Progress on Polynomial Identity Testing

Nitin Saxena*

Abstract

Polynomial identity testing (PIT) is the problem of checking whether a given arithmetic circuit is the zero circuit. PIT ranks as one of the most important open problems in the intersection of algebra and computational complexity. In the last few years, there has been an impressive progress on this problem but a complete solution might take a while. In this article we give a soft survey exhibiting the ideas that have been useful.

1 Introduction

One learns a number of *identities* as part of Algebra in the school curriculum. For example, the *difference of squares* identity $(x+y)(x-y) = (x^2-y^2)$ or a more impressive *sum of four squares* identity (probably first communicated by Euler in a letter to Goldbach on May 4, 1748): $(a_1^2+a_2^2+a_3^2+a_4^2)(b_1^2+b_2^2+b_3^2+b_4^2) = (a_1b_1-a_2b_2-a_3b_3-a_4b_4)^2 + (a_1b_2+a_2b_1+a_3b_4-a_4b_3)^2 + (a_1b_3-a_2b_4+a_3b_1+a_4b_2)^2 + (a_1b_4+a_2b_3-a_3b_2+a_4b_1)^2$. There is of course an easy way to test them: just completely expand the products and check whether the monomials cancel in the resulting sum. This way you can easily verify that the above two expressions are indeed identities. But the process of *expanding products* blows up monomials and would be very expensive when both the number of variables and the degree of the expression are increased. Roughly, for n variables and d degree the number of monomials grows as $\binom{n+d}{d}$ which is small if one of n or d is small but *exponentially* large if both n, d are large. So the question we ask is - can this identity or zero testing be done in $(nd)^{O(1)}$ steps?

Notice that to formalize this question there seems to be a need of carefully defining the way the algebraic expression is given to us in the input. Fortunately, there is already an object defined in computational complexity,

---

*Hausdorff Center for Mathematics, Endenicher Allee 62, 53115 Bonn, Germany, ns@hcm.uni-bonn.de

called *arithmetic circuit*, that we could directly use [?]. An arithmetic circuit C, on say n variables and a field $\mathbb{F}$, is a directed acyclic graph with *input* variables at the leaves and *output* at the root. The internal nodes are called *gates*, they are of two kinds - multiplication and addition - and perform the respective operations over the field $\mathbb{F}$. The edges or *wires* of C can have *constants* on them from the field which get multiplied to the value at the tail of the respective edge. It is easy to see that the value at the root of the circuit C is just an n-variate polynomial and lives in $\mathbb{F}[x_1, \ldots, x_n]$. Thus a circuit is a natural combinatorial way to capture algebraic computation. One might want to consider the base object $\mathbb{F}$ to be a general *ring* instead of a field but we will be more concerned with the latter in this survey. For the purposes of identity testing we usually regard the operations in the field $\mathbb{F}$ to be doable in *unit* time. The bulk of the computation in the identity testing algorithms is seen as a function of the *size* of the input circuit, which is basically the number of gates and wires in the circuit. Another useful parameter of a circuit is *depth*, which is the number of levels between the root and the leaves. *Fanin/Fanout* refers to the maximum number of inputs/outputs a gate has in the circuit, and a circuit with fanout 1 is called a *formula*. Finally, we use the notation $poly(s, t)$ to denote a positive-valued function whose asymptotic behaviour is $(s + t)^{O(1)}$. The problem of identity testing is then:

Problem 1.1 (PIT). Given an arithmetic circuit C in the input that computes a polynomial $p(x_1, \ldots, x_n)$ in $\mathbb{F}[x_1, \ldots, x_n]$. Find a deterministic algorithm that tests if p is the zero polynomial, and uses only $poly(size(C))$ many $\mathbb{F}$ operations.

PIT is currently an open question and, as we will see in this survey, an important question in complexity theory. But it has an easy "practical" solution, i.e. there are *randomized* polynomial-time algorithms that are easy to implement. The first randomized polynomial time algorithm was given (independently) by Schwartz [?] and Zippel [?]. It simply evaluates the input circuit at a randomly chosen point in $\mathbb{F}^n$ and outputs YES iff the specific evaluation is zero. This idea works mainly because a nonzero polynomial cannot have "too many" roots over a field:

Lemma 1.2 (Schwartz-Zippel). *Let $P \in F[x_1, x_2, \ldots, x_n]$ be a non-zero polynomial of degree $d \geq 0$ over a field $\mathbb{F}$. Let S be a finite subset of $\mathbb{F}$. Then,*

$$Prob_{r_1,\ldots,r_n \in S}\left[P(r_1, \ldots, r_n) = 0\right] \leq \frac{d}{|S|}.$$

Proof. The proof is by induction on n. For $n = 1$, P can have at most d roots and hence the probability of hitting a root is at most $\frac{d}{|S|}$.

Now, assume that the statement holds for all polynomials upto $(n-1)$ variables. Wlog we can then consider P to be a polynomial in x_1 by writing it as,

$$P(x_1, \ldots, x_n) = \sum_{i=0}^{d} x_1^i P_i(x_2, \ldots, x_n).$$

Since P is a nonzero polynomial, $\exists i$ such that P_i is nonzero. Take the largest such i, clearly $degP_i \le (d-i)$. Now we randomly pick $r_2, \ldots, r_n$ from S. By the induction hypothesis, $Prob[P_i(r_2, \ldots, r_n) = 0] \le \frac{d-i}{|S|}$. If $P_i(r_2, \ldots, r_n) \ne 0$ then $P(x_1, r_2, \ldots, r_n)$ is of degree i so by the univariate case:

$$Prob[P(r_1, \ldots, r_n) = 0 | P_i(r_2, \ldots, r_n) \ne 0] \le \frac{i}{|S|}$$

By a lazy probability estimation we get:

$$\begin{aligned} Prob_{r_1,\ldots,r_n \in S}\left[P(r_1, \ldots, r_n) = 0\right] &\le Prob[P_i(r_2, \ldots, r_n) = 0] + \\ &\qquad Prob[P(r_1, \ldots, r_n) = 0 | P_i(r_2, \ldots, r_n) \ne 0] \\ &\le \frac{d-i}{|S|} + \frac{i}{|S|} \\ &\le \frac{d}{|S|} \end{aligned}$$

Thus completing the proof by induction. □

This lemma shows that as long as the field $\mathbb{F}$ has twice as many elements as the degree of the input circuit, we have a good randomized algorithm that has error probability at most $\frac{1}{2}$. In case $\mathbb{F}$ is too small compared to the degree d of the input circuit C we go to a suitable *extension* of $\mathbb{F}$ and pick random points there. The two issues here that are worth mentioning: (1) The degree of the input polynomial can only be at most $2^{size(C)}$. (2) A field extension of $\mathbb{F}$ of degree $O(size(C))$ can either be found by a *deterministic* construction of irreducible polynomials [?] or we can simply work over the *cyclotomic* extension $\mathbb{F}[z]/(z^r - 1)$ for a "suitable" $r = poly(size(C))$. Finally, the actual evaluation of C at a randomly chosen point (even from the extension algebra) can be trivially simulated in $poly(size(C))$ many $\mathbb{F}$ operations.

Randomized algorithms that use fewer random bits and have lower error probability (at the cost of time) were given by Chen & Kao [?], Lewin & Vadhan [?], and Agrawal & Biswas [?]. As we care more about deterministic methods in this survey, we will not discuss the details of these methods here. These randomized algorithms show that PIT is in the complexity class BPP which in turn is conjectured to be equal to P (see the survey [?] for the theoretical evidence). Thus it seems to be a reasonable goal to *derandomize* PIT.

Some applications of PIT

Being a fundamental problem it is not surprising that PIT appears in several other seemingly unrelated problems. We see below an example each from complexity theory, graph theory and number theory.

The idea of comparing two multivariate polynomials for *equality* by evaluating them at randomly chosen points was crucial in the proof of the complexity result: IP=PSPACE [**?**]. The multivariate polynomial in that case is the arithmetized version of a quantified boolean formula (QBF) ϕ, and using PIT it becomes possible to give an interactive protocol (IP) to verify the truth of ϕ. Here PIT helped in upper bounding the complexity of QBF problem, on the other hand, PIT also has several lower bound implications (see Section 7)

Another, much older, application of PIT is due to the following theorem proved by Tutte [**?**] in 1947: *A graph has no perfect matching iff the determinant of its Tutte matrix is zero.* Recall that for a graph $G = (V, E)$ on n vertices its *Tutte matrix* is an $n \times n$ matrix A with its (i, j)th entry defined as:

$$A_{i,j} := \begin{cases} x_{i,j}, \text{ if } (i,j) \in E \text{ and } i < j \\ -x_{j,i}, \text{ if } (i,j) \in E \text{ and } i > j \\ 0, \text{ otherwise} \end{cases}$$

To use this theorem in a matching algorithm we will have to check whether the multivariate polynomial $det(A)$ is zero, which can be seen as a special case of PIT. This formulation immediately gives a randomized algorithm which has the added advantage of being in (randomized) NC, i.e. it is a highly *parallel* algorithm since determinant has known fast parallel algorithms. It is an open question to find a deterministic parallel algorithm for perfect matching, and it appears that a derandomization of this special case of PIT might be the way to go (see a related conjecture in [**?**] and a special case in [**?**]).

