k_i, \vec{a}_i}(\text{flip}(\alpha, \{x\})) = 1$  for any  $C_{q, k_i, \vec{a}_i} \in C_B$ . Since  $u \notin B$ , there is a  $C_{q, k_j, \vec{a}_j} \in C_B$  such that  $u \notin \text{vars}(C_{q, k_j, \vec{a}_j})$ . Hence,  $F(\text{flip}(\alpha, \{x, u\})) = C_{q, k_j, \vec{a}_j}(\text{flip}(\alpha, \{x, u\})) = C_{q, k_j, \vec{a}_j}(\text{flip}(\alpha, \{x\})) = 1$ .  $\square$

**Lemma 5.2.** For any  $S \subseteq \text{vars}(F)$  with exactly  $p$  variables such that they all have the same value in  $\alpha$ ,  $F(\text{flip}(\alpha, S)) = 0$  if and only if  $S \subseteq B$  for some modulo-block  $B \in MB_F$ .

**Proof.** The sufficient condition is trivial by the definition of modulo-blocks. Assume  $F(\text{flip}(\alpha, S)) = 0$  and suppose by contradiction that  $S$  is not a subset of any modulo-blocks of  $F$ . this implies that there are two distinct modulo-blocks  $B_1$  and  $B_2$  in  $MB_F$  such that  $S \cap B_1 \neq \emptyset$  and  $S \cap B_2 \neq \emptyset$ . Hence, by the definition of modulo-blocks, there are one counting function in  $C_{B_1}$  and another in  $C_{B_2}$  such that each of them has at least one but less than  $p$  variables of  $S$ . So, after flipping all variables in  $S$  in  $\alpha$ , those two counting functions (thus  $F$ ) will have value 1, a contradiction to the early assumption.  $\square$

**Lemma 5.3.** For any counting function  $C_{q, k_i, \vec{a}_i}$  in  $F$ , there are modulo-blocks  $B_1, \dots, B_m \in MB_F$  such that  $\vec{a}$  is the characteristic vector of  $B = B_1 \cup \dots \cup B_m$ ,  $k = \sum_{x \in B} \alpha[x] \pmod{q}$ .

**Proof.** We first show that there are modulo-blocks  $B_1, \dots, B_m \in MB_F$  such that  $\text{vars}(C_{q, k_i, \vec{a}_i}) = B_1 \cup \dots \cup B_m$ . Fix a variable  $x_1 \in \text{vars}(C_{q, k_i, \vec{a}_i})$ . Let

$$Q_1 = \bigcap \{ \text{vars}(C_{q, k_j, \vec{a}_j}) \mid x_1 \in \text{vars}(C_{q, k_j, \vec{a}_j}) \}.$$

Then,  $x_1 \in Q_1$ . Define  $B_1 = \{y \in Q_1 \mid \forall C_{q, k_j, \vec{a}_j} \notin Q_1, y \notin \text{vars}(C_{q, k_j, \vec{a}_j})\}$ . It is easy to see that,  $x \in B_1$ , and  $B_1$  is a modulo-block of  $F$ . Note that  $B_1 \subseteq \text{vars}(C_{q, k_i, \vec{a}_i})$ . If  $B_1 = \text{vars}(C_{q, k_i, \vec{a}_i})$ , then we are done. Otherwise, fix a variable  $x_2 \in \text{vars}(C_{q, k_i, \vec{a}_i}) - B_1$ . We define  $Q_2$  and  $B_2$  in the same manner, thus we obtain a new modulo-block  $B_2$  with  $x_2 \in B_2 \subseteq \text{vars}(C_{q, k_i, \vec{a}_i})$ . If  $B_1 \cup B_2 = \text{vars}(C_{q, k_i, \vec{a}_i})$ , then we are done. Otherwise, repeat the above process to obtain a new modulo-block. Note that  $\text{vars}(C_{q, k_i, \vec{a}_i})$  contains at most  $n$  variables. We eliminate at least one variable from  $\text{vars}(C_{q, k_i, \vec{a}_i})$  when we obtain a new modulo-block. Thus, we have  $m$  modulo-blocks  $B_1, \dots, B_m$ ,  $m \leq n$ , such that  $\text{vars}(C_{q, k_i, \vec{a}_i}) = B_1 \cup \dots \cup B_m$ ,  $m \leq n$ . It then follows that  $\vec{a}$  is the characteristic vector of  $B = B_1 \cup \dots \cup B_m$ . By Lemma 5.1,  $B_i \cap B_j = \emptyset$  if  $i \neq j$ , for any  $i, j \in \{1, \dots, t\}$ .  $C_{q, k_i, \vec{a}_i}(\alpha) = 0$  implies that  $k = \sum_{x \in B_1 \cup \dots \cup B_m} \alpha[x] \pmod{q}$ .  $\square$

