Symmetric Functions Capture General Functions

Richard J. Lipton
College of Computing
Georgia Tech, Atlanta, GA, USA

Kenneth W. Regan
Department of CSE
University at Buffalo (SUNY)

Atri Rudra*
Department of CSE
University at Buffalo (SUNY)

August 20, 2011

Abstract

We show that the set of all functions is equivalent to the set of all symmetric functions up to deterministic time complexity. In particular, for any function f, there is an equivalent symmetric function f_{sym} such that f can be computed form f_{sym} and vice-versa (modulo an extra deterministic linear time computation). For f over finite fields, f_{sym} is (necessarily) over an extension field. This reduction is optimal in size of the extension field. For polynomial functions, the degree of f_{sym} is not optimal. We present another reduction that has optimal degree "blowup" but is worse in the other parameters.

1 Introduction

Symmetric polynomials have been central in both arithmetic complexity and Boolean circuit complexity. All function families in ACC^0 are known to reduce to symmetric polynomials via small low-depth circuits. The Boolean majority function and related symmetric functions on $\{0,1\}^n$ are hard for low-depth circuit classes, but analogous functions over infinite fields have small constant-depth arithmetic formulas [8]. The best-known lower bounds of $\Omega(n \log n)$ on arithmetic circuit size apply to some simple symmetric functions such as $x_1^n + \cdots + x_n^n$ [5]. Symmetric polynomials have a rich algebraic structure. It is therefore interesting to ask whether they are easier to compute than general polynomials.

Our main results are reductions from a general polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d to symmetric polynomial(s) $g_f \in \mathbb{F}'[y_1, \ldots, y_N]$ of some degree D. Upper and lower bounds are classified according to whether $\mathbb{F}$ is a finite or infinite field (and later, a ring), whether one or more queried polynomials g_f are involved, by the degree s of $\mathbb{F}'$ as an extension of $\mathbb{F}$, and by D and N in relation to the given d and n. Once the best achievable combinations of those parameters are determined, the issues become the running time of the reduction, whether it is computable by low-depth circuits or formulas (with oracle gate(s) for the call(s) to g_f), and whether the entire computation is randomized and/or correct only for a large portion of the input space $\mathbb{F}^n$.

*Supported by NSF CAREER grant CCF-0844796.

1.1 Symmetric Functions—Hard or Easy?

Here are some contexts in which symmetric functions are hard or powerful. Lu [18] remarked that all of the early circuit lower bounds were proved for symmetric functions such as parity and congruence of Hamming weight modulo m [11, 25, 20, 21]. Grigoriev and Razborov [13] (see also [12]) proved exponential lower bounds for depth-3 circuits computing certain symmetric functions over finite fields. Beigel and Tarui [7] showed that every language in ACC^0 can be represented as a Boolean symmetric function in quasi-polynomially many arguments, with each argument a conjunction of inputs x_i or their negations.

In other contexts, symmetric functions are easy. Over GF(2) they depend on only $O(\log n)$ bits of information, and hence have linear-size, $O(\log n)$-depth Boolean circuits. Beame et al. [6] showed that every n-input Boolean symmetric function can be computed by threshold circuits of linear size and $O(\log\log n)$ depth. Over infinite fields, some symmetric polynomials that are hard over finite fields become easy owing to the availability of polynomial interpolation (see [24]). The *elementary symmetric polynomials*, which form an algebra basis for all symmetric polynomials, are easy over all fields, and surprisingly easy in the rings of integers modulo certain composite numbers [14].

So are symmetric functions hard or easy? Or are there contexts in which they have all levels of complexity? In this paper, we prove some results of this last type. Thus we aim to transfer general issues of complexity entirely into the domain of symmetric functions, where we can avail their special algebraic properties.

1.2 Our Results and Techniques

Given a polynomial $f(x_1, \dots, x_n)$ over a field (or ring) $\mathbb{F}$, the objective is to find a symmetric polynomial f_{sym}, possibly over a larger field $\mathbb{F}'$ and/or in a different number m of variables, such that f reduces to f_{sym}, and importantly, vice-versa. The reductions from f to f_{sym} are substitutions, meaning we have functions $\gamma_1, \dots, \gamma_n$ of possibly-new variables $y = y_1, \dots, y_m$ such that

$$f_{\text{sym}}(y) = f(\gamma_1(y), \dots, \gamma_n(y)).$$

Because symmetric functions form an algebra, if $\gamma_1, \dots, \gamma_n$ are symmetric then so is f_{sym}. We presume that the γ_i are easy to compute; whereupon if f is easy then so is f_{sym}. Thus f_{sym} reduces to f. The reverse reduction, to recover values of $f(x)$ from calls to $f_{\text{sym}}(y)$ for suitable y, is the more-problematic one. Let E_f be an encoding function that generates y from x, or E'_f to generate a next y from past history, and let D_f stand for decoding the values of f_{sym}. The procedure in general is:

1. On input x, compute $y = E_f(x)$.
2. Compute $z = f_{\text{sym}}(y)$.
3. Possibly iterate, computing $y^{(j)} = E'_f(x, z, z_2, \dots, z_{j-1})$ and $z_j = f_{\text{sym}}(y^{(j)})$.
4. Output $f(x)$ as a function D_f of the z value(s).

We first make the observation that the reduction above becomes trivial if we do not care about the complexity of the function D_f. In particular, consider the following instantiation of the general procedure above. Define $f_{\text{sym}}(u_1, \dots, u_n) = \sum_{i=1}^{n} u_i$, and for $1 \leqslant j \leqslant n$ define $y^{(j)} \in \mathbb{F}^n$ to be the vector that is zero in every position except position j, where it has the value x_j. Note that by our choices $z_j = x_j$ for every $1 \leqslant j \leqslant n$. Thus, if we pick D_f to be f, then trivially

$D_f(z_1, \ldots, z_n) = f(x)$. Of course this reduction is not interesting as the complexity of the reduction is exactly the same as that of computing f, which is undesirable.

Our first symmetrization makes $z = f(x)$, so D_f is the identity function, but generally needs the encoding function $E_f(x)$ to map into an extension field. For $1 \leqslant j \leqslant m = n$ it takes γ_j to be the j-th elementary symmetric polynomial

$$e_j(x) = \sum_{S \subseteq [n], |S|=j} \prod_{i \in S} x_i, \quad \text{and defines} \quad f_{\text{sym}}(x) = f(e_1(x), \ldots, e_n(x)).$$

Since the e_j are easily computed [14], so is f_{sym}.

To compute $f(b) = f(b_1, \ldots, b_n)$ by reduction to f_{sym}, we find $a = a_1, \ldots, a_n$ such that for each j,

$$b_j = e_j(a_1, \ldots, a_n), \qquad \text{so that} \qquad f(b) = f_{\text{sym}}(a).$$

The values a_j are found by splitting the univariate polynomial

$$\phi_b(X) = X^n + \sum_{i=1}^{n} b_i \cdot X^{i-1}$$

into linear factors. This is guaranteed to be possible only in the splitting field $\mathbb{F}'$ of ϕ_b. In other words, the complexity of computing f from f_{sym} via the reduction above is the same as finding roots of a degree n polynomial over $\mathbb{F}'$. Using known randomized algorithms for computing roots of a univariate polynomial, the (time) complexity of the reduction is determined by the degree of the extension field $\mathbb{F}'$.

