Hitting Sets for Algebraic Models

Constructions and Consequences

A Thesis

submitted to the

Tata Institute of Fundamental Research, Mumbai

for the degree of Doctor of Philosophy in

Computer Science

by

Anamay Gurunath Tengse

Tata Institute of Fundamental Research, Mumbai

June, 2021

DECLARATION

This thesis, titled **"Hitting Sets for Algebraic Models: Constructions and Consequences"** is a presentation of my original research work. Whatever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions.

The work was done under the guidance of **Dr. Ramprasad Saptharishi** at the Tata Institute of Fundamental Research, Mumbai.

Anamay G. Tengse

In my capacity as supervisor of the candidate's thesis, I certify that the above statements are true to the best of my knowledge.

Ramprasad Saptharishi

Date:

Acknowledgements

While I was writing this thesis, everyone in the world was struggling with the COVID19 pandemic in some way or the other. I am deeply grateful to the doctors, nurses, and all essential workers, who faced the worst of this crisis to keep the rest of us safe and healthy. A million thanks to each one of them.

I want to start by thanking all the people unknown to me, whose work makes the various services and facilities accessible to students like me, that I have used for carrying out my work, academic and otherwise. I also want to thank all the people who have contributed in recording, compiling, maintaining and broadcasting, all the scientific and mathematical literature that has helped my research till now.

I have been fortunate to have enjoyed tremendous support, inspiration and knowledge from all the wonderful people who have graced me with their friendship. Without the friends I made at TIFR, my time here would have been quite dull and dark. There are numerous conversations which I recall having with them, that have helped me correct my course, in research and in life. At the same time, had it not been for the solid backing that I received from my friends outside TIFR, I would not have even dreamed to reach a place like TIFR! I owe a lot of what I have to all these amazing people.

My love for research and teaching is a large part of why I chose to pursue a PhD. All of that comes from studying from some of the most dedicated and skilled teachers who have taught me; I shall always aim to be as earnest as them in whatever I do. I can never forget the guidance and support I received from all the teachers from S. S. Samiti's school at Kavale, who helped me restore my confidence in my abilities. After graduating from Kavale, my interactions with the members of faculty at the Goa College of Engineering, and IIT Bombay were instrumental in leading me to pursue a PhD.

I had joined TIFR with little understanding of research in theoretical CS; a significant portion of what I now know has come from my time here. The quality of teaching and exposition at STCS-TIFR is something that I have not seen anywhere else. I hope that one day I would able to express ideas like some of the faculty here. Even beyond the fabulous lectures, the numerous formal and informal interactions with all the people in STCS and TIFR in general, have been extremely pleasant and educational. I will surely keep coming back to TIFR and STCS whenever possible in search of more such interactions.

I have been fortunate to receive useful advice from many experts in the area. In particular, I have had the good fortune of working closely with Mrinal, and on many occasions, I have

gained several valuable insights from him about mathematics, research and more, that have greatly enhanced my understanding.

It has also been my privilege to have some amazing people in my family and social circles. I always found help whenever I needed it, just like the reassuring call with my cousin Prasad (Paresh) that led me to choose computer engineering to begin with. I spent almost all of my childhood being the beloved younger brother of Vaishnavi, Aparna and Rekha. I doubt anything else will ever give me that much joy[1].

Having Ramprasad as my PhD advisor is perhaps the best thing that has happened to me in my academic life. After my first year at TIFR, I had thought that algebra was "not for me", and had asked Jaikumar about how I should move forward. In addition to his timeless and sound advice that I should *'keep an open mind'*, the key reason that I even dared to explore *algebraic complexity*, was my semester-long experience of working with Ramprasad. I have always felt very comfortable sharing my ideas with him, even though they have turned out to be crazy or stupid most of the times. His never-ending support has greatly helped me manage even the most testing of times, just like the day of my thesis defence! I hope that my work lives up to the efforts that he has put in, and I truly wish that every PhD student gets an advisor like him.