**Lemma 5.4.**  $\|MB_F\| \leq 2^t$ . In other words,  $F$  has at most  $2^t$  modulo-blocks.

**Proof.** According to Lemma 5.3, given a negative counterexample for  $F$ , each  $C_{q,k_i,\vec{a}_i}$  in  $F$  is determined by the modulo-blocks that consist of  $\text{vars}(C_{q,k_i,\vec{a}_i})$ . Thus, we can represent  $F$  with a matrix  $M$ ,  $M$  has  $t$  rows and  $m$  columns. The  $i$ -th row of  $M$  stands for the the function  $C_{q,k_i,\vec{a}_i}$ . Each column contains a modulo-block, and no two columns have the same modulo-block. Let  $e_{i,j}$  denote the entry of  $M$  at the  $i$ -th row and the  $j$ -th column. Assume that the  $j$ -th column contains the modulo-block  $B_j$ . Then  $e_{i,j} = B_j$  if  $B_j \subseteq \text{vars}(C_{q,k_i,\vec{a}_i})$ , otherwise let  $e_{i,j} = \text{"blank"}$ . We now estimate how large  $t$  can be. For column  $a$  and column  $b$ ,  $a \neq b$ , by the definition of modulo-blocks, there exists at least one  $i$  such that  $e_{i,a}$  differs from  $e_{i,b}$ , i.e., either  $e_{i,a} = B_a$  but  $e_{i,b} = \text{"blank"}$ , or  $e_{i,a} = \text{"blank"}$  but  $e_{i,b} = B_b$ . This implies that  $m \leq 2^t$ , since there are at most  $2^t$  many possible ways to place a modulo-block in a column. Thus,  $\|MB_F\| \leq 2^t$ .  $\square$

**Theorem 5.1.** *There is an algorithm for learning the class of disjunctions of no more than  $\log \log n^c$  many Boolean-weighted counting functions with modulus  $q$  over the domain  $Z_2^n$ , using  $O(n^q + n^{c(q+1)})$  many queries. The time complexity of the algorithm is bounded by  $O(n^{c+1} + n^{2c(q+1)+1})$ . So for constant  $c$ , the algorithm is polynomial.*

**Proof.** Assume that  $F = C_{q,k_1,\vec{a}_1} \vee \cdots \vee C_{q,k_t,\vec{a}_t}$  is the target function. The learning algorithm LEARNER runs in stages.

At stage 0, the learner issues the initial hypothesis  $H_1 = \text{"TRUE"}$  to ask an equivalence query. If he receives “yes” then stop. Otherwise, he receives a negative example  $\alpha$  for  $F$ . One query is used at this stage, the time complexity is constant.

At stage 1, for any  $x \in V_n$ , the learner asks a membership query for  $\text{flip}(\alpha, x)$ . By Lemma 5.2, the learner finds  $\text{vars}(F)$ , i.e., the set of all those variables such that flipping any one of them in  $\alpha$  will cause  $F$  to output 1. The number of queries used at this stage is  $n$ , the time complexity is  $O(n^2)$ .

At stage 2, using Lemma 5.1, the learner finds all those modulo-blocks in which there are two distinct variables in each of them with different values in  $\alpha$ : For any  $u \in \text{vars}(F)$ , for any  $v \in \text{vars}(F) - \{u\}$  such that  $u$  and  $v$  have different values in  $\alpha$ , ask a membership query for  $\text{flip}(\alpha, \{u, v\})$ . Let  $A(u)$  be the set of all those  $v$  such that the learner receives “no”. Let  $E(u)$  be the set of all those  $w$  such that  $A(w) = A(u) \neq \phi$  and  $\alpha[w] = \alpha[u]$ . Set  $B_u = A_u \cup E_u$ , then  $B_u$  is a modulo-block. At this stage at most  $n^2$  membership queries are required and the time complexity is  $O(n^3)$ .