Now, the degree of $\mathbb{F}'$ over $\mathbb{F}$ is known to equal the least common multiple of the degrees of the irreducible factors of ϕ_b (see [10]). An easy calculation shows that this degree cannot be more than $n^{O(\sqrt{n})}$. However, this would be too expensive for lower bounds against low-level complexity classes.

Next, we sketch how we get around this predicament. We use the fact proved in [10] that the degree of the splitting field of a *random* monic polynomial is $n^{O(\log n)}$ with high probability. We employ this under two relaxations: (i) If we consider the average-case complexity of f, then it follows immediately that for *most* inputs, f and f_{sym} are equivalent under pseudo-polynomial time reductions. (ii) Under certain restrictions on the degree of f and $|\mathbb{F}|$, we can talk about worst-case complexity of f and f_{sym}. In particular, we use well-known properties of Reed-Muller codes that have been used numerous times in the local testing algorithms and PCP constructions [3, 2, 1, 16, 15]. However, unlike the local testing results which need to handle errors, in our application we only need to handle erasures—roughly because we can efficiently determine the degree of $\mathbb{F}'$ without computing the vector a, which leads to better bounds.

The drawback of this reduction is that the degree of $\mathbb{F}'$ is very large. For comparison, we show by a counting argument that the degree of $\mathbb{F}'$ need only be $O(\log_q n)$, which is super-exponentially better than the bound obtained above. However, note that by construction the degree of f_{sym} is no worse than a factor n more than that of f– we show this to be tight with an *additive* constant factor from the optimal (again by a counting argument).

Our second symmetrization is superior in that it gives linear-time equivalence over any finite field $\mathbb{F}_q$. It is inferior only in giving somewhat higher degree of the resulting symmetric polynomial.

The intuition behind the second reduction is somewhat similar to the "reduction" outline earlier that defined $D_f = f$. In particular for every input value x_i, we will associate it with the pair (i, x_i). (We can do this over extension fields of finite fields and over reals by thinking of (i, x_i) to be the "base n" representation of x_i.) The main idea in defining f_{sym} from f is to convert every input (i, x_i) back to x_i. Doing this while ensuring that f_{sym} is symmetric requires a bit of care.

We compare the two methods in the table below, giving the deterministic reductions only. (We've shortened DTIME to DT to fit the table.)

	f from f_{sym}	f_{sym} from f	s	$\deg(f_{\text{sym}})$
Elem. Sym.	$\mathsf{DT}(n^{O(\sqrt{n})}q^{O(1)})$	$\mathsf{DT}(n\log^2 n)$	$n^{O(\sqrt{n})}$	$n \cdot \deg(f)$
Direct	$\mathsf{DT}(n)$	$\mathsf{DT}(n)$	$\lceil \log_q n \rceil + 1$	$snq^2 \cdot \deg(f)$
Lower Bds	?	?	$\lceil \log_q n \rceil - 3$	$\frac{n\deg(f)}{e^2}(1\text{-}o(1)) - 2n^2/5$

2 Preliminaries

We denote a field by $\mathbb{F}$. For a prime power q, we denote the finite field with q elements as $\mathbb{F}_q$. A function $f : \mathbb{F}_q^n \to \mathbb{F}_q$ is equivalent to a polynomial over $\mathbb{F}_q$ and we will use $\deg(f)$ to denote the degree of the corresponding polynomial. To express the complexity of our reductions, we set up more notation. For any function $f : \mathbb{F}^n \to \mathbb{F}$: (i) $\mathrm{C}(f)$ denotes its time complexity; (ii) $\mathrm{C}_\varepsilon(f)$ denotes the time complexity of computing f correctly on all but an ε fraction of the inputs, where $0 < \varepsilon < 1$; (iii) Over a field $\mathbb{F}$, $\mathsf{DTIME}(t(n))$ denotes $O(t(n))$ deterministic operations over $\mathbb{F}$;[1] (iv) $\mathsf{RTIME}(t(n))$ likewise denotes $O(t(n))$ (Las Vegas) randomized operations over $\mathbb{F}$; while (v) for any $0 \leqslant \delta < 1/2$, $\mathsf{RTIME}_\delta(t(n))$ denotes randomized $\mathbb{F}_r$-operations when we allow for an error of δ.

When moving from a general function (over a finite field $\mathbb{F}$) to an equivalent symmetric function, the latter must be over an extension field of $\mathbb{F}$. We start with two elementary counting lower bounds on the degree of the extension field and on other parameters. The proofs are in the Appendix.

Theorem 1. *Let q be a prime power and $n \geqslant 1$ be an integer. If every $f : \mathbb{F}_q^n \to \mathbb{F}_q$ is equivalent to a symmetric function $f_{\text{sym}} : (\mathbb{F}_{q^s})^n \to \mathbb{F}_{q^s}$, then*

$$s \geqslant \lceil \log_q n \rceil - 3.$$

Recall that every function $f : \mathbb{F}_q^n \to \mathbb{F}_q$ is equivalent to a polynomial (over $\mathbb{F}_q$) in n variables and degree at most qn. Next, we will lower bound the degree blow-up necessary in assigning every function to an equivalent symmetric function.

Theorem 2. *If every function $f : \mathbb{F}_q^n \to \mathbb{F}_q$ of degree d is equivalent to some symmetric function $f_{\text{sym}} : \mathbb{F}_{q^s}^n \to \mathbb{F}_{q^s}$ of degree at most d_{sym} such that $s(d+n-1) \leqslant 2^{o(n)}$, then we must have*

$$d_{\text{sym}} \geqslant \frac{dn}{e^2} \cdot (1 - o(1)) - \frac{2}{5} \cdot n(n+2).$$

3 Results for the Elementary Symmetrization

We study the following particular substitution by the elementary symmetric polynomials:

$$f_{\text{sym}}(x) = f(e_1(x), e_2(x), \ldots, e_m(x)).$$

We first note that f_{sym} is almost as easy as f itself.

[1]We use parentheses $(\cdots)$ to emphasize that these time measures can be added in time expressions, whereas $\mathsf{DTIME}[t(n)]$ with $[\cdots]$ strictly denotes a class of functions.

Proposition 3. *For any function $f : \mathbb{F}_q^n \to \mathbb{F}_q$,*

$$\mathrm{C}(f_{\mathrm{sym}}) \leqslant \mathrm{C}(f) + \mathsf{DTIME}\left(n(\log n)^2\right).$$

***Proof*.** Note that the result will follow if we can show how to compute $e_1(\mathbf{a}), \ldots, e_n(\mathbf{a})$ in $\mathsf{DTIME}(n(\log n)^2)$. This follows from the well-known identity that

$$X^n + \sum_{i=1}^{n} e_i(\mathbf{a}) \cdot X^{i-1} = \prod_{j=1}^{n}(X - a_i),$$

since the polynomial on the RHS can be multiplied out via divide-and-conquer and FFT-based polynomial multiplication. ■

We now consider the problem of showing the converse of Proposition 3, i.e. we would like to bound the complexity of f in terms of the complexity of f_{sym}. We can show the following converses:

