Low Variate Polynomials: Hitting Set and Bootstrapping

A Thesis Submitted

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

by

Sumanta Ghosh

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

September, 2019

DECLARATION

This is to certify that the thesis titled "Low Variate Polynomials: Hitting Set and Bootstrapping" has been authored by me. It presents the research conducted by me under the supervision of Prof. Nitin Saxena at the Department of Computer Science and Engineering, IIT Kanpur. To the best of my knowledge, it is an original work, both in terms of research content and narrative, and has not been submitted elsewhere, in part or in full, for a degree. Further, due credit has been attributed to the relevant state-of-the-art and collaborations (if any) with appropriate citations and acknowledgements, in line with established norms and practices.

Sumanta Ghosh
Program: Doctor of Philosophy
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur
Kanpur 208016

September, 2019

CERTIFICATE

It is certified that the work contained in the thesis entitled "Low Variate Polynomials: Hitting Set and Bootstrapping", by "Sumanta Ghosh", has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Prof. Nitin Saxena
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

September, 2019

Synopsis

In this thesis, we study Polynomial Identity Testing (PIT) problem. It is one of the central problems in algebraic complexity theory. PIT problem asks whether a multivariate polynomial is *zero*, where the input polynomial is given as an *algebraic circuit.* We have a polynomial-time randomized algorithm for PIT. However, designing a deterministic polynomial-time algorithm for PIT is a long-standing open question in algebraic complexity theory. It has deep connections with both circuit lower bounds and many other algorithmic problems like perfect matching, multivariate polynomial factorization.

We consider PIT problem in the blackbox setting, where we are not allowed to see the internal structure of the circuit, but evaluations at points are allowed. For instance, the randomized algorithm for PIT is a blackbox algorithm. Designing a *deterministic* blackbox PIT algorithm for a circuit class is equivalent to finding a set of points such that for every nonzero circuit, the set contains a point where it evaluates to a nonzero value. Such a set is called *hitting* set. So by derandomizing PIT, we mean designing a poly(s)-time computable hitting set for s-variate size s degree s circuits.

We study the phenomenon of *bootstrapping* in PIT: converting a hitting set for $n(s)$-variate size s degree s circuits to a hitting set for s-variate size s degree s circuits where $n(s) \ll s$. The analog of this phenomenon in boolean settings is well understood due to a seminal work by Nisan-Wigderson. Their result is optimal for boolean circuits. So, no further improvement is possible there. However, the situation was less clear in algebraic settings. We have bridged this gap in knowledge by showing that we can efficiently perform the bootstrapping when the number of variables $n(s)$ is as small as $\log^{\circ c} s$, where $\log^{\circ c} s$ is c-times composition of logarithmic function with itself. For example,

$\log^{\circ 2} s = \log\log s$. Furthermore, at the cost of making the final hitting set "slightly" superpolynomial ($s^{\exp\circ\exp(O(\log^\star s))}$), we show that bootstrapping can be done from even a *constant* number of variables!

Our results can also be viewed as a powerful *amplification* of derandomization: a s^{n^δ}-size hitting set for some $\delta < 1/2$, for n-variate size s degree s circuit implies a "slightly" super-polynomial ($s^{\exp\circ\exp(O(\log^\star s))}$) size hitting set for size s degree s circuits. Trivial derandomization of the randomized algorithm for PIT gives a $(s+1)^n$-size hitting set for n-variate size s degree s circuits. To completely derandomize PIT, our goal is to design a poly(s)-size hitting set for s-variate size s degree s circuits. Our result implies that a "small" improvement over trivial hitting set is sufficient to "almost" completely derandomize PIT. We prove an additional bootstrapping result for shallow circuits: $o(s^{n/2})$-size hitting set for n-variate size s depth-4 circuits (for some constant $n \geq 3$) implies quasi-polynomial ($s^{O(\log s)}$) time blackbox PIT algorithm for general circuits.

Our result, bootstrapping in PIT, motivates us to study the circuits with "few" variables, for example, the number of variables is logarithmic with respect to the circuit size. We give the first poly(s)-time blackbox PIT algorithm for $O(\log s)$-variate size s circuits computing polynomials with poly(s) dimensional partial derivative space. As depth-3 diagonal circuit is a prominent circuit class which has polynomially large dimensional partial derivative space, our result gives the first polynomial-time blackbox PIT algorithm for log-variate depth-3 diagonal circuits. Moreover, to design blackbox PIT for those models, we show an efficient blackbox PIT algorithm for polynomials whose support set contains a "low-cone" monomials. Our result shows that the cone size of monomials is a useful measure in the log-variate regime. So, it may be useful in studying other log-variate models.

We introduce the concept of *cone-closed basis* for the polynomials over the vector space $\mathbb{F}^k$. It is a stronger notion of rank concentration compared to low-support rank concentration and low-cone concentration. We showed that if a polynomial over $\mathbb{F}^k$ is shifted by its basis isolating weight assignment, then the new polynomial becomes cone-closed. It is currently the best known rank concentration result for polynomials over vector spaces.

Acknowledgements

I am very fortunate to have Nitin Saxena as my PhD advisor. Being a student of him is a privilege for me. I owe my progress in research to him. He was always generous with his time, and we have had so many instructive discussions. I have learned many techniques and ideas from those countless hours of discussions. His belief in hard work and incremental developments of students has been a constant source of motivation for me. He has always provided me enough freedom to do research at my own pace. His strong perseverance towards solving problems is the biggest lesson I have learnt from him about research.

At IIT Kanpur, I was fortunate to experience the teaching of Somenath Biswas, Manindra Agrawal, Sumit Ganguly and Surender Baswana. The algorithm courses offered by Surender Baswana made me realize how beautifully an algorithm can be taught as well as learnt. Discussing with Manindra Agrawal has always been an inspiring experience for me. Even though I had few opportunities to discuss with him, his way of thinking, articulating ideas and teaching have been influential in shaping my own way of thinking on problems. My sincere thanks to Somenath Biswas for motivating us in research. I am grateful to Piyush P Kurur for guiding me in M. Tech. From him, I have learned how to write better technical articles. Thanks to Rajat Mittal and Satyadev Nandakumar for many discussions on various topics and helpful advice. Above all, I would like to express my gratitude to all the faculties in the department for providing us a wonderful friendly environment in the department. Thanks to the technical and administrative staff in the CSE department for always being helpful. I thank the institute as a whole for providing us such a conducive environment.

This thesis is based on joint works with Manindra Agrawal, Michael A. Forbes and my advisor. I am thankful to them. This work would not have been possible without them.

During my PhD, I got the opportunity to attend some wonderful workshops, which had given me an exposition of wide range of topics in my research area. I want to thank the organizers of the workshops I participated in, WACT 2016 at Tel Aviv University, Mysore Park workshop in 2016, NMI workshop in arithmetic circuit complexity at IMSc Chennai in 2017 and WACT 2019 at ICTS Bangalore. Thanks to Google India Pvt. Ltd. and Research-I foundation of Infosys for funding my travel expenses to the conferences CCC 2018, STOC 2018 and ICALP 2018. Thanks to Michal Koucky for hosting me during a research visit at Charles University Prague and giving me opportunity to participate in the Summer School on Algorithms and Lower Bounds 2018 organized by him.

I would like to thank Amit, Ashish, Anamay, Diptarka, Mahesh, Pranjal, Pranav, Prateek, Rajendra, Subhayan, Vishwas, Zeyu for countless discussions covering wide range of topics in computer science. Without them, I would not have learnt as much as I did. I got opportunities to meet Ramprasad on multiple occasions. His vast knowledge in computer science, enthusiasm for research and way of explaining ideas is always inspiring.

Life at IIT Kanpur would not be so easy and joyful without the company I had here. Thanks to all my labmates and friends for the wonderful time we had together. I sincerely thank Diptarka, Rajendra, Mahesh, Pranjal, Prateek for being so helpful all the time. Special thanks to Amit for our decade-old friendship spanning from my undergraduate days and being a constant source of support all the time.

Many thanks to my maternal uncle, Sanat Ghosh, for guiding me and being a source of inspiration from my childhood. My sincere thanks to my uncle, Dibakar Ghosh, for teaching me mathematics with great care and affection during my school days. I am grateful to kakima, Manasi Saha, for her love and care.

Most of all, thanks to my parents, my elder sister and my younger brother. I am indebted to them for their continual and unconditional love, care and support. Without their support and love, it would not be possible for me to reach this day. I fondly dedicate this thesis to them.

Contents

List of Publications

[AGS18] **Bootstrapping variables in algebraic circuits**
with Manindra Agrawal and Nitin Saxena
50th ACM Symposium on Theory of Computing (STOC), 2018
Article in Proceedings of the National Academy of Sciences of the USA, PNAS, 2019

[FGS18] **Towards Blackbox Identity Testing of Log-Variate Circuits**
Michael A. Forbes and Nitin Saxena
45th International Colloquium on Automata, Languages, and Programming (ICALP), 2018

Chapters 3, 4 and 5 are based on [AGS18]. Chapters 6 and 7 are based on [FGS18].

List of Figures

Chapter 1

Introduction

Our motivation for studying complexity theory is to classify the computational problems based on the computational resources needed to solve them. We mainly focus on three computational resources *time*, *space* and *random bits*. Computational complexity theory consists of studying various complexity classes and their relations. Among various complexity classes, P and BPP are two major classes that contain the computational problems which are efficiently solvable. The class P is the set of problems which can be solved in *deterministic* polynomial time and the class BPP is the set of problems which can be solved in *randomized* polynomial time. In many scenarios, the use of randomness gives us simpler and more efficient solutions compared to the known deterministic ones. Besides giving faster algorithms, randomness is also essential for the security of cryptosystems, distributed computing, and in other areas. However, in practice, random bits are hard to generate. Therefore, one can ask whether randomness is essential in efficiently solving computational problems. In complexity theoretic terms, we are interested to know whether P=BPP and it is believed to be true.

Another well-studied class in complexity theory is NP. The class NP is the set of problems which can be solved in *non-deterministic* polynomial time. Informally, the solution to every problem in NP is efficiently *certifiable*, but it may not be easy to find that certificate. For example, consider the problem SAT of deciding whether a given boolean formula has a satisfying assignment. If the given boolean formula has a satisfying assignment, someone can give such an assignment as a certificate and also easily verify that assignment by putting the values on the formula. However, it is believed that finding such an assignment is not efficient. One nice property of NP is that it contains a subset of problems, called *NP-complete problems*, such that if one can find a polynomial-time solution to any one of these problems, all problems in NP will be polynomial-time solvable. The notion of NP-completeness reduces the study of whole NP class to some concrete and well-structured problems. Like the P vs BPP question, people are also interested in studying whether P is equal to NP, and it is widely believed that these two classes are different.

Unlike NP, we do not know any complete problem for BPP. Hence, we try to find "good" problems which have efficient randomized solutions and try to derandomize them. This gives us the hope that the newly developed tools for derandomizing that good problem may help to derandomize BPP class completely. In this thesis, we study one such problem called *polynomial identity testing*, in short, we call PIT.

1.1 Polynomial Identity Testing Problem

Polynomial identity testing (PIT) problem is to decide whether a given multivariate polynomial is identically zero, i.e., the coefficient of every monomial is zero. For example, $(x+y)(x-y)-x^2+y^2$ is a identically zero polynomial. The hardness of PIT depends on how we represent the input polynomial. If the input polynomial is given as a list of monomials and their coefficients, we can easily solve PIT just by checking the coefficients. However, such an explicit representation always may not be available. In this work, we consider a very compact representation of polynomials called an *algebraic circuit.*

An algebraic circuit over a field $\mathbb{F}$ is an acyclic directed graph with one sink node called output node, source nodes are called input nodes and are labeled by variables from the set $\mathbf{x} = \{x_1, \ldots, x_n\}$ or constants from $\mathbb{F}$, non-input nodes are labeled by $\times$ (multiplication gate) or $+$ (addition gate). Every edge is labeled by a constant. The computation is defined in a natural way: Every edge collects the polynomial computed at its tail node, scale by

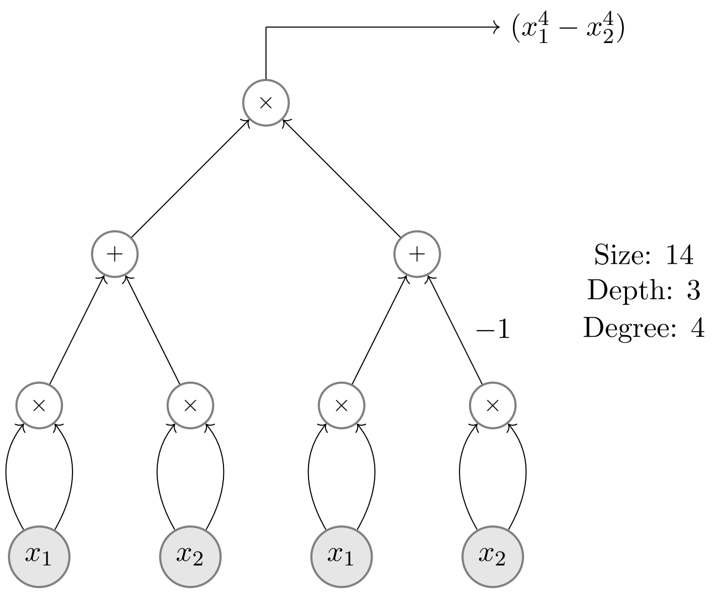


Figure 1.1: Algebraic circuit (default edge label is 1)

the constant labeled on it and sends to its head node. Each addition gate computes the sum of all polynomials coming to it by the incoming edges. Similarly, each multiplication gate computes the product of all the polynomials coming to it by the incoming edges. The polynomial computed at the output node is called the polynomial computed by the circuit. The *fan-in* of a node is defined as its in-degree and the *fan-out* is defined as its out-degree. The complexity parameters of a circuit are: 1) *size*- the number of edges which is equal to the number of additions and multiplications we perform to compute the polynomial computed by the circuit, 2) *depth*- the length of a longest path in the circuit, and 3) *degree*- the maximum degree among all polynomials computed at each node. For example, see Figure 1.1. Note that the degree of a polynomial can be smaller than the degree of a circuit computing it. With a polynomial blow-up in size, any circuit can be converted to a circuit where the $+$ and $\times$ gates appear in alternate layers, and the edges appear only between the consecutive layers. Hence, we can assume the given circuit has this structure. By Σ we denote a layer of $+$ gates, and by Π we denote a $\times$ layer. If each multiplication gate of a Π-layer computes power of some polynomial, we denote that Π-layer by $\wedge$.

Polynomial identity testing problem can be defined as follows: Given a circuit C, decide whether C computes zero polynomial. The polynomial computed by a circuit may have, in

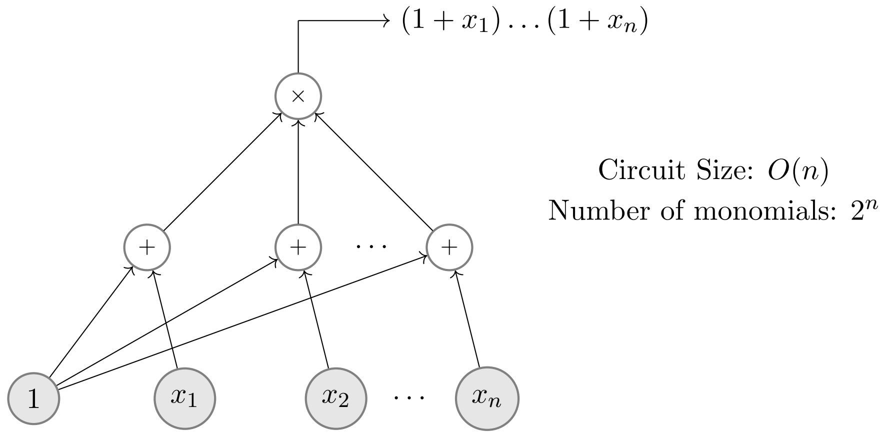


Figure 1.2: Circuit computing exponentially many monomials

the worst-case, an exponential number of monomials compared to its size. For example, see Figure 1.2. Therefore, by computing the explicit polynomial from the input circuit, we cannot solve PIT problem in polynomial time. However, evaluation of a circuit at a point can be done efficiently by assigning the values at the input nodes. This helps us to get a polynomial-time randomized algorithm for PIT by evaluating the circuit at a random point [DL78, Zip79, Sch80]. However, designing a deterministic polynomial-time algorithm for PIT is a long-standing open question in algebraic complexity theory. For solving PIT, we can assume the output gate (also called top gate) of the input circuit is a $+$ gate. Otherwise, the given circuit C computes a polynomial of the form $C_1 \cdots C_k$, where each C_i is a circuit of smaller size with $+$ as the top gate, and testing nonzeroness of C reduces to testing nonzeroness of C_i's.

PIT algorithms are of two kinds: 1) *whitebox*- allowed to use the internal structure of the circuit, and 2) *blackbox*- not allowed to see the internal structure of the circuit, only evaluation of the circuit is allowed at points from $\mathbb{F}$ (or a 'small' extension of $\mathbb{F}$). Blackbox PIT algorithms are stronger compared to whitebox PIT algorithms. Designing a deterministic blackbox PIT algorithm for a set of n-variate polynomials $\mathcal{P}$ is equivalent to finding an $\mathcal{H} \subseteq \mathbb{F}^n$, called *hitting set*, such that for every nonzero polynomial $P \in \mathcal{P}$, the set $\mathcal{H}$ contains a point at which P evaluates to a nonzero value. Sometimes a functional representation of hitting set would be more convenient. We can see the hitting set $\mathcal{H}$ in terms of a tuple of univariates $\mathbf{f}(y) = (f_1(y), \dots, f_n(y))$, whose set of evaluations contain $\mathcal{H}$. Such an $\mathbf{f}(y)$ can be efficiently obtained from a given $\mathcal{H}$ via interpolation. Clearly, if $\mathcal{H}$ is a hitting-set for $\mathcal{P}$ then $P(\mathbf{f}(y)) \neq 0$, for any nonzero $P \in \mathcal{P}$. This tuple of univariates is called *hitting-set generator* (or *HSG*, in short) and its *degree* is $\max_{i\in[n]} \deg(f_i)$, which is $\leq |\mathcal{H}|$. In this thesis, we focus only on blackbox PIT algorithms.

The randomized PIT algorithm due to [DL78, Zip79, Sch80] is a blackbox algorithm for PIT. If we try to derandomize that trivially, we get an exponential size hitting set which is also computable in same time complexity (see Section 2.3.1). To completely derandomize PIT, we want a polynomial-time computable hitting set. First, [HS80] showed the existence of a poly(s)-size hitting set for polynomials computed by size s degree s circuits. Moreover, they showed that any random subset of poly(s)-size is a hitting set for the same class of polynomials. Their proof was non-constructive. Hence, their hitting set is not explicit. Using [HS80], Mulmuley in [Mul17] gave a PSPACE construction of a polynomial-size explicit hitting set. This is currently the best known deterministic construction of polynomial-size hitting set for PIT.

PIT has applications both in proving circuit lower bounds and in designing various algorithms. Kabanets and Impagliazzo [KI04] proved that a sub-exponential whitebox PIT algorithm implies a separation between two algebraic classes (VP$\neq$VNP) or two boolean classes (NEXP $\not\subseteq$ P/poly). [HS80, Agr05] showed that a polynomial-time blackbox PIT algorithm implies an exponential algebraic circuit lower bound (see Section 2.3.3). The polynomial-time primality testing by Agrawal, Kayal and Saxena can be seen as a special instance of PIT problem [AKS04]. Kopparty, Saraf and Shpilka showed the equivalence between polynomial identity testing and deterministic multivariate polynomial factorization [KSS14]. The perfect matching problem reduces to PIT question for a special class of polynomials [Tut47, Lov79, MVV87, FGT16, ST17]. PIT has applications to many other problems [DdOS14, GT17, GTV18, Sha90].

The overall development of PIT can be divided into two categories: 1) conditional PIT algorithms, where we design PIT algorithms under some assumptions like the existence of an explicit hard polynomial family and 2) unconditional PIT algorithms, where we design PIT algorithms without any assumptions. Currently, efficient unconditional PIT algorithms are known only for very restricted circuit models. However, several conditional PIT algorithms are known for general circuit classes.

First, [KI04] gave a quasi-polynomial time blackbox algorithm from an exponentially hard mutilinear polynomial family (see Lemma 2.3.8). Later, [DSY09, CKS18] showed that lower bound for bounded depth (e.g., assume Δ depth) circuits implies efficient blackbox PIT algorithms for bounded depth (i.e., $\Delta - O(1)$ depth) circuits. Surprisingly, [AV08] showed that polynomial time blackbox PIT algorithm for *depth-4* circuits gives quasi-polynomial time blackbox PIT algorithm for *general* circuits. A similar result was also shown in [GKKS16] by assuming polynomial time blackbox PIT algorithm for *depth-3* circuits. However, the result in [GKKS16] works only over characteristic zero fields.

Now we describe the unconditional PIT algorithms known for various special classes of circuits. Depth-2 ($\Sigma\Pi$) circuits are the simplest circuit model. They compute polynomials as a small sum of monomials. It is easy to give an efficient whitebox test for them just by checking the coefficients of the monomials. A polynomial time blackbox PIT algorithm is also known for depth-2 circuits [BT88, KS01]. Our next model of interest is depth-3 circuits. In the previous paragraph, we have seen that designing polynomial time blackbox PIT algorithm for depth-3 ($\Sigma\Pi\Sigma$) circuits is almost as hard as designing PIT algorithm for general circuits. Hence, understanding the depth-3 circuits is crucial to completely derandomize PIT for general circuits. After a long line of work [DS07, KS07, KS09, KS11, SS11, SS12, SS13], we have a poly$(n) \cdot d^k$-time blackbox PIT algorithm for n-variate degree d depth-3 circuits with top fan-in bounded by k. PIT for various special cases of depth-3 circuits are also extensively studied. One such model is depth-3 diagonal ($\Sigma \wedge \Sigma$) circuits, which compute sum of power of linear polynomials. [Sax08] introduced this model and gave a polynomial time whitebox algorithm. Later, blackbox PIT algorithm is also obtained for this model [ASS13, FS13b, FSS14]. Currently, the best known blackbox test for this model runs in time $s^{O(\log\log s)}$, where s is the circuit size [FSS14]. PIT for many other special cases of depth-3 circuits are also studied [SSS13, ASS13, AGKS15, dOSV15]. Like depth-3 circuits, PIT for various special cases of depth-4 circuits are also known

[BMS11, SV11, ASSS12, For15, KS17, PSS18].

Besides circuits, PIT of *read-once oblivious algebraic branching programs* and its variants (for definition, see Section 2.2) are also well-studied. First, [RS05] gave a polynomial time whitebox PIT algorithm for this model. A long line of work has been done on designing blackbox PIT algorithms for ROABPs and its variants [FS13b, FSS14, AGKS15, GKST15, GKS17]. The best known blackbox PIT algorithm for ROABPs is due to [AGKS15]. It runs in $(dwn)^{O\log n}$ time, where n is the number of variables, d is the individual degree and w is the width of ROABP. [GKS17] gave a blackbox algorithm of running time $dn^{O(\log w)}$ for ROABPs with known variable order. It is currently the best known algorithm for this model. For constant width, it is a polynomial time algorithm. For commutative ROABPs, the best known algorithm ($(dnw)^{O(\log\log w)}$) is due to [GKS17].

For more details about PIT and its applications, see surveys [Sax09, Sax13, SY10] and theses [Sap13b, For14, Gur15].

1.2 Contribution of this thesis

Now we describe our results in PIT. For notations and definitions, see Sections 2.1 and 2.2 of Chapter 2.

1.2.1 Bootstrapping in PIT

We study the phenomenon of *bootstrapping*: converting an HSG for n-variate size s degree s circuits to an HSG for $L(n)$-variate size s degree s circuits with $L(n) > n$. In boolean settings, this phenomenon is well understood. The analog of HSG in boolean settings is *pseudo-random generator (PRG)*. Informally, PRG is a "easily" computable function which stretches a given random string (called *seed*) to a "longer" pseudorandom string which still "looks" random for "small" size circuits (see [AB09, Chapter 20]). It is a well-studied object in the boolean world [Yao82, BM84, NW94, IW97, STV01, Uma03]. By [NW94, Section 2-3], it is known that a PRG for $n = O(\log s)$-variate size s boolean circuits (with seed length less than n) can be converted to a PRG for size s circuits with $L(n) = 2^{\Theta(n)}$

many variables. No further reduction in number of variables is possible since the size of any circuit on fewer than $\log s$ variables has a circuit of size smaller than s.

The situation is less clear in algebraic settings. On one hand, for every n and s, we have n-variate polynomials requiring circuits of size s (since polynomials can have arbitrarily large degrees unlike boolean settings where every function is multilinear). On the other hand, bootstrapping from $O(\log s)$ variables to s variables is not studied explicitly in the literature.

We bridge this gap in knowledge by showing that an HSG for $(\log^{\circ c} s)$-variate size s degree s circuits can be efficiently converted to an HSG for s-variate size s degree s circuits. Furthermore, at the cost of making the final HSG "slightly" superpolynomial $(s^{\exp \circ \exp(O(\log^\star s))})$, we show that bootstrapping can be done from even a *constant* number of variables! Our results can also be viewed as a powerful *amplification* of derandomization: a "slight" derandomization (s^{n^δ}-degree HSG for n-variate size s degree s circuits, for a constant $\delta < 1/2$) implies "nearly" complete derandomization $(s^{\exp \circ \exp(O(\log^\star s))})$ of PIT. Compare the required s^{n^δ}-degree HSG with the $(s+1)^n$-degree HSG due to trivial derandomization of [DL78, Zip79, Sch80] (see Section 2.3.1).

We prove an additional result for shallow circuits: poly(s)-time computable $o(s^{n/2})$-degree HSG for n-variate size s depth four circuits (for some constant $n \geq 3$) implies quasi-polynomial time ($s^{O(\log s)}$) blackbox PIT for size s degree s circuits.

With all these bootstrapping results, we also get a circuit lower bound result in conclusion, which we refer as Conjecture 1 in the formal statements. It says about a separation between two algebraic classes (VP$\neq$ VNP) or two boolean classes (E$\not\subseteq$P/poly). Later, in Section 3.1 of Chapter 3, we give a detailed explanation of Conjecture 1.

Now we give the formal statements of our bootstrapping results. Since hitting sets are more frequently used notion in the context of blackbox PIT compared to hitting set generator, we describe our bootstrapping theorems in terms of hitting set. However, in all the statements, one can replace hitting set with hitting set generator. In that case, size of hitting set will be replaced by the degree of hitting set generator.

Theorem (Perfect Bootstrapping, Theorem 3.1.1)**.** *Let c be a positive integer. For all* *sufficiently large s, suppose that we have a poly(s)-time computable hitting set for $\lceil \log^{\circ c} s \rceil$-variate polynomials computed by size s degree s circuits. Then, we have a poly(s)-time computable hitting set for the class of s-variate polynomials computed by size s degree s circuits and Conjecture 1 holds.*

Next, we describe the bootstrapping of PIT for constant variate circuits to a "slightly" super-polynomial PIT for general circuits.

Theorem (Constant Bootstrapping, Theorem 4.1.1). *Let $e \geq 2$ and $1 > \epsilon \geq (3 + 6\log(128e^2))/(128e^2)$ be some constants. Let n be a constant greater than or equal to $\lceil \max\{192e^2 \log(128e^2)^{1/\epsilon}, (64e^2)^{1/\epsilon}\} \rceil$. For all sufficiently large s, suppose that we have an s^e-size poly(s)-time computable hitting set for n-variate degree s polynomials computed by size s circuits. Then, we have an $s^{\exp \circ \exp(O(\log^\star s))}$-time computable hitting set for the class of s-variate polynomials computed by size s degree s circuits and Conjecture 1 holds.*

For $e = 2$ and $\epsilon = 6912/6913$, the hypothesis of the above theorem needs a s^2-size poly(s)-time computable hitting set for 6913-variate size s circuits computing degree s polynomials. Our previous theorem implies the following one, where we describe a powerful amplification of derandomization for PIT.

Theorem (Theorem 4.1.2). *Let δ be a constant less than $1/2$. For some sufficiently large constant n and all $s \geq n$, suppose that we have an s^{n^δ}-size poly(s)-time computable hitting set for all n-variate polynomials computed by size s degree s circuits. Then, we have an $s^{\exp \circ \exp(O(\log^\star s))}$-time computable hitting set for the class of s-variate polynomials computed by size s degree s circuits and Conjecture 1 holds.*

Remark. In terms of the number of variables n, the hitting set in the hypothesis could be computable in $s^{\mu(n)}$, where $\mu(\cdot)$ is an arbitrary function.

Next, we describe the bootstrapping phenomenon for shallow depth circuits.

Theorem (Shallow Bootstrapping, Theorem 5.1.1). *Let $n \in \mathbb{N}$ be a constant greater than or equal to 3. For all sufficiently large s, suppose that we have a $o(s^{n/2})$-size poly(s)-time computable hitting set for n-variate polynomials computed by size s depth-4 circuits. Then,* *we have an $s^{O(\log s)}$-time computable hitting set for the class of s-variate polynomials computed by size s degree s circuits and Conjecture 1 holds.*

We see our results as a positive development; since they reduce PIT to cases that are special in an unprecedented way. Such special-case PIT algorithms are waiting to be discovered.

1.2.2 Blackbox PIT for certain log-variate models

Our work on bootstrapping in PIT inspires us to study circuits with "few" variables. For example, even a poly(s)-time blackbox PIT for ($\log\log s$)-variate size s depth-4 circuits would lead to a quasi-polynomial time blackbox PIT for general circuits. It is a weaker version of our Shallow Bootstrapping result. The hypothesis in our example ensures a poly(s)-time computable s^e-size, for some constant e, hitting set for ($\log\log s$)-variate size-s depth-4 circuits. Let n be a constant greater than $2^{2^{4e}}$. Then for all $s \geq n$, we have a poly(s)-time computable s^e-size hitting set for $4e$-variate size s depth-4 circuits. To satisfy the hypothesis of Shallow Bootstrapping, we need a poly(s)-time computable $o(s^{2e})$-size hitting set for $4e$-variate size s depth-4 circuits, and it is satisfied by the hypothesis of our example.

Consider the circuit size s and the number of variables n to be independent parameters. Then, prior to our work on Shallow Bootstrapping, we wanted a poly(sn)-time blackbox PIT algorithm for depth-4 circuits to get a quasi-polynomial time blackbox PIT algorithm for general circuits [AV08]. However, the previous example says that now a poly(s)$\cdot 2^{2^n}$-time PIT algorithm is sufficient for the same conclusion. This reduces the dependence of PIT algorithms on the number of variables in a significant way and motivates us to study low variate circuits.

We study the blackbox PIT question for $n = O(\log s)$-variate size s circuits computing polynomials of poly(s)-dimensional partial derivative space. The study of the partial derivative space in algebraic circuit complexity was initiated by Nisan and Wigderson [NW97] to prove lower bounds for some circuit classes. Later, it was studied in many other works both on circuit lower bounds and PIT algorithms [GR98, SW01, Raz09, RY09, Kay10, FS13b]. However, no polynomial blackbox PIT was known for the previously mentioned circuit model. We first give a poly$(s2^n)$-time blackbox PIT algorithm for that circuit class.

Theorem (Theorem 6.1.1)**.** *Let $\mathbb{F}$ be a field of characteristic 0 or greater than d. Let $\mathcal{P}$ be a set of n-variate degree d polynomials, over $\mathbb{F}$, computed by circuits of size s such that for all $P \in \mathcal{P}$, the dimension of the partial derivative space of P is at most k. Then, blackbox PIT for $\mathcal{P}$ can be solved in* $\mathrm{poly}(sdk) \cdot \left(\frac{3n}{\log k}\right)^{O(\log k)}$ *time.*

Note that for $k = \mathrm{poly}(sd)$ and $n = O(\log sd)$, the time complexity of the algorithm becomes poly(sd), and we get a polynomial time blackbox PIT algorithm for log-variate circuits (i.e., number of variables is logarithmic with respect to the circuit size) computing polynomials with poly(sd)-dimensional partial derivative space. This was not known before our work. Prior to our work, [FSS14] gave a poly$(s) \cdot (ndk)^{O(\log \log k)}$-time blackbox PIT algorithm for $\mathcal{P}$. Unlike our algorithm, in the log-variate case the running time of their algorithm remains super-polynomial.

In particular, depth-3 diagonal circuit is a prominent model which computes polynomial with polynomially large (with respect to the circuit size) dimensional partial derivative space. Hence, the above theorem gives a polynomial time PIT algorithm for log-variate depth-3 diagonal circuits. Prior to our work, the $s^{O(\log \log s)}$-time algorithm due to [FSS14] was the best known blackbox PIT algorithm for this model. We give a polynomial time blackbox PIT algorithm for a more general class of depth-3 diagonal circuits, i.e. log-rank depth-3 diagonal circuits. For the definition of log-rank depth-3 diagonal circuit, see Section 2.2 of Chapter 2. From the definition of log-rank depth-3 diagonal circuits, it easy to see that they subsume log-variate depth-3 diagonal circuits. Later, we show a polynomial-time computable reduction from *nonzero* log-rank depth-3 diagonal circuits to *nonzero* log-variate depth-3 diagonal circuits (Lemma 6.2.4). This reduction combined with Theorem 6.1.1 gives a polynomial time blackbox PIT algorithm for log-rank depth-3 diagonal circuits.

Theorem (Theorem 6.1.2)**.** *Let $\mathbb{F}$ be a field of characteristic 0 or $> d$. Let $\mathcal{D}$ be the set* *of n-variate degree d polynomials computed by size s depth-3 diagonal circuits with rank $O(\log sd)$. Then, blackbox PIT for $\mathcal{D}$ can be solved in poly(sd)-time.*

1.2.3 Cone closed bases: A stronger notion of rank concentration

As mentioned earlier, for solving PIT, we can assume that the top gate (output gate) of the input circuit is an addition gate. Hence, any polynomial f computed by such a circuit can be written as $f(\mathbf{x}) = \sum_{i=1}^{k} c_i f_i$, where c_i's are constants from the underlying field $\mathbb{F}$ and f_i's are polynomials computed by simpler circuits (i.e. lower depth and smaller size circuits). Let $P(\mathbf{x})$ be the polynomial over $\mathbb{F}^k$ defined as $(f_1, \ldots, f_k)^\intercal$ and $\mathbf{c} = (c_1, \ldots, c_k)^\intercal$. Then f can be written as $\mathbf{c}^\intercal \cdot P(\mathbf{x})$. A similar representation is also available for polynomials computed by ROABPs and its variants. Many PIT algorithms are designed by studying the structure of $P(\mathbf{x})$ for various classes of polynomials [RS05, ASS13, FSS14, AGKS15, GKST15, GKS17]. The main object studied in all these works is the *coefficient space* of P which is denoted by sp(P).

Suppose that the coefficient space sp(P) is generated by a "small" set of monomials S. We describe this phenomenon as the rank of the sp(P) is concentrated on S. Let f_S and P_S be the projections of f and P, respectively, on the set of monomials S. Then $f \neq 0$ if and only if $f_S = \mathbf{c}^\intercal \cdot P_S \neq 0$. This says that if f is a nonzero polynomial then it has a monomial from S with nonzero coefficient. Therefore, to test nonzeroness of f, it is sufficient to design a hitting set for the set of polynomials which have a monomial from S with nonzero coefficient. In many places, this approach helps us to design efficient blackbox PIT algorithms (may not be polynomial-time, but far better than the trivial one) for various classes of polynomials. One frequently used S in PIT literature is the set of "low-support" monomials, by which we mean monomials with support size poly$(\log k)$. There is also a standard way to design efficient hitting set for polynomials having a low-support monomial with nonzero coefficient (Lemma 2.3.14). However, in general, P may not have low-support concentration property. We may need to apply some linear transformation φ (e.g., shifting variables), such that the new polynomial $\varphi(P)$ achieves that property. For example, the rank of the coefficient space of $(\prod_{i=1}^{n} x_i, 0, \ldots, 0)^\intercal$ is not concentrated on the low-support monomials. However, after shifting each variable by 1, in the new polynomial $(\prod_{i=1}^{n}(x_i+1), 0, \dots, 0)^{\intercal}$ constant term spans the whole coefficient space. [ASS13] introduced the concept of *rank concentration*. Many results on PIT are obtained by achieving low-support rank concentration [ASS13, FSS14, AGKS15, GKST15, GKS17].

One way of strengthening low-support rank concentration is through *low-cone rank concentration*. The cone of a monomial m is the set of monomials dividing m and the cone-size is the cardinality of that set. In low-cone rank concentration, the rank of $\mathrm{sp}(P)$ is concentrated in the coefficients of "low-cone" monomials, by which we mean monomials with cone-size $\mathrm{poly}(k)$. Low-cone concentration property on P implies the existence of a low-cone monomial in f with nonzero coefficient. It is not hard to show that low-cone concentration also implies low-support concentration. Hence, similar to low-support concentration, Lemma 2.3.14 also gives an efficient PIT for f when P has low-cone concentration property. However, this algorithm remains super-polynomial in log-variate regime, and our work on designing polynomial time blackbox PIT for log-variate depth-3 diagonal circuits can be seen as an improvement over it. There, we develop a method which gives a different blackbox PIT for polynomials having a low-cone monomial with nonzero coefficient, and the running time becomes polynomial in log-variate case.