At stage 3, using Lemma 5.2, the learner finds all those modulo-blocks such that each of them has at least  $q$  variables and all of the variables in it have the same value in  $\alpha$ : For any  $u \in \text{vars}(F)$ , for any set  $S \subseteq \text{vars}(F) - \{u\}$  with exactly  $q - 1$  variables such that  $u$  and variables in  $S$  have the same value in  $\alpha$ , ask a membership query for  $\text{flip}(\alpha, \{u\} \cup S)$ . Let  $S(u)$  be the union of all those subsets  $S$  and  $\{u\}$  such that the learner receives “no” for  $\text{flip}(\alpha, \{u\} \cup S)$ , then  $S(u)$  is a modulo-block if it is not empty. The number of queries used at this stage is at most  $n^q$ , and the time complexity is  $O(n^{q+1})$ .

At stage 4, the learner finds all *possible* modulo-blocks such that each of them has

at most  $q - 1$  variables and all variables in it have the same value in  $\alpha$ : Let  $FB$  be the set of all modulo-blocks found at the above stage 2 and 3, let  $RB$  be the set of all variables in  $\text{vars}(F)$  but not in any modulo-blocks in  $FB$ . Then, each modulo-block  $B \in MB_F - FB$  has less than  $q$  variables and all variables in it have the same value in  $\alpha$ . It is trivial that  $B$  is a subset of  $RB$ . By Lemma 5.4,  $\|RB\| \leq q2^t$ . Actually, one finds  $RB$  as a by-product of stage 2 and stage 3, i.e., whenever one finds a modulo-block at those two stages one eliminates all variables in it from  $\text{vars}(F)$ . The remaining variables in  $\text{vars}(F)$  is  $RB$ . Thus, the number of queries required at this stage is 0, the time complexity is  $O(n^3 + n^{q+1})$ .

At stage 5, the learner constructs all possible counting functions using modulo-blocks in  $FB$  and subsets in  $RB$ : For any modulo-blocks  $B_1, \dots, B_m \in FB$ , for any subset  $R$  of  $RB$ , set  $W = B_1 \cup \dots \cup B_m \cup R$ . Define a counting function  $H(B_1, \dots, B_m, R)$  as  $C_{q,l,\vec{\alpha}}$ , where  $\vec{\alpha}$  is the characteristic vector of  $W$ , and  $l = \sum_{x \in W} \alpha[x] \bmod q$ . Finally, the learner sets the hypothesis

$$H_2 = \bigvee_{B_1, \dots, B_m \in MB, R \subseteq MR} H(B_1, \dots, B_m, R).$$

With Lemma 5.3, every counting function in  $F$  is contained in  $H_2$ . The number of queries required at this stage is 0, the time complexity is  $O(n2^{2^t}2^{q2^t})$ .

At stage 6, the learner asks equivalence queries for the hypothesis  $H_2$ . If the answer is “yes” then stop. Otherwise one receives a negative counterexample  $\beta$ , since  $H_2$  contains all counting functions in  $F$ . Thus, one eliminates every counting functions in  $H_2$  that outputs 1 for  $\beta$ . One still uses  $H_2$  to denote the disjunction of the remaining counting functions in  $H_2$ . Repeat the above process until one receives “yes”. The number of queries used at this stage is at most  $2^{2^t}2^{q2^t}$ , since  $H_2$  originally contains at most  $2^{2^t}2^{q2^t}$  counting functions. For each equivalence query one needs to write down the hypothesis, so the time complexity of this stage is at most  $O(n2^{2^{t+1}}2^{q2^{t+1}})$ .

Combining the above analysis, the learner needs  $O(n^q + 2^{2^t}2^{q2^t})$  many queries to learn  $F$ , and the time complexity is bounded by  $O(n^{q+1} + n2^{2^{t+1}}2^{q2^{t+1}})$ . When  $t \leq \log \log n^c$ , the number of queries is bounded by  $O(n^q + n^{c(q+1)})$ , and the time complexity is bounded by  $O(n^{q+1} + n^{2c(q+1)+1})$ .  $\square$

## 6 Graph-Based Counting Function

In this section, we examine the problem of learning graph-based counting functions. These functions are different from those studied in sections 4 and 5. It is not hard to observe that the task of learning a disjunction of counting functions is as easy as that of finding the relevant variables of each of the counting functions (see also [BCJ] for similar observations about embedded symmetric concepts). However, relevant variables of the counting functions in a given disjunction may be overlapped in arbitrary ways (one will note that the graph-based counting functions are good examples of the arbitrary overlapping). It is in general very difficult to find relevant variables for each of the counting functions.

