Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 3 (2014)

Testing Equivalence of Polynomials under Shifts

Zeev Dvir* Rafael Oliveira† Amir Shpilka‡

Abstract

Two polynomials $f, g \in \mathbb{F}[x_1, \ldots, x_n]$ are called *shift-equivalent* if there exists a vector $(a_1, \ldots, a_n) \in \mathbb{F}^n$ such that the polynomial identity $f(x_1 + a_1, \ldots, x_n + a_n) \equiv g(x_1, \ldots, x_n)$ holds. Our main result is a new randomized algorithm that tests whether two given polynomials are shift equivalent. Our algorithm runs in time polynomial in the circuit size of the polynomials, to which it is given black box access. This complements a previous work of Grigoriev [Gri97] who gave a deterministic algorithm running in time $n^{O(d)}$ for degree d polynomials.

Our algorithm uses randomness only to solve instances of the Polynomial Identity Testing (PIT) problem. Hence, if one could de-randomize PIT (a long-standing open problem in complexity) a de-randomization of our algorithm would follow. This establishes an equivalence between de-randomizing shift-equivalence testing and de-randomizing PIT (both in the black-box and the white-box setting). For certain restricted models, such as Read Once Branching Programs, we already obtain a deterministic algorithm using existing PIT results.

*Department of Computer Science and Department of Mathematics, Princeton University. Email: zeev.dvir@gmail.com. Research supported by NSF grants CCF-1217416 and CCF-0832797.

†Department of Computer Science, Princeton University. Email: rmo@cs.princeton.edu.

‡Department of Computer Science, Technion — Israel Institute of Technology, Haifa, Israel, shpilka@cs.technion.ac.il. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 257575.

ISSN 1433-8092

1 Introduction

In this paper we address the following problem, which we call *Shift Equivalence Testing* (SET). Given two polynomials $f, g \in \mathbb{F}[\mathbf{x}]$ (we use boldface letters to denote vectors), decide whether there exists a shift $\mathbf{a} \in \mathbb{F}^n$ such that $f(\mathbf{x}+\mathbf{a}) \equiv g(\mathbf{x})$ and output one if it exist. The symbol $\equiv$ is used to denote polynomial identity (the polynomials should have the same coefficients). We will focus mainly on the case where $\mathbb{F}$ is a field of characteristic zero (such as the rational numbers) or has a sufficiently large positive characteristic.

Observe that f is shift-equivalent to the zero polynomial if and only if f itself is the zero polynomial. Hence, SET is a natural generalization of the well-known Polynomial Identity Testing problem (PIT) in which we need to test whether $f(\mathbf{x}) \equiv 0$ given access to a succinct representation of f (say, as a circuit). A classical randomized algorithm by Schwartz-Zippel-DeMillo-Lipton [Sch80, Zip79, DL78] is known for PIT: evaluate f on a random input (from a large enough domain) and test if f evaluates to zero on that point. If f is non-zero, then it is not zero on a random point with very high probability. In contrast, it is not clear at all how to devise a randomized algorithm for SET. Unlike PIT, which is a 'co-NP' type problem (there is short proof that a polynomial is *not* zero), the SET problem is an '$\mathrm{RP}^{\mathrm{NP}}$' type problem (there is a short witness (the shift itself) that polynomials *are* shift equivalent, and verifying that witness is in RP).

The problem of equivalence of polynomials under shifts of the input first appeared in the works of Grigoriev, Lakshman, Saunders and Karpinski [GK93, GL95, LS94] (see also references therein), in the context of finding sparse shifts of a polynomial. That is, they were interested in finding a shift that will make a given polynomial sparse, if such a shift indeed exists. The main motivation for this question comes from considering polynomials in their sum-of-monomials representation (also called dense representation or depth-2 circuit complexity), and the goal is to find a shift that will make the representation more succinct. Later, in [Gri97], Grigoriev asked the following question: given two polynomials $f, g \in \mathbb{F}[\mathbf{x}]$, is there an efficient algorithm that can find whether there exists a shift $\mathbf{a} \in \mathbb{F}^n$ such that $f(\mathbf{x}+\mathbf{a}) \equiv g(\mathbf{x})$? In the same paper, Grigoriev gave algorithms for three versions of this problem: one deterministic for characteristic zero, one randomized for large enough characteristic $0 < p$ and one quantum for characteristic 2. The running time of Grigoriev's algorithms was polynomial in the dense representation. That is, for polynomials of degree d in n variables, the running time was $n^{O(d)}$ (which is an upper bound on the number of coefficients). In this paper, we address the same question as Grigoriev, but assume that the polynomials are given in some succinct representation (say, as arithmetic circuits). In this representation, one can hope for running time which is polynomial in the size of the given circuits (which can be exponentially small relative to the dense representation). For example the determinant polynomial has n^2 variables and degree n but can be given as a circuit of size $n^{O(1)}$ in the succinct representation.

Our main result is a new randomized (two-sided error) algorithm for SET. The algorithm runs in time polynomial in the circuit size of the given polynomials. In fact, we only require *black-box* access to the polynomials f and g and a bound on their degree and circuit size. Our algorithm is obtained as a reduction to the PIT problem. Hence, if we were able to perform deterministic PIT, we could also perform deterministic SET. For certain interesting restricted models of arithmetic computation, this already gives deterministic SET. For general circuits, our results show that it is equivalently hard to de-randomize PIT and SET, which is somewhat surprising as by the explanation above it seems as if SET is a much harder problem than PIT.

Below, we will state our results in the most general way, assuming f and g belong to some circuit classes closed under certain operations. The reason for doing this is that, in this way, one can see exactly what conditions are required to de-randomize the algorithm. That is, what kind of deterministic PIT is required to derive deterministic SET (in general we require PIT for a slightly larger class). Before giving a formal description of our results we take a moment to set up some necessary background on PIT and hitting sets.

1.1 PIT and Hitting Sets

We start by formally defining arithmetic circuits. For more background on arithmetic computation and arithmetic complexity we refer the reader to the survey [SY10].

Definition 1.1 (Arithmetic circuit)**.** *An* arithmetic circuit *C is a directed acyclic labeled graph in which the vertices are called 'gates'. The gates of C with in-degree 0 are called* inputs *and are labeled by either a variable from $\{x_1, \ldots, x_n\}$ or by field element from $\mathbb{F}$. Every other gate of C is labeled by either '$\times$' or '$+$' and has in-degree 2. There is one gate with out-degree 0, which we call the output gate. Each gate in C computes a polynomial in $\mathbb{F}[\mathbf{x}]$ in the natural way. We call the polynomial computed at the output gate 'the polynomial computed by C'. An arithmetic circuit is called a* formula *if its underlying graph is a tree.*

The PIT problem is defined as follows: we are given an arithmetic circuit C computing a polynomial $f \in \mathbb{F}[\mathbf{x}]$, and we have to determine whether the polynomial f is the zero polynomial or not. PIT is a central problem in algebraic complexity. Deterministically solving PIT is known to imply lower bounds for arithmetic circuits [HS80, Agr05, KI04, DSY09]. PIT also has some algorithmic implications. The famous AKS primality test [AKS04] is based on solving PIT for a specific polynomial. Randomized algorithms for finding a perfect matching in a given graph reduce the problem to PIT of certain determinants with variables as entries [Lov79, KUW86, MVV87].

In recent years, there has been considerable progress on the problem of obtaining deterministic PIT algorithms for restricted classes of circuits. The study of restricted models began with the class of sparse polynomials, which are also referred to as depth 2 circuits (of the form $\Sigma\Pi$). A long line of work, culminating in the algorithm of Klivans and Spielman [KS01] gives deterministic PIT for sparse polynomials. In the past decade a series of algorithms [DS06, KS07, KS11, SS11, KS09b, SS10, ASSS12] were devised to solve PIT for circuits of depth 3 with bounded fan-in, which are denoted as $\Sigma\Pi\Sigma(k)$ circuits. A more recent line of work, to which we will go back later in the paper, deals with read once branching programs and low rank tensors [FS12, FS13, FSS13].

There are two variants of the PIT problem: in the white-box model the PIT algorithm is given as input an actual arithmetic circuit computing $f \in \mathbb{F}[\mathbf{x}]$ and has to determine if $f \equiv 0$, possibly by inspecting the structure of the circuit. In the (harder) black-box model, we can only access the polynomial f by querying its value at points $\mathbf{a} \in \mathbb{F}^n$ of our choice (we are still assuming f *has* some small circuit). It is not hard to see that any deterministic black-box PIT algorithm works by evaluating f on some fixed set of points and outputs $f \equiv 0$ iff all of these evaluations result in zero. Such a set of evaluation points is called a *Hitting Set* for the class of circuits to which f is assumed to belong. It is clear that solving PIT in the black-box model is at least as hard as solving it in the white-box model and indeed, in some cases we have better algorithms in the white-box model than in the black-box model (compare e.g. [RS05] to [FS13] and [FSS13]).

More formally, to deterministically solve black-box PIT for a class of circuits $\mathcal{M}$, we need to be able to generate a hitting set $\mathcal{H}$ such that for each non-zero polynomial f computed by a circuit in $\mathcal{M}$, there exists a point $\mathbf{a} \in \mathcal{H}$ such that $f(\mathbf{a}) \neq 0$. If this is the case, we say that the set $\mathcal{H}$ *hits* the class $\mathcal{M}$, and that the point $\mathbf{a}$ *hits* f.

The following folklore result shows that there *exists* a small hitting set for the class of poly-size circuits (see Theorem 4.3 of [SY10] for a proof).

Theorem 1.2 (Non-constructive hitting sets). *For every n, d, s and a field $\mathbb{F}$ of size $|\mathbb{F}| \geq \max(d^2, s)$, there exists a set $\mathcal{H} \subseteq \mathbb{F}^n$ of size $|\mathcal{H}| = \mathsf{poly}(d, s)$ that is a hitting set for all circuits of size at most s and degree at most d. Furthermore, a random set $\mathcal{H}$ of the appropriate size is such a hitting set with high probability.*

We remark that the theorem above requires that the field size is at least polynomially larger than some of the parameters. This is necessary for constructing hitting sets since two non-identical polynomials might evaluate to the same value on all inputs from a sufficiently small sub-field (e.g., $x = x^p$ in $\mathbb{F}_p$). For simplicity, we will assume that we work over sufficiently large finite fields (so that they contain a hitting set), if necessary by going to an extension field. When working over characteristic zero we will implicitly assume that all constants involved in the hitting sets or in the computation have polynomially long bit representation so we do not have to keep track of that measure as well. This is quite reasonable given that explicit constructions of hitting sets have this property and that we can achieve this with randomized constructions as well.

Now, if what we want is to hit only a specific nonzero polynomial, then we do not need the full power of a hitting set. As we mentioned before, the randomized algorithm by Schwartz-Zippel-DeMillo-Lipton [Sch80, Zip79, DL78] gives us a point that hits a given nonzero polynomial with high probability. More formally we have:

Lemma 1.3 ([Sch80, Zip79, DL78]). *Let $f(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ be a nonzero polynomial of degree at most d, and let $T \subseteq \mathbb{F}$ be a finite set. If we choose $\mathbf{a} = (a_1, \ldots, a_n) \in T^n$ uniformly at random, then $\mathbf{Pr}[f(a) = 0] \leq d/|T|$.*

Notice that, to achieve error at most ε with this lemma, we should pick a set T of size $|T| \geq d/\varepsilon$. Generating such a uniformly random element $\mathbf{a}$ from T^n requires $n \cdot \lceil \log(d/\varepsilon) \rceil$ random bits.

1.2 Formal statement of our results

Our results rely on closure properties of the underlying circuit classes.

Definition 1.4. *Given a class of arithmetic circuits $\mathcal{M}$ we will say that $\mathcal{M}$ is closed under an operator $A : \mathbb{F}[\mathbf{x}] \mapsto \mathbb{F}[\mathbf{x}]$ if the following property holds. Let f be an n-variate polynomial of total degree d that is computed by a circuit of size s from $\mathcal{M}$. Then we require that $A(f)$ is computed by a circuit of size $\mathsf{poly}(n, d, s)$ from $\mathcal{M}$.*

For instance, one operator that is very common and under which all of the most studied circuit classes are closed is the restriction operator, namely, the operator that substitutes some of the variables of $f(\mathbf{x})$ by field elements. It is easy to see that by substituting some variables by field elements, the new polynomial will also be computed by a circuit of size less than s, and in general the new polynomial will also belong to the same class as f.