We further strengthen low-support rank concentration by introducing *cone-closed basis*. We say P has a cone-closed basis, if there is a set of monomials B whose coefficients form a basis of $\mathrm{sp}(P)$ and B is closed under sub-monomials.. Cone-closed basis also strengthens low-cone concentration (Lemma 7.2.1). The definition of cone-closed basis is motivated by the polynomials of the following form: $\sum_{i=1}^{k} c_i(1+a_{i1}x_1+\cdots+a_{in}x_n)^d$. These polynomials are computed by a special type of depth-3 diagonal circuits, and they can be written as follows: $\mathbf{c}^{\intercal}\cdot D(\mathbf{x})$, where $\mathbf{c}^{\intercal}=(c_1,\dots,c_k)$ and $D(\mathbf{x})=(\mathbf{1}+\ell_1x_1+\cdots+\ell_nx_n)^d\in\mathbb{F}^k[\mathbf{x}]$. We show that polynomials (over $\mathbb{F}^k$) of form $D(\mathbf{x})$ have cone-closed property naturally (Lemma 7.2.7). Currently, we do not know any interesting application of cone-closed basis in designing PIT algorithms. However, in the following result, we relate cone-closed basis with Basis Isolating Weight Assignment (Definition 7.1.2), another well studied concept in PIT. It was introduced by [AGKS15] and also used in many other subsequent works

[GKST15, GKS17]. Here, we show that a polynomial P over $\mathbb{F}^k$, when shifted by a basis isolating weight assignment, becomes cone-closed. It is currently the best known rank concentration result for polynomials over vector spaces. Our result strengthens some previously proven properties like a polynomial over $\mathbb{F}^k$ when shifted by formal variables becomes low-support concentrated [FSS13, Corollary 3.22] (extended version of [FSS14]) or, when shifted by a basis isolating weight assignment becomes low-support concentrated [GKST15, Lemma 5.2].

Theorem (Theorem 7.1.3)**.** *Let $P(\mathbf{x})$ be an n-variate degree d polynomial over $\mathbb{F}^k$, and* $\mathrm{char}(\mathbb{F}) = 0$ *or* $> d$*. Let $\mathbf{w} = (w_1, \ldots, w_n) \in \mathbb{N}^n$ be a basis isolating weight assignment for $P(\mathbf{x})$. Then, $P(\mathbf{x} + t^{\mathbf{w}}) := P(x_1 + t^{w_1}, \ldots, x_n + t^{w_n})$ has a cone-closed basis over $\mathbb{F}(t)$.*

1.3 Follow up works

In follow up works, [KST18] and [GKSS19] explored the phenomenon of bootstrapping in PIT and improved our Theorem 4.1.2. They resolved some of the main open questions in our work [AGS18]. First, [KST18] improved Theorem 4.1.2 by weakening the hypothesis from s^{n^δ} to $s^{n-\epsilon}$, where ϵ can be any constant greater than 0. Their proof also works for other weaker models of computation like formula, algebraic branching programs. Later, [GKSS19] strengthened the theorem by giving a construction of polynomial-time computable hitting set in the conclusion. For more detailed discussion, see Section 4.4 of Chapter 4.

1.4 Organization of the thesis

In Chapter 2, we discuss all the basic notations and the known results used in this thesis. In the next three consecutive chapters, we discuss the bootstrapping phenomenon in PIT. In Chapters 3, 4 and 5, we prove Perfect Bootstrapping, Constant Bootstrapping and Shallow Bootstrapping, respectively. In Chapter 6, we give a polynomial-time blackbox PIT algorithm for log-variate circuits which compute polynomials of low-dimensional partial derivative space. In Chapter 7, we introduce the concept of cone-closed basis and show that if a polynomial over a vector space is shifted by its basis isolating weight assignment, the new polynomial becomes cone-closed.

Chapter 2

Preliminaries

In this chapter, we mention some known results in PIT that will be useful in our work. We also describe some necessary notations and definitions that we use in this thesis.

2.1 Notations and Definitions

We use $\mathbb{Q}$ to denote the field of rational numbers, $\mathbb{C}$ to denote the field of complex numbers, $\mathbb{F}$ to denote a general field, $\mathbb{Z}$ to denote the set of integers, and $\mathbb{N}$ to denote the set of natural numbers, i.e., $\{1, 2, \ldots\}$. For a prime number q and a positive integer t, by $\mathbb{F}_{q^t}$ we denote the finite field of size q^t. For a prime q, the set of integers $\{0, 1, \ldots, q-1\}$ forms a field under addition and multiplication modulo q, and we denote it by $\mathbb{F}_q$. For an k-tuple $(a_1, \ldots, a_k)$,

$$(a_1, \ldots, a_k)^\intercal = \begin{pmatrix} a_1 \\ \vdots \\ a_k \end{pmatrix}.$$

For a field $\mathbb{F}$, $\mathbb{F}[x]$ denotes the *ring of polynomials* in a variable x and the coefficients are from $\mathbb{F}$. By $\mathbb{F}(x)$ we denote the *field of all rational functions* in a variable x. The *characteristic* of a field $\mathbb{F}$, denoted by *char*($\mathbb{F}$), is the smallest positive integer k such that for all $a \in \mathbb{F}$, $\underbrace{a + \cdots + a}_{k\text{-times}} = 0$. If no such k exists, then we say $\mathbb{F}$ is a field of characteristic *zero*. The characteristic of field can be zero or a prime number. For basic definitions regarding field and field extension see [Her06, Section 3.2 and Chapter 5].

For a given positive integer n, by $[n]$ we denote the set of natural numbers $\{1, \ldots, n\}$. For a set S, $|S|$ denotes the cardinality of S. For a set S, and a $k \in \mathbb{N}$, $\binom{S}{k}$ denotes the set of all k-size subsets of S. Let f be a mapping from A to B. Then by $\mathrm{Img}(f)$ we denote the *image* of f, which is $\{f(a) \mid a \in A\}$. For a $b \in B$, $f^{-1}(b)$ is the *preimage* of b, i.e., $\{a \in A \mid f(a) = b\}$.

For a positive integer n, $n!$ denotes the expression $\prod_{i=1}^{n} i$. For $n = 0$, $n!$ is defined as 1. For two nonnegative integers $n \geq k$, $\binom{n}{k} = \frac{n!}{k!(n-k)!}$. When $n < k$, $\binom{n}{k}$ is defined as 0. Let $\mathbf{a} = (a_1, \ldots, a_n), \mathbf{b} = (b_1, \ldots, b_n) \in \mathbb{N}^n$. Then $\binom{\mathbf{a}}{\mathbf{b}}$ denotes $\prod_{i=1}^{n} \binom{a_i}{b_i}$. For a nonnegative integer n and a set of nonnegative integers $k_1, \ldots, k_\ell$ with $k_1 + \cdots + k_\ell = n$,

$$\binom{n}{k_1, \ldots, k_\ell} = \frac{n!}{k_1! \cdots k_\ell!}.$$

By $\mathbf{x}$, we denote the set of variables $\{x_1, \ldots, x_n\}$. For any $\mathbf{e} = (e_1, \ldots, e_n) \in \mathbb{N}^n$, $\mathbf{x}^{\mathbf{e}}$ denotes the monomial $\prod_{i=1}^{n} x_i^{e_i}$. Sometimes we may use only $\mathbf{e}$ to denote the monomial $\mathbf{x}^{\mathbf{e}}$. The *degree* of a monomial $\mathbf{x}^{\mathbf{e}}$ is $= \sum_{i=1}^{n} e_i$ and denoted by $\deg(\mathbf{x}^{\mathbf{e}})$. For $n, d \in \mathbb{N}$, $M_{n,d} = \{\mathbf{e} \in \mathbb{N}^n \mid \sum_{i=1}^{n} e_i \leq d\}$, which denotes the set of exponent vectors of n-variate monomials with degree$\leq d$.

A *weight assignment* on the set of variables is a mapping from $\mathbf{x}$ to $\mathbb{N}$. A weight assignment $\mathbf{w}(x_i) := w_i$ for all $i \in [n]$, is denoted by $(w_1, \ldots, w_n)$. It extends to monomials as follows: for a monomial $m = \mathbf{x}^{\mathbf{e}}$, $\mathbf{w}(m) := \sum_{i=1}^{n} e_i w_i$. Similarly, for a set of monomials B, the weight of B is $\mathbf{w}(B) := \sum_{m \in B} \mathbf{w}(m)$.

For a field $\mathbb{F}$, $\mathbb{F}[\mathbf{x}]$ denotes the ring of polynomials over the variables $\{x_1, \ldots, x_n\}$ and the coefficients from $\mathbb{F}$. For a monomial m and a polynomial f, by $\mathrm{coef}_m(f)$ we denote the coefficient of m in f. For a polynomial f, the *support* of f is the set of monomials which have nonzero coefficients in f, and it is denoted by $\mathrm{support}(f)$. The cardinality of $\mathrm{support}(f)$ is called the *sparsity* of f. By degree d polynomial, we mean the degree of each monomial in the support is$\leq d$. We say f is a polynomial with *individual degree d*, if for every monomial $\prod_{i=1}^{n} x_i^{e_i}$ in the support of f, $e_i \leq d$. A degree 1 polynomial is called *linear polynomial.* A linear polynomial is called *linear form*, if its constant term is zero. A polynomial with individual degree 1 is called *multilinear polynomial.*

For basic definitions in linear algebra like vector space, linear independence, basis, dimension, see [Her06, Sections 4.1 and 4.2]. For a vector space V, $\dim V$ denotes its dimension. By $\mathbb{F}^k$ we denote the vector space $\{(a_1, \dots, a_k)^\intercal \mid a_i \in \mathbb{F}\}$, where the addition between two elements is defined by coordinate wise addition, and the scalar multiplication of an element is defined by multiplying each coordinate by the scalar. Suppose that $\mathbf{a} = (a_1, \dots, a_k)^\intercal$ and $\mathbf{b} = (b_1, \dots, b_k)^\intercal$ be two elements from $\mathbb{F}^k$. Then by $\mathbf{a}^\intercal \cdot \mathbf{b}$ we denote the expression $\sum_{i=1}^k a_i b_i$. For any two positive integers, m and n, $\mathbb{F}^{m\times n}$ denotes the set of all $m \times n$ matrices over $\mathbb{F}$. It also forms a vector space over $\mathbb{F}$ under matrix addition and scalar multiplication.

In this thesis, we also study multivariate polynomials (over variables $\mathbf{x}$) whose coefficients are coming from $\mathbb{F}^k$. We denote this set by $\mathbb{F}^k[\mathbf{x}]$. For a polynomial $P \in \mathbb{F}^k[\mathbf{x}]$, the *support* of P is the set of monomials whose coefficients in P are nonzero vector. For a polynomial $P(\mathbf{x}) = \sum_{\mathbf{e}} \mathbf{c}_{\mathbf{e}} \mathbf{x}^{\mathbf{e}}$ over $\mathbb{F}^k$ and $\mathbf{a} \in \mathbb{F}^k$, $\mathbf{a}^\intercal \cdot P(\mathbf{x})$ denotes the polynomial $\sum_{\mathbf{e}} (\mathbf{a}^\intercal \cdot \mathbf{c}_{\mathbf{e}}) \mathbf{x}^{\mathbf{e}}$ (over $\mathbb{F}$). Let $P(\mathbf{x})$ be a polynomial from $\mathbb{F}^k$. For all $i \in [k]$, let P_i be the polynomial (over $\mathbb{F}$) defined as follows: for every monomial m, $\mathrm{coef}_m(P_i)$ is the projection of $\mathrm{coef}_m(P)$ at ith coordinate. Then P can also be seen as $(P_1, \dots, P_k)^\intercal$, i.e., a vector of polynomials. Sometimes we can use this representation for P. The *coefficient space* of P is the subspace of $\mathbb{F}^k$ generated by the coefficients of P, and it is denoted by $\mathrm{sp}(P)$. For a set of monomials B, we say *B is a basis of P*, if the coefficients of all the monomials in B form a basis for $\mathrm{sp}(P)$.

For a finite set B and a vector $\mathbf{u} \in \mathbb{F}^{|B|}$, the coordinates of $\mathbf{u}$ can be indexed by the elements of B and for an element $e \in B$, by $\mathbf{u}_e$ we denote the value at coordinate indexed by e. Suppose that A is a matrix whose rows are indexed by the elements of U and the columns are indexed by the elements of V. Then for any two subsets $U' \subseteq U$ and $V' \subseteq V$, $A_{U',V'}$ denotes the submatrix formed by taking the rows indexed by S' and the columns indexed by V'. For any elements $e \in V'$, $A_{U',e}$ denote the column of $A_{U',V'}$ indexed by e.

For a monomial $\mathbf{x}^{\mathbf{e}}$, $\partial_{\mathbf{x}^{\mathbf{e}}}(f)$ denotes the *partial derivative* of f with respect to the monomial $\mathbf{x}^{\mathbf{e}}$. For a polynomial f over $\mathbb{F}$, by $\langle \partial^{<\infty}(f) \rangle$ we denote the vector space (over

$\mathbb{F}$) formed by taking all possible linear combinations of the partial derivatives of f, and we call it *partial derivative space* of f. For a nonnegative integer k and a polynomial f, $\langle \partial^{=k}(f) \rangle$ denotes the vector space formed by taking all possible linear combinations of the partial derivatives of f corresponding to exactly k-degree monomials.

By (t,d)-*hitting set*, we mean a hitting set of size d and computable in time t. By $\mathcal{P}[n,d,s]$, we denote the set of n-variate degree d polynomials computed by size s circuits. We say a polynomial family $\{q_n\}_{n\geq 1}$ is $t(n)$-*time computable*, if there exists a turing machine M such that given n as unary (i.e., n is given as 1^n), M computes q_n in $t(n)$-time as a form of depth-2 circuits, i.e., sum of monomials.

Some functions and asymptotic notations: For a function $t : \mathbb{N} \to \mathbb{N}$, by $\tilde{O}(t(n))$ we denote the class of functions which are in $O(t(n) \log^k n)$ for some $k \in \mathbb{N}$. By $\exp \circ \exp(t(n))$ we denote the function $2^{2^{t(n)}}$. We say $t(n)$ is a *quasi-polynomial* function if $t(n) = n^{\mathrm{poly(\log n)}}$. For any $c \in \mathbb{N}$, $\log^{\circ c} n$ denotes c-times composition of logarithmic function (in base 2) with itself. For example, $\log^{\circ 2} n = \log\log n$. By $\log^{\star} s$ we denote the smallest integer i such that $\log^{\circ i} n \leq 1$. For any $c \in \mathbb{N}$, $\exp^{\circ c}(n)$ is the c-times composition of the exponential function (in base 2) with itself. For example, $\exp^{\circ 2}(n) = 2^{2^n}$.

Support and cone of a monomial: For a monomial $\mathbf{x}^{\mathbf{e}}$, the *support* of $\mathbf{x}^{\mathbf{e}}$ is the set $\{x_i \mid e_i > 0\}$, (i.e., the set variables whose exponents are positive in $\mathbf{x}^{\mathbf{e}}$) and the *support size* of $\mathbf{x}^{\mathbf{e}}$ is the cardinality of that set. For an n-tuple $\mathbf{f} = (f_1, \ldots, f_n)$, we say $\mathbf{f} \leq \mathbf{e}$ if $f_i \leq e_i$ for all $i \in [n]$. Similarly, we say $\mathbf{f} \geq \mathbf{e}$ if $f_i \geq e_i$ for all $i \in [n]$. A monomial $\mathbf{x}^{\mathbf{f}}$ is called a *sub-monomial* (respectively, *super-monomial*) of $\mathbf{x}^{\mathbf{e}}$, if $\mathbf{f} \leq \mathbf{e}$ (respectively, $\mathbf{f} \geq \mathbf{e}$). The *cone* of a monomial $\mathbf{x}^{\mathbf{e}}$ is the set of sub-monomials of $\mathbf{x}^{\mathbf{e}}$, which is also same as the set of monomials which divide $\mathbf{x}^{\mathbf{e}}$, and the *cone size* of $\mathbf{x}^{\mathbf{e}}$ is the cardinality of that set. The cone-size of $\mathbf{x}^{\mathbf{e}}$ is $\prod_{i=1}^{n}(e_i+1)$. We say $\mathbf{x}^{\mathbf{f}}$ is a *proper sub-monomial* (respectively, *proper super-monomial*) of $\mathbf{x}^{\mathbf{e}}$ if $\mathbf{f} \leq \mathbf{e}$ and $\mathbf{f} \neq \mathbf{e}$ (respectively, $\mathbf{f} \geq \mathbf{e}$ and $\mathbf{f} \neq \mathbf{e}$). We say a monomial is a ℓ-support monomial if its support size is $\leq \ell$. Similarly, a monomial is a k-cone monomial if its cone size is $\leq k$.

Rank concentration: Suppose that P is a polynomial over the vector space $\mathbb{F}^k$. We say P has *ℓ-support rank concentration*, if the coefficients of ℓ-support monomials in P span the coefficient space. Similarly, we say P has *k-cone rank concentration*, if the coefficients of k-cone monomials span the coefficient space of P. Sometimes, informally, by *low-support rank concentration* we mean poly$(\log k)$-support rank concentration, and by *low-cone rank concentration* we mean poly(k)-cone rank concentration.

Monomial ordering: A *monomial ordering* $\prec$ is a total order on the set of all monomials over $\mathbf{x}$ such that

1. for all $\mathbf{a} \in \mathbb{N}^n \setminus \{\mathbf{0} = (0, \ldots, 0)\}$, $1 \prec \mathbf{x}^{\mathbf{a}}$.

2. for all $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{N}^n$, if $\mathbf{x}^{\mathbf{a}} \prec \mathbf{x}^{\mathbf{b}}$ then $\mathbf{x}^{\mathbf{a}+\mathbf{c}} \prec \mathbf{x}^{\mathbf{b}+\mathbf{c}}$.

For a nonzero polynomial f, the *leading monomial* (with respect to a monomial ordering $\prec$) is the largest monomial in the support of f. To know more about monomial ordering, see [CLO15, Chapter 2].

VP and VNP: A polynomial family $\{f_n\}_{n\geq 1}$ is in VP if there exists a polynomially bounded function $t : \mathbb{N} \to \mathbb{N}$ such that for every n, the number of variables in f_n is at most $t(n)$ and it is computed by a $t(n)$-size $t(n)$-degree circuit. We say a polynomial family $\{f_n\}_{n\geq 1}$ is in VNP if there exist two polynomially bounded function $t, u : \mathbb{N} \to \mathbb{N}$ and a polynomial family $\{g_n\}_{n\geq 1}$ in VP such that for every n,

$$f_n(x_1, \ldots x_{u(n)}) = \sum_{w \in \{0,1\}^{t(n)-u(n)}} g_n(x_1, \ldots, x_{u(n)}, w_{u(n)+1}, \ldots, w_{t(n)}).$$

An interesting polynomial family in VP is the family of determinants,

$$\mathrm{Det}_n(X) = \sum_{\sigma \in S_n} \mathrm{sgn}(\sigma) \prod_{i=1}^{n} x_{i,\sigma(i)},$$

where $X = (x_{i,j})$ is an $n \times n$ matrix, S_n is the set of all permutations of $[n]$ and $\mathrm{sgn}(\sigma)$ is the signature of the permutation σ. The canonical example for a family in VNP is the family of permanents,

$$\mathrm{Perm}_n(X) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} x_{i,\sigma(i)}.$$

The classes VP and VNP can be seen as algebraic analogue of P and NP, respectively. These two algebraic classes were introduced by Valiant [Val79] and he conjectured that VP$\neq$VNP.

E and #P/poly: A boolean function $f : \{0,1\}^* \rightarrow \{0,1\}$ is in E if there exists a Turing machine M such that for a given binary string x of length n, M computes $f(x)$ in time $2^{O(n)}$. A family of functions $\{f_n\}_{n\geq 1}$, where $f_n : \{0,1\}^n \rightarrow \mathbb{N}$, is in *#P/poly*, if there exists a polynomially bounded function $t : \mathbb{N} \rightarrow \mathbb{N}$ and a family of boolean functions $\{g_n\}_{n\geq 1}$ in P/poly such that for every n and for all $\mathbf{a} = (a_1, \ldots, a_n) \in \{0,1\}^n$,

$$f_n(\mathbf{a}) = \left| \left\{ b \in \{0,1\}^{t(n)-n} \; : \; g_{t(n)}(a_1, \ldots, a_n, b_1, \ldots, b_{t(n)-n}) = 1 \right\} \right|.$$

The domain of both the classes of functions is $\{0,1\}^*$, however the range for the functions in E is a subset of the range for the functions in #P/poly. Hence, it would be interesting to know whether E$\subseteq$#P/poly.

E-computable polynomial family: Let $\{f_n\}_{n\geq 1}$ be a multilinear polynomial family with integral coefficients and the number of variables in f_n is $O(n)$. We say $\{f_n\}_{n\geq 1}$ is an *E-computable* polynomial family if there exists a Turing machine M such that for a given n as unary, M computes f (as a sum of monomials, i.e., form of depth-2 circuit) in time $2^{O(n)}$. If the number variables of f_n is m, then f_n induces a function g_n from $\{0,1\}^m$ to $\mathbb{Z}$, called *coefficient function*, defined as follows: for all $\mathbf{b} \in \{0,1\}^n$, $g_n(\mathbf{b})$ is the coefficient of the monomial $\mathbf{b}$ in f_n.

2.2 Models of Computation

Now, we describe various models of computation, which are referred in this thesis.

Depth-3 circuits: Depth-3 circuits compute polynomials of form

$$C(\mathbf{x}) = \sum_{i=1}^{k} \prod_{j=1}^{a} \ell_{ij},$$

where ℓ_{ij}'s are linear polynomials over the underlying field $\mathbb{F}$. The standard notation to denote depth-3 circuits is $\Sigma\Pi\Sigma$. Sometimes we use $\Sigma^k\Pi^a\Sigma$ to denote the upper bound in fan-in of gates in the respective layers. In this thesis, we always assume the fan-out of each gate in a depth-3 circuit is at most 1. This implies that the *degree* of the polynomial computed by a depth-3 circuit is bounded by the size of the circuit.

Depth-3 diagonal circuits: Depth-3 diagonal circuits compute polynomials of form

$$C(\mathbf{x}) = \sum_{i=1}^{k} \ell_i^{d_i},$$

where ℓ_i's are the linear polynomials over the underlying field $\mathbb{F}$. We denote the class of depth-3 diagonal circuits by $\Sigma \wedge \Sigma$. For all $i \in [k]$, let f_i be the homogeneous degree 1 part of ℓ_i. Then the *rank* of a depth-3 diagonal circuit, denoted by $\mathrm{rk}(C)$, is the dimension of the subspace (over $\mathbb{F}$) generated by f_i's. Note that the rank of C can be equal or one less than the dimension of the subspace generated by ℓ_i's. By *log-variate* depth-3 diagonal circuit we mean the class of depth-3 diagonal circuits where the number variables is at most logarithmic with respect to the circuit size. Similarly, by *log-rank* depth-3 diagonal circuit we mean the class of depth-3 diagonal circuits where the rank of each circuit is at most logarithmic with respect to the circuit size.

Depth-4 Circuits: Depth-4 circuits compute polynomials of form

$$C(\mathbf{x}) = \sum_{i=1}^{k} \prod_{j=1}^{a} Q_{ij},$$

where Q_{ij}'s are degree b polynomials over the underlying field $\mathbb{F}$. The standard notation to denote depth-4 circuits is $\Sigma\Pi\Sigma\Pi$. Sometimes we use $\Sigma^k\Pi^a\Sigma\Pi^b$ to denote the upper bound in fan-in of gates in the respective layers. Like depth-3 circuits, in this thesis, we also assume the fan-out of each gate in a depth-4 circuit is at most 1. This implies the *degree* of the polynomial computed by a depth-4 circuit is bounded by the size of the circuit.

Read-once oblivious branching program: Let $f(x_1, \ldots, x_n)$ be an n-variate polynomial over $\mathbb{F}$. Let π be a permutation on $[n]$. We say f is computed by a width w individual degree d *read-once oblivious branching program (ROABP)* with variable order π, if f can be written as

$$f = a^{\intercal} M_1(x_{\pi(1)}) M_2(x_{\pi(2)}) \cdots M_n(x_{\pi(n)}) b,$$

where $a, b \in \mathbb{F}^{w \times 1}$ and for all $i \in [n]$, $M_i(x_{\pi(i)}) \in \mathbb{F}^{w \times w}[x_{\pi(i)}]$ can be viewed as a polynomial over the matrix algebra with degree less than d. There is also an equivalent definition for ROABP using graphs (see [For14, Section 4.4]). We say f is computed by a width w individual degree d *commutative read-once oblivious branching program*, if each M_i is a polynomial over a commutative sub-algebra of the matrix algebra. For example, consider the coefficients of each M_i are diagonal matrices. All PIT algorithms for ROABPs are designed by analyzing the coefficient space of $M_1(x_{\pi(1)}) M_2(x_{\pi(2)}) \cdots M_n(x_{\pi(n)})$.

2.3 Some known results

In this section, we discuss some known results that we use later in our work.

2.3.1 Randomized PIT algorithm

It is well known that a nonzero univariate degree d polynomial can have at most d roots. Hence, to test nonzeroness of degree d univariate polynomials, we take a set of $d+1$ distinct points and evaluate at those points. If a univariate polynomial evaluates zero on all $d+1$ different points, we output the given polynomial is the zero polynomial. An extension of this fact to the multivariate case is also known due to [DL78, Zip79, Sch80].

Lemma 2.3.1 ([DL78, Zip79, Sch80]). *Let f be an n-variate degree d nonzero polynomial over a field $\mathbb{F}$. Let S be a subset of $\mathbb{F}$. Then*

$$\Pr_{\mathbf{a} \in_r S^n} [f(\mathbf{a}) \neq 0] \geq 1 - \frac{d}{|S|}\,.$$

In the above lemma, if we pick the subset S of size greater than $3d$, then the probability of $f(\mathbf{a}) \neq 0$ becomes greater than $\frac{2}{3}$. This directly gives us a poly(sdn)-time randomized blackbox PIT algorithm for n-variate degree d polynomials computed by size s circuits since given a circuit we can evaluate it at a point in $O(s)$ time. However, when $\mathbb{F}$ is a finite field, we may not be able to pick such a large S. For example, consider the polynomial $x^3 - x$ over $\mathbb{F}_3$. It evaluates to zero at each point in $\mathbb{F}_3$. In such scenarios, we are allowed to pick points from some polynomially large extension of $\mathbb{F}$. A trivial way of derandomizing the above lemma gives us the following hitting set.

Corollary 2.3.2. *For n-variate degree d polynomials over $\mathbb{F}$, there exists a hitting set of size $(d+1)^n$ and computable in poly(d^n) time.*

Proof. Let f be an n-variate degree d nonzero polynomial. Let S be any subset of $\mathbb{F}$ of size $d+1$. If the size of $\mathbb{F}$ is less than $d+1$, we go to an extension $\mathbb{K}$ of $\mathbb{F}$ such that $|\mathbb{K}| \geq d+1$ and take a subset $S \subseteq \mathbb{K}$ of size $d+1$. Then according to the Lemma 2.3.1, at a random point $\mathbf{a}$ from S^n, the probability that $f(\mathbf{a}) \neq 0$ is greater than zero. Therefore, the set S^n must contain a point where f is nonzero. Since in poly(d) time we can construct $\mathbb{K}$, the description of S says that the set S^n is computable in poly(d^n) time. $\square$

Remark. Suppose that $\mathcal{P}_{\mathbb{F}_{q^t}}$ is a class of n-variate degree d polynomials over a finite field $\mathbb{F}_{q^t}$, and they are computed by size s circuits. Due to the size constraint of $\mathbb{F}_{q^t}$, some construction of hitting set for $\mathcal{P}_{\mathbb{F}_{q^t}}$ may need to take points from $\mathbb{K}^n$ where $\mathbb{K}$ is a "small" extension of $\mathbb{F}_{q^t}$, i.e., $|\mathbb{K}| = $ poly(sdn). Let $\mathcal{H} \subseteq \mathbb{K}^n$ be such a hitting set for $\mathcal{P}_{\mathbb{F}_{q^t}}$. Then, in that scenario, we assume that $\mathcal{H}$ is also a hitting set for $\mathcal{P}_{\mathbb{K}}$ where the coefficients of the polynomials will be from $\mathbb{K}$. The motivation for this assumption is that all our known construction of hitting set for various polynomial classes satisfies this property. Hence, it is believable that the property we use to derandomize PIT for $\mathcal{P}_{\mathbb{F}_{q^t}}$ would not change if we consider the coefficients from an extension of $\mathbb{F}_{q^t}$.

2.3.2 Polynomial factorization

One frequently used result in this thesis is Kaltofen's factoring algorithm [Kal89]. It says that for any multivariate polynomial computed by a "small" size circuit, its factors also have "small" size circuits. Formally, they showed the following result.

Lemma 2.3.3. *Let f be an n-variate degree d polynomial over $\mathbb{F}$ computed by a size s circuit C. Let $f = \prod_{i=1}^{r} h_i^{e_i}$, where h_i's are irreducible factors of f. Then each h_i has a circuit C' of size*

$$O(sd^2) + \tilde{O}(d^3).$$

In case, if $p = \mathrm{char}(\mathbb{F})$ divides any e_i, that is $e_i = p^{\hat{e}_i} \cdot \overline{e}_i$ with $\overline{e}_i$ not divisible by p, then C' computes $h_i^{p^{\hat{e}_i}}$.

In most of the cases, the existence of a poly(sd) size circuits h_i's is sufficient for us. Only in Chapter 4, we work with such accurate size bound mentioned in the lemma. For proof, see [Kal89, Section 6] or [Bür13, Theorem 2.21].

2.3.3 PIT vs Lower bound

Next, we describe a technique of constructing hard polynomial from a hitting set. It was shown by [HS80, Agr05]. In general, their approach shows how to construct a polynomial g from the hitting set for a set of polynomial $\mathcal{P}$ such that $g \notin \mathcal{P}$. Now, consider $\mathcal{P}$ is the set of n-variate degree d polynomials computed by size s circuits. Suppose that one can also ensure the polynomial g constructed from the hitting set of $\mathcal{P}$ has the degree and the number of variables is less than d and n, respectively. Then we can say that g is not computable by size s circuits. In this way we use the following lemma to design hard polynomials from hitting sets.

Lemma 2.3.4 (Hitting set to hardness)**.** *Let $\mathcal{H}$ be a (t, d)-hitting set for a set of polynomials $\mathcal{P}$ over n-variables. Let $\delta, n_1 \in \mathbb{N}$ such that $n_1 \leq n$ and $\delta^{n_1} \geq d + 1$. Then there exists an n_1-variate nonzero polynomial $g(\mathbf{x}) \notin \mathcal{P}$ with individual degree less than δ. Furthermore, $g(\mathbf{x})$ is computable by a depth-2 circuit of size less than $2n\delta d$, and the circuit can be constructed in poly(t) time.*

Proof. A natural candidate for $g(\mathbf{x})$ is any polynomial which vanishes on the set of points $\mathcal{H}$, since for every nonzero $h \in \mathcal{P}$, there exists a point $\alpha \in \mathcal{H}$ such that $h(\alpha)$ is nonzero. Let M be the set of first $d+1$ monomials (in lexicographic order) on n_1 variables and the individual degree is less than δ. The set M exists since $\delta^{n_1} \geq d+1$. Consider $g(\mathbf{x})$ as an n_1-variate polynomial with individual degree less than δ, and the support of g is a subset of M. Then, $g(\mathbf{x})$ can be written as:

$$g(\mathbf{x}) \;=\; \sum_{\mathbf{e} \in M} c_{\mathbf{e}} \mathbf{x}^{\mathbf{e}},$$

where $c_{\mathbf{e}}$'s are unknown to us. For each $\alpha \in \mathcal{H}$, we get a linear equation over $c_{\mathbf{e}}$'s by evaluating g at α, and we want them to be zero. Thus, we have a system of *homogeneous* linear equations, where the number of linear equations is $|\mathcal{H}| \leq d$, and the number of variables is $d+1$. Therefore, our system of linear equations has a nontrivial solution, which gives us a nonzero g.

Since the sparsity of g is $d+1$ and the degree of each monomial is upper bounded by δn_1, g has a depth-2 circuit of size less than $2n\delta d$. The time complexity of computing g has three components: 1) computing $\mathcal{H}$, 2) computing M, and 3) constructing the system of linear equations and solving it. Since each of these steps takes poly(t) time, the total time complexity is poly(t). $\square$

Lemma 2.3.5 (Valiant Class Separation)**.** *Let $\{f_n\}_{n\geq 1}$ be a multilinear polynomial family such that f_n is computable in time $t(n) = 2^{O(n)}$, but no $2^{o(n)}$-size circuit can compute it. Then, there exists a multilinear E-computable polynomial family $\{g_n\}_{n\geq 1}$ such that g_n has circuit complexity $2^{\Omega(n)}$. Furthermore, either E$\not\subseteq$ #P/poly or VNP has a polynomial family of algebraic circuit complexity $2^{\Omega(n)}$.*

Proof. First we show how to get an E-computable polynomial family $\{g_n\}_{n\geq 1}$ from $\{f_n\}_{n\geq 1}$. Only reason that $\{f_n\}_{n\geq 1}$ may not be an E-computable family is that its coefficients may not be integral. For that, we apply some transformation on f_n such that its coefficients become integral. When f_n is a polynomial over $\mathbb{Q}$ (or, isomorphic to $\mathbb{Q}$), we can write it in the following form

$$\sum_{\mathbf{e}\in \text{support}(f_n)} \frac{p_{\mathbf{e}}}{q_{\mathbf{e}}}\mathbf{x}^{\mathbf{e}},$$

where the size of support$(f_n) = 2^{O(n)}$, and the bitsize of both $p_{\mathbf{e}}$ and $q_{\mathbf{e}}$ is also $2^{O(n)}$. Let $N = \prod_{\mathbf{e}\in\text{support}(f_n)} q_{\mathbf{e}}$. Then consider $g_n := Nf_n$. It is easy to verify that g_n has integral coefficients, and it is computable in $2^{O(n)}$-time, but no $2^{o(n)}$-size circuit can compute it. Hence, $\{g_n\}n \geq 1$ is a multilinear E-computable polynomial family $\{g_n\}_{n\geq 1}$ such that g_n has circuit complexity $2^{\Omega(n)}$.

Now we describe the case when the coefficients of f_n are coming from a finite field of characteristic q. In that case, our coefficients of f_n are elements from an extension $\mathbb{K}$ of $\mathbb{F}_q$, and the degree of the extension ℓ can be at most $t(n)$. Otherwise representation of an element in $\mathbb{K}$ will take more than $t(n)$-time, therefore, f_n cannot be computable in time $t(n)$. Then f_n can be written as follows

$$f_n(\mathbf{x}) = \sum_{\mathbf{e}\in\text{support}(f_n)} c_{\mathbf{e}}\mathbf{x}^{\mathbf{e}},$$

where $c_{\mathbf{e}} \in \mathbb{K}$. Since $\mathbb{K}$ is a simple extension of $\mathbb{F}_q$, there exists an $\alpha \in \mathbb{K}$ such that each $c_{\mathbf{e}}$ can be written as

$$c_{\mathbf{e}} = \sum_{i=0}^{\ell} c_{\mathbf{e},i}\alpha^i, \text{ where } c_{\mathbf{e},i} \in \mathbb{F}_q.$$

Now each integer $0 \leq i \leq \ell$, i can be expressed as $\sum_{j=1}^{\lceil\log(\ell+1)\rceil} b_j 2^{j-1}$ where $b_j \in \{0,1\}$. Therefore α^i can be seen as evaluation of $\prod_{j=1}^{\lceil\log(\ell+1)\rceil} y_j^{b_j}$ at $y_j = \alpha^{2^{j-1}}$ for all $j \in [\lceil\log(\ell+1)\rceil]$. For $0 \leq i \leq \ell$, let $\mathbf{b}_i$ denote the bit representation of i. Then f_n is the evaluation of the polynomial

$$g_n(\mathbf{x},\mathbf{y}) := \sum_{\mathbf{e}\in\text{support}(f_n)} \sum_{i=0}^{\ell} c_{\mathbf{e},i}\mathbf{x}^{\mathbf{e}} \cdot \mathbf{y}^{\mathbf{b}_i},$$

at $y_j = \alpha^{2^{j-1}}$ for all $j \in [\lceil\log(\ell+1)\rceil]$. This kind of variable stretching transformation was used in [KP09, Lemma 3.9]. Now g_n is a $O(n)$-variate multilinear E-computable polynomial with circuit complexity $2^{\Omega(n)}$. This completes the proof of our first part.

Since g_n is E-computable, the coefficients of g_n have bitsize $2^{O(n)}$. Thus, one can index a bit of a coefficient using $O(n)$ bits. Now, using the variable increasing transformation from the proof of [KP09, Lemma 3.9], we get a multilinear polynomial family $\{h_n\}_{n\geq 1}$ such that it is E-computable, algebraic circuit complexity $2^{\Omega(n)}$, and the coefficients are from $\{0,1\}$.

Assume E⊆#P/poly. Since each coefficient of h_n is 0 or 1 and h_n is E-computable, the coefficient function of h_n is in #P/poly. [Val79] showed that if the coefficient function of a multilinear polynomial family over $\mathbb{N}$ is computable in #P/poly, then that polynomial family belongs to VNP. For proof, one can see [Bür13, Proposition 2.20]. Thus, $\{h_n\}_{n\geq 1}$ is in VNP and has algebraic circuit complexity $2^{\Omega(n)}$.

□

Now we describe how to design a hitting set from hard polynomial family. Towards that, a crucial ingredient is Nisan-Wigderson design [NW94]. It is a combinatorial design. Formally, we define it as follows.