Prerona is perhaps the simplest and the most honest person that I have ever known, and I have been extremely fortunate to have met her. The time I spent with her (and other "cool people") at TIFR was the reason I always felt at home here, during all these years. Just talking to her has made everything easy to deal with and get through; I hope to enjoy this privilege in the years to come.

In addition to all of the above, what gives me a clear advantage over almost everyone else, is being born to my parents. My father's love for mathematics was what introduced me to solving puzzles at a very young age, and led me to like computer programming and eventually computer science. My mother's belief in me (which is way more than my own), has given me the energy to keep going through all the inconveniences, big and small. I have happily inherited a strong affinity for music, literature and art from them, which has greatly enriched my life. My father's principled approach to life and my mother's kindness continue to inspire me every day, to become a better version of myself, in every aspect. Quite literally, I would not have reached anywhere without their love and support, and it is to them that I dedicate my thesis.

[1]Other than proving cool theorems, of course. :-)

To Aai and Baba,

whose love and support makes every goal look achievable.

Contents

1 | Introduction

A fundamental challenge in theoretical computer science is to understand *what makes some computational tasks harder than others*. Formalizing and trying to answer this natural and seemingly simple question has led to the vast area of research that we now call *complexity theory*: the area that aims to determine the amount of resources like time, space and randomness, required by various computational tasks.

Within complexity theory, if we restrict ourselves to the tasks that can be performed using the basic *arithmetic* operations of addition and multiplication, we obtain the area of *algebraic complexity theory*, which analyses the complexity of computing multivariate polynomials. Several well known computational tasks can be naturally viewed as polynomials. For instance, almost all the tasks concerning matrices like matrix multiplication, computing determinants or adjoints etc., can be viewed as polynomials in the entries of the underlying matrices.

Perhaps the most natural way to judge the complexity of a polynomial is to pin down the number of basic operations required to compute it on any given input. The models of *algebraic circuits* and *algebraic formulas* formalise this intuition: they start with formal variables and constants, and inductively build the required polynomial using additions and multiplications. Here are the formal definitions of these models, followed by some examples in Figure 1.1.

Definition 1.1 (Algebraic circuit)**.** *An* algebraic circuit *is a directed acyclic graph (DAG), with the* leaves *(nodes without incoming edges) being labelled by variables and field constants, and the* gates *(nodes with incoming edges) being labelled by addition (+) and multiplication (×). The* unique *node without any outgoing edges is called as the* output *node of the circuit.*

An algebraic circuit computes a polynomial in the natural way. That is, the addition gates compute the sum[1] *of the polynomials computed by their children and the multiplication gates multiply the polynomials computed by their children. The circuit is said to compute the polynomial computed by its output node.*

The size *of a circuit is the number of nodes in the graph, and the* depth *is the length of the longest path from the output gate to a leaf.* ◊

Definition 1.2 (Algebraic formula)**.** *An* algebraic formula *is an algebraic circuit whose underlying DAG is a tree.*

The size *of a formula is the total number of* leaves *in the underlying tree, and the* depth *is the*

[1]We shall also allow field constants on the incoming edges of addition gates (assumed to be 1 by default), in which case these gates compute the corresponding linear combination of the input polynomials.

length of the longest path from the output gate to a leaf. ◊

The complexity of a certain polynomial f is then defined as the *size of the smallest circuit computing f*. Note that the complexity of f as defined above closely resembles the fewest number of operations required to compute f on any input, just as we wanted.