In addition, we will need to discuss closure under three different operators:

- **Directional partial derivatives:** The partial derivatives $\frac{\partial f}{\partial x_i}$ of a polynomial f are defined in the usual sense (over finite fields we use the formal definition for polynomials). We define the *first order partial derivative of f in direction* $\mathbf{a} \in \mathbb{F}^n$ to be

  $$f^{(1)}(\mathbf{a}, \mathbf{x}) \triangleq \sum_{t=1}^{n} a_t \cdot \frac{\partial f}{\partial x_t}(\mathbf{x})$$

  (see Definition 3.1). Apart from the class of general circuits (and formulas) that are closed under taking first order derivatives [**?**], the class of sparse polynomials (depth 2 circuits) is also closed under directional partial derivatives. Note, however, that depth-3 circuits with at most k multiplication gates, also known as $\Sigma\Pi\Sigma(k)$ circuits, are *not* closed under directional partial derivatives as these might increase the top fanin.

- **Homogeneous components:** If $f \in \mathbb{F}[\mathbf{x}]$ is a polynomial of degree d, we will denote the homogeneous component of degree k of f by $H^k(f(\mathbf{x}))$. General circuits and formulas are close under taking homogeneous components, and the same also holds for the class of sparse polynomials (see e.g the proof of Lemma 2.1).

- **Shifts:** Here we require that a class will be closed under the operation $f(\mathbf{x}) \mapsto f(\mathbf{x} + \mathbf{a})$ for some $\mathbf{a} \in \mathbb{F}^n$. Again, circuits and formulas as closed to shifts, however, the class of sparse polynomials is not.

We now describe our main result that solves the SET problem given a PIT algorithm.

Theorem 1.5 (Main theorem)**.** *Let $\mathbb{F}$ be a field of characteristic zero. Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two circuit classes such that*

1. *$\mathcal{M}_1$ is closed under taking homogeneous components and closed under (first-order) directional derivatives.*

2. *$\mathcal{M}_2$ is closed under taking shifts.*

3. *We have a (white-box) black-box PIT algorithm $\mathcal{P}$ for polynomials in $\mathcal{M}_1, \mathcal{M}_2$ and for polynomials of the form $f - g$, where $f \in \mathcal{M}_1$ and $g \in \mathcal{M}_2$.*

Then, there exists an algorithm $\mathcal{S}$ that, given (white-box) black-box access to polynomials $f \in \mathcal{M}_1, g \in \mathcal{M}_2$ and a bound d on the their degree, returns $\mathbf{a} \in \mathbb{F}^n$ so that $g(\mathbf{x} + \mathbf{a}) \equiv f(\mathbf{x})$, if such a shift exists, or returns FAIL, if none exist.

Furthermore:

- *The running time of $\mathcal{S}$ is polynomial in the running time of $\mathcal{P}$ and in the other parameters (n, d).*
- *If the PIT algorithm $\mathcal{P}$ is deterministic then so is $\mathcal{S}$.*
- *All of the above holds also for the case when $\mathbb{F}$ is a finite field with characteristic greater than d.*

Combining Theorem 1.5 with Lemma 1.3 we obtain a randomized SET algorithm for any pair of polynomials.

Theorem 1.6 (Randomized SET for pairs of polynomials)**.** *Let $\mathbb{F}$ be a field of characteristic zero or of characteristic larger than d. There exists a randomized algorithm that, given black box access to $f, g \in \mathbb{F}[\mathbf{x}]$ of degree at most d, returns $\mathbf{a} \in \mathbb{F}^n$ such that $g(\mathbf{x} + \mathbf{a}) \equiv f(\mathbf{x})$, if such a shift exists, or FAIL otherwise. The algorithm runs in time $\mathsf{poly}(n, d, \log(1/\varepsilon))$, where ϵ is the probability or returning a wrong answer (i.e., FAIL if a shift exists or a shift if none exists).*

Remark 1.7. *An interesting fact about Theorem 1.6 is that the algorithm we obtain has a two sided error (this can be seen from the proof). This fact is in contrast to the fact that most randomized algorithms in the algebraic setting have one-sided error.*

Theorem 1.5 already leads to deterministic algorithms for certain restricted models. For instance, in the recent works of Forbes and Shpilka [FS12, FS13] and of Forbes, Saptharishi and Shpilka [FSS13], the authors obtain a quasi-polynomial deterministic PIT algorithm for read-once oblivious algebraic branching programs (ROABPs). Their result, together with our algorithm, imply that we can find out whether two ROABPs are shift-equivalent in deterministic quasi-polynomial time. Since this class also captures tensors,[1] an application of our result is that we can find out whether two tensors are shift-equivalent in quasi-polynomial time (we refer the reader to [FS13, FSS13] for definitions of ROABPs and tensors).

Corollary 1.8. *There is a deterministic quasi-polynomial time algorithm that given black-box access to two polynomials f and g computed by read-once oblivious algebraic branching programs, decides whether there exists $\mathbf{a} \in \mathbb{F}^n$ such that $f(\mathbf{x} + \mathbf{a}) \equiv g(\mathbf{x})$ and in case that such a shift exists, the algorithm outputs one.*

As the class of sparse polynomials is closed under taking homogeneous components and

[1]We note that the work [ASS13] also gives a black-box PIT algorithm for tensors.

under first order directional derivatives (a directional derivative blows up the size of the circuit by at most a factor of n) we obtain the following corollary.

Corollary 1.9. *Let $\mathcal{M}_2$ be any circuit class so that*

1. *$\mathcal{M}_2$ is close under shifts.*
2. *There is a deterministic PIT algorithm testing if $f - g$ is zero for sparse f and $g \in \mathcal{M}_2$.*

Then, we can test whether f and g are shift-equivalent deterministically in time $\mathsf{poly}(n, s)$

As an application of our main theorem in the white-box model, we note that Saha et al. gave a polynomial time algorithm for testing whether a given sparse polynomial equals a $\Sigma\Pi\Sigma(k)$ circuit [SSS13]. Since their algorithm works in the white-box model, we can utilize it in the variant of our main theorem in the white-box model to find whether a given sparse polynomial and a polynomial in $\Sigma\Pi\Sigma(k)$ are shift-equivalent. We also note that we can make their algorithm work in the black-box case as well. Using the reconstruction algorithms of [Shp09, KS09a] we can first reconstruct the $\Sigma\Pi\Sigma(k)$ circuit in quasi-polynomial time. We can also interpolate the sparse polynomial in polynomial time (for interpolation of sparse polynomials see e.g. [KS01]) and then apply our methods together with the PIT algorithm of Saha et al. to solve the shift-equivalence problem.[2]

1.3 Overview of the algorithm

In this section we give a short overview our algorithm and its analysis. Assume we are given $f(\mathbf{x})$ and $g(\mathbf{x})$ and we have to find $\mathbf{a} \in \mathbb{F}^n$ such that $f(\mathbf{x}+\mathbf{a}) = g(\mathbf{x})$. Let us assume w.l.o.g. that $\deg(f) = \deg(g) = d$. Let us also denote $f(\mathbf{x}) = \sum_{i=0}^{d} H^i(f(\mathbf{x}))$ where each $H^i(f)$ is homogeneous of degree i and similarly, $g(\mathbf{x}) = \sum_{i=0}^{d} H^i(g(\mathbf{x}))$.

Now, let us compute the homogeneous components of $f(\mathbf{x}+\mathbf{a})$. Denote with $H^i(f(\mathbf{x}+\mathbf{a}))$ the homogeneous part of degree i of $f(\mathbf{x}+\mathbf{a})$. We have that

$$H^d(f(\mathbf{x}+\mathbf{a})) = H^d(f(\mathbf{x})).$$

Thus, our first step of the algorithm is to verify that

$$H^d(g(\mathbf{x})) = H^d(f(\mathbf{x})).$$

Next we move to degree $d-1$. A quick calculation gives

$$H^{d-1}(g(\mathbf{x})) = H^{d-1}(f(\mathbf{x}+\mathbf{a})) = H^{d-1}(f(\mathbf{x})) + \sum_{k=1}^{n} a_k \cdot \frac{\partial H^d(f(\mathbf{x}))}{\partial x_k}.$$

Observe that this is a linear equation in the entries of $\mathbf{a}$. It turns out that if our circuit class is closed under directional derivatives, that is, if the polynomial $\sum_{k=1}^{n} a_k \cdot \frac{\partial H^d(f(\mathbf{x}))}{\partial x_k}$ belongs to

[2]Note that the reconstruction algorithm of [Shp09, KS09a] returns so-called generalized $\Sigma\Pi\Sigma(k)$ circuits. Nevertheless, one can observe that the algorithm of Saha et al. works for such circuits as well.

the same circuit class as $f(\mathbf{x})$ (or a slightly larger class), and if we have a hitting set for the class of polynomials of the form $\sum_{k=1}^{n} a_k \cdot \frac{\partial H^d(f(\mathbf{x}))}{\partial x_k}$, for every $\mathbf{a} \in \mathbb{F}^n$, then we can solve this system of equations and find some solution $\mathbf{b}$ such that

$$H^{d-1}(g(\mathbf{x})) = H^{d-1}(f(\mathbf{x}+\mathbf{b})) = H^{d-1}(f(\mathbf{x})) + \sum_{k=1}^{n} b_k \cdot \frac{\partial H^d(f(\mathbf{x}))}{\partial x_k}.$$

As we will see in section 2.3, if we allow randomness then we can also solve this system of equations without having a hitting set.

Note that at this point we might have $\mathbf{b} \neq \mathbf{a}$. Hence, we have found a shift $\mathbf{b}$ that makes the homogeneous parts of degree d and $d-1$ in f and g equal. We now consider the homogeneous component of degree $d-2$. Here we have the system of equations

$$H^{d-2}(g(\mathbf{x})) = H^{d-2}(f(\mathbf{x}+\mathbf{a})) = H^{d-2}(f(\mathbf{x})) + \sum_{k=1}^{n} a_k \cdot \frac{\partial H^{d-1}(f(\mathbf{x}))}{\partial x_k} + \sum_{\ell,k=1}^{n} a_\ell a_k \frac{\partial^2 H^d(f(\mathbf{x}))}{\partial x_\ell \partial x_k}. \tag{1}$$

And now we seem to be in trouble as this is a system of quadratic equations in the entries of $\mathbf{a}$. Here comes our crucial observation. Recall that we have found $\mathbf{b}$ such that

$$H^{d-1}(g(\mathbf{x})) = H^{d-1}(f(\mathbf{x})) + \sum_{k=1}^{n} b_k \cdot \frac{\partial H^d(f(\mathbf{x}))}{\partial x_k}.$$

We also have that

$$H^{d-1}(g(\mathbf{x})) = H^{d-1}(f(\mathbf{x}+\mathbf{a})) = H^{d-1}(f(\mathbf{x})) + \sum_{k=1}^{n} a_k \cdot \frac{\partial H^d(f(\mathbf{x}))}{\partial x_k}.$$

Hence,

$$\sum_{k=1}^{n} (a_k - b_k) \cdot \frac{\partial H^d(f(\mathbf{x}))}{\partial x_k} = 0.$$

This means that the directional derivative of $H^d(f(\mathbf{x}))$ in direction $\mathbf{a}-\mathbf{b}$ is zero. Or, in other words, that the polynomial $H^d(f(\mathbf{x}))$ is fixed along that direction. This means that no matter how many derivatives we take along direction $\mathbf{a}-\mathbf{b}$ we always get the zero polynomial. Therefore, if we take a second derivative in direction $\mathbf{c}$ or in direction $\mathbf{c}+(\mathbf{a}-\mathbf{b})$ we will get the same answer, no matter what $\mathbf{c}$ is. In particular, this gives

$$\sum_{\ell,k=1}^{n} a_\ell a_k \frac{\partial^2 H^d(f(\mathbf{x}))}{\partial x_\ell \partial x_k} = \sum_{\ell,k=1}^{n} b_\ell b_k \frac{\partial^2 H^d(f(\mathbf{x}))}{\partial x_\ell \partial x_k},$$

as both sides compute the second directional derivatives in directions $\mathbf{a}$ and $\mathbf{b}$, respectively. Going back we now have that system (1) is equivalent to the system

$$H^{d-2}(g(\mathbf{x})) = H^{d-2}(f(\mathbf{x}+\mathbf{a})) = H^{d-2}(f(\mathbf{x})) + \sum_{k=1}^{n} a_k \cdot \frac{\partial H^{d-1}(f(\mathbf{x}))}{\partial x_k} + \sum_{\ell,k=1}^{n} b_\ell b_k \frac{\partial^2 H^d(f(\mathbf{x}))}{\partial x_\ell \partial x_k}. \tag{2}$$

Since we already computed **b**, we can look for a solution to both systems of equations (1) and (2) (as linear systems in the coefficients of **a**). Once we find such a solution, say **c**, we can use it to set up a new system of equations involving the homogeneous components of degree $d-3$ and so on.