Definition 2.3.6 (Nisan-Wigderson design)**.** *Let $\ell > n > d$. A family of subsets $\mathcal{D} = \{I_1, \ldots, I_m\}$ of $[\ell]$ is called an (ℓ, n, d)-*design*, if $|I_i| = n$ and for all $i \neq j \in [m]$, $|I_i \cap I_j| \leq d$.*

First, [NW94] gave construction of such design. Here, we mention a construction given in [AB09, Chapter 20].

Lemma 2.3.7. *There exists an algorithm which takes (ℓ, n, d) and a base set S of size $\ell > 10n^2/d$ as input, and outputs an (ℓ, n, d)-design $\mathcal{D}$ having $\geq 2^{d/10}$ subsets, in time $2^{O(\ell)}$.*

Later, in Lemma 4.2.1 of Chapter 4, we describe a slightly more optimized version of this lemma. However, the proof will be same as the proof given for the above construction in [AB09, Chapter 20], except technical differences. In the next lemma, we describe how to design a quasi-polynomial time blackbox PIT algorithm from an exponentially hard but E-computable multilinear polynomial family. It was shown in [KI04, Theorem 7.7].

Lemma 2.3.8 (Hardness to hitting set)**.** *Let $\{q_m\}_{m\geq 1}$ be a multilinear polynomial family such that q_m is computable in $2^{O(m)}$-time but has no $2^{o(m)}$-size algebraic circuit for it. Then, we have an $s^{O(\log s)}$-time computable hitting set for polynomials computed by size s degree s circuits.*

Proof. According to hypothesis, there is a constant $c_0 > 0$ such that q_m requires more than $2^{c_0 m}$-size algebraic circuits. Let $\mathcal{P}$ be the set of polynomials computed by size s degree s circuits. Let $\mathcal{D} = \{S_1, \ldots, S_s\}$ be a $(c_2 \log s, c_1 \log s, 10 \log s)$-design on the variable set $\mathbf{z} = \{z_1, \ldots, z_{c_2 \log s}\}$ (Lemma 2.3.7). Constants $c_2 > c_1 > 10$ will be fixed later. Our hitting-set generator for $\mathcal{P}$ is defined as follows. For all $i \in [n]$, let $p_i := q_{c_1 \log s}(S_i)$, i.e., the polynomial $q_{c_1 \log s}$ with S_i as the set of variables. Let $P(x_1, \ldots, x_n)$ be a nonzero polynomial in $\mathcal{P}$. Then $n \le s$. Now we show that $P(p_1, \ldots, p_n)$ is also nonzero.

For the sake of contradiction, assume that $P(p_1, \ldots, p_n)$ is zero. Since $P(\mathbf{x})$ is nonzero, we can find the smallest $j \in [n]$ such that $P(p_1, \ldots, p_{j-1}, x_j, \ldots, x_n) =: P_1$ is nonzero, but $P_1\big|_{x_j = p_j}$ is zero. Thus, $(x_j - p_j)$ divides P_1. Let $\mathbf{a}$ be an assignment on all the variables in P_1, except x_j and the variables S_j in p_j, with the property: P_1 at $\mathbf{a}$ is nonzero. Since P_1 is nonzero, there exists such an assignment. Now our new polynomial P_2 on the variables x_j and S_j is of the form:

$$P_2(S_j, x_j) \;=\; P(p'_1, \ldots, p'_{j-1}, x_j, a_{j+1}, \ldots, a_n),$$

where for each $i \in [j-1]$, p'_i is the polynomial on the variables $S_i \cap S_j$, and a_i's are field constants decided by the assignment $\mathbf{a}$. By the design, for each $i \in [j-1]$, $|S_i \cap S_j| \le 10 \log s$. Since p'_i are multilinear polynomials on variables $S_i \cap S_j$, each p'_i has a circuit (of form $\Sigma\Pi$) of size at most $10 \log s \cdot 2^{10 \log s} \le 2^{11 \log s}$. Then we have a circuit for P_2 of size at most $s_1 := s + n \cdot 2^{11 \log s}$, and degree at most $d_1 := s \cdot 10 \log s$. Since $(x_j - p_j)$ divides P_2, we can invoke the Kaltofen's factorization algorithm (see Lemma 2.3.3) and get an algebraic circuit for p_j of size $(s_1 d_1)^{c_3}$, for some constant c_3 (independent of c_1, c_2).

Now we fix constants c_1, c_2. Pick c_1 such that $2^{c_0 \cdot c_1 \log s}$ becomes greater than $(s_1 d_1)^{c_3}$, and $c_1 := 15 c_3 / c_0$ satisfies that condition. Pick c_2, according to Lemma 2.3.7, such that $c_2 \log s > 10 \cdot (c_1 \log s)^2 / (10 \log s)$. Hence, $c_2 := 1 + c_1^2$ works.

Moreover, if $P(p_1, \ldots, p_n)$ is zero then, by the above discussion, $p_j = q_{c_1 \log s}(S_j)$ has a circuit of size $(s_1 d_1)^{c_3} < 2^{c_0 \cdot c_1 \log s}$. This violates the lower bound hypothesis for p_j. Thus, $P(p_1, \ldots, p_n)$ is nonzero.

The time for computing $(p_1, \ldots, p_n)$ depends on: **(1)** computing the design, which takes $2^{O(c_2 \log s)} = \text{poly}(s)$ time, and **(2)** computing $q_{c_1 \log s}$ which takes $2^{O(\log s)} = \text{poly}(s)$ time. Thus, in poly(s) time we can convert a size s degree s circuit to a $O(\log s)$-variate poly(s) size poly(s) degree circuit while preserving nonzeroness. Now using Corollary 2.3.2, we get an $s^{O(\log s)}$-time computable hitting set for polynomials computed by size s degree s circuits. □

Remark. When the underlying field $\mathbb{F}$ is a finite field of characteristic p and size p^r, then by invoking Kaltofen's factoring algorithm, we get a small size circuit for $q_{c_1 \log s}^{p^t}$ for some nonnegative integer t divisible by r. Over a finite field, it is not known whether a small size circuit for $q_{c_1 \log s}^{p^t}$ implies a small size circuit for $q_{c_1 \log s}$ itself. However, the evaluations of $q_{c_1 \log s}^{p^t}$ coincides with the evaluations of $q_{c_1 \log s}$ over $\mathbb{F}^{c_1 \log s}$. Hence, [KI04] defined the circuit complexity of an n-variate polynomial q over a finite field $\mathbb{F}$ as the size of a smallest circuit that agrees with q over $\mathbb{F}^n$.

2.3.4 Lifting hardness from depth-4 circuits to general circuits

Now, we discuss a result from [AV08], where they first showed that in some cases we can reduce a circuit to a depth-4 circuit of nontrivial size (i.e., better than depth-2 representation). Our next lemma says that if an n-variate multilinear polynomial has a $2^{o(n)}$ size circuit, then it has also a $2^{o(n)}$ size depth-4 circuit. The contrapositive of the previous statement says that if the depth-4 circuit complexity of an n-variate multilinear polynomial is $2^{\Omega(n)}$, then any circuit computing that polynomial must have size $2^{\Omega(n)}$.

Lemma 2.3.9 (Corollary 2.5 [AV08])**.** *Let $\{q_m\}_{m\in\mathbb{N}}$ be a multilinear polynomial family. If there exists a $2^{o(m)}$-size circuit computing q_m, then there exists a $2^{o(m)}$-size depth-4 circuit for q_m. Furthermore, the fan-in of the top multiplication layer $a(m)$ is any sufficiently slow growing function in $\omega(1)$, and the fan-in of the bottom multiplication layer $b(m) = o(m)$.*

Proof. Suppose that the degree of q_m is $o(m)$. Then, there exists a $2^{o(m)}$-size depth-2 circuit ($\Sigma\Pi$) for q_m, and the fan-in of the multiplication layer is $o(m)$. This implies we have a $2^{o(m)}$-size depth-4 circuit for q_m with the fan-in of the top multiplication layer

$a(m) = O(1)$, and the fan-in of the bottom multiplication layer $b(m) = o(m)$. Next, we analysis the case when the degree of q_m is $\Omega(m)$.

According to hypothesis, q_m has a $2^{o(m)}$-size circuit. From 'log-depth reduction result' [Sap16, Section 5.3.2] [1] we get a circuit C, of $d_m = \Theta(\log m)$ depth and $s_m = 2^{o(m)}$ size, with the additional properties:

1. alternative layers of addition/multiplication gates with the top-gate (root) being addition,
2. from top as we go below, at each multiplication layer the degree of the related polynomials at least halves compared to the previous multiplication layer, and
3. fan-in of each multiplication gate is at most 5.

Now we cut the circuit C at the t-th layer of multiplication gates from the top, where $t = t(d_m)$ will be fixed later, to get the following two parts:

Top part: The top part computes a polynomial of degree at most 5^t, and the number of variables is at most s_m. Therefore, it can be reduced to a trivial $\Sigma\Pi$ circuit of size $\binom{s_m+5^t}{5^t} = s_m^{O(5^t)}$ (see Lemma 2.3.19).

Bottom part: In the bottom part, we can have at most s_m many top-multiplication gates that feed into the top part as input. Each multiplication gate computes a polynomial of degree at most $m \cdot 2^{-t}$, and the number of variables is at most m. Therefore, each multiplication gate can be reduced to a trivial $\Sigma\Pi$ circuit of size $\binom{m+m/2^t}{m/2^t} = 2^{O(mt/2^t)}$ (see Lemma 2.3.19).

From the above discussion, we have a $\Sigma\Pi^{5^t}\Sigma\Pi^{m/2^t}$ circuit C', computing q_m, that has size $s_m^{O(5^t)} + 5^t s_m \cdot 2^{O(mt/2^t)}$. The second summand is multiplied by an additional factor of 5^t to ensure that the fan-out of each node in the final circuit is at most 1.

Now we fix t. The second summand becomes $2^{o(m)}$ if we pick a $t = \omega(1)$ such that $5^t = o(m)$ (recall that $s_m = 2^{o(m)}$). To get a similar upper bound on the first summand we

[1] [VSBR83] first showed the log-depth reduction result. However, we are using the proof given in [Sap16, Section 5.3.2].

need to pick $5^t \log s_m = o(m)$. Finally, we also want $5^t \leq a(m)$, to satisfy the fan-in bound of the top multiplication layer. A function $t = t(d_m)$, satisfying the three conditions, exists as $\log s_m = o(m)$ and $a(\cdot)$ is an increasing function. Let us fix such a function t. (As C has super-constant depth, we can also assume that the cut at depth t is possible.) Thus the circuit C', computing q_m, has size $s'_m = 2^{o(m)}$.

Finally, we have a $2^{o(m)}$-size depth-4 circuit, where the fan-in of the top layer multiplication gates is $5^t \leq a(m)$, and the fan-in of the bottom layer multiplication layer is $m/2^t = o(m)$.

□

2.3.5 Reduction from depth-4 circuits to depth-3 circuits

In the next lemma, we describe a reduction from $\Sigma\Pi^{[a]}\Sigma\Pi^{[b]}$ circuits to $\Sigma\Pi\Sigma$ circuits. This result is from [GKKS16] which improves the depth-4 reduction result of [AV08, Koi12, Tav13] and showed a "nontrivial" depth-3 reduction for algebraic circuits. The statement we mention here does not appear in that form in [GKKS16]. They proved it in different parts (see [GKKS16, Section 4]). We combine them and present as a single lemma.

Lemma 2.3.10. *Let f be an n-variate degree d polynomial computed by a size s $\Sigma\Pi^{[a]}\Sigma\Pi^{[b]}$ circuit over $\mathbb{Q}$. Then there exists a size poly($s2^{a+b}$) $\Sigma\Pi\Sigma$ circuit computing f.*

Remark. The above lemma works over any field of characteristic *zero.*

Proof. The proof has three following steps:

- Convert a size s $\Sigma\Pi^{[a]}\Sigma\Pi^{[b]}$ to a $\Sigma\wedge^{[a]}\Sigma\wedge^{[b]}\Sigma$ circuit of size s_1 =poly($s2^{a+b}$).

- Convert a size s_1 $\Sigma\wedge^{[a]}\Sigma\wedge^{[b]}\Sigma$ circuit to a $\Sigma\Pi\Sigma$ circuit of size s_2 =poly(s_1) over $\mathbb{C}$.

- Convert a size s_2 $\Sigma\Pi\Sigma$ over $\mathbb{C}$ to a $\Sigma\Pi\Sigma$ circuit of size poly(s_2) over $\mathbb{Q}$.

Step 1: A size s $\Sigma\Pi^{[a]}\Sigma\Pi^{[b]}$ circuit C computes polynomial of the following form:

$$C = \sum_{i\in[s]} \prod_{j\in[a]} Q_{ij},$$

where $\deg(Q_{ij}) \leq b$ and the sparsity of $Q_{ij} \leq s$. Applying Lemma 2.3.13, we get an $s2^b b^2$-size $\Sigma^{[s2^b]} \wedge^{[b]} \Sigma^{[b]}$ circuit for each Q_{ij}. Again using Lemma 2.3.13, each $T_i = \prod_{j\in[a]} Q_{ij}$ can be written as a sum of power of 2^a ℓ_k's, where each ℓ_k is a linear combination of Q_{ij}'s. Hence, for each T_i, we have a $\Sigma^{[2^a]} \wedge^{[a]} \Sigma^{[sa2^b]} \wedge^{[b]} \Sigma^{[b]}$ circuit of size $s2^{a+b}a^2b^2$. Therefore, overall we get a $\Sigma \wedge^{[a]} \Sigma \wedge^{[b]} \Sigma$ circuit of size $s^2a^2b^2 \cdot 2^{a+b}$ =poly($s2^{a+b}$). Furthermore, by multiplying necessary number of 1's, we can get a $\Sigma \wedge^{[a]} \Sigma \wedge^{[b]} \Sigma$ circuit where the fan-in of the bottom layer powering gates is exactly b. In next step, we work with $\Sigma \wedge^{[a]} \Sigma \wedge^{[=b]} \Sigma$ circuits.

Step 2: Any size s circuit C of form $\Sigma \wedge^{[a]} \Sigma \wedge^{[=b]} \Sigma$ computes polynomial of form $C = T_1 + \cdots + T_{s_1}$, where each T_i is of form $(\ell_{i1}^b + \cdots + \ell_{is_1}^b)^a$ and ℓ_{ij}'s are linear polynomials. Using Lemma 2.3.12, each T of form $(\ell_1^b + \cdots + \ell_{s_1}^b)^a$ can be written as

$$\begin{aligned} T &= \left(\ell_1^b + \cdots + \ell_{s_1}^b\right)^a \\ &= \sum_{i=0}^{s_1a}\sum_{j=0}^{a} \beta_{ij}(\alpha_i + \ell_1^b)^j \cdots (\alpha_i + \ell_{s_1}^b)^j \\ &= \sum_{i=0}^{s_1a}\sum_{j=0}^{a} \beta_{ij} f_i(\ell_1)^j \cdots f_i(\ell_{s_1})^j, \end{aligned}$$

where $f_i(t) = (\alpha_i + t^b)$. Since $f_i(t)$ is a univariate polynomial over $\mathbb{C}$, it completely splits into linear factors. Therefore, each T can be written as follows

$$T = \sum_{i=0}^{s_1a}\sum_{j=0}^{a} \beta_{ij} \prod_{k=1}^{s_1}\prod_{r=1}^{b} (\ell_k - \gamma_{ir})^j .$$

Thus, each T need a depth-3 circuit of top fan-in at most $(s_1a+1)(a+1) = O(s_1a^2)$ and the degree is at most s_1ab. Hence, to compute C we need a depth-3 circuit of size at most

$$O(s_1^2a^3b(n+1)) = \text{poly}(s_1) .$$

Step 3: After Step 2, we get a depth-3 circuit which computes a polynomial over $\mathbb{Q}$, but the field constants in the circuit are from $\mathbb{C}$. However, we want a poly(s_1)-size circuit over $\mathbb{Q}$. In the previous step, the reason we need $\mathbb{C}$ is that we want to split $f_i(t) = (\alpha_i + t^b)$

into linear factors. The polynomial

$$f_i(t) = (\alpha_i + t^b) = \prod_{i=1}^{b}((-\alpha_i)^{\frac{1}{b}} \cdot \omega_b^i - t),$$

where ω_b is a bth primitive root of unity. Since we have complete freedom to pick α_i's, we can pick them in a way such that $(-\alpha_i)^{\frac{1}{b}}$ is also in $\mathbb{Q}$. Then, to completely split $f_i(t)$, we only need to go to the field $\mathbb{Q}(\omega_b)$, an extension of $\mathbb{Q}$, not $\mathbb{C}$. The field $\mathbb{Q}(\omega_b)$ is a simple extension over $\mathbb{Q}$ of degree at most b. There is a generic way of converting a circuit over a small extension (of the base field) to a circuit over the base field. We skip the details here and refer to [GKKS16, Section 4.3.1]. Thus, we have a $\Sigma\Pi\Sigma$ circuit of size $\mathrm{poly}(s2^{a+b})$ computing f. □

2.3.6 Reducing the degree of a circuit

In general, the degree of a circuit can be much larger than the degree of the polynomial it computes. However, [Str73] showed that in those scenarios the polynomial is also computable by a slightly larger size circuit whose degree is same as the degree of the polynomial.

Lemma 2.3.11. *Let f be a degree d polynomial computed by a size s circuit. Then f is also computed by a size $O(sd^2)$ degree d circuit.*

Proof. Let C be a size s circuit which computes f. Without loss of generality, we can assume the fan-in of each node is 2. Now we construct a new circuit C' of degree d and it also computes f. For every gate g in C, we introduce $d+1$ many copies, $\{g_0, \ldots, g_d\}$, of it with the following motivation: g_i will compute the homogeneous degree i part of the polynomial computed at g. Since f is a degree d polynomial, we do not need to consider the higher degree parts of g. Now we inductively build the circuit C'. Let g has two children u and v.

When $g = u + v$: For all $i \in \{0, \ldots, d\}$, the gate g_i is defined as $u_i + v_i$.

When $g = u \times v$: For all $i \in \{0, \ldots, d\}$, the gate g_i is defined as $\sum_{j=0}^{d} u_j v_{d-j}$.

Hence, for each g_i we need to add at most $2(d+1)$ many edges. By this process, we get a $(d+1)$-output circuit of size $O(sd^2)$ such that the ith output computes the homogeneous degree i part of f. Now we add a addition gate on the top which adds all the homogeneous components of f. Thus we have a $O(sd^2)$ size degree d circuit computing f. $\square$

2.3.7 Miscellaneous results

Now, we describe the *duality trick* by [Sax08]. The statement below is slightly different from the statement in [Sax08], and the proof below is by [FGS13]. The original statement has a dependence on the characteristic of the underlying $\mathbb{F}$. However, the proof by [FGS13] removes that dependence, and it works over any sufficiently large fields.

Lemma 2.3.12 (Duality trick)**.** *Let m, d be two positive integers. Let $\mathbb{F}$ be a field of size greater than $d(m-1)$. Then for every distinct $\alpha_0, \alpha_1, \ldots \alpha_{(m-1)d} \in \mathbb{F}$, there exist β_{ij}'s such that*

$$(z_1 + \ldots + z_m)^d = \sum_{i=0}^{(m-1)d} \sum_{j=0}^{d} \beta_{ij}(z_1+\alpha_i)^j \cdots (z_m+\alpha_i)^j \,.$$

Proof. Consider the polynomial $p(t) = (z_1+t)\cdots(z_m+t) - t^m$. The coefficient of t^{m-1} in $p(t)$ is $(z_1 + \cdots + z_m)$. Let $g(t) := p(t)^d$. Then the coefficient of $t^{(m-1)d}$ in $g(t)$ is $(z_1+\cdots+z_m)^d$. Now consider $\{g(\alpha_0), \ldots, g(\alpha_{(m-1)d})\}$, the set of evaluations of $g(t)$ at α_i's. It is well known that for any t^i, the coefficient of t^i in $g(t)$ can be extracted by taking linear combinations of $g(\alpha_i)$'s. Therefore, there exist β'_i's

$$\begin{aligned}(z_1+\cdots+z_m)^d &= \sum_{i=0}^{(m-1)d} \beta'_i g(\alpha_i)\\ &= \sum_{i=0}^{(m-1)d} \beta'_i\left((z_1+\alpha_i)\cdots(z_m+\alpha_i) - \alpha_i^m\right)^d\\ &= \sum_{i=0}^{(m-1)d} \sum_{j=0}^{d} \beta_{ij}(z_1+\alpha_i)^j \cdots (z_m+\alpha_i)^j\end{aligned}$$

where $\beta_{ij} = \binom{d}{j}\beta'_i(-\alpha_i^m)^{d-j}$. $\square$

Next lemma describes a way to write a monomial into a sum of powers. The statement given below can be derived from Fischer's trick [Fis94] or Ryser's formula [Rys63]. It requires $\mathrm{char}(\mathbb{F}) = 0$ or large.

Lemma 2.3.13. *Over a field $\mathbb{F}$ of $\mathrm{char}(\mathbb{F}) = 0$ or greater than n, any expression of the form $g = \prod_{i\in[n]} g_i$ can be rewritten as $g = \sum_{j\in[2^n]} c_j f_j^n$ where each f_j is a linear combination of g_i's.*

Proof. First we prove that

$$n! \cdot x_1 \cdots x_n = \sum_{S\subseteq[n]} (-1)^{n-|S|} \left(\sum_{i\in S} x_i\right)^n .$$

The only summand which contributes the monomial $x_1 \cdots x_n$ is when $S = [n]$, and in $(x_1 + \cdots + x_n)^n$, the coefficient of $x_1 \cdots x_n$ is $n!$. Next we show that the coefficients of all other monomials are zero. The proof is based on inclusion-exclusion principle. Let $m = x_1^{e_1} \cdots x_n^{e_n}$ be a degree n monomial other than $x_1 \cdots x_n$. Let M be the subset of $[n]$ such that for all $i \in [n]$, the variable x_i is present in m if and only if $i \in M$. Since $m \neq x_1 \cdots x_n$, M is a proper subset of $[n]$. In the above expression, the only summands which contribute m are when $S \supseteq M$, and the coefficient of m in that summand is $(-1)^{n-|S|} \cdot \binom{n}{e_1,\ldots e_n}$. Therefore, the coefficient of m is

$$\sum_{M\subseteq S\subseteq[n]} (-1)^{n-|S|} \cdot \binom{n}{e_1, \ldots, e_n} = 0,$$

since M is not equal to $[n]$. Now put $x_i = g_i$ in the above relation and get our desired expression for g. $\square$

In the following lemma we describe a standard technique to design a hitting set for the set of polynomials having a "low-support" monomial in their support. It has been used in almost all PIT results regarding the construction of hitting sets for various restricted classes of circuits.

Lemma 2.3.14. *Let $\mathcal{P}$ be the set of n-variate degree d polynomials such that every nonzero polynomial in $\mathcal{P}$ has a ℓ-support monomial with nonzero coefficient. Then there exists a hitting set for $\mathcal{P}$ computable in time $(nd)^{O(\ell)}$.*

Proof. Let $\mathbf{x} = \{x_1, \ldots, x_n\}$ be the set of variables over which $\mathcal{P}$ is defined. For every ℓ-size subset S of $[n]$, let $\varphi_S : \mathbf{x} \to \mathbf{x}$ be the mapping defined as, for all $i \in [n]$,

$$\varphi_S(x_i) := \begin{cases} x_i & \text{if } i \in S \\ 0 & \text{otherwise.} \end{cases}$$

Let P be a nonzero polynomial in $\mathcal{P}$. Since P has a ℓ-support monomial with nonzero coefficient, there exists a ℓ-size subset S of $[n]$ such that $P(\varphi_S(\mathbf{x}))$ is also nonzero. For every ℓ-size subset S of $[n]$, $P(\varphi_S(\mathbf{x}))$ becomes a ℓ-variate degree d polynomial. From Corollary 2.3.2, $P(\varphi_S(\mathbf{x}))$ has a hitting set computable in time $d^{O(\ell)}$. Since for given a P we do not know the ℓ-size subset S of $[n]$ for which $P(\varphi_S(\mathbf{x}))$ is nonzero, we try all possible φ_S's. This combined with Corollary 2.3.2 gives a $(nd)^{O(\ell)}$-time computable hitting-set for $\mathcal{P}$. □

Next lemma describes a relation between the dimension of partial derivative space and cone size of monomials. It was shown in [For14, Corollary 4.14] (with origins in[FS13a]).

Lemma 2.3.15. *Let $\mathbb{F}$ be a field of characteristic 0 or greater than d. Let $\mathcal{P}$ be a set of n-variate degree d polynomials over $\mathbb{F}$ such that for all $P \in \mathcal{P}$, the dimension of the partial derivative space of P is at most k. Then for every nonzero $P \in \mathcal{P}$ has a k-cone monomial with nonzero coefficient.*

Proof. Let $\mathbf{x}$ be the set of variables over which $\mathcal{P}$ is defined. Let $\prec$ be a monomial ordering on the set of monomials over $\mathbf{x}$. For example, one can assume $\prec$ is the lexicographic ordering on the monomials. Then for any polynomial f, by $\mathrm{LM}(f)$ we denote the leading monomial of f (with respect to $\prec$). For notations, see Section 2.1. Our proof has the following steps.

1. First, we show that the dimension of the partial derivative space of P is lower bounded by the number of distinct leading monomials (with respect to the monomial ordering $\prec$) we can get from all the polynomials in the partial derivative space of P, i.e.

$$\dim \langle \partial^{<\infty}(P) \rangle \geq \left| \left\{ \mathrm{LM}(f) \mid f \in \langle \partial^{<\infty}(P) \rangle \right\} \right|.$$

2. Next, we prove that the number of distinct leading monomials one can get from all the polynomials in the partial derivative space of P is lower bounded by the cone size of the leading monomial of P, i.e.

$$\left|\{\mathrm{LM}(f) \ \mid \ f \in \langle \partial^{<\infty}(P) \rangle\}\right| \geq |\mathrm{cone}(\mathrm{LM}(P))| \,.$$

Step 1: Since the degree of every polynomial in $\langle \partial^{<\infty}(P) \rangle$ is upper bounded by d, $|\{\mathrm{LM}(f) \ \mid \ f \in \langle \partial^{<\infty}(P) \rangle\}|$ is finite. Let it be k. Suppose that $\{f_1, \ldots, f_k\}$ be the set of polynomials in $\langle \partial^{<\infty}(P) \rangle$ such that

$$\{\mathrm{LM}(f) \mid f \in \langle \partial^{<\infty}(P) \rangle\} = \{\mathrm{LM}(f_i) \ \mid \ i \in [k]\}.$$

Let $m_i = \mathrm{LM}(f_i)$. Without loss of generality, we can assume that $m_1 \prec m_2 \prec \cdots \prec m_k$. Now we show that f_i's are linearly independent over $\mathbb{F}$. Let $g = \sum_{i=1}^{k} c_i f_i$ be a nonzero linear combination of f_i's, i.e., there exists an i such that c_i is nonzero. Then we show that g is also a nonzero polynomial. Let j be the largest i such that c_j is nonzero. Then the leading monomial $\mathrm{LM}(f_j)$ do not cancel in the expression $\sum_{i=1}^{k} c_i f_i$. Hence, g is a nonzero polynomial. This completes our first step.

Step 2: Let $\mathbf{x}^{\mathbf{e}}$ be the leading monomial of P with respect to $\prec$. Then P can be written as

$$P = c_{\mathbf{e}} \mathbf{x}^{\mathbf{e}} + \sum_{\mathbf{h} \in \mathrm{support}(P) \setminus \{\mathbf{e}\}} c_{\mathbf{h}} \mathbf{x}^{\mathbf{h}},$$

where $c_{\mathbf{e}}$ and $c_{\mathbf{h}}$'s are the coefficients of the respective monomials in P. Now we show that for every monomial m in the cone of $\mathbf{x}^{\mathbf{e}}$, there exists a polynomial P' in the partial derivative space of P such that $\mathrm{LM}(P') = m$. Let $\mathbf{x}^{\mathbf{f}}$ be a monomial in the cone of $\mathbf{x}^{\mathbf{e}}$ and $\mathbf{g} := \mathbf{e} - \mathbf{f}$. Let $\mathbf{e} = (e_1, \ldots, e_n)$ and $\mathbf{g} = (g_1, \ldots, g_n)$. Then the partial derivaive of P with respect to $\mathbf{x}^{\mathbf{g}}$ is

$$\partial_{\mathbf{x}^{\mathbf{g}}}(P) = c'_{\mathbf{x}^{\mathbf{e}}} \mathbf{x}^{\mathbf{e}-\mathbf{g}} + \sum_{\mathbf{h} \in \mathrm{support}(P) \setminus \{\mathbf{e}\}} c'_{\mathbf{h}} \mathbf{x}^{\mathbf{h}-\mathbf{g}},$$

where

$$c'_{\mathbf{e}} = c_{\mathbf{e}} \cdot \prod_{i=1}^{n} \frac{e_i!}{(e_i - g_i)!}.$$

Similarly, $c'_{\mathbf{h}}$'s are also defined. From the definition of the monomial ordering, $\mathbf{x}^{\mathbf{h}-\mathbf{g}} \prec \mathbf{x}^{\mathbf{e}-\mathbf{g}}$ for all $\mathbf{h} \in \text{support}(P) \setminus \{\mathbf{e}\}$. Since the characteristic of $\mathbb{F}$ is 0 or greater than d, $c'_{\mathbf{e}}$ is a nonzero element in $\mathbb{F}$. Hence, $\text{LM}(\partial_{\mathbf{x}^{\mathbf{g}}}(P))$ is $\mathbf{x}^{\mathbf{f}}$. This implies that for every monomial $\mathbf{x}^{\mathbf{f}}$ in the cone of $\mathbf{x}^{\mathbf{e}}$, there exists a polynomial h in $\langle \partial^{<\infty}(P) \rangle$ such that $\text{LM}(h)$ is same as $\mathbf{x}^{\mathbf{f}}$. Thus, we proved that

$$|\{\text{LM}(f) \mid f \in \langle \partial^{<\infty}(P) \rangle\}| \geq |\text{cone}(\text{LM}(P))|.$$

This completes the proof of our lemma. □

Our next lemma implies that the dimension of the partial derivative space of a polynomial computed by a depth-3 diagonal circuit is polynomially large with respect to the circuit size.

Lemma 2.3.16. *Let P be an n-variate polynomial over $\mathbb{F}$ which can be written as $\sum_{i=1}^{k} c_i (a_{i0} + a_{i1}x_1 + \cdots + a_{in}x_n)^{d_i}$, where c_i's and a_{ij}'s are in $\mathbb{F}$. Then the dimension of the partial derivative space of P is less than or equal to $k(d+1)$, where $d = \max_i d_i$.*

Proof. Let $P_i = (a_{i0} + a_{i1}x_1 + \cdots + a_{in}x_n)^{d_i}$. Let $\mathbf{x}^{\mathbf{e}}$ be a degree b monomial and $\mathbf{e} = (e_1, \ldots, e_n)$. Then

$$\partial_{\mathbf{x}^{\mathbf{e}}}(P_i) = \binom{b}{e_1, \ldots, e_n} (a_{i0} + a_{i1}x_1 + \cdots + a_{in}x_n)^{d_i - b} \prod_{j=1}^{n} a_{ij}^{e_j}.$$

Then for all $b \leq d_i$,

$$\dim \langle \partial^{=b}(P_i) \rangle \leq 1,$$

and for all $b > d_i$, it is zero. Hence, the dimension of the partial derivative space of P_i is at most $(d_i + 1)$. Since the dimension of the partial derivative space follows sub-additivity property,

$$\begin{aligned} \dim \langle \partial^{<\infty}(P) \rangle &\leq \sum_{i=1}^{k} \dim \langle \partial^{<\infty}(P_i) \rangle \\ &\leq \sum_{i=1}^{k} (d_i + 1) \\ &\leq k(d+1). \end{aligned}$$

□

In the following lemma, we describe a formula for the determinant of the multiplication of two "fat" matrices. It is known as Cauchy-Binet formula [Zen93].

Lemma 2.3.17. *Let $n \geq m$ be two positive integers. Let A be an $m \times n$ and B be an $n \times m$ matrix over $\mathbb{F}$. Then*

$$\det(AB) = \sum_{S \in \binom{[n]}{m}} \det(A_{[m],S}) \cdot \det(B_{S,[m]}).$$

Proof. Let M be an $n \times n$ diagonal matrix with (i,i)th entry is x_i. Then $\det(AMB)$ is a polynomial f over the variables $\mathbf{x} = \{x_1, \ldots, x_n\}$ such that $f(1, \ldots, 1) = \det(AB)$. Also note that f is a homogeneous degree m polynomial, since each entry in AMB is degree 1 polynomial and AMB is an $m \times m$ matrix. Let S be a subset of $[n]$ of size m. Let $\rho_S : \mathbf{x} \to \mathbf{x}$ be a mapping defined as, for all $i \in [n]$,

$$\rho_S(x_i) := \begin{cases} x_i & \text{if } i \in S \\ 0 & \text{otherwise,} \end{cases}$$

Then $f(\rho_S(\mathbf{x}))$ becomes $\det(A_{[m],S}) \cdot \det(B_{S,[m]}) \prod_{i \in S} x_i$. This also implies that the coefficient of any non-multilinear monomial in f is zero. Hence, we can write

$$\begin{aligned} f(\mathbf{x}) &= \sum_{S \in \binom{[n]}{m}} f(\rho_S(\mathbf{x})) \\ &= \sum_{S \in \binom{[n]}{m}} \det(A_{[m],S}) \cdot \det(B_{S,[m]}) \prod_{i \in S} x_i \end{aligned}$$

Now, assign each x_i to 1 and get the required expression for $\det(AB)$. □

Now we describe a property of matrices whose entries are binomial coefficients. It was shown in [GKS17, Claim 3.3]. They used this property in designing PIT algorithms for ROABPs.

Lemma 2.3.18. *Let $a_1, \ldots, a_n$ be n distinct nonnegative integers and $\mathbb{F}$ be a field such that* $\mathrm{char}(\mathbb{F}) = 0$ *or greater than the maximum of all a_j's. Let A be an $n \times n$ matrix such that for all $i, j \in [n]$, $A_{i,j} := \binom{a_j}{i-1}$. Then, A has full rank over $\mathbb{F}$.*

Proof. We show that for every nonzero vector $\mathbf{b} = (b_1, \ldots, b_n) \in \mathbb{F}^{n\times 1}$, $\mathbf{b}A$ is also a nonzero vector. Consider the polynomial

$$g(x) = \sum_{i=1}^{n} b_i \frac{x(x-1)\cdots(x-i+2)}{(i-1)!}.$$

It is a nonzero polynomial of degree at most $n-1$. The jth entry of $\mathbf{b}A$ can be seen as the evaluation of g at a_j. Since g is a polynomial of degree at most $n-1$, it can have at most $n-1$ roots. This implies that $\mathbf{b}A$ has a nonzero coordinate. Hence, we prove our claim. $\square$

An alternative proof of the above lemma can be done using Lindstrm-Gessel-Viennot Lemma. For interested reader, we refer [GV16]. Next, we show an estimation of binomial coefficients.

Lemma 2.3.19. *For all $0 < k \leq n$ integers,*

$$\sum_{i=0}^{k} \binom{n}{i} \leq \left(\frac{\mathrm{e}n}{k}\right)^k .$$

Proof. For $0 < t \leq 1$,

$$\sum_{i=0}^{k} \binom{n}{i} \leq \frac{1}{t^k} \sum_{i=0}^{k} \binom{n}{i} t^i \leq \frac{(1+t)^n}{t^k} .$$

Since $(1+t) < \mathrm{e}^t$ for all $t \neq 0$, from the above expression we get $\sum_{i=0}^{k} \binom{n}{k} \leq \frac{\mathrm{e}^{tn}}{t^k}$. Now put $t = k/n$ and get the given inequality. $\square$

Chapter 3

Perfect Bootstrapping

Abstract

This chapter is based on joint work with Manindra Agrawal and Nitin Saxena [AGS18].

In this chapter, we show that if for some $c \in \mathbb{N}$ and all sufficiently large s, we have a poly(s)-size hitting set for $\log^{\circ c} s$-variate size s degree s circuits, then we have a poly(s)-size hitting set for s-variate size s degree s circuits. We refer to this phenomenon as Perfect Bootstrapping. We also study a case when the number of variables is $s^{o(1)}$ and the size of the hitting set is sub-exponential in the hypothesis, and bootstrap it to a sub-exponential size hitting set in the conclusion.

3.1 Motivation and our result

Pseudorandom generator (PRG) is a well studied object in boolean circuit complexity theory and cryptography [AB09, Chapters 9 & 20]. One of the main motivation of studying PRG is to efficiently derandomize all randomized algorithms. Indeed, one can show that if we have an *optimal PRG* against BPP, then BPP=P. By optimal PRG, we mean a PRG which stretches an n-length string to a $2^{\Omega(n)}$-length string and is computable in $2^{O(n)}$ time.

Interestingly, constructing an optimal PRG is closely related to strong circuit lower bounds. It is a celebrated result that designing optimal PRG against P/poly is equivalent to finding a boolean function in E with (boolean) circuit complexity $2^{\Omega(n)}$ [NW94, Sections 2 and 3], [IW97, Theorem 2].

Naturally, an algebraic analog of the latter property would be to identify an E-computable polynomial family which has *algebraic* circuit complexity $2^{\Omega(n)}$. By Valiant's criterion [Bür13, Proposition 2.20], if the coefficients of a polynomial family are computable in #P/poly, then that polynomial family is in VNP . Hence, if one replaces E by #P/poly in the first line, then we are directly talking about identifying a polynomial family in VNP which has *algebraic* circuit complexity $2^{\Omega(n)}$. Such a statement is a stronger version of VP≠VNP, since in VP≠VNP question we are only interested in finding a polynomial family in VNP which has circuit complexity $n^{\omega(1)}$. As a first challenge, we can pose the following reasonable complexity conjecture.