Finally, the problem of *primality testing* was solved in an elementary way by working with a PIT formulation. It was observed by Agrawal & Biswas [**?**] that a positive integer n is prime iff $(x + 1)^n = (x^n + 1) \pmod{n}$, and they exploited this simple binomial fact to design a new randomized primality test. If we define $P(x) := (x + 1)^n - (x^n + 1)$ then the question is that of testing whether $P(x)$ is the zero polynomial over the ring $\mathbb{Z}/n\mathbb{Z}$, which is just a special case of PIT. Note that although $P(x)$ is a univariate polynomial it has degree n which is *exponential* in the input size $\log n$, and so we cannot afford to completely expand $P(x)$. The neat idea in [**?**] was to test $P(x) = 0 \pmod{n, Q(x)}$ for a randomly chosen polynomial Q of degree

$O(\log n)$. As Q has "small" degree we can do this in $poly(\log n)$ time, using *repeated squaring* of $(x+1)$ and x. This randomized algorithm was later derandomized by Agrawal, Kayal & Saxena [?] to get the first deterministic polynomial time algorithm for primality testing. They essentially showed that if $P(x) = 0 \pmod{n, a^r x^r - 1}$ for all $1 \leq a, r \leq (\log n)^5$ then $P(x) = 0 \pmod{n}$. It is astonishing that the zeroness of a polynomial of a high degree can be determined by just looking modulo very few, very small polynomials!

Survey Overview

The goal of this survey is not to be exhaustive but to cover the main ideas and to pose the closely related open questions. One of the interesting topics related to PIT which we would not be discussing in this survey are: PIT for circuits over *general* rings (see [?]), *interpolation* of polynomials (see [?, ?, ?]) and *learning* arithmetic formulas (see [?, ?]). A brief overview of the topics that we do cover in this survey now follows.

Sparse PIT. A circuit C that computes a polynomial which has at most m nonzero monomials is called *m-sparse*. The problem of sparse PIT is to design an algorithm that runs in $poly(size(C), m)$ field operations. This problem has several known solutions (see [?]). We will see the one by Agrawal [?] as I find it the simplest conceptually.

Low Degree PIT. A circuit $C(x_1, \ldots, x_n)$ that computes a polynomial of degree $poly(n)$ is called a *low degree* circuit. The term low degree is used to contrast with circuits that use repeated squaring to exponentially increase the degree, for example the circuit $P(x)$ that appears in the primality test above is *not* low degree. It can be seen that formulas, circuits of constant depth, and bounded fanin circuits of $O(\log n)$ depth; all compute a low degree polynomial. Thus, it seems natural to study the problem of PIT for low degree circuits and we call it *low degree PIT*.

It was shown by Agrawal & Vinay [?] that for the purposes of low degree PIT it is enough, somewhat surprisingly, to just consider depth-4 circuits. The main idea is to "shrink" any low degree circuit into a depth-4 circuit by paying only a *sub*exponential price in the circuit size. Thus if one solves depth-4 PIT in deterministic polynomial time then one has solved low degree PIT in subexponential time. I mainly see this as a strong evidence that PIT for "shallow" circuits, i.e. those of depth 3 or 4, gives us enough clues to tackle the bigger PIT problem.

Depth-3 PIT (Non Black-Box). Convinced that shallow circuits are already interesting cases for PIT, we now focus on the PIT algorithms for depths $2, 3$ and 4. A depth 2 circuit is either a sum of monomials ($\Sigma\Pi$) or a product of linear polynomials ($\Pi\Sigma$), both of which have obvious deterministic polynomial time PIT algorithms if we can look "inside" the circuit (which is why we use the term *non* black-box). A depth 3 circuit can either be a product of sum of monomials or a sum of product of linear polynomials. As the former case is again trivial to check for zeroness, we only worry about the latter case. Thus, for us a *depth*-3 *circuit* C over a field $\mathbb{F}$ is $C(x_1, \ldots, x_n) = \sum_{i=1}^{k} T_i$, where T_i (*a multiplication term*) is a product of d_i linear polynomials $L_{i,j}$ over $\mathbb{F}$. Note that by homogenization we can assume wlog that $L_{i,j}$'s are linear *forms* (i.e. linear polynomials with a zero constant coefficient) and that $d_1 = \cdots = d_k =: d$. Such a circuit is referred to as a $\Sigma\Pi\Sigma(n, k, d)$ circuit, where k is the *top fanin* of C and d is the *degree* of C. Depth-3 circuits are a good starting point and are under intense study from various viewpoints [?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?].

It was shown by Kayal & Saxena [?] that if the top fanin k is small then PIT is easy. Note that $k = 2$ is the trivial case (because $\mathbb{F}[x_1, \ldots, x_n]$ is a unique factorization domain) but $k = 3$ is already nontrivial. The main idea of [?] was to look at C modulo several linear forms and use a generalized form of Chinese remaindering. The cost of doing this grows like d^k and hence is meaningful when k is small even if d, n are arbitrarily large. As one needs to look "inside" the circuit this algorithm we label as non black-box.

Depth-3 PIT (Black-Box). One might wish to develop algorithms for PIT that do not even *look* inside a given circuit C, but merely *evaluate* C at several points in $\mathbb{F}$ or algebraic extensions of $\mathbb{F}$. Of course this is an impossible dream if we do not have an *a priori* bound on $size(C)$. But with such a bound given in the input together with a *black-box* access to C, the question of testing $C = 0$ in $poly(size(C))$ many $\mathbb{F}$ operations becomes reasonable, as the randomized Schwartz-Zippel PIT algorithm does not look inside the circuit at all! Intuitively it seems that to devise a black-box PIT algorithm for a circuit family, one would need a very good understanding about the structure of identities in that family. There is some progress in that direction for $\Sigma\Pi\Sigma(n, k, d)$ identities with constant top fanin k.

Note that a $\Sigma\Pi\Sigma(n, k, d)$ identity C is composed of linear forms and hence we can associate a natural notion of *rank*, which will be the rank of the vector space that these linear forms span. It was first shown by Dvir & Shpilka [?] that, under some mild assumptions on C, the rank of C is bounded by $\log^k d$. For a constant k this is saying something nontrivial about the $\Sigma\Pi\Sigma(n, k, d)$ identities. Later Karnin & Shpilka [?] used this property to develop a black-box PIT algorithm for $\Sigma\Pi\Sigma(n, k, d)$ circuits that runs in time $\sim d^{rank(C)}$, or $d^{\log^k d}$ which is a subexponential complexity when k is constant. This connection between rank and black-box PIT is quite encouraging, and has already led to several improvements. Saxena & Seshadhri [?] showed a rank bound of $k^3 \log d$ which is almost optimal and translates into an improved black-box PIT algorithm of complexity $d^{k^3 \log d}$. Their main idea was to look at C modulo various *ideals* and deduce lots of dependencies between the various multiplication terms of the identity C.

It is believed that over the fields of zero characteristic, especially complex numbers, the identities should be even more restricted. Towards that goal, Kayal & Saraf [?] showed a rank bound of k^k over the field of *reals*. It gives a corresponding black-box PIT algorithm of complexity d^{k^k}, which is polynomial time for constant k. Their main idea is to look at the linear forms appearing in C as points in a higher dimensional space and then use certain properties of real geometry [?] to rule out their arrangement in a $\Sigma\Pi\Sigma(n, k, d)$ identity.

Depth-4 PIT. PIT algorithms for circuits of depth higher than 3 are currently few [?, ?, ?, ?, ?]. The ones that are known are based on insights obtained from depth-2 and depth-3 circuits and put further restrictions so that those ideas could be lifted to higher depths. We will discuss the PIT algorithms for noncommutative formulas [?] and depth-4 circuits with multiplication gates that only do powering [?].

A *noncommutative* formula is one that has noncommuting variables, i.e. $x_i x_j \neq x_j x_i$ for all $i \neq j \in [n]$. The main idea of Raz & Shpilka [?] was that a multiplication gate in a noncommutative formula can be gradually *opened-up* without getting into the problem of monomial explosion. They then used linear algebra to complete the PIT algorithm.

Saxena [?] solved the case of depth-4 circuits when each multiplication gate is just *powering*, i.e. an input $p(x_1, \ldots, x_n)$ is converted to $\alpha \cdot p(x_1, \ldots, x_n)^e$ for some $\alpha \in \mathbb{F}$ and $e \in \mathbb{N}$. The main idea was to transform such a circuit to another one wherein each multiplication gate has factors with *unmixed* variables. The PIT algorithm then follows an algebraic generalization of the idea of [?].

General PIT and Lower Bounds. As seen above PIT is a fascinating fundamental problem with direct connections to other problems. As if this was not enough, Kabanets & Impagliazzo [?] further emphasized the importance of PIT by showing that a complete solution of PIT would imply circuit lower bounds. They showed that if PIT is in P then *either* Permanent (the naughtier sibling of Determinant) does not have polynomial sized arithmetic circuits *or* NEXP does not have polynomial sized boolean circuits. Even though we “believe” both the conclusions to be independently true, nevertheless, the connection with PIT is intriguing. Their main idea is to show that low-degree-PIT$\in$P together with the existence of small circuits for permanent and those for NEXP, implies NEXP$\subseteq$NP, which is a contradiction. In the proof PIT is only used to test whether a small arithmetic circuit equals the permanent function.