Thus, our algorithm works in iterations. We start by solving a system of linear equations. We then use the solution that we found to set up another system and then we find a common solution to both systems. We use the solution that we have found to construct a third system of equations and then solve all three systems together etc. At the end we have a solution for all systems, and at this point it is not difficult to verify, that if such a shift **a** exists, then the solution that we found is indeed a valid shift. This can be verified by running one PIT for checking whether the shift of f that we have found and g are equivalent.

All the steps above can be completed using randomness, including solving the black-box system of equations, or using PIT for the relevant circuit classes.

1.4 Related work

The works of Grigoriev, Lakshman, Saunders and Karpinski [GK93, GL95, LS94], try to solve the problem of finding sparse shifts of given polynomials, in order to make their representation more succinct. In [GK93], Grigoriev and Karpinski studied the problem of finding sparse affine-shifts of multivariate polynomials $f(\mathbf{x})$, that is, transformations of the form $\mathbf{x} \mapsto A\mathbf{x}+\mathbf{b}$ where A is full-rank, which make the input polynomial $f(A\mathbf{x}+\mathbf{b})$ sparse. In [LS94], the authors consider the problem of finding sparse shifts of univariate polynomials, and of determining uniqueness of a sparse shift. Given an input polynomial $f(x)$, they use a criterion based on the vanishing of the Wronskian of some carefully designed polynomials, which depend on the derivatives $f^{(i)}(x)$, in order to obtain an efficient algorithm for the univariate case.

Later, in [Gri97], Grigoriev gave three algorithms for the SET problem, which were polynomial in the size of the dense representation of the input polynomials. His algorithms were based on a structural result about the set of shifts that stabilize the polynomial, that is, the set of points $\mathbf{a} \in \mathbb{F}^n$ for which $f(\mathbf{x}+\mathbf{a}) \equiv f(\mathbf{x})$. We denote this stabilizer by $\mathcal{S}_f$. He noticed that $\mathcal{S}_f$ is a subspace of $\mathbb{F}^n$ and that the set of shifts that are solutions to the SET problem with input polynomials f, g, which we denote $\mathcal{S}_{f,g}$, is a coset of $\mathcal{S}_f$. After this observation, Grigoriev established the following recursive relations between $\mathcal{S}_{f,g}$ and $\mathcal{S}_{\frac{\partial f}{\partial x_i},\frac{\partial g}{\partial x_i}}$, for each x_i:

$$\mathcal{S}_{f,g} = \bigcap_{i=1}^{n} \mathcal{S}_{\frac{\partial f}{\partial x_i},\frac{\partial g}{\partial x_i}} \cap \{\mathbf{a} \in \mathbb{F}^n \ : \ f(\mathbf{a}) = g(0)\}.$$

From these relations, Grigoriev devised a recursive algorithm that finds $\mathcal{S}_{f,g}$ by finding the subspaces corresponding to $\mathcal{S}_{\frac{\partial f}{\partial x_i},\frac{\partial g}{\partial x_i}}$. Because this recursive procedure will find all the subspaces $\mathcal{S}_{\frac{\partial f}{\partial m},\frac{\partial g}{\partial m}}$ for every monomial m of degree less than or equal to $d = \max(d_f, d_g)$, the running time of his algorithm is bounded by $n^{O(d)}$. Our approach is different from Grigoriev's in the sense that we avoid the recursive relations and find a shift by iteratively constructing a shift which makes f and g agree on their homogeneous parts of up to a certain degree, starting from the homogeneous parts of highest degree down to the homogeneous parts of lowest degree (i.e., the constant term).

The study of equivalences of general polynomials under affine transformations, which we refer to as affine-equivalence, was started by Kayal in [Kay12] (note that this generalizes the problem studied in [GK93]). We say that f and g are affine-equivalent if there exists a matrix A and a shift $\mathbf{b}$ such that $f(\mathbf{x}) = g(A\mathbf{x} + \mathbf{b})$. In this work, Kayal analyzes whether a given polynomial f can be obtained by an affine transformation of a given polynomial g, where g is usually taken to be a "complete" polynomial in some arithmetic circuit class, such as the Determinant or Permanent polynomials. In his paper, Kayal establishes NP-hardness of the general problem of determining affine-equivalence between two arbitrary polynomials. Moreover, he provides randomized algorithms for the affine-equivalence problem when one of the polynomials is the Permanent or the Determinant and the affine transformations $\mathbf{x} \mapsto A\mathbf{x} + \mathbf{b}$ are of a special form (in the case of Determinant and Permanent, the matrix A must be invertible). Kayal provides randomized algorithms for some other classes of homogeneous polynomials, and for more details we refer the reader to the paper [Kay12]. Our work is different from Kayal's work since in our setting we are only interested in shift-equivalences, and in this feature we are less general than Kayal's work, but we also consider larger classes of polynomials, in which case we are more general than Kayal's work.

Another line of works that has some resemblance to our results is the study of black-box groups. The well-known algorithm of Sims (see the book [Ser03]) finds a small set of generators for a permutation group given by black-box access. Our algorithm can be seen as finding a basis for the affine space of all shifts from f to g so in that sense it also finds generators for a black-box group where we do not have direct access to the group but rather to the objects it acts upon. An interesting point is that while Sims' algorithm works by constructing the group in a "bottom to top" fashion, namely starting with the identity element and slowly finding more generators, we on the other hand find a sequence of affine spaces, each contained in the proceeding ones until we reach the final space.

1.5 Organization

The rest of the paper is organized as follows: in section 2 we introduce some useful lemmas that one obtains from having PIT for a class of circuits. In section 3 we introduce some properties of homogeneous components of shifts of polynomials. In section 4 we define the space of shifts of a polynomial that do not change the polynomial at all (i.e. the stabilizer) and describe some of its properties. In section 5 we formally state and prove the main theorem of this paper, describing and analyzing the algorithms for testing shift-equivalence.

2 Preliminaries

In this section, we establish some notation that will be used throughout the paper and introduce some useful lemmas about simulation of circuits in the black-box setting. In addition, we state and prove a lemma on how to solve a linear system of polynomial equations in the black-box (or white-box) setting, given that one has a black-box (or white-box) PIT algorithm for linear combinations of the polynomials in question. In section 2.2, we show that if we are given white-box access to the input polynomials, then white-box PIT for linear combinations of these polynomials is enough to solve linear system of equations with these polynomials. On the other hand, in section 2.3, if we are given black-box access to the input polynomials, then we show that having a hitting set is enough. Notice that although the result in section 2.3 seems to be stronger than the result in section 2.2, the two results are actually not comparable, since in section 2.3 we are assuming that we have a hitting set, which is a stronger assumption than only having a white-box PIT algorithm, which is the assumption in section 2.2.

From this point on, we will use boldface for vectors, and regular font for scalars. Thus, we will denote the vector $(x_1, \dots, x_n)$ by $\mathbf{x}$ and if we want to multiply the vector $\mathbf{x}$ by a scalar z we will denote this product by $z\mathbf{x}$.

We will also assume that the ground field $\mathbb{F}$ either has characteristic zero or that its characteristic is larger than the degree of any polynomial that we will be working with. This assumption will be crucially used throughout sections 3 and 4. In addition, we denote the characteristic of $\mathbb{F}$ by $\mathsf{char}(\mathbb{F})$.

2.1 Interpolation in the Black-Box setting

For many problems in algebraic computation, it is useful to work with the homogeneous components of a polynomial, instead of directly working with the entire polynomial. In the black-box setting, we do not have direct black-box access to the homogeneous components of the given polynomial f. However, the next lemma shows that from black-box access to f we can obtain black-box access to its homogeneous components $H^0(f), \dots, H^d(f)$.

Lemma 2.1. *If we are given black-box access to a circuit $C(\mathbf{x})$ that computes a polynomial $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ of degree d, then we can obtain black-box access to the homogeneous components of f.*

Proof. We know that $f(\mathbf{x}) = \sum_{i=0}^{d} H^i(f(\mathbf{x}))$. Hence, we have that $f(z\mathbf{x}) = \sum_{i=0}^{d} z^i H^i(f(\mathbf{x}))$. If we let $\{\alpha_i\}_{0 \le i \le d}$ be $d+1$ distinct elements of $\mathbb{F}$ (or of an extension field of $\mathbb{F}$) and if we evaluate C on the points $\alpha_i \mathbf{x}$ we obtain the following equality:

$$\begin{pmatrix} 1 & \alpha_0 & \alpha_0^2 & \dots & \alpha_0^d \\ 1 & \alpha_1 & \alpha_1^2 & \dots & \alpha_1^d \\ 1 & \alpha_2 & \alpha_2^2 & \dots & \alpha_2^d \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha_d & \alpha_d^2 & \dots & \alpha_d^d \end{pmatrix} \cdot \begin{pmatrix} H^0(f(\mathbf{x})) \\ H^1(f(\mathbf{x})) \\ H^2(f(\mathbf{x})) \\ \vdots \\ H^d(f(\mathbf{x})) \end{pmatrix} = \begin{pmatrix} f(\alpha_0\mathbf{x}) \\ f(\alpha_1\mathbf{x}) \\ f(\alpha_2\mathbf{x}) \\ \vdots \\ f(\alpha_d\mathbf{x}) \end{pmatrix}$$

The matrix on the left side is a Vandermonde matrix, which is known to be invertible. Hence, by left-multiplying by its inverse we obtain:

$$\begin{pmatrix} 1 & \alpha_0 & \alpha_0^2 & \dots & \alpha_0^d \\ 1 & \alpha_1 & \alpha_1^2 & \dots & \alpha_1^d \\ 1 & \alpha_2 & \alpha_2^2 & \dots & \alpha_2^d \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha_d & \alpha_d^2 & \dots & \alpha_d^d \end{pmatrix}^{-1} \cdot \begin{pmatrix} f(\alpha_0\mathbf{x}) \\ f(\alpha_1\mathbf{x}) \\ f(\alpha_2\mathbf{x}) \\ \vdots \\ f(\alpha_d\mathbf{x}) \end{pmatrix} = \begin{pmatrix} H^0(f(\mathbf{x})) \\ H^1(f(\mathbf{x})) \\ H^2(f(\mathbf{x})) \\ \vdots \\ H^d(f(\mathbf{x})) \end{pmatrix}$$

Since we have black-box access to the values $f(\alpha_i\mathbf{x})$ through the circuit C, we also have black-box access to the homogeneous components of f through this construction. □

2.2 Finding linear dependencies among polynomials in the White-Box Setting

Suppose we have explicit access to the circuits computing the polynomials $g, h_1, h_2, \dots, h_k \in \mathbb{F}[\mathbf{x}]$. Then, how can we decide whether g is in the linear span of $h_1, h_2, \dots, h_k$? That is, does there exist an $\mathbf{a} \in \mathbb{F}^k$ such that $g(\mathbf{x}) \equiv \sum_{i=1}^k a_i h_i(\mathbf{x})$? Notice that we cannot try to solve a linear system for each possible monomial of g, since this process might lead us to an exponential number of equations.

In this subsection we answer the question above, assuming that we have a white-box PIT algorithm that hits the $\mathbb{F}$-span of the polynomials $g, h_1, \dots, h_k$, that is, polynomials of the form $a_0 g(\mathbf{x}) + \sum_{i=1}^k a_i h_i(\mathbf{x})$, where $a_i \in \mathbb{F},\ 0 \le i \le k$. Moreover, we can find such a linear combination, if it exists.