Conjecture 1. There is an E-computable polynomial family which has algebraic complexity $2^{\Omega(n)}$. Thus, either $E \not\subseteq$ #P/poly or VNP has a polynomial family of algebraic circuit complexity $2^{\Omega(n)}$.

Hitting-set generator (HSG) in the algebraic world can be viewed as an analogy to PRG in the boolean world. So one can naturally ask about the relation between HSG and algebraic circuit lower bound. [HS80, Theorem 4.5] introduced the concept of efficient annihilator of the HSG. They showed that if we can efficiently compute an HSG for a set of polynomials $\mathcal{P}$, then we can also efficiently compute a polynomial (namely, the annihilator) which does not belong to $\mathcal{P}$. This technique can be easily extended to get a circuit lower bound (Theorem 0). Like in the boolean world, our hard polynomial is also E-computable and has algebraic circuit complexity $2^{\Omega(n)}$.

Theorem 0 (Connection). If we have a poly(s)-time computable HSG for polynomials computed by size s degree s circuits, then Conjecture 1 holds.

Proof sketch. For all $s \in \mathbb{N}$, let $\mathcal{P}_s$ be the set of polynomials computed by size s degree s circuits. Using basic linear algebra, in $2^{O(m)}$-time, we can construct an m-variate multilinear annihilator q_m, where $m = O(\log s)$, of the HSG of $\mathcal{P}_s$. This q_m cannot lie in $\mathcal{P}_s$, otherwise q_m evaluated at the HSG would be a nonzero polynomial (contradicting the annihilation property). For details, see the proof of [Agr05, Theorem 51][1]. For the sake of contradiction, assume that it has a circuit of size $s^{o(1)}$. Since the degree of q_m is $O(\log s)$, we can use Lemma 2.3.11 and get a $s^{o(1)}$-size $O(\log s)$-degree circuit computing q_m. Therefore, we get that $q_m \in \mathcal{P}_s$, which is a contradiction. So we have a multilinear polynomial family $\{q_m\}_{m\geq 1}$ such that it is computable in $2^{O(m)}$-time but the algebraic circuit complexity is $s^{\Omega(1)} = 2^{\Omega(m)}$. Now applying Lemma 2.3.5, we get either $E \not\subseteq$ #P/poly or VNP has a polynomial family of algebraic circuit complexity $2^{\Omega(m)}$. $\square$

A weak converse of the above theorem, i.e. hardness to HSG, is well-known due to [KI04, Theorem 7.7]. We state it as Lemma 2.3.8. Suppose that we have an exponentially hard but $2^{O(m)}$-time computable multilinear polynomial family $\{q_m\}_{m\geq 1}$. Then by using Lemma 2.3.8, we get a quasi-polynomial time computable HSG for $\mathcal{P}_s$. This suggests that the 'hardness vs randomness' connection here is less satisfactory than the boolean world. Nonetheless, one wonders whether the conclusion in Theorem 0 can be strengthened in a different way, so that we get a perfect equivalence. In this work, we answer this question by introducing the concept of partial HSG.

Partial HSG. For all $s \in \mathbb{N}$, let $\mathcal{P}_s$ be the set polynomials over $(x_1, \ldots, x_s)$ variables and computed by size s degree s circuits. For any $m \leq s$, let $\mathcal{P}_{s,m}$ be the subset of those polynomials in $\mathcal{P}_s$ which depend only on the first m variables, i.e., $(x_1, \ldots, x_m)$. Suppose that, for some $m \ll s$, one efficiently compute an HSG $\mathbf{g}_{s,m}$ for $\mathcal{P}_{s,m}$. We call such an HSG is a partial HSG of $\mathcal{P}_s$. Then can we *bootstrap* this partial HSG, i.e., using $\mathbf{g}_{s,m}$, can we also efficiently design a complete HSG $\mathbf{g}_s$ for $\mathcal{P}_s$?

Suppose that $m = s^{1/c}$ for some $c \in \mathbb{N}$ and we can compute $\mathbf{g}_{s,m}$ in poly(s)-time. Then we can also design $\mathbf{g}_s$ in poly(s)-time and the reason is the following: The set $\mathcal{P}_s$ can be thought of as a subset of those polynomials in $\mathcal{P}_{s^c}$ which depend only on the first s variables. Therefore, $\mathbf{g}_{s^c,s}$ is an HSG for $\mathcal{P}_s$. Clearly, $\mathbf{g}_{s^c,s}$ can be computed in poly(s)-time. However, for $m = s^{o(1)}$, we cannot use the same argument for the following reason. To

[1]Lemma 2.3.4 describes the same phenomenon using the notion of hitting set.

compute the HSG of $\mathcal{P}_s$, we have to compute the partial HSG for $\mathcal{P}_{s^{\omega(1)}}$, which may not be computable in poly(s)-time. Naively speaking, there is no reason why a partial HSG $\mathbf{g}_{s,s^{o(1)}}$ could be bootstrapped efficiently to $\mathbf{g}_s$. The former is a property of the polynomial ring $\mathbb{F}[x_1, \ldots, x_{s^{o(1)}}]$ compared to the latter one which is a property of the "much larger" polynomial ring $\mathbb{F}[x_1, \ldots, x_s]$; so a considerable blow-up might be expected. Somewhat surprisingly, we give a positive answer when m is as small as $\log^{\circ c} s$ for some $c \in \mathbb{N}$.

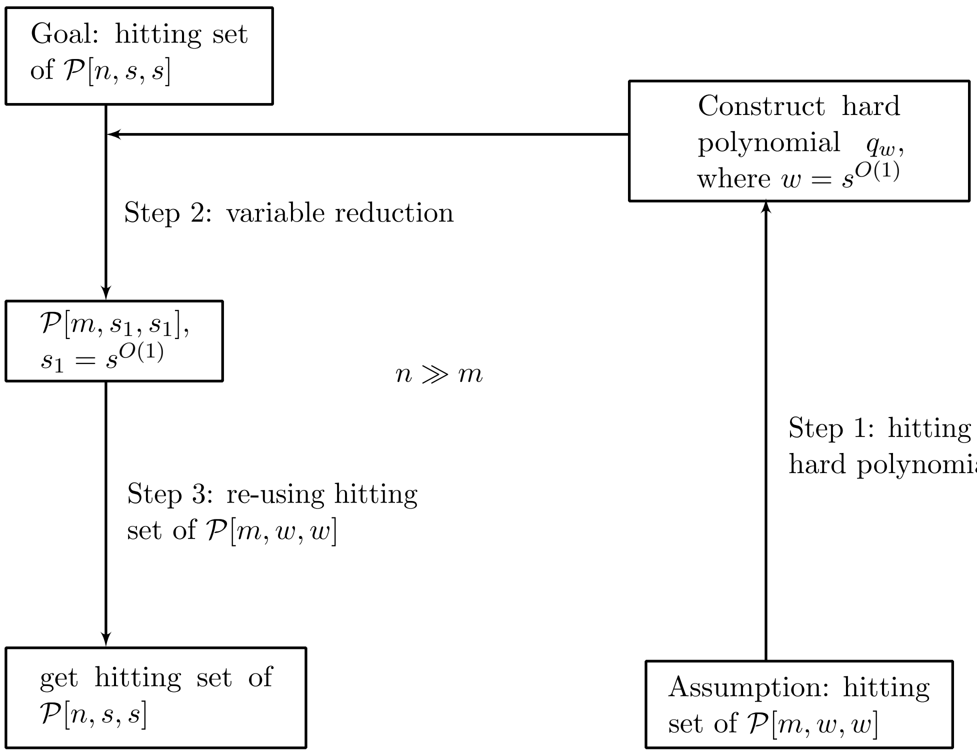
Theorem 3.1.1 (Perfect Bootstrapping)**.** *Let c be a positive integer. For all sufficiently large s, suppose that we have a poly(s)-time computable hitting set for $\lceil \log^{\circ c} s \rceil$-variate polynomials computed by size s degree s circuits. Then, we have a poly(s)-time computable hitting set for the class of s-variate polynomials computed by size s degree s circuits and Conjecture 1 holds.*

Remark. In the boolean world, we cannot reduce the number of variables beyond $\log s$. For more details, see Section 1.2.1.

We also study the case when our partial HSG can be computed in sub-exponential time, which is far worse than polynomial time. In this case, our result is not as strong as Theorem 3.1.1. However, in the hypothesis we still deal with a partial HSG $\mathbf{g}_{s,m}$ where $m = s^{o(1)}$ and manage to bootstrap that partial HSG in subexponential time. Also, an E-computable super-polynomially hard polynomial family is implied (say, *weak Conjecture 1*). For details see Theorem 3.3.1.

3.2 Proof of our result

Our proof of the Theorem 3.1.1 is iterative in nature. We start from a poly(s)-size hitting set for size s degree s circuits over $\lceil \log^{\circ c} s \rceil$ variables. At each step, we perform an exponential stretch of variables and get a polynomial size hitting set for circuits with more number of variables. After applying this step for constantly many rounds we get a poly(s)-size hitting set for s-variate size s degree s circuits. Each variable stretching step has a structure like Figure 3.1. One crucial component in that structure is Step 2, where we reduce the number of variables of polynomials in $\mathcal{P}[n, s, s]$ using a hard polynomial. As we mentioned earlier,



Goal: hitting set of $\mathcal{P}[n, s, s]$
Construct hard polynomial q_w, where $w = s^{O(1)}$
Step 2: variable reduction
$\mathcal{P}[m, s_1, s_1]$, $s_1 = s^{O(1)}$
$n \gg m$
Step 1: hitting set to hard polynomial
Step 3: re-using hitting set of $\mathcal{P}[m, w, w]$
get hitting set of $\mathcal{P}[n, s, s]$
Assumption: hitting set of $\mathcal{P}[m, w, w]$

Figure 3.1: Structure of a single step of Perfect Bootstrapping

such technique is already available due to [KI04, Theorem 7.7] (see Lemma 2.3.8). They showed an efficient variable reduction from an exponentially hard multilinear polynomial family. However, while bootstrapping in Theorem 3.1.1, we work in a scenario where the number of variables can be as low as $\log^{\circ c} s$ compared to s (i.e., size of the circuit). In this extremely low variate regime, the hard polynomials we get from Step 1 have *non*-constant individual degree. This imposes a bit more technical challenges in proving Step 2 than the proof of Lemma 2.3.8. However, by carefully fixing the associated parameters, one can extend the proof of Lemma 2.3.8 and prove Step 2. In the proof of our following lemma, we give a detailed proof of Step 2. Now we formally describe what happens in each variable stretching step.

Lemma 3.2.1 (Induction step)**.** *For any nonnegative integer i, let g_i be a function defined as $g_i(s) := (\log^{\circ i} s)^2$. For all sufficiently large s, suppose that we have a poly(s)-time computable hitting set for $g_i(s)$-variate degree s polynomials computed by size s circuits. Then we have a poly(s)-time computable hitting set for $g_{i-1}(s)$-variate degree s polynomials* *computed by size s circuits.*

Proof. In the proof, we use $\mathcal{P}[g_{i-1}, s, s]$ to denote the set of polynomials $\mathcal{P}[g_{i-1}(s), s, s]$. It is a slight abuse of notation. However, it will simplify our representation. Similarly, we use for $\mathcal{P}[g_i, s, s]$ for $\mathcal{P}[g_i(s), s, s]$. For a given s, we want to construct a poly(s)-time computable hitting set for $\mathcal{P}[g_{i-1}, s, s]$. According to hypothesis, for some constant e, we have a w^e-size w^e-time computable hitting set for $\mathcal{P}[g_i, w, w]$ for all $w \geq s$. Our proof has three steps:

1. constructing a hard polynomial from the hitting set of $\mathcal{P}[g_i, w, w]$,
2. nonzeroness preserving variable reduction from $\mathcal{P}[g_{i-1}, s, s]$ to $\mathcal{P}[g_i, w, w]$ for some $w = s^{O(1)}$, and
3. re-use the hitting set of $\mathcal{P}[g_i, w, w]$.

Before proceeding further, we define some notations. Let $c_3 \geq 1$ be a constant such that for every degree d nonzero polynomial computed by a size s circuit, we have a $(sd)^{c_3}$-size circuit for each of its irreducible factors. Kaltofen's factoring algorithm (Lemma 2.3.3) ensures us the existence of such constant. Let

$$c_0 := \lceil 16c_3 \rceil, \; c_1 := \lceil 50ec_3 \rceil \text{ and } c_2 := 1 + c_1^2 .$$

Let ε be defined as $\varepsilon(w) := 2\lceil \log^{\circ i} w \rceil$. Now we discuss the steps in detail.

Constructing hard polynomial: Let $m_w := c_1\varepsilon(w)$ and $\delta_w := \lceil w^{3e/m_w} \rceil$. Since $\delta_w^{m_w}$ is greater than w^e, using Lemma 2.3.4, we get a polynomial q_w from the hitting set of $\mathcal{P}[g_i, w, w]$ such that

- m_w-variate and the individual degree is less than δ_w, so the degree is less than $\delta_w m_w \leq w^{o(1)}$.
- is not computable by size-w circuits, since $q_w \notin \mathcal{P}[g_i, w, w]$, the degree $\leq w$ and the number of variables $m_w \leq g_i(w)$.
- has a depth-2 circuit of size poly(w) and that circuit can be constructed in poly(w) time.

Variable reduction: Now we construct a nonzeroness preserving variable reduction map for polynomials in $\mathcal{P}[g_{i-1}, s, s]$ such that every nonzero polynomial in $\mathcal{P}[g_{i-1}, s, s]$ composed with the variable reduction map becomes a nonzero polynomial in $\mathcal{P}[g_i, s^{O(1)}, s^{O(1)}]$. Let $s_0 := s^{c_0}$ and $n := g_{i-1}(s)$. Let $\{S_1, \ldots, S_n\}$ be a $(c_2\varepsilon(s_0), m_{s_0}, 10\varepsilon(s_0))$-design on the variable set $\mathbf{z} = \{z_1, \ldots, z_{c_2\varepsilon(s_0)}\}$. Constants $c_2 > c_1 > 10$ will ensure the existence of the design by Lemma 2.3.7. Define for all $j \in [n]$, $q_{s_0}(S_j) =: p_j$ with S_j as the set of variables. Then, we show that for every nonzero polynomial $P(\mathbf{x}) \in \mathcal{P}[g_{i-1}, s, s]$, $P(p_1, \ldots, p_n)$ is also nonzero.

For the sake of contradiction, assume that $P(p_1, \ldots, p_n)$ is zero. Since $P(\mathbf{x})$ is nonzero, we can find the smallest $j \in [n]$ such that $P(p_1, \ldots, p_{j-1}, x_j, \ldots, x_n) =: P_1$ is nonzero, but $P_1\big|_{x_j = p_j}$ is zero. Thus, $(x_j - p_j)$ divides P_1. Let $\mathbf{a}$ be an assignment to all the variables in P_1, except x_j and the variables S_j in p_j maintaining its nonzeroness. Since P_1 is nonzero, we can find such an assignment. Now our new polynomial P_2 on the variables S_j and x_j is of the form $P_2(S_j, x_j) = P(p'_1, \ldots, p'_{j-1}, x_j, a_{j+1}, \ldots, a_n)$, where for each $i \in [j-1]$, p'_i is the polynomial on the variables $S_i \cap S_j$, and a_k's are field constants decided by the assignment $\mathbf{a}$. By the design, for each $i \in [j-1]$, $|S_i \cap S_j| \leq 10\varepsilon(s_0)$. Since p'_i are polynomials on variables $S_i \cap S_j$ of individual degree$\leq \delta_{s_0}$, each p'_i has a circuit (of the form $\Sigma\Pi$) of size at most

$$m_{s_0}\delta_{s_0} \cdot \delta_{s_0}^{10\varepsilon(s_0)} = m_{s_0}\delta_{s_0} \cdot \delta_{s_0}^{\frac{10 m_{s_0}}{c_1}} .$$

Thus, we have a circuit for P_2 of size at most s_1 and the degree at most d_1, where

$$s_1 := s + n m_{s_0}\delta_{s_0} \cdot \delta_{s_0}^{\frac{10 m_{s_0}}{c_1}} \text{ and } d_1 := s m_{s_0}\delta_{s_0}$$

Since $(x_j - p_j)$ divides P_2, we can invoke Kaltofen's factorization algorithm (Lemma 2.3.3) and get an algebraic circuit for p_j of size $(s_1 d_1)^{c_3}$. For finite fields, see the remark given after the proof.

$$\begin{aligned}
(s_1 d_1)^{c_3} &\le (snm_{s_0}\delta_{s_0} \cdot \delta_{s_0}^{\frac{10m_{s_0}}{c_1}} \cdot sm_{s_0}\delta_{s_0})^{c_3} \\
&= \left(s^2 n m_{s_0}^2 \delta_{s_0}^{2+\frac{10m_{s_0}}{c_1}} \right)^{c_3} \\
&< \left(s^{3+o(1)} \cdot \delta_{s_0}^{\frac{10m_{s_0}}{c_1}} \right)^{c_3} \qquad (\because n \le s \text{ and both } m_{s_0}, \delta_{s_0} = s^{o(1)}) \\
&< s^{(4+\ 30ec_0/c_1)c_3} \qquad (\because \delta_{s_0} = \lceil s_0^{3e/m_{s_0}} \rceil \text{ and } s_0 = s^{c_0})
\end{aligned}$$

This exponent is

$$\begin{aligned}
(4+\ 30ec_0/c_1)c_3 &= \left(\frac{4c_3}{c_0} + \frac{30ec_3}{c_1} \right) c_0 \\
&= \left(\frac{4c_3}{\lceil 16c_3 \rceil} + \frac{30ec_3}{\lceil 50ec_3 \rceil} \right) c_0 \\
&= \left(\frac{1}{4} + \frac{3}{5} \right) c_0 \quad (\because c_0 := \lceil 16c_3 \rceil \text{ and } c_1 := \lceil 50ec_3 \rceil) \\
&< c_0
\end{aligned}$$

So, $p_j = q_{s_0}(S_j)$ has a circuit of size smaller than s_0, which contradicts the hardness of q_{s_0}. Thus, $P' := P(p_1, \ldots, p_n)$ is nonzero.

Reusing the given hitting set: Now we calculate the degree and the circuit size of the polynomial P'.

- **Circuit size:** From the the property of the hard polynomial, we know each p_i has a circuit (of the form $\Sigma\Pi$) of size poly(s). So the total circuit size of P' is upper bounded by $s + ns^{O(1)}$, which is poly(s).
- **Degree:** Since the degree of each $p_i \le s^{o(1)}$ and the degree of $P \le s$, the total degree of P' is upper bounded by poly(s).

So, the polynomial $P' \in \mathcal{P}[g_i, s^{e_1}, s^{e_1}]$, for some constant e_1. Now, we use the poly(s)-size hitting set of $\mathcal{P}[g_i, s^{e_1}, s^{e_1}]$ to test the nonzeroness of P'.

Time Complexity: In brief, we have the following steps to compute the hitting set for $\mathcal{P}[g_{i-1}, s, s]$.

1. Compute the hard polynomial q_{s_0}, where $s_0 = s^{c_0}$.
2. Compute NW-design $\{S_1, \dots, S_n\}$ on the variable set $\{z_1, \dots, z_{c_2\varepsilon(s_0)}\}$, where $n = g_{i-1}(s)$.
3. Let $p_i := q_{s_0}(S_i)$. Let $\mathbf{p}(\mathbf{z}) := (p_1, \dots, p_n)$. Let $\mathcal{H}$ be the hitting set of $\mathcal{P}[g_i, s^{e_1}, s^{e_1}]$ as promised in the hypothesis. Then output
$$\mathcal{H}' := \{\mathbf{p}(\mathbf{a}) | \mathbf{a} \in \mathcal{H}\},$$
as the hitting set of $\mathcal{P}[g_{i-1}, s, s]$.

The time complexity of computing hitting set for $\mathcal{P}[g_{i-1}, s, s]$ has the following components.

1. Computing q_{s_0} takes poly(s) time, as mentioned in the properties of the hard polynomial.
2. According to the Lemma Lemma 2.3.7, computing NW-design takes $2^{c_2\varepsilon(s_0)} \leq s^{O(1)}$ time.
3. For each $\mathbf{a} \in \mathcal{H}$, computing $\mathbf{p}(\mathbf{a})$ takes poly(s) time since each p_i has a poly(s) size circuit. So, computing all the points in $\mathcal{H}'$ takes poly(s) time

So, the overall process takes poly(s) time. $\square$

Remark. When the underlying field $\mathbb{F}$ is a finite field of characteristic q, then in the variable reduction part of the above proof, instead of p_j, Kaltofen's factoring algorithm ensures us a size $(s_1 d_1)^{c_3}$ circuit for $p_j^{q^t}$ for some $t \in \mathbb{N}$. According to the proof of Lemma 2.3.4, p_j is a polynomial which vanishes at all points in the hitting set of $\mathcal{P}[g_i, s_0, s_0]$. Thus, $p_j^{q^t}$ also has the same property. This implies that $p_j^{q^t}$ also has no algebraic circuit of size$\leq$ s_0. However, our calculation show that $p_j^{q^t}$ has a circuit of size $(s_1 d_1)^{c_3} < s_0$, which leads to the contradiction we need to run our argument.

Now we give the proof of Perfect Bootstrapping.

Theorem 3.1.1 (restated). *Let c be a positive integer. For all sufficiently large s, suppose that we have a poly(s)-time computable hitting set for $\lceil \log^{\circ c} s \rceil$-variate polynomials computed by size s degree s circuits. Then, we have a poly(s)-time computable hitting set* *for the class of s-variate polynomials computed by size s degree s circuits and Conjecture 1 holds.*

Proof. Consider the following two statements. **S1:** we have a poly(s)-time computable hitting set for size s degree s circuits over $\lceil \log^{\circ c} s \rceil$ variables, and **S2:** we have a poly(s)-time computable hitting set for $\lceil \log^{\circ c} s \rceil$-variate degree s polynomials computed by size s circuits. According to the theorem statement, **S1** is our given hypothesis. However, in this proof, we work with **S2** which is stronger than **S1**, as in the former case circuits may have degree *larger* than s. So we first argue that they are equivalent up to polynomial overhead. **S2** trivially implies **S1**. For the other direction, we invoke Lemma 2.3.11. It ensures that for any size s circuit C computing a degree s polynomial, we have an s^4-size s-degree circuit C' computing the same polynomial. Now apply **S1** for s^4-size s-degree circuits and get a poly(s)-time computable hitting set for C. Next, we focus on designing poly(s)-size hitting set for s-variate degree s polynomials computed by size s circuits, using our stronger hypothesis **S2**.

For any nonnegative integer i, let g_i be a function defined as $g_i(s) := (\log^{\circ i} s)^2$. Since $g_{c+1}(s) \leq \lceil \log^{\circ c} s \rceil$, from the hypothesis we get a poly(s) time computable hitting set for $\mathcal{P}[g_{c+1}(s), s, s]$. Now we apply the Lemma 3.2.1 $c+1$ times and get a poly(s)-time computable hitting set for $\mathcal{P}[s, s, s]$.

Now we show that Conjecture 1 holds. For some constant e, we obtained a s^e-time computable hitting set for $\mathcal{P}[s, s, s]$. Let $m := \lceil (e+1) \log s \rceil$. Then applying Lemma 2.3.4, we get a family of multilinear polynomials $\{q_m\}_{m \geq 1}$ such that computable in $2^{O(m)}$-time, but has no $2^{o(m)}$-size circuit. Now using Lemma 2.3.5, we get Conjecture 1. $\square$

3.3 Bootstrapping of sub-exponential size hitting set

First, we recall the following standard definition.

subexp: A function $f(s)$ is in *subexp* if $f(s) = \exp(s^{o(1)})$. For example, $2^{\sqrt{s}} \notin$ subexp, but $\exp(2^{\sqrt{\log s}})$ is in subexp. One can recall the standard complexity class, SUBEXP $:= \cap_{\epsilon > 0}$DTIME$(\exp(n^{\epsilon}))$. Basically, these are decision problems whose time-complexity is a subexp function.

In our next theorem, we show bootstrapping of sub-exponential size hitting set.

Theorem 3.3.1 (Subexp bootstrap)**.** *Let f be a function in subexp. Suppose that we have a poly($f(s)$)-time computable hitting set for $10\lceil\log f(s)\rceil$-variate polynomials computed by size s degree s circuits. Then, we have a sub-exponential time, more precisely in $\exp(O(\log^2 f(s^c)))$ time for some constant c, computable hitting set for s-variate polynomials computed by size s degree s circuits. Furthermore, either $E \nsubseteq \#P/poly$ or VNP$\neq$VP.*

Remark. The hitting set size $f(s)$ in the hypothesis of the above theorem is a sub-exponential function, which is a much weaker assumption compare to the hitting set size we considered in the hypothesis of Theorem 3.1.1. On the other hand, the number of variables $10\lceil\log f(s)\rceil$ becomes much larger than the number of variables ($\log^{\circ c} s$) we considered in the hypothesis of Theorem 3.1.1. Therefore, the amount of variable stretch we obtain in the conclusion is much less compare to Theorem 3.1.1. Also, the final hitting set is not as strong as Theorem 3.1.1. Since the above theorem is weaker than Theorem 3.1.1 on some points as well as stronger on some other points, it is hard to compare them. Hence, we see the above theorem as a bootstrapping of hitting set in a different setting compare to Theorem 3.1.1.

Proof. Our proof is divided into three parts. First, we show how to construct a hard polynomial family using sub-exponential time hitting set. Next, we show a nontrivial variable reduction for circuits using the hard polynomial family. Finally, we use Corollary 2.3.2 and get our desired hitting set. For lower bound part, we apply a transformation on our hard polynomial family and show that the new polynomial family satisfies the required conditions.

Define the function $\varepsilon(s) := 10\lceil\log f(s)\rceil$. For all $s \in \mathbb{N}$, let $\mathcal{P}_s$ be the set of $\varepsilon(s)$-variate polynomials computed by size s degree s circuits. According to the hypothesis, we have, for some constant e, an $f(s)^e$-size $f(s)^e$-time computable hitting set for $\mathcal{P}_s$. Using Lemma 2.3.4, we get an m-variate polynomial $q_{m,s}$, where $m := \varepsilon(s)$, such that: **1)** its individual degree is less than δ for some constant δ, **2)** computable in poly$(f(s))$ time, and **3)** $q_{m,s} \notin \mathcal{P}_s$. Now we prove that $q_{m,s}$ is not computable by circuits of size less than $\sqrt{s}$.

For the sake of contradiction assume that $q_{m,s}$ has a circuit of size $s_1 < \sqrt{s}$. Since $f \in$ subexp, the number of variables $m = \varepsilon(s) = s^{o(1)}$. So, the degree $d := m\delta$ of $q_{m,s}$ is also $s^{o(1)}$. Now applying Lemma 2.3.11, we get a d-degree circuit C of size $O(s_1 d^2)$ for $q_{m,s}$. Since $d = s^{o(1)}$, the size of C is $< s$. This implies that $q_{m,s} \in \mathcal{P}_s$, which is a contradiction. So $q_{m,s}$ is not computed by circuits of size less than $\sqrt{s}$. This gives us a family of hard polynomials $\mathcal{F} := \{q_{m,s} \mid s \in \mathbb{N}, m = \varepsilon(s)\}$ such that it is: **1)** m-variate and the individual degree is less than δ for some constant δ, and **2)** computable in poly$(f(s))$ time but no circuits of size less than $\sqrt{s}$ can compute it.

In the following claim, we describe how to reduce variables nontrivially using $\mathcal{F}$'s hardness.

Claim 3.3.2 (Subexp var.reduction)**.** *Using $\mathcal{F}$, for some constant c, we have an $\exp(\varepsilon(s^c)^2/\log s)$-time computable variable reduction map, from s to $\lceil \varepsilon(s^c)^2/\log s \rceil$, that preserves nonzeroness for s-variate degree s polynomials computed by size s circuits. Furthermore, after the variable reduction, the degree of the new polynomial will be poly(s).*

Define $\varepsilon' := \varepsilon'(s) := \lceil \varepsilon(s^c)^2/\log s \rceil$. Using the above claim, any s-variate degree s nonzero polynomial P computed by a size s circuit can be converted to a ε'-variate nonzero polynomial P' of degree $s^{O(1)}$. P' has a $s^{O(\varepsilon')}$-time computable hitting set (Corollary 2.3.2). Total time taken (variable reduction + hitting set complexity) is $\exp(O(\varepsilon')) + \exp(O(\varepsilon')\log s)$. Since $f \in$ subexp, $\varepsilon' = s^{o(1)}$. So the total time is also in subexp. In terms of f, our time complexity is $\exp(O(\log^2 f(s_0)))$, where $s_0 = s^c$.

Now we discuss the hardness of $\{q_{m,s} \mid s \in \mathbb{N},\ m = \varepsilon(s)\}$ with respect to m, i.e., the number of variables of $q_{m,s}$. We know that $q_{m,s}$ requires circuit size $\geq \sqrt{s}$. Since $m = \varepsilon(s) = s^{o(1)}$, the circuit size is $m^{\omega(1)}$. On the other hand, $q_{m,s}$ is poly$(f(s)) = 2^{O(m)}$-time computable and has individual degree less than δ for some constant δ. Now apply the following transformation on $q_{m,s}$: replace every monomial $\prod_{i=1}^{m} x_1^{e_i}$ in the support of $q_{m,s}$ by $\prod_{i=1}^{n}\prod_{j=1}^{e_i} y_{ij}$ [2]. It is not hard to verify that our new polynomial is a δm-variate multilinear

[2] If one becomes more careful, this transformation can done with $m\lceil \log \delta \rceil$ variables. It is similar to the

polynomial with circuit size remains $m^{\omega(1)}$ and also $2^{O(m)}$-time computable. This gives a $2^{O(m)}$-time computable multilinear polynomial polynomial family $\{q_m\}_{m\geq 1}$ with circuit complexity $m^{\omega(1)}$. Then using Lemma 2.3.5, we get our lower bound result. However, the situation here is slightly different from Lemma 2.3.5. In the hypothesis of Lemma 2.3.5, we assume exponentially hard polynomial family. But, our q_m is super-polynomially hard. So, if we follow the same proof, in the conclusion we get an E-computable super-polynomially hard polynomial family which gives us E $\not\subseteq$ #P/poly or VP$\neq$VNP.

The lower bound for $q_{m,s}$ can also be calculated in terms of f. Since $m = \varepsilon(s) = 10\lceil \log f(s) \rceil$ and f is an increasing function, so $s = f^{-1}(2^{\Omega(m)})$. This implies that $q_{m,s}$ requires circuit size $(f^{-1}(2^{\Omega(m)}))^{\Omega(1)}$. □

Claim 3.3.2 (restated). *Using $\mathcal{F}$, for some constant c, we have an $\exp(\varepsilon(s^c)^2/\log s)$-time computable variable reduction map, from s to $\lceil \varepsilon(s^c)^2/\log s \rceil$, that preserves nonzeroness for s-variate degree s polynomials computed by size s circuits. Furthermore, after the variable reduction, the degree of the new polynomial will be poly(s).*

Proof. Idea is the same as in the proof of Lemma 2.3.8. Technical difference is due to the different parameters of Nisan-Wigderson design. Here we provide the details.

Let $\mathcal{P}$ be the set of n-variate degree s polynomials computed by size s circuits. The number of variables $n \leq s$. Let $\varepsilon' := \varepsilon'(s) := \lceil \varepsilon(s^c)^2/\log s \rceil$. Constant c will be fixed later. Next we describe how to reduce number of variables for every $P \in \mathcal{P}$, from n to ε'.

Let $s_0 := s^c$. Let $\{S_1, \ldots, S_n\}$ be an $(\varepsilon', \varepsilon(s_0), 10\lceil \log s \rceil)$-design on the variable set $\mathbf{z} := \{z_1, \ldots, z_{\varepsilon'}\}$ (Lemma 2.3.7). Define for all $i \in [n]$, $q_{\varepsilon(s_0),s_0}(S_i) =: p_i$ with S_i as the set of variables. For p_i, we do not have circuit of size $< \sqrt{s_0}$. Then, we show that for any nonzero polynomial $P(\mathbf{x}) \in \mathcal{P}$, $P(p_1, \ldots, p_n)$ is also nonzero.

For the sake of contradiction, assume that $P(p_1, \ldots, p_n) = 0$. Since $P(\mathbf{x})$ is nonzero, we can find the smallest $j \in [n]$ such that $P(p_1, \ldots, p_{j-1}, x_j, \ldots, x_n) =: P_1$ is nonzero, but $P_1\big|_{x_j=p_j}$ is zero. Thus, $(x_j - p_j)$ divides P_1. Let $\mathbf{a}$ be an assignment on all the variables in P_1, except x_j and the variables S_j in p_j, with the property: P_1 at $\mathbf{a}$ is nonzero. Since P_1 is

transformation we mentioned in the proof of Lemma 2.3.5. Only difference is that here we applying this transformation on the monomials.

nonzero, we can find such an assignment. Now our new polynomial P_2 on the variables S_j and x_j is of the form:

$$P_2(S_j, x_j) \;=\; P(p'_1, \ldots, p'_{j-1}, x_j, a_{j+1}, \ldots, a_n)$$

where, for each $i \in [j-1]$, p'_i is the polynomial on the variables $S_i \cap S_j$, and a_k's are field constants decided by the assignment $\mathbf{a}$. By the design, for each $i \in [j-1]$, $|S_i \cap S_j| \leq 10\lceil \log s \rceil$. Since p'_i's are polynomials on variables $S_i \cap S_j$ of individual-degree $< \delta$, each p'_i has a circuit (of form $\Sigma\Pi$) of size at most $10\lceil \log s\rceil \delta \cdot \delta^{10\lceil \log s \rceil} \leq s^{c_0}$ for some constant c_0. Then we have a circuit for P_2 of size at most $s_1 := s + n \cdot s^{c_0}$, and the degree of P_2 is at most $d_1 := s \cdot 10\lceil \log s \rceil \delta$. Since $(x_j - p_j)$ divides P_2, we can invoke Kaltofen's factoring algorithm (Lemma 2.3.3) and get an algebraic circuit for p_j of size $(s_1 d_1)^{c_1}$, for some constant c_1. Consequently, we have a circuit for p_j of size $\leq s^{c'_1}$ for some constant c'_1 (independent of c). On the other hand, p_j has no circuit of size $< s^{c/2}$. Pick c greater than $2c'_1$. Then we get a contradiction. So, for $c > 2c'_1$, $P(p_1, \ldots, p_n)$ remains nonzero.

The time for computing $(p_1, \ldots, p_n)$ depends on: **(1)** computing the design (i.e., $2^{O(\varepsilon')}$-time), and **(2)** computing p_i's (i.e, $2^{O(\varepsilon(s_0))}$-time). Thus, the variable reduction map is computable in $\exp(O(\varepsilon'))$ time, as $\varepsilon(s_0) \geq \log s$.

After variable reduction, the degree of the new polynomial will be $\leq s \cdot \deg(q_{\varepsilon(s_0), s_0}) = s^{O(1)}$. □

3.4 Discussion

Our Theorem 3.1.1 can also be seen through the lens of *fixed-parameter tractable* (FPT) algorithms. In FPT-algorithm, one provides input with multiple parameters, with the intention that the running time will be polynomial in input size but possibly exponential (or worse) in other parameters [DF13]. Here, circuit is the primary input and the running time of the FPT-blackbox PIT must depend polynomially on the circuit size s. We consider the number variables n as the extra parameter in the FPT-algorithm for PIT. Then, our Theorem 3.1.1 says that if, for some $c \in \mathbb{N}$, we have a blackbox fpt-algorithm for PIT of running time $\text{poly}(s) \cdot \exp^{\circ c}(n)$, we completely solve PIT. This reduces the dependence of PIT algorithm on the number of variables significantly.

One immediate open question after Perfect Bootstrapping can be to show a polynomial time blackbox PIT algorithm from a polynomial size hitting set for $\log^{\star} s$-variate circuits. If we try an iterative approach like our proof of Perfect Bootstrapping to stretch the variables, the size of the final hitting set would not be polynomial. So, we need a different idea. Later, in Section 4.4 of Chapter 4, we discuss a result by [GKSS19] which resolves this question.

Chapter 4

Constant Bootstrapping

Abstract

This chapter is based on joint work with Manindra Agrawal and Nitin Saxena [AGS18].

In this chapter, we study the bootstrapping of hitting sets for constant variate circuits. Unlike Perfect Bootstrapping in the previous chapter, here in the conclusion we get a "slightly" super polynomial ($s^{\exp\circ\exp(O(\log^\star s))}$) size hitting set for size s degree s circuits. This phenomenon of constant bootstrapping implies a powerful amplification of derandomization for PIT. We show that a hitting set for n-variate size s degree s circuits of size as bad as s^{n^δ}, where δ is a constant $< 1/2$, can be used to get an $s^{\exp\circ\exp(O(\log^\star s))}$ size hitting set for size s degree s circuits.

4.1 Our results and proof ideas

The bootstrapping idea brings forth pleasant surprises if we are willing to content ourselves with a "slightly" super-polynomial time blackbox PIT in the conclusion. In that case, we can even do bootstrapping from the hitting sets of constant variate circuits. Formally, we show the following.