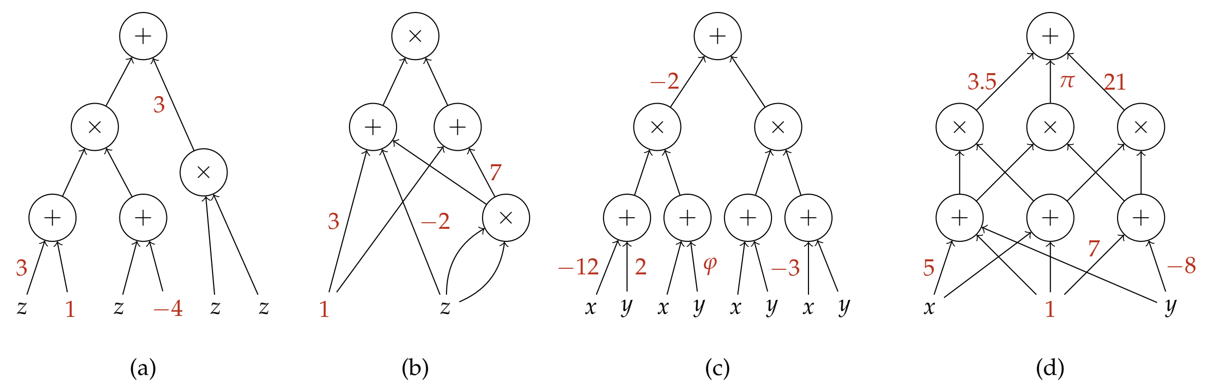


Figure 1.1: Algebraic Circuits and Formulas

The examples (a) and (c) in Figure 1.1 are formulas, and hence they “do not reuse computation”. That is, each node in these examples has at most one outgoing edge. Therefore formulas are believed to be a *strictly* weaker model than circuits: there are polynomials that are efficiently computable by circuits, but not by formulas. However, proving such a result remains a major open problem.

1.1 Easy and Hard Polynomials

With a natural measure for the complexity of polynomials at hand, the natural next step is to establish a good definition for “easy” and “hard” polynomials. For this, we shall turn to the known literature. The seminal work of Valiant [Val79] is widely regarded as the first step (although some research on the complexity of algebraic computation already existed [Str69, Str73, Hya77]) in formalizing the notions of easy and hard polynomials. We will therefore use the relevant definitions from Valiant’s work, as is commonly done.

For a fair comparison between different polynomials, we express their complexity (size) as a function of their basic parameters, the number of variables n and the degree d. This naturally leads us to the notion of *polynomial families*, which is a fairly intuitive concept. Let us consider the determinant polynomial, which for an $n \times n$ matrix is a polynomial of degree n on its n^2 entries. The determinant polynomial thus has a fixed meaning for every positive integer n, and combining all the $n \times n$ determinants gives us the *determinant family*, which we denote by $\{\mathrm{Det}_n\}$.

We can similarly extend almost all of the known polynomials to polynomial families. Most of the natural questions in algebraic complexity theory focus on polynomial families where the degree of the n^{th} polynomial is $\mathrm{poly}(n)$. A good reason for this is that we want to be able to *efficiently* evaluate the polynomials on certain inputs, and a super-polynomial degree might lead to super-polynomially long outputs even on small inputs [2]. We are now ready to define the class of easy polynomial families, which we now call VP (Valiant's P).

Definition 1.3 (VP: Efficiently computable polynomials). *A family of polynomials $\{f_n\}$ of degree $d(n) = \mathrm{poly}(n)$ is said to be in* VP *if f_n can be computed by a circuit of size $s(n) = \mathrm{poly}(n)$ for all large enough n.* $\diamondsuit$

This definition of easy polynomials has an important consequence: almost all polynomials are hard to compute. This can be informally argued as follows. An n-variate degree-d polynomial potentially has about $d^{O(n)}$ monomials when $d >= n$, which is exponential in n; whereas an algebraic circuit of size $\mathrm{poly}(n)$ only has $\mathrm{poly}(n)$ "parameters". Therefore for any large enough value of n, the set of all easy polynomials should form a tiny subset of all polynomials. This intuitive argument can in fact be formalised using a *dimension counting argument*, giving us that a "random polynomial" is hard to compute with very high probability. A natural question is then, do we "know" examples of polynomials that are hard to compute?