Lemma 2.2 (Decision to search reduction for white-box PIT)**.** *Given an arithmetic circuit C computing a non-zero n-variate polynomial f of degree d, and a white-box deterministic PIT algorithm that runs in polynomial time, we can find, in deterministic polynomial time, a point $\mathbf{a} \in \mathbb{F}^n$ such that $f(\mathbf{a}) \neq 0$.*

Proof. Let $S = \{a_0, \dots, a_d\}$ be a set of $d+1$ distinct values from $\mathbb{F}$. Notice that we can check, using the PIT algorithm, whether the restriction $x_1 = a_i \in S$ makes f vanish. Since the degree of f is d and $f \not\equiv 0$, there exists a value of $a_i \in S$ such that $f(a_i, x_2, \dots, x_n) \not\equiv 0$. Hence, by a linear scan over S we can find such an index $0 \le i \le d$ such that $f(a_i, x_2, \dots, x_n) \not\equiv 0$. Fix $x_1 = a_i$ and repeat this procedure with the other variables $\{x_2, \dots, x_n\}$. The running time is clearly bounded by nd times the running time of the PIT algorithm. □

Lemma 2.3. *Suppose we are given circuits computing the polynomials $g, h_1, h_2, \dots, h_k \in \mathbb{F}[\mathbf{x}]$. Assume further that we have a deterministic white-box PIT algorithm for linear combinations of $g, h_1, h_2, \dots, h_k$. Then, there exists a deterministic algorithm, with running time polynomial in the sizes of the circuits and k, which decides whether there exists $\mathbf{a} \in \mathbb{F}^k$ such that $\sum_{i=1}^k a_i h_i(\mathbf{x}) \equiv g(\mathbf{x})$. Moreover, the algorithm will output such an $\mathbf{a}$, if there exists one.*

Proof. We will be relying on the decision-to-search reduction of Lemma 2.2 and we will use it implicitly throughout in the proof.

As a first step, find a point $\mathbf{a}_1$ such that $h_1(\mathbf{a}_1) \neq 0$. Next, find a point $\mathbf{a}_2$ such that $h_1(\mathbf{a}_2)h_2(\mathbf{a}_1) - h_1(\mathbf{a}_1)h_2(\mathbf{a}_2) \neq 0$. If no such point $\mathbf{a}_2$ exists then we can discard h_2, since in this case h_2 will be in the span of h_1. If $h_1(\mathbf{x})h_2(\mathbf{a}_1) - h_1(\mathbf{a}_1)h_2(\mathbf{x}) \not\equiv 0$, then by Lemma 2.2 we can find such $\mathbf{a}_2$. That is why we can discard h_2 in case we are not able to find such a point. Proceed in this manner until we have scanned through all $h_1, \ldots, h_k$. More accurately, assume (wlog) that the polynomials $h_1, \ldots, h_c$, for some $c < \ell$, are linearly independent and their span contains the polynomials $h_1, \ldots, h_{\ell-1}$. At the ℓ^{th} step we consider the $c+1 \times c+1$ matrix M_ℓ that is defined as follows: $M_\ell[i,j] = \begin{cases} h_i(a_j), & \text{if } j \leq c \\ h_i(\mathbf{x}), & \text{if } j = c+1 \end{cases}$. We find a point $\mathbf{a}_\ell$ for which the determinant of M_ℓ is non-zero. Notice that this determinant is merely a linear combination of the polynomials $h_1, \ldots, h_\ell$, hence we have PIT for this polynomial and we can find such point $\mathbf{a}_\ell$, if one exists.

W.l.o.g., we can assume that $h_1, \ldots, h_r$ form a basis for the space defined by the $\mathbb{F}$-span of the polynomials $h_1, \ldots, h_k$. Hence, by our linear scan through the h_i's, we have found $\mathbf{a}_1, \ldots, \mathbf{a}_r$ such that the $r \times r$ matrix H having in its $(i,j)^{th}$ entry the value $h_i(\mathbf{a}_j)$ is full rank.

We now evaluate the polynomial g on all those r points and find the unique linear combination yielding $\sum_{i=1}^{r} b_i h_i(\mathbf{a}_j) = g(\mathbf{a}_j)$ for $1 \leq j \leq r$. Notice that we can find $\mathbf{b} \in \mathbb{F}^r$ by solving a system of linear equations over $\mathbb{F}$. This vector $\mathbf{b}$ will be unique since H is full rank. Notice that, by uniqueness of $\mathbf{b}$ and by the fact that $h_1, \ldots, h_r$ form a basis for the linear span of the h_i's, we have that g is a linear combination of the h_i's if, and only if, $\sum_{i=1}^{r} b_i h_i(\mathbf{x}) \equiv g(\mathbf{x})$. Hence, all we need to do is to check whether $\sum_{i=1}^{r} b_i h_i(\mathbf{x}) \equiv g(\mathbf{x})$. We can test this polynomial equality by running our PIT algorithm on the polynomial $g(\mathbf{x}) - \sum_{i=1}^{r} b_i h_i(\mathbf{x})$. If the PIT algorithm returns that this polynomial is the zero polynomial, then we found a linear combination. Otherwise, the algorithm returns that there exists no linear combination. □

2.3 Finding linear dependencies among polynomials in the Black-Box Setting

Here we assume that we only have black-box access to polynomials $g, h_1, h_2, \ldots, h_k \in \mathbb{F}[\mathbf{x}]$ and we wish to solve the same question as the one posed in the previous subsection, assuming a black-box PIT. We first note that the proof of Lemma 2.3 also works in the black-box case, but since we have a stronger assumption, namely, a hitting set rather than a white-box PIT algorithm, we have a more direct solution: We shall find a set of points S for which the equation $g(\mathbf{x}) \equiv \sum_{i=1}^{k} a_i h_i(\mathbf{x})$ is true if, and only if, $g(\mathbf{c}) \equiv \sum_{i=1}^{k} a_i h_i(\mathbf{c})$ for every $\mathbf{c} \in S$.

It turns out that if we have a hitting set $\mathcal{H}$ that hits the $\mathbb{F}$-span of the polynomials $g, h_1, \ldots, h_k$, then the points of $\mathcal{H}$ give the required set S.

The following lemma states formally the answer to the question above:

Lemma 2.4. *Suppose we have black-box access to polynomials $g, h_1, h_2, \ldots, h_k \in \mathbb{F}[\mathbf{x}]$ and that we have a hitting set $\mathcal{H}$ that hits the $\mathbb{F}$-span of the polynomials $g, h_1, h_2, \ldots, h_k$. Then, there exists a deterministic algorithm, with running time polynomial in $|\mathcal{H}|$ and k, which decides whether there exists $\mathbf{a} \in \mathbb{F}^k$ such that $g(x) \equiv \sum_{i=1}^{k} a_i h_i(\mathbf{x})$. Moreover, the algorithm will output such an $\mathbf{a}$, if one exists.*

Proof. Let $s = |\mathcal{H}|$ and let $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_s$ be an arbitrary ordering of the elements of $\mathcal{H}$. For a polynomial $f \in \mathbb{F}[\mathbf{x}]$, define the vector $\mathbf{v}_f \in \mathbb{F}^s$ as follows: $\mathbf{v}_f = (f(\mathbf{c}_1), f(\mathbf{c}_2), \ldots, f(\mathbf{c}_s))^T$. Then, it is enough to prove the following equivalence: $g(\mathbf{x}) \equiv \sum_{i=1}^{k} a_i h_i(\mathbf{x})$ if, and only if, $\mathbf{v}_g = \sum_{i=1}^{k} a_i \mathbf{v}_{h_i}$. This implies the lemma, since given the polynomials $g, h_1, \ldots, h_k$ and $\mathcal{H}$, we can construct the vectors $\mathbf{v}_g, \mathbf{v}_{h_1}, \ldots, \mathbf{v}_{h_k}$ and just solve the system of linear equations $\mathbf{v}_g = \sum_{i=1}^{k} a_i \mathbf{v}_{h_i}$, where the a_i's are the unknowns.

Here is the proof of the equivalence above: $g(\mathbf{x}) \equiv \sum_{i=1}^{k} a_i h_i(\mathbf{x})$ implies that $g(\mathbf{c}_r) = \sum_{i=1}^{k} a_i h_i(\mathbf{c}_r)$ for all $r \in [s]$, which implies that $\mathbf{v}_g = \sum_{i=1}^{k} a_i \mathbf{v}_{h_i}$. On the other hand, if $\mathbf{v}_g = \sum_{i=1}^{k} a_i \mathbf{v}_{h_i}$, then we have $\mathbf{v}_g - \sum_{i=1}^{k} a_i \mathbf{v}_{h_i} = 0$, which implies that $g(\mathbf{c}_r) - \sum_{i=1}^{k} a_i h_i(\mathbf{c}_r) = 0$, for all $r \in [s]$. Since $\mathcal{H}$ hits linear combinations of $g, h_1, \ldots, h_k$, the last set of equalities implies that the polynomial $g(\mathbf{x}) - \sum_{i=1}^{k} a_i h_i(\mathbf{x})$ vanishes on all points of $\mathcal{H}$, and therefore it must be the zero polynomial. This implies that $g(\mathbf{x}) \equiv \sum_{i=1}^{k} a_i h_i(\mathbf{x})$ and proves the lemma. □

Now, what if we do not have such a hitting set $\mathcal{H}$, but we are allowed randomness? Then, we can still answer the question above in the positive, with high probability, and find such a linear combination if one exists. More formally, we have:

Lemma 2.5. *Suppose we have black-box access to polynomials $g, h_1, h_2, \ldots, h_k \in \mathbb{F}[\mathbf{x}]$ and an upper bound d on their degrees. Let $0 < \varepsilon < 1$. Then, there exists a randomized algorithm, with running time $\mathsf{poly}(d, \log(1/\varepsilon), k)$, which decides correctly with probability at least $1 - \varepsilon$ whether there exists $\mathbf{a} \in \mathbb{F}^k$ such that $g(x) \equiv \sum_{i=1}^{k} a_i h_i(\mathbf{x})$. Moreover, with the same error probability the algorithm will output such an $\mathbf{a}$, if one exists.*

Proof. The proof of this lemma is similar to the proof of the white-box case, the difference being in the fact that we will choose our evaluation points according to Lemma 1.3. Let S be a set of size $|S| = \lfloor \frac{2dk}{\varepsilon} \rfloor$.

As a first step, pick a point $\mathbf{a}_1$ at random from S^n. By Lemma 1.3, if $h_1 \not\equiv 0$ then $h_1(\mathbf{a}_1) = 0$ with probability $\leq \varepsilon/2k$. If $h_1(\mathbf{a}_1) = 0$ but $h_1 \not\equiv 0$, then we will just assume that $h_1 \equiv 0$ and we will discard it (and in this part that our algorithm may make a mistake). Next, pick a point $\mathbf{a}_2$ at random from S^n. Again, by Lemma 1.3, if $h_1(\mathbf{a}_1)h_2(\mathbf{x}) - h_1(\mathbf{x})h_2(\mathbf{a}_1) \not\equiv 0$ then $h_1(\mathbf{a}_2)h_2(\mathbf{a}_1) - h_1(\mathbf{a}_1)h_2(\mathbf{a}_2) = 0$ with probability $\leq \varepsilon/2k$. If $h_1(\mathbf{a}_2)h_2(\mathbf{a}_1) - h_1(\mathbf{a}_1)h_2(\mathbf{a}_2) = 0$ we will always assume that h_2 is in the span of h_1 and thereby we will discard h_2 (in this part our algorithm may again make a mistake). We thus proceed in this manner, following the footsteps of the proof of Lemma 2.3.

As before we assume (wlog) that $h_1, \ldots, h_r$ form a basis for the space defined by the $\mathbb{F}$-span of the polynomials $h_1, \ldots, h_k$. Hence, by our linear scan through the h_i's, we have found $\mathbf{a}_1, \ldots, \mathbf{a}_r$ such that the $r \times r$ matrix H having in its $(i,j)^{th}$ entry the value $h_i(\mathbf{a}_j)$ is full rank. The probability that we made a mistake until this point will be $\leq r\varepsilon/2k \leq \varepsilon/2$, by the union bound.

