# Learning Read-Constant Polynomials of Constant Degree Modulo Composites

Arkadev Chattopadhyay[1], Ricard Gavaldà[2],
Kristoffer Arnsfelt Hansen[3], and Denis Thérien[4]

[1] University of Toronto, arkadev@cs.toronto.edu
[2] Universitat Politècnica de Catalunya, gavalda@lsi.upc.edu
[3] Aarhus University, arnsfelt@cs.au.dk
[4] McGill University, denis@cs.mcgill.ca

**Abstract.** Boolean functions that have constant degree polynomial representation over a fixed finite ring form a natural and strict subclass of the complexity class $\mathrm{ACC}^0$. They are also precisely the functions computable efficiently by *programs* over fixed and finite nilpotent groups. This class is not known to be learnable in any reasonable learning model. In this paper, we provide a deterministic polynomial time algorithm for learning Boolean functions represented by polynomials of constant degree over arbitrary finite rings from membership queries, with the additional constraint that each variable in the target polynomial appears in a constant number of monomials. Our algorithm extends to superconstant but low degree polynomials and still runs in quasipolynomial time.

## 1 Introduction

Understanding the computational power of computation over rings of the form $\mathbb{Z}_m$, for an arbitrary composite number $m$, is a fundamental open problem. A concrete and natural setting in which to explore this power is the model of representing Boolean functions by low degree polynomials over such rings, in the following sense [4]: an assignment to the variables is a 1 of the Boolean function if and only if the polynomial on it evaluates to an element of a prespecified accepting subset of the ring.

When the modulus is a prime number and the ring thus turns into a finite field, our knowledge of representations is far better than the general case. For instance, it is known that degree $\Omega(n)$ is required in order to represent the Boolean function $\mathrm{MOD}_q$ by polynomials over the field $\mathbb{Z}_p$, when $p$ is a prime and $q$ has a prime factor different from $p$. The stronger result that $\mathrm{MOD}_q$ remains hard to even approximate well by such polynomials of low degree, is a key insight in the celebrated lower bound of Razborov [22] and Smolensky [24] on the size of bounded-depth circuits.

In contrast, we do not even know the exact degree of the Parity function for polynomials over $\mathbb{Z}_m$, as soon as $m$ is an odd number having two distinct prime factors. In a beautiful work, Barrington, Beigel and Rudich [4] showed that composite moduli give non-trivial advantage to polynomials as compared

to prime moduli. More precisely, they showed that the degree of the OR and the AND function over $\mathbb{Z}_m$ is $O\big(n^{1/t}\big)$ if $m$ has $t$ distinct prime factors. On the other hand, it is well known that if $m$ is a fixed prime, then this degree is $\Omega(n)$. This surprising construction of Barrington *et al.* has found diverse applications. Indeed, Efremenko [14] recently built efficient locally decodable codes from it. Also, Gopalan [16] shows that several previously known constructions of explicit Ramsey graphs can all be derived from this construction.

The best known lower bounds on the composite degree of any Boolean function is $\Omega(\log n)$ (see for example [17, 25, 9] and the survey [13]). Proving anything better is a tantalizingly open problem. In this work, we look at low degree polynomials through the lens of computational learning theory. The motivation and hope is that this approach will lead to new insights into the structure of these polynomials, thus benefiting both the fields of learning theory and complexity theory.

Given that we know degree lower bounds of $\Omega(\log n)$, it is reasonable to hope that we can learn functions represented by constant degree polynomials. We take on this task in this paper in the setting where the learner is allowed to ask membership queries. The main difficulty that one faces is essentially the same that confronts one when proving lower bounds on the degree: while computation by the target polynomial takes place in the entire ring $\mathbb{Z}_m$, the information revealed to the learner is just Boolean. That is, we learn only whether the polynomial when evaluated on the chosen point yields an element of the unknown accepting set. Although several equivalent low degree representations may exist for the target concept, it is a non-trivial fact that polynomially many such queries are able to isolate a unique function in the concept class that agrees with the answers of the teacher. The computational challenge, of course, is to recognize this unique function.

**Our Result** We consider the concept class of functions that have a representation by a constant degree polynomial in which every variable appears in a constant number of monomials. We show that this class is exactly learnable in polynomial time from the values of the target function at all input assignments of Hamming weight bounded by another constant. These values can be obtained, in particular, from membership queries. Additionally, our learning algorithm is proper in the sense that it outputs a constant degree polynomial equivalent to the target polynomial with respect to the Boolean function they compute. It is worth remarking that there are very few instances in which concepts are known to be properly learnable, especially when there is no guarantee of a unique representation.

**Overview of Our Techniques** Our learning algorithm uses some novel ideas exploiting the following structural property of low degree polynomials first discovered in the work of Péladeau and Thérien [20] (see the translation [21]): for every constant degree polynomial $P$ over any fixed finite commutative ring with identity, there exists a "magic set" of variables of constant cardinality such that

every value in the range of $P$ can be attained by only setting a subset of variables from the magic set to 1 and setting all other variables to 0. This property is very convenient and in particular, implies that every Boolean function that can be represented by a constant degree polynomial gets uniquely determined by the values it takes on points of constant Hamming weight. It is worthwhile to note that although the function gets fixed by knowing its behavior on all low weight points, it is not clear how to efficiently determine the value of this function on any other input point of the Boolean cube. This is the essential challenge that the learning algorithm has to overcome.

To be more specific, using this magic set we define an equivalence relation among monomials of the same degree. We show that there always exists a polynomial representing the same function that the teacher holds, in which all monomials belonging to the same equivalence class have identical coefficients. The number of equivalence classes is upper bounded by a constant and there is a very efficient test of equivalence. These properties allow us to enumerate all possible values of coefficients and then choose any that satisfies the polynomially many points of constant weight.