Note that a "random polynomial" will most likely have an exponentially long description, where we basically just list out all its coefficients. This goes against most intuitive definitions of a "known" polynomial. We should therefore try to find an *explicit* polynomial that is hard to compute; but how do we decide if a polynomial is "explicit enough"? Ideally, we would like to fix a definition that is also in some sense defined using circuits, like VP.

Certainly, a polynomial whose coefficients can be computed efficiently should be "explicit" under our definition. Fortunately, Valiant [Val79] has already defined a class of *efficiently definable polynomials*, which we now call VNP (Valiant's NP). The formal definition is slightly more involved, and we will defer it to Chapter 2. As we shall see later, the class VNP can indeed be defined using circuits, and also includes most polynomials we would like to call *explicit*, due to the following condition.

Fact 1.4 (Consequence of Valiant's criterion [Val82] (Informal)). *Suppose $f(\mathbf{x})$ is an n-variate polynomial of degree $\mathrm{poly}(n)$ whose coefficients are computable in time $\mathrm{poly}(n)$; then $f \in$* VNP.

We are now ready to define the first fundamental question in *algebraic circuit complexity*, that of *finding explicit hard polynomials*. In its full generality, this task actually forms a class of questions based on which algebraic model we work with, and the hardness (or easiness) parameter that we fix. Such questions are called *hardness questions* or *lower bound questions*, as they require us to prove that the complexity of a polynomial is *bounded by some function from below*. Of course, when we work with VP being the class of easy polynomials we get the following question, analogous to the famous P vs NP question.

Question 1.5 (Circuit lower bounds). *Is there a family of* explicit, hard *polynomials $\{h_n\}$?*
In other words, is VP = VNP*?*

[2]See Grochow's answer on `stackexchange.com` [Gro13] or Forbes' thesis [For14] for a more detailed discussion.

1.2 Polynomial Identity Testing

A typical *lower bound* statement looks like *"Any circuit computing f requires size $> s$"*. While this is certainly a statement about f, it is also a statement about the *set of all circuits of size* $\leq s$, in that they cannot compute f. We now introduce another fundamental question about classes of circuits which, among other things, also has intimate connections to the lower bound question.

Consider the following *algorithmic* task about a class of circuits. Given a circuit C of size $\leq s$, determine if it computes the zero polynomial: $C(\mathbf{a}) = 0$ for all inputs $\mathbf{a}$. This task is called *polynomial identity testing (PIT)* [3]. The term "identity testing" is due to the fact that we are essentially trying to verify whether the circuit is a wiring of some (non-trivial) algebraic identity like $(a+b)(a-b)+b^2-a^2$, or $(x+y)^2-x^2-y^2$ over characteristic 2.

In our search of explicit, hard low-degree polynomials, the class of n-variate circuits of size and degree $\mathrm{poly}(n)$ automatically becomes interesting. In fact, since we are interested in syntactic computation, this class is exactly the class VP. So we can just ask if there is a $\mathrm{poly}(n)$-time PIT-algorithm for n-variate circuits within VP, for every n? However, it also makes sense to study this question outside the context of VP, which we will state as follows.

Question 1.6 (Polynomial Identity Testing (PIT)). *Is there a* $\mathrm{poly}(n,d,s)$*-time algorithm, which when given as input any n-variate, degree d, size s circuit C, determines if $C \equiv 0$?*

This question is indeed simpler than Question 1.5, and has a pretty straightforward solution using randomness, thanks to the following lemma which has been independently proven multiple times in different guises[4].

Lemma 1.7 (Polynomial Identity Lemma [Ore22, DL78, Zip79, Sch80]). *Suppose $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ is an n-variate polynomial of degree d, and let $S \subseteq \mathbb{F}$ be a finite set of size strictly larger than d. Then $f(\mathbf{a}) \neq 0$ for at least a $\left(1-\frac{d}{|S|}\right)$ fraction of $\mathbf{a}$'s from S^n.*

Thus, if we take a set S of $1000d$ distinct points and evaluate the given circuit on a random point from S^n, then we will succeed almost every time. A subtle issue here is that the output of the given circuit could have exponential bit complexity, even with respect to the bit-complexities of the constants in the circuit[5]. We shall therefore just assume a computational model where all operations in the underlying field are unit cost. As a consequence of Lemma 1.7, the interesting question in the identity-testing-world is to obtain an *efficient, deterministic* algorithm (note that both the properties are important).