Graph-based parity functions, which was initially introduced by Tseitin [T], have played a key role in the study of the complexity of the resolution method. These functions are hard for resolution, because the relevant variables of the parity functions are overlapped arbitrarily. Graph-based counting functions are general cases of Tseitin’s original definition, since we associate each vertex of a graph with a counting function instead of a parity function. Another extension of Tseitin’s definition to hypergraphs was given in [CS].

Now, we assume that a graph  $G(V, E)$  is undirected, connected, and has no multiple edges or cyclic edges. Let  $q > 1$  be a given integer. Given a graph  $G(V, E)$ , we label each edge  $e$  of  $G(V, E)$  with an independent variable (denoted by  $label(e)$ ) and call such a variable an “edge-variable”. We assign each vertex  $v$  of  $G$  with an index  $k$ ,  $0 \leq k \leq q$  (write  $index(v)$ ). We also label each vertex  $v$  of  $G(V, E)$  with a set of independent variables (denoted by  $att(v)$ ), which are called “vertex-variables”. We assume that variables are not duplicated among vertices or edges. A graph  $G(V, E)$  with the above labeling is denoted by  $G(V, E, index, label, att)$ , where  $V$  is the vertex set,  $E$  is the edge set,  $index$  is the index-mapping from  $V$  to  $Z_q$ ,  $label$  is a variable-mapping from  $E$  to  $V_n$ ,  $att$  is the variable-set-mapping from  $V$  to the power set of  $V_n$ . Given any variable  $x$ , let  $vertex(x)$  denote the vertex at which  $x$  is labeled. When  $x$  is an edge-variable,  $vertex(x)$  denotes the set of two vertices connected by the edge on which  $x$  is labeled. Given a vertex  $v$ , let  $edge(v)$  denote the set of all variables labeled on the edges adjacent to  $v$ .

Given a graph  $G(V, E, index, label, att)$ , for any vertex  $v \in V$ , we define a counting function  $C^v$  at the vertex  $v$  as  $C_{q, index(v), \vec{a}}$ , where  $\vec{a}$  is the characteristic vector of the set  $att(v) \cup edge(v)$ . We finally define  $F_{q,G} = \bigvee \{C^v | v \in V\}$ . We call  $F_{q,G}$  a counting function based on the graph  $G$ . For any graph  $G(V, E, index, label, att)$ , define  $size(G) = \|E\| + \sum_{v \in V} \|att(v)\|$ . In other words,  $size(G)$  is the number of variables labeling  $G$ .

**Theorem 6.1.** *There is a algorithm for learning the class of counting functions based on graphs  $G(V, E, index, label, att)$  such that  $size(G) \leq n$  and  $\|att(v)\| \geq q$  for any  $v \in V$  over the domain  $Z_q^n$ . The algorithm uses only one equivalence query and  $O(n^q)$  membership queries, while its time complexity is bounded by  $O(n^{q+1})$ .*

**Proof.** Let  $V = \{v_1, \dots, v_m\}$ . Then,  $F_{q,G} = C^{v_1} \vee \dots \vee C^{v_m}$ . The learning algorithm Learn-GBC (where “GBC” stands for “graph-based counting functions”) runs in stages.

At stage 0, the learner first uses the hypothesis  $H_1 = \text{“TRUE”}$  to ask an equivalence query. If “yes” then stop. Otherwise, one receives a negative example  $\alpha$ . Only one query and constant time are required at this stage.

At stage 1, one asks a membership query for the example  $flip(\alpha, x)$  for any  $x \in V_n$ . By Lemma 5.2, one finds  $vars(F_{q,G})$ , i.e., the set of all variables such that flipping any one of them in  $\alpha$  will cause  $F_{q,G}$  to output 1. The number of queries used here is  $n$ , the time complexity is  $O(n^2)$ .