**Relations to Existing Work** Polynomials have been widely studied in learning theory. When the learner can use evaluation queries returning the precise value of the polynomial over the base ring or field, polynomials of arbitrary degree over finite fields and even finite rings can be learned from evaluation+equivalence queries [23, 8, 12]. On the other hand, when the accepting set of the target polynomial is guaranteed to be a singleton set, it can be learned in the PAC model, and also approximately from membership queries alone, by a variation of the subspace-learning algorithm in [18] (see also [15]); this holds for all finite (and many infinite) rings. For the field $\mathbb{Z}_p$, a standard use of Fermat's little theorem shows that every polynomial of degree $d$ with an arbitrary accepting set can be turned into an equivalent polynomial of degree $d(p-1)$ whose range is $\{0,1\}$; this allows us to learn polynomials of constant degree over $\mathbb{Z}_p$ both in the PAC model, as above, and exactly from membership queries.

In this paper we make progress, for the first time to our best knowledge, in the equivalent learning problem for the non-field case. Note however that the problem was mentioned in [15], where the degree 1 case was solved by a technique that does not seem to extend to higher degrees. The emphasis in [15] was the classification of families of Boolean functions computed by *programs* over finite monoids (cf. [3, 7, 6]), with respect to their learnability in different models. In this setting, polynomials of constant degree over finite rings are equivalent in power to programs over nilpotent groups (as shown in [20]) with degree-1 polynomials corresponding to programs over Abelian groups. The class of functions computed by such programs is a natural subclass of functions computable by programs over solvable groups. Starting with the famous and surprising work of Barrington [3] that showed the class of functions computed by polynomial length programs over finite non-solvable groups is exactly the complexity class $\mathrm{NC}^1$, programs

over groups, or monoids in general, have been used (see for example [7, 6]) to characterize natural subclasses of $\mathrm{NC}^1$.

## 2 Preliminaries

### 2.1 Polynomials over Finite Rings

Let $\mathcal{R}$ be a commutative finite ring with unit, and let $P(x_1, \ldots, x_n)$ be a polynomial over $\mathcal{R}$. We say $P$ is a *read-$k$* polynomial, if every variable in $P$ appears in at most $k$ monomials of $P$.

Consider a family of polynomials $\mathcal{P} = \{P_i\}_{i=1}^{\infty}$, where $P_i$ is a polynomial in $i$ variables. We say the family $\mathcal{P}$ is *read-constant*, if there exist a $k$ such that every $P_i \in \mathcal{P}$ is read-$k$. Similarly, we say that $\mathcal{P}$ is *constant degree* if there exists $d$ such that every $P_i \in \mathcal{P}$ is of degree at most $d$.

In this work, we will restrict our attention to variables ranging over the set $\{0,1\} \subseteq \mathcal{R}$, and as a consequence we can without loss of generality restrict our attention to *multilinear* polynomials. Formally we consider the ring of polynomials $\mathcal{R}[x_1, \ldots, x_n]/N$, where $N$ is the ideal generated by the set of polynomials $\{x_i^2 - x_i \mid i = 1, \ldots, n\}$. Any function $\{0,1\}^n \to \mathcal{R}$ is uniquely expressed by such a polynomial. Define the *range* of $P$ as $\mathrm{range}(P) = \{r \in \mathcal{R} \mid \exists x \in \{0,1\}^n : P(x) = r\}$.

Equipping a polynomial $P$ with an *accepting set* $A \subseteq \mathcal{R}$, we say that the pair $(P, A)$ computes a Boolean function $f : \{0,1\}^n \to \{0,1\}$ if it holds that $P(x) \in A$ if and only if $f(x) = 1$, for all $x \in \{0,1\}^n$.

Given a set of indices $J \subseteq [n]$, we let $\chi_J \in \{0,1\}^n$ denote the characteristic vector of $J$. Conversely, for $w \in \{0,1\}^n$, define $I_w = \{i \in [n]\} \mid w_i = 1\}$. Thus $\chi_{I_w} = w$ and $I_{\chi_J} = J$. For $u, v \in \{0,1\}^n$, let $u \vee v \in \{0,1\}^n$ be defined by $I_{u \vee v} = I_u \cup I_v$.

Consider now a degree $d$ polynomial, $P(x) = \sum_{I \subset [n], |I| \leq d} c_I \prod_{i \in I} x_i$. For a subset $S \subseteq [n]$ we define the polynomial $P_S$ of monomials from $S$ by, $P_S(x) = \sum_{I \subseteq S, |I| \leq d} c_I \prod_{i \in I} x_i$. For disjoint subsets $S, T \subseteq [n]$ define the polynomial $P_{S \times T}$ consisting of cross terms between $S$ and $T$:

$$P_{S \times T}(x) = \sum_{I, J \neq \emptyset; I \subseteq S, J \subseteq T; |I| + |J| \leq d} c_{I \vee J} \prod_{i \in I \cup J} x_i \ .$$

For a polynomial we associate the graph $G_P$ defined as follows. The set of vertices of $G_p$ is $\{1, \ldots, n\}$ and the set of edges is $E(G_P) = \{(i,j) \mid x_i \text{ and } x_j$ appear together in some monomial of $P$ $\}$ This will allow us to speak of the *distance* between variables of $P$, namely as distances in the graph $G_P$.

### 2.2 Structural Properties of Polynomials

Using an inductive Ramsey-theoretic argument, the following important structural result about constant degree polynomials over finite rings was proved by Péladeau and Thérien [20].

**Theorem 1 (Péladeau and Thérien).** *Let $\mathcal{R}$ be a finite commutative ring with unity and let $d$ be any number. Then there exists a constant $c = c(\mathcal{R}, d)$ with the following property: For any multilinear polynomial $P$ over $\mathcal{R}$ of degree at most $d$ and for any $r \in \text{range}(P)$ there exists $w \in \{0,1\}^n$ with $|I_w| \leq c$ such that $P(w) = r$.*

*Remark 2.*
- The theorem as stated above is actually only implicitly given in the proof of Lemma 2 of [20].
- In Sect. 4 we shall present with full proof a quantitative strengthening of the theorem based on a result of Tardos and Barrington [25].