In a more explicit way, Agrawal [?] showed that if there are *black-box* PIT algorithms for a circuit family then they also exhibit lower bounds for that family.

2 Sparse PIT

A lot of papers have focused on the case of circuits that compute a sparse polynomial. In this case we are given a circuit C together with an upper bound m on the number of nonzero monomials in the computed polynomial, and the goal is to devise a $poly(size(C), m)$ time PIT algorithm. Notice that this is a "benign" goal as usually in PIT a given circuit would produce an exponential (in $size(C)$) number of nonzero monomials.

There exist a host of solutions for this case and also for the seemingly more general problem of *interpolating* such circuits [?, ?, ?, ?, ?, ?, ?]. All these algorithms are based on the idea of evaluating the given circuit at cleverly chosen points so that a specific nonzero monomial gets *isolated*. Since there are "few" nonzero monomials one of them can be efficiently isolated by just doing evaluations, hence these tend to be *black-box* algorithms. We exhibit one such algorithm, following Agrawal [?]. The basic idea is to go to a cyclotomic extension and evaluate the circuit at the *virtual* roots of unity available in this extension algebra.

Theorem 2.1. *Let $p(x_1, \ldots, x_n)$ be a nonzero polynomial (over a field $\mathbb{F}$) whose degree in each variable is less than d and the number of monomials is at most m. Then there exists an $1 \leq r \leq (mn \lg d)^2$ such that, $p(y, y^d, \ldots, y^{d^{n-1}}) \neq 0 \pmod{y^r - 1}$.*

Proof. Consider the polynomial $q(y) := p(y, y^d, \ldots, y^{d^{n-1}})$ in $\mathbb{F}[y]$. Note that a monomial $x_1^{i_1} \cdots x_n^{i_n}$ in p is mapped to the monomial $y^{i_1 + i_2 d + \cdots + i_n d^{n-1}}$ in q. Since we have assumed $i_1, \ldots, i_n < d$, observe that the map is one-to-one. Consequently, $q(y) \neq 0$. Say y^a is a monomial with nonzero coefficient in q. Now we look at $q(y)$ modulo $(y^r - 1)$.

If $q(y) = 0 \pmod{y^r - 1}$, then there ought to be another monomial $y^b \neq y^a$ with nonzero coefficients in $q(y)$ such that $y^b = y^a \pmod{y^r - 1}$. This is possible iff $r|(b-a)$. Thus, to avoid picking such a "bad" r we need one that satisfies:

$$r \nmid \prod_{y^b \in q(y), b \neq a} (b - a) =: R.$$

Clearly, integer R can be at most $(d^n)^m$ in value. Since R has at most $\lg R$ prime factors and since we would encounter at least $(\lg R + 1)$ primes in the range $1 < r \leq (\lg R)^2 = (mn \lg d)^2$, it is clear that we have the required (prime) r for which $q(y) \neq 0 \pmod{y^r - 1}$. $\square$

Algorithm. The above property immediately gives a black-box PIT algorithm for sparse polynomials. Given an m-sparse circuit $C(x_1, \ldots, x_n)$ over $\mathbb{F}$, fix $d := 2^{size(C)}$ and for every $1 \leq r \leq (mn \lg d)^2$: compute $d, d^2, \ldots, d^{n-1}$ modulo r using repeated squaring and then evaluate $C(y, y^d, \ldots, y^{d^{n-1}})$ over the extension algebra $\mathbb{F}[y]/(y^r - 1)$. Finally, we declare C to be an identity iff all these evaluations are zero. It is routine to verify that this is a correct algorithm with time complexity $poly(size(C), m)$.

Open. One might wonder what happens if we replace the parameter $(mn \lg d)^2$ in the above analysis by a milder parameter like $poly(size(C))$? If we could still prove the statement in Theorem 2.1 then we would get a conceptually simple black-box algorithm for general PIT! Such a generalization of Theorem 2.1 is currently an open question, even for the "smallest" case of depth-3 circuits. (Note that the theorem trivially applies to the case of depth-2 circuits.) It is conjectured by Agrawal [?] that Theorem 2.1 should be true if we replace $(mn \lg d)^2$ by $size(C)^{depth(C)}$, thus, solving PIT at least for *constant* depth circuits.

3 Low Degree PIT

The circuit model is a very *expressive* representation for polynomials, for example, in size s it is possible to achieve degree 2^s by repeated squaring (although the number of monomials produced remains *singly*-exponential in s and not *doubly*-exponential). What if we reduce the expressive nature of a circuit, say, by restricting the degree of the computed polynomial to be "only" $poly(s)$? Intuitively, PIT for these circuits should be easier.

It was shown by Agrawal & Vinay [?] that a *low degree* circuit $C(x_1, \ldots, x_n)$, i.e. of degree $poly(n)$, can be shrunk to a depth-4 circuit by a reasonable blowup in the size. We will now see the main idea of the proof. As we can always add some useless variables to C, we can assume wlog that $C(x_1, \ldots, x_n)$ is computing a polynomial of degree $d = O(n)$. Furthermore, as any n variate, d degree polynomial can be trivially computed by a depth-2 circuit of size $\sim \binom{n+d}{d} = 2^{d \lg \frac{n}{d}}$, it is only reasonable to assume that C has size $2^{o(d \lg \frac{n}{d})}$ (note the *small o* in the exponent). In that case the theorem of [?] states:

Theorem 3.1. *If a polynomial $P(x_1, \ldots, x_n)$ of degree $d = O(n)$ has a circuit C of size $2^{o(d \lg \frac{n}{d})}$ then there is a depth-4 circuit C' of size $2^{o(d \lg \frac{n}{d})}$. Moreover, it can be explicitly constructed in $2^{o(d \lg \frac{n}{d})}$ time, given C in the input.*

Proof Sketch: The depth reduction is done in two stages. The first stage reduces the depth to $O(\lg d)$ by an efficient construction of Allender, Jiao, Mahajan and Vinay [**?**]. The second stage is more expensive but it reduces the depth to 4.

The main idea in the first stage is to look at certain *intermediate* polynomials $[g, h]$ computed inside the circuit C: for any gate g in C and any gate h in the subtree rooted at g, $[g, h]$ is defined to be the polynomial computed at the node g if the subtree at h is *replaced* by a leaf labelled 1. This immediately gives us the simple relation: $C(x_1, \ldots, x_n) = \sum_i [root(C), x_i] x_i$. Next the polynomial $[g, h]$ is recursively expanded wrt the multiplication gates p in the subtree (rooted at g) for which the degree $\geq \frac{1}{2} deg(gh) >$ degree of the children of p. This expansion is then used to construct a circuit C'' whose gates correspond to $[g, h]$, for "all" gates g, h in C. A clever argument in [**?**] shows that every multiplication gate in C'' has at least a doubling effect on the degree of its children, hence, the depth of C'' can be at most $O(\lg d)$. Also, $size(C'')$ remains at most a polynomial in $size(C)$.

Let s be the size of C'' and define a parameter ℓ sufficiently smaller than $\frac{d \lg \frac{n}{d}}{\lg s}$. The second stage has an even simpler idea: cut C'' into two parts, the *top* has exactly $t := \lg \ell$ layers of multiplication gates and the rest of the layers form the *bottom*. Let $g_1, \ldots, g_k$ (where $k \leq s$) be the output gates of the bottom part. Thus, we can think of the top part as computing a polynomial P_{top} in new variables $y_1, \ldots, y_k$ and each of the g_i computing a polynomial P_i in the input variables $x_1, \ldots, x_n$. The polynomial computed by the circuit C'' then equals: $P_{top}(P_1(x_1, \ldots, x_n), \ldots, P_k(x_1, \ldots, x_n))$. Since the top half consists of t levels of multiplication gates $deg(P_{top})$ is bounded by 2^t. And since the degree drops by a factor of two across multiplication gates, we also have $deg(P_i) \leq \frac{d}{2^t}$. Expressing P_{top} and P_i's as a sum of product, we have a depth-4 circuit C' computing the same polynomial as C''. The size of this circuit C' is:

$$\sim \binom{k + 2^t}{k} + k \cdot \binom{n + \frac{d}{2^t}}{n}$$

An easy calculation shows that the dominating terms above are: $s^\ell + (n\ell)^{\frac{d}{\ell}}$. Both of which, by the choice of ℓ, are smaller than $2^{o(d \lg \frac{n}{d})}$. This completes the proof. $\square$

This theorem already suggests that a PIT algorithm for depth-4 circuits would imply a nontrivial one for low degree circuits. [**?**] goes a step further and shows that a black-box polynomial time algorithm for depth-4 PIT gives an $n^{\lg n}$ time algorithm for low degree PIT!

Open. The above proof has an interesting byproduct: if we could prove an exponential lower bound for a low degree polynomial for depth-4 circuits then it implies an exponential lower bound for general circuits! For example, we know that permanent (on $n \times n$ matrices) has a depth-4 circuit of size $2^{O(n)}$. But whether it has depth-4 circuits of size $2^{o(n)}$ is not known. Such a lower bound would now imply that permanent does not have (general) arithmetic circuits of size $2^{o(\sqrt{n})}$.