At stage 2, the learner finds those  $att(v)$  for  $v \in V$  such that there are at least

two variables in it with different values in  $\alpha$ : For any  $v \in \text{vars}(F_{q,G})$ , for any  $u \in \text{vars}(F_{q,G}) - \{v\}$  such that  $v$  and  $u$  have different values in  $\alpha$ , ask a membership query for  $\text{flip}(\alpha, \{v, u\})$ . Let  $A(v)$  be the set of all those  $v$  such that one receives “no”. Let  $E(v)$  be the set of all those  $w$  such that  $A(w) = A(v) \neq \phi$  and  $\alpha[w] = \alpha[v]$ . Set  $B_v = A_v \cup E_v$ . At this stage at most  $n^2$  membership queries are required, the time complexity is  $O(n^3)$ . The correctness of this stage is guaranteed by Lemma 6.1. Define  $V^1 = \{B_v | B_v \neq \phi\}$ .

**Lemma 6.1.**  *$B_v \neq \phi$  if and only if  $\text{vertex}(v) \in V$  and there are two variables in  $\text{att}(\text{vertex}(v))$  with different values in  $\alpha$ . Moreover, when  $B_v \neq \phi$ , then  $B_v = \text{att}(\text{vertex}(v))$ .*

**Proof.** Suppose that  $v$  is a vertex-variable, i.e.,  $\text{vertex}(v) \in V$ . For any variable  $u \in \text{vars}(F_{q,G})$  with  $\alpha[v] \neq \alpha[u]$ , we consider two cases. When  $u$  is a edge-variable, let  $f$  be the vertex in  $\text{vertex}(u) - \{v\}$ , then  $u \in \text{vars}(C^f)$  but  $v \notin \text{vars}(C^f)$ . Thus,  $F(\text{flip}(\alpha, \{v, u\})) = C^f(\text{flip}(\alpha, \{v, u\})) = 1$ . When  $u$  is a vertex-variable with  $\text{vertex}(u) = \text{vertex}(v)$ , then  $u$  and  $v$  occur only in  $C^{\text{vertex}(v)}$ , thus  $F(\text{flip}(\alpha, \{v, u\})) = C^{\text{vertex}(v)}(\text{flip}(\alpha, \{v, u\})) = 0$ , since  $C^{\text{vertex}(v)}(\alpha) = 0$ . If  $\text{vertex}(u) \neq \text{vertex}(v)$ , then  $v$  occurs only in  $C^{\text{vertex}(v)}$ , and  $u$  occurs only in  $C^{\text{vertex}(u)}$ , thus  $F(\text{flip}(\alpha, \{v, u\})) = C^{\text{vertex}(v)}(\text{flip}(\alpha, \{v, u\})) = C^{\text{vertex}(u)}(\text{flip}(\alpha, \{v, u\})) = 1$ , since  $C^{\text{vertex}(v)}(\alpha) = 0$  and  $C^{\text{vertex}(u)}(\alpha) = 0$ .

Now suppose that  $v$  is an edge-variable. Let  $\text{vertex}(v) = \{f, g\}$ . For any variable  $u \in \text{vars}(F_{q,G}) - \{v\}$  with  $\alpha[v] \neq \alpha[u]$ , there is at least one vertex in  $\text{vertex}(v)$ , say,  $f$ , such that  $v$  and  $u$  do not occur in  $C^f$  simultaneously. Thus,  $F(\text{flip}(\alpha, \{v, u\})) = C^f(\text{flip}(\alpha, \{v, u\})) = 1$ , since  $C^f(\alpha) = 0$ .

Combing the above analysis, our Lemma holds.  $\square$

At stage 3, the learner finds all those  $\text{att}(v)$  for  $\text{vertex}(v) \in V$  such that each variable in it has the same value in  $\alpha$ : For any  $v \in \text{vars}(F_{q,G})$ , for any set  $S \subseteq \text{vars}(F_{q,G}) - \{v\}$  with exactly  $q - 1$  variables such that  $v$  and variables in  $S$  have the same value in  $\alpha$ , ask a membership query for  $\text{flip}(\alpha, \{v\} \cup S)$ . Let  $S(v)$  be the union of all those subsets  $S$  and  $\{u\}$  such that the learner receives “no” for  $\text{flip}(\alpha, \{v\} \cup S)$ . The number of queries required at this stage is  $n^q$ , the time complexity is bounded by  $O(n^{q+1})$ . The correctness of this stage is guaranteed by Lemma 6.2. Define  $V^2 = \{S(v) | S(v) \neq \phi\}$ .