Two easy consequences of this theorem are given below. Our learning algorithm will be heavily based on these results.

**Corollary 3.** *There exists a constant $s = s(\mathcal{R}, d)$, such that for every multilinear polynomial $P$ over $\mathcal{R}$ of degree at most $d$, there exists a set $J \subset \{1, \ldots, n\}$ with the following properties: (1). $|J| \leq s$. (2). For every $r \in \text{range}(P)$ there exists $w \in \{0,1\}^n$ with $I_w \subseteq J$ such that $P(w) = r$.*

*Proof.* Let $s = |\mathcal{R}|c(\mathcal{R}, d)$, with $c(\mathcal{R}, d)$ as given by Theorem 1. We can then simply take $J$ to be the union of $|\text{range}(P)|$ sets $I_w$ provided by Theorem 1 for each $r \in \text{range}(P)$. □

For a given polynomial $P$ we will refer to the set $J$ as guaranteed above to exist as the *magic set* set of variables for $P$.

**Corollary 4.** *There exists a constant $c' = c'(\mathcal{R}, d)$ with the following property: Let $P$ and $Q$ be polynomials of degree at most $d$ with accepting sets $A$ and $B$, respectively. If the Boolean functions computed by the pairs $(P, A)$ and $(Q, B)$ agree on all inputs $w \in \{0,1\}^n$ with $|I_w| \leq c'$, then the two Boolean functions are identical.*

*Proof.* We take $c' = c(\mathcal{R} \times \mathcal{R}, d)$ as given by Theorem 1. Now, write $P(x) = \sum c_I \prod_{i \in I} x_i$ and $Q(x) = \sum d_I \prod_{i \in I} x_i$. Consider the polynomial $(P \times Q)$ over $\mathcal{R} \times \mathcal{R}$ given by $(P \times Q)(x) = \sum (c_I, d_I) \prod_{i \in I} x_i$ If $(P, A)$ and $(Q, B)$ do not compute the same Boolean function there is $(r, s) \in \text{range}(P \times Q)$ such that either $r \in A$ and $s \notin B$ or $r \notin A$ and $s \in B$. Then by Theorem 1 and the choice of $c'$ this would be witnessed by a $w \in \{0,1\}^n$ with $|I_w| \leq c'$ such that $(P \times Q)(w) = (r, s)$. □

## 3 Learning with Membership Queries

In this section we will present our algorithm for learning read-constant, constant degree polynomials. For convenience we choose to present the algorithm as a nondeterministic algorithm, that when terminating with success always output a correct polynomial. Afterwards we will be able to convert this nondeterministic algorithm into a deterministic algorithm simply by enumerating over all possible

sequences of guesses of the algorithm, arguing that there are only polynomially many such sequences.

For ensuring that the nondeterministic algorithm always produces a correct output we use a consistency check procedure, described as Algorithm 1.

**Algorithm 1: Consistent**$(Q, A, f)$

**Input**: Polynomial $Q$ with accepting set $A \subseteq \mathbb{Z}_m$. Membership query access to Boolean function $f$.

**Output**: Decides if the pair $(Q, A)$ computes the function $f$.

1: Query $f$ on all $w \in \{0,1\}^n$ with $|I_w| \leq c'(\mathcal{R}, d)$

2: Return true if and only if for each queried $w$, $f(w) = 1$ if and only if $Q(w) \in A$

The correctness of the procedure is immediate from Corollary 4.

### 3.1 Equivalence Relations Between Monomials

For our algorithm we need the following somewhat technical definition of parameterized equivalence relations of monomials. Intuitively, they serve the following purpose: we want to learn an unknown polynomial, singling it out from exponentially many possibilities. One way to reduce this huge search space is to deduce, from membership queries, that some of the $n^{O(d)}$ coefficients must be the same, as they can only have a constant ($|R|$) number of values. Equivalence among two monomials, as defined below, is intended to suggest that they define isomorphic subpolynomials of the target polynomial.

For example, if a target polynomial contains terms $2x_1$, $3x_2$, $x_1x_2$, $2x_3$, $3x_4$, $x_3x_4$ we would like to say that monomials $x_1x_2$ and $x_3x_4$ are equivalent, and when searching for coefficients for these monomials we can discard all settings of coefficients where they differ.

The idea of the learning algorithm, to be explained in more detail in the next section, is to first implement equivalent tests among all monomials and then, on the basis of this information, actually find the values of all coefficients exploring a polynomial search space rather than an exponential one.

For any monomial $M$, let $I_M = \{i \mid x_i \text{ appears in } M\}$. Conversely, for any set of indices $I$ let $M_I$ denote the monomial $\Pi_{i \in I} x_i$.

Given a polynomial $P$ and a set $J$ of indices in $\{1, \ldots, n\}$, we define a parameterized equivalence relation $\sim_{d,J}$ on tuples $(M, \preceq)$, where $M \subset [n], M \cap J = \emptyset$, $|M| = d$, and $\preceq$ is a total ordering[5] on $[n]$, by induction on $d$. We say $(M, \preceq_1) \sim_{d,J} (M', \preceq_2)$ if the following is satisfied:

1. For every assignment $w$, such that $I_w \subseteq J$, we have $P(w \vee \chi_M) \in A$ if and only if $P(w \vee \chi_{M'}) \in A$.

[5] Alternatively one could fix the same ordering, say $1, \ldots, n$ for all monomials. However we find it natural to identify monomials that that are identical up to a permutation of the variables.

2. Let $M_1, \ldots, M_d$ (and $M'_1, \ldots, M'_d$) be the subsets of $M$ ($M'$) of size $d-1$ listed in the lexicographic order w.r.t $\preceq_1$ ($\preceq_2$). Then $(M_i, \preceq_1) \sim_{d-1,J} (M'_i, \preceq_2)$, for all $i \leq d$.