Theorem 4.1.1 (Constant Bootstrapping). *Let $e \geq 2$ and $1 > \epsilon \geq (3+6\log(128e^2))/(128e^2)$ be some constants. Let n be a constant greater than or equal to $\lceil \max\{192e^2 \log(128e^2)^{1/\epsilon}, (64e^2)^{1/\epsilon}\}\rceil$. For all sufficiently large s, suppose that we have an s^e-size poly(s)-time computable hitting set for n-variate degree s polynomials computed by size s circuits. Then, we have an $s^{\exp\circ\exp(O(\log^\star s))}$-time computable hitting set for the class of s-variate polynomials computed by size s degree s circuits and Conjecture 1 holds.*

In the above theorem, the exponent e in the size of the hitting set is a constant just below $\sqrt{n}/8$, where n is the number of variables. This can be achieved from a "poor" quality blackbox PIT algorithm (thinking both s and n as independent parameters).

Theorem 4.1.2. *Let δ be a constant less than $1/2$. For some sufficiently large constant n and all $s \geq n$, suppose that we have an s^{n^δ}-size poly(s)-time computable hitting set for all n-variate polynomials computed by size s degree s circuits. Then, we have an $s^{\exp\circ\exp(O(\log^\star s))}$-time computable hitting set for the class of s-variate polynomials computed by size s degree s circuits and Conjecture 1 holds.*

The trivial derandomization of [DL78, Zip79, Sch80] gives a $(s+1)^n$-size hitting set for n-variate degree-s polynomials (Corollary 2.3.2). On the other hand, our aim is to design a poly(s)-size hitting set for s-variate polynomials computed by size s degree s circuits. Therefore, the above theorem can be seen as a powerful amplification of derandomization for PIT. It converts a "slightly" better hitting set (over the trivial one) to an "almost" optimal hitting set for PIT. Additionally, the lower bound result that it gives is truly exponential.

The overall proof strategy is similar to the proof of Perfect Bootstrapping (Theorem 3.1.1). However, we have to be more careful while fixing the associated parameters. This makes the proof technically more challenging than the proof of Theorem 3.1.1. In Perfect Bootstrapping, we assume an s^e-size hitting set, where e is a constant, for $\log^{\circ c} s$-variate degree s polynomials computed by size s circuits. On the other hand, Theorem 4.1.1 assumes an s^e-size hitting set for n-variate degree s polynomials computed by size s circuits, where $n \geq \lceil \max\{192e^2 \log(128e^2)^{1/\epsilon}, (64e^2)^{1/\epsilon}\}\rceil$ and $1 > \epsilon \geq (3+6\log(128e^2))/(128e^2)$

are constants. In both cases, our hypotheses demand improved hitting set over the trivial ones (namely, $(s+1)^{\log^{\circ c} s}$ and $(s+1)^n$ respectively). This is the common strength of both the hypotheses, which is exploited in the proofs.

In Theorem 4.1.1 we desire, for a given e, to find the minimum number of variables for which we can reach the conclusion. This imposes more technical challenges and in many steps of the proof we have to work with much finer parameters compare to Theorem 3.1.1. For example, our calculation suggests that for $e = 2$, the number of variables that we need is $n = 6913$ (or, for $e = 3$, $n = 17574$ suffices).

Like the proof of Theorem 3.1.1, in each inductive step, we stretch the number of variables exponentially. However, here we finally stretch n variables to s variables, where n is a constant. Therefore, we need around $\log^\star s$ steps, which is non-constant with respect to the circuit size s. We show that if we have an (s^{t_i}, s^{f_i})-hitting set, in the i-th induction step, then in the next step we get an $(s^{t_{i+1}}, s^{f_{i+1}})$-hitting-set where $t_{i+1} = O(t_i^2)$ and $f_{i+1} := 16f_i^2$. Hence, after $\log^\star s$ steps, we get a hitting set of our desired complexity.

Like in Lemma 3.2.1 (Perfect Bootstrapping), Lemma 4.3.1 combines all the crucial tools needed in the inductive step of Theorem 4.1.1. Also, the proof outline of Lemma 4.3.1 is the same as the proof outline of Lemma 3.2.1. Our key ingredients here are again Nisan-Wigderson design and Kaltofen's factoring. However, we use them in a more optimized way. It helps us improve the relation between n, e and ϵ.

For proving Theorem 4.1.2, we show that the hypothesis of it implies the hypothesis of Constant Bootstrapping (Theorem 4.1.1) for some constants e, ϵ and n. Then, we simply apply Constant Bootstrapping result and get Theorem 4.1.2.

4.2 Preliminaries

Here, we discuss some notations and results which will be used later in the proof of Constant Bootstrapping. First, we describe a slightly better version (compare to Lemma 2.3.7) of Nisan-Wigderson(NW) design. It has helped us to get better relation between the constants (e, ϵ and n) in Theorem 4.1.1. For the definition of (ℓ, n, d)-design, see Definition 2.3.6.

Lemma 4.2.1 (NW design)**.** *There exists an algorithm which takes (ℓ, n, d), with $\ell \geq 100$ and $d \geq 13$, and a base set S of size $\ell := \lceil 4n^2/d \rceil$ as input, and outputs an (ℓ, n, d)-design $\mathcal{D}$ having $\geq 2^{d/4}$ subsets, in time $2^{O(\ell)}$.*

Proof. Proof is similar to Lemma 2.3.7. Due to differences in the parameters, we provide the details. We describe a greedy algorithm to construct $\mathcal{D}$.

```
$\mathcal{D} \leftarrow \emptyset$;
while $|\mathcal{D}| < 2^{d/4}$ do
    Find the first $n$-subset $D$ of $S$ such that $\forall I \in \mathcal{D}$, $|I \cap D| \leq d$;
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{D\}$;
end while
```

Using probabilistic method, we show that for $|\mathcal{D}| < 2^{d/4}$, we can always find an n-size subset D of S such that $\forall I \in \mathcal{D}$, $|I \cap D| \leq d$. Let D be a random subset of S, constructed by the following procedure: For all $s \in S$, s will be in D with probability $2n/\ell$. Therefore, the expected size of D is $\mathrm{Exp}[|D|] = 2n$ and for each $I \in \mathcal{D}$, the expected size of $I \cap D$ is $\mathrm{Exp}[|I \cap D|] = 2n^2/\ell \leq d/2$. Using Chernoff bound [AB09, Theorem A.14] we have that

$$\Pr[|D| < n] \;=\; \Pr\left[|D| < \left(1 - \frac{1}{2}\right) 2n\right] \;\leq\; \left(\frac{2}{e}\right)^n \quad \text{and}$$

$$\begin{aligned}
\Pr[|D \cap I| > d] \;&\leq\; \Pr\left[|D \cap I]| \geq \frac{4n^2}{\ell}\right] \; \left(\because d \geq \frac{4n^2}{\ell}\right) \\
&\leq \left(\frac{e}{4}\right)^{\frac{2n^2}{\ell}} \;\; \text{(applying Chernoff bound)} \\
&\leq \left(\frac{e}{4}\right)^{\frac{d}{2}(1-\frac{1}{\ell})} \; \left(\because \ell \leq \frac{4n^2}{d} + 1\right).
\end{aligned}$$

Let E denote the event that $|D| < n$. For all $I \in \mathcal{D}$, let E_I denote the event that

$|D \cap I| > d$. Then,

$$\begin{aligned}\Pr[E \text{ or } \exists I \in \mathcal{D}, E_I] &\leq \Pr[E] + |\mathcal{D}| \cdot \Pr[E_I] \qquad \text{(Union bound)}\\ &\leq \left(\frac{2}{e}\right)^d + 2^{\frac{d}{4}}\left(\frac{e}{4}\right)^{\frac{d}{2}(1-\frac{1}{\ell})}\\ &< 1 \; (\because \ell > d \geq 13)\,.\end{aligned}$$

Thus, an n-size subset D does exist when $|\mathcal{D}| < 2^{d/4}$ and the algorithm will grow the collection $\mathcal{D}$.

It is easy to verify that the running time of the above algorithm is $2^{O(\ell)}$. $\square$

Remark. In the above lemma, if the set S has size greater than $\frac{4n^2}{d}$, then we take a subset of S of size $\lceil \frac{4n^2}{d} \rceil$ and run the argument.

Exponent vs variables. As we mentioned, our proof of Constant Bootstrapping is iterative in nature. To keep track of how the hitting-set size and the number of variables grow at each iteration, we take two sequences of numbers: 1) $(f_i)_{i\geq 0}$ ("exponent of hitting set size") and $(m_i)_{i\geq 0}$ ("number of variables"). They are defined as follows: $f_0 \geq 2$ and $m_0 \geq 1024$ are two numbers and for all $i \geq 1$,

$$f_i := 16 f_{i-1}^2 \quad \text{and} \quad m_i := 2^{m_{i-1}/(64 f_{i-1}^2)}\,.$$

Our strategy is to use an $(m_i, \frac{m_i}{8f_i}, \frac{m_i}{16f_i^2})$-design to stretch m_i variables to m_{i+1}. We want to show that m_i grows much faster in contrast to f_i. In particular, we need m_i to be a *tetration* in i (i.e., iterated exponentiation), while f_i is "merely" a double-exponentiation in i. In the next few propositions, we prove some properties about these two sequence of numbers.

From now on we will assume that ϵ is a constant fraction satisfying $1 > \epsilon \geq \frac{3+6\log(128f_i^2)}{128f_i^2}$, for $i = 0$. Since f_i increases with i, the fraction $\frac{3+6\log(128f_i^2)}{128f_i^2}$ decreases. Thus, the constant ϵ remains larger than the latter, for all $i \geq 0$.

Proposition 4.2.2. *If, for some $i \geq 0$, $m_i \geq 192 f_i^2 \cdot \frac{1}{\epsilon}\log(128f_i^2)$, then the same relation holds between m_{i+1} and f_{i+1}.*

Proof. From the definition,

$$\begin{aligned} m_{i+1} &= 2^{\frac{m_i}{64f_i^2}} \\ &\geq 2^{\frac{192f_i^2\cdot\frac{1}{\epsilon}\log(128f_i^2)}{64f_i^2}} \qquad \text{(from hypothesis)} \\ &= 2^{\frac{3}{\epsilon}\log(128f_i^2)} \\ &= 2^{\frac{21}{\epsilon}} f_i^{\frac{6}{\epsilon}} \end{aligned}$$

and $192f_{i+1}^2 \cdot \frac{1}{\epsilon}\log(128f_{i+1}^2) = 3 \times 2^{14} f_i^4 \cdot \frac{1}{\epsilon}\log(2^{15}f_i^4)$. Since $m_{i+1} \geq 2^{21/\epsilon} f_i^{6/\epsilon} > 2^{21} f_i^6 \geq 3 \times 2^{14} f_i^4 \cdot \frac{1}{\epsilon}\log(2^{15}f_i^4)$, we prove the required statement. ☐

Proposition 4.2.3 (m_i is a tetration)**.** *Suppose that* $m_0 \geq \max\{(8f_0)^{\frac{2}{\epsilon}},\ 192f_0^2\cdot\frac{1}{\epsilon}\log(128f_0^2)\}$. *Then for all* $i \geq 0$*: 1)* $m_{i+1} \geq 2^{m_i^{1-\epsilon}}$ *and 2)* $m_{i+1} \geq 2m_i > 3456f_i^2$.

Proof. (1) Proving $m_{i+1} \geq 2^{m_i^{1-\epsilon}}$ is equivalent to proving $m_i \geq (8f_i)^{\frac{2}{\epsilon}}$. Hence, at $i = 0$, the hypothesis implies that $m_1 \geq 2^{m_0^{1-\epsilon}}$.

For $i \geq 1$, to prove $m_i \geq (8f_i)^{\frac{2}{\epsilon}}$, it is sufficient to prove that $m_{i-1} \geq 64f_{i-1}^2 \cdot \frac{2}{\epsilon}\log(8 \times 16f_{i-1}^2) = 128f_{i-1}^2 \cdot \frac{1}{\epsilon}\log(128f_{i-1}^2)$. The latter relation is true for $i = 1$. Using Proposition 4.2.2, we can claim that for all $i \geq 0$, $m_i \geq 192f_i^2 \cdot \frac{1}{\epsilon}\log(128f_i^2)$. Consequently, for all $i \geq 0$, $m_{i+1} \geq 2^{m_i^{1-\epsilon}}$.

(2) From $m_i \geq 192f_i^2 \cdot \frac{1}{\epsilon}\log(128f_i^2)$ we get that $m_i > 192f_i^2 \times 9 = 1728f_i^2$.

Proving $m_{i+1} \geq 2m_i$ is equivalent to proving $m_i \geq 64f_i^2\log(2m_i)$. For all f_i, let $g(f_i) := 192f_i^2 \cdot \frac{1}{\epsilon}\log(128f_i^2)$. Then,

$$\frac{g(f_i)}{\log(2g(f_i))} = 64f_i^2 \cdot \frac{3\log 128f_i^2}{\epsilon\log(2g(f_i))}.$$

Note that the right multiplicand of the right hand side is ≥ 1 for all i. Hence, $\frac{g(f_i)}{\log(2g(f_i))} \geq 64f_i^2$ for all i.

The hypothesis on m_0 and Proposition 4.2.2 ensure $m_i \geq g(f_i)$ for all i. Since $\frac{x}{\log 2x}$ is an increasing function for all $x \geq 2$, we can say that

$$\frac{m_i}{\log(2m_i)} \geq \frac{g(f_i)}{\log(2g(f_i))} \geq 64f_i^2.$$

☐

Once we know that m_i grows rapidly, we want to estimate the number of iterations before which it reaches s.

Proposition 4.2.4 (Iteration count)**.** *The least i, for which $m_i \geq s$, is $\leq \frac{3}{1-\epsilon} \log\left(\frac{3}{1-\epsilon}\right) + 2\log^\star s$.*

Proof. Let $i_0 := \frac{3}{1-\epsilon} \log\left(\frac{3}{1-\epsilon}\right)$. Note that for all $m \geq 2^{i_0}$, $m^{1-\epsilon} > \frac{\log m}{1-\epsilon}$. By Proposition 4.2.3 we know that m_j more than doubles as we go from j to $j+1$. Hence, for all $j \geq i_0$, $m_j > 2^{i_0}$. Therefore, for all $j \geq i_0$,

$$m_j^{1-\epsilon} > \frac{\log m_j}{1-\epsilon}, \quad \text{which implies} \quad 2^{(1-\epsilon)m_j^{1-\epsilon}} > m_j\,.$$

Also, from the previous proposition we have that $m_{j+1} \geq 2^{m_j^{1-\epsilon}}$. Hence, for all $j \geq i_0$,

$$m_{j+2} \geq 2^{2^{(1-\epsilon)\cdot m_j^{1-\epsilon}}} > 2^{m_j}\,.$$

Hence, beyond $i = i_0 + 2\log^\star s$, we have $m_i \geq s$. □

4.3 Proofs of our results

Now we describe a single step of the bootstrapping. In the proof, we assume that m_i's and f_i's satisfy the hypothesis of Proposition 4.2.3.

Lemma 4.3.1 (Induction step)**.** *Suppose that for all sufficiently large s, we have an (s^{t_i}, s^{f_i})-hitting set for m_i-variate degree s polynomials computed by size s circuits. Then we have an $(s^{t_{i+1}}, s^{f_{i+1}})$-hitting set for m-variate degree s polynomials computed by size s circuits, where $m = \min\{m_{i+1}, s^{\frac{1}{4}}\}$ and $t_{i+1} = ct_i^2$ for some constant c.*

Proof. For given m and s, we want to design hitting set for polynomials in $\mathcal{P}[m, s, s]$. From the hypothesis, for all $w \geq s$, we have a (w^{t_i}, w^{f_i})-hitting set for $\mathcal{P}[m_i, w, w]$. Without loss of generality, we can assume that m_i is less than $s^{\frac{1}{4}}$. Let $n := \min\{m_i, 16f_i^2\lceil\log s\rceil\}$. For all $w \geq s$, the hypothesis promises a (w^{t_i}, w^{f_i})-hitting set for $\mathcal{P}[n, w, w]$. See Figure 4.1 for the outline of the proof.

Constructing hard polynomial: First we describe how to construct a hard polynomial against w-size circuits, for all $w \geq s$. Let $\delta_w := \lceil w^{\frac{f_i(8f_i+1)}{n}} \rceil$ and $n_1 := \lfloor \frac{n}{8f_i} \rfloor$. Since $\delta_w^{\frac{n}{8f_i}}$ is at least $w^{f_i+\frac{1}{8}}$ and δ_w is much less than $w^{\frac{1}{8}}$ for sufficiently large value of s, $\delta_w^{n_1}$ is greater than w^{f_i}. Now using Lemma 2.3.4, for all $w \geq s$, we get a polynomial q_w such that

- individual degree is less than δ_w, n_1-variate and the total degree $n_1\delta_w \leq \sqrt{w}$
- not computable by size w circuits, since $q_w \notin \mathcal{P}[n, w, w]$, the total degree is $\leq w$ and the number of variables is $\leq n$
- computable by a depth-2 circuit of size $O(w^{f_i+\frac{1}{2}})$ and the circuit can be constructed in time $w^{O(t_i)}$.

Variable reduction map: Now we construct a nonzeroness preserving variable reduction for polynomials in $\mathcal{P}[m, s, s]$ such that every nonzero polynomial in $\mathcal{P}[m, s, s]$ composed with the variable reduction becomes a nonzero polynomial in $\mathcal{P}[n, s^{12f_i}, s^{12f_i}]$. Let $\{S_1, \ldots, S_m\}$ be an $(n, \lfloor \frac{n}{8f_i} \rfloor, \frac{n}{16f_i^2})$-design on the variable set $\mathbf{z} = \{z_1, \ldots, z_n\}$. The Lemma 4.2.1 ensures that such a design exists. Let $s_0 := s^8$. Define for all $j \in [m], p_j := q_{s_0}(S_j)$ with S_j as the set of variables. Next, we show that for any nonzero $P \in \mathcal{P}[m, s, s]$, $P(p_1, \ldots, p_m)$ is also nonzero.

For the sake of contradiction, assume that $P(p_1, \ldots, p_m)$ is zero. Since $P(\mathbf{x})$ is nonzero, we can find the smallest $j \in [m]$ such that $P(p_1, \ldots, p_{j-1}, x_j, \ldots, x_m) =: P_1$ is nonzero, but $P_1\big|_{x_j=p_j}$ is zero. Thus, $(x_j - p_j)$ divides P_1. Let $\mathbf{a}$ be an assignment on all the variables in P_1, except x_j and the variables S_j in p_j, with the property: P_1 at $\mathbf{a}$ is nonzero. Since P_1 is nonzero, we can find such an assignment. Now our new polynomial P_2, on the variables S_j and x_j, is of the form $P_2(S_j, x_j) := P(p'_1, \ldots, p'_{j-1}, x_j, a_{j+1}, \ldots, a_m)$, where for each $i \in [j-1]$, p'_i is the polynomial on the variables $S_i \cap S_j$, and a_i's are field constants decided by the assignment $\mathbf{a}$. By the design, for each $i \in [j-1]$, $|S_i \cap S_j| \leq \frac{n}{16f_i^2}$. Since each p'_i is a polynomials on variable $S_i \cap S_j$ of individual degree less than δ_{s_0}, each p'_i has a circuit (of form $\Sigma\Pi$) of size at most

$$\frac{n}{16f_i^2}\delta_{s_0} \cdot \delta_{s_0}^{\frac{n}{16f_i^2}}.$$

Thus, we have a circuit for P_2 of size at most s_1 and the degree is at most d_1, where

$$s_1 := s + \frac{mn\delta_{s_0}}{16f_i^2} \cdot \delta_{s_0}^{\frac{n}{16f_i^2}} \quad \text{and} \quad d_1 := s \cdot \frac{n\delta_{s_0}}{16f_i^2}\,.$$

Since $(x_j - p_j)$ divides P_2, we can invoke Kaltofen's factoring algorithm (Lemma 2.3.3) and get an algebraic circuit for p_j of size

$$s_0' := O(s_1 d_1^2) + \tilde{O}(d_1^3)\,.$$

From Claim 4.3.2, we know that $s_0' < s_0$, for large enough s [1]. This implies that q_{s_0} has a circuit of size$\leq s_0$, which contradicts the hardness of q_{s_0}. Hence, $P' := P(p_1, \ldots, p_m)$ is nonzero n-variate polynomial.

Using the given hitting set: Now we calculate the degree and the circuit size of the polynomial P'.

- **Circuit size:** From the the property of the hard polynomial, we know each p_i has a circuit (of form $\Sigma\Pi$) of size $O(s^{8f_i+4})$. Therefore, the total circuit size of P' is upper bounded by $s + O(ms^{8f_i+4}) \leq s^{12f_i}$.
- **Degree:** Since the degree of each $p_i \leq s^4$ and the degree of $P \leq s$, the total degree of P' is upper bounded by s^5.

Hence, the polynomial $P' \in \mathcal{P}[n, s^{12f_i}, s^{12f_i}]$. Now, we use the s^{12f_i}-size hitting set to test nonzeroness of P'.

Time Complexity: In brief, we have the following steps to construct the hitting set of $\mathcal{P}[m, s, s]$.

1. Constructing the hard polynomial q_{s^8} (as $\Sigma\Pi$ circuit) from the hitting set of $\mathcal{P}[n, s^8, s^8]$.
2. Compute NW design $\{S_1, \ldots, S_m\}$ on the variable set $\mathbf{z} = \{z_1, \ldots, z_n\}$.
3. Let $p_i := q_{s^8}(S_i)$ and $\mathbf{p}(\mathbf{z}) = (p_1, \ldots, p_m)$. Let $\mathcal{H}$ be the hitting set of $\mathcal{P}[n, s^{12f_i}, s^{12f_i}]$

[1] Over finite fields, the explanation will be same as the the explanation given in the remark after the proof of Lemma 3.2.1

promised in the hypothesis. Then output

$$\mathcal{H}' := \{\mathbf{p}(\mathbf{a}) \mid \mathbf{a} \in \mathcal{H}\}$$

as a hitting set of $\mathcal{P}[m, s, s]$.

The time complexity of constructing hitting set of $\mathcal{P}[m, s, s]$ has the following components:

1. Computing q_{s^8} takes $s^{O(t_i)}$ time, as mentioned in the properties of the hard polynomial.
2. Computing Nisan-Wigderson design takes $s^{O(f_i^2)}$ time, since $n \leq 16f_i^2\lceil\log s\rceil$ (see Lemma 4.2.1).
3. Computing $\mathcal{H}$, the hitting set of $\mathcal{P}[n, s^{12f_i}, s^{12f_i}]$, takes $s^{O(t_i f_i)} = s^{O(t_i^2)}$ time. For each $\mathbf{a} \in \mathcal{H}$, computing $\mathbf{p}(\mathbf{a})$ takes s^{12f_i} time. Therefore, computing all the points in $\mathcal{H}'$ takes $s^{12f_i(f_i+1)}$ time.

Hence, the overall process takes $s^{ct_i^2}$ time, for some constant c. □

Claim 4.3.2. *Let $s_0 := s^8$. Then for sufficiently large s, $s_0' = O(s_1 d_1^2) + \tilde{O}(d_1^3)$ is less than s_0, where*

$$s_1 := s + \frac{mn\delta_{s_0}}{16f_i^2} \cdot \delta_{s_0}^{\frac{n}{16f_i^2}} \quad \textit{and} \quad d_1 := s \cdot \frac{n\delta_{s_0}}{16f_i^2}.$$

Proof. We bound each summand of s_0'. From Claim 4.3.3, we know that

$$\delta_{s_0}^{\frac{n}{16f_i^2}} \leq 2s^{5+\frac{1}{4}} \text{ and } \delta_{s_0} \leq 2s^{\frac{1}{27}+\frac{1}{432}}.$$

First part:

$$\begin{aligned}
O(s_1 d_1^2) &\leq O\left(s + \frac{mn}{16f_i^2} \cdot \delta_{s_0}^{1+\frac{n}{16f_i^2}}\right) \cdot O\left(\frac{sn\delta_{s_0}}{16f_i^2}\right)^2 \\
&\leq s^{3+o(1)}\delta_{s_0}^2 + s^{\frac{9}{4}+o(1)}\delta_{s_0}^{3+\frac{n}{16f_i^2}} \qquad \left(\because m = s^{\frac{1}{4}}, n \leq 16f_i^2\lceil\log s\rceil\right) \\
&\leq 4s^{4+o(1)} + 8s^{\frac{9}{4}+\frac{3}{27}+\frac{3}{432}+o(1)} \cdot \delta_{s_0}^{\frac{n}{16f_i^2}} \qquad (\text{substituting } \delta_{s_0} \leq 2s^{\frac{1}{27}+\frac{1}{432}}) \\
&\leq 4s^{4+o(1)} + 16s^{\frac{9}{4}+\frac{3}{27}+\frac{3}{432}+5+\frac{1}{4}+o(1)} \qquad (\text{substituting } \delta_{s_0}^{\frac{n}{16f_i^2}} \leq 2s^{5+\frac{1}{4}}) \\
&\leq 4s^{4+o(1)} + 16s^{7.62+o(1)}
\end{aligned}$$

Second part:

$$\begin{aligned}
\tilde{O}(d_1^3) &\le \tilde{O}\Big(\frac{sn\delta_{s_0}}{16f_i^2}\Big)^3\\
&\le s^{3+o(1)}\cdot\delta_{s_0}^3 \qquad \big(\because n\le 16f_i^2\lceil\log s\rceil\big)\\
&\le s^{3+o(1)}\cdot 8s^{\frac{3}{27}+\frac{3}{432}} \qquad \Big(\text{substituting } \delta_{s_0}\le 2s^{\frac{1}{27}+\frac{1}{432}}\Big)\\
&\le s^4, \text{for sufficiently large } s.
\end{aligned}$$

Hence $s_0' = O(s_1 d_1^2) + \tilde{O}(d_1^3) < s_0 = s^8$, for sufficiently large s. □

Claim 4.3.3. *For sufficiently large s, we have*

1. $\delta_{s_0}^{\frac{n}{16f_i^2}} \le 2\cdot s^{5+\frac{1}{4}}$.

2. $\delta_{s_0} \le 2s^{\frac{1}{27}+\frac{1}{432}}$.

Proof. 1.

$$\begin{aligned}
\delta_{s_0}^{\frac{n}{16f_i^2}} &\le \Big(2s_0^{\frac{f_i(8f_i+1)}{n}}\Big)^{\frac{n}{16f_i^2}} \qquad (\because \delta_{s_0} := \lceil s_0^{\frac{f_i(8f_i+1)}{n}}\rceil)\\
&\le 2^{\lceil\log s\rceil}\cdot s_0^{\frac{1}{2}+\frac{1}{32}} \qquad (\because f_i\ge 2 \text{ and } n\le 16f_i^2\lceil\log s\rceil)\\
&\le 2\cdot s^{5+\frac{1}{4}} \qquad (\because s_0 = s^8)
\end{aligned}$$

2. When $n = m_i$,

$$\begin{aligned}
\delta_{s_0} &\le 2s_0^{\frac{f_i(8f_i+1)}{m_i}}\\
&\le 2s^{\frac{64}{1728}+\frac{8}{3456}} \qquad (\because s_0 = s^8,\ m_i\ge 1728f_i^2 \text{ and } f_i\ge 2)\\
&\le 2s^{\frac{1}{27}+\frac{1}{432}}
\end{aligned}$$

and when $n = 16f_i^2\lceil\log s\rceil$,

$$\begin{aligned}
\delta_{s_0} &\le 2s_0^{\frac{f_i(8f_i+1)}{16f_i^2\lceil\log s\rceil}}\\
&\le 2s^{\frac{4}{\lceil\log s\rceil}+\frac{1}{4\lceil\log s\rceil}} \qquad (\because s_0 = s^8,\ f_i\ge 2)\\
&\le 2s^{\frac{1}{27}+\frac{1}{432}}, \text{ for large } s
\end{aligned}$$

□

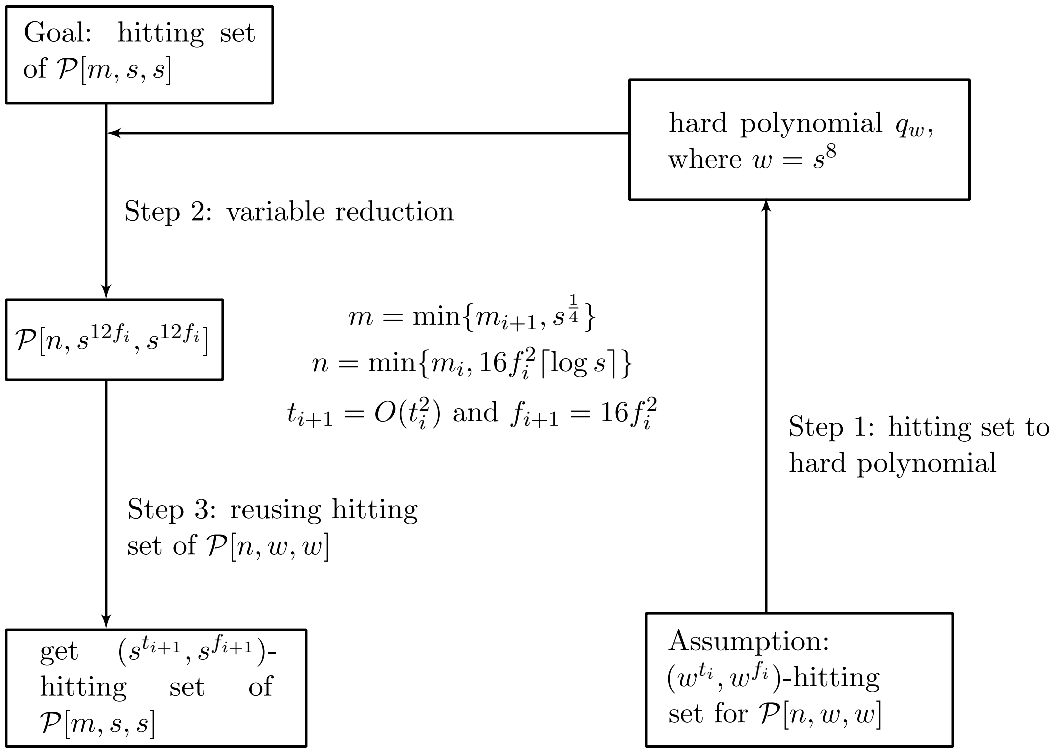

Goal: hitting set of $\mathcal{P}[m, s, s]$
hard polynomial q_w, where $w = s^8$
Step 2: variable reduction
$\mathcal{P}[n, s^{12f_i}, s^{12f_i}]$
$m = \min\{m_{i+1}, s^{\frac{1}{4}}\}$
$n = \min\{m_i, 16f_i^2\lceil \log s\rceil\}$
$t_{i+1} = O(t_i^2)$ and $f_{i+1} = 16f_i^2$
Step 1: hitting set to hard polynomial
Step 3: reusing hitting set of $\mathcal{P}[n, w, w]$
get $(s^{t_{i+1}}, s^{f_{i+1}})$-hitting set of $\mathcal{P}[m, s, s]$
Assumption: (w^{t_i}, w^{f_i})-hitting set for $\mathcal{P}[n, w, w]$

Figure 4.1: Proof structure of Lemma 4.3.1

Now we describe the proofs of our theorems.

Theorem 4.1.1 (restated). *Let $e \geq 2$ and $1 > \epsilon \geq (3 + 6\log(128e^2))/(128e^2)$ be some constants. Let n be a constant greater than or equal to $\lceil \max\{192e^2 \log(128e^2)^{1/\epsilon}, (64e^2)^{1/\epsilon}\}\rceil$. For all sufficiently large s, suppose that we have an s^e-size poly(s)-time computable hitting set for n-variate degree s polynomials computed by size s circuits. Then, we have an $s^{\exp\circ\exp(O(\log^\star s))}$-time computable hitting set for the class of s-variate polynomials computed by size s degree s circuits and Conjecture 1 holds.*

Proof. From the hypothesis, we get an (s^{t_0}, s^{f_0})-hitting set for $\mathcal{P}[m_0, s, s]$, where t_0 is some constant b, $f_0 = e$ and $m_0 = n$. The hypothesis also ensures that m_0 and f_0 satisfy the condition mentioned in the hypothesis of Proposition 4.2.3. Now we apply the Lemma 4.3.1 repeatedly and get a hitting set for $\mathcal{P}[s^{\frac{1}{4}}, s, s]$. According to Proposition 4.2.4, we have to apply the Lemma 4.3.1 k many times to get a hitting set for $\mathcal{P}[s^{\frac{1}{4}}, s, s]$, where $k = O(\log^\star s)$. It is easy to verify that

- $f_k = (16)^{2^k - 1} \cdot e^{2^k} = \exp\circ\exp(O(\log^\star s))$.

- $t_k = c^{2^k - 1} \cdot b^{2^k} = \exp \circ \exp(O(\log^\star s))$, where c is the constant mentioned in Lemma 4.3.1.

Now we have an $s^{\exp \circ \exp(O(\log^\star s))}$-time computable hitting set for $\mathcal{P}[s^{\frac{1}{4}}, s, s]$. Since $\mathcal{P}[s, s, s]$ is a subset of $\mathcal{P}[s, s^4, s^4]$, we construct a hitting set for $\mathcal{P}[s, s^4, s^4]$ in time $s^{\exp \circ \exp(O(\log^\star s))}$ using the above mentioned procedure and get a hitting set for $\mathcal{P}[s, s, s]$ of desired complexity. For more details, see Algorithm 1.

The final the hitting set we are getting for $\mathcal{P}[s, s, s]$ is not a polynomial time computable hitting set. Therefore using Lemma 2.3.4, we can not obtain a E-computable polynomial family with exponential hardness. To prove Conjecture 1, we invoke our Shallow Bootstrapping result (Theorem 5.1.1). In Shallow Bootstrapping, we show the following: if for a constant $n \geq 3$ and all sufficiently large s we have an $o(s^{n/2})$-size explicit hitting set for n-variate size s depth-4 circuits, then we get a quasi-polynomial time PIT for general circuits and Conjecture 1 holds. Since $f_0 < m_0/2$, one can see that the hypothesis of Theorem 5.1.1 is satisfied. Therefore, applying Theorem 5.1.1 we get Conjecture 1. For details, see the proof of Theorem 5.1.1 in Section 5.2. $\square$

Theorem 4.1.2 (restated). *Let δ be a constant less than $1/2$. For some sufficiently large constant n and all $s \geq n$, suppose that we have an s^{n^δ}-size poly(s)-time computable hitting set for all n-variate polynomials computed by size s degree s circuits. Then, we have an $s^{\exp \circ \exp(O(\log^\star s))}$-time computable hitting set for the class of s-variate polynomials computed by size s degree s circuits and Conjecture 1 holds.*

Proof. Suppose that we have, for some constant $\delta < 1/2$ and some sufficiently large n, an $(s^{O(1)}, s^{n^\delta})$-hitting set for size s degree s circuits over n variables. By invoking Lemma 2.3.11, we know that $\mathcal{P}[n, s, s]$ is a subset of poly(s)-size degree s circuits over n variables. Therefore, for $\mathcal{P}[n, s, s]$ we have a hitting set of size $(s^{O(1)}, s^{O(n^\delta)})$. For sufficiently large n, $s^{O(n^\delta)} \leq s^{n^{\delta_1}}$ where δ_1 is a constant from $(\delta, 1/2)$. This implies that we have a $(s^{O(1)}, s^{n^{\delta_1}})$-hitting set for $\mathcal{P}[n, s, s]$ where δ_1 is a constant less than $1/2$.

For every $\epsilon \in (2\delta_1, 1)$, we can pick some large enough n such that

$$n \geq \lceil (64e^2)^{1/\epsilon} \rceil \geq 192e^2 \log(128e^2)^{1/\epsilon}, \text{ where } e := n^{\delta_1},$$

since $\frac{2\delta_1}{\epsilon} < 1$. For such an n, we have an $(s^{O(1)}, s^e)$-hitting set for $\mathcal{P}[n, s, s]$. Now we simply invoke Theorem 4.1.1. $\square$

Remark. We give an explanation why we need the second exponent δ in Theorem 4.1.2 to be slightly less than 1/2. The reason is technical. For explaining it, we need to go back to the variable reduction step in the proof of Lemma 4.3.1. Let (ℓ, k, d) be the parameters used for the NW design in that step. To show the contradiction, we calculated the circuit size for the polynomial $P_2 = P(p'_1, \ldots, p'_{j-1}, x_j, a_{j+1}, \ldots, a_m)$, and the major contribution came from the estimation of the circuit size of each p'_i. We estimated the size needed for a depth-2 representation of each p'_i, and it becomes $\delta^d_{s_0}$. Since δ_{s_0} is approximately $s_0^{f_i/k}$, our estimated circuit size for each p'_i was around $s_0^{f_i d/k}$. For contradiction (i.e., circuit size of p_j is less than s_0), k has to be greater than $f_i d$. This says that the ratio $k : d$ is roughly equal to $f_i : 1$. According to Lemma 4.2.1, to get a (ℓ, k, d)-design we need ℓ to be around $\frac{k^2}{d}$. Hence, the (ℓ, k, d)-design we are using in Lemma 4.3.1 has its respective parameters in the ratio $f_i^2 : f_i : 1$ (roughly). This seems to be the reason why we need the second-exponent δ to be slightly less than 1/2. However, in the next section, we mention some recent developments which improve our Theorem 4.1.2 significantly.