**Lemma 6.2.**  *$S(v) \neq \phi$  if and only if,  $\text{vertex}(v) \in V$ ,  $\|\text{att}(\text{vertex}(v))\| \geq q$ , and all variables in  $\text{att}(\text{vertex}(v))$  have the same value in  $\alpha$ . Moreover, when  $S(v) \neq \phi$ ,  $S(v) = \text{att}(\text{vertex}(v))$ .*

**Proof.** Suppose that  $v$  is an edge-variable, let  $\text{vertex}(v) = \{f, g\}$ . For any  $S \subseteq \text{vars}(F_{q,G})$  such that  $S$  has exactly  $q - 1$  variables with the same value as  $v$  does in  $\alpha$ , there is at least one vertex in  $\text{vertex}(v)$ , say,  $f$ , such that  $1 \leq \|(S \cup \{v\}) \cap \text{att}(f)\| \leq q - 1$ . Thus,  $F(\text{flip}(\alpha, S \cup \{v\})) = C^f(\text{flip}(\alpha, S \cup \{v\})) = 1$ , since  $C^f(\alpha) = 0$ .

Now suppose that  $v$  is a vertex-variable. For any  $S \subseteq \text{vars}(F_{q,G})$  such that  $S$  has exactly  $q - 1$  variables with the same value as  $v$  does in  $\alpha$ , if  $S \not\subseteq \text{att}(\text{vertex}(v))$  then  $1 \leq \|(S \cup \{v\}) \cap \text{att}(\text{vertex}(v))\| \leq q - 1$ . Thus,  $F(\text{flip}(\alpha, S \cup \{v\})) = C^{\text{vertex}(v)}(\text{flip}(\alpha, S \cup \{v\})) = 0$ .

$\{v\}) = 1$ , since  $C^f(\alpha) = 0$ . When  $S \subseteq \text{att}(\text{vertex}(v))$ , then  $\|(S \cup \{v\}) \cap \text{att}(\text{vertex}(v))\| = q$ . Thus,  $F(\text{flip}(\alpha, S \cup \{v\})) = C^{\text{vertex}(v)}(\text{flip}(\alpha, S \cup \{v\})) = C^f(\alpha) = 0$ . Our lemma then follows.  $\square$

At stage 4, the learner finds all  $\text{edge}(v)$  for  $\text{vertex}(v) \in V$ : Fix a  $D(v) \in V^1 \cup V^2$ . For any  $D(u) \in V^1 \cup V^2$ , for any  $w \in \text{vars}(F_{p,G}) - (D(v) \cup D(u))$ , define  $\beta$  to be the example obtained from  $\alpha$  by (1)flipping  $w$ ; (2)flipping exactly one variable in  $D(v)$  such that this variable and  $w$  have different values in  $\alpha$ , if such a variable exists, otherwise flipping exactly  $q - 1$  variables in  $D(v)$ ; (3)flipping exactly one variable in  $D(u)$  such that this variable and  $w$  have different values in  $\alpha$ , if such a variable exists, otherwise flipping exactly  $q - 1$  variables in  $D(u)$ . Ask a membership query for  $\beta$ . Set  $E(v)$  to be the set of all  $w$  such that one receives “no” for  $\beta$ . The number of queries required at this stage is at most  $n^3$ , the time complexity of this stage is bounded by  $O(n^4)$ . The correctness of this stage is guaranteed by Lemma 6.3. Let  $E = \{E(v) | D(v) \in V^1 \cup V^2\}$ .