For every pair of monomials $M$ and $M'$, each of degree $d$, with $I_M, I_{M'}$ disjoint from $J$, we say that $M \equiv_{d,J} M'$ if there exist $\preceq_1$ and $\preceq_2$ such that $(I_M, \preceq_1) \sim_{d,J} (I_{M'}, \preceq_2)$.

### 3.2 Idea of the Learning Algorithm

Let $P$ be a polynomial with accepting set $A$. Let $\mathcal{M}_{\text{range}(P)}$ be the subgroup of the additive subgroup of $\mathcal{R}$ generated by range($P$) (i.e. the $\mathbb{Z}$-module generated by range($P$)). Consider next the following equivalence relation on monomials of a polynomial $P$

For tuples $(M, \preceq_1)$ and $(M', \preceq_2)$, where $M, M' \subset [n]$ and $\preceq_1$ and $\preceq_2$ are total orderings, we say $(M, \preceq_1)\widehat{\sim}_d(M', \preceq_2)$ if the following is satisfied:

1. For every $r \in \mathcal{M}_{\text{range}(P)}$, $r + P(\chi_M) \in A$ if and only if $r + P(\chi_{M'}) \in A$.
2. Let $M_1, \ldots, M_d$ (and $M'_1, \ldots, M'_d$) be the subsets of $M$ ($M'$) of size $d-1$ listed in the lexicographic order w.r.t $\preceq_1$ ($\preceq_2$). Then $(M_i, \preceq_1)\widehat{\sim}_{d-1}(M'_i, \preceq_2)$, for all $i \leq d$.

For every pair of monomials $M$ and $M'$, each of degree $d$, we say that $M\widehat{\equiv}_d M'$ if there exist $\preceq_1$ and $\preceq_2$ such that $(I_M, \preceq_1)\widehat{\sim}_d(I_{M'}, \preceq_2)$.

Assume now (by induction) that in $P$, that for all monomial $M_1$ and $M_2$ of degree $s < r$ we have $c_{M_1} = c_{M_2}$, whenever $M_1\widehat{\equiv}_s M_2$. Next consider monomials $M$ and $M'$ with $M\widehat{\equiv}_r M'$, and let $P'$ be the polynomial obtained from $P$ by replacing the coefficient of $M'$ with the coefficient of $M$. By our (inductive) assumption we have $P(\chi_M) = P'(\chi_{M'})$. Let $x \in \{0,1\}^n$ be arbitrary. Now, since $P(x), P(\chi_{M'}) \in \text{range}(P)$ we have $r = P(x) - P(\chi_{M'}) \in \mathcal{M}_{\text{range}(P)}$. We then have $r + P(\chi_M) \in A$ if and only if $r + P(\chi_{M'}) \in A$. But $r + P(\chi_{M'}) = (P(x) - P(\chi_{M'})) + P(\chi_{M'}) = P(x)$ and $r + P(\chi_M) = (P(x) - P(\chi_{M'})) + P(\chi_M) = P(x) - P(\chi_{M'}) + P'(\chi_{M'}) = P'(x)$. Hence $P(x) \in A$ if and only if $P'(x) \in A$.

Thus, if we would be able to actually *implement* testing of the above equivalence relation, we would be have a simple learning algorithm as follows: First compute all equivalence classes. Then enumerate all candidate polynomials obtained by all possible coefficients for these equivalence classes, and test for correctness using the **Consistent** procedure. We do not know how to accomplish this. However using the notion of a magic set, we *are* in fact able to implement (possibly a refinement of) this equivalence relation on $P$ restricted to *all but a constant number of variables.*

### 3.3 Properties of Polynomials Equipped with a Magic Set

Before stating our learning algorithm, we establish a number of properties to be used later for polynomials $P$ equipped with a magic set $J$.

**Lemma 5.** *Let $P(x) = \sum_{I\subseteq[n],|I|\leq d} c_I \prod_{i\in I} x_i$ be any polynomial over $\mathcal{R}$, with a magic set $J$. Let $N$ be the set of indices that (viewed as vertices in the graph $G_P$) are at distance at least 2 from $J$ in $G_P$. Then, $r + \sum_{I\subseteq N,|I|\leq d} \lambda_I c_I \in \mathrm{range}(P)$, for all $r \in \mathrm{range}(P)$ and all $\lambda_I \in \{0,\ldots,|\mathcal{R}|-1\}$.*

*Proof.* We prove the statement by induction, first by induction on the degree $d$, and then by further induction on the monomials of degree $d$. We take as our induction hypothesis that $r + \sum_{I\subseteq N,|I|<d} \lambda_I c_I \in \mathrm{range}(P)$ for all $r$ and $\lambda_I$. The base case $d = 0$ trivially holds. Consider now for the inductive step the case $d+1$. Enumerate all $\binom{|N|}{d}$ subsets of $N$ of cardinality $d$, and let $I^i$ denote the $i$th set in this enumeration. We shall now further induct on $k$ to show that $r + \sum_{I\subseteq N,|I|<d} \lambda_I c_I + \sum_{i=1}^{k} \lambda_{I^i} c_{I^i} \in \mathrm{range}(P)$ for every $r \in \mathrm{range}(P)$. The $k = 0$ base case trivially holds. For the inductive step, we want to show that $r + \sum_{I\subseteq N,|I|<d} \lambda_I c_I + \sum_{i=1}^{k} \lambda_{I^i} c_{I^i} + \lambda c_{I^{k+1}} \in \mathrm{range}(P)$. By induction hypothesis, there exists $u$ with $I_u \subset J$ such that $P(u) = r + \sum_{I\subset L,|I|<d} \lambda_I c_I + \sum_{i=1}^{k} \lambda_{I^i} c_{I^i} = r_0$. Then clearly, $P(u \vee \chi_{I^{k+1}}) = r_0 + \sum_{I\subset I^{k+1}} c_I + c_{I^{k+1}} = r_1 \in \mathrm{range}(P)$. Hence, there is $u_1$ with $I_{u_1} \subseteq J$ such that $P(u_1) = r_1$. Continuing in this way for $\lambda$ times we see that $r_\lambda = r_0 + \lambda \cdot (\sum_{I\subset I^{k+1}} c_I) + \lambda c_{I^{k+1}} \in \mathrm{range}(P)$. Applying once more our outer induction hypothesis, we conclude $r_\lambda + (|\mathcal{R}| - \lambda)(\sum_{I\subset I^{k+1}} c_I) = r_0 + \lambda \cdot c_{I^i} \in \mathrm{range}(P)$. This completes the inner and outer induction. □