4 Depth-3 PIT (Non Black-Box)

The case of depth-2 being too easy (it has a black-box polynomial time PIT algorithm) and that of depth-4 being too general (its PIT algorithm will also give a nontrivial one for low degree circuits), leaves us with the intermediate case of depth-3 PIT. There are a host of results for it but the case is still not completely solved. Here we will sketch the idea of the best known PIT algorithm [**?**]. It is a non black-box algorithm as it needs to look at the input circuit to use the linear polynomials that occur in it.

Let the input circuit C be computing over a field $\mathbb{F}$. As discussed before we can assume wlog that C looks like: $C(x_1, \ldots, x_n) = T_1 + \cdots + T_k$, where T_i is a product of linear polynomials $L_{i,1}, \ldots, L_{i,d}$, i.e. each $L_{i,j} = (a_{i,j,0} + a_{i,j,1}x_1 + \cdots + a_{i,j,n}x_n)$ for some constant a's from $\mathbb{F}$. Note that the case of $k = 2$ is trivial as checking $T_1 + T_2 = 0$ entails comparing the linear factors of T_1 and T_2, which we know explicitly. Thus, $k = 3$ is the first bonafide case and indeed therein lies the main idea of [**?**]. So we sketch the algorithm only for the case $C = T_1 + T_2 + T_3$.

Chinese Remaindering. The starting idea is to study C modulo linear polynomials. So pick $(d+1)$ *coprime* linear polynomials $p_1, \ldots, p_{d+1}$ from the set $\{L_{i,j} \mid i \in [3], j \in [d]\}$. Note that by elementary algebra, $C = 0$ iff for all $i \in [d+1]$, $C = 0 \pmod{p_i}$. The latter conditions are easy to check because C modulo p_i is just a sum of two multiplication gates, say $C = T_1 + T_2 \pmod{p_i}$. Now we can further simplify the situation by mapping $p_i \mapsto x_1$ by applying a suitable invertible linear transformation τ on $x_1, \ldots, x_n$ (i.e. it replaces x_i by a linear combination of the x's). It is easy to see that $C = 0 \pmod{p_i}$ iff $C(\tau(x_1), \ldots, \tau(x_n)) = 0 \pmod{x_1}$. The latter can be tested by simply comparing the linear factors of $\tau(T_1)$ and $\tau(T_2)$ after fixing $x_1 = 0$ in them.

Thus, zero testing of C just boils down to picking the right set of linear polynomials amongst the ones that appear in the definition of C. But the above idea fails if the set $\{L_{i,j} \mid i \in [3], j \in [d]\}$ does not have $(d+1)$ *coprime* linear polynomials. This could easily happen, for example when $C = x_1^9 + x_2^5 x_3^4 - (x_1 + x_2 + x_3)^9$ we have only 4 coprime linear polynomials while we need 10. In that case we require more algebra:

Chinese Remaindering over Ideals. The idea that finally works is to study C modulo "nice looking" *ideals.* Formally, pick coprime linear polynomials $p_1, \ldots, p_\ell$ from the set $\{L_{i,j} \mid i \in [3], j \in [d]\}$ such that there exist exponents $e_1, \ldots, e_\ell$ satisfying:

1) every $p_i^{e_i}$ divides some T_j.

2) $e_1 + \cdots + e_\ell > d$.

A simple calculation shows that such *powers of linear polynomials* $p_1^{e_1}, \ldots, p_\ell^{e_\ell}$ always exist (unless the multiplication terms in C are not distinct). Also $C = 0$ iff for all $i \in [\ell]$, $C = 0 \pmod{p_i^{e_i}}$. But how do we check the latter condition? We can again first simplify it to $p_i \mapsto x_1$ by applying an invertible linear map τ on $x_1, \ldots, x_n$. Then $C = 0 \pmod{p_i^{e_i}}$ iff $C(\tau(x_1), \ldots, \tau(x_n))$ vanishes over the algebra $\mathbb{F}[x_1]/(x_1^{e_i})$. Since $\tau(C) \pmod{x_1^{e_i}}$ is a sum of two multiplication gates, say $\tau(T_1) + \tau(T_2)$, testing $\tau(T_1) + \tau(T_2) = 0$ modulo the ideal $(x_1^{e_i})$ boils down to a more sophisticated comparison of the linear factors of $\tau(T_1)$ and $\tau(T_2)$.

Algorithm & Complexity. The final PIT algorithm for a $\Sigma\Pi\Sigma(n,k,d)$ circuit $C = T_1 + \cdots + T_k$ thus identifies a set $\mathcal{I}$ of nice looking ideals, namely,

$$\begin{aligned}\mathcal{I} := \{(f_1, \ldots, f_\ell) \mid \ell \in [k-1], \forall i \in [\ell], & f_i \text{ is a maximal factor of some } T_j \text{ s.t.} \\ & f_i \text{ is not a zero-divisor modulo } (0, f_1, \ldots, f_{i-1}) \text{ and} \\ & f_i \text{ is power of a linear polynomial modulo the radical of } (0, f_1, \ldots, f_{i-1})\}.\end{aligned}$$

Actually, the ideals useful for the algorithm of [?] are a subset of $\mathcal{I}$ but this set captures the most important properties of the ideals. The PIT algorithm just checks $C = 0 \pmod{I}$, for every $I \in \mathcal{I}$, and that implies the zeroness of C. The test $C = 0 \pmod{I}$ is easy to do because $C \pmod{I}$ is basically just *one* multiplication gate. Moreover, since $|\mathcal{I}| < d^k$ and the dimension of the factor-algebra of any ideal in $\mathcal{I}$ is also at most d^k, the algorithm has time complexity $poly(n, d^k)$.

5 Depth-3 PIT (Black-Box)

The depth-3 PIT algorithm seen above is inherently non black-box as it uses the linear polynomials that define the input circuit. It is more desirable to have a black-box PIT algorithm as it will imply circuit lower bounds (see [?, ?]) and it tends to be useful in learning algorithms [?, ?]. Note that the randomized PIT algorithm based on Schwartz-Zippel lemma is black-box, hence, by the conditional derandomizations of BPP [?] such a deterministic polynomial-time black-box PIT algorithm is conjectured to exist.

The black-box version of depth-3 PIT is the one in which we are given a black-box $C(x_1, \ldots, x_n)$ with the promise that C computes a depth-3 circuit over a known field $\mathbb{F}$ and has a known size bound $size(C)$. This question has received a fair amount of attention [?, ?, ?, ?] but is not yet completely solved. The known black-box methods are successful, to a varying degree, only in the case of $\Sigma\Pi\Sigma(n, k, d)$ circuits when the top fanin k is "small". We discuss in this survey techniques that are based on the notion of *rank* of a depth-3 circuit, i.e. the dimension of the vector space spanned by the linear polynomials that appear in its multiplication terms. For a $\Sigma\Pi\Sigma(n, k, d)$ identity a trivial bound on the rank is kd. It is not immediately clear whether the rank of an identity should be significantly smaller than kd, but it is and this property helps in developing the black-box PIT algorithms. To show the rank bounds we need to put certain "mild" conditions on the circuits:

Definition 5.1. (Minimal and Simple circuits) A $\Sigma\Pi\Sigma(n, k, d)$ circuit $C = T_1 + \cdots + T_k$ is said to be *minimal* if no proper subset of $\{T_i\}_{1 \leq i \leq k}$ sums to zero.

The circuit is said to be *simple* if there is no non-trivial common factor dividing all the T_i's.

It was first shown by Dvir & Shpilka [?] that a minimal, simple $\Sigma\Pi\Sigma(n, k, d)$ identity has rank at most $\log^k d$. If k is small then this is a much better bound than the trivial bound of kd. For larger k's this bound is an overkill, and was improved to a more optimal-looking $k^3 \log d$ by Saxena & Seshadhri [?]. We will sketch the idea of the latter rank bound but first we discuss how a rank bound implies a *black-box* PIT algorithm.

5.1 Rank Bounds entail Black-box PIT

Karnin & Shpilka [?] showed that if we have a rank bound of $R(k, d)$ for minimal, simple $\Sigma\Pi\Sigma(n, k, d)$ identities then black-box PIT can be done in $poly(n, d^{R(k,d)})$ many field operations. Their idea was to come up with a small set of linear transformations such that: (1) for each non-zero $\Sigma\Pi\Sigma(n, k, d)$ circuit, at least one of the linear transformations continues to keep it non-zero, and (2) these linear transformations map the n variables to $R(k, d)$ variables. It is easy to see that once we have such a linear transformation τ, we can *compose* it with the given black-box for C to get a new black-box computing $C'(x_1, \ldots, x_m) := C(\tau(x_1), \ldots, \tau(x_n))$. Since C' now has "fewer" variables ($m = R(k,d)$) and is still of degree at most d, we could just apply a brute-force version of Schwartz-Zippel to test it for zeroness. This entails evaluating C' on $(d+1)^m$ points in $\mathbb{F}^m$ (an extension of it, if required), hence it gives an overall complexity of $poly(n, d^{R(k,d)})$.

These linear transformations τ are inspired from the *Fourier transform* matrix and they preserve the rank of arbitrary subspaces. The following lemma by Gabizon & Raz [?] (which they used to construct *extractors* for affine sources) tells us how to identify such transformations.