Question 1.8 (Deterministic PIT). *Is there a* $\mathrm{poly}(n,d,s)$*-time* deterministic *algorithm, which when given as input any n-variate, degree d, size s circuit C, determines if $C \equiv 0$?*

[3]Some texts use other names like: arithmetic circuit identity testing (ACIT), polynomial/circuit zero testing, but polynomial identity testing (PIT) seems to be the most widely used name.

[4]See for example [BCPS18] for a detailed survey of such results.

[5]We can get around this by executing the circuit gate-by-gate modulo a *uniformly* random prime of $\mathrm{poly}(s,n,d)$ bits. We skip the details, as that discussion is slightly off-topic.

1.2.1 Hitting Sets

An interesting characteristic of the above-mentioned randomized algorithm is that it only needs *evaluation-access* to the circuit being tested, and in particular does not need to "look inside" the circuit. As a result, such tests are called *blackbox polynomial identity tests (blackbox PITs)* [6]. Correspondingly, the tests that depend on the underlying graph of the circuit being tested, are called *whitebox PITs*. Therefore, Question 1.8 can be seen as the *whitebox PIT* question.

Let us take a closer look at the blackbox variant of Question 1.8: given a low-degree, efficiently computable polynomial, we evaluate it on some points to find out if it is zero everywhere. It is easy to show that *adaptive querying* does not improve the *worst case behaviour* of such an algorithm. This means that we could just collect all the potential evaluation points that our algorithm could ever use, and use the entire set for every input. Such a set is called a *hitting set*. Formally, a set of points $\mathcal{H}$ is a *hitting set* for a class of polynomials $\mathcal{C}$, if for every nonzero $f \in \mathcal{C}$ we have that $f(h) \neq 0$ on some point $h \in \mathcal{H}$. The blackbox variant of Question 1.8 then becomes the following question about designing a set of points, which is much more specific than designing an algorithm.

Question 1.9 (Small hitting sets)**.** *Does there exist a set $\mathcal{H}$ of* $\text{poly}(n, d, s)$ *points, such that for any nonzero n-variate, degree d, size s circuit C, there exists a point $h \in \mathcal{H}$ such that $C(h) \neq 0$?*

Interestingly, the answer to Question 1.9 is '*Yes*'; we already know that such a set of points *exists* (details in Chapter 2). However, that does not necessarily mean that we have an *efficient algorithm* for blackbox PIT. For that, we have to additionally insist that an algorithm should be able to *generate* the set $\mathcal{H}$ efficiently. We will call a set *explicit* if it can be generated efficiently, as just described.

Question 1.10 (Blackbox PIT: explicit hitting sets)**.** *Is there an* explicit *set $\mathcal{H}$ of* $\text{poly}(n, d, s)$ *points, such that for any* nonzero *n-variate, degree d, size s circuit C, there is a point $h \in \mathcal{H}$ such that $C(h) \neq 0$?*

We shall now spend some more time on the two central questions in algebraic complexity, Question 1.5 and Question 1.10, before moving on to the contributions of this thesis. In the next section, we will first see the highlights of research on both these fronts and then briefly explore some fascinating connections between the two seemingly unrelated questions.

1.3 Background

1.3.1 Determinants and Permanents

We saw that finding whether explicit hard polynomials (Question 1.5) exist is essentially equivalent to studying the classes VP and VNP. When dealing with complexity classes, it

[6] We will abbreviate both 'polynomial identity testing' and 'polynomial identity test' as PIT, whenever the intended meaning is clear from the context.

often helps to have objects that are good representatives for the strengths and limitations of these classes. For us, these representatives will be what are called *complete polynomials*. Intuitively, a complete polynomial could be seen as "the hardest polynomial" in that class. We will now see two polynomials, one that is complete for VNP and the other which is complete for VBP, a (possibly smaller) subclass of VP. These will therefore be our representatives for easy and explicit (and possibly hard) polynomials.