For a polynomial $P$ with accepting set $A$ we can always obtain equivalent polynomial in which the constant term is 0 by *shifting* the accepting set according to the constant term. Thus in the following assume the constant term of $P$ is 0. Let $J$ be a magic set of $P$. Let $N$ be the set of indices that are at distance 2 or more from $J$ in $G_P$. Let $P_N$ be the polynomial obtained from $P$ by fixing to 0 every variable indexed in the set $[n] \setminus N$.

The crucial insight required for limiting the amount of nondeterministic guesses in our learning algorithm is expressed in the following lemma.

**Lemma 6.** *Let $P$ be any polynomial of degree $d$ with accepting set $A$ and a magic set $J$, and let $r \leq d$. Assume that for all monomials $M_1$ and $M_2$ in $P_N$ of degree $s < r$ we have $c_{M_1} = c_{M_2}$ whenever $M_1 \equiv_{s,J} M_2$. Consider now monomials $M$ and $M'$ of degree $r$ in $P_N$ such that $M \equiv_{r,J} M'$. Let $P'$ be the polynomial obtained from $P$ by replacing the coefficient of $M'$ with the coefficient of $M$. Then the polynomials $P$ and $P'$ compute the same Boolean function.*

*Proof.* Since $M \equiv_{r,J} M'$, we have $(I_M, \preceq_M) \sim_{r,J} (I_{M'}, \preceq_{M'})$ for some $\preceq_M$ and $\preceq_{M'}$. Let us enumerate lexicographically the subsets of $I_M$ and $I_N$ according to $\preceq_M$ and $\preceq_{M'}$. Let $M_i$ and $M'_i$ be the monomial corresponding to the $i$th such subsets and let $d_i$ denote their degree. By definition we have $M_i \equiv_{d_i,J} M'_i$, and so by assumption the coefficients of $M_i$ and $M'_i$ are the same. We thus have that $P(\chi_M) = P'(\chi_{M'})$.

To prove that $P$ and $P'$ with accepting set $A$ compute the same Boolean function, let $x \in \{0,1\}^n$ be arbitrary. Obviously, $P(x) \in \mathrm{range}(P)$. By Lemma 5 we then have that also $P(x) - P(\chi_{M'}) \in \mathrm{range}(P)$. It follows there exist $u$ with $I_u \subseteq J$ such that $P(u) = P(x) - P(\chi_{M'})$. Since $M \equiv_{r,J} M'$ we have $P(u \vee \chi_M) =$

$P(u) + P(\chi_M) \in A$ if and only if $P(u \vee \chi_{M'}) = P(u) + P(\chi_{M'}) \in A$. But $P(u) + P(\chi_M) = P(x) - P(\chi_{M'}) + P(\chi_M) = P(x) - P(\chi_{M'}) + P'(\chi_{M'}) = P'(x)$ and $P(u) + P(\chi_{M'}) = P(x)$. Hence we can conclude that $P(x) \in A$ if and only if $P'(x) \in A$. □

### 3.4 The learning Algorithm

We are now finally in position to state our algorithm.

**Algorithm 2: Learn-Poly**(f)

**Input**: Membership query access to Boolean function $f$.
**Output**: Returns pair $(Q, A)$ computing the function $f$, or **fail**.

1: Nondeterministically guess the following:
   A magic set $J \subseteq [n]$, $|J| \leq s(\mathcal{R}, d)$ and polynomial $Q_J$.
   The set $K \subseteq [n]$ at distance 1 in $G_P$ from $J$ and polynomials $Q_K$ and $Q_{K \times J}$.
   The set $L \subseteq [n]$ at distance 2 in $G_P$ from $J$, and polynomial $Q_{K \times L}$.
   An accepting set $A \subseteq \mathcal{R}$ for $Q$.
2: Let $N = [n] \setminus (J \cup K)$.
3: Query $f$ on all inputs $(w \vee x)$ where $I_w \subseteq J$, $I_x \subseteq [n] \setminus N$, and $|I_x| \leq 2d$.
4: Compute the equivalence classes of $\equiv_{r,J}$, for all $r = 1, \ldots, d$, over monomials $M_I$ with $I \subseteq N$ and $|I| \leq d$.
5: Nondeterministically guess an element of $\mathcal{R}$ for each equivalence class.
6: Construct polynomial $Q = Q_J + Q_K + Q_{J \times K} + Q_{K \times L} + \sum_{I \subset N, |I| \leq d} c_I \cdot M_I$, where $c_I \in \mathcal{R}$ is the element guessed for the equivalence class of monomial $M_I$.
7: If **Consistent**$(Q, A, f)$, output $(Q, A)$, otherwise output fail.