We continue as in the proof of Lemma 2.3. Assuming that we made no mistake so far, we can find (by solving linear equations over $\mathbb{F}$) the unique point $\mathbf{b}$ such that g is a linear combination of the h_i's if, and only if, $\sum_{i=1}^{r} b_i h_i(\mathbf{x}) \equiv g(\mathbf{x})$. Hence, all we need to do is to check whether $\sum_{i=1}^{r} b_i h_i(\mathbf{x}) \equiv g(\mathbf{x})$. We can test this polynomial equality by applying

Lemma 1.3 on the polynomial $g(\mathbf{x}) - \sum_{i=1}^{r} b_i h_i(\mathbf{x})$, again drawing the point at random from S^n. If the PIT algorithm returns that this polynomial is the zero polynomial, then we found a linear combination. Otherwise, the algorithm returns that there exists no linear combination. The probability of the PIT making a mistake at this step is $\leq \varepsilon/2k \leq \varepsilon/2$. Hence, the total error of the entire algorithm is bounded by $\varepsilon/2 + \varepsilon/2 = \varepsilon$, as claimed. $\square$

3 Homogeneous Components of Shifts of a Polynomial

In this section we describe some properties of the homogeneous components of a shift of a polynomial. Throughout this section, let $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ be a polynomial of degree d, $\mathbf{a} \in \mathbb{F}^n$ be a point and $f(\mathbf{x} + \mathbf{a})$ be a shift of f. In general, when the field $\mathbb{F}$ is such that $\mathsf{char}(\mathbb{F}) = 0$ or $\mathsf{char}(\mathbb{F}) > d$, the homogeneous components of $f(\mathbf{x}+\mathbf{a})$ can be expressed as a linear combination of the appropriate (formal) directional derivatives of the homogeneous components of f on the direction $\mathbf{a}$. Before we state these properties more formally, we will need the following definitions:

Definition 3.1 (Directional Derivatives)**.** *The (formal) directional derivative of $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ of order 1 on the direction $\mathbf{a}$ is given by the following formula:*

$$f^{(1)}(\mathbf{a}, \mathbf{x}) \triangleq \sum_{t=1}^{n} a_t \cdot \frac{\partial f}{\partial x_t}(\mathbf{x}). \tag{3}$$

More, generally, The (formal) directional derivative of $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ of order r on the direction $\mathbf{a}$ is given by the following formula:

$$f^{(r)}(\mathbf{a}, \mathbf{x}) \triangleq \sum_{\mathbf{e} \in [n]^r} \left(\prod_{k=1}^{r} a_{e_k} \right) \cdot \frac{\partial^r f}{\partial x_{e_1} \ldots \partial x_{e_r}}(\mathbf{x}) \tag{4}$$

where we define $f^{(0)}(\mathbf{a}, \mathbf{x}) \triangleq f(\mathbf{x})$. If f is not homogeneous, for each homogeneous component $H^\ell(f)$ of f we define:

$$f_\ell^{(r)}(\mathbf{a}, \mathbf{x}) \triangleq \sum_{\mathbf{e} \in [n]^r} \left(\prod_{k=1}^{r} a_{e_k} \right) \cdot \frac{\partial^r H^\ell(f)}{\partial x_{e_1} \ldots \partial x_{e_r}}(\mathbf{x}). \tag{5}$$

Note that equation (4) in definition 3.1 agrees with the usual notion of directional derivatives in the continuous setting. We define $f_\ell^{(r)}(\mathbf{a}, \mathbf{x})$ to simplify the statement of Lemma 3.5. From this definition, and by using the fact that the degree of f is smaller than $\mathsf{char}(\mathbb{F})$, it is easy to see the following observations:

Observation 3.2. $f_i^{(1)}(\mathbf{a}, \mathbf{x}) = H^{(i-1)}(f_i(\mathbf{x}+\mathbf{a}))$. *Thus,* $f^{(1)}(\mathbf{a}, \mathbf{x}) = \sum_{i=1}^{\deg(f)} H^{(i-1)}(f_i(\mathbf{x}+\mathbf{a}))$.

Observation 3.3. *The polynomials $f^{(r)}(\mathbf{a}, \mathbf{x})$ have the following recursive structure:*

$$f^{(r+1)}(\mathbf{a}, \mathbf{x}) \equiv \sum_{j=1}^{n} a_j \cdot \frac{\partial(f^{(r)}(\mathbf{a}, \mathbf{x}))}{\partial x_j}. \tag{6}$$

This recursive structure implies that the directional derivatives of lower order exhibit a "domino effect," which can be captured in the following observation:

Observation 3.4. *If $f^{(1)}(\mathbf{a},\mathbf{x}) \equiv f^{(1)}(\mathbf{b},\mathbf{x})$ then $f^{(r)}(\mathbf{a},\mathbf{x}) \equiv f^{(r)}(\mathbf{b},\mathbf{x})$, for all $r \geq 1$.*

Proof. We will prove this observation by induction on r. We know that the claim is true for $r = 1$. Now, given that the claim is true for all values $1 \leq t \leq r$, we have:

$$\begin{aligned}
f^{(r+1)}(\mathbf{a},\mathbf{x}) &\equiv \sum_{j=1}^{n} a_j \cdot \frac{\partial(f^{(r)}(\mathbf{a},\mathbf{x}))}{\partial x_j} && \text{(by observation 3.3)}\\
&\equiv \sum_{j=1}^{n} a_j \cdot \frac{\partial(f^{(r)}(\mathbf{b},\mathbf{x}))}{\partial x_j} && \text{(by induction hypothesis on } r\text{)}\\
&\equiv \sum_{j=1}^{n} a_j \cdot \frac{\partial}{\partial x_j}\left(\sum_{\mathbf{e}\in[n]^r}\left(\prod_{k=1}^{r} b_{e_k}\right)\cdot \frac{\partial^r f}{\partial x_{e_1}\dots\partial x_{e_r}}(\mathbf{x})\right) && \text{(by definition 3.1)}\\
&\equiv \sum_{\mathbf{e}\in[n]^r}\left(\prod_{k=1}^{r} b_{e_k}\right)\cdot \frac{\partial^r}{\partial x_{e_1}\dots\partial x_{e_r}}\left(\sum_{j=1}^{n} a_j\cdot\frac{\partial f}{\partial x_j}(\mathbf{x})\right) && \text{(by rearranging the sum)}\\
&\equiv \sum_{\mathbf{e}\in[n]^r}\left(\prod_{k=1}^{r} b_{e_k}\right)\cdot \frac{\partial^r}{\partial x_{e_1}\dots\partial x_{e_r}}\left(f^{(1)}(\mathbf{a},\mathbf{x})\right) && \text{(by definition 3.1)}\\
&\equiv \sum_{\mathbf{e}\in[n]^r}\left(\prod_{k=1}^{r} b_{e_k}\right)\cdot \frac{\partial^r}{\partial x_{e_1}\dots\partial x_{e_r}}\left(f^{(1)}(\mathbf{b},\mathbf{x})\right) && \text{(by induction hypothesis)}\\
&\equiv \sum_{\mathbf{e}\in[n]^r}\left(\prod_{k=1}^{r} b_{e_k}\right)\cdot \frac{\partial^r}{\partial x_{e_1}\dots\partial x_{e_r}}\left(\sum_{j=1}^{n} b_j\cdot\frac{\partial f}{\partial x_j}(\mathbf{x})\right) && \text{(by definition 3.1)}\\
&\equiv \sum_{\mathbf{e}\in[n]^{r+1}}\left(\prod_{k=1}^{r+1} b_{e_k}\right)\cdot \frac{\partial^{r+1} f}{\partial x_{e_1}\dots\partial x_{e_{r+1}}}(\mathbf{x}) && \text{(by rearranging the sum)}\\
&\equiv f^{(r+1)}(\mathbf{b},\mathbf{x}) && \text{(by definition 3.1)}
\end{aligned}$$

and this concludes the inductive proof. ☐

Observation 3.4 tells us that if the first order directional derivatives are equal for two different directions $\mathbf{a}$ and $\mathbf{b}$, then all of the higher-order directional derivatives will also be equal. This observation will be crucial in the design of our algorithm.

Now that we defined directional derivatives, we can state the main lemma of this section, which gives us relations between the homogeneous components of $f(\mathbf{x})$ and $f(\mathbf{x}+\mathbf{a})$:

Lemma 3.5 (Taylor Expansion Lemma)**.** *Let* $\mathbf{a} \in \mathbb{F}^n$ *and let* $f \in \mathbb{F}[\mathbf{x}]$ *be such that* $deg(f) = d < \mathsf{char}(\mathbb{F})$ *Then, the following relations hold for all* $0 \leq i \leq d$*:*

$$H^i(f(\mathbf{x}+\mathbf{a})) \equiv \sum_{j=i}^{d} \frac{1}{(j-i)!} \cdot f_j^{(j-i)}(\mathbf{a},\mathbf{x}) \tag{7}$$

Proof. Notice that it is enough to show this lemma for the case where f is a single monomial, since the general case follows by additivity of partial derivatives.

If $f(\mathbf{x}) = \prod_{j=1}^{n} x_j^{d_j}$, we have that

$$f(\mathbf{x}+\mathbf{a}) = \prod_{k=1}^{n} (x_k + a_k)^{d_k}$$

which implies

$$\begin{aligned}
H^i(f(\mathbf{x}+\mathbf{a})) &= \sum_{\substack{j_1+j_2+\ldots+j_n=i \\ j_k \geq 0,\ k\in[n]}} \prod_{k=1}^{n} \binom{d_k}{j_k} a_k^{d_k-j_k} x_k^{j_k} \\
&= \sum_{\substack{j_1+j_2+\ldots+j_n=i \\ j_k \geq 0,\ k\in[n]}} \left(\prod_{k=1}^{n} \frac{1}{(d_k-j_k)!} a_k^{d_k-j_k} \right) \cdot \left(\prod_{k=1}^{n} \frac{d_k!}{j_k!} x_k^{j_k} \right) \\
&= \sum_{\substack{j_1+j_2+\ldots+j_n=i \\ j_k \geq 0,\ k\in[n]}} \left(\prod_{k=1}^{n} \frac{1}{(d_k-j_k)!} a_k^{d_k-j_k} \right) \cdot \frac{\partial^{d-i} f}{\prod_{k\in[n]} (\partial x_k)^{d_k-j_k}}(\mathbf{x}) \\
&=^{(*)} \frac{1}{(d-i)!} \cdot \sum_{\mathbf{e}\in[n]^{d-i}} \left(\prod_{k=1}^{d-i} a_{e_k} \right) \cdot \frac{\partial^{d-i} f}{\partial x_{e_1} \ldots \partial x_{e_{d-i}}}(\mathbf{x}) \\
&= \frac{1}{(d-i)!} \cdot f^{(d-i)}(\mathbf{a},\mathbf{x}),
\end{aligned}$$

where equality $(*)$ follows as each term $\prod_{k=1}^{n} a_k^{d_k-j_k}$ is counted $\binom{d-i}{d_1-j_1,\ldots,d_n-j_n}$ many times when in the sum $\sum_{\mathbf{e}\in[n]^{d-i}} \left(\prod_{k=1}^{d-i} a_{e_k} \right)$.

The equations above and additivity of partial derivatives imply that the lemma is true when f is a homogeneous polynomial. The proof of the general case is as follows:

$$\begin{aligned}
H^i(f(\mathbf{x}+\mathbf{a})) = H^i\left(\sum_{j=i}^{d} H^j(f)(\mathbf{x}+\mathbf{a}) \right) &= \sum_{j=i}^{d} H^i\left(H^j(f)(\mathbf{x}+\mathbf{a}) \right) \\
&= \sum_{j=i}^{d} \frac{1}{(j-i)!} \cdot f_j^{(j-i)}(\mathbf{a},\mathbf{x})
\end{aligned}$$

Where the last equality is true because the lemma is true for homogeneous polynomials, and each $H^j(f)$ is a homogeneous polynomial of degree j. □

Lemma 3.5 can be seen as the multivariate Taylor expansion of the polynomial $f(\mathbf{x})$ around the point $\mathbf{a} \in \mathbb{F}^n$.

In order to use Lemma 3.5, we need to have access to the polynomials $f_k^{(r)}(\mathbf{a}, \mathbf{x})$ defined in the lemma. The following observations allow us to have the type of accesses we need.