**Lemma 6.3.**  $E = \{\text{edge}(v) | v \in V\}$ .

**Proof.** We first show that  $E \subseteq \{\text{edge}(v) | v \in V\}$ . Fix  $E(v) \in E$ . We then have  $D(v) \in V^1 \cup V^2$ . By Lemma 6.1 and 6.2,  $D(v) = \text{att}(\text{vertex}(v))$ . According to the process of stage 4, there are either one or  $p - 1$  variables in  $\text{att}(\text{vertex}(v))$  flipped in  $\alpha$  to obtain  $\beta$ . For any  $w \in E(v)$ , if  $w \notin \text{edge}(\text{vertex}(v))$ , then no variables are flipped in  $\text{edge}(\text{vertex}(v))$ . Thus,  $F(\beta) = C^{\text{vertex}(v)}(\beta) = 1$ , since  $C^{\text{vertex}(v)}(\alpha) = 0$ . This contradicts to the fact that one receives “no” for  $\beta$ . Hence,  $w \in \text{edge}(\text{vertex}(v))$ . On the other hand, for any  $w \in \text{edge}(\text{vertex}(v))$ , let  $\text{vertex}(u)$  be the other vertex connected by the edge on which  $w$  is labeled. By Lemma 6.1 and 6.2,  $D(u) = \text{att}(\text{vertex}(u)) \in V^1 \cup V^2$ . Again according to the process of stage 4, one receives “no” for  $\beta$  obtained from flipping  $w$  and either  $q - 1$  (or one) variables in each of  $D(v)$  and  $D(u)$ . This implies that  $w \in E(v)$ . Hence,  $E \subseteq \{\text{edge}(v) | v \in V\}$ .

Now consider any  $\text{edge}(\text{vertex}(v))$ . For any  $w$  in it, let  $\text{vertex}(u)$  be the other vertex connected by the edge on which  $w$  is labeled. By Lemma 6.1 and 6.2,  $D(v) = \text{att}(\text{vertex}(v))$  and  $D(u) = \text{att}(\text{vertex}(u))$  are in  $V^1 \cup V^2$ . According to the process of stage 4, one receives “no” for  $\beta$  obtained from flipping  $w$  and either  $q - 1$  (or one) variables in each of  $D(v)$  and  $D(u)$ , thus  $w \in E(v)$ . Note that at stage 4 one only considers variables in  $\text{vars}(F_{q,G}) - (D(v) \cup D(u))$ . For any  $w \notin \text{edge}(\text{vertex}(v)) \cup D(v)$ , according to the process of stage 4, there are either one or  $p - 1$  variables in  $D(v) = \text{att}(\text{vertex}(v))$  flipped. Thus,  $F(\beta) = C^{\text{vertex}(v)}(\beta) = 1$ , since  $C^{\text{vertex}(v)}(\alpha) = 0$ . This implies that  $w \notin E(v)$ . Hence,  $E(v) = \text{edge}(\text{vertex}(v))$ , so  $\{\text{edge}(v) | v \in V\} \subseteq E$ .  $\square$

At stage 5, the learner constructs  $F_{q,G}$  from  $V^1 \cup V^2$  and  $E$ . For any  $D_v \in V^1 \cup V^2$ , one finds  $E(v) \in E$ . Let  $k = \sum_{y \in D_v \cup E_v} \alpha[y] \bmod q$  and  $\vec{a}$  be the characteristic vector of  $D_v \cup E_v$ . Define a counting function  $C(D_v, E_v)$  as  $C_{q,k,\vec{a}}$ . By Lemmas 6.1, 6.2, and 6.3,  $H_2 = \bigvee \{C(D_v, E_v) | D_v \in V^1 \cup V^2\}$  is equivalent to  $F_{q,G}$ . So, the learner concludes that  $H_2$  is equivalent to  $F_{q,G}$  and then stops. No queries are required at this stage, the time complexity is bounded by  $O(n^2)$ .

Putting the above analysis together, the algorithm learns  $F_{q,G}$  using only one equivalence query and  $O(n^q)$  membership queries, while its time complexity is bounded by

$O(n^{q+1})$ .  $\square$

## 7 Concluding Remarks

We have shown that, for any prime number  $p$ , the class of disjunctions of integer-weighted counting functions with modulus  $p$  over the domain  $Z_q^n$  (or  $Z^n$ ) with  $q > 1$  is polynomial time learnable with only equivalence queries. We don't know, however, whether similar results hold for composite number  $q > 1$ . The linear algebra approach is not suitable for this case, since  $Z_q$  is not a field.

As argued in [BHS], it is reasonable to believe that an equivalence query is more expensive than a membership query. A practically ideal learning algorithm will use equivalence queries as less as possible. On one hand, there is no obvious way so far, based on the linear algebra approach, to learn counting functions using substantially less than  $n + 1$  equivalence queries, with a polynomial number of additional membership queries. On the other hand, we have shown that, for any given integer  $q \geq 2$ , over the domain  $Z_2^n$ , only one equivalence is sufficient for learning the class of read-once disjunctions of Boolean-weighted counting functions with modulus  $q$ , and the class of counting functions based on those graphs  $G(V, E, index, label, att)$  such that  $\|att(v)\| \geq q$  for any  $v \in V$ , using polynomially many membership queries.