**Theorem 7.** *Let $\mathcal{R}$ be a fixed commutative finite ring with unit. Let $F$ be the class of Boolean functions that can be computed by read-constant and constant degree polynomials.* **Learn-poly** *non-deterministically learns exactly any function $f \in F$ in polynomial time.*

*Proof.* Take a computation path of **Learn-poly** in which it made right guesses in step 1. Then using Lemma 6, we maintain an equivalent polynomial if the guesses for coefficients of each equivalence class is correct. If incorrect guesses result in a wrong candidate polynomial it will be detected by the **Consistent** procedure. □

It is easy to see that a deterministic variant of **Learn-poly** can be derived and it runs in poly-time. This can be done by simply going through all possible guesses. Since cardinality of $J$ is bounded by a constant (using Corollary 3) determined by the degree of the polynomial, there are only polynomially many sets to guess. Since $P$ is read-$k$ for some constant $k$, $|K| \leq k(d-1)|J|$, and the number of guesses for $K$ is at most $\binom{n}{k(d-1)|J|}$, which is again polynomial in $n$. Observe that the size of $K$ is also bounded by a constant. Thus, guessing

$P_J, P_K, P_{J\times K}$ involves at most $|\mathcal{R}|^s$ guesses, where $s$ is the number of monomials of degree at most $d$ involving variables indexed by set $K \cup J$. A similar argument shows that polynomially many guesses are needed to get the correct $L$ for each possible $K$ and then constantly many guesses for a given $K$ and $L$ are involved for $P_{K\times L}$. Since the equivalence relation $\equiv_{d,J}$ is finite indexed for each $d$, constantly many guesses have to be enumerated to also make this deterministic and we are done.

## 4 Extensions to Higher Degrees

For a Boolean function $f$ on $n$ variables, define $\Delta(f, \mathcal{R})$ to be the minimal degree of a polynomial over $\mathcal{R}$ computing $f$. Consider a family of Boolean functions $f = \{f_n\}_{n=1}^{\infty}$, one for each input length. Define $\Delta(f, \mathcal{R}, n) = \Delta(f_n, \mathcal{R})$. Define $\Lambda(f, \mathcal{R}, d)$ as the maximal $n$ such that $\Delta(f, \mathcal{R}, n) \leq d$.

The notion of the degree of the Boolean AND function allows the following quantitative version of Theorem 1.

**Proposition 8.** $c(\mathcal{R}, d) \leq \Lambda(\mathrm{AND}, \mathcal{R}, d)$

*Proof.* Let $P$ be a multilinear polynomial of degree $d$ over $\mathcal{R}$ in $n$ variables. Let $r \in \mathrm{range}(P)$. We will find $w \in \{0,1\}^n$ with $|I_w| \leq \Lambda(\mathrm{AND}, \mathcal{R}, d)$ such that $P(w) = r$. If $P(0) = r$, we are done. Otherwise, pick $w \in \{0,1\}^n$ such that $|I_w|$ is minimal with $P(w) = r$. Consider now the restriction $P'$ of $P$ to the variables indexed by $I_w$. By minimality of $|I_w|$, we have that $P'$ computes the AND function with accepting set $\{r\}$ on $|I_w|$ variables. Thus it follows that $|I_w| \leq \Lambda(\mathrm{AND}, \mathcal{R}, d)$. □

As a consequence we obtain the following bounds for $s(\mathcal{R}, d)$-the size of the magic set for polynomials over $\mathcal{R}$ of degree $d$, and $c'(\mathcal{R}, d)$-the Hamming weight of assignments that uniquely identify a Boolean function represented by such a polynomial, in terms of $\Lambda(\mathrm{AND}, \mathcal{R}, d)$ as well, following the proofs of Corollary 3 and Corollary 4.

**Corollary 9.** $s(\mathcal{R}, d) \leq |\mathcal{R}|\Lambda(\mathrm{AND}, \mathcal{R}, d)$ *and* $c'(\mathcal{R}, d) \leq \Lambda(\mathrm{AND}, \mathcal{R} \times \mathcal{R}, d)$.

Thus lower bounds for the degree of the AND function implies upper bounds on the above quantities. The degree of the AND function have been intensively studied over the ring $\mathbb{Z}_m$ [4, 25]. Let in the following $m = p_1^{k_1} \cdots p_r^{k_r}$ have $r$ distinct prime factors, and let $q_{\min} = \min(p_1^{k_1}, \ldots, p_r^{k_r})$ and $q_{\max} = \max(p_1^{k_1}, \ldots, p_r^{k_r})$. With these definitions, Tardos and Barrington [25] obtained the following lower bound, which is currently the best known.

**Theorem 10 (Tardos and Barrington).**
$\Delta(\mathrm{AND}, \mathbb{Z}_m, n) \geq ((1/(q_{\min} - 1) - o(1)) \log n)^{1/(r-1)}$.
*Equivalently,* $\Lambda(\mathrm{AND}, \mathbb{Z}_m, d) \leq 2^{(q_{\min}-1+o(1))d^{r-1}}$

For the purpose of our learning algorithm we are interested in bounds for the ring $\mathcal{R} = \mathbb{Z}_m^l$. In fact, without loss of generality we may assume that $\mathcal{R}$ is of this form. We can transfer the above results to this ring using standard methods. Let $m' = p_1 \cdots p_r$. Let $p_{\min} = \min(p_1, \ldots, p_r)$.

**Lemma 11.** $\Delta(\mathrm{AND}, \mathbb{Z}_{m'}, n) \leq l(q_{\max} - 1)\Delta(\mathrm{AND}, \mathbb{Z}_m^l, n)$.
*Equivalently,* $\Lambda(\mathrm{AND}, \mathbb{Z}_m^l, d) \leq \Lambda(\mathrm{AND}, \mathbb{Z}_{m'}, l(q_{\max} - 1)d)$.

*Proof.* Let $P$ be a polynomial over $\mathbb{Z}_m^l$ of degree $d$ computing the AND function. Without loss of generality, the accepting set is $\{0\}$. Let $P_1, \ldots, P_l$ be the $l$ coordinate polynomials. Consider a fixed $j$, and the polynomials $P_1, \ldots, P_l$ modulo $p_j^{k_j}$. By well known arguments (see e.g [25]) we can find polynomials $Q_1^j, \ldots, Q_l^j$ of degree at most $(p_j^{k_j} - 1)d$ such that $Q_i^j(x) \equiv 0 \pmod{p_j}$ if and only if $P_i(x) \equiv 0 \pmod{p_j^{k_j}}$, and furthermore $(Q_i^j(x) \mod p_j) \in \{0, 1\}$ for all $x$.