Observation 3.6. *The polynomials $f_k^{(r)}(\mathbf{a}, \mathbf{x})$ are a constant multiple of the homogeneous components of degree $k - r$ of $H^k(f)(\mathbf{x} + \mathbf{a})$.*

This observation is important because given black-box access to f, we can obtain black-box access to the polynomials $f_k^{(r)}(\mathbf{a}, \mathbf{x})$ by interpolation of the polynomials $H^k(f)(\mathbf{x} + \mathbf{a})$, as we do in Lemma 2.1.

Notice that if we are only concerned with a bound on the size of a circuit computing the homogeneous components of a polynomial f, then by a result of Strassen in [Str73] we have the following theorem:

Theorem 3.7. *If f can be computed by an arithmetic circuit of size s, then for every $k \in \mathbb{N}$, there is a homogeneous circuit of size at most $O(r^2 s)$ computing all of the polynomials $H^k(f)$, where $0 \leq k \leq r$. Moreover, given access to the circuit computing f, we can construct the homogeneous circuit computing the homogeneous components of f.*

A straightforward consequence of this theorem and of observation 3.6 is stated below:

Corollary 3.8. *If f has degree d and can be computed by an arithmetic circuit of size s, then for every shift $\mathbf{a} \in \mathbb{F}^n$, there is a homogeneous circuit of size at most $O(d^4 s)$ computing all of the polynomials $f_k^{(r)}(\mathbf{a}, \mathbf{x})$, where $0 \leq k, r \leq d$. Moreover, given access to the circuit computing f, we can construct the homogeneous circuit computing all the polynomials $f_k^{(r)}(\mathbf{a}, \mathbf{x})$.*

4 Kernel of Shifts of a Polynomial

As observed by Grigoriev in [Gri97], the set of points $\mathbf{a} \in \mathbb{F}^n$ such that $f(\mathbf{x} + \mathbf{a}) \equiv f(\mathbf{x})$, which we call the *kernel* of f, forms a linear subspace of $\mathbb{F}^n$. In this section, we describe some properties of the kernel and introduce some lemmas which describe the relationship between points $\mathbf{a}$ in the kernel and the directional derivatives of f on the direction $\mathbf{a}$.

We begin with the following definitions:

Definition 4.1. *Let $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ be a polynomial of degree d. We define the* kernel *of f as the set*

$$\mathcal{S}_f = \{\mathbf{a} \in \mathbb{F}^n \mid f(\mathbf{x} + \mathbf{a}) \equiv f(\mathbf{x})\},$$

that is, $\mathcal{S}_f$ is the set of all points $\mathbf{a} \in \mathbb{F}^n$ such that if we shift the input of $f(\mathbf{x})$ by $\mathbf{a}$ we obtain the same formal polynomial. Here $\mathbf{x}$ is regarded as a formal set of variables and $\mathbf{a}$ is a point in $\mathbb{F}^n$.

We can observe the following properties of the kernel $\mathcal{S}_f$:

Observation 4.2. *Let $f \in \mathbb{F}[\mathbf{x}]$ be a polynomial of degree d, where $d < \mathsf{char}(\mathbb{F})$ if $\mathsf{char}(\mathbb{F}) \neq 0$. Then the kernel $\mathcal{S}_f$ is a subspace of $\mathbb{F}^n$.*

Proof. We need to check three conditions:

(i) $0 \in \mathcal{S}_f$

(ii) $\mathbf{a}, \mathbf{b} \in \mathcal{S}_f \Rightarrow \mathbf{a} + \mathbf{b} \in \mathcal{S}_f$

(iii) $\mathbf{a} \in \mathcal{S}_f \Rightarrow t\mathbf{a} \in \mathcal{S}_f$, for all $t \in \mathbb{F}$

Conditions (i) and (ii) are trivial to check. Hence, we only need to show that condition (iii) holds. By repeatedly applying (ii), we have $\mathbf{a} \in \mathcal{S}_f \Rightarrow k\mathbf{a} \in \mathcal{S}_f$, for any $k \in \mathbb{N}$. In particular, since the degree of f is less then the characteristic of $\mathbb{F}$, we have that $k\mathbf{a} \in \mathcal{S}_f$ for $0 \leq k \leq d$, which are all distinct values. Hence, the polynomial $f(\mathbf{x} + t\mathbf{a}) - f(\mathbf{x}) \in \mathbb{F}(\mathbf{x})[t]$ has degree $\leq d$ in t and has at least $d + 1$ distinct roots. This implies that $f(\mathbf{x} + t\mathbf{a}) - f(\mathbf{x})$ must vanish as a polynomial in t, which implies that $f(\mathbf{x} + t\mathbf{a}) - f(\mathbf{x}) \equiv 0$ for all $t \in \mathbb{F}$. This proves condition (iii) and therefore $\mathcal{S}_f$ is a subspace of $\mathbb{F}^n$. ☐

More generally, we can define the set of shift-equivalences between two polynomials $f, g \in \mathbb{F}[\mathbf{x}]$:

Definition 4.3. *Given two polynomials $f, g \in \mathbb{F}[\mathbf{x}]$, we define the set of shift-equivalences of f and g as*

$$\mathcal{S}_{f,g} = \{\mathbf{a} \in \mathbb{F}^n \mid f(\mathbf{x} + \mathbf{a}) \equiv g(\mathbf{x})\},$$

that is, $\mathcal{S}_{f,g}$ is the set of all points $\mathbf{a} \in \mathbb{F}^n$ such that if we shift the input of $f(\mathbf{x})$ by $\mathbf{a}$ we obtain the formal polynomial $g(\mathbf{x})$.

Note that the set $\mathcal{S}_{f,g}$ is intrinsically related to $\mathcal{S}_f$, since if we have any two elements $\mathbf{a}$ and $\mathbf{b}$ of $\mathcal{S}_{f,g}$, we must have that $\mathbf{a} - \mathbf{b} \in \mathcal{S}_f$. In other words, $\mathcal{S}_{f,g}$ is a coset of $\mathcal{S}_f$. Furthermore, it must hold that $\mathcal{S}_f = \mathcal{S}_g$.

Lemma 4.4. *Let $f, g \in \mathbb{F}[\mathbf{x}]$ such that there exists $\mathbf{a} \in \mathbb{F}^n$ for which $f(\mathbf{x} + \mathbf{a}) = g(\mathbf{x})$. Then $\mathcal{S}_f = \mathcal{S}_g$. Furthermore, for any such $\mathbf{a}$ we have $\mathcal{S}_{f,g} = \mathcal{S}_f + \mathbf{a}$.*

Proof. Let $\mathbf{b} \in \mathcal{S}_g$. Then $f(\mathbf{x} + \mathbf{b}) = f((\mathbf{x} - \mathbf{a} + \mathbf{b}) + \mathbf{a}) = g(\mathbf{x} - \mathbf{a} + \mathbf{b}) = g(\mathbf{x} - \mathbf{a}) = f(\mathbf{x})$. Hence, $\mathcal{S}_g \subseteq \mathcal{S}_f$. The other direction is similar.

Given $\mathbf{a}$ and $\mathbf{b}$ in $\mathcal{S}_{f,g}$ we have $f(\mathbf{x} - \mathbf{a} + \mathbf{b}) = g(\mathbf{x} - \mathbf{a}) = f(\mathbf{x})$. Hence $\mathbf{b} - \mathbf{a} \in \mathcal{S}_f$. Thus, $\mathcal{S}_{f,g} \subseteq \mathcal{S}_f + \mathbf{a}$. It is also straightforward to verify that $\mathcal{S}_f + \mathbf{a} \subseteq \mathcal{S}_{f,g}$. ☐

Another interesting property which relates the kernel to directional derivatives is captured by the following lemma, which states that a shift $\mathbf{a} \in \mathbb{F}^n$ is in the kernel of shifts $\mathcal{S}_f$ if, and only if, the first order directional derivative of f in the direction $\mathbf{a}$ is zero.

Lemma 4.5. *Let $f \in \mathbb{F}[\mathbf{x}]$ be a polynomial of degree d, where $d < \mathsf{char}(\mathbb{F})$ if $\mathsf{char}(\mathbb{F}) \neq 0$. Then, $\mathbf{a} \in \mathcal{S}_f$ if, and only if, $f^{(1)}(\mathbf{a}, \mathbf{x}) \equiv 0$.*

Proof. After a suitable change of basis that maps $\mathbf{a}$ to $\mathbf{e}_1$, we need to prove that $\mathbf{e}_1 \in \mathcal{S}_f$ if, and only if, $f^{(1)}(\mathbf{e}_1, \mathbf{y}) \equiv 0$, where $\mathbf{y}$ is the image of $\mathbf{x}$ under this change of basis. Since $f^{(1)}(\mathbf{e}_1, \mathbf{y}) = \dfrac{\partial f}{\partial y_1}(\mathbf{y})$, we must show that $\mathbf{e}_1 \in \mathcal{S}_f$ if, and only if, $\dfrac{\partial f}{\partial y_1}(\mathbf{y}) \equiv 0$.

To see the first direction, note that if $\dfrac{\partial f}{\partial y_1}(\mathbf{y}) \equiv 0$ and $d < \mathsf{char}(\mathbb{F})$ then $f \in \mathbb{F}[y_2, \ldots, y_n]$ which implies $f(\mathbf{y} + \mathbf{e}_1) \equiv f(\mathbf{y})$. Hence, $\mathbf{e}_1 \in \mathcal{S}_f$.

On the other hand, let us write $f(\mathbf{y}) = \sum_{i=0}^{d} y_1^i \cdot f_i(y_2, \ldots, y_n)$. Let k be the highest index for which $f_k(y_2, \ldots, y_n) \neq 0$. If $\dfrac{\partial f}{\partial y_1}(\mathbf{y}) \not\equiv 0$, then $k > 0$. Let $\mathbf{c} \in \mathbb{F}^n$ be such that $f_k(c_2, \ldots, c_n) \neq 0$ and $c_1 = 0$. Then, if we define $b_i = f_i(c_2, \ldots, c_n)$, for $0 \le i \le k$, we have $f(\mathbf{c}) = b_0$ and $f(\mathbf{c} + t\mathbf{e}_1) = b_k t^k + \sum_{i=1}^{k-1} b_i t^i$, where $b_k \neq 0$. This implies $f(\mathbf{c} + t\mathbf{e}_1) - f(\mathbf{c}) = b_k t^k + \sum_{i=1}^{k-1} b_i t^i$. Hence, there exists $t \in \mathbb{F}$ such that $b_k t^k + \sum_{i=1}^{k-1} b_i t^i \neq 0 \Rightarrow f(\mathbf{c} + t\mathbf{e}_1) - f(\mathbf{c}) \neq 0$. Thus, $f(\mathbf{y} + t\mathbf{e}_1) - f(\mathbf{y}) \not\equiv 0$ and so $\mathbf{e}_1 \notin \mathcal{S}_f$. □

An easy corollary of Lemma 4.5 and of observation 3.4 is the following:

Corollary 4.6. *If* $\mathbf{a} \in \mathcal{S}_f$ *then* $f^{(r)}(\mathbf{a}, \mathbf{x}) \equiv 0$*, for all* $r \ge 1$.

Another property that easily follows from linearity of $f^{(1)}(\mathbf{a}, \mathbf{x})$ (in $\mathbf{a}$) and from Lemma 4.4 is captured by the following lemma:

Lemma 4.7. *If* $\mathbf{a} \in \mathcal{S}_{f,g}$ *and* $\mathbf{b} \in \mathbb{F}^n$ *then* $\mathbf{b} \in \mathcal{S}_{f,g}$ *if, and only if,* $f^{(1)}(\mathbf{a}, \mathbf{x}) \equiv f^{(1)}(\mathbf{b}, \mathbf{x})$*. Thus,* $\mathbf{b} \in \mathcal{S}_{f,g}$ *if, and only if,* $f_i^{(1)}(\mathbf{b}, \mathbf{x}) \equiv f_i^{(1)}(\mathbf{a}, \mathbf{x})$ *for all* $0 \le i \le d$.

Proof. $f^{(1)}(\mathbf{a}, \mathbf{x}) \equiv f^{(1)}(\mathbf{b}, \mathbf{x})$ if, and only if, $f^{(1)}(\mathbf{b} - \mathbf{a}, \mathbf{x}) \equiv 0$. By Lemma 4.5 this is equivalent to $\mathbf{b} - \mathbf{a} \in \mathcal{S}_f$ and hence to $\mathbf{b} \in \mathcal{S}_f + \mathbf{a}$. This is equivalent, by Lemma 4.4, to having $\mathbf{b} \in \mathcal{S}_{f,g}$ as desired.