Theorem 4. *Let $f : \mathbb{F}_q^n \to \mathbb{F}_q$ be a function, then (for any $0 < \delta < 1/2$) the following are true:*

$$\begin{aligned}
\mathrm{C}(f) &\leqslant \mathrm{C}(f_{\mathrm{sym}}) + \mathsf{DTIME}(n^{O(\sqrt{n})} \cdot q^{O(1)}). && (1)\\
\mathrm{C}_{\exp(-\Omega(\sqrt{\log n}))}(f) &\leqslant \mathrm{C}(f_{\mathrm{sym}}) + \mathsf{RTIME}(n^{O(\log n)} \cdot \log^{O(1)} q), && (2)\\
\mathrm{C}_{\exp(-\Omega(\sqrt{\log n}))}(f) &\leqslant \mathrm{C}(f_{\mathrm{sym}}) + \mathsf{DTIME}(n^{O(\log n)} \cdot q^{O(1)}), && (3)\\
\mathrm{C}(f) &\leqslant \mathrm{C}(f_{\mathrm{sym}}) \cdot O(n \deg(f) \log(\frac{1}{\delta})) + \mathsf{RTIME}_{\delta}((n \log q)^{O(1)}) && (4)
\end{aligned}$$

provided $q \geqslant \Omega(n \deg(f) \log(\frac{1}{\delta}))$. Also:

$$\mathrm{C}(f) \leqslant O(\deg(f) \log(\frac{1}{\delta}) \cdot \mathrm{C}(f_{\mathrm{sym}})) + \mathsf{RTIME}_{\delta}(n^{O(\log n)} \cdot \log^{O(1)} q + q^{O(1)}), \quad (5)$$

provided $q \geqslant \Omega(\deg(f))$, and

$$\mathrm{C}(f) \leqslant O(q^x \cdot \log(\frac{1}{\delta}) \cdot \mathrm{C}(f_{\mathrm{sym}})) + \mathsf{RTIME}_{\delta}(n^{O(\log n)} \cdot \log^{O(1)} q + q^{O(1)}), \quad (6)$$

provided $1 \leqslant x \leqslant n$ and $\deg(f) \leqslant \min(x(q-1), O(q\sqrt{\log n}/\log q))$.

All of the results above start with the same basic idea, which we present next.

3.1 The Basic Idea

Note that we can prove the converse of Proposition 3 if for every $\mathbf{b} = (b_1, \ldots, b_n)$ for which we want to compute $f(\mathbf{b})$, we could compute $\mathbf{a} = (a_1, \ldots, a_n)$ such that for every $1 \leqslant i \leqslant n$, $b_i = e_i(\mathbf{a})$ and evaluate $f_{\mathrm{sym}}(\mathbf{a})$ (which by definition would be $f(\mathbf{b})$). In other words, given the polynomial

$$\phi_{\mathbf{b}}(X) = X^n + \sum_{i=1}^{n} b_i \cdot X^{i-1},$$

we want to completely factorize it into linear factors, i.e. compute $\mathbf{a} = (a_1, \ldots, a_n)$ such that

$$\prod_{i=1}^{n}(X - a_i) = \phi_{\mathbf{b}}(X).$$

It is not very hard to see that such an $\mathbf{a}$ might not exist in $\mathbb{F}_q^n$. Thus, we will have to look into extension fields of $\mathbb{F}_q$. In particular, the *splitting field* of any polynomial over $\mathbb{F}_q$ is the smallest extension field over which the polynomial factorizes completely into linear factors. The following result is well-known:

Proposition 5 (cf. [10]). *Let $h(X)$ be a univariate polynomial over $\mathbb{F}_q$ of degree k. Let s denote the least common multiple of all the distinct degrees of irreducible factors of $h(X)$ (over $\mathbb{F}_q$). Then the splitting field of $h(X)$ is $\mathbb{F}_{q^s}$.*

Given the above, the algorithm for computing $\mathbf{a}$ is direct:

```
Invert(b)

  1. Compute the polynomial $\phi_{\mathbf{b}}(X)$.
  2. Compute the smallest $s = s(\mathbf{b})$ such that $\mathbb{F}_{q^s}$splits $\phi_{\mathbf{b}}(X)$.
  3. Compute an irreducible polynomial of degree $s$ over $\mathbb{F}_q$.
  4. Compute the roots of $\phi_{\mathbf{b}}(X)$ over $\mathbb{F}_{q^s}$.
```

The correctness of the algorithm follows from the discussion above. (The computation of the irreducible polynomial in Step 3 is to form a representation of the finite field $\mathbb{F}_{q^s}$.) To analyze its running time, we define some notation: $\mathrm{split}(\mathbf{b}, q)$ denotes the time required to compute the smallest $s = s(\mathbf{b}, q)$ such that $\mathbb{F}_{q^s}$ is the splitting field of $\phi_{\mathbf{b}}(X)$, $\mathrm{irr}(k, q)$ is the time required to generate a degree k irreducible polynomial over $\mathbb{F}_q$ and $\mathrm{root}(k, q)$ is the time required compute all the roots of a degree k polynomial over $\mathbb{F}_q$.

The above implies the following:

Lemma 6. *For any $\mathbf{b} \in \mathbb{F}_q^n$, $\mathrm{Invert}(\mathbf{b})$ can be computed in time $\mathsf{DTIME}(n) + \mathrm{split}(\mathbf{b}, q) + \mathrm{irr}(s, q) + \mathrm{root}(n, q^s)$.*

This would imply that if $\mathrm{Invert}(\cdot)$ can be computed in time $T(n)$, then one has $\mathrm{C}(f) \leqslant \mathrm{C}(f_{\mathrm{sym}}) + T(n)$. Unfortunately, the quantity s in Step 2 above can be as large as $n^{O(\sqrt{n})}$. This implies that $T(n) = n^{O(\sqrt{n})}$, which leads to (1). To get a better time complexity, we will use the fact that for *random* $\mathbf{b} \in \mathbb{F}_q^n$, $s(\mathbf{b})$ is quasi-polynomial with high probability. This almost immediately implies (2) and (3). The rest of the claims follow from the usual testing algorithms for Reed-Muller codes (though in our case we only need to handle *erasures*).

Some known results that we use to prove Theorem 4 are given in the Appendix.

3.2 Proof of Theorem 4

We first bound the time complexity of the $\mathrm{Invert}(\cdot)$ algorithm. In particular, Lemma 6, Theorems 18, 20 and 21 show that $\mathrm{Invert}(\mathbf{b})$ can be computed in $\mathsf{DTIME}\left(n^{O(1)} \cdot s(\mathbf{b}, q)^{O(1)} \cdot q^{O(1)}\right)$. Also by using Theorems 17 and 19 instead of Theorem 18 and 20 respectively, one obtains that $\mathrm{Invert}(\mathbf{b})$ can be computed in $\mathsf{RTIME}\left(n^{O(1)} \cdot s(\mathbf{b}, q)^{O(1)} \cdot \log^{O(1)} q\right)$.

Lemma 13 along with the discussion above proves statement (1).

Corollary 16 says that for all but an $\exp(-\Omega(\sqrt{n}))$ fraction of $\mathbf{b} \in \mathbb{F}_q^n$, we have $s(\mathbf{b}, q) \leqslant 2^{\log^2 n} = n^{\log n}$. This along with the discussion above proves (2) and (3). (After Step 3 in $\mathrm{Invert}(\cdot)$ if $\log s > \log^2 n$ then we halt and output "fail.")