Lemma 5.2. *Let $W_1, \ldots, W_s \subseteq \mathbb{F}^n$ be fixed subspaces, each of dimension at most t. Consider the linear transformation (for some $\alpha \in \mathbb{F}$),*

$$\phi_{\alpha,t,n}(x_1, \ldots, x_n) := ((\alpha^{i(j-1)}))_{1 \le i \le t, 1 \le j \le n} \cdot [x_1 \cdots x_n]^T.$$

Then there are at most snt^2 elements $\alpha \in \mathbb{F}$ for which the dimension of some W_i drops, i.e. $dim(\phi_{\alpha,t,n}(W_i)) < dim(W_i)$.

Proof. The proof idea is to capture all the "bad" α's in a univariate equation, then its degree upper bounds their number.

Whenever the dimension of the image of W_i under $\phi_{\alpha,t,n}$ drops, it means that the top-left $dim(W_i) \times dim(W_i)$ submatrix of (the matrix defining) $\phi_{\alpha,t,n}$ is singular. Thus, its determinant gives us a univariate equation in α of degree at most nt^2. Doing this for all W_i's gives us the promised bound of snt^2. ☐

Now, following [?], we give the construction of the subspaces W_i's for the case of $\Sigma\Pi\Sigma(n,k,d)$ circuits assuming a rank bound of $R(k,d)$ for the minimal simple identities.

Theorem 5.3. *Let C be a $\Sigma\Pi\Sigma(n,k,d)$ circuit and $S \subseteq \mathbb{F}$ be of size at least $n2^k d^2 R(k,d)^2$. If C is a nonzero circuit then there is an $\alpha \in S$ such that $\phi_{\alpha,n,R(k,d)}(C)$ is also nonzero.*

Proof. Let $C = T_1 + \cdots + T_k$ be nonzero. We define its *gcd part*, $gcd(C) := gcd(T_1, \ldots, T_k)$ and its *simple part*, $sim(C) := \frac{C}{gcd(C)}$.

We now define some subspaces spanned by the linear polynomials appearing in C. These subspaces shall have the property that any linear transformation which preserves their dimensions, leaves C nonzero. The subspaces are:

1. For every pair of linear forms ℓ, ℓ' that appear in the circuit define $W_{\ell,\ell'} := sp(\ell, \ell')$, where $sp(\cdot)$ refers to the linear *span*, over $\mathbb{F}$, of the set.

2. For every nonempty subset $A \subseteq [k]$, define $C_A := \sum_{i \in A} T_i$ and let $r_A := min\{R(k,d), rank(sim(C_A))\}$. Let W_A be the subspace spanned by r_A independent linear polynomials that appear in $sim(C_A)$.

Notice that the number of such subspaces is strictly less than $s := (k^2d^2 + 2^k)$. The claim is that for any linear transformation $\tau = \phi_{\alpha,n,R(k,d)}$ that preserves (the rank of) all these subspaces, also satisfies $\tau(C) \neq 0$. We will prove this by contradiction.

The first observation is that such a τ cannot map two linear functions, that appear in the circuit, to the same linear function since it preserves $W_{\ell,\ell'}$'s. Hence the simple part does not reduce further under τ, i.e. we have $sim(\tau(C_A)) = \tau(sim(C_A))$. So we can assume wlog that C (and hence $\tau(C)$ as well) is simple.

If $\tau(C) = 0$ then the rank bound entails that either $\tau(C)$ is not minimal or the $rank(\tau(C)) < R(k,d)$. If the latter is the case then $rank(C) < R(k,d)$, but then τ will preserve $W_{[k]}$, which means that the circuit C is itself zero. This contradicts the hypothesis.

The other case then is: $\tau(C)$ is zero but not minimal. Let A be a minimal subset such that $\tau(C_A) = 0$ with $C_A \neq 0$. Hence $\tau(sim(C_A))$ is a simple, minimal and zero circuit, therefore it is of rank less than $R(k,d)$. But then τ will preserve the rank of W_A, which together with $sim(C_A) \neq 0$ means that $\tau(sim(C_A)) \neq 0$. This is again a contradiction.

It is now a consequence of Lemma 5.2 that we will find such a τ if we try out $nsR(k,d)^2$ many α's. This finishes the proof. ☐

Algorithm & Complexity. The final black-box PIT algorithm is: given an access to a $\Sigma\Pi\Sigma(n,k,d)$ circuit C, try out $n2^kd^2R(k,d)^2$ many α's from the field (or its extension) and consider $C'(x_1, \ldots, x_{R(k,d)}) := \phi_{\alpha,n,R(k,d)}(C)$. Evaluate each such C' on $(d+1)^{R(k,d)}$ points on $\mathbb{F}^{R(k,d)}$, and announce C to be zero iff all these evaluations are zero. It is now evident that this is a valid algorithm and it requires "only" $poly(n, 2^k, d^{R(k,d)})$ many $\mathbb{F}$ operations.

5.2 An almost Optimal Rank Bound

The best known rank bound for minimal, simple $\Sigma\Pi\Sigma(n,k,d)$ identities is $k^3 \log d$ [**?**]. It is also close to optimal as there are identities of rank $\Omega(k \log d)$ [**?**, **?**]. This rank bound holds for *any* field. For special fields there is scope for improvement, for example Kayal & Saraf [**?**] showed a rank bound of k^k over reals using real geometry. Note that it is independent of d. Both kinds of rank bounds have quite involved proofs and tend to have a strong combinatorial flavor. We will only exhibit the basic ideas of the two rank bounds by working with the "toy" example of top fanin 3. Fortunately, these basic ideas extend to the higher fanins by developing the higher-dimensional generalizations.

The $k^3 \log d$ rank bound of Saxena & Seshadhri [?] hinges on a combinatorial *doubling* argument. It is best visible in the top fanin 3 case and proves a *sharp* upper bound of $(\lg d + 2)$, we prove it next.

Suppose $C = T_1 + T_2 + T_3 = 0$ is a minimal, simple $\Sigma\Pi\Sigma(n, 3, d)$ identity over some field $\mathbb{F}$. We will look at C modulo various linear forms that occur in the multiplication gate T_1. This reflects a dependency between the forms occurring in T_2 and T_3. For instance, pick a (nonzero) linear form q from T_1 and consider $C(\mathrm{mod}\ q)$ which gives $T_2 + T_3 = 0\ (\mathrm{mod}\ q)$. By unique factorization of polynomials modulo q this gives us a bijection π between the forms of T_2 with those in T_3, which we call a *q-matching between T_2, T_3*.

Definition 5.4. (Matchings) Let U, V be two lists of linear forms and I be a form. An *I-matching π between U, V* is a bijection π between lists U, V such that: for all $\ell \in U$, $\pi(\ell) = c\ell + v$ for some $c \in \mathbb{F}^*$ and $v \in sp(I)$.

Now as we pick different q's we get different matchings between T_2, T_3. The interesting property is that there cannot be too many such matchings if we only pick linearly independent q's. This we prove in the following lemma and it immediately gives a *sharp* rank bound for C.

Lemma 5.5. *Let U, V be two lists of linear forms each of size $d > 0$ and $I_1, \ldots, I_r$ be linearly independent linear forms such that for all $i \in [r]$, there is an I_i-matching π_i between U, V. If $r > (\lg d + 2)$ then U, V are similar lists (upto constant factors).*

Proof. For contradiction assume that $r > (\lg d + 2)$ but U, V are not similar lists. In that case we can assume wlog that U and V are *coprime* lists, i.e. there is no linear form that occurs (upto constant factors) in both the lists.

The proof is in the form of a combinatorial process that happens on a bipartite graph. The graph $G = (U, V, E)$ has vertices labelled with the respective forms. The various π_i's can be seen as bipartite matchings of G. For each π_i and each $\ell \in U$, we add an (undirected) edge tagged with I_i between the vertices ℓ and $\pi_i(\ell)$. There may be many tagged edges between a pair of vertices[1]. We call $\pi_i(\ell)$ the I_i-neighbor of ℓ (and vice versa). Abusing notation, we use *vertex* to refer to a form in $U \cup V$. We denote $\bigcup_{j \le i} I_j$ by J_i.

We will show that there cannot be more than $(\lg d + 2)$ such perfect matchings in G. The proof is done by following an iterative process that has r phases, one for each I_i. We maintain a partial basis for the forms in $U \cup V$

---

[1] It can be shown, using the independence of I_i's, that an edge can have at most *two* distinct tags.

which will be updated iteratively. This basis is denoted by the set B. The goal is to completely *span* the forms $U \cup V$ using the forms I_i's.

We start with an empty B and initialize by adding some $\ell \in U$ to B. In the ith round, we will add the form I_i to B. All forms of $U \cup V$ in $sp(\{\ell\} \cup J_i)$ are now spanned. We then proceed to the next round. To introduce some colorful terminology: A *green* vertex is one that is in the set $sp(B)$ (i.e. a form in $(U \cup V) \cap sp(B)$). Let vertex v be green, so $v \in sp(B)$. The I_1-neighbor of v is a linear combination of v and I_1. Therefore, the neighbor is also in $sp(B)$ and is colored green. This shows that the number of green vertices in U is equal to the number of those in V, at the end of each round.

Let $i_0 \in [r]$ be the least index such that $\{\ell\}$, $I_1, \ldots, I_{i_0}$ are linearly dependent, if it does not exist then set $i_0 := r + 1$. Now we have the following easy claim.