The second part of the lemma is immediate. □

5 Proof of Equivalence Under Shifts

In this section we give intuition and an overview of our algorithm in subsection 5.1, followed by a formal description of the algorithm and its analysis in subsection 5.2.

5.1 Overview of the Algorithm

In this section, we will describe an overview of the steps in our algorithm. The high level idea of the algorithm was given in section 1.3. For the sake of clarity, we will leave the explanations of the preprocessing stage for the analysis of the algorithm and we will assume that the input given is already preprocessed accordingly.

In the highest level, our algorithm will produce a candidate shift $\mathbf{a}$ such that $f(\mathbf{x}+\mathbf{a}) \equiv g(\mathbf{x})$ and then use PIT on the polynomial $f(\mathbf{x}) - g(\mathbf{x}-\mathbf{a})$, to check that the solution $\mathbf{a}$ is indeed a good shift. We need to perform the PIT on the polynomial $f(\mathbf{x}) - g(\mathbf{x}-\mathbf{a})$ because $\mathcal{M}_2$ is closed under shifts. We proceed in this way because this approach allows us to assume from the beginning on that $\mathcal{S}_{f,g} \neq \emptyset$. For this section, we can assume that $\mathcal{S}_{f,g} \neq \emptyset$, that $\mathbf{c} \in \mathcal{S}_{f,g}$ and that $d_f = d_g = d$.

By our Taylor Expansion Lemma (Lemma 3.5), to find a good shift $\mathbf{c} \in \mathcal{S}_{f,g}$ we need to solve the system of polynomial equations (in the variables $\mathbf{a}$) given by the set of equations (7) in the Lemma. We cannot hope to solve these equations directly, since that would involve solving non-linear systems of equations. However, Lemma 4.7 tells us that in order to find a good shift, we only need to obtain black-box access to the polynomials $f_k^{(1)}(\mathbf{c}, \mathbf{x})$, where $\mathbf{c} \in \mathcal{S}_{f,g}$. If we succeed in obtaining black-box access to these polynomials, finding a good shift will only involve solving a linear system of polynomial equations in the black-box setting, which we can do by any of the lemmas: Lemma 2.3, Lemma 2.4 or Lemma 2.5, depending on which case we are in. Hence, our approach to solve the original set of equations is to obtain black-box access to the polynomials $f_k^{(1)}(\mathbf{c}, \mathbf{x})$.

Note that we cannot obtain direct access to $f_k^{(1)}(\mathbf{c}, \mathbf{x})$ (through interpolation) from neither f nor g, for a general k. However, from f and g we have black-box access to $f_d^{(1)}(\mathbf{c}, \mathbf{x})$, since $f_d^{(1)}(\mathbf{c}, \mathbf{x}) = H^{d-1}(g(\mathbf{x})) - H^{d-1}(f(\mathbf{x}))$. It turns out that this initial information is enough for the algorithm to find a good shift. To find the shift we will iteratively find candidate solutions $\mathbf{a}_r$, such that $f_k^{(1)}(\mathbf{c}, \mathbf{x}) \equiv f_k^{(1)}(\mathbf{a}_r, \mathbf{x})$ for all $d - r \leq k \leq d$. Then, by the domino effect from Observation 3.4 we have that $f_k^{(t)}(\mathbf{c}, \mathbf{x}) \equiv f_k^{(t)}(\mathbf{a}_r, \mathbf{x})$, for all $t \geq 1$. Hence, once we find $\mathbf{a}_k$ the domino effect and Lemma 3.5 imply that we can find $\mathbf{a}_{r+1}$ simply by solving linear equations. In the end, if the algorithm does not fail, we will obtain $\mathbf{a}_d$ such that $f_k^{(1)}(\mathbf{c}, \mathbf{x}) \equiv f_k^{(1)}(\mathbf{a}_d, \mathbf{x})$ for all $0 \leq k \leq d$, and thus by Lemma 4.7 we must have $\mathbf{a}_d \in \mathcal{S}_{f,g}$. This domino effect lies at the crux of the proof of correctness of our algorithm.

5.2 Formal Description and Proof of Correctness

For simplicity, we will describe the algorithm receiving the input already preprocessed.

```
Algorithm 1: Main Algorithm
  Input: black-boxes (or white-boxes) for polynomials $f \in \mathcal{M}_1, g \in \mathcal{M}_2$, and degree of $f$,
         which we denote by $d$.
  Output: a non-zero shift in $\mathcal{S}_{f,g}$, if one exists, or FAIL, if $\mathcal{S}_{f,g} = \emptyset$.
  By interpolation, obtain black-box access to the homogeneous components
  $H^0(f), H^1(f), \dots, H^d(f)$ and $H^0(g), H^1(g), \dots, H^d(g)$.
  Set $\mathbf{a}_0 \leftarrow 0$.
  for $k = 1, \dots, d$ do
      Solve, via any appropriate lemma from subsections 2.2 or 2.3[a], the linear system
      given by the following equations, where in these equations the variables are the
      entries of $\mathbf{b}$ and we have one equation for each $i$ such that $d - k \leq i \leq d$.

          $$H^i(g(\mathbf{x})) = H^i(f(\mathbf{x})) + f_{i+1}^{(1)}(\mathbf{b}, \mathbf{x}) + \sum_{j=i+2}^{d} \frac{1}{(j-i)!} \cdot f_j^{(j-i)}(\mathbf{a}_{k-1}, \mathbf{x}) \quad (8)$$

      If system of equations (8) has no solution, return FAIL
      Else, $\mathbf{a}_k \leftarrow \mathbf{b}$.
  end
  Perform PIT on $f(\mathbf{x}) - g(\mathbf{x} - \mathbf{a}_d)$.
  If $f(\mathbf{x}) - g(\mathbf{x} - \mathbf{a}_d) \equiv 0$, return $\mathbf{a}_d$
  Else, return FAIL
```

[a]If we are in the white-box setting, we will use Lemma 2.3, if we are in the black-box setting and we have a hitting set, then we will use Lemma 2.4, or if we are in the black-box setting and are allowed randomness we will use Lemma 2.5.

Preprocessing Stage: In case $\mathcal{S}_{f,g} = \emptyset$, our algorithm might do meaningless computations, but because we will verify our candidate solution, our algorithm is sure to return that no shift exists in the end. Notice that if we have $d_f \neq d_g$, even the interpolation step that we perform in the beginning will err when computing homogeneous components of g (because we will not interpolate with the proper degree). However, this is ok because we have the PIT step in the end, which will prevent us from returning any wrong answers that may arise from the meaningless computations.

Hence, from now on we can, and will, assume that $\mathcal{S}_{f,g} \neq \emptyset$. In particular, this implies that we can assume that there exists $\mathbf{c} \in \mathcal{S}_{f,g}$ and that $d_f = d_g = d$. Thus, our algorithm will assume that the input is given by two polynomials $f, g \in \mathbb{F}[\mathbf{x}]$ of degree upper bounded by the degree of f, which we will denote by d. Notice that we can also assume that d is the exact degree of f, since from the upper bound on the degree we can interpolate f and perform PIT on each homogeneous components of f (recall $\mathcal{M}_1$ is closed under homogeneous components). Then, the degree of f will be the value of the highest non vanishing homogeneous component.

Thus, we will assume that d is the exact degree of f, as opposed to an upper bound.

Analysis of the Algorithm in the Black-Box case, with a Hitting Set:

Proof. Notice that if $\mathcal{S}_{f,g} = \emptyset$, then even if the algorithm finishes the **for** loop, it will return FAIL, since PIT on $f(\mathbf{x}) - g(\mathbf{x} - \mathbf{a}_d)$ will return that the polynomial is non-zero. Therefore, we never err in this case. Hence, for the rest of the analysis, let us assume that $\mathcal{S}_{f,g} \neq \emptyset$. This implies that there exists a shift $\mathbf{c} \in \mathcal{S}_{f,g}$. Since $g(\mathbf{x}) \equiv f(\mathbf{x}+\mathbf{c})$ it holds that $H^i(g(\mathbf{x})) \equiv H^i(f(\mathbf{x}+\mathbf{c}))$. From Lemma 3.5, we have

$$H^i(f(\mathbf{x}+\mathbf{c})) \equiv \sum_{j=i}^{d} \frac{1}{(j-i)!} \cdot f_j^{(j-i)}(\mathbf{c},\mathbf{x}) \equiv H^i(f(\mathbf{x})) + f_{i+1}^{(1)}(\mathbf{c},\mathbf{x}) + \sum_{j=i+2}^{d} \frac{1}{(j-i)!} \cdot f_j^{(j-i)}(\mathbf{c},\mathbf{x}).$$

Hence,

$$H^i(g(\mathbf{x})) - H^i(f(\mathbf{x})) - \sum_{j=i+2}^{d} \frac{1}{(j-i)!} \cdot f_j^{(j-i)}(\mathbf{c},\mathbf{x}) \equiv f_{i+1}^{(1)}(\mathbf{c},\mathbf{x}) \tag{9}$$

for all $0 \le i \le d$.

Recall, that by Lemma 4.7, to find a shift in $\mathcal{S}_{f,g}$ it is enough to find a shift $\mathbf{b}$ such that $f_i^{(1)}(\mathbf{b},\mathbf{x}) \equiv f_i^{(1)}(\mathbf{c},\mathbf{x})$ for all $0 \le i \le d$.

Observation 3.4 implies that if $f_i^{(1)}(\mathbf{b},\mathbf{x}) \equiv f_i^{(1)}(\mathbf{c},\mathbf{x})$ for some $0 \le i \le d$, then $f_i^{(r)}(\mathbf{b},\mathbf{x}) \equiv f_i^{(r)}(\mathbf{c},\mathbf{x})$ for all $r \ge 1$, for this particular i. Therefore, if we show that our algorithm maintains the invariant

$$f_i^{(1)}(\mathbf{a}_k,\mathbf{x}) \equiv f_i^{(1)}(\mathbf{c},\mathbf{x}), \quad \text{for all } i \text{ s.t. } d-k+1 \le i \le d \tag{10}$$

at every iteration of the loop, then at the end of the loop we will have $f_i^{(1)}(\mathbf{a}_d,\mathbf{x}) \equiv f_i^{(1)}(\mathbf{c},\mathbf{x})$ for all $0 \le i \le d$, which by Lemma 4.7 implies that $\mathbf{a}_d \in \mathcal{S}_{f,g}$. Thus, it is enough to show that our algorithm preserves invariant (10).

For $k=1$, we need to solve equations $H^{d-1}(g(\mathbf{x})) \equiv H^{d-1}(f(\mathbf{x})) + f_d^{(1)}(\mathbf{b},\mathbf{x})$ and $H^d(g(\mathbf{x})) \equiv H^d(f(\mathbf{x}))$. Notice that equation $H^d(g(\mathbf{x})) \equiv H^d(f(\mathbf{x}))$ is always true, due to the assumptions we are making about our input after preprocessing. Therefore, we will not mention this equation anymore and the only relevant polynomial equation to solve in this case is $H^{d-1}(g(\mathbf{x})) \equiv H^{d-1}(f(\mathbf{x})) + f_d^{(1)}(\mathbf{b},\mathbf{x})$.

By identity (9) we have that $H^{d-1}(g(\mathbf{x})) - H^{d-1}(f(\mathbf{x})) \equiv f_d^{(1)}(\mathbf{c},\mathbf{x})$, which is a directional derivative of f and therefore is an element of $\mathcal{M}_1$. Notice that, for each $\mathbf{a} \in \mathbb{F}^n$,

$$\sum_{j=1}^{n} a_j \cdot \frac{\partial H^d(f(\mathbf{x}))}{\partial x_j} \equiv f_d^{(1)}(\mathbf{a},\mathbf{x}) \in \mathcal{M}_1$$

which implies that we have PIT for linear combinations of the partial derivatives of $H^d(f(\mathbf{x}))$. Thus, by solving the polynomial equation $H^{d-1}(g(\mathbf{x})) - H^{d-1}(f(\mathbf{x})) \equiv f_d^{(1)}(\mathbf{b},\mathbf{x})$ on the variables $\mathbf{b}$, using Lemma 2.4, we get a solution $\mathbf{a}_1$ such that $H^{d-1}(g(\mathbf{x})) \equiv H^{d-1}(f(\mathbf{x})) + f_d^{(1)}(\mathbf{a}_1,\mathbf{x})$

(since we know $\mathcal{S}_{f,g} \neq \emptyset$). Notice that we can use Lemma 2.4 because we have black-box access to all polynomials in the equation. Hence, we have a solution $\mathbf{a}_1$ such that

$$f_d^{(1)}(\mathbf{a}_1, \mathbf{x}) \equiv H^{d-1}(g(\mathbf{x})) - H^{d-1}(f(\mathbf{x})) \equiv f_d^{(1)}(\mathbf{c}, \mathbf{x})$$

and hence, our invariant (10) holds true in the first case.