We now move to the proof of (4). Call $\mathbf{c} \in \mathbb{F}_q^n$ *good* if $s(\mathbf{c}, q) \leqslant n$. Note that by Step 2 of Invert($\mathbf{c}$), we will know if $\mathbf{c}$ is good or not. If it is good then we continue with the algorithm else we halt and output "fail." Recall that we want to compute $f(\mathbf{b})$. Note that if $\mathbf{b}$ is good then we can run Invert($\mathbf{b}$) in $\mathsf{DTIME}(n^{O(1)} \log^{O(1)} q)$ (and hence compute $f_{\text{sym}}(\text{Invert}(\mathbf{b})) = f(\mathbf{b})$). However, in the worst-case $\mathbf{b}$ need not be good. Thus, we do the following: we pick a random line through $\mathbf{b}$ and evaluate the function f restricted to the line on points other than $\mathbf{b}$.

In particular, consider the univariate polynomial $P_{\mathbf{b}}(X) = f(\mathbf{b} + \mathbf{m} \cdot X)$ for a uniformly random $\mathbf{m} \in \mathbb{F}_q^n$. Consider any subset $S \subseteq \mathbb{F}_q^*$ with $|S| = 4n(\deg(f) + 1)$. Now by Proposition 23 and Theorem 14 in expectation (over the choice of $\mathbf{m}$), at least $2n(\deg(f) + 1)$ points in the set $\{\mathbf{b} + \alpha \cdot \mathbf{m} | \mathbf{a} \in S\}$ are good.

We now move to the proof of (5). Call $\mathbf{c} \in \mathbb{F}_q^n$ *good* if $s(\mathbf{c}, q) \leqslant n^{\log n}$. (Otherwise call it *bad*.) Note that by Step 2 of Invert($\mathbf{c}$), we will know if $\mathbf{c}$ is good or bad. If it is good then we continue with the algorithm else we halt and output "fail." Recall that we want to compute $f(\mathbf{b})$. Note that if $\mathbf{b}$ is good then we can run Invert($\mathbf{b}$) in $\mathsf{RTIME}(n^{O(\log n)} \log^{O(1)} q)$ (and hence compute $f_{\text{sym}}(\text{Invert}(\mathbf{b})) = f(\mathbf{b})$). However, in the worst-case $\mathbf{b}$ might be bad. Thus, we do the following: we pick a random line through $\mathbf{b}$ and evaluate the function f restricted to the line on points other than $\mathbf{b}$.

In particular, consider the univariate polynomial $P_{\mathbf{b}}(X) = f(\mathbf{b} + \mathbf{m} \cdot X)$ for a uniformly random $\mathbf{m} \in \mathbb{F}_q^n$. Consider any subset $S \subseteq \mathbb{F}_q^*$ with $|S| = 3(\deg(f) + 1)$. (Note that we will need $q - 1 \geqslant 3(\deg(f) + 1)$, which will be satisfied by the condition on $q \geqslant \Omega(\deg(f))$. Now by Theorem 14 in expectation (over the choice of $\mathbf{m}$), at most $\exp(-\Omega(\sqrt{\log n}) \cdot 3(\deg(f) + 1) \leqslant (\deg(f) + 1)$ points in the set $\{\mathbf{b} + \alpha \cdot \mathbf{m} | \alpha \in S\}$ are bad. (The inequality follows for large enough n.) Thus, by Markov's inequality, with probability at least $1/2$ (over the choice of $\mathbf{m}$) there are at most $2(\deg(f) + 1)$ bad points in $\{\mathbf{b} + \alpha\mathbf{m} | \alpha \in S\}$. (Recall that we know when a point is bad, and thus we can recognize when we have at most $2(\deg(f) + 1)$ bad points.) In other words, our algorithm will compute $P_{\mathbf{b}}(\mathbf{b} + \alpha\mathbf{m})$ correctly for at least $\deg(f) + 1$ points $\alpha \in S$. Proposition 23 then implies that we can compute $P_{\mathbf{b}}(X)$ in $\mathsf{DTIME}(\deg(f)^3) \in \mathsf{DTIME}(q^3)$ by our assumption on q. Note that we can now read off $f(\mathbf{b}) = P_{\mathbf{b}}(0)$. Note that the procedure above has an error probability of at most $1/2$. We can reduce this to δ by running $O(\log(1/\delta))$ independent runs of this procedure and stop whenever we compute $f(\mathbf{b})$.

The proof of (6) is a generalization of the proof for (5) to the multivariate case. In particular, given an integer $1 \leqslant x \leqslant n$, we pick a random subspace of $\mathbb{F}_q^n$ of dimension x by picking x basis vectors (say $\mathbf{e}_1, \ldots \mathbf{e}_x$) at random. Now consider the set $S = \{\mathbf{b} + \sum_{j=1}^{x} \alpha_j \cdot \mathbf{e}_j | (\alpha_1, \ldots, \alpha_x) \in \mathbb{F}_q^x\}$. It is easy to see that the function f restricted to S is an x-variate polynomial with the degree at most $\deg(f)$ (here the scalars $\alpha_1, \ldots, \alpha_x$ are thought of as the variables). Thus, by Corollary 25, we can recover $f(\mathbf{b})$ if at most $q^{-\deg(f)/q}$ fraction of the points in S are bad. This happens with probability at least a $1/2$ (by Markov's argument and Theorem 14) if $\exp(-\Omega(\sqrt{\log n}))$ is at most $q^{-\deg(f)/q}/2$. This inequality is implied by the assumption that $\deg(f) \leqslant O(q\sqrt{\log n}/q)$. Also we need $\deg(f) \leqslant (q-1)x$ to ensure that we can apply Corollary 25 to f projected down to S. Finally, we run the procedure above independently $O(\log(1/\delta))$ times to bring the error probability down to δ.

4 Results for the Second Symmetrization

4.1 Functions over Finite Fields

We state our main result for functions defined over finite fields:

Theorem 7. *Let $n \geqslant 1$ be an integer and q be a prime power. Define $s = 1 + \lceil \log n \rceil$. Then for every function $f : \mathbb{F}_q^n \to \mathbb{F}_q$, there exists a symmetric function $f_{\mathrm{sym}} : \mathbb{F}_{q^s}^n \to \mathbb{F}_q$ such that*

$$C(f_{\mathrm{sym}}) \leqslant C(f) + \mathsf{DTIME}(n), \tag{7}$$

$$C(f) \leqslant C(f_{\mathrm{sym}}) + \mathsf{DTIME}(n). \tag{8}$$

Further, $\deg(f_{\mathrm{sym}}) \leqslant snq^2 \cdot \deg(f)$.

In the rest of the section, we will prove the theorem above. Before we describe f_{sym}, we first set up some simple notation (and assumptions). We will assume that we have access to an irreducible polynomial of degree s over $\mathbb{F}_q$.[2] In particular, we will assume that every element $\alpha \in \mathbb{F}_{q^s}$ is represented as $\sum_{\ell=0}^{s-1} \alpha_\ell \cdot \gamma^\ell$ for some root $\gamma \in \mathbb{F}_{q^s}$ of the irreducible polynomial. Further, we will assume that $[n]$ is embedded into $\mathbb{F}_q^{s-1}$. (Note that by definition of s, $q^{s-1} \geqslant n$.) From now on we will think of $i \in [n]$ interchangeably as an integer in $[n]$ and an element in $\mathbb{F}_q^{s-1}$. We first claim the existence of certain polynomials.