We shall now begin by describing the class VBP, which comes from the model of *algebraic branching programs*.

Definition 1.11 (Algebraic Branching Programs (ABPs)). *An* algebraic branching program *is a layered, directed graph. There are two special vertices,* source *(labelled s) and* sink/target *(labelled t), which are the unique vertices in the first (leftmost) and the last (rightmost) layers, respectively. All the edges in the graph go from left to right, and are between vertices in two* distinct *layers. Each edge is labelled by a* linear *polynomial in the underlying variables over the underlying field.*

Each path from s to t is said to compute the product *of the labels on its edges, and the ABP computes the* sum *of all the paths from s to t.*

Alternatively, an ABP can be seen as a product of several matrices, with each matrix having linear polynomials as its entries, and the ABP computing the $(1,1)$*th entry of the matrix product.*

The number of layers (or the number of matrices) is called the length *of the ABP and the maximum number of vertices in a single layer (or the larger dimension of the biggest matrix) is called its* width. *The* size *of the ABP is just the total number of vertices (sum of the dimensions of all the matrices).* $\Diamond$

Definition 1.12 (VBP (Informal)). *An n-variate polynomial f is said to be in VBP, if it has an ABP of size* $\mathrm{poly}(n)$. $\Diamond$

The computational power of ABPs lies between that of formulas and circuits. The fact that a formula can be efficiently simulated by an ABP is easier to see using the layered graph view, while the matrix multiplication view almost immediately tells us that circuits can efficiently simulate ABPs. While it is believed that there are super-polynomial separations between formulas, ABPs and circuits, none of these separations have been proven yet.

Definition of completeness. We first need to decide what it means for a polynomial f to be able to express another polynomial g efficiently. We want our definition to yield an efficient representation for g, whenever we have an efficient representation of f, and ideally this should be independent of what model we choose for these representations. It therefore makes sense to use one of the simplest ways of going from one polynomial to the other. The following definition is a good way of achieving this.

Definition 1.13 (Polynomial projection). *A polynomial* $f(\mathbf{x}_n)$ *is said to be a projection of the polynomial* $g(\mathbf{y}_m)$*, if there exist* linear *polynomials* $\ell_1(\mathbf{x}), \ell_2(\mathbf{x}), \ldots, \ell_m(\mathbf{x})$ *such that the following holds.* $f(\mathbf{x}) = g(\ell_1(\mathbf{x}), \ell_2(\mathbf{x}), \ldots, \ell_m(\mathbf{x}))$. $\Diamond$

Important complete polynomials. While there *are* complete polynomials known for the class VP, the class VBP has a well-known polynomial that is complete for it.

Theorem 1.14 ([Val79]). *Let $f(\mathbf{x})$ be a polynomial that is computable by an ABP of size s. Then there is an $m = poly(s)$ such that $f(\mathbf{x})$ is a projection of Det_m, the $m \times m$ symbolic determinant.*

Note that Theorem 1.14 only says that the symbolic determinant can efficiently simulate any ABP. The other part of the completeness result is implied by the following important result of Mahajan and Vinay [MV97].

Theorem 1.15 ([MV97]). *For all large enough $n \in \mathbb{N}$, the $n \times n$ symbolic determinant Det_n is computable by an ABP of size $\mathrm{poly}(n)$. In other words, the family $\mathsf{Det}_n \in$ VBP.*

For the class VNP, a close cousin of the determinant polynomial called the *permanent* polynomial was shown to be complete by Valiant in his foundational work [Val79]. An easy way to obtain the nth permanent polynomial Perm_n is to change all the -1 coefficients in Det_n to $+1$. We defer the formal definitions of Det_n and Perm_n to Chapter 2.