Claim 5.6. *The forms* $\{\ell\}$, $I_1, \ldots, I_{i_0-1}$ *are independent and the subspaces:* $sp(\{\ell\} \cup J_{i_0}), sp(I_{i_0+1}), \ldots, sp(I_r)$ *are independent.*

Proof of Claim 5.6. The forms $\{\ell\}$, $I_1, \ldots, I_{i_0-1}$ are independent by the minimality of i_0.

As $I_1, \ldots, I_{i_0}$ are independent but $\{\ell\}$, $I_1, \ldots, I_{i_0}$ are not, we deduce that $\ell \in sp(J_{i_0})$. Thus, the subspace $sp(\{\ell\} \cup J_{i_0}) = sp(J_{i_0})$ is independent to the forms $I_{i_0+1}, \ldots, I_r$ by the independence of $I_1, \ldots, I_r$. □

We shall now show that for $i \notin \{1, i_0\}$, the number of green vertices doubles in the ith round. Let ℓ' be a green vertex, say in U, at the end of the $(i-1)$th round (at that point $B = \{\ell\} \cup J_{i-1}$). Consider the I_i-neighbor of ℓ'. This is in V and is equal to $(c\ell' + v)$ where $c \in \mathbb{F}^*$ and v is a *nonzero* element in $sp(I_i)$ (since U, V are coprime). If this neighbor is green, then v would be a linear combination of two green forms, implying $v \in sp(B)$. But I_i is independent to B, implying $v \in sp(B) \cap sp(I_i) = \{0\}$ which is a contradiction. Therefore, the I_i-neighbor of any green vertex is *not* green. On adding I_i to B, the number of green vertices doubles (for at least $(r-2)$ rounds).

We started off with one green vertex ℓ, and lists U, V each of size d. Thus, this doubling can happen at most $\lg d$ times, implying that $(r-2) \leq \lg d$. This is a contradiction, implying that U, V are indeed similar lists. □

The above lemma immediately implies a rank bound for our identity C. As T_2, T_3 are coprime multiplication terms (by simplicity of C) the number of linearly independent forms q in T_1 can be at most $(\lg d + 2)$. Repeating this argument wrt T_2 and T_3 proves that $rank(C) = O(\lg d)$. Interestingly, the combinatorial procedure in the proof of Lemma 5.5 also suggests an identity that achieves this rank bound. It was first constructed by Kayal & Saxena [?]:

$$\begin{aligned} C(x_1,\ldots,x_r) := & \prod_{\substack{b_1,\ldots,b_{r-1}\in\mathbb{F}_2 \\ b_1+\cdots+b_{r-1}\equiv 1}} (b_1x_1+\cdots+b_{r-1}x_{r-1}) \\ & + \prod_{\substack{b_1,\ldots,b_{r-1}\in\mathbb{F}_2 \\ b_1+\cdots+b_{r-1}\equiv 0}} (x_r+b_1x_1+\cdots+b_{r-1}x_{r-1}) \\ & + \prod_{\substack{b_1,\ldots,b_{r-1}\in\mathbb{F}_2 \\ b_1+\cdots+b_{r-1}\equiv 1}} (x_r+b_1x_1+\cdots+b_{r-1}x_{r-1}) \qquad (5.1) \end{aligned}$$

It can be seen that, over $\mathbb{F}_2$, C is a simple and minimal $\Sigma\Pi\Sigma$ zero circuit of degree $d = 2^{r-2}$ with $k = 3$ multiplication terms and $rank(C) = r = \lg d + 2$.

In General. The above description gives a fair snapshot of the general rank bound. For a minimal, simple $\Sigma\Pi\Sigma(n,k,d)$ identity $C = T_1 + \cdots + T_k$, we need to consider *form-ideals* $I = (\ell_1,\ldots,\ell_{k-2})$, where form ℓ_i occurs in T_i. C modulo I then gives us I-matchings between T_{k-1}, T_k. If we look at such matchings modulo several linearly *independent* form-ideals I's then a generalization of Lemma 5.5 says that there can be at most $O(\lg d)$ independent form-ideals. Since each form-ideal I itself contains $(k-2)$ independent forms, this suggests an overall rank bound of $O(k \lg d)$. This proof idea when formalized gets into several problems, but can be salvaged to *prove* a rank bound of $k^3 \lg d$. The highest rank (minimal, simple) identities known are constructed using Equation 5.1, and have rank $\Omega(k \lg d)$. Thus, there is a slight gap in our understanding of rank.

5.3 A Rank Bound over Reals

The high rank identities that we saw in the last section are over fields with *nonzero* characteristic. When one tries to construct $\Sigma\Pi\Sigma(n,k,d)$ identities over zero characteristic fields, say rationals, one feels that no matter how large degree d is, the rank grows only like k. It was first conjectured by Dvir & Shpilka [?] that the rank of minimal simple $\Sigma\Pi\Sigma(n,k,d)$ identities over zero characteristic fields should be only $O(k)$. A weak form of this conjecture was shown true by Kayal & Saraf [?]. They proved a rank bound of k^k over the reals ($\mathbb{R}$). It is not trivial even for fanin $k = 3$. So we give below the proof for that case and then only state its generalization.

Let $C = T_1 + T_2 + T_3 = 0$ be a minimal, simple $\Sigma\Pi\Sigma(n,3,d)$ identity over $\mathbb{R}$. Suppose it has rank $(r+1)$. We identify every linear form ℓ in C with the corresponding point in $\mathbb{R}^r$. This form-to-point correspondence is just going to the *projective* space, roughly, a form $(a_1x_1 + \cdots + a_{r+1}x_{r+1})$ is mapped to the point $\left(\frac{a_1}{a_{r+1}}, \ldots, \frac{a_r}{a_{r+1}}\right)$ in $\mathbb{R}^r$. This mapping gives us sets of points A_1, A_2, A_3 corresponding to the linear forms occurring in T_1, T_2, T_3 respectively. Furthermore for any forms ℓ_1, ℓ_2 occurring in T_1, T_2 respectively, $C = 0$ modulo (ℓ_1, ℓ_2), implying that there exists a linear form $\ell_3 \in sp(\ell_1, \ell_2)$ that occurs in T_3. This means that any *line* passing through a point in A_1 and a point in A_2, also passes through a point in A_3. By symmetry this means that any line passing through two of the sets A_1, A_2, A_3 also passes through the third! Such sets $A_1, A_2, A_3 \subset \mathbb{R}^r$ are rather special and we will show below, following [**?**], that their existence implies $r \leq 3$. The proof is based on a famous theorem in incidence geometry - *Sylvester-Gallai theorem.*

Theorem 5.7 (Sylvester-Gallai). *Given a finite number of non-collinear points S in the plane $\mathbb{R}^2$, there always exists a line which passes through exactly two points in S.*

Proof. The simple proof below is due to Kelly (see the survey by Borwein & Moser [**?**]).

Define a *connecting line* to be a line which contains at least two points from S. For contradiction assume that every connecting line has a third point from S. Let (P, ℓ) be a point and a connecting line pair that are the smallest nonzero distance apart amongst all such point-line pairs.

The line ℓ goes through at least three points of S. Drop a perpendicular from P to ℓ, there must be two points on the same side of the perpendicular (one might be exactly on the intersection of the perpendicular with ℓ). Call the point closer to the perpendicular B, and the farther point C. Draw the line m connecting P to C. Then the distance from B to m is smaller than the distance from P to ℓ, which is a contradiction! One way to see this is to notice that the right triangle with hypotenuse BC is similar and contained in the right triangle with hypotenuse PC. This contradiction implies that there cannot be a nonzero distance between point-line pairs, thus every point must be at distance 0 from every connecting line, or in other words, every point must lie on the same line. But as S was non-collinear, we finally deduce that there exists a connecting line with exactly two points. ☐

Let us go back to our special sets $A_1, A_2, A_3 \subset \mathbb{R}^r$ obtained from the identity C and assume (for contradiction) that $r = 4$. Pick points p_1, p_2 from

A_1, A_2 respectively and consider the following *pencil of planes*,

$$\mathcal{P} := \{sp(p_1 - q, p_2 - q, q) \mid q \in A_1 \cup A_2 \cup A_3\}.$$

Notice that $\mathcal{P}$ consists of planes (formally 2-*flats*) in the space $\mathbb{R}^4$ and all of them contain the line (1-*flat*) joining the points p_1, p_2. The *dual* of this pencil of planes would give us a corresponding *pencil of lines* $\mathcal{L}$ in $\mathbb{R}^3$. Now if we look at a section of $\mathcal{L}$ cut by a plane in general position, we see $|\mathcal{L}|$ non-collinear points. By Theorem 5.7 there exists a line passing through exactly two of these points, which means there exists a plane in $\mathbb{R}^3$ containing exactly two of the lines in $\mathcal{L}$, which finally means that there exists a 3-flat in $\mathbb{R}^4$ containing exactly two of the planes in $\mathcal{P}$. Let us denote this 3-flat by H and the two planes it contains by H_1, H_2. Say H_1, H_2 are affine spans of the three points $(p_1, p_2, q_1), (p_1, p_2, q_2)$ respectively. Now if q_1, q_2 are not in the same A_i then the line joining them (so "across" H_1, H_2) should pass through a point in the third set A_j. But that is impossible as H contains only the planes H_1, H_2 from $\mathcal{P}$. Thus, q_1, q_2 have to be in the same A_i, say A_1. But then look at the line joining q_1, p_2, it has to contain a point from A_3, say q_3, which will of course be in H_1. The line joining q_3, q_2 (so "across" H_1, H_2) should pass through a point in the third set A_2, which is again impossible.