Lemma 8. *There exist s-many explicit univariate polynomials $\pi_k : \mathbb{F}_{q^s} \to \mathbb{F}_q$ $(0 \leqslant k \leqslant s-1)$ such that for any $\alpha = \alpha_{s-1} \cdot \gamma^{s-1} + \cdots + \alpha_0 \in \mathbb{F}_{q^s}$, $\pi_k(\alpha) = \alpha_k$. Further, $\deg(\pi_k) = q^{s-1}$.*

Proof. For any $\alpha = \sum_{i=0}^{s-1} \alpha_i \gamma^i \in \mathbb{F}_{q^s}$, $\alpha^{q^j} = \sum_{i=0}^{s-1} \alpha_i (\gamma^i)^{q^j}$ for every $0 \leqslant j \leqslant s-1$. Thus, we have

$$(\alpha \;\; \alpha^q \;\; \alpha^{q^2} \;\; \cdots \;\; \alpha^{q^{s-1}})^T = V \cdot (\alpha_0 \;\; \alpha_1 \;\; \alpha_2 \;\; \cdots \;\; \alpha_{s-1})^T,$$

where V is the Vandermonde matrix with the ℓth row being the first s powers of γ^{q^ℓ} (starting from the 0th power)—note that all the elements γ^{q^ℓ} are distinct. Thus, we have that α_k is the inner product of the kth row of the inverse of the Vandermonde matrix and $(\alpha \;\; \alpha^q \;\; \alpha^{q^2} \;\; \cdots \;\; \alpha^{q^{s-1}})$. The definition for $\pi_k(X)$ then follows from the (known) expressions for entries of the inverse of the Vandermonde matrix (cf. [17]). ■

Lemma 9. *Fix $j \in [n]$. There exists an explicit n-variate* symmetric *polynomial $\phi_j(X_1, \ldots, X_n) : (\mathbb{F}_{q^s})^n \to \mathbb{F}_q$ of degree at most sq^s such that for any choice of $\alpha^i = \alpha^i_{s-1} \cdot \gamma^{s-1} + \cdots + \alpha^i_1 \cdot \gamma + \alpha^i_0 \in \mathbb{F}_{q^s}$ $(1 \leqslant i \leqslant n)$,*

$$\phi_j(\alpha^1, \alpha^2, \ldots, \alpha^n) = \sum_{i \in [n], (\alpha^i_{s-1}, \alpha^i_{s-2}, \cdots, \alpha^i_1) = j} \alpha^i_0, \tag{9}$$

where we consider $(\alpha^i_{s-1}, \alpha^i_{s-2}, \cdots, \alpha^i_1) \in \mathbb{F}_q^{s-1}$.

Proof. For any $j = (j_1, \ldots, j_{s-1}) \in \mathbb{F}_q^{s-1}$, consider the degree $(s-1)(q-1)$ polynomial $A_j(Y_1, \ldots, Y_{s-1})$ over $\mathbb{F}_q$:

$$A_j(Y_1, \ldots, Y_{s-1}) = \prod_{\ell=1}^{s-1} \prod_{\beta \in \mathbb{F}_q, \beta \neq j_\ell} \left(\frac{Y_\ell - \beta}{j_\ell - \beta} \right).$$

Note that $A_j(i) = 1$ if $i = j$ else $A_j(i) = 0$ for every $i \in \mathbb{F}_q^{s-1}$. Now consider the polynomial

$$\phi_j(X_1, \ldots, X_n) = \sum_{i=1}^{n} A_j(\pi_1(X_i), \ldots, \pi_{s-1}(X_i)) \cdot \pi_0(X_i).$$

[2]Otherwise in the reduction we can compute one in time $\mathsf{DTIME}(s^{O(1)} \cdot q^{O(1)})$.

Now by the properties of $A_j(\cdot)$ mentioned above, in the RHS of the equation above for $\phi_j(\alpha^1, \dots, \alpha^n)$, the only summands that will contribute are those i for which $(\alpha^i_{s-1}, \alpha^i_{s-2}, \cdots, \alpha^i_1) = j$ (this follows from Lemma 8). Further each such summand contributes α^i_0 to the sum (by Lemma 8). This proves (9). Further, it follows from definition that ϕ_j is a symmetric polynomial. Finally, note that for every $i \in [n]$, $A_j(\pi_1(X_i), \dots, \pi_{s-1}(X_i))$ has degree $(s-1)(q-1) \cdot q^{s-1}$. Further, as π_0 has degree q^{s-1}, $\deg(\phi_j) = q^{s-1}(s-1)(q-1) + q^{s-1} \leqslant sq^s$. ■

We are now ready to define $f_{\text{sym}} : \mathbb{F}_{q^s} \to \mathbb{F}_q$. For any $\mathbf{a} = (a_1, \dots, a_n) \in \mathbb{F}^n_{q^s}$, define

$$f_{\text{sym}}(\mathbf{a}) = f\left(\phi_1(\mathbf{a}), \phi_2(\mathbf{a}), \dots, \phi_n(\mathbf{a})\right).$$

Since each of $\phi_1, \dots, \phi_n$ are symmetric, so is f_{sym}. Further, as each of $\phi_1, \dots, \phi_n$ has degree at most sq^s, $\deg(f_{\text{sym}}) \leqslant sq^s \cdot \deg(f) \leqslant snq^2 \cdot \deg(f)$, where the last inequality follows from the fact that our choice of s implies $q^{s-1} \leqslant nq$.

In what follows, we will assume that any $\alpha \in \mathbb{F}_{q^s}$ is presented to us as $(\alpha_{s-1}, \dots, \alpha_1, \alpha_0)$, where $\alpha = \alpha_{s-1}\gamma^{s-1} + \cdots + \alpha_1\gamma + \alpha_0$. Note that this implies that $\pi_k(\alpha)$ (for $0 \leqslant k \leqslant s-1$) can be "computed" in constant time.[3]

We first prove (7). Given $\mathbf{b} \in \mathbb{F}^n_{q^s}$, we will compute $\mathbf{a} \in \mathbb{F}^n_q$ in $\mathsf{DTIME}(n)$ such that $f_{\text{sym}}(\mathbf{b}) = f(\mathbf{a})$. Note that this suffices to prove (7). Notice that by definition of f_{sym}, this is satisfied if $a_j = \phi_j(\mathbf{b})$ for $j \in [n]$. Further note that given $\beta_1, \dots, \beta_{s-1} \in \mathbb{F}_q$, one can compute $A_j(\beta_1, \dots, \beta_{s-1})$ in $\mathsf{DTIME}(s)$.[4] This along with the assumption that $\pi_k(\alpha)$ can be computed in constant time for any $\alpha \in \mathbb{F}_{q^s}$, implies that $\phi_j(\mathbf{b})$ can be computed in $\mathsf{DTIME}(ns)$. This would immediately imply a total of $\mathsf{DTIME}(n^2 s)$ for computing $\mathbf{a}$. However, note that as all the a_j values are sums, we can compute $\mathbf{a}$ in one pass over $\mathbf{b}$ with space $O(n)$. Thus, we can compute $\mathbf{a}$ from $\mathbf{b}$ in $\mathsf{DTIME}(n)$, as desired.