Now, assume that our invariant is true for $\mathbf{a}_{k-1}$, $k \geq 2$. At the k^{th} iteration, equations (8) are equivalent to

$$\begin{aligned} H^i(g(\mathbf{x})) &= H^i(f(\mathbf{x})) + f_{i+1}^{(1)}(\mathbf{b}, \mathbf{x}) + \sum_{j=i+2}^{d} \frac{1}{(j-i)!} \cdot f_j^{(j-i)}(\mathbf{a}_{k-1}, \mathbf{x}) \\ &\equiv H^i(f(\mathbf{x})) + f_{i+1}^{(1)}(\mathbf{b}, \mathbf{x}) + \sum_{j=i+2}^{d} \frac{1}{(j-i)!} \cdot f_j^{(j-i)}(\mathbf{c}, \mathbf{x}) \qquad \forall d-k \leq i \leq d. \end{aligned}$$

Where in the last equality we used the fact that our invariant holds for $\mathbf{a}_{k-1}$ together with Observation 3.4. These equations, together with equation (9) imply:

$$\begin{aligned} f_{i+1}^{(1)}(\mathbf{b}, \mathbf{x}) &\equiv H^i(g(\mathbf{x})) - H^i(f(\mathbf{x})) - \sum_{j=i+2}^{d} \frac{1}{(j-i)!} \cdot f_j^{(j-i)}(\mathbf{a}_{k-1}, \mathbf{x}) \\ &\equiv f_{i+1}^{(1)}(\mathbf{c}, \mathbf{x}), \quad \text{for all } d-k \leq i \leq d \end{aligned}$$

In other words, for all $d - k \leq i \leq d$

$$\sum_{\ell=1}^{n} b_\ell \cdot \frac{\partial f_{i+1}}{\partial x_\ell}(\mathbf{x}) = f_{i+1}^{(1)}(\mathbf{b}, \mathbf{x}) \equiv f_{i+1}^{(1)}(\mathbf{c}, \mathbf{x})$$

Notice that both sides of each of the equations above belong to the circuit class $\mathcal{M}_1$, as both sides are first-order directional derivatives of $H^{i+1}(f(\mathbf{x}))$. Since we have black-box access to both sides of the equations above, Lemma 2.4 and PIT for $\mathcal{M}_1$ imply that we can solve the system of polynomial equations (8).

Thus, since the invariant is maintained until the end, we must have that $\mathbf{a}_d$ is such that $f_i^{(1)}(\mathbf{a}_d, \mathbf{x}) \equiv f_i^{(1)}(\mathbf{c}, \mathbf{x})$, for all $0 \leq i \leq d$, for some $\mathbf{c} \in \mathcal{S}_{f,g}$. By Lemma 4.7 we must have that $\mathbf{a}_d \in \mathcal{S}_{f,g}$.

Runtime Analysis: Notice that we iterate through the loop d times and at each iteration we solve a linear system of at most $d \cdot |\mathcal{H}_1|$ equations in n variables, where $\mathcal{H}_1 \subset \mathbb{F}$ is a hitting set for the circuit class $\mathcal{M}_1$. After exiting the loop, we only need to perform PIT on $f(\mathbf{x}) - g(\mathbf{x} - \mathbf{a}_d)$, which we assumed it takes polynomial time, for we have PIT for polynomials of the form $f - g$, where $f \in \mathcal{M}_1$ and $g \in \mathcal{M}_2$. Hence, the total running time is polynomial in the size of the input.

□

Analysis in the Randomized Case: The randomized case is analogous to the deterministic black-box case. Whenever we need to perform PIT in our main algorithm, we will use Lemma 1.3. Whenever we need to solve a system of polynomial equations given black box access to the polynomials in question, we will use Lemma 2.5 (when we are allowed randomness), instead of Lemma 2.4 (which handles the case when we have a hitting set).

We need to solve d systems of polynomial equations and we perform the PIT algorithm as in Lemma 1.3 $O(d)$ times. Hence, by setting the error parameter each time as ε/d^2 and by a union bound, our algorithm will err with probability at most ε. Since the amount of randomness that we need to solve a polynomial system or to perform PIT is polynomial in the logarithm of the error parameter, this gives us the desired running time as claimed in Theorem 1.6.

Analysis in the White-Box Case: Notice that by Theorem 3.7 and Corollary 3.8, given access to circuits computing f, g implies that we also have access to circuits computing the polynomials $f_\ell^{(r)}(\mathbf{a}, \mathbf{x})$ and $H^\ell(g)$. Thus, we also have white-box access to linear combinations of $m = \max(n, d)$ of these polynomials.

After the step above, the white-box case is analogous to the deterministic case. Whenever we need to perform PIT in our main algorithm, we will use the appropriate PIT algorithm for the white box class that we are considering. For instance, whenever the algorithm above uses PIT for the class $\mathcal{M}_1$, we will use the white-box algorithm, and the same happens for the other classes. In addition, whenever we need to solve a linear system of polynomial equations, we will use the method in section 2.2 to solve our system. Thus, the same argument as the one given above for the deterministic black-box case will go through, even for the preprocessing stage, and therefore we are done.

6 Conclusion and Open Questions

In this paper, we reduced the problem of shift-equivalence to the problem of solving PIT, and as a consequence of this reduction we obtained a polynomial-time randomized algorithm for the shift-equivalence problem, over characteristic zero or when the characteristic of the base field is larger than the degrees of the polynomials.

We gave some examples for classes of circuits where this can be performed deterministically in quasi-polynomial time. One example where we "almost" have such a result is when testing whether a given sparse polynomial is equivalent to a shift of another sparse polynomial. Note that while the class of sparse polynomials is closed under partial derivatives and homogeneous components, it is not closed under shifts and so we cannot use our approach. Nevertheless, it is quite likely that this simple case can be solved using other techniques.

Acknowledgment

The third author would like to thank Michael Forbes, Ankit Gupta, Elad Haramaty, Swastik Kopparty, Ramprasad Saptharishi and Shubhangi Saraf for helpful discussions on related problems.

References

[Agr05] M. Agrawal. Proving lower bounds via pseudo-random generators. In *Proceedings of the 25th FSTTCS*, volume 3821 of *LNCS*, pages 92–105, 2005.

[AKS04] M. Agrawal, N. Kayal, and N. Saxena. Primes is in P. *Annals of Mathematics*, 160(2):781–793, 2004.

[ASS13] M. Agrawal, C. Saha, and N. Saxena. Quasi-polynomial hitting-set for set-depth-d formulas. In *STOC*, pages 321–330, 2013.

[ASSS12] M. Agrawal, C. Saha, R. Saptharishi, and N. Saxena. Jacobian hits circuits: hitting-sets, lower bounds for depth-d occur-k formulas & depth-3 transcendence degree-k circuits. In *STOC*, pages 599–614, 2012.

[DL78] R. A. DeMillo and R. J. Lipton. A probabilistic remark on algebraic program testing. *Inf. Process. Lett.*, 7(4):193–195, 1978.

[DS06] Z. Dvir and A. Shpilka. Locally decodable codes with 2 queries and polynomial identity testing for depth 3 circuits. *SIAM J. on Computing*, 36(5):1404–1434, 2006.

[DSY09] Z. Dvir, A. Shpilka, and A. Yehudayoff. Hardness-randomness tradeoffs for bounded depth arithmetic circuits. *SIAM J. on Computing*, 39(4):1279–1293, 2009.

[FS12] M. A. Forbes and A. Shpilka. On identity testing of tensors, low-rank recovery and compressed sensing. In *Proceedings of the 44th annual STOC*, pages 163–172, 2012.

[FS13] M. A. Forbes and A. Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. In *Proceedings of the 54th Annual FOCS*, 2013.

[FSS13] M. A. Forbes, R. Saptharishi, and A. Shpilka. Pseudorandomness for multilinear read-once algebraic branching programs, in any order. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:132, 2013.

[GK93] D. Grigoriev and M. Karpinski. A zero-test and an interpolation algorithm for the shifted sparse polynomials. In Grard Cohen, Teo Mora, and Oscar Moreno, editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 673 of *Lecture Notes in Computer Science*, pages 162–169. Springer Berlin Heidelberg, 1993.

[GL95] D. Grigoriev and Y. N. Lakshman. Algorithms for computing sparse shifts for multivariate polynomials. In *Proceedings of the 1995 International Symposium on Symbolic and Algebraic Computation*, ISSAC '95, pages 96–103, New York, NY, USA, 1995. ACM.

[Gri97] D. Grigoriev. Testing shift-equivalence of polynomials by deterministic, probabilistic and quantum machines. *Theoretical Computer Science*, 180(12):217 – 228, 1997.

[HS80] J. Heintz and C. P. Schnorr. Testing polynomials which are easy to compute (extended abstract). In *Proceedings of the 12th annual STOC*, pages 262–272, 1980.

[Kay12] N. Kayal. Affine projections of polynomials: extended abstract. In *Proceedings of the 44th symposium on Theory of Computing*, STOC '12, pages 643–662, New York, NY, USA, 2012. ACM.

[KI04] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[KS01] A. Klivans and D. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the 33rd Annual STOC*, pages 216–223, 2001.

[KS07] N. Kayal and N. Saxena. Polynomial identity testing for depth 3 circuits. *Computational Complexity*, 16(2):115–138, 2007.

[KS09a] Z. S. Karnin and A. Shpilka. Reconstruction of generalized depth-3 arithmetic circuits with bounded top fan-in. In *Proceedings of the 24th Annual CCC*, pages 274–285, 2009.

[KS09b] N. Kayal and S. Saraf. Blackbox polynomial identity testing for depth 3 circuits. In *Proceedings of the 50th Annual FOCS*, pages 198–207, 2009.

[KS11] Z. S. Karnin and A. Shpilka. Black box polynomial identity testing of generalized depth-3 arithmetic circuits with bounded top fan-in. *Combinatorica*, 31(3):333–364, 2011.

[KUW86] R. Karp, E. Upfal, and A. Wigderson. Constructing a perfect matching is in random nc. *Combinatorica*, 6:35–48, 1 1986.

[Lov79] L. Lovasz. On determinants, matchings, and random algorithms. In L. Budach, editor, *Fundamentals of Computing Theory*. Akademia-Verlag, 1979.

[LS94] Y. N. Lakshman and B. D. Saunders. On computing sparse shifts for univariate polynomials. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, ISSAC '94, pages 108–113, New York, NY, USA, 1994. ACM.

[MVV87] K. Mulmuley, U. Vazirani, and V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[RS05] R. Raz and A. Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005.

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.

[Ser03] Á. Seress. *Permutation group algorithms*, volume 152. Cambridge University Press, 2003.

[Shp09] A. Shpilka. Interpolation of depth-3 arithmetic circuits with two multiplication gates. *SIAM J. on Computing*, 38(6):2130–2161, 2009.

[SS10] N. Saxena and C. Seshadhri. From Sylvester-Gallai Configurations to Rank Bounds: Improved Black-Box Identity Test for Deph-3 Circuits. In *Proceedings of the 51st Annual FOCS*, pages 21–30, 2010.

[SS11] N. Saxena and C. Seshadhri. An almost optimal rank bound for depth-3 identities. *SIAM J. Comput.*, 40(1):200–224, 2011.

[SSS13] C. Saha, R. Saptharishi, and N. Saxena. A case of depth-3 identity testing, sparse factorization and duality. *Computational Complexity*, pages 1–31, 2013.

[Str73] V. Strassen. Vermeidung von divisionen. *J. of Reine Angew. Math.*, 264:182–202, 1973.

[SY10] A. Shpilka and A. Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3–4):207–388, 2010.

[Zip79] R. Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSAM*, pages 216–226, 1979.