4.4 Discussion

In the original version of our work [AGS18], improving Theorem 4.1.2 was left as an open question. Later, in subsequent works, it is improved significantly. First, [KST18] gave an improvement by weakening the hypothesis of Theorem 4.1.2. They showed that instead of an s^{n^δ}-size (for some constant $\delta < 1/2$) explicit hitting set, an $s^{n-\epsilon}$-size (for some constant $\epsilon > 0$) explicit hitting set for n-variate polynomials computed by size s degree s circuits also gives the same conclusion as Theorem 4.1.2. Their proof also works for weaker models of computation like formula, algebraic branching programs. Later, [GKSS19] further improved it. For n-variate polynomials with *individual degree* s and computable by size s circuits has a trivial hitting set of size $(s+1)^n$. They showed that even saving a single point over the trivial hitting set (i.e., hitting set of size $(s+1)^n - 1$) gives a polynomial-size

Algorithm 1 Constant Bootstrapping

```
Input: $1^s$
Output: Hitting set of $\mathcal{P}[s, s, s]$.
Assumption: $w^{f_0}$-size explicit hitting set for $\mathcal{P}[m_0, w, w]$. We denote it by $\mathcal{H}_w$.

Compute the smallest integer $k$ such that $m_k \geq s$;
$n \leftarrow \min\{m_{k-1}, 16f_{k-1}^2\lceil\log s^4\rceil\}$;

$\mathcal{H} \leftarrow$ COMPUTE HITTING-SET($s^4$, $k$);
return $\mathcal{H}$ as a hitting set of $\mathcal{P}[s, s, s]$;

function COMPUTE HITTING-SET($w$, $i$) //computes hitting set for $\mathcal{P}[m_i, w, w]$.
    if $i = 0$ then
        return $\mathcal{H}_w$;

    else if $i = k$ then
        $\mathcal{H} \leftarrow$ COMPUTE HITTING-SET($w^8$, $k-1$);
        From $\mathcal{H}$, compute the hard polynomial $q_{w^8}$;

        Let $\{S_1, \ldots, S_s\}$ be $\left(n, \lfloor\frac{n}{8f_{k-1}}\rfloor, \frac{n}{16f_{k-1}^2}\right)$-design over $\mathbf{z} = \{z_1 \ldots, z_n\}$;
        Let $p_j$ be the polynomial $q_{w^8}(S_j)$ and $\mathbf{p}(\mathbf{z}) = (p_1, \ldots, p_s)$;

        $\mathcal{H} \leftarrow$ COMPUTE HITTING-SET($w^{12f_{k-1}}$, $k-1$)
        $\mathcal{H}' \leftarrow \{\mathbf{p}(\mathbf{a}) \mid \mathbf{a} \in \mathcal{H}\}$;
        return $\mathcal{H}'$;

    else if $i < k$ then
        $i \leftarrow i - 1$;
        $\mathcal{H} \leftarrow$ COMPUTE HITTING-SET($w^8$, $i$);
        From $\mathcal{H}$, compute the hard polynomial $q_{w^8}$;

        Let $\{S_1, \ldots, S_{m_{i+1}}\}$ be $\left(m_i, \lfloor\frac{m_i}{8f_i}\rfloor, \frac{m_i}{16f_i^2}\right)$-design over $\mathbf{z} = \{z_1 \ldots, z_{m_i}\}$;
        Let $p_j$ be the polynomial $q_{w^8}(S_j)$ and $\mathbf{p}(\mathbf{z}) = (p_1, \ldots, p_{m_{i+1}})$;

        $\mathcal{H} \leftarrow$ COMPUTE HITTING-SET($w^{12f_i}$, $i$)
        $\mathcal{H}' \leftarrow \{\mathbf{p}(\mathbf{a}) \mid \mathbf{a} \in \mathcal{H}\}$;
        return $\mathcal{H}'$;
    end if
end function
```

hitting set in the conclusion. Their proof does not rely on clever combinatorial designs like Nisan-Wigderson design. Their hitting set generator is purely algebraic. Unlike [KST18], the proof in [GKSS19] does not work for weaker models like formula, algebraic branching programs.

Chapter 5

Shallow Bootstrapping

Abstract

This chapter is based on joint work with Manindra Agrawal and Nitin Saxena [AGS18].

In this chapter, we study the bootstrapping for shallow depth circuits. We show that if for a constant $n \geq 3$ and all sufficiently large s we have a $o(s^n)$-size explicit hitting set for n-variate size s depth-4 circuits, then for all s, we have an $s^{O(\log s)}$-time computable hitting set for s-variate size s degree s circuits. We also show similar bootstrapping results for two other special classes of depth-4 circuits, which are depth-4 diagonal circuits ($\Sigma \wedge \Sigma\Pi$) and preprocessed depth-3 circuits ($\Sigma\Pi\Sigma\wedge$).

5.1 Our results and proof ideas

Unlike boolean circuits, the algebraic circuits have amazing depth reduction properties. Suppose that a size s circuit C computing a d-degree polynomial. Then [VSBR83] showed that C can be converted to a poly(sd)-size $O(\log d)$-depth circuit computing the same polynomial. Later, [AV08, Koi12, Tav13] showed that the circuit C can even be converted to an $s^{O(\sqrt{d})}$-size depth-4 circuit computing the same polynomial. It was further improved by [GKKS16] where they showed that C can be reduced to a depth-3 circuit of size $s^{O(\sqrt{d})}$ [1]. One interesting outcome of these results is that any circuit computing low degree (i.e., $d = O(s)$) polynomial can be converted to a depth-4 (or depth-3) circuit of nontrivial size (i.e., better than $\Sigma\Pi$ representation). These depth reduction results have also interesting consequences in the derandomization of PIT. [AV08] showed that polynomial time blackbox PIT algorithm for depth-4 circuits gives quasi-polynomial time blackbox PIT algorithm for general circuits. A similar result also shown in [GKKS16] by assuming polynomial time blackbox PIT algorithm for depth-3 circuits.

The above results show that even if one can completely derandomize PIT for depth-4 (or depth-3) circuits, that would be a very good progress towards derandomizing PIT for general circuits. We strengthen [AV08] in the following way: our conclusion is as strong as theirs, but our hypothesis needs a hitting set which is slightly better than the trivial one. Formally, we show the following.

Theorem 5.1.1 (Shallow Bootstrapping)**.** *Let $n \in \mathbb{N}$ be a constant greater than or equal to* 3*. For all sufficiently large s, suppose that we have a $o(s^{n/2})$-size poly(s)-time computable hitting set for n-variate polynomials computed by size s depth-4 circuits. Then, we have an $s^{O(\log s)}$-time computable hitting set for the class of s-variate polynomials computed by size s degree s circuits and Conjecture 1 holds.*

Remark. If we fix $n = 3$, then the hypothesis demands a $o(s^{1.5})$-size explicit hitting set for *tri-variate* size s depth-4 circuits. While $(s+1)^3$-size hitting set is trivial to design.

Our proof of Shallow Bootstrapping is via an intermediate model. For all $s \in \mathbb{N}$, let $\mathcal{T}_s$ be the set of $n \log s$-variate multilinear polynomials computed by size s depth-4 circuits. First, we show that a $o(s^n)$-size poly(s)-time computable hitting set for $\mathcal{T}_s$ implies the conclusion of Shallow Bootstrapping. Here, the proof is along the lines of [AV08, Theorem 3.2]. First step is to construct a $2^{O(m)}$-time computable but exponentially hard (for depth-4 circuits) multilinear polynomial family $\{q_m\}_{m\geq 1}$ from the hitting set of $\mathcal{T}_s$. Then we show that the multilinear polynomial family is also exponentially hard for general circuits. From this hard polynomial family, we get $s^{O(\log s)}$-time blackbox PIT algorithm for size s degree

[1]It only holds over characteristic zero fields.

s circuits using Lemma 2.3.8 and Conjecture 1 using Lemma 2.3.5. Next, we show that the hypothesis of Shallow Bootstrapping (Theorem 5.1.1) gives a $o(s^n)$-size poly(s)-time computable hitting set for $\mathcal{T}_s$. We show that by giving a nonzeroness preserving reduction from $\mathcal{T}_s$ to n-variate polynomials computed by $O(s^2)$-size depth-4 circuits.

We also have similar results like Shallow Bootstrapping for two special classes of depth-4 circuits: 1) *depth-4 diagonal circuits*, and 2) *preprocessed depth-3 circuits.* Depth-4 diagonal circuits compute the polynomials of form $\sum_{i\in[k]} c_i f_i^{a_i}$, where f_i's are sparse polynomials (i.e., sum of monomials) of degree $\leq b$, $a_i \leq a$ and c_i's are in $\mathbb{F}$. A standard notation to denote this class is $\Sigma \wedge^a \Sigma\Pi^b$. Preprocessed depth-3 circuits ($\Sigma\Pi\Sigma\wedge$) are quite close to depth-3 circuits. We get this model by simply substituting univariate monomials in the variables of a depth-3 circuit. Unlike our previous bootstrapping theorems, these results require the characteristic of the underlying field to be zero. For depth-4 diagonal circuits, we prove the following theorem.

Theorem 5.1.2 (Tiny variate $\Sigma \wedge^a \Sigma\Pi$)**.** *Let $n \in \mathbb{N}$ be a constant greater than or equal to* 3 *and $a(\cdot)$ be some growing function over $\mathbb{N}$. For all sufficiently large s, suppose that we have a $o(s^{n/2})$-size poly(s)-time computable hitting set for n-variate polynomials computed by size s $\Sigma \wedge^a \Sigma\Pi$ circuits. Then, we have an $s^{O(\log s)}$-time computable hitting set for the class of s-variate polynomials computed by size s degree s circuits and Conjecture 1 holds.*

For preprocessed depth-3 circuits, we show the following.

Theorem 5.1.3 (Tiny variate $\Sigma\Pi\Sigma\wedge$)**.** *Let $n \in \mathbb{N}$ be a constant greater than or equal to* 3*. For all sufficiently large s, suppose that we have a $o(s^{n/2})$-size poly(s)-time computable hitting set for n-variate polynomials computed by size s $\Sigma\Pi\Sigma\wedge$ circuits. Then, we have an $s^{O(\log s)}$-time computable hitting set for the class of s-variate polynomials computed by size s degree s circuits and Conjecture 1 holds.*

Proofs of both the theorems are similar to the proof of Shallow Bootstrapping (Theorem 5.1.1). The proofs will go via intermediate models. In the next section, we give the detailed proofs.

5.2 Proofs of our results

First we prove the bootstrapping results obtained from depth-4 diagonal circuits ($\Sigma \wedge^a \Sigma\Pi$) and preprocessed depth-3 circuits ($\Sigma\Pi\Sigma\wedge$). As we mentioned, both the proofs will go via intermediate models. In the following theorem, we describe the intermediate result needed for proving the bootstrapping result regarding $\Sigma \wedge^a \Sigma\Pi$ circuits.

Theorem 5.2.1. *Let $n \in \mathbb{N}$ be a constant greater than or equal to 2 and $a(\cdot)$ be some growing function over $\mathbb{N}$. For all sufficiently large s, suppose that we have a poly(s)-time computable hitting set of size less than s^n for $n\log s$-variate* multilinear *polynomials computed by size s $\Sigma \wedge^a \Sigma\Pi$ circuits. Then, we have an $s^{O(\log s)}$-time computable hitting set for the class of s-variate polynomials computed by size s degree s circuits and Conjecture 1 holds.*

Proof. The proof is along the lines of [AV08, Theorem 3.2]. For all $s \in \mathbb{N}$, let $\mathcal{T}_s$ be the set of $n\log s$-variate multilinear polynomials computed by size s $\Sigma \wedge^a \Sigma\Pi$ circuits. According to the hypothesis, we have a poly(s)-time computable hitting set $\mathcal{H}_s$ of size less than s^n for $\mathcal{T}_s$. First, we show how to construct an exponentially hard but $2^{O(m)}$-time computable multilinear polynomial family $\{q_m\}_{m\geq 1}$ from the hitting set of $\mathcal{T}_s$. From this hard polynomial family, we get an poly($s^{\log s}$)-time computable hitting set for size s degree s circuits by using Lemma 2.3.8 and get Conjecture 1 using Lemma 2.3.5. Next we discuss how to construct the hard polynomial family from the promised hitting set of $\mathcal{T}_s$.

Let $m := n\log s$. The number of m-variate multilinear monomials is

$$2^m = s^n > |\mathcal{H}_s|\,, \text{ for all } s \in \mathbb{N} \text{ of form } 2^k.$$

Hence, using Lemma 2.3.4, we get an m-variate multilinear polynomial $q_m \notin \mathcal{T}_s$ and it is computable in $s^{O(1)} = 2^{O(m)}$ time. Since $q_m \notin \mathcal{T}_s$, no $\Sigma \wedge^a \Sigma\Pi$ circuit of size $< s = 2^{\Theta(m)}$ can compute it. Next we show that it is also not computable by any $2^{o(m)}$-size circuit.

For the sake of contradiction, assume that q_m has a $2^{o(m)}$-size circuit. Then using Lemma 2.3.9, we get an $s_m = 2^{o(m)}$ size $\Sigma\Pi^{a_1}\Sigma\Pi$ for q_m, where $a_1(m) \leq a(m)$ and $a_1(m) = o(m)$. Applying Lemma 2.3.13, we get a circuit computing q_m of the form

$\Sigma \wedge^{a_1} \Sigma\Pi$ and size $\text{poly}(s_m \cdot 2^{a_1}) = 2^{o(m)}$ which is $< s$. This contradicts the hardness of q_m. Thus, there is no algebraic circuit for q_m of size $2^{o(m)}$. Hence proved. $\square$

Theorem 5.1.2 (restated). *Let $n \in \mathbb{N}$ be a constant greater than or equal to 3 and $a(\cdot)$ be some growing function over $\mathbb{N}$. For all sufficiently large s, suppose that we have a $o(s^{n/2})$-size poly(s)-time computable hitting set for n-variate polynomials computed by size s $\Sigma \wedge^a \Sigma\Pi$ circuits. Then, we have an $s^{O(\log s)}$-time computable hitting set for the class of s-variate polynomials computed by size s degree s circuits and Conjecture 1 holds.*

Proof. For all $s \in \mathbb{N}$, let $\mathcal{T}_s$ be the set of $n \log s$-variate multilinear polynomials computed by size s $\Sigma \wedge^a \Sigma\Pi$ circuits. For all $s \in \mathbb{N}$, let $\mathcal{P}_s$ be the set of n-variate polynomials computed by size s $\Sigma \wedge^a \Sigma\Pi$ circuits. By the hypothesis, we have a $\big(\text{poly}(s), o(s^{n/2})\big)$-hitting set for $\mathcal{P}_s$. Now we show how to convert every nonzero polynomial in $\mathcal{T}_s$ to a nonzero polynomial in $\mathcal{P}_{O(s^2)}$ in poly(s) time. Then applying the given hitting set for $\mathcal{P}_{O(s^2)}$, we get a $(\text{poly}(s), o(s^n))$-hitting set for $\mathcal{T}_s$. Next we use Theorem 5.2.1 and get our conclusion.

Now, we describe the nonzeroness preserving reduction from $\mathcal{T}_s$ to $\mathcal{P}_{O(s^2)}$. Let T be a nonzero polynomial in $\mathcal{T}_s$. Let $m := n \log s$. Partition the variable set $\{x_1, \ldots, x_m\}$ into n blocks, each of size $\log s$. For all $j \in [n]$, the jth block looks like $\{x_{u(j)+1}, x_{u(j)+2}, \ldots, x_{u(j)+\log s}\}$ where $u(j) := (j-1)\log s$. Consider the variable-reducing "local Kronecker" map $\varphi : x_{u(j)+i} \mapsto y_j^{2^i}$. Note that $\varphi(T) \in \mathbb{F}[y_1, \ldots, y_n]$. It is easy to see that $\varphi(T) \neq 0$. Basically, we use the fact that T is a nonzero multilinear polynomial over $n \log s$ variables, and φ keeps all the multilinear monomials over $n \log s$ variables distinct. Finally, $\varphi(T)$ becomes an n-variate polynomial computed by $\Sigma \wedge^a \Sigma\Pi$ circuit of size at most $s + s \cdot 2^{\log s} = O(s^2)$. Thus, the hitting set for $\mathcal{P}_{O(s^2)}$ promised in the hypothesis gives a $(\text{poly}(s), o(s^n))$-hitting set for $\mathcal{T}_s$. $\square$

Next we prove the bootstrapping result obtained from $\Sigma\Pi\Sigma\wedge$ circuits (Theorem 5.2.2). First we show the intermediate result needed for proving the bootstrapping result regarding $\Sigma\Pi\Sigma\wedge$ circuits.

Theorem 5.2.2. *Let $n \in \mathbb{N}$ be a constant greater than or equal to 2. For all sufficiently large s, suppose that we have a poly(s)-time computable hitting set of size less than s^n for*

$n\log s$-variate multilinear *polynomials computed by size s depth-3 circuits. Then, we have an $s^{O(\log s)}$-time computable hitting set for the class of s-variate polynomials computed by size s degree s circuits and Conjecture 1 holds.*

Proof. Proof will be similar to the proof of Theorem 5.2.1. Main difference is that there we were dealing with depth-4 circuits, but here we have depth-3 circuits. Therefore, we need 'depth-3-reduction' result [GKKS16] with 'depth-4-reduction' result [AV08]. For all $s \in \mathbb{N}$, let $\mathcal{T}_s$ be the set of $n\log s$-variate multilinear polynomials computed by size s depth-3 circuits. According to hypothesis, we have a poly(s)-time computable hitting set $\mathcal{H}_s$ of size less than s^n for $\mathcal{T}_s$.

First, we show that how to construct a hard multilinear polynomial family from the hitting set of $\mathcal{T}_s$. Let $m := n\log s$. The number of m-variate multilinear monomials is

$$2^m = s^n > |\mathcal{H}_s|\,, \text{ for all } s \in \mathbb{N} \text{ of form } 2^k.$$

Hence, using Lemma 2.3.4, we get an m-variate multilinear polynomial $q_m \notin \mathcal{T}_s$, and it is computable in poly$(s) = 2^{O(m)}$-time. Thus, no depth-3 circuit of size $< s = 2^{\Theta(m)}$ can compute it. Next we show that it is also not computable by any $2^{o(m)}$-size algebraic circuit.

For the sake of contradiction, assume that q_m has a $2^{o(m)}$-size circuit. Using Lemma 2.3.9, we get an $s_m = 2^{o(m)}$-size $\Sigma\Pi^a\Sigma\Pi^b$ circuit, where $a, b = o(m)$. Next applying Lemma 2.3.10, we get an s'_m =poly$(s_m 2^{a+b})$-size $\Sigma\Pi\Sigma$ circuit for q_m. Since both $a, b = o(m)$, $s'_m = 2^{o(m)}$. This contradicts the hardness of q_m. Thus, there is no algebraic circuit for q_m of size $2^{o(m)}$.

Thus, we have a multilinear polynomial family $\{q_m\}_{m\geq 1}$ such that q_m is computable in $2^{O(m)}$-time but has no circuit of size $2^{o(m)}$. Now we get poly$(s^{\log s})$-time computable hitting set for size s degree s circuits using Lemma 2.3.8 and Conjecture 1 using Lemma 2.3.5. □

Theorem 5.1.3 (restated). *Let $n \in \mathbb{N}$ be a constant greater than or equal to 3. For all sufficiently large s, suppose that we have a $o(s^{n/2})$-size poly(s)-time computable hitting set for n-variate polynomials computed by size s $\Sigma\Pi\Sigma\wedge$ circuits. Then, we have an $s^{O(\log s)}$-time computable hitting set for the class of s-variate polynomials computed by size s degree*

s circuits and Conjecture 1 holds.

Proof. The proof is similar to Theorem 5.1.2. For all $s \in \mathbb{N}$, let $\mathcal{T}_s$ be the set of $n \log s$-variate multilinear polynomials computed by size s depth-3 circuits. For all $s \in \mathbb{N}$, let $\mathcal{P}_s$ be the set of n-variate polynomials computed by size s $\Sigma\Pi\Sigma\wedge$ circuits. According to the hypothesis, we have a $o(s^{n/2})$-size poly(s)-time computable hitting set for $\mathcal{P}_s$. Next we show how to convert every nonzero polynomial in $\mathcal{T}_s$ to a nonzero polynomial in $\mathcal{P}_{O(s^2)}$ in poly(s) time.

Let T be a nonzero polynomial in $\mathcal{T}_s$. Let $m := n \log s$. Partition the variable set $\{x_1, \ldots, x_m\}$ into n blocks, each of size $\log s$. For all $j \in [n]$, the jth block looks like $\{x_{u(j)+1}, x_{u(j)+2}, \ldots, x_{u(j)+\log s}\}$ where $u(j) := (j-1)\log s$. Consider the variable-reducing "local Kronecker" map $\varphi : x_{u(j)+i} \mapsto y_j^{2^i}$. Note that $\varphi(T) \in \mathbb{F}[y_1, \ldots, y_n]$. It is easy to see that $\varphi(T) \neq 0$. Basically, we use the fact that T computes a nonzero multilinear polynomial over $n \log s$ variables, and φ keeps all the multilinear monomials over $n \log s$ variables distinct. Finally $\varphi(T)$ becomes an n-variate $\Sigma\Pi\Sigma\wedge$ circuit of size at most $s + s \cdot 2^{\log s} = O(s^2)$. Thus, using the hitting set for $\mathcal{P}_{O(s^2)}$ promised in the hypothesis, we get a (poly$(s), o(s^n)$)-hitting set for T.

Now we have a $o(s^n)$-size poly(s)-time computable hitting set for $\mathcal{T}_s$. Next applying Theorem 5.2.2, we get our conclusion. $\square$

Now we give the proof of Shallow Bootstrapping.

Theorem 5.1.1 (restated). *Let $n \in \mathbb{N}$ be a constant greater than or equal to 3. For all sufficiently large s, suppose that we have a $o(s^{n/2})$-size poly(s)-time computable hitting set for n-variate polynomials computed by size s depth-4 circuits. Then, we have an $s^{O(\log s)}$-time computable hitting set for the class of s-variate polynomials computed by size s degree s circuits and Conjecture 1 holds.*

Proof. Since both depth-4 diagonal circuits ($\Sigma \wedge^a \Sigma\Pi$) and preprocessed depth-3 circuits ($\Sigma\Pi\Sigma\wedge$) are special cases of depth-4 circuits, each of Theorem 5.1.2 and Theorem 5.1.3 gives a proof for Shallow Bootstrapping when the characteristic of the underlying field is zero. Next we discuss the reasons why Theorem 5.1.2 becomes dependent on the characteristic of the underlying field and give a brief sketch how to fix them.

There are two places where the proof of Theorem 5.1.2 becomes dependent on the characteristic of the field and both of them lie in the proof of the intermediate theorem (Theorem 5.2.1). The proof of Theorem 5.2.1 has two steps. First we show how to get an exponentially hard multilinear polynomial family $\{q_m\}_{m\geq 1}$ from the hitting set of $\mathcal{T}_s$. Then, we invoke Lemma 2.3.8 which gives the construction of a quasi-polynomial size hitting set from $\{q_m\}_{m\geq 1}$. In the first step, we need to convert a $\Sigma\Pi^a\Sigma\Pi$ to $\Sigma\wedge^a\Sigma\Pi$, and for that, we use Lemma 2.3.13 which depends on the characteristic of the field. However, for general depth-4 circuits we do not need this step.

The other place where we become dependent on field's characteristic is during the use of Lemma 2.3.8, which gives the construction of a quasi-polynomial size hitting set from $\{q_m\}_{m\geq 1}$. If the underlying field is a finite field $\mathbb{F} = \mathbb{F}_{p^r}$ of characteristic p, by invoking Kaltofen's factoring algorithm in the proof of Lemma 2.3.8, instead of q_m, we get a small size circuit for $q_m^{p^t}$ for some nonnegative integer t divisible by r. Over the finite fields, it is not known whether a small size circuit for $q_m^{p^t}$ implies a small size circuit for q_m. Therefore, like in [KI04, Remark 7.5], we redefine algebraic complexity of q_m over $\mathbb{F}$ suitably and it works. For details, see the remark given after the proof of Lemma 2.3.8.

Hence, Theorem 5.1.1 holds over any field and the proof is same as the proof of Theorem 5.1.2, except those modifications. □

Remark-1) In the above proof, observe that the way we fixed our second problem, which arises due to the finite fields is different from the approach we took for Perfect Bootstrapping to fix the same problem. For details about the approach used there, see the remark given after the proof of Lemma 3.2.1. It is currently unknown to us whether we can use some similar method (used in Perfect Bootstrapping) to fix this problem.

2) Can we get a result like Theorem 5.1.1 with depth-3 circuits in the hypothesis? At this point it is not clear how to get to arbitrarily tiny variate $\Sigma\Pi\Sigma$ circuits because: **1)** The above trick of applying local Kronecker map, which reduces the number of variables from $n\log s$ to n, increases the circuit depth to 4. **2)** If we do not go via intermediate model as we did for Theorem 5.1.1, then in this tiny variate regime the hard polynomial we get from the hitting set given in the hypothesis has degree $\geq s^{\Omega(1)}$. With such a high degree, we cannot apply depth-reduction results (Lemma 2.3.9 and 2.3.10) to amplify the hardness from depth-3 to general circuits. In the hypothesis, even for $n = o(\log s)$, if we assume poly(s)-size hitting set for n-variate size s depth-3 circuits, we do not know how to bootstrap it to a quasi-polynomial time blackbox time PIT algorithm for general circuits. Here also the degree of the hard polynomial we try to get from the promised hitting set will be $\omega(n)$. Thus, we will not be able to apply the depth reduction results.

5.3 Discussion

As we go to the larger depth (i.e., depth is $\omega(1)$ compared to size) circuits, the structure of the polynomial they compute become more complex. Hence, it becomes challenging to study those larger depth circuits. However, due to amazing depth reduction results, [AV08, GKKS16], the study of algebraic circuits, in some sense, reduces to the study of the constant depth (depth-3 or depth-4) circuits. [AV08] (respectively, [GKKS16]) showed that if have a strong enough lower bound against depth-4 (respectively, depth-3) circuits, we get a super-polynomial lower bound for general circuits. Since these constant depth circuits are much simpler, it is believed that discovering techniques to get such strong enough lower bounds against these circuits is easier compared to a direct approach of getting super-polynomial lower bound for general circuits. A Similar thing is also true for PIT. We already mentioned that [AV08] (or [GKKS16]) showed that if we have a polynomial-time blackbox PIT algorithm for depth-4 (respectively depth-3), then we have a quasi-polynomial time blackbox PIT algorithm for general circuits. Our results in this chapter strengthen those results. Now, we need a slight improvement over the trivial hitting set to get the same conclusion. We see our results positively. In near future, it may be possible to discover techniques that can give us such weak hitting set for those well-structured circuit classes studied in this chapter. We also expect that those techniques will be powerful enough to provide a direction to the solution of PIT problem.

Chapter 6

Blackbox PIT for certain Log-variate Models

Abstract

This Chapter is based on joint work with Michael A. Forbes and Nitin Saxena [FGS18].

In this chapter, we give a $\mathrm{poly}(sdk) \cdot \left(\frac{n}{\log k}\right)^{O(\log k)}$ time blackbox PIT algorithm for n-variate degree d polynomials having k dimensional partial derivative space and computable by size s circuits. When $k = \mathrm{poly}(s)$ and $n = O(\log s)$, our algorithm runs in polynomial time. Since depth-3 diagonal circuit is a prominent circuit model of computing polynomials with polynomially large dimensional partial derivative space, our algorithm gives the first polynomial-time blackbox PIT algorithm for log-variate depth-3 diagonal circuits.

6.1 Our results and proof ideas

In Section 1.2.2, we discussed how our work, bootstrapping in PIT, motivates us to study PIT for circuits with "few" variables. After the bootstrapping results, we can now focus on discovering new techniques that can give efficient PIT algorithms for the low-variate circuit models. Here, we study the *log-variate* case of a circuit model, called depth-3 diagonal circuits $(\Sigma \wedge \Sigma)$. By log-variate, we mean the number of variables is logarithmic with respect to the circuit size. Depth-3 diagonal circuits compute the sum of power of linear polynomials. We give a *polynomial-time* blackbox PIT algorithm for the log-variate case of this model. Depth-3 diagonal circuit model was introduced by [Sax09] and has since drawn significant attention of PIT research community. [Sax09] first gave a polynomial-time whitebox algorithm and exponential lower bound for this model, by introducing duality trick (Lemma 2.3.12). In subsequent work, [Kay10] gave a different polynomial-time whitebox algorithm for depth-3 diagonal circuits based on the partial derivative method, which was introduced by [NW97] to prove circuit lower bounds. However, one limitation of these approaches was that they depend on the characteristic of the underlying field. Later, [FGS13] gave an alternative proof of duality trick which depends only on the field size (as mentioned in [GKKS16, Lemma 4.7]) and [Sap13b, Chapter 3] extended [Kay10] for large enough field.

Although this model is very weak (requires $2^{\Omega(n)}$ size to even compute the monomial $x_1 \cdots x_n$), studying this model has proved quite fruitful. Duality trick was crucially used in work by [GKKS16], where they showed that depth-3 circuits, in some sense, capture the complexity of general arithmetic circuits.

As we know a polynomial time white-box algorithm for depth-3 diagonal circuits, a natural question is whether we get a polynomial time black-box algorithm for the same model. This question is addressed in some recent papers. Both [ASS13] and [FS12] gave two independent and different $s^{O(\log s)}$-time (s is the circuit size) blackbox PIT algorithms for this model. [FS13a] also gave a blackbox PIT for this model with same time complexity by exploiting its partial derivative space. Later, [FSS14] gave an $s^{O(\log \log s)}$-time blackbox PIT algorithm for this model. Mulmuley [Mul12, Mul17] showed that Noether Normalization for representations of $SL_m(\mathbb{F})$, for constant m, can be reduced to designing blackbox PIT algorithm for depth-3 diagonal circuits. We cannot give the detailed notation here and would like to refer to [Mul17, Section 9.3]. Despite a lot of effort, no polynomial-time blackbox PIT for this model is known. After depth-2 circuits (or sparse polynomials), this can be thought of as the simplest model for which no polynomial-time blackbox PIT is known. Because of its simplicity, this model is a good test case for generating new ideas in PIT.

Our method to design polynomial-time blackbox PIT algorithm for log-variate depth-3 diagonal circuits extends to a possibly more general class of polynomials. It gives a polynomial-time blackbox PIT algorithms for log-variate circuits having "low-dimensional" partial derivative space. By low-dimensional partial derivative space, we mean the dimension of the partial derivative space is polynomially bounded with respect to the circuit size. Formally, we show the following.

Theorem 6.1.1. *Let $\mathbb{F}$ be a field of characteristic 0 or greater than d. Let $\mathcal{P}$ be a set of n-variate degree d polynomials, over $\mathbb{F}$, computed by circuits of size s such that for all $P \in \mathcal{P}$, the dimension of the partial derivative space of P is at most k. Then, blackbox PIT for $\mathcal{P}$ can be solved in* $\operatorname{poly}(sdk) \cdot \left(\frac{3n}{\log k}\right)^{O(\log k)}$ *time.*

When $k = \operatorname{poly}(sd)$ and $n = O(\log sd)$, the above theorem gives a poly(sd)-time blackbox PIT algorithm for log-variate circuits computing polynomials of poly(sd)-dimensional partial derivative space (Corollary 6.2.3). Previously, the best known blackbox PIT algorithm for this model was due to [FSS14], and it runs in time $(sd)^{O(\log\log sd)}$.

Since depth-3 diagonal circuits have low-dimensional partial derivative space, the above theorem directly gives a polynomial time blackbox PIT algorithm for log-variate depth-3 diagonal circuits. In our next result, we give a polynomial time blackbox PIT algorithm for a more general class of depth-3 diagonal circuits, i.e., log-rank depth-3 diagonal circuits.

Theorem 6.1.2. *Let $\mathbb{F}$ be a field of characteristic 0 or $> d$. Let $\mathcal{D}$ be the set of n-variate degree d polynomials computed by size s depth-3 diagonal circuits with rank $O(\log sd)$. Then, blackbox PIT for $\mathcal{D}$ can be solved in poly(sd)-time.*

Our proof of Theorem 6.1.1 can be seen as a way of strengthening the blackbox PIT algorithm for depth-3 diagonal circuits given by [FS13a]. Their algorithm also works for circuits with low-dimensional partial derivative space. They first showed that every nonzero polynomial $P \in \mathcal{P}$ has a $\log k$-support monomial with nonzero coefficient. Then, they used the standard hitting-set available for polynomials having a $\log k$-support monomial with nonzero coefficient (Lemma 2.3.14). It gives a blackbox PIT algorithm for $\mathcal{P}$ which runs in time $\text{poly}(s) \cdot (nd)^{O(\log k)}$. For $n = O(\log sd)$, the running time remains super-polynomial. Hence, their algorithm does not give a polynomial time PIT for $\mathcal{P}$ for the log-variate case.

Instead of support size, we work with a different measure of monomials, i.e., cone-size. It was already shown in [For14, Corollary 4.14] (with origins in[FS13a]) that every nonzero polynomial $P \in \mathcal{P}$ has a k-cone monomial with nonzero coefficient. However, it was not known how to design an efficient PIT algorithm for circuits having a "low-cone" monomial (with nonzero coefficient) such that the running time becomes polynomial in the log-variate case. One of our main technical contributions is to do that. The strategy is to check whether the coefficients of all the monomials in P with cone-size $\leq k$ are zero. We show that the number of such monomials is small (Lemma 6.2.2), i.e., quasi-polynomial in general, but, merely polynomial in the log-variate case. Next, we give a method to extract the coefficient of a low cone monomial of a circuit (Lemma 6.2.1). Our method runs in polynomial time with respect to the cone-size of the given monomial. This gives us an efficient way to extract any low-cone monomial of a circuit. Using these two facts, we get an efficient PIT algorithm for circuits having a low-cone monomial with nonzero coefficient. Hence, we have Theorem 6.1.1.

For Theorem 6.1.2, we show a polynomial time computable and nonzero preserving reduction from a $O(\log sd)$-rank depth-3 diagonal circuit to a $O(\log sd)$-variate depth-3 diagonal circuit (Lemma 6.2.4). This can be done by applying a Vandermonde based linear map on the variables. Since a depth-3 diagonal circuit has low-dimensional partial derivative space (see Lemma 2.3.16), apply Theorem 6.1.1 to the log-variate depth-3 diagonal circuits and get $\text{poly}(sd)$-time PIT for $\mathcal{D}$.

6.2 Proofs of our results

First we show that the coefficient of a low-cone monomial can be efficiently extracted from a circuit given as blackbox.

Lemma 6.2.1 (Coefficient extraction). *Let C be a circuit which computes an n-variate degree d polynomial over a field of size greater than d. Suppose that we have blackbox access to the circuit C. Then for any monomial $m = \prod_{i\in[n]} x_i^{e_i}$, we have a* $\mathrm{poly}(|C|d, \mathrm{cs}(m))$*-time algorithm to compute the coefficient of m in C, where* $\mathrm{cs}(m)$ *denotes the cone-size of m and $|C|$ denotes the size of C.*

Proof. Our proof has two steps. First, we inductively build a circuit computing a polynomial which has two parts; one is $\mathrm{coef}_m(C)\cdot m$, and the other (that we shall call 'junk') is a polynomial where every monomial in the support of that polynomial is a proper super-monomial of m. Using this, in the next step, we construct a circuit which computes the coefficient of m. In both these steps the key is a standard interpolation trick.

We induct on the variables. For each $i \in [n]$, let $m_{[i]}$ denote $\prod_{j\in[i]} x_j^{e_j}$. We construct a circuit $C^{(i)}$ which computes a polynomial of the form,

$$C^{(i)}(\mathbf{x}) \;=\; \mathrm{coef}_{m_{[i]}}(C)\cdot m_{[i]} \;+\; C^{(i)}_{\mathrm{junk}}, \tag{6.1}$$

where for every monomial m' in the support of $C^{(i)}_{\mathrm{junk}}$, $m_{[i]}$ is a proper sub-monomial of $m'_{[i]}$.

Base case: Since $C =: C^{(0)}$ computes an n-variate degree-d polynomial, $C(\mathbf{x})$ can be written as $C(\mathbf{x}) \;=\; \sum_{j=0}^{d} c_j x_1^j$, where $c_j \in \mathbb{F}[x_2,\ldots,x_n]$. Let $\alpha_0,\ldots,\alpha_{e_1}$ be some (e_1+1) distinct elements from $\mathbb{F}$. For every α_j, let $C_{\alpha_j x_1}$ denote the circuit $C(\alpha_j x_1, x_2,\ldots,x_n)$ which computes $c_0 + c_1\alpha_j x_1 + \ldots + c_{e_1}\alpha_j^{e_1}x_1^{e_1} + \cdots + c_d\alpha_j^d x_1^d$. Since

$$M = \begin{bmatrix} 1 & \alpha_0 & \ldots & \alpha_0^{e_1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha_{e_1} & \ldots & \alpha_{e_1}^{e_1} \end{bmatrix}$$

is an invertible Vandermonde matrix, one can find an $\mathbf{a} = [a_0,\ldots,a_{e_1}] \in \mathbb{F}^{1\times(e_1+1)}$ such that $\mathbf{a}M = [\ 0,\ 0,\ \ldots,\ 1]$. Using this $\mathbf{a}$, we get the circuit $C^{(1)} := \sum_{j=0}^{e_1} a_j C^{(0)}_{\alpha_j x_1}$. Its least monomial with respect to x_1 has $\deg_{x_1} \geq e_1$, which is the property we wanted. The size of the circuit $C^{(1)}$ is at most $|C| \cdot (e_1 + 1) + (e_1 + 1)$.