Finally, we prove (8). We will show that given $\mathbf{a} \in \mathbb{F}^n_q$ we can compute $\mathbf{b} \in \mathbb{F}^n_{q^s}$ in $\mathsf{DTIME}(n)$ such that $f(\mathbf{a}) = f_{\text{sym}}(\mathbf{b})$. Note that this will prove (8). Further, notice that we will be done if we can show that for $j \in [n]$, $a_j = \phi_j(\mathbf{b})$. The definition actually is pretty easy: for every $i = (i_{s-1}, \dots, i_1) \in [n]$, define $b_i = i_{s-1}\gamma^{s-1} + \cdots + i_1\gamma + a_i$. Now it can be checked that $\phi_j(\mathbf{b}) = \pi_0(b_j) = a_j$, as desired. Further, it is easy to check that one can compute $\mathbf{b}$ in $\mathsf{DTIME}(n)$.

4.2 Functions over Reals

We state our main result for $\mathbb{F} = \mathbb{R}$:

Theorem 10. *Let $n \geqslant 1$ be an integer. Then for every function $f : \mathbb{R}^n \to \mathbb{R}$, there exists a symmetric function $f_{\text{sym}} : \mathbb{R}^n \to \mathbb{R}$ such that*

$$C(f_{\text{sym}}) \leqslant C(f) + \mathsf{DTIME}(n), \tag{10}$$

$$C(f) \leqslant C(f_{\text{sym}}) + \mathsf{DTIME}(n). \tag{11}$$

The proof is similar to the one for finite fields, so we only sketch the differences here. Towards this end, we think of every element $x \in \mathbb{R}$ as a triple $(\lfloor \lfloor x \rfloor / n \rfloor, \lfloor x \rfloor \mod n, x - \lfloor x \rfloor) \in \mathbb{Z} \times \mathbb{Z} \times [0, 1)$. In particular, for any $1 \leqslant j \leqslant n$, we define

$$\phi_j((u_1, v_1, w_1), (u_2, v_2, w_2), \dots, (u_n, v_n, w_n)) = \sum_{i=1}^{n} \delta_{v_i, j} \cdot (u_i + w_i),$$

[3]If not, by the proof of Lemma 8, one can compute both values in $O(s^2 \log q)$ operations over $\mathbb{F}_{q^s}$.

[4]Here we are also assuming that the map from $[n]$ to the corresponding element in $\mathbb{F}^{s-1}_q$ can be computed in constant time.

where $\delta_{\ell,k} = 1$ if $\ell = k$ and is zero otherwise. f_{sym} is defined as before and the reduction from f_{sym} to f it also as before. The reduction from f to f_{sym} is also pretty much the same as before except we define $b_i = i + n \cdot \lfloor a_i \rfloor + a_i - \lfloor a_i \rfloor$.

5 Conclusions

We have given two efficient ways to reduce a general polynomial r to a symmetric polynomial s. If r is over a finite field $\mathbb{F}$, then s will be over a larger finite field $\mathbb{F}'$, but our results show that the increases in degree, field size, and running time are reasonable.

This sheds new light on the question of how usefully the special structure of symmetric polynomials can be leveraged for complexity lower bounds. Among questions for further research, we are interested in aspects of how universality properties of functions such as the determinant polynomials are preserved under our symmetrizations.

References

[1] Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and hardness of approximation problems. J. Assn. Comp. Mach. 45, 501–555 (1998)

[2] Arora, S., Safra, S.: Probabilistic checking of proofs: a new characterization of NP. J. Assn. Comp. Mach. 45, 70–122 (1998)

[3] Arora, S., Sudan, M.: Improved low-degree testing and its applications. Combinatorica 23(3), 365–426 (2003)

[4] Assumus Jr., E.F., Key, J.D.: Polynomial codes and Finite Geometries in Handbook of Coding Theory, Vol II , Edited by V. S. Pless Jr., and W. C. Huffman, chap. 16. Elsevier (1998)

[5] Baur, W., Strassen, V.: The complexity of partial derivatives. Theor. Comp. Sci. 22, 317–330 (1982)

[6] Beame, P., Brisson, E., Ladner, R.: The complexity of computing symmetric functions using threshold circuits. Theor. Comp. Sci. 100, 253–265 (1992)

[7] Beigel, R., Tarui, J.: On ACC. In: Proc. 32nd Annual IEEE Symposium on Foundations of Computer Science. pp. 783–792 (1991)

[8] Ben-Or, M.: Lower bounds for algebraic computation trees. In: Proc. 15th Annual ACM Symposium on the Theory of Computing. pp. 80–86 (1983)

[9] Berlekamp, E.: Factoring polynomials over large finite fields. Mathematics of Computation 24, 713–735 (1970)

[10] Dixon, J.D., Panario, D.: The dgree of the splitting field of a random polynomial over a finite field. The Electronic Journal of Combinatorics 11(1) (2004), `http://www.combinatorics.org/Volume11/Abstracts/v11i1r70.html`

[11] Furst, M., Saxe, J., Sipser, M.: Parity, circuits, and the polynomial-time hierarchy. Math. Sys. Thy. 17, 13–27 (1984)

[12] Grigoriev, D., Karpinski, M.: An exponential lower bound for depth 3 arithmetic circuits. In: Proc. 30th Annual ACM Symposium on the Theory of Computing. pp. 577–582 (1998)

[13] Grigoriev, D., Razborov, A.: Exponential lower bounds for depth 3 algebraic circuits in algebras of functions over finite fields. Applicable Algebra in Engineering, Communication, and Computing 10, 465–487 (2000), (preliminary version FOCS 1998)

[14] Grolmusz, V.: Computing elementary symmetric polynomials with a sub-polynomial number of multiplications. SIAM Journal on Computing 32, 2002–02 (2002)

[15] Jutla, C.S., Patthak, A.C., Rudra, A., Zuckerman, D.: Testing low-degree polynomials over prime fields. Random Struct. Algorithms 35(2), 163–193 (2009)

[16] Kaufman, T., Ron, D.: Testing polynomials over general fields. SIAM Journal on Computing 36(3), 779–802 (2006)

[17] Klinger, A.: The Vandermonde matrix. The American Mathematical Monthly 74(5), 571–574 (1967)

[18] Lu, C.J.: An exact characterization of symmetric functions in $q\mathrm{AC}^0[2]$. In: Proc. 4th International Combinatorics and Computing Conference. pp. 167–173 (1998)

[19] Parvaresh, F., Vardy, A.: Correcting errors beyond the Guruswami-Sudan radius in polynomial time. In: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS). pp. 285–294 (2005)

[20] Razborov, A.: Lower bounds for the size of circuits of bounded depth with basis $\{\wedge, \oplus\}$. Mathematical Notes, (formerly of the Academy of Natural Sciences of the USSR) 41, 333–338 (1987)

[21] Razborov, A.: On the method of approximations. In: Proc. 21st Annual ACM Symposium on the Theory of Computing. pp. 167–176 (1989)

[22] Shoup, V.: New algorithms for finding irreducible polynomials over finite fields. Mathematics of Computation 54, 435–447 (1990)

[23] Shoup, V.: A computational introduction to number theory and algebra. Cambridge University Press, New York, NY, USA (2008)

[24] Shpilka, A., Wigderson, A.: Depth-3 arithmetic formulae over fields of characteristic zero. Computational Complexity 10, 1–27 (2001)

[25] Smolensky, R.: Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In: Proc. 19th Annual ACM Symposium on the Theory of Computing. pp. 77–82 (1987)

A Appendix—Omitted Proofs from Section 2

A.1 Proof of Theorem 1

For notational simplicity, define $r = q^s$.