Define $Q^j(x) = 1 - \prod_{i=1}^{l}(1 - Q_i^j(x))$. We then have $Q^j(x) \equiv 0 \pmod{p_j}$ if and only if $P_i(x) \equiv 0 \pmod{p_j^{k_j}}$ for all $x$ and $i$. Note the degree of $Q^j$ is at most $l(p_j^{k_j} - 1)d$.

Considering all such polynomials, $Q^1, \ldots, Q^l$, from the Chinese Remainder Theorem we may find a polynomial $Q$, of degree at most $l(q_{\max} - 1)d$ such that $Q(x) \equiv 0 \pmod{m'}$ if and only if $P(x) = 0$ for all $x$. □

Combining Proposition 8, Theorem 10, and Lemma 11 we obtain the following concrete bounds (following the proofs of Corollaries 3 and 4):

**Proposition 12.**

$$c(\mathbb{Z}_m^l, d) \leq \Lambda(\mathrm{AND}, \mathbb{Z}_m^l, d) \leq 2^{(p_{\min} - 1 + o(1))(l(q_{\max} - 1)d)^{r-1}} .$$
$$s(\mathbb{Z}_m^l, d) \leq |\mathbb{Z}_m^l| c(\mathbb{Z}_m^l, d) \leq m^l 2^{(p_{\min} - 1 + o(1))(l(q_{\max} - 1)d)^{r-1}} .$$
$$c'(\mathbb{Z}_m^l, d) \leq c(\mathbb{Z}_m^{2l}, d) \leq 2^{(p_{\min} - 1 + o(1))(2l(q_{\max} - 1)d)^{r-1}} .$$

We now provide a brief analysis of the running time of the deterministic version of our algorithm **Learn-Poly**, presented in the last section, in terms of parameters $s(\mathcal{R}, d)$ and $c'(\mathcal{R}, d)$. The algorithm asks membership queries on points of Hamming weight at most $c'(\mathcal{R}, d) + 2d$. Thus, $O(n^{c'+2d})$ many membership queries are asked in total. The algorithm runs over all possible choices of a magic set $J$ of size $s = s(\mathcal{R}, d)$, the set $K$, of size $ks(d-1)$, of variables that are at distance 1 from the set $J$ and set $L$, of size $k^2 s(d-1)^2$, at distance 2 from $J$. Thus the total number of such choices is at most $n^{k^2 s d^2}$. For each such choice of $J, K, L$, the algorithm considers all possible degree $d$ polynomials of variables indexed in $J \cup K \cup L$. Thus, it has to consider at most $|\mathcal{R}|^{d(k^2 s d^2)^d}$ many polynomials. Further, for each choice of $J, K, L$ it does equivalence testing for monomials that are free of variables indexed by $J$ or $K$. There are at most $dn^d$ such monomials and the test for each involves $2^s$ assignments of variables in $J$. Thus, equivalence testing takes $O\big(dn^d \cdot 2^s\big)$ time. It is not hard to see that

degree $d$ monomials split up into at most $|\mathcal{R}|^{2^d}$ equivalence classes. Thus, one has to consider all possible ways of coloring equivalence classes with elements of $\mathcal{R}$ giving rise to $|\mathcal{R}|^{d|\mathcal{R}|^{2^d}}$ such choices. Finally having guessed an entire candidate polynomial, the algorithm invokes procedure **Consistent** that verifies the consistency of the polynomial with all weight $c' = c'(\mathcal{R}, d)$ assignments. This requires $O\big(n^d \cdot cn^{c'+1}\big)$ time. Summing these up, the total running time is

$$O(2^{|\mathcal{R}|}) \times O(n^{O(k^2 s d^2)}) \times O(|\mathcal{R}|^{d(k^2 s d^2)^d}) \times O(dn^d 2^s) \times O(|\mathcal{R}|^{d|\mathcal{R}|^{2^d}}) \times O\big(n^d \cdot cn^{c'+1}\big)$$

Using Proposition 12, we see that for each $\mathcal{R} = \mathbb{Z}_m^\ell$ with a fixed $m$ and $\ell$, there exists a constant $\gamma$ such that $s(\mathcal{R}, d), c'(\mathcal{R}, d) \leq \gamma^{d^{r-1}}$, where $r$ is the number of distinct prime factors of $m$. Hence, combining the above observations we get the following:

**Theorem 13.** *Let $m$ and $\ell$ be any fixed positive numbers. The class of Boolean functions representable by read-$k$ polynomials of degree $d$ over $\mathbb{Z}_m^\ell$ are exactly learnable from membership queries by a deterministic algorithm of running time $O\big(n^{k^2 \gamma^{d^r}} d^2 \times \gamma^{k^{2d} \gamma^{d^r}} \times \gamma^{\gamma^{2^d}}\big)$, where $\gamma = \gamma(m, \ell)$ is a constant and $r$ is the number of distinct prime factors of $m$.*

Theorem 13 gives us a range of super-constant $k$ and $d$ for which we get sub-exponential running time. For instance, if we choose $k = o(\log \log n)$, and $d = o(\log \log \log n)$, the running time is $n^{(\log n)^{o(1)}}$.

## 5 Future Work

While the progress we make is limited from a learning theory perspective, the combinatorics involved is unexpectedly delicate, and suggests some further questions in understanding the structure of polynomials over rings of the form $\mathbb{Z}_m$.

The obvious next question is to remove the read-constant restriction in our result. Read-constant restrictions have been used, on several occasions, both in complexity theory and in learning theory. For example in complexity theory, Barrington and Straubing [5] proved superlinear bounds on the length of read-constant branching programs of bounded-width. Very recently, several works have been concerned with constructing pseudorandom generators for read-once branching programs of small width [10, 11, 19]. In learning theory, read-constant conditions have been sometimes shown to be unavoidable for efficient learning. For example, read-once Boolean formulas can be learned efficiently from membership and equivalence queries [2]. On the other hand, under cryptographic assumptions, even read-thrice Boolean formulas are impossible to learn no matter what polynomially-evaluatable hypothesis class is used (i.e., hard to learn in a representation-independent way).