Induction step $(i \to i+1)$*:* From induction hypothesis, we have the circuit $C^{(i)}$ with the properties mentioned in Equation 6.1. The polynomial computed by $C^{(i)}$ can also be written as $b_0 + b_1 x_{i+1} + \ldots + b_{e_{i+1}} x_{i+1}^{e_{i+1}} + \ldots b_d x_{i+1}^d$, where every b_j is in $\mathbb{F}[x_1, \ldots, x_i, x_{i+2}, \ldots, x_n]$. Like the proof of the base case, for $(e_{i+1} + 1)$ distinct elements $\alpha_0, \ldots, \alpha_{e_{i+1}} \in \mathbb{F}$, we get $C^{(i+1)} = \sum_{j=0}^{e_{i+1}} a_j C^{(i)}_{\alpha_j x_{i+1}}$, for some $\mathbf{a} = [a_0, \ldots, a_{e_{i+1}}] \in \mathbb{F}^{1 \times (e_{i+1}+1)}$ and the structural constraint of $C^{(i+1)}$ is easy to verify. This completes the induction. The size of the circuit $C^{(i+1)}$ is at most $|C^{(i)}| \cdot (e_{i+1} + 1) + (e_{i+1} + 1)$.

Now we describe the second step of the proof. After the first step, we get

$$C^{(n)}(\mathbf{x}) \;=\; \mathrm{coef}_m(C) \cdot m \;+\; C^{(n)}_{\text{junk}}\,,$$

where for every monomial m' in the support of $C^{(n)}_{\text{junk}}$, m is a proper sub-monomial of m'. The size of the circuit $C^{(n)}$ is $O(|C| \cdot \mathrm{cs}(m))$. Consider the circuit $C' := C^{(n)}(x_1 t, \ldots, x_n t)$ over $\mathbf{x}$ and a new variable t. Let d' be the degree of the monomial m. Then C' computes the polynomial of form

$$C'(\mathbf{x}, t) = \sum_{i=0}^{d} c_i t^i,$$

where c_i's are from $\mathbb{F}[\mathbf{x}]$ and $c_{d'}$ is $\mathrm{coef}_m(C) \cdot m$. Let $a_0, \ldots, a_d$ be $d+1$ distinct elements from $\mathbb{F}$. Then

$$\begin{bmatrix} C'(\mathbf{x}, a_0) \\ C'(\mathbf{x}, a_1) \\ \vdots \\ C'(\mathbf{x}, a_d) \end{bmatrix} = \begin{bmatrix} 1 & a_0 & \ldots & a_0^d \\ 1 & a_1 & \ldots & a_1^d \\ \vdots & \vdots & \vdots & \vdots \\ 1 & a_d & \ldots & a_d^d \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_d \end{bmatrix}$$

Since the matrix in the right hand side is an invertible matrix, we can compute $b_0, \ldots, b_d$ in $\mathbb{F}$ such that

$$\mathrm{coef}_m(C) \cdot m = \sum_{i=0}^{d} b_i C'(\mathbf{x}, a_i).$$

Let $C_1(\mathbf{x})$ be the final circuit computing $\mathrm{coef}_m(C) \cdot m$. Then evaluating C_1 at $\mathbf{1} = (1, \ldots, 1)$, we get $\mathrm{coef}_m(C)$. From the description of the final circuit C_1, it is easy to see that the time taken by the whole procedure depends polynomially on $|C|, d$ and $\mathrm{cs}(m)$. $\square$

In the following lemma, we give an estimate on the number of low-cone monomials. Our calculation shows that the number of low-cone monomials in the log-variate regime are polynomially bounded. Though, in general, they are quasi-polynomially many [Sap13a].

Lemma 6.2.2 (Counting low-cones)**.** *The number of n-variate monomials with cone-size at most k is $O(rk^2)$, where $r = \left(\frac{3n}{\log k}\right)^{\log k}$.*

Proof. First, we prove that for any fixed support set, the number of cone-size $\leq k$ monomials is less than k^2. Next, we multiply by the number of possible support sets to get the estimate.

Let $T(k, \ell)$ denote the number of cone-size$\leq k$ monomials with support set, say, exactly $\{x_1, \ldots, x_\ell\}$. Since the exponent of x_ℓ in such a monomial is at least 1 and at most $k-1$, we have the following by the disjoint-sum rule: $T(k, \ell) \leq \sum_{i=2}^{k} T(k/i, \ell - 1)$. This recurrence affords an easy inductive proof as, $T(k, \ell) < \sum_{i=2}^{k} (k/i)^2 < k^2 \cdot \sum_{i=2}^{k} \left(\frac{1}{i-1} - \frac{1}{i}\right) < k^2$.

From the definition of cone, a cone-size $\leq k$ monomial can have support size at most $\ell := \lfloor \log k \rfloor$. The number of possible support sets, thus, is $\sum_{i=0}^{\ell} \binom{n}{i}$. Using Lemma 2.3.19, we get $\sum_{i=0}^{\ell} \binom{n}{i} \leq (3n/\ell)^\ell$. $\square$

Now we give the proof of Theorem 6.1.1.

Theorem 6.1.1 (restated). *Let $\mathbb{F}$ be a field of characteristic 0 or greater than d. Let $\mathcal{P}$ be a set of n-variate degree d polynomials, over $\mathbb{F}$, computed by circuits of size s such that for all $P \in \mathcal{P}$, the dimension of the partial derivative space of P is at most k. Then, blackbox PIT for $\mathcal{P}$ can be solved in $\mathrm{poly}(sdk) \cdot \left(\frac{3n}{\log k}\right)^{O(\log k)}$ time.*

Proof. According to Lemma 2.3.15, we know that every nonzero $P \in \mathcal{P}$ has a k-cone monomial with nonzero coefficient. Using this, we can get a blackbox PIT algorithm for $\mathcal{P}$ by testing whether the coefficients of all k-cone monomials in P are zero. Next, we analyze the time complexity to do this.

We apply Lemma 6.2.1, on the circuit of P and a monomial m of cone-size $\leq k$, to get the coefficient of m in P in $\mathrm{poly}(sdk)$-time. Finally, Lemma 6.2.2 tells that we have to check at most $k^2 \cdot (3n/\log k)^{\log k}$ many monomials. Multiplying these two expressions, we get our time bound. $\square$

This immediately gives us the following corollary.

Corollary 6.2.3. *Let $\mathbb{F}$ be a field of characteristic 0 or $> d$. Let $\mathcal{P}$ be a set of n-variate d-degree polynomials, over $\mathbb{F}$, computed by circuits of size s with $n = O(\log sd)$. Suppose that, for all $P \in \mathcal{P}$, the dimension of the partial derivative space of P is poly(sd). Then, blackbox PIT for $\mathcal{P}$ can be solved in poly(sd)-time.*

Next, we discuss our result regarding depth-3 diagonal circuits. First, we recall the definition of the rank of a depth-3 diagonal circuit. Depth-3 diagonal circuits compute polynomials of form $\sum_{i=1}^{k} c_i \ell^{d_i}$, where c_i's are field constants and ℓ_i's are linear polynomials. For each ℓ_i, let f_i be the non-constant part of ℓ_i (i.e., degree one homogeneous component of ℓ_i). The rank of a depth-3 diagonal circuit denotes the dimension of the subspace generated by f_i's. Now we show a nonzeroness preserving variable reduction for depth-3 diagonal circuits.

Lemma 6.2.4 (Variable reduction)**.** *Let $P(\mathbf{x})$ be an n-variate degree d polynomial computed by a size s depth-3 diagonal circuit C over a field $\mathbb{F}$. Let r be the rank of C. Then, there exists a* poly(n)*-time computable map $\varphi : \mathbb{F}^n \leftarrow \mathbb{F}[t]^r$ such that $P(\varphi(\mathbf{x}))$ is an r-variate degree d polynomial computed by a* poly(s)*-size depth-3 diagonal circuit over $\mathbb{F}[t]$ and if $P \neq 0$, then $P(\varphi(\mathbf{x})) \neq 0$.*

Proof. The polynomial $P(\mathbf{x})$ can be written as $\sum_{i\in[k]} c_i \ell_i^{d_i}$, where ℓ_i's are linear polynomials over $\mathbb{F}$, c_i's are in $\mathbb{F}$ and $k \leq s$. Let f_i be the non-constant part of ℓ_i for all $i \in [k]$. Then $r = \mathrm{rk}_{\mathbb{F}}\{f_1, \dots f_k\}$, i.e., the dimension of the subspace generated by taking all possible linear combinations (over $\mathbb{F}$) of f_i's. Without loss of generality, we can assume that $\{f_1, \dots, f_r\}$ is a basis of the space spanned by f_i's. Then there exists an r-variate polynomial $A(z_1, \dots, z_r)$ such that $P(\mathbf{x}) = A(f_1, \dots, f_r)$.

Consider φ is the map defined as

$$\varphi(x_i) := \sum_{j=1}^{r} t^{ij} y_j \quad \text{for all } i \in [n].$$

For any set of n-variate linear polynomials S over $\mathbb{F}$ with $\text{rk}_{\mathbb{F}}(S) \leq r$, one can show that $\text{rk}_{\mathbb{F}}(S) = \text{rk}_{\mathbb{F}(t)}(\varphi(S))$ [GR08, Lemma 6.1]. Let $g_i := \varphi(f_i)$ for all $i \in [r]$. Then $\text{rk}_{\mathbb{F}(t)}\{g_1, \ldots, g_r\}$ is same as r. Now, we show that $P(\mathbf{x}) \neq 0$ implies $P(\varphi(\mathbf{x})) = A(g_1, \ldots, g_r) \neq 0$. If $P(\mathbf{x})$ is a nonzero polynomial, then A is also a nonzero polynomial. Next, applying Proposition 6.2.5, we can claim that $A(g_1, \ldots, g_r) \neq 0$.

From the definition of φ, it is clear that given an x_i, $\varphi(x_i)$ is computable in poly(n) time. Since for every x_i, $\varphi(x_i)$ is a linear polynomial over $\mathbb{F}[t]$, the polynomial $P(\varphi(\mathbf{x}))$ is a degree d polynomial computable by a poly(s)-size depth-3 diagonal circuit over $\mathbb{F}[t]$. $\square$

Proposition 6.2.5. *Let $g_1, \ldots, g_r$ are linearly independent linear forms in $\mathbf{y} = \{y_1, \ldots, y_r\}$ over some field $\mathbb{F}$. Let $A(g_1, \ldots, g_r) = 0$. Then A is the zero polynomial.*

Proof. Since $g_1, \ldots, g_r$ are linearly independent, there exists r linearly independent linear forms $p_1, \ldots, p_r$ in $\mathbf{y}$ such that for all $i \in [r]$,

$$p_i(g_1, \ldots, g_r) = y_i.$$

We can express this relation in matrix form as follows:

$$PG = I,$$

where

1. P and G are $n \times n$ matrices such that ith rows denote the linear forms p_i and g_i, respectively.
2. I is the $n \times n$ identity matrix. Its ith row denotes the linear form y_i.

From the above relation, we can say that P is the inverse of G and vice-versa. Hence, GP is also same as I. This gives us for all $i \in [r]$, $g_i(p_1, \ldots, p_r) = y_i$.

Now consider the homomorphism, θ from $\mathbb{F}[\mathbf{y}]$ to $\mathbb{F}[\mathbf{y}]$, defined as, $\theta(y_i) := p_i$ for all $i \in [r]$. From the above discussion, $\theta(g_i) = g_i(p_1, \ldots, p_r) = y_i$. Therefore, applying θ on $A(g_1, \ldots, g_r)$, we get

$$\theta(A(g_1, \ldots, g_r)) = A(\theta(g_1), \ldots, \theta(g_r)) = A(y_1, \ldots, y_r).$$

Since θ maps identity to identity, $A(g_1, \ldots, g_r) = 0$ implies A is the zero polynomial, which completes the proof. □

Theorem 6.1.2 (restated). *Let $\mathbb{F}$ be a field of characteristic 0 or $> d$. Let $\mathcal{D}$ be the set of n-variate degree d polynomials computed by size s depth-3 diagonal circuits with rank $O(\log sd)$. Then, blackbox PIT for $\mathcal{D}$ can be solved in poly(sd)-time.*

Proof. Let P be a polynomial in $\mathcal{D}$. Then, Lemma 6.2.4 gives us a nonzeroness preserving variable reduction ($n \mapsto \text{rk}(P)$) that reduces P to a $O(\log(sd))$-variate degree d polynomial P' computed by a poly(s)-size depth-3 diagonal circuit over $\mathbb{F}[t]$.

The dimension of the partial derivative space of P' (over $\mathbb{F}(t)$) is poly(s) (Lemma 2.3.16). Therefore, if P' is a nonzero polynomial, it has a poly(s)-cone monomial with nonzero coefficient. However, in P', the coefficients are not from $\mathbb{F}$, they are poly(sd)-degree univariate polynomials over $\mathbb{F}$. Hence, when we apply Lemma 6.2.1 to extract a low-cone monomial, we get a poly(sd)-degree univariate polynomial over $\mathbb{F}$. To test nonzeroness of that, we evaluate it at poly(sd) many values from $\mathbb{F}$. Therefore, the coefficient-extraction of a low-cone monomial and testing its nonzeroness can be done in poly(sd)-time. Since Lemma 6.2.2 ensures that we need to do this operation only for poly(sd) many times, we have a blackbox PIT for $\mathcal{D}$ in poly(sd)-time.

□

6.3 Discussion

After solving blackbox PIT of log-variate depth-3 diagonal circuits in polynomial time, one can try to design a polynomial time blackbox PIT algorithm for general depth-3 diagonal circuits. One major problem of extending our idea for log-variate case to general variate case is that the number of "low-cone" monomials (i.e., monomials of polynomially large cone size with respect to the circuit size) is quasi-polynomially many in general variate case. Hence, a direct extension of our idea to solve log-variate case would not work for general variate case. It would be interesting to develop some new techniques which can give us a polynomial time blackbox PIT algorithm for general depth-3 diagonal circuits.

Chapter 7

Cone-closed Bases: A stronger notion of rank concentration

Abstract

This chapter is based on joint work with Michael A. Forbes and Nitin Saxena [FGS18].

In this chapter, we introduce the concept of cone-closed basis for polynomials over the vector space $\mathbb{F}^k$, which is a stronger notion of rank concentration compared to both low-support and low-cone rank concentration. We show that if any polynomial P over $\mathbb{F}^k$ is shifted by a basis isolating weight assignment for P, the new polynomial becomes cone-closed.

7.1 Motivation and our result

In Chapter 6, we showed a polynomial time blackbox PIT for log-variate depth-3 diagonal circuits where the number of variables can be at most logarithmic compared to the circuit size. Our bootstrapping results inspires us to study such "low-variate" circuit models. The technique we developed in Chapter 6 is applicable to a more general setting than log-variate depth-3 diagonal circuits. It gives a polynomial time PIT for any log-variate circuit model which has the following property: any circuit in the model computing a nonzero polynomial has a "low-cone" monomial with nonzero coefficient. By low-cone monomial we mean the cone-size of the monomial is polynomially large compared to the circuit size. However, in general, there are very simple polynomials which do not have low-cone monomial (with nonzero coefficient). For example, consider the polynomial $x_1^{d_1} \cdots x_n^{d_n}$. It has a circuit of size $d_1 + \cdots + d_n$, but the cone-size is $(d_1 + 1) \cdots (d_n + 1)$. However, if we shift each variable by 1, the new polynomial has a circuit of almost same size and also has a low-cone monomial. Hence, it would be interesting to study the circuit models under shifts and try to find 'good' shifts which give low-cone monomial in the shifted polynomial.

In Section 1.2.3, we discussed how the study of polynomials over vector space can be helpful in designing PIT algorithms for circuit models. We also described the notion of rank concentration and its usefulness to study the polynomials over vector spaces. One important variant of rank concentration is low-support rank concentration and it has been widely used in designing many PIT algorithms. For more details, see Section 1.2.3. However, it is not clear how the notion of low-support rank concentration can help to get polynomial time PIT in log-variate regime. Hence, we defined a stronger notion of rank concentration called low-cone rank concentration, and it can be helpful in log variate regime because of the following reason: If a polynomial $P \in \mathbb{F}^k[\mathbf{x}]$ satisfies low-cone concentration property, then for every $\mathbf{c} \in \mathbb{F}^k$, the support of the polynomial $\mathbf{c}^\intercal \cdot P$ has a monomial with cone-size poly(k). Now we can apply our technique from Chapter 6 and obtain a polynomial time PIT when the number of variables is few. However, in general, a polynomial P over $\mathbb{F}^k$ may not have low-cone concentration. Therefore, it would be interesting to know some transformation ϕ such that $\phi(P)$ achieves it. In this chapter, we show that after shifting a polynomial P over $\mathbb{F}^k$ by a basis isolating weight assignment (Definition 7.1.2), the new polynomial P' achieves low-cone rank concentration. More specifically, the new polynomial P' achieves a much stronger rank concentration property (than low-cone rank concentration), which we call *cone-closed basis.* At this point, we do not know any interesting application of our result in designing PIT algorithm. However, it investigates a nontrivial structural property of polynomials over $\mathbb{F}^k$.

Definition 7.1.1 (Cone-closed Basis). *A set of monomials B is called* cone-closed set *of monomials, if for every monomial in B, all its sub-monomials also belong to B. Let P be an n-variate polynomial over $\mathbb{F}^k$. We say that P has a* cone-closed basis *if there is a cone-closed set of monomials B whose coefficients in P form a basis for the coefficient space of P.*

We recall the definition of basis isolating weight assignment. For the definition of weight assignment see Section 2.1.

Definition 7.1.2. (Basis Isolating Weight Assignment [AGKS15, Definition.5]). *A weight assignment* $\mathbf{w}$ *is called a* basis isolating weight assignment *for a polynomial $P(\mathbf{x}) \in \mathbb{F}^k[\mathbf{x}]$, if there exists a set of monomials B such that*

1. *the coefficients of the monomials in B form a basis for $sp(P)$,*
2. *weights of all the monomials in B according to $\mathbf{w}$ are distinct, and*
3. *the coefficient of every m in the support of P but not in B is in the linear span of $\{coef_{m'}(P) \mid m' \in B, \mathbf{w}(m') < \mathbf{w}(m)\}$.*

Formally, we show the following structural property of polynomials over $\mathbb{F}^k$.

Theorem 7.1.3. *Let $P(\mathbf{x})$ be an n-variate degree d polynomial over $\mathbb{F}^k$, and $\mathrm{char}(\mathbb{F}) = 0$ or $> d$. Let $\mathbf{w} = (w_1, \ldots, w_n) \in \mathbb{N}^n$ be a basis isolating weight assignment for $P(\mathbf{x})$. Then, $P(\mathbf{x} + t^{\mathbf{w}}) := P(x_1 + t^{w_1}, \ldots, x_n + t^{w_n})$ has a cone-closed basis over $\mathbb{F}(t)$.*

An outline of the proof of our theorem is as follows. First, with respect to the weight assignment $\mathbf{w}$, we define an ordering among the set of bases. Then, we show that for the basis isolating weight assignment $\mathbf{w}$, there exists a *unique minimum weight basis* for the coefficient space of P (Lemma 7.2.2). Let B be the set of monomials whose coefficients form the minimum weight basis (with respect to $\mathbf{w}$) of the coefficient space of P.

Now, we consider the set of all sub-monomials of those in B and identify a subset A that is cone-closed. We define A in an algorithmic way (see Algorithm 2). Besides the cone-closed property, A also satisfies the following algebraic property. In the *transfer matrix* $T^{(n)}$, that captures the variable-shift transformation (Equation 7.2), the sub-matrix

$T^{(n)}_{A,B}$ is *full* rank (Lemma 7.2.5). We prove that the coefficients of the monomials in A is a basis of the shifted P by studying the action of the shift on the coefficient vectors. The properties of $\mathbf{w}$ mentioned above and Cauchy-Binet Formula (Lemma 2.3.17) are crucially used in the study of the coefficient vectors after the variable-shift.

Theorem 7.1.3 has an immediate consequence that any polynomial $P(\mathbf{x})$ over $\mathbb{F}^k$, when shifted by formal variables (i.e., $P(x_1+t_1,\ldots,x_n+t_n)$), becomes cone-closed; since the weight assignment induced by the formal variables on the monomials (i.e., lexicographic ordering) is a basis isolating weight assignment. This seems to be quite a nontrivial property of polynomials (over vector spaces).

7.2 Proof of our result

First, we show that the notion of cone-closed basis subsumes the other two notions of rank concentration, i.e., low-support rank concentration and low-cone rank concentration.

Lemma 7.2.1. *Let $P(\mathbf{x})$ be a polynomial in $\mathbb{F}^k[\mathbf{x}]$. Suppose that $P(\mathbf{x})$ has a cone-closed basis. Then, $P(\mathbf{x})$ has k-cone rank concentration and $\log k$-support rank concentration.*

Proof. Let B be a cone-closed set of monomials such that it is a basis of P. Clearly, $|B| \le k$. Since B is cone-closed, for every monomial in B, all its sub-monomials are also in B. Thus, each $m \in B$ has cone size $\le k$. In other words, P has k-cone rank concentration.

Moreover, each $m \in B$ has support size $\le \log k$. Otherwise, there will be a monomial in B whose cone-size is greater than k. This is not possible, since B is cone-closed. Hence, P has $\log k$-support rank concentration. $\square$

Next, we define the notions which are useful for the proof of Theorem 7.1.3.

Bases & weights. For notations, see Section 2.1. Consider a weight assignment $\mathbf{w} = (w_1,\ldots,w_n) \in \mathbb{N}^n$ on the variables $\mathbf{x} = \{x_1,\ldots,x_n\}$. Let $B = \{m_1,\ldots,m_\ell\}$,respectively $B' = \{m'_1,\ldots,m'_\ell\}$, be a ordered set of monomials (nondecreasing with respect to $\mathbf{w}$) such that it is a basis of P. We say that $B < B'$ (with respect to $\mathbf{w}$), if there exists $i \in [\ell]$ such that for all $j < i$, $\mathbf{w}(m_j) = \mathbf{w}(m'_j)$ but $\mathbf{w}(m_i) < \mathbf{w}(m'_i)$.

We say that $B \le B'$ if either $B < B'$ or if for all $i \in [\ell]$, $\mathbf{w}(m_i) \le \mathbf{w}(m'_i)$. A basis B is called a *least basis*, if for any other basis B', $B \le B'$. Next, we describe a condition on $\mathbf{w}$ such that least basis will be unique.

Lemma 7.2.2. *If $\mathbf{w}$ is a basis isolating weight assignment for $P \in \mathbb{F}^k[\mathbf{x}]$, then P has a unique least basis B (with respect to $\mathbf{w}$). Furthermore, for any other basis B' of P, we have $\mathbf{w}(B) < \mathbf{w}(B')$.*

Proof. The proof idea is similar to the idea used for proving [AGKS15, Lemma 8]. Let ℓ be the dimension of $\mathrm{sp}(P)$. Since $\mathbf{w}$ is a basis isolating weight assignment, we get a basis B that satisfies the conditions 2 and 3 in Definition 7.1.2. We will show that B is the unique least basis. Let $B = \{m_1, \ldots, m_\ell\}$ with $\mathbf{w}(m_1) < \ldots < \mathbf{w}(m_\ell)$.

Consider any other basis $B' = \{m'_1, \ldots, m'_\ell\}$, with $\mathbf{w}(m'_1) \le \ldots \le \mathbf{w}(m'_\ell)$. Let j be the minimum number such that $m_j \ne m'_j$. Such a j exists since $B \ne B'$. Suppose that $\mathbf{w}(m_j) \ge \mathbf{w}(m'_j)$. Then $m'_j \notin \{m_j, \ldots, m_\ell\}$, since $\mathbf{w}(m'_j) \le \mathbf{w}(m_j) < \cdots < \mathbf{w}(m_\ell)$ and $m_j \ne m'_j$. As $\{m_1, \ldots, m_{j-1}\} = \{m'_1, \ldots, m'_{j-1}\}$, m'_j also does not belong to $\{m_1, \ldots, m_{j-1}\}$. This implies that $m'_j \notin B$. Therefore, according to the definition of basis isolating weight assignment, we can say that the coefficient of m'_j can be written as a linear combination of the coefficients of m_i's for $i < j$. Thus, the coefficient of m'_j can also be written as a linear combination of the coefficients of m'_i's for $i < j$. This contradicts that B' is a basis. Therefore, $\mathbf{w}(m_j) < \mathbf{w}(m'_j)$. This says that $B < B'$ for every other basis B'. Hence, B is the unique least basis of P.

Now we prove that for all $i \in [\ell]$, $\mathbf{w}(m_i) \le \mathbf{w}(m'_i)$. For the sake of contradiction assume that there exists a number a such that $\mathbf{w}(m_a) > \mathbf{w}(m'_a)$. Pick the least such a. Let V be the span of the coefficients of monomials in P whose weights are $\le \mathbf{w}(m'_a)$. Since, for all $i \in [a]$, the coefficient of m'_i is in V and all of them are linearly independent, we know that $\dim(V) \ge a$. On the other hand, for every monomial m in P of $\mathbf{w}(m) \le \mathbf{w}(m'_a) < \mathbf{w}(m_a)$, the coefficient of m can be written as a linear combination of the coefficients of m_i's where $i < a$. This implies that $\dim(V) < a$, which yields a contradiction. Thus, for all $i \in [\ell]$, $\mathbf{w}(m_i) \le \mathbf{w}(m'_i)$. Togetherwith $\mathbf{w}(m_j) < \mathbf{w}(m'_j)$ proved in the previous paragraph, we get that $\mathbf{w}(B) < \mathbf{w}(B')$. $\square$

Next we want to study the effect of shifting P by a basis isolating weight assignment. As before $P(\mathbf{x})$ is an n-variate degree d polynomial over $\mathbb{F}^k$. For a weight assignment $\mathbf{w} = (w_1, \ldots, w_n)$, by $P(\mathbf{x} + t^{\mathbf{w}})$ we denote the polynomial $P(x_1 + t^{w_1}, \ldots, x_n + t^{w_n})$. For every $\mathbf{a} \in M_{n,d}$, $\operatorname{coef}_{\mathbf{x}^{\mathbf{a}}}(P(\mathbf{x} + t^{\mathbf{w}}))$ can be expanded using the binomial expansion, and we get:

$$\sum_{\mathbf{b} \in M_{n,d}} \binom{\mathbf{b}}{\mathbf{a}} \cdot t^{\mathbf{w}(\mathbf{b}) - \mathbf{w}(\mathbf{a})} \cdot \operatorname{coef}_{\mathbf{x}^{\mathbf{b}}}(P(\mathbf{x})) \,. \tag{7.1}$$

We express this information in matrix form as

$$F' = D^{-1} T^{(n)} D \cdot F, \tag{7.2}$$

where the matrices involved are,

1. F and F': rows are indexed by the elements of $M_{n,d}$ and columns are indexed by $[k]$. In F, respectively F', the $\mathbf{a}$-th row is $\operatorname{coef}_{\mathbf{x}^{\mathbf{a}}}(P(\mathbf{x}))$, respectively $\operatorname{coef}_{\mathbf{x}^{\mathbf{a}}}(P(\mathbf{x} + t^{\mathbf{w}}))$.

2. D: is a diagonal matrix with both the rows and columns indexed by $M_{n,d}$. For $\mathbf{a} \in M_{n,d}$, $D_{\mathbf{a},\mathbf{a}} := t^{\mathbf{w}(\mathbf{a})}$.

3. $T^{(n)}$: both the rows and columns are indexed by $M_{n,d}$. For $\mathbf{a}, \mathbf{b} \in M_{n,d}$, $T^{(n)}_{\mathbf{a},\mathbf{b}} := \binom{\mathbf{b}}{\mathbf{a}}$. It is known as *transfer matrix* (for n-variables).

We prove Theorem 7.1.3 by analyzing the above matrix equation. Our method of analyzing that equation can be seen as a way of strengthening the method used in [FSS13, GKST15] to study the action of shift on polynomials over $\mathbb{F}^k$. [FSS13, Corollary 3.22] (extended version of [FSS14]) showed that when a polynomial P over $\mathbb{F}^k$ shifted by formal variables, the new polynomial becomes $\log k$-support concentrated. Later, [GKST15, Lemma 5.2] improved it and showed that if P is shifted by a basis isolating weight assignment, the new polynomial becomes $\log k$-support concentrated. One crucial difference in our proof is that we use a stronger property of the transfer matrix $T^{(n)}$ compared to them. We prove that for every $B \subseteq M_{n,d}$, there is a *cone-closed* $A \subseteq M_{n,d}$ such that the submatrix $T^{(n)}_{A,B}$ has full rank. It strengthens the property of transfer matrix shown in [FSS13, Lemma 3.19]. They showed that for every $B \subseteq M_{n,d}$, there is a $A \subseteq M_{n,d}$ of monomials of support size

$\leq \lfloor \log(|B|) \rfloor$ such that the submatrix $T^{(n)}_{A,B}$ has full rank. [GKST15, Lemma 5.3] gave a different proof of [FSS13, Lemma 3.19]. To prove our property about the transfer matrix, we describe the construction of the cone-closed set A as Algorithm 2.

Algorithm 2 Finding cone-closed set

```
1: Input: A subset $B$ of the $n$-tuples from $M_{n,d}$.
2: Output: A cone-closed $A \subseteq M_{n,d}$ with full rank $T^{(n)}_{A,B}$.
3: function FIND-CONE-CLOSED($B$, $n$)
4:     if $n = 1$ then
5:         $s \leftarrow |B|$;
       return $\{0. \dots, s-1\}$;
6:     else
7:         Let $\pi$ be the map which projects every tuple in $B$ on the first $n-1$ coordinates;
8:         $\ell \leftarrow \max_{\mathbf{f} \in \mathrm{Img}(\pi)} |\pi^{-1}(\mathbf{f})|$;
9:         $\forall i \in [\ell]$, $F_i$ collects those elements in Img($\pi$) whose preimage size$\geq i$;
10:        $A \leftarrow \emptyset$;
11:        for $i \leftarrow 1$ to $\ell$ do
12:            $S_i \leftarrow$ FIND-CONE-CLOSED($F_i, n-1$);
13:            $W_i \leftarrow \{(b_1, , \dots, b_{n-1}, i-1) \mid (b_1, \dots, b_{n-1}) \in S_i\}$
14:            $A \leftarrow A \bigcup W_i$;
15:        end for
       return $A$;
16:    end if
17: end function
```

In Algorithm 2, if the input B is a set of univariate monomials (or, $n = 1$), it is not hard to see why the output set A is cone-closed. Also, the full rank property of $T^{(1)}_{A,B}$ directly follows from Lemma 2.3.18. However, it is not obvious how the algorithm works for $n \geq 2$. Therefore, we provide a simple example with bivariate monomials explaining how the algorithm works, and our general proof will be based on this idea. For example, let $B = \{(b_1, a_1), (d_1, c_1), (d_2, c_1)\}$. Then the value of ℓ is 2. Now applying π we get $F_1 = \{a_1, c_1\}$ and $F_2 = \{c_1\}$. After the completion of 'for' loop, we get $W_1 = \{(0,0), (0,1)\}$, $W_2 = \{(1,0)\}$ and $A = \bigcup_{i=1}^{2} W_i$. To prove $T^{(2)}_{A,B}$ has full rank, we show that for any $\mathbf{u} \in \mathbb{F}^{|B|}$, $T^{(2)}_{A,B} \cdot \mathbf{u} = \mathbf{0}$ implies $\mathbf{u} = \mathbf{0}$. We can think that $T^{(2)}_{A,B}$ is a matrix consists of two disjoint submatrices $T_{W_1,B}$ and $T_{W_2,B}$. Hence, $T^{(2)}_{A,B} \cdot \mathbf{u}$ can be seen as the following way:

$$T^{(2)}_{A,B} \cdot \mathbf{u} = \begin{pmatrix} T^{(2)}_{W_1,B} \cdot \mathbf{u} \\ T^{(2)}_{W_2,B} \cdot \mathbf{u} \end{pmatrix}$$

Now we see each component individually. Let $\mathbf{u} = (u_1, u_2, u_3)^\intercal$. Then

$$T^{(2)}_{W_1,B} \cdot \mathbf{u} = \binom{b_1}{0} u_1 \begin{pmatrix} \binom{a_1}{0} \\ \binom{a_1}{1} \end{pmatrix} + \left(\binom{d_1}{0} u_2 + \binom{d_2}{0} u_3 \right) \begin{pmatrix} \binom{c_1}{0} \\ \binom{c_1}{1} \end{pmatrix}, \text{ and}$$

$$T^{(2)}_{W_2,B} \cdot \mathbf{u} = \binom{b_1}{1} u_1 \binom{a_1}{0} + \left(\binom{d_1}{1} u_2 + \binom{d_2}{1} u_3 \right) \binom{c_1}{0}$$

Now $T^{(2)}_{A,B} \cdot \mathbf{u} = \mathbf{0}$ implies that both $T^{(2)}_{W_1,B} \cdot \mathbf{u}$ and $T^{(2)}_{W_2,B} \cdot \mathbf{u}$ are zero vector. Since $(\binom{a_1}{0}, \binom{a_1}{1})^\intercal$ and $(\binom{c_1}{0}, \binom{c_1}{1})^\intercal$ are linearly independent (Lemma 2.3.18), both $\binom{b_1}{0} u_1$ and $\left(\binom{d_1}{0} u_2 + \binom{d_2}{0} u_3\right)$ are zero. This implies $u_1 = 0$. Since $T^{(2)}_{W_2,B} \cdot \mathbf{u} = \mathbf{0}$ and $u_1 = 0$, we get $\left(\binom{d_1}{1} u_2 + \binom{d_2}{1} u_3\right)$ is also zero. One can see that $\left(\binom{d_1}{0}, \binom{d_1}{1}\right)^\intercal$ and $\left(\binom{d_2}{0}, \binom{d_2}{1}\right)$ are linearly independent (again use Lemma 2.3.18). Hence, both u_2 and u_3 are zero. This implies $\mathbf{u}$ is a zero vector. In the proof, we extend this idea using induction. Observe that $T^{(2)}_{W_1,B} \cdot \mathbf{u}$ and $T^{(2)}_{W_2,B} \cdot \mathbf{u}$ can be written as $T^{(1)}_{S_1,F_1} \cdot \mathbf{u}_1$ and $T^{(1)}_{S_2,F_2} \cdot \mathbf{u}_2$, respectively, for some $\mathbf{u}_1 \in \mathbb{F}^{|F_1|}$ and $\mathbf{u}_2 \in \mathbb{F}^{|F_2|}$. This point of view will be helpful in our proof since from induction hypothesis, we can say both $T^{(1)}_{S_1,F_1}$ and $T^{(1)}_{S_2,F_2}$ have full rank. In the next few lemmas, we prove the correctness of Algorithm 2.

Lemma 7.2.3 (Comparison). *Let B and B' be two nonempty subsets of M such that $B \subseteq B'$. Let $A =$ FIND-CONE-CLOSED(B, n) and $A' =$ FIND-CONE-CLOSED(B', n) in Algorithm 2. Then $A \subseteq A'$.*

Proof. We prove the lemma using induction on n.

Base case ($n = 1$)*:* For $n = 1$, the set A is $\{0, \ldots, |B| - 1\}$ and the other one A' is $\{0, \ldots, |B'| - 1\}$. Since B is a subset of B', $|B| \leq |B'|$. Hence, A is also a subset of A'.

Induction step ($n - 1 \to n$)*:* Let ℓ, respectively ℓ', be the maximum size of preimages of π in B, respectively B'. To denote the set of all elements in $\mathrm{Img}(\pi)$ whose preimage size $\geq i$, we use F_i for the set B and F'_i for the set B'. Since $B \subseteq B'$, we have $\ell \leq \ell'$, and for all $i \in [\ell']$, $F_i \subseteq F'_i$. Therefore, from induction hypothesis, $S_i \subseteq S'_i$. Since $A = \bigcup_{i=1}^{\ell} S_i \times \{i-1\}$ and $A' = \bigcup_{i=1}^{\ell'} S'_i \times \{i - 1\}$, we deduce that $A \subseteq A'$. □

Lemma 7.2.4 (Closure). *Let B be a nonempty subset of M. If $A =$ FIND-CONE-CLOSED(B, n) in Algorithm 2, then A is cone-closed. Moreover, $|A| = |B|$.*

Proof. We prove it by induction on n.

Base case ($n = 1$)*:* For $n = 1$, $A = \{0, \ldots, |B| - 1\}$. Hence, A is cone-closed.

Induction step ($n - 1 \to n$)*:* Now $A = \bigcup_{i=1}^{\ell} S_i \times \{i - 1\}$. Let $\mathbf{f}$ be an element in A and $\mathbf{x}^{\mathbf{e}}$ be a sub-monomial of $\mathbf{x}^{\mathbf{f}}$. We will show that $\mathbf{e} \in A$. Let $\mathbf{f} = (\mathbf{f}', k)$ and $\mathbf{e} = (\mathbf{e}', t)$. Then $t \leq k$. We divide our proof into the following two cases.

Case 1 ($t = k$)*:* We have $\mathbf{f}' \in S_{k+1} = \textsc{Find-Cone-closed}(F_{k+1}, n - 1)$. By induction hypothesis, S_{k+1} is cone-closed. Since $\mathbf{e}' \leq \mathbf{f}'$, we get $\mathbf{e}' \in S_{k+1}$. Hence, $\mathbf{e} = (\mathbf{e}', k) \in S_{k+1} \times \{k\}$, which implies that it is also in A.

Case 2 ($t < k$)*:* We have $F_{k+1} \subseteq F_{t+1}$. By Lemma 7.2.3, we get $S_{k+1} \subseteq S_{t+1}$. Therefore, $\mathbf{f}' \in S_{t+1}$. From induction hypothesis, S_{t+1} is a cone-closed set. This implies that $\mathbf{e}' \in S_{t+1}$ and $\mathbf{e} \in S_{t+1} \times \{t\}$. Thus, $\mathbf{e}$ is also in A.

Since $\mathbf{e}$ was arbitrary, we deduce that A is cone-closed.