Note that every symmetric function from $(\mathbb{F}_r)^n$ to $\mathbb{F}_r$ is determined by a table giving the value for every r-tuple of nonnegative integers summing to n, which represent the count of variables set equal to the respective element of $\mathbb{F}_r$. If T denotes the number of entries in the table above, there are r^T symmetric functions. Now notice that T is exactly the number of ways one can split $[n]$ into r disjoint subsets (some of which can be empty). It is well-known that

$$T = \binom{r+n-1}{n}.$$

On the other hand there are q^{q^n} functions from $\mathbb{F}_q^n$ to $\mathbb{F}_q$ overall.

This enables us to give a lower bound on the degree of the extension field $\mathbb{F}_r$ over $\mathbb{F}_q$. To make sure that any function over $\mathbb{F}_q$ in n variables is equivalent to a symmetric polynomial over $\mathbb{F}_r$ in n variables, by a simple counting argument, we need $r^T \geqslant q^{q^n}$:

$$(q^s)^{\binom{q^s+n-1}{n}} \geqslant q^{q^n}.$$

This translates to

$$s \cdot \binom{q^s+n-1}{n} \geqslant q^n,$$

which we claim implies $q^s > n/8$. For the sake of contradiction assume that $q^s \leqslant n/8$. Then note that

$$s \cdot \binom{q^s + n - 1}{n} \leqslant \log_q n \cdot \binom{9n/8}{n} \leqslant \log_q n \cdot (9e)^{n/8} \leqslant \log n \cdot (\sqrt[8]{9e})^n < 2^n \leqslant q^n,$$

which is a contradiction (note that we have used $q \geqslant 2$).

A.2 Proof of Theorem 2

Towards this end given a vector $\mathbf{a} = (a_1, \ldots, a_n)$ where $a_i \geqslant 1$ are integers, define the set

$$\mathcal{M}_{n,\mathbf{a},d} = \left\{ (i_1, \ldots, i_n) | \sum_{j=1}^{n} a_j \cdot i_j \leqslant d; \quad i_j \in \mathbb{Z}_{\geqslant 0} \right\}.$$

Note that $\mathcal{M}_{v,\mathbf{1},d}$ denotes the number of monomials of degree at most d and thus, the number of functions f with $\deg(f) \leqslant d$ is exactly $q^{|\mathcal{M}_{n,\mathbf{1},d}|}$, where $\mathbf{1} = (1, 1, \ldots, 1)$. On the other hand, recall that any symmetric polynomial on m variables can be thought of a polynomial in the m elementary symmetric polynomials. Thus, the number of symmetric polynomial f_{sym} with $\deg(f_{\text{sym}}) \leqslant d_{\text{sym}}$ is exactly $(q^s)^{|\mathcal{M}_{m,[\mathbf{m}],d_{\text{sym}}}|}$, where $[\mathbf{m}] = (1, 2, \ldots, m)$. Note that if all degree at most d polynomials were equivalent to at most degree d_{sym} symmetric polynomials, we must have

$$s \cdot \left| \mathcal{M}_{m,[\mathbf{m}],d_{\text{sym}}} \right| \geqslant \left| \mathcal{M}_{n,\mathbf{1},d} \right|. \tag{12}$$

Towards this end, we first bound the size of $\mathcal{M}$.

A.2.1 Bounding $|\mathcal{M}|$

We bound $|\mathcal{M}_{v,\mathbf{a},D}|$ by a similar argument used in [19] to bound for the special case of $\mathbf{a} = (1, k, \ldots, k)$.

Consider the following simplex:

$$\mathcal{S}_{n,\mathbf{a},D'} = \left\{ (i_1, \ldots, i_n) | \sum_{j=1}^{n} a_j \cdot i_j \leqslant D'; \quad i_j \geqslant \mathbb{R}_{\geqslant 0} \right\}.$$

We will use $|\mathcal{S}|$ to denote the volume of the simplex $\mathcal{S}$.

We begin with the following claim:

Claim 1. *For all integers $n, D \geqslant 1$ and vector $\mathbf{a} \in \mathbb{Z}^n$ with $a_i \geqslant 1$ for $i \in [n]$, we have*

$$|\mathcal{S}_{n,\mathbf{a},D}| \leqslant |\mathcal{M}_{n,\mathbf{a},D}| \leqslant \left| \mathcal{S}_{n,\mathbf{a},D+\sum_{i=1}^{n} a_i} \right|.$$

***Proof*.** For notational simplicity define $\mathcal{M} = \mathcal{M}_{n,\mathbf{a},D}$ and $\mathcal{S}_\ell = \mathcal{S}_{n,\mathbf{a},D+\ell}$.

Consider the map ϕ that maps every element $(i_1, \ldots, i_n) \in \mathcal{M}$ to the cube in $\mathbb{R}^n$ with sides of length 1 and $(i_1, \ldots, i_n)$ as it's lowest corner point, i.e.

$$\phi(i_1, \ldots, i_n) = [i_1, i_1 + 1) \times [i_2, i_2 + 1) \times \cdots \times [i_n, i_n + 1).$$

Note that the volume of $\phi(i_1, \ldots, i_n) = 1$ (we'll denote this volume as $|\phi(i_1, \ldots, i_n)|$). Note that for every $(i_1, \ldots, i_n) \in \mathcal{M}$, $\phi(i_1, \ldots, i_n) \subseteq \mathcal{S}_{\sum_{i=1}^{n} a_i}$ and $(i_1, \ldots, i_n) \in \mathcal{S}_0$. These imply that

$$\mathcal{S}_0 \subseteq \cup_{(i_1,\ldots,i_n) \in \mathcal{M}} \phi(i_1, \ldots, i_n) \subseteq \mathcal{S}_{\sum_{i=1}^{n} a_i},$$

which proves the claim. ■

Next, we bound the volume of a simplex:

Lemma 11. *For all integers $n, D \geqslant 1$ and vector $\mathbf{a} \in \mathbb{Z}^n$ with $a_i \geqslant 1$ for $i \in [n]$, we have*

$$|\mathcal{S}_{n,\mathbf{a},D}| = \frac{D^n}{n! \prod_{i=1}^n a_i}.$$

***Proof*.** We will prove the lemma by induction on n. Note that when $n = 1$, the volume of the simplex is D/a_1, which proves the base case. We assume that the equality holds up to $n-1$. Note that

$$|\mathcal{S}_{n,\mathbf{a},D}| = \int_0^{D/a_n} \frac{(D - a_n \cdot i_n)^{n-1}}{(n-1)! \prod_{i=1}^{n-1} a_i} \, di_n$$

where the equality follows from the inductive hypothesis. Computing the integral completes the proof. ■

Claim 1 and Lemma 11 implies the following:

Corollary 12. *For all integers $n, D \geqslant 1$ and vector $\mathbf{a} \in \mathbb{Z}^n$ with $a_i \geqslant 1$ for $i \in [n]$, we have*

$$\frac{D^n}{n! \prod_{i=1}^n a_i} \leqslant |\mathcal{M}_{n,\mathbf{a},D}| \leqslant \frac{(D + \sum_{i=1}^n a_i)^n}{n! \prod_{i=1}^n a_i}.$$

A.2.2 Computing the lower bound for $m = n$

Note that we have

$$|\mathcal{M}_{n,\mathbf{1},D}| \geqslant \binom{n+D-1}{D} \geqslant \left(\frac{D+n-1}{n-1}\right)^{n-1}.$$

The lower bound along with the upper bound in Corollary 12 with (12) implies that for the reduction to go through, we need

$$s \cdot \frac{(d_{\text{sym}} + m(m+1)/2)^m}{(m!)^2} \geqslant \left(\frac{d+n-1}{n-1}\right)^{n-1}.$$

For now, we will only consider the case $m = n$, which implies that we need

$$(d_{\text{sym}} + n(n+1)/2)^n \geqslant \frac{1}{s} \cdot \frac{(n!)^2}{(n-1)^{n-1}} \cdot (d+n-1)^{n-1} \geqslant \frac{1}{s(d+n-1)} \cdot \left(\frac{(d+n-1)n}{e^2}\right)^n,$$

which in turn implies that we need

$$d_{\text{sym}} \geqslant d \cdot \frac{n}{e^2 \cdot \sqrt[n]{s(d+n-1)}} - \left(\frac{1}{2} - \frac{1}{e^2}\right) \cdot n(n+2).$$

The claimed bound follows from the assumption that $s(d+n-1) \leqslant 2^{o(n)}$.