The graph-based counting functions are very interesting, because they correspond to graphs and have played an important role in the study of complexity of resolution. In general they seem difficult to learn. We have shown that the class of disjunctions of no more than  $\log \log n^c$  many Boolean-weighted counting functions with modulus  $q$  for any given integer  $q \geq 2$  over the domain  $Z_2^n$  is polynomial time learnable. Very recently, Jeffrey Jackson [J] observed from Fourier analysis that the class of disjunctions of  $O(\log n)$  parities is polynomial time learnable. It might be possible to extend his result to the class of disjunctions of  $O(\log n)$  counting functions with a composite modulus.

### Acknowledgments.

We are very grateful to Robert Schapire for valuable discussions on this topic, and especially for providing us with his algorithm for learning disjunctions of parities with equivalence queries. We credit the algorithms Learn-HS and Learn-IHS to Schapire and, Helmbold, Sloan and Warmuth, because they are general cases of Schapire's parity-learning algorithm [S], and they are motivated by the algorithm V developed by Helmbold, Sloan and Warmuth [HRS]. Early results of this paper on learning disjunctions of parities with queries [HC] was presented in an impromptu talk at COLT' 93.

## References

- [Aa] D. Angluin, "Queries and concept learning", *Machine Learning*, 2, 1988, pages 319-342.

- [Ab] D. Angluin, “Learning  $k$ -term DNF formulas using queries and counterexamples”, Technical Report YALE/DCS/RR-559, Yale University, August 1987.
- [AHM] D. Angluin, L. Hellerstein, M. Karpinsky, “Learning learning read-once formulas with queries”, *J. ACM*, 1, 1993, pages 185-210.
- [BR] A. Blum, S. Rudich, “Fast learning of  $k$ -term DNF formulas with queries”, *Proc of the 24th Annual ACM Symposium on Theory of Computing*, May 1992, pages 382-389.
- [BCJ] A. Blum, P. Chalasani, J. Jackson, “On learning embedded symmetric concepts”, *Proc of the Sixth Annual ACM Conference on Computational Learning Theory*, pages 337-346, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1993.
- [BS] A. Blum, M. Singh, “Learning functions of  $k$  terms”, *Proc of the Third Annual Workshop on Computational Learning Theory*, pages 144-153, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.
- [BHH] N. Bshouty, T. Hancock, L. Hellerstein, “Learning boolean read-once formulas with arbitrary symmetric and constant fan-in gates”, *Proc of the 5th Annual Workshop on Computational Learning Theory*, pages 1-15, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1992.
- [B] M. Ben-Ari, “A simplified proof that regular resolution is exponential”, *Inf. Proc. Lett.*, 1980, pages 96-98.
- [CS] V. Chvátal, E. Szemerédi, “Many hard examples for resolution”, *JACM*, 1988, pages 759-768.
- [FS] P. Fisher, H. Simon, “On learning ring-sum-expansions”, *SIAM J. Comput.*, 1992, pages 181-192.
- [G] Z. Galil, “On the complexity of regular resolution and the Davis-Putnam procedure”, *Theor. Comput. Sci.*, 1977, pages 23-46.
- [HRS] D. Helmbold, R. Sloan, M. Warmuth, “Learning integer lattices”, *SIAM J. Comput.*, 1992, pages 240-266.
- [HC] S. Homer, Z. Chen, “On learning parity functions with queries”, Manuscript, July, 1993.
- [HLC] W. Huang, C. Lai, Z. Chen, “The hardness of the grid problem  $G_i$  under the routine resolution method”, *Zeitschr. Math. Logik und Grundlagen Math.*, 1987, pages 79-94.
- [J] J. Jackson, Personal communications.
- [S] R. Schapire, Personal communications.
- [T] G.S. Tseitin, “On the complexity of derivation in propositional calculus”, *Studies in Constructive Mathematics and Mathematical logic, Part 2*, Consultants Burea, New York-London, 1968, pages 115-125.
- [U] A. Urquhart, “Hard examples for resolution”, *J. ACM*, 1987, pages 209-219.