Theorem 1.16 (Permanent is VNP complete [Val79] (Informal)). *Let $f(\mathbf{x}_n)$ be a polynomial in* VNP. *Then there is an $m = \mathrm{poly}(n)$ such that $f(\mathbf{x})$ can be written as a projection of Perm_m.*

These completeness results for the determinant and the permanent give us a (weak) reformulation for Question 1.5.

Question 1.17 (Determinant vs Permanent). *Does there exist a function $m(n) = \mathrm{poly}(n)$ such that for all large enough $n \in \mathbb{N}$, we have that Perm_n is a projection of Det_m?*

1.3.2 Depth Reduction Results

It is easy to see that to compute a monomial of degree d, a formula of depth $O(\log d)$ suffices, even if we restrict all product gates to have at most 2 inputs. Furthermore, this restriction on the depth does not impact the size of our formula by much. We shall now see that very similar statements are true for algebraic formulas and even circuits. Furthermore, unlike their boolean counterparts, even constant depth algebraic formulas seem to have non-trivial computational power.

Efficient Depth Reduction

Algebraic formulas. Intuitively, since the graphs corresponding to formulas are just trees, it is perhaps not too ambitious to expect that we can "balance" the tree so that its depth is logarithmic in its size. This indeed turns out to be true, formalised by the following result.

Theorem 1.18 (Depth reduction for formulas [Bre74]). *Let $f(\mathbf{x})$ be a polynomial that is computable by a formula Φ of size s. Then, there exists a formula Φ' of size $\mathrm{poly}(s)$ and depth $O(\log s)$ which also computes $f(\mathbf{x})$.*

Algebraic circuits. While we intuitively expect to reduce the depth of a formula by "balancing the underlying tree", it is not at all clear that circuits should also have such a property. For instance, we do not know if it is possible to reduce *boolean circuits* to logarithmic depth. In fact it is believed that boolean circuits of logarithmic depth (NC^1) are not as powerful as polynomial sized boolean circuits ($\mathsf{P}/\operatorname{poly}$). In the algebraic world however, the following remarkable result of Valiant, Skyum, Berkowitz and Rackoff [VSBR83], and a strengthening of it due to Allender, Jiao, Mahajan and Vinay [AJMV98], show us that circuits *can* in fact be "balanced".

Theorem 1.19 (Efficient depth reduction [VSBR83, AJMV98])**.** *Suppose $f(\mathbf{x})$ is a polynomial of degree d that has a circuit of size s. Then $f(\mathbf{x})$ also has a circuit of depth $O(\log d)$, with each multiplication gate having at most 2 children, of size* $\operatorname{poly}(s)$*.*

Some useful properties of the proof of the above result from [AJMV98], are worth mentioning. Firstly, the proof is *constructive*, in that it also gives an algorithm to convert a given algebraic circuit into one with a smaller depth, computing the same polynomial. Secondly, this algorithmic procedure largely involves "rewiring" the original circuit, which means that the procedure preserves a wide variety of attributes of the original circuit. In fact, as we will later see in Chapter 3, one of the results in this thesis crucially depends on the second property.

Note that both the above results maintain the efficiency of the representation they begin with and yet allow us to reduce the depth quite significantly. So can we gain a bit more if we are willing to let the size increase even further, as long as it is less than some trivial bound?

Depth Reduction to Constant Depth

Representing a polynomial as sum of its monomials ($f(\mathbf{x}) = \sum_m f_m \cdot m$) trivially gives us an algebraic circuit/formula for it of *depth* 2. However, here the size is equal to the number of monomials that appear in f, which could be as large as $d^{O(n)}$. Thus $d^{O(n)}$ is a trivial upper bound on the size of any polynomial; and we expect an easy polynomial to have smaller and smaller circuits/formulas as we allow the depth to grow. An interesting question here is to know how quickly the size drops as the depth grows.