B Appendix—Known Results Used in Section 3

We begin with known results on the degree of the splitting field of $\phi_{\mathbf{b}}(X)$ for $\mathbf{b}$ chosen uniformly at random from $\mathbb{F}_q^n$.

Lemma 13. *For every $\mathbf{b} \in \mathbb{F}_q^n$,*

$$s(\mathbf{b}, q) \leqslant n^{O(\sqrt{n})}.$$

***Proof*.** Recall that the degree of the splitting field is the least common multiple of all the distinct degree of the irreducible factors of the polynomial. Thus, we want to bound the maximum least common multiple of distinct positive integers $i_1, \ldots, i_m$ such that $i_1, \ldots, i_m = n$. We consider the contribution of $i_j \leqslant \sqrt{n}$ and $i_j > \sqrt{n}$ separately. Note that integers smaller than $\sqrt{n}$ can contribute at most $(\sqrt{n})!$ to the l.c.m. Further, note that there can be at most $\sqrt{n}$ integers larger than $\sqrt{n}$, which by themselves can contribute at most $n^{\sqrt{n}}$. Thus, we have

$$s \leqslant (\sqrt{n})! \cdot n^{\sqrt{n}} \leqslant \sqrt{n}^{\sqrt{n}} \cdot n^{\sqrt{n}} \leqslant n^{3\sqrt{n}/2},$$

which completes the proof. ■

Theorem 14 (Folklore, cf. [23])**.**

$$\Pr_{\mathbf{b} \in \mathbb{F}_q^n} [s(\mathbf{b}, q) = n] \geqslant \frac{1}{2n}.$$

Theorem 15 ([10])**.** *For every $x \geqslant 1$ and an absolute constant $c_0 > 0$,*

$$\Pr_{\mathbf{b} \in \mathbb{F}_q^n} \left[\left| \log(s(\mathbf{b}, q)) - \frac{1}{2} \cdot \log^2 n \right| > \frac{x}{\sqrt{3}} \cdot \sqrt[2]{\log^3 n} \right] \leqslant c_0 \exp\left(-\frac{x}{4}\right).$$

The above implies the following corollary

Corollary 16.

$$\Pr_{\mathbf{b} \in \mathbb{F}_q^n} \left[\log(s(\mathbf{b}, q)) > \log^2 n\right] \leqslant \exp\left(-\Omega\left(\sqrt{\log n}\right)\right).$$

Next we recall the known results about the time complexity of basic finite field algorithms to implement the last three steps in Invert$(\cdot)$.

Theorem 17 ([23])**.** *For any prime power q and integer $k \geqslant 1$,*

$$\mathrm{irr}(k, q) \in \mathsf{RTIME}\left(k^4 \log q\right).$$

Theorem 18 ([22])**.** *For any prime power q and integers $k, t \geqslant 1$,*

$$\mathrm{irr}(k, q^t) \in \mathsf{DTIME}\left(k^5 t^3 \sqrt{q}\right).$$

Since factorizing a polynomial gives all its roots, we have

Theorem 19 ([23])**.** *For any prime power q and integer $k \geqslant 1$,*

$$\mathrm{root}(k, q) \in \mathsf{RTIME}\left(k^3 \log q\right).$$

Theorem 20 ([9])**.** *For any prime power q and integers $k, t \geqslant 1$,*

$$\mathrm{root}(k, q^t) \in \mathsf{DTIME}\left(k^{O(1)} q^{O(1)} t^{O(1)}\right).$$

Next, we consider the problem of determining the degree of the splitting field of $\phi_{\mathbf{b}}(X)$. Note that this problem is solved by the distinct factorization problem, where the goal is to compute all the distinct degrees of the irreducible factors (which is enough to compute the degree of the splitting field) as well as the product of all irreducible factors of given degree. This problem can be solved in $O(k^3 \log q)$ deterministic time when the input is a degree k square-free polynomial [23]. Converting a general degree k polynomial into a square-free one in turn takes $O(k^2 + k \log q)$ deterministic time [23]. This in turn implies the following:

Theorem 21. *For any prime power q, integer $n \geqslant 1$ and $\mathbf{b} \in \mathbb{F}_q^n$*

$$\mathrm{split}(\mathbf{b}, q) \in \mathsf{DTIME}(n^3 \log q).$$

Finally, we recall some basic results from coding theory.

Proposition 22. *Any code with distance d can tolerate $d-1$ erasures. Further, for a linear code with block length n and distance d,one can decode from at most $d-1$ erasures in $\mathsf{DTIME}(n^3)$.*

The well-known Reed-Solomon code, formed by evaluating univariate polynomials of degree k is a well-known linear codes with distance $n-k$. This implies the following:

Proposition 23. *Given the points $(\alpha_i, y_i))$ for $1 \leqslant i \leqslant k+1$ for distinct a_i's, one can compute the unique degree k polynomial $P(X)$ such that $P(\alpha_i) = y_i$ for every i in $\mathsf{DTIME}(k^3)$.*

In fact, the above can be easily done in $\mathsf{DTIME}(k^2)$ as the unique polynomial $P(X)$ is given by Newton's interpolation formula:

$$P(X) = \sum_{i=1}^{k+1} \frac{\prod_{j \in [k+1], j \neq i} (X - \alpha_j)}{\prod_{j \in [k+1], j \neq i} (\alpha_i - \alpha_j)} \cdot y_i.$$

and thus, can be computed in $\mathsf{DTIME}(n^2)$.

The generalization of Reed-Solomon codes to multivariate polynomial, known as Reed-Muller codes, are also linear-codes with well know distance properties. In particular,

Lemma 24 ([4])**.** *The Reed-Muller code defined by evaluating n-variate degree $k \leqslant (q-1)n$ polynomial over $\mathbb{F}_q^n$ has distance $(R+1)q^Q$, where Q and R are the quotient and reminder from dividing $(q-1)n-k$ by $q-1$.*

Note that the above implies that the Reed-Muller code has distance at least $q^{n-\lceil k/(q-1) \rceil}$, which along with Proposition 22 implies that

Corollary 25. *Given the evaluations of an n-variate degree k polynomial on all but $q^{-k/q}$ fraction of the points in $\mathbb{F}_q^n$, the polynomial can be uniquely determined in $\mathsf{DTIME}(n^3)$.*

We have now set up everything referenced in the proof of Theorem 4.