This contradiction shows that our assumption $r = 4$ cannot hold, in fact, the above contradiction appears as long as $r \geq 4$. Thus, r can be at most 3. This gives us a rank bound of 4 for simple $\Sigma\Pi\Sigma(n, 3, d)$ identities over reals. Interestingly, this bound is tight and there is a *unique* (upto transformations, see [?]) identity of rank 4 over $\mathbb{R}$:

$$\begin{aligned} &x_1x_2x_3(2y + x_1 + x_2 + x_3) - (y + x_1)(y + x_2)(y + x_3)(y + x_1 + x_2 + x_3) + \\ &y(y + x_1 + x_2)(y + x_2 + x_3)(y + x_1 + x_3) \;=\; 0. \end{aligned}$$

In General. The above idea extends to higher fanins, but the rank bound obtained is "weaker". Suppose $C = T_1 + \cdots + T_k = 0$ is a minimal, simple $\Sigma\Pi\Sigma(n, k, d)$ identity over $\mathbb{R}$. Instead of working in $\mathbb{R}^4$ and applying Sylvester-Gallai theorem as above, we now have to work in a much bigger space of $\mathbb{R}^m$ (roughly $m = k^k$) and use a higher-dimensional generalization of Sylvester-Gallai theorem that says:

Theorem 5.8. *([?], [?]) Let S be a finite set of points spanning an affine space $V \subseteq \mathbb{R}^n$ such that $dim(V) \geq 2t$. Then, there exist $(t + 1)$ points in S that span a t dimensional affine space $H \subset V$ such that $|H \cap S| = t + 1$.*

(Note that putting $t = 1$ above gives the statement of Sylvester-Gallai theorem.) In our case S is taken to be the set of points corresponding to all the forms that appear in C. Then $dim(V) = rank(C)$, which we assume large enough, say k^k, to derive a contradiction. Kayal & Saraf [?] now use Theorem 5.8 to identify a subspace of V and its *decomposition* (analogous to H and its "decomposition" H_1, H_2 above), and from that deduce the existence of a linear form ℓ in C such that $C(\text{mod } \ell)$ has a minimal, simple *sub*-identity of rank at least $(k-1)^{k-1}$. This gives the promised contradiction as $C(\text{mod } \ell)$, and hence the sub-identity, has a smaller fanin.

Open. It would be interesting to improve the above rank bound: (1) to other zero characteristic fields, (2) to a more optimal-looking bound $O(k)$.

It is known that Sylvester-Gallai theorem 5.7 is false in $\mathbb{C}^2$ (cubic curves give counter examples [?]). Nevertheless, it is known to hold in the following sense [?, ?]: *Given a finite number of non-coplanar points S in $\mathbb{C}^3$, there always exists a line which passes through exactly two points in S.* This immediately gives us a rank bound of 5 (unlike 4 before) for simple $\Sigma\Pi\Sigma(n, 3, d)$ identities over $\mathbb{C}$. Unfortunately, a higher-dimensional generalization of this version is not known. A natural conjecture for it would be:

Conjecture 5.9. Let S be a finite set of points spanning an affine space $V \subseteq \mathbb{C}^n$ such that $dim(V) \geq 3t$. Then there exist $(t+1)$ points in S that span a t dimensional affine space $H \subset V$ such that $|H \cap S| = t + 1$.

(Currently, a proof exists only for $t = 1$.)

6 Depth-4 PIT

We know that depth-4 case of PIT has direct relations to more general cases of PIT. But currently we have little understanding of depth-4 circuits. For example, we do not even know how to test $f_1 \ldots f_m = g_1 \ldots g_m$ where f_i's and g_i's are multivariate polynomials given in *fully expanded* form (also called the *sparse representation*). Here we will discuss two simple ideas that solve PIT for certain restricted forms of depth-4 circuits.

Noncommuting Idea. The first idea is easiest to see on depth-4 circuits C whose multiplication gates have *unmixed* variables, i.e. $C(x_1, \ldots, x_n) = M_1 + \cdots + M_k$, where for all $i \in [k]$, $M_i = f_{i,1}(x_1) \cdots f_{i,n}(x_n)$, where each $f_{i,j}$ is a univariate polynomial given in the sparse representation and is of degree at most d. Note that "expanding out" M_i in a brute force way could potentially produce d^n monomials and hence is not recommended!

Interestingly, Raz & Shpilka formulated a "controlled" way of doing this expansion using basic linear algebra. Their idea was to compute $f_{i,1}(x_1)f_{i,2}(x_2)$, for all $i \in [k]$. View each of these polynomials as vectors in a natural way, i.e. each coefficient is a coordinate of the vector. Thus we have k vectors $V := \{v_1, \ldots, v_k\}$ in a space of dimension at most d^2. Pick some maximal subset of V that has linearly independent vectors, say they are $v_1, \ldots, v_\ell$ ($1 \leq \ell \leq k$). Now we form the circuit C_1 from C by replacing $f_{i,1}(x_1)f_{i,2}(x_2)$, for all $i \in [\ell]$, by a *fresh* variable $z_{1,i}$. For the other i's, replace $f_{i,1}(x_1)f_{i,2}(x_2)$ by the same linear combination of $\{z_{1,1}, \ldots, z_{1,\ell}\}$ as the one that expresses v_i in terms of $\{v_1, \ldots, v_\ell\}$. It is easy to verify that $C(x_1, \ldots, x_n) = 0$ iff $C_1(z_{1,1}, \ldots, z_{1,\ell},\ x_3, \ldots, x_n) = 0$. Thus this one round reduced the number of factors in each M_i by *one* at the cost of increasing the number of variables by $O(k)$. If we repeat this round $(n-2)$ more times then instead of M_i's we would have just linear forms and the number of variables would be $O(nk)$. As the linear algebra in each round requires just $poly(ndk)$ operations, and testing the zeroness of the circuit after the last round is trivial, we get an overall complexity of $poly(ndk)$ field operations.

It can be easily verified that the technique above is applicable in several other cases, in particular: (1) when C is a *set-multilinear* formula, i.e. each multiplication gate M_i has t inputs that are polynomials in disjoint variables $S_1, \ldots, S_t$ fixed such that $S_1 \sqcup \cdots \sqcup S_t = \{x_1, \ldots, x_n\}$. (2) more generally, when C is a *noncommutative* formula, i.e. variables $x_1, \ldots, x_n$ do not commute wrt multiplication.

Powering Idea. The second idea is to consider depth-4 circuits C whose multiplication gates just do *powering*, i.e. $C(x_1, \ldots, x_n) = M_1 + \cdots + M_k$, where for all $i \in [k]$, $M_i = \alpha_i \cdot (f_{i,1}(x_1) + \cdots + f_{i,n}(x_n))^{e_i}$, where each $f_{i,j}$ is a univariate polynomial given in the sparse representation and is of degree at most d, $\alpha_i \in \mathbb{F}$ and $e_i \in \mathbb{N}$. Again note that "expanding out" M_i in a brute force way could potentially produce more than $\binom{n+e_i}{n}$ monomials. Interestingly, although this case looks to be on the other extreme of the "unmixed variable" case discussed above, a reduction of the former to the latter was given by Saxena [**?**]. The following lemma gives the main transformation:

Lemma 6.1. *Let $g_1(x_1), \ldots, g_n(x_n)$ be univariate polynomials of degree at most d, over a field $\mathbb{F}$ of zero characteristic. Then we can compute univariate polynomials $h_{i,j}$'s in $poly(nda)$ field operations such that for $t = (na+1)$:*

$$(g_1(x_1) + \cdots + g_n(x_n))^a \;=\; \sum_{i=1}^{t} h_{i,1}(x_1) \cdots h_{i,n}(x_n)$$

Proof. We will prove this using the formal power series: $\exp(x) = 1 + x + \frac{x^2}{2!} + \cdots$, where $\exp(x) = e^x$ and e is the base of natural logarithm. Define the degree a truncation of the series to be $E_a(x) = 1 + x + \cdots + \frac{x^a}{a!}$. We will use the *operator* $[z^a]$ to extract the coefficient of z^a from a polynomial.