Once again, algebraic circuits admit depth reduction to a significant extent, as we shall now see. The following result due to Agrawal and Vinay [AV08], which was later simplified and strengthened by Koiran [Koi12] and Tavenas [Tav15], tells us that even *constant depth formulas* have non-trivial expressive power.

Theorem 1.20 (Depth reduction to depth 4 [AV08, Koi12, Tav15] (Informal))**.** *Suppose f is a degree d polynomial that is computable by a circuit of size s. Then f can also be computed by a depth 4 formula of size $s^{O(\sqrt{d})}$.*

Furthermore, all the multiplication gates in the formula have $O(\sqrt{d})$ children.

Remark. *The following more general version of the theorem can also be proven using similar techniques.*

Suppose f is a degree d polynomial that is computable by a circuit of size s. Then for any constant k, f can also be computed by a depth $2k$ circuit of size $s^{O(\sqrt[k]{d})}$. $\Diamond$

Note that when the degree is $O(n)$ (e.g. Det_n, Perm_n), the above theorem gives us a much smaller circuit than the trivial circuit of size $n^{O(n)}$.

Now that we know about depth-2 and depth-4 size of a polynomial with a size s circuit, it would be great to know what happens at depth 3. This question was answered by Gupta, Kamath, Kayal and Saptharishi [GKKS16] who showed that over fields of characteristic zero (e.g. rationals, reals), it was possible to further reduce the depth to just 3, while essentially maintaining the same size bound.

Theorem 1.21 (Depth reduction to depth 3 [GKKS16] (Informal))**.** *Suppose $f \in \mathbb{R}[\mathbf{x}]$ is a degree d polynomial that is computable by a circuit of size s. Then f can also be computed by a depth 3 formula of size $s^{O(\sqrt{d})}$.*

1.3.3 Hardness - Randomness Connections

Observe that the two central questions Question 1.5 and Question 1.10 are quite similar, in that both the tasks require us to find a "weakness" of the model being studied. Due to the existence of an efficient randomized algorithm, the task of blackbox PIT for algebraic models is seen as a *derandomisation* task, and a hitting set is seen as an object that any small algebraic circuit "can not distinguish from a random set". The fact that hard objects are useful in derandomisation tasks is well known and extensively studied across complexity theory, and algebraic circuit complexity is no exception, with proven connections going in either directions.

Hard polynomials from hitting sets. Perhaps the easier direction to verify is the fact that explicit hitting sets yield (somewhat) explicit hard polynomials.

Theorem 1.22 ([HS80, Agr05] (Informal))**.** *Suppose $\mathcal{C}$ is a class of n-variate polynomials of individual degree at most d, and let $\mathcal{H}$ be a hitting set for $\mathcal{C}$ of size $\ll d^n$. Then there exists an n-variate polynomial f of individual degree $\leq d$, such that f vanishes on all the points in $\mathcal{H}$ and therefore $f \notin \mathcal{C}$.*

It is not hard to see that a polynomial vanishing on a given set of points can be constructed by solving a system of linear equations, which is essentially the proof of the above theorem. However, solving such a system need not always yield an *explicit* polynomial, and pinning down the exact properties of $\mathcal{H}$ that give us a hard polynomial in say, VNP, remains an interesting open question.

Hitting sets from hard polynomials. The earliest known result in the other direction was proven by Kabanets and Impagliazzo [KI04], by extending the ideas from a similar work in the boolean setting [NW94]. The basic intuition was that for a "hard enough" polynomial f, the evaluations of f should look close to random to any "small" circuit. Of course, formalizing this intuition and getting a handle on the words "hard enough" and "small" requires all the work and the combinatorial machinery that goes into this proof. For now, we will state an informal version of a consequence of the original result.

Theorem 1.23 ([KI04] (Informal)). *Suppose $f(\mathbf{x})$ is a k-variate polynomial that requires circuits of size $\exp(\Omega(k))$ to compute it. Then for $n \approx \exp(k)$, the class $\mathcal{C}(