Note that $|A| = |B|$ is true when $n = 1$. Let us prove the induction step from $n - 1$ to n. Since $|A| = \sum_{i \in [\ell]} |S_i|$, and by induction hypothesis $|S_i| = |F_i|$, we deduce that $|A| = \sum_{i \in [\ell]} |F_i|$. From the definition of F_i's we get that $\text{Img}(\pi) = F_1 \supseteq F_2 \supseteq \cdots \supseteq F_\ell$. A tuple $\mathbf{e} \in \text{Img}(\pi)$ that has preimage size j is counted exactly once in each of $F_1, \ldots, F_j$, but is not counted in F_k for $j < k \leq \ell$. Thus,

$$|A| = \sum_{i \in [\ell]} |F_i| = \sum_{m \in \text{Img}(\pi)} |\pi^{-1}(m)| = |B|.$$

□

In the next lemma, we prove that the sub-matrix $T_{A,B}^{(n)}$ has full rank, where $B \subseteq M_{n,d}$ and A is the output of Algorithm 2 on input B. It requires $\text{char}(\mathbb{F}) = 0$ or greater than d.

Lemma 7.2.5 (Full rank)**.** *If* $A = \textsc{Find-Cone-Closed}(B, n)$ *for some* $B \subseteq M_{n,d}$*, then* $T_{A,B}^{(n)}$ *has full rank.*

Proof. The proof will be by induction on n.

Base case: For $n = 1$, our set $A = \{0, 1, \ldots, |B| - 1\}$. Therefore, applying Lemma 2.3.18, we get $T_{A,B}^{(n)}$ has full rank for $n = 1$.

Induction step $(n-1 \to n)$*:* In induction step, we show that for any $B \subseteq M_{n,d}$, if $A = \textsc{Find-Cone-Closed}(B, n)$, then $T_{A,B}^{(n)}$ has full rank. By B_i we denote the subset of B such that $B_i = \pi^{-1}(F_i)$, i.e., for each $\mathbf{e} \in B_i$, the size of the preimage of $\pi(\mathbf{e})$ is $\geq i$. From induction hypothesis, we can say that if $V = \textsc{Find-Cone-Closed}(U, n-1)$ for $U \subseteq M_{n-1,d}$, then $T_{U,V}^{(n-1)}$ has full rank. Using this induction hypothesis, we prove Claim 7.2.6 which is crucial for the induction step.

To show $T_{A,B}^{(n)}$ has full rank, we prove that if $T_{A,B}^{(n)} \cdot \mathbf{b}$ is a zero vector for $\mathbf{b} \in \mathbb{F}^{|B|}$, then $\mathbf{b}$ itself is zero vector. For this, we show an invariant holds at the end of each iteration of the 'for' loop in Algorithm 2. Let A_i be the the value of A at the end of ith iteration. Then our invariant is the following.

Invariant: If $T_{A_i,B}^{(n)} \cdot \mathbf{b}$ is a zero vector, then $\mathbf{b}_{\mathbf{e}} = 0$ for each $\mathbf{e} \in B$ with the preimage size of $\pi(\mathbf{e}) \leq i$.

At the end of iteration $i = 1$, we have the vector $T_{A_1,B}^{(n)} \cdot \mathbf{b}$. According to our notations, $B = B_1$ and $W_1 = A_1$. Hence, $T_{A_1,B}^{(n)} \cdot \mathbf{b}$ is same as the vector $T_{A_1,B_1}^{(n)} \cdot \mathbf{b}$. Let $\mathbf{e}$ be an element in B such that the preimage size of $\pi(\mathbf{e})$ is 1 and $\mathbf{e} = (\mathbf{f}, k)$ for some $k \in \mathbb{N}$ and $\mathbf{f} \in F_1$. Then using Claim 7.2.6, we can say that

$$\binom{k}{0} \cdot \mathbf{b}_{\mathbf{e}} = 0.$$

This implies $\mathbf{b}_{\mathbf{e}} = 0$, since $\binom{k}{0} \neq 0$.

$(i-1 \to i)$: Let $T_{A_i,B}^{(n)} \cdot \mathbf{b}$ be a zero vector. This implies that for all $j \in [i]$, $T_{W_j,B}^{(n)} \cdot \mathbf{b}$ is also a zero vector since W_j is a subset of A_i. As for all $j < i$, A_j is a subset of A_i, $T_{A_j,B}^{(n)} \cdot \mathbf{b}$ is also a zero vector. Therefore, applying the loop invariant of previous iterations, we get $\mathbf{b}_{\mathbf{e}} = 0$ for all $\mathbf{e} \in B$ with the preimage size of $\pi(\mathbf{e})$ is less than i. This has the following two implications.

1. For all $j \in [i]$, $T_{W_j,B}^{(n)} \cdot \mathbf{b}$ is same as $T_{W_j,B_j}^{(n)} \cdot \mathbf{b}^{(j)}$, where $\mathbf{b}^{(j)}$ is the projection of $\mathbf{b}$ on the coordinates indexed by B_j.

2. For proving the loop invariant at the end of ith iteration, we have to only show that $\mathbf{b}_{\mathbf{e}} = 0$ for any $\mathbf{e} \in B$ with the preimage size of $\pi(\mathbf{e})$ is exactly i.

For any such $\mathbf{e}$ there exists an $\mathbf{f} \in F_i$, such that $\mathbf{e} = (\mathbf{f}, k_1)$ for some $k_1 \in \mathbb{N}$ and $\pi^{-1}(\mathbf{f}) = \{(\mathbf{f}, k_1), \ldots, (\mathbf{f}, k_i)\}$. Since for all $j \in [i]$, $T^{(n)}_{W_j,B_j} \cdot \mathbf{b}^{(j)}$ is a zero vector and $\mathbf{b}^{(j)}$ is the projection of $\mathbf{b}$ on the coordinates indexed by B_j, using Claim 7.2.6, we get that

$$\sum_{r=1}^{i} \binom{k_r}{j-1} \mathbf{b}_{(\mathbf{f},k_r)} = 0, \text{ for all } j \in [i].$$

These set of equations we can write in matrix form as follows:

$$\begin{bmatrix} \binom{k_1}{0} & \binom{k_2}{0} & \cdots \binom{k_i}{0} \\ \binom{k_1}{1} & \binom{k_2}{1} & \cdots \binom{k_i}{1} \\ \vdots & \vdots & \vdots \\ \binom{k_1}{i-1} & \binom{k_2}{i-1} & \cdots \binom{k_i}{i-1} \end{bmatrix} \begin{bmatrix} \mathbf{b}_{(\mathbf{f},k_1)} \\ \mathbf{b}_{(\mathbf{f},k_2)} \\ \vdots \\ \mathbf{b}_{(\mathbf{f},k_i)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Now invoking Lemma 2.3.18, we get $\mathbf{b}_{(\mathbf{f},k_r)} = 0$ for all $r \in [i]$. In other words, for any $\mathbf{e} \in B$ with the preimage size of $\pi(\mathbf{e})$ is i, the coordinate $\mathbf{b}_{\mathbf{e}} = 0$.

$(i = \ell)$: Since $A_\ell = A$, the output of Find-Cone-closed(B, n), using our invariant at the end of ℓ-th iteration we deduce that $T^{(n)}_{A,B} \cdot \mathbf{b} = 0$ implies $\mathbf{b} = 0$. Thus, $T^{(n)}_{A,B}$ has full rank. ☐

Claim 7.2.6. *Let $i \in [\ell]$ and $\mathbf{u} \in \mathbb{F}^{|B_i|}$. Let $T^{(n)}_{W_i,B_i} \cdot \mathbf{u}$ be zero vector. Then, for every $\mathbf{f} \in F_i$,*

$$\sum_{(\mathbf{f},k) \in \pi^{-1}(\mathbf{f})} \binom{k}{i-1} \mathbf{u}_{(\mathbf{f},k)} = 0.$$

Proof. Let $\mathbf{e} \in B_i$, and $\mathbf{e} = (\mathbf{f}, k)$ for $\mathbf{f} \in F_i$ and $k \in \mathbb{N}$. Then $T^{(n)}_{W_i,\mathbf{e}}$, the column of $T^{(n)}_{W_i,B_i}$ indexed by $\mathbf{e}$, is same as $\binom{k}{i-1} T^{(n-1)}_{S_i,\mathbf{f}}$. Now the vector $T^{(n)}_{W_i,B_i} \cdot \mathbf{u}$ can be written as follows.

$$\begin{aligned} T^{(n)}_{W_i,B_i} \cdot \mathbf{u} &= \sum_{\mathbf{e} \in B_i} \mathbf{u}_{\mathbf{e}} T^{(n)}_{W_i,\mathbf{e}} \\ &= \sum_{\mathbf{f} \in F_i} T^{(n-1)}_{S_i,\mathbf{f}} \left(\sum_{(\mathbf{f},k) \in \pi^{-1}(\mathbf{f})} \binom{k}{i-1} \mathbf{u}_{(\mathbf{f},k)} \right) \\ &= T^{(n-1)}_{S_i,F_i} \cdot \mathbf{v}, \end{aligned}$$

where $\mathbf{v} \in \mathbb{F}^{|F_i|}$ and $\mathbf{v}_{\mathbf{f}} = \sum_{(\mathbf{f},k) \in \pi^{-1}(\mathbf{f})} \binom{k}{i-1} \mathbf{u}_{(\mathbf{f},k)}$. Therefore, if $T^{(n)}_{W_i,B_i} \cdot \mathbf{u}$ is a zero vector, then $T^{(n-1)}_{S_i,F_i} \cdot \mathbf{v}$ is also a zero vector. Now from the induction hypothesis, we know that

$T^{(n-1)}_{S_i,F_i}$ has full rank. Hence, $\mathbf{v}$ is a zero vector, which implies that for every $\mathbf{f} \in F_i$,

$$\mathbf{v_f} = \sum_{(\mathbf{f},k)\in\pi^{-1}(\mathbf{f})} \binom{k}{i-1} \mathbf{u}_{(\mathbf{f},k)} = 0.$$

□

Now we prove our main theorem using the transfer matrix equation.

Theorem 7.1.3 (restated). *Let $P(\mathbf{x})$ be an n-variate degree d polynomial over $\mathbb{F}^k$, and* $\mathrm{char}(\mathbb{F}) = 0$ *or $> d$. Let $\mathbf{w} = (w_1, \ldots, w_n) \in \mathbb{N}^n$ be a basis isolating weight assignment for $P(\mathbf{x})$. Then, $P(\mathbf{x} + t^{\mathbf{w}}) := P(x_1 + t^{w_1}, \ldots, x_n + t^{w_n})$ has a cone-closed basis over $\mathbb{F}(t)$.*

Proof. As we mentioned in Equation 7.1, the shifted polynomial $P(\mathbf{x} + t^{\mathbf{w}})$ yields a matrix equation $F' = D^{-1}T^{(n)}D \cdot F$. Let k' be the rank of F. We consider the following two cases.

Case 1 ($k' < k$): We reduce this case to the other one where $k' = k$. Let S be a subset of k' columns such that $F_{M,S}$ has rank k', where $M = M_{n,d}$. The matrix $F_{M,S}$ denotes the polynomial $P_S(\mathbf{x}) \in \mathbb{F}^{k'}[\mathbf{x}]$, where for every monomial m, $\mathrm{coef}_m(P_S)$ is the projection of $\mathrm{coef}_m(P)$ on the coordinates indexed by S. Therefore, any linear dependence relation among the coefficients of $P(\mathbf{x})$ is also valid for $P_S(\mathbf{x})$. This implies $\mathbf{w}$ is also a basis isolating weight assignment for $P_S(\mathbf{x})$. Now from our Case 2, there exists a set of monomials A such that A is a cone-closed basis of $P_S(\mathbf{x}+t^{\mathbf{w}})$. Thus, A is also a cone-closed basis of $P(\mathbf{x})$. This implies that $P(\mathbf{x} + t^{\mathbf{w}})$ has a cone-closed basis.

Case 2 ($k' = k$): Let B be the least basis of $P(\mathbf{x})$ with respect to $\mathbf{w}$, and $A = \textsc{Find-Cone-closed}(B, n)$. We prove that A is a basis for $P(\mathbf{x} + t^{\mathbf{w}})$. To prove this, we show that $\det(F'_{A,[k]}) \neq 0$. Define $T' := T^{(n)}DF$. Hence, $F' = D^{-1}T'$. Using Cauchy-Binet formula (Lemma 2.3.17), we get that

$$\det(F'_{A,[k]}) \;=\; \sum_{C\in\binom{M}{k}} \det(D^{-1}_{A,C}) \cdot \det(T'_{C,[k]})\,.$$

Since for all $C \in \binom{M}{k} \setminus \{A\}$, the determinant of $D^{-1}_{A,C}$ is zero, we have $\det(F'_{A,[k]}) =$

$\det(D^{-1}_{A,A}) \cdot \det(T'_{A,[k]})$. Again applying Cauchy-Binet formula for $\det(T'_{A,[k]})$, we get

$$\det(F'_{A,[k]}) \;=\; \det(D^{-1}_{A,A}) \cdot \sum_{C \in \binom{M}{k}} t^{\mathbf{w}(C)} \det(T^{(n)}_{A,C}) \cdot \det(F_{C,[k]})\,.$$

From Lemma 7.2.2, we have that for all basis $C \in \binom{M}{k} \setminus \{B\}$ of P, $\mathbf{w}(C) > \mathbf{w}(B)$. The determinant of $T^{(n)}_{A,B}$ is nonzero by Lemma 7.2.5, and the other one $F_{B,[k]}$ has nonzero determinant since B is a basis of P. Hence, the sum is a nonzero polynomial in t. In particular, $\det(F'_{A,[k]}) \neq 0$, which ensures that the coefficients of the monomials corresponding to A form a basis of $\mathrm{sp}_{\mathbb{F}(t)}(P(\mathbf{x} + t^{\mathbf{w}}))$. Since Lemma 7.2.4 says that A is also cone-closed, we get that $P(\mathbf{x} + t^{\mathbf{w}})$ has a cone-closed basis. $\square$

7.2.1 Models with a cone-closed basis

Consider the polynomials of form $D(\mathbf{x}) = (\mathbf{1} + \mathbf{a}_1 x_1 + \ldots + \mathbf{a}_n x_n)^d$ in $\mathbb{F}^k[\mathbf{x}]$, where $\mathbb{F}^k$ is seen as an $\mathbb{F}$-algebra with coordinate-wise multiplication. For any $\mathbf{c} \in \mathbb{F}^k$, $\mathbf{c}^{\intercal} \cdot D(\mathbf{x})$ becomes a polynomial (over $\mathbb{F}$) of form computed by a special type of depth-3 diagonal circuits, i.e., each linear polynomial is raised to the same power.

Lemma 7.2.7. *If* $\mathrm{char}(\mathbb{F}) = 0$ *or* $> d$*, then* $D(\mathbf{x})$ *has a cone-closed basis*

Proof. Consider the n-tuple $L := (\mathbf{a}_1, \ldots, \mathbf{a}_n)$. Then for every monomial $\mathbf{x}^{\mathbf{e}}$ of degree $\leq d$, the coefficient of $\mathbf{x}^{\mathbf{e}}$ in D is $L^{\mathbf{e}} := \prod_{i=1}^{n} \mathbf{a}_i^{e_i}$, with some nonzero scalar factor (note that here we need $\mathrm{char}(\mathbb{F})$ zero or large). We ignore this constant factor, since it does not affect linear dependence relations. Consider *deg-lex* monomial ordering, i.e., first order the monomials by lower to higher total degree, then within each degree arrange them according to lexicographic order. Now we prove that the 'least basis' of $D(\mathbf{x})$ with respect to this monomial ordering is cone-closed.

We incrementally devise a monomial set B as follows: Arrange all the monomials in ascending order. Starting from the least monomial, put a monomial in B if its coefficient cannot be written as a linear combination of its previous (thus, smaller) monomials. From construction, the coefficients of monomials in B form the least basis for the coefficient space of $D(\mathbf{x})$. Now we show that B is cone-closed. We prove it by contradiction.

Let $\mathbf{x}^{\mathbf{f}} \in B$ and let $\mathbf{x}^{\mathbf{e}}$ be its sub-monomial that is not in B. Then we can write

$$L^{\mathbf{e}} = \sum_{\mathbf{x}^{\mathbf{b}} \prec \mathbf{x}^{\mathbf{e}}} c_{\mathbf{b}} L^{\mathbf{b}},$$

where $c_{\mathbf{b}}$'s in $\mathbb{F}$ and $\prec$ denotes the deg-lex monomial ordering. Multiplying by $L^{\mathbf{f}-\mathbf{e}}$ on both sides, we get

$$L^{\mathbf{f}} = \sum_{\mathbf{x}^{\mathbf{b}} \prec \mathbf{x}^{\mathbf{e}}} c_{\mathbf{b}} L^{\mathbf{b}+\mathbf{f}-\mathbf{e}} = \sum_{\mathbf{x}^{\mathbf{b}'} \prec \mathbf{x}^{\mathbf{f}}} c'_{\mathbf{b}'} L^{\mathbf{b}'}.$$

Note that $\mathbf{x}^{\mathbf{b}'} \prec \mathbf{x}^{\mathbf{f}}$ holds true by the way a monomial ordering is defined. This equation contradicts the fact that $\mathbf{x}^{\mathbf{f}} \in B$, and completes the proof. $\square$

7.3 Discussion

In this chapter, we proved that after shifting a polynomial (over $\mathbb{F}^k$) by a basis isolating weight assignment, the coefficient space of the new polynomial contains a cone-closed basis. Basis isolating weight assignment is much weaker than the weight assignment induced by lexicographic monomial ordering. An interesting direction for future work can be to relate cone closed basis with other new kind of weight assignments (or, in general, polynomial map[1]) which are even "weaker" than the basis isolating weight assignment. Till now, no known blackbox PIT algorithm for ROABPs gives a polynomial time blackbox PIT algorithm for log-variate ROABPs . Hence, achieving cone-closed basis (in polynomial time) for log-variate ROABPs is also interesting, since it will also ensure a low-cone monomial in the support of the transformed polynomial. Then the technique we developed in Chapter 6 can be used to get a polynomial time blackbox PIT algorithm for log-variate ROABPs.

[1]In polynomial map, we map each variable to a univariate polynomial. It is more general than weight assignment.

Chapter 8

Conclusion

We studied the phenomenon of *bootstrapping* which describes how to convert a hitting set for $n(s)$-variate size s degree s circuits to a hitting set for s-variate size s degree s circuits where $n(s) \ll s$. We described the bootstrapping in the following cases.

1. Perfect Bootstrapping: $n(s) = \log^{\circ c} s$, and get a polynomial size hitting set in the conclusion.

2. Constant Bootstrapping: $n(s) =$ constant, and get a "slightly" super-polynomial size hitting set in the conclusion.

3. Shallow Bootstrapping: $n(s) =$ constant with depth-4 circuits in the hypothesis, and get a quasi-polynomial size hitting set in the conclusion.

A trivial derandomization of [DL78, Zip79, Sch80] gives an $(s+1)^n$-size hitting set for n-variate size s degree s circuits. Our goal is to design a poly(s)-size hitting set for s-variate size s degree s circuits. Our Constant Bootstrapping implies a powerful amplification of derandomization for PIT, i.e., for some constant $\delta < 1/2$, an s^{n^δ}-size hitting set for n-variate size s degree s circuits implies a "slightly" super-polynomial size hitting set for size s degree s circuits.

Subsequet to our results, [KST18] showed that an even an $s^{n-\epsilon}$-size hitting set, for some constant $\epsilon > 0$, in the hypothesis is sufficient to get the same conclusion. Their proof also works for the much weaker models than circuits like formulas and algebraic branching programs. It is further improved by [GKSS19]. There exists a "trivial" hitting set of size $(s+1)^n$ for n-variate polynomials of *individual degree* s and computable by size s circuits. They showed that even saving a single point over the trivial hitting set (i.e., hitting set of size $(s+1)^n - 1$) gives a polynomial-size hitting set in the conclusion. However, their result does not work for formulas or algebraic branching programs. As a subsequent development, one can try to extend [GKSS19] to these weaker models. Another shortcoming of their work is that it does not work over finite fields. Hence, it will be interesting to generalize [GKSS19] over all fields.

Another interesting direction would be to study the bootstrapping phenomenon in lower bound setting, i.e., converting a polynomial with "weak" circuit lower bound to a polynomial with "strong" circuit lower bound. Recently, [CILM18] showed such result for *non-commutative* algebraic circuits. They showed that proving mildly super-linear (for some $\epsilon > 0$, $n^{\frac{\omega}{2}+\epsilon}$ where ω is the exponent of the matrix multiplication) lower bounds for non-commutative algebraic circuits implies exponential lower bounds for non-commutative circuits. However, such analogous statement is not available in commutative setting. Therefore, it would be interesting to prove such a result for commutative circuits.

Our result bootstrapping in PIT motivates to study the circuits with "few" variables, for example, number of variables is logarithmic with respect to the circuit size. We gave a poly(s) time blackbox for $O(\log s)$-variate size s circuits computing polynomials with poly(s) dimensional partial derivative space. As the depth-3 diagonal circuit is a prominent circuit class which has polynomially large dimensional partial derivative space, our result gives the first polynomial-size hitting set for log-variate depth-3 diagonal circuits. Moreover, to design blackbox PIT for those models, we give an efficient blackbox PIT for polynomials whose support set contains a low-cone monomial. We saw that cone-size was a useful measure (for monomials) in the log-variate regime. Hence, the cone size measure needs to be further explored in other log-variate models like log-variate ROABPs, log-variate depth-3 semi-diagonal circuits (i.e., $\sum_{i=1}^{k} m_i \cdot \ell_i^{d_i}$ where m_i's are monomials and ℓ_i's are linear polynomials).

We studied polynomials over the vector space $\mathbb{F}^k$. We introduced the notion of cone-closed basis for polynomials over $\mathbb{F}^k$. It is a stronger notion of rank concentration compared to low-support rank concentration and low-cone rank concentration. We showed that if a polynomial over $\mathbb{F}^k$ is shifted by its basis isolating weight assignment, then the new polynomial becomes cone-closed. It is currently the best known rank concentration result for polynomials over vector spaces.

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, New York, NY, USA, 1st edition, 2009. 7, 29, 43, 62

[AGKS15] Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Hitting-sets for ROABP and sum of set-multilinear circuits. *SIAM Journal on Computing*, 44(3):669–697, 2015. 6, 7, 12, 13, 97, 99

[Agr05] Manindra Agrawal. Proving lower bounds via pseudo-random generators. In *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science, 25th International Conference, Hyderabad, India, December 15-18, 2005, Proceedings*, pages 92–105, 2005. 5, 26, 45

[AGS18] Manindra Agrawal, Sumanta Ghosh, and Nitin Saxena. Bootstrapping variables in algebraic circuits. pages 1166–1179, 2018. Article in Proceedings of the National Academy of Sciences of the USA (PNAS), 2019. xv, 14, 43, 59, 72, 75

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of mathematics*, pages 781–793, 2004. 5

[ASS13] Manindra Agrawal, Chandan Saha, and Nitin Saxena. Quasi-polynomial hitting-set for set-depth-Δ formulas. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 321–330, 2013. 6, 12, 13, 86

[ASSS12] Manindra Agrawal, Chandan Saha, Ramprasad Saptharishi, and Nitin 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. 7

[AV08] Manindra Agrawal and V. Vinay. Arithmetic circuits: A chasm at depth four. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 67–75, 2008. 6, 10, 31, 33, 75, 76, 78, 80, 83

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984. 7

[BMS11] Malte Beecken, Johannes Mittmann, and Nitin Saxena. Algebraic independence and blackbox identity testing. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II*, pages 137–148, 2011. 7

[BT88] Michael Ben-Or and Prasoon Tiwari. A deterministic algorithm for sparse multivariate polynominal interpolation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 301–309, 1988. 6

[Bür13] Peter Bürgisser. *Completeness and reduction in algebraic complexity theory*, volume 7. Springer Science & Business Media, 2013. 26, 29, 44

[CILM18] Marco L. Carmosino, Russell Impagliazzo, Shachar Lovett, and Ivan Mihajlin. Hardness amplification for non-commutative arithmetic circuits. In *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, pages 12:1–12:16, 2018. 110

[CKS18] Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. Hardness vs randomness for bounded depth arithmetic circuits. In *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, pages 13:1–13:17, 2018. 6

[CLO15] David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer Publishing Company, Incorporated, 4th edition, 2015. 21

[DdOS14] Zeev Dvir, Rafael Mendes de Oliveira, and Amir Shpilka. Testing Equivalence of Polynomials under Shifts. In *41st International Colloquium on Automata, Languages, and Programming, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 417–428, 2014. 5

[DF13] Rodney G Downey and Michael R Fellows. *Fundamentals of parameterized complexity*, volume 4. Springer, 2013. 56

[DL78] Richard A. Demillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193 – 195, 1978. 4, 5, 8, 24, 60, 109

[dOSV15] Rafael Mendes de Oliveira, Amir Shpilka, and Ben Lee Volk. Subexponential size hitting sets for bounded depth multilinear formulas. In *30th Conference on Computational Complexity, CCC 2015, June 17-19, 2015, Portland, Oregon, USA*, pages 304–322, 2015. 6

[DS07] Zeev Dvir and Amir Shpilka. Locally decodable codes with two queries and polynomial identity testing for depth 3 circuits. *SIAM J. Comput.*, 36(5):1404–1434, 2007. 6

[DSY09] Zeev Dvir, Amir Shpilka, and Amir Yehudayoff. Hardness-randomness tradeoffs for bounded depth arithmetic circuits. *SIAM J. Comput.*, 39(4):1279–1293, 2009. 6

[FGS13] Michael A. Forbes, Ankit Gupta, and Amir Shpilka. private communication, 2013. 36, 86

[FGS18] Michael A. Forbes, Sumanta Ghosh, and Nitin Saxena. Towards blackbox identity testing of log-variate circuits. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 54:1–54:16, 2018. xv, 85, 95

[FGT16] Stephen A. Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite perfect matching is in quasi-NC. In *48th Annual ACM Symposium on Theory of Computing*, pages 754–763, 2016. 5

[Fis94] Ismor Fischer. Sums of like powers of multivariate linear forms. *Mathematics Magazine*, 67(1):59–61, 1994. 37

[For14] Michael A. Forbes. *Polynomial Identity Testing of Read-Once Oblivious Algebraic Branching Programs.* PhD thesis, Massachusetts Institute of Technology, 2014. 7, 24, 38, 88

[For15] Michael A Forbes. Deterministic divisibility testing via shifted partial derivatives. In *56th Annual Symposium on Foundations of Computer Science*, pages 451–465, 2015. 7

[FS12] Michael A. Forbes and Amir Shpilka. On identity testing of tensors, low-rank recovery and compressed sensing. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 163–172, 2012. 86

[FS13a] Michael A Forbes and Amir Shpilka. Explicit noether normalization for simultaneous conjugation via polynomial identity testing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 527–542. Springer, 2013. 38, 86, 87, 88

[FS13b] Michael A. Forbes and Amir Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 243–252, 2013. 6, 7, 11

[FSS13] Michael A. Forbes, Ramprasad Saptharishi, and Amir Shpilka. Pseudorandomness for multilinear read-once algebraic branching programs, in any order. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:132, 2013. 14, 100, 101

[FSS14] Michael A. Forbes, Ramprasad Saptharishi, and Amir Shpilka. Hitting sets for multilinear read-once algebraic branching programs, in any order. In *Symposium on Theory of Computing (STOC), New York, NY, USA, May 31 - June 03, 2014*, pages 867–875, 2014. 6, 7, 11, 12, 13, 14, 86, 87, 100

[GKKS16] Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Arithmetic circuits: A chasm at depth 3. *SIAM J. Comput.*, 45(3):1064–1079, 2016. 6, 33, 35, 76, 80, 83, 86

[GKS17] Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Identity testing for constant-width, and any-order, read-once oblivious arithmetic branching programs. *Theory of Computing*, 13(2):1–21, 2017. 7, 12, 13, 14, 41

[GKSS19] Zeyu Guo, Mrinal Kumar, Ramprasad Saptharishi, and Noam Solomon. Derandomization from algebraic hardness. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:65, 2019. Preliminary version in FOCS 2019. 14, 57, 72, 74, 110

[GKST15] Rohit Gurjar, Arpita Korwar, Nitin Saxena, and Thomas Thierauf. Deterministic identity testing for sum of read-once oblivious arithmetic branching programs. In *30th Conference on Computational Complexity, CCC 2015, June 17-19, 2015, Portland, Oregon, USA*, pages 323–346, 2015. 7, 12, 13, 14, 100, 101

[GR98] Dima Grigoriev and Alexander A. Razborov. Exponential complexity lower bounds for depth 3 arithmetic circuits in algebras of functions over finite fields. In *39th Annual Symposium on Foundations of Computer Science, FOCS '98, November 8-11, 1998, Palo Alto, California, USA*, pages 269–278, 1998. 11

[GR08] Ariel Gabizon and Ran Raz. Deterministic extractors for affine sources over large fields. *Combinatorica*, 28(4):415–440, 2008. Preliminary version in 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2005. 93

[GT17] Rohit Gurjar and Thomas Thierauf. Linear matroid intersection is in quasi-nc. In *49th Annual ACM Symposium on Theory of Computing*, pages 821–830, 2017. 5

[GTV18] Rohit Gurjar, Thomas Thierauf, and Nisheeth K. Vishnoi. Isolating a vertex via lattices: Polytopes with totally unimodular faces. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 74:1–74:14, 2018. 5

[Gur15] Rohit Gurjar. *Derandomizing PIT for ROABP and Isolation Lemma for Special Graphs.* PhD thesis, Indian Institute of Technology Kanpur, 2015. 7

[GV16] Ira M. Gessel and X. G. Viennot. Determinants, paths, and plane partitions. 2016. 42

[Her06] Israel N Herstein. *Topics in algebra.* John Wiley & Sons, 2006. 18, 19

[HS80] Joos Heintz and Claus-Peter Schnorr. Testing polynomials which are easy to compute (extended abstract). In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA*, pages 262–272, 1980. 5, 26, 44

[IW97] Russell Impagliazzo and Avi Wigderson. P = bpp if e requires exponential circuits: Derandomizing the xor lemma. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 220–229, New York, NY, USA, 1997. ACM. 7, 44

[Kal89] Erich Kaltofen. Factorization of polynomials given by straight-line programs. *Advances in Computing Research*, 5:375–412, 1989. 26

[Kay10] Neeraj Kayal. Algorithms for arithmetic circuits. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:73, 2010. 11, 86

[KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004. Preliminary version in the 35th Annual ACM Symposium on Theory of Computing (STOC), 2003. 5, 6, 29, 31, 45, 47, 82

[Koi12] Pascal Koiran. Arithmetic circuits: The chasm at depth four gets wider. *Theoretical Computer Science*, 448:56–65, 2012. 33, 75

[KP09] Pascal Koiran and Sylvain Perifel. VPSPACE and a transfer theorem over the reals. *Computational Complexity*, 18(4):551–575, 2009. 28, 29

[KS01] Adam R. Klivans and Daniel A. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 216–223, 2001. 6

[KS07] Neeraj Kayal and Nitin Saxena. Polynomial identity testing for depth 3 circuits. *Computational Complexity*, 16(2):115–138, 2007. 6

[KS09] Neeraj Kayal and Shubhangi Saraf. Blackbox polynomial identity testing for depth 3 circuits. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 198–207, 2009. 6

[KS11] Zohar Shay Karnin and Amir Shpilka. Black box polynomial identity testing of generalized depth-3 arithmetic circuits with bounded top fan-in. *Combinatorica*, 31(3):333–364, 2011. 6

[KS17] Mrinal Kumar and Shubhangi Saraf. Arithmetic circuits with locally low algebraic rank. *Theory of Computing*, 13(1):1–33, 2017. Preliminary version in the 31st Conference on Computational Complexity (CCC), 2016. 7

[KSS14] Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. Equivalence of polynomial identity testing and deterministic multivariate polynomial factorization. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*, pages 169–180, 2014. 5

[KST18] Mrinal Kumar, Ramprasad Saptharishi, and Anamay Tengse. Near-optimal bootstrapping of hitting sets for algebraic models. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:132, 2018. Preliminary version in the Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2019. 14, 72, 74, 109

[Lov79] László Lovász. On determinants, matchings, and random algorithms. In *FCT*, volume 79, pages 565–574, 1979. 5

[Mul12] Ketan D. Mulmuley. Geometric complexity theory V: Equivalence between blackbox derandomization of polynomial identity testing and derandomization of Noether's normalization lemma. In *FOCS*, pages 629–638, 2012. 86

[Mul17] Ketan Mulmuley. Geometric complexity theory V: Efficient algorithms for Noether normalization. *Journal of the American Mathematical Society*, 30(1):225–309, 2017. 5, 86

[MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 345–354, 1987. 5

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994. 7, 29, 44

[NW97] Noam Nisan and Avi Wigderson. Lower bounds on arithmetic circuits via partial derivatives. *Computational Complexity*, 6(3):217–234, 1997. Preliminary version in the 36th Annual Symposium on Foundations of Computer Science (FOCS), 1995. 10, 86

[PSS18] Anurag Pandey, Nitin Saxena, and Amit Sinhababu. Algebraic independence over positive characteristic: New criterion and applications to locally low-algebraic-rank circuits. *Computational Complexity*, 27(4):617–670, 2018. Preliminary version in the 41st International Symposium on Mathematical Foundations of Computer Science (MFCS), 2016. 7

[Raz09] Ran Raz. Multi-linear formulas for permanent and determinant are of super-polynomial size. *J. ACM*, 56(2):8:1–8:17, 2009. 11

[RS05] Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005. 7, 12

[RY09] Ran Raz and Amir Yehudayoff. Lower bounds and separations for constant depth multilinear circuits. *Computational Complexity*, 18(2):171–207, 2009. 11

[Rys63] H.J. Ryser. *Combinatorial Mathematics*. Carus mathematical monographs. Mathematical Association of America, 1963. 37

[Sap13a] Ramprasad Saptharishi. personal communication, 2013. 91

[Sap13b] Ramprasad Saptharishi. *Unified Approaches to Polynomial Identity Testing and Lower Bounds*. PhD thesis, Chennai Mathematical Institute, 2013. 7, 86

[Sap16] Ramprasad Saptharishi. A survey of lower bounds in arithmetic circuit complexity. Technical report, https://github.com/dasarpmar/lowerbounds-survey/, 2016. 32

[Sax08] Nitin Saxena. Diagonal circuit identity testing and lower bounds. In *ICALP*, volume 5125 of *Lecture Notes in Computer Science*, pages 60–71. Springer, 2008. 6, 36

[Sax09] Nitin Saxena. Progress on polynomial identity testing. *Bulletin of the EATCS*, 99:49–79, 2009. 7, 86

[Sax13] Nitin Saxena. Progress on polynomial identity testing - II. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:186, 2013. 7

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, October 1980. 4, 5, 8, 24, 60, 109

[Sha90] Adi Shamir. Ip=pspace. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 11–15, 1990. 5

[SS11] Nitin Saxena and C. Seshadhri. An almost optimal rank bound for depth-3 identities. *SIAM J. Comput.*, 40(1):200–224, 2011. 6

[SS12] Nitin Saxena and C. Seshadhri. Blackbox identity testing for bounded top-fanin depth-3 circuits: The field doesn't matter. *SIAM Journal on Computing*, 41(5):1285–1298, 2012. 6

[SS13] Nitin Saxena and C. Seshadhri. From sylvester-gallai configurations to rank bounds: Improved blackbox identity test for depth-3 circuits. *J. ACM*, 60(5):33:1–33:33, 2013. 6

[SSS13] Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena. A case of depth-3 identity testing, sparse factorization and duality. *Computational Complexity*, 22(1):39–69, 2013. 6

[ST17] Ola Svensson and Jakub Tarnawski. The matching problem in general graphs is in quasi-nc. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 696–707, 2017. 5

[Str73] Volker Strassen. Vermeidung von divisionen. *Journal fr die reine und angewandte Mathematik*, 264:184–202, 1973. 35

[STV01] Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001. 7

[SV11] Shubhangi Saraf and Ilya Volkovich. Black-box identity testing of depth-4 multilinear circuits. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 421–430, 2011. 7

[SW01] Amir Shpilka and Avi Wigderson. Depth-3 arithmetic circuits over fields of characteristic zero. *Computational Complexity*, 10(1):1–27, 2001. 11

[SY10] Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010. 7

[Tav13] Sébastien Tavenas. Improved bounds for reduction to depth 4 and depth 3. In *Mathematical Foundations of Computer Science 2013 - 38th International Symposium, MFCS 2013, Klosterneuburg, Austria, August 26-30, 2013. Proceedings*, pages 813–824, 2013. 33, 75

[Tut47] W. T. Tutte. The factorization of linear graphs. *Journal of the London Mathematical Society*, s1-22(2):107–111, 1947. 5

[Uma03] Christopher Umans. Pseudo-random generators for all hardnesses. *J. Comput. Syst. Sci.*, 67(2):419–440, 2003. 7

[Val79] Leslie G. Valiant. Completeness classes in algebra. In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 249–261, 1979. 22, 29

[VSBR83] Leslie G. Valiant, Sven Skyum, S. Berkowitz, and Charles Rackoff. Fast parallel computation of polynomials using few processors. *SIAM J. Comput.*, 12(4):641–644, 1983. 32, 75

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91, 1982. 7

[Zen93] Jiang Zeng. A bijective proof of Muir's identity and the Cauchy-Binet formula. *Linear Algebra and its Applications*, 184:79–82, 1993. 41

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, EUROSAM '79, pages 216–226, 1979. 4, 5, 8, 24, 60, 109

Index