In other cases, read-constant conditions for learning a target concept class can be removed at the expense of moving to a larger hypothesis class, which bypasses some computational bottleneck. For example, Aizenstein *et al.* [1] showed

that read-$k$, satisfy-$j$ DNF formulas[6] are learnable (as DNF formulas). Without the read-$k$ condition, satisfy-$j$ DNF formulas are not known to be learnable as DNF, but they can be learned as Multiplicity Automata, as pointed out in [8]. Analogously, it is possible that constant degree polynomials over finite rings can be learned (in some reasonable learning model) by not insisting that the output is itself a constant degree polynomial.

**Acknowledgments** A. Chattopadhyay is partially supported by a Natural Sciences and Engineering Research Council (NSERC) postdoctoral fellowship and research grants of Prof. T. Pitassi. R. Gavaldà is partially funded by the Spanish Ministry of Science and Technology contract TIN-2008-06582-C03-01 (SESAAME), by the Generalitat de Catalunya 2009-SGR-1428 (LARCA), and by the EU PASCAL2 Network of Excellence (FP7-ICT-216886).

## References

1. H. Aizenstein, A. Blum, R. Khardon, E. Kushilevitz, L. Pitt, and D. Roth. On learning read-k-satisfy-j dnf. *SIAM J. Comput.*, 27(6):1515–1530, 1998.
2. D. Angluin, L. Hellerstein, and M. Karpinski. Learning read-once formulas with queries. *J. ACM*, 40(1):185–210, 1993.
3. D. A. M. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in $NC^1$. *J. Comput. Syst. Sci.*, 38(1):150–164, 1989.
4. D. A. M. Barrington, R. Beigel, and S. Rudich. Representing boolean functions as polynomials modulo composite numbers. *Comput. Complexity*, 4:367–382, 1994.
5. D. A. M. Barrington and H. Straubing. Superlinear lower bounds for bounded-width branching programs. *J. Comput. Syst. Sci.*, 50(3):374–381, 1995.
6. D. A. M. Barrington, H. Straubing, and D. Thérien. Non-uniform automata over groups. *Inform. and Comput.*, 89(2):109–132, 1990.
7. D. A. M. Barrington and D. Thérien. Finite monoids and the finite structure of $NC^1$. *J. ACM*, 35(4):941–952, 1988.
8. A. Beimel, F. Bergadano, N. Bshouty, E. Kushilevitz, and S. Varricchio. Learning functions represented as multiplicity automata. *J. ACM*, 47:506–530, 2000.
9. J. Bourgain. Estimation of certain exponential sums arising in complexity theory. *C. R. Acad. Sci. Paris*, 340(9):627–631, 2005.
10. M. Braverman, A. Rao, R. Raz, and A. Yehudayoff. Pseudorandom generators for regular branching programs. In *51th Annual IEEE Symposium on Foundations of Computer Science*, pages 40–47. IEEE Computer Society, 2010.
11. J. Brody and E. Verbin. The coin problem, and pseudorandomness for branching programs. In *51th Annual IEEE Symposium on Foundations of Computer Science*, pages 30–39. IEEE Computer Society, 2010.
12. N. Bshouty, C. Tamon, and D. Wilson. Learning matrix functions over rings. *Algorithmica*, 22:91–111, 1998.
13. A. Chattopadhyay. Multilinear polynomials modulo composites. *Bulletin of the European Association on Theoretical Computer Science, Computational Complexity Column*, 100, February 2010.

[6] A DNF formula is read-$k$ if every variable appears at most $k$ times; a DNF formula is satisfy-$j$ if no assignment satisfies more than $j$ terms simultaneously

14. K. Efremenko. 3-query locally decodable codes of subexponential length. In *41st Annual Symposium on Theory of Computing*, pages 39–44. ACM Press, 2009.
15. R. Gavaldà and D. Thérien. An algebraic perspective on boolean function learning. In *ALT 2009*, LNCS, pages 201–215. Springer, 2009.
16. P. Gopalan. Constructing ramsey graphs from boolean function representations. In *21st Annual IEEE Conference on Computational Complexity*, pages 115–128. IEEE Computer Society, 2006.
17. V. Grolmusz. On the weak mod $m$ representation of boolean functions. *Chicago J. Theoret. Comput. Sci.*, 1995.
18. D. P. Helmbold, R. H. Sloan, and M. K. Warmuth. Learning nested differences of intersection-closed concept classes. *Machine Learning*, 5:165–196, 1990.
19. M. Koucký, P. Nimbhorkar, and P. Pudlák. Pseudorandom generators for group products. In *43rd Annual ACM Symposium on Theory of Computing*. ACM Press, 2011. (to appear).
20. P. Péladeau and D. Thérien. Sur les langages reconnus par des groupes nilpotents. *C. R. Math. Acad. Sci. Paris Sér I Math.*, 306(2):93–95, 1988.
21. P. Péladeau and D. Thérien. On the languages recognized by nilpotent groups (a translation of "sur les langages reconnus par des groupes nilpotents"). *Electronic Colloquium on Computational Complexity (ECCC)*, 8(40), 2001.
22. A. A. Razborov. Lower bounds on the size of bounded-depth networks over a complete basis with logical addition. *Math. Notes of the Acad. of Sci. of USSR*, 41(3):333–338, 1987.
23. R. E. Schapire and L. Sellie. Learning sparse multivariate polynomials over a field with queries and counterexamples. *J. Comput. Syst. Sci.*, 52(2):201–213, 1996.
24. R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *19th Annual ACM Symposium on Theory of Computing*, pages 77–82. ACM Press, 1987.
25. G. Tardos and D. A. M. Barrington. A lower bound on the MOD-6 degree of the OR function. *Comput. Complexity*, 7(2):99–108, 1998.