Observe that:

$$\begin{aligned}(a!)^{-1} \cdot (g_1(x_1) + \cdots + g_n(x_n))^a &= [z^a]\ \exp\left((g_1(x_1) + \cdots + g_n(x_n)) \cdot z\right) \\ = [z^a]\ \exp(g_1(x_1)z) \cdots \exp(g_n(x_n)z) & \\ = [z^a]\ E_a(g_1(x_1)z) \cdots E_a(g_n(x_n)z) &\end{aligned}$$

The product $E_a(g_1(x_1)z) \cdots E_a(g_n(x_n)z)$ can be viewed as a univariate polynomial in z of degree na. Hence, its coefficient of z^a can be computed by evaluating the polynomial at t distinct points $\alpha_1, \ldots, \alpha_t \in \mathbb{F}$ (remember $\mathbb{F}$ is large enough) and by interpolation we can compute $\beta_1, \ldots, \beta_t \in \mathbb{F}$ such that:

$$\begin{aligned}&[z^a]\ E_a(g_1(x_1)z) \cdots E_a(g_n(x_n)z) \\ &= \sum_{i=1}^{t} \beta_i \cdot E_a(\alpha_i g_1(x_1)) \cdots E_a(\alpha_i g_n(x_n))\end{aligned}$$

This can be seen as the *dual form* of the multiplication gate $(g_1(x_1) + \cdots + g_n(x_n))^a$. It is routine to verify that all the univariate polynomials $E_a(\cdot)$ in the above sum can be computed in $poly(nda)$ field operations. □

Applying this lemma to all the gates M_i's of $C(x_1, \ldots, x_n) = M_1 + \cdots + M_k$, we convert our given circuit C to another circuit $C'(x_1, \ldots, x_n)$ which is a sum-of-product of univariates. Such a C' can now be tested for zeroness using the noncommuting idea seen above.

The technique of Lemma 6.1 also applies to a slightly general case of (s is any constant):

$$C(x_1, \ldots, x_n) = \sum_{i=1}^{k} L_{i,1}^{e_{i,1}} \cdots L_{i,s}^{e_{i,s}}$$

where the $L_{i,j}$'s are sums of univariate polynomials, i.e. for all $i \in [k], j \in [s]$: $L_{i,j}(x_1, \ldots, x_n) = f_{i,j,1}(x_1) + \cdots + f_{i,j,n}(x_n)$ where $f_{i,j,j'} \in \mathbb{F}[x_{j'}]$.

Open. The PIT algorithms in the above two restricted cases of depth-4 circuits are inherently *non* black-box. Are there black-box PIT algorithms for these family of depth-4 circuits? Currently, there are no black-box PIT algorithms known for any nontrivial family of depth-4 circuits.

7 General PIT and Lower Bounds

We saw in the above sections that PIT algorithms, at least the ones currently known, are quite involved and require ideas from algebra, geometry and combinatorics. This proof complexity is partially explained by the connection

PIT has to certain circuit lower bounds (that are historically considered difficult to prove!). We will now discuss how a theorem like PIT $\in$ P would imply lower bounds [?], and that a black-box PIT algorithm would imply even stronger lower bounds [?].

Implications of PIT in P. Kabanets & Impagliazzo [?] showed that if PIT has a deterministic polynomial time algorithm then *either* NEXP (i.e. nondeterministic exponential time class) does not have polynomial sized boolean circuits (i.e. $P/poly$ class) *or* the *perm* function (i.e. permanent of a square matrix of rationals) does not have polynomial sized arithmetic circuits (i.e. $AlgP/poly$ class). Note that both the claims NEXP $\not\subseteq P/poly$ and $perm \notin AlgP/poly$ are conjectured to be true by "most" people. So the main attraction of the following theorem is that it *connects* an algorithmic conjecture like PIT$\in$P with these lower bound conjectures.

Theorem 7.1. *([?]) If low-degree-PIT $\in$ P then: NEXP $\not\subseteq$ P/poly or perm $\notin$ AlgP/poly.*

Proof Sketch: The proof is by contradiction, so we assume:

1. low-degree-PIT $\in$ P

2. $perm \in AlgP/poly$

3. NEXP $\subseteq P/poly$

Assumptions (1) and (2) imply that, one can guess an arithmetic circuit C and then actually check whether $C = perm$ using PIT. As permanent of an $m \times m$ matrix has degree m, one only needs to guess C of depth $O(\lg m)$ and multiplication fanin 2 [?], then do low degree PIT. The part where we check $C = perm$ using PIT, actually makes use of the *downward self-reducibility* of the permanent function, i.e. *perm* of an $m \times m$ matrix can be expressed as the sum of m permanents each of $(m-1) \times (m-1)$ submatrices. Finally,

4. $\mathrm{P}^{perm} \subseteq$ NP.

Now it is known that assumption (3) implies NEXP $\subseteq \mathrm{P}^{perm}$ [?]. This together with deduction (4) implies that NEXP $\subseteq$ NP, which is a contradiction as there are classical *diagonalization* methods proving NEXP different from NP [?]. $\square$

One might wonder whether this theorem has a converse. As a step in that direction, it was shown in [?]: *if permanent has superpolynomial arithmetic circuit complexity then PIT has a subexponential time algorithm.* The idea is to apply a hard function (here permanent) on the *designs* defined by Nisan

& Wigderson [?], and evaluate the given circuit C at the resulting *point*. The claim is that this evaluation is zero iff C is a zero circuit. Thus, hard algebraic functions give black-box PIT algorithms!

Implications of black-box PIT. As we have mentioned before, black-box PIT algorithms seem to require a very good understanding of the circuit family and hence should, intuitively, also imply what that circuit family cannot compute! It is interesting, this intuition can also be proven formally, as we will now show following Agrawal [?].

A black-box PIT algorithm is only allowed to evaluate a given circuit $C(x_1, \ldots, x_n)$ at points in the extensions of the given field $\mathbb{F}$. Thus, it seems reasonable to assume that such an algorithm just "feeds in" $x_i = f_i(y) \pmod{g(y)}$ for all $i \in [n]$, where f_i's and g are univariate polynomials of a "small" degree $2^{\ell(n)}$. Note that the algorithm feeds these polynomials to every input circuit $C(x_1, \ldots, x_n)$, only assuming that C has a size bound of (wlog) n. Clearly, the time complexity of such a black-box PIT algorithm is dominated by $2^{\ell(n)}$, and the time taken to actually construct f_i's and g. This motivates the definition of a *pseudo-random generator (prg)* for arithmetic circuits.

Definition 7.2. Fix a field $\mathbb{F}$. A function $f : \mathbb{N} \to (\mathbb{F}[y])^*$ is called an *efficient* $(\ell(n), n)$*-prg* if,

- $f(n) \in (\mathbb{F}[y])^{n+1}$ for all $n > 0$.
- $f(n) = (f_1(y), \ldots, f_n(y), g(y))$ where the polynomials f_i's and g are of degree at most $2^{\ell(n)}$, and are also constructible in $poly(2^{\ell(n)})$ time.
- For any circuit $C(x_1, \ldots, x_n)$ of size at most n, $C(x_1, \ldots, x_n) = 0$ iff $C(f_1(y), \ldots, f_n(y)) = 0 \pmod{g(y)}$.

If we drop the requirement of efficient constructibility then such functions f, for any $\ell(n) = \Omega(\lg n)$, can be easily shown to exist using the Schwartz-Zippel lemma. On the other hand it can be seen, by the methods of Section 2, that efficient ones for $\ell(n) = O(n^2)$ exist. The really interesting cases for us are in between, and so we will always assume $\ell(n) = \Omega(\lg n)$ and $\ell(n) = o(n)$. The existence of an efficient $(\ell(n), n)$-prg immediately gives a black-box PIT algorithm with time complexity $poly(2^{\ell(n)})$. Thus, to completely solve PIT we "just" need an efficient $(O(\lg n), n)$-prg. We now show that such a prg implies arithmetic circuit lower bounds (that are beyond the scope of current methods).

Theorem 7.3. *([?]) If there is an efficient $(\ell(n), n)$-prg. Then there is a multilinear polynomial that is $poly(2^{\ell(n)})$ time computable but has no circuits of size n.*

Proof Sketch: Let $f(n) = (f_1, \ldots, f_n(y), g(y))$ be an efficient $(\ell(n), n)$-prg. Let $m := \ell(n)$. In the interesting case of $\ell(n) = o(n)$, we have $n > 2m$. We define a polynomial $q_f(x_1, \ldots, x_{2m})$ as:

$$q_f(x_1, \ldots, x_{2m}) := \sum_{S \subseteq [1,2m]} c_S \cdot \prod_{i \in S} x_i.$$

Where the coefficient c's are picked such that they satisfy:

$$q_f(f_1(y), \ldots, f_{2m}(y)) = \sum_{S \subseteq [1,2m]} c_S \cdot \prod_{i \in S} f_i(y) = 0.$$

The existence of such coefficient c's can be seen by comparing the degree of the above equation in y and the number of the unknowns. Furthermore, the polynomial q_f can be computed by solving a system of $poly(2^m)$ linear equations in $poly(2^m)$ variables over the field $\mathbb{F}$. Each of these equations can be computed in time $poly(2^m)$ using the computability of f. Therefore, q_f can be computed in time $poly(2^m)$.

Now suppose q_f can be computed by a circuit C of size n. By the definition of polynomial q_f, it follows that $C(f_1(y), f_2(y), ..., f_{2m}(y)) = 0$. On the other hand, the size of the circuit C is n and it computes a nonzero polynomial. This contradicts to f being a prg. Hence $q_f(x_1, \ldots, x_{2\ell(n)})$ is a multilinear polynomial computable in $poly(2^{\ell(n)})$ time but not by n-sized circuits. □

The way q_f is defined above has the nice property that it can be expressed as the *permanent* of a "small" matrix [**?**]. Thus the existence of an efficient prg f would not only give a "hard" polynomial q_f but indeed prove the hardness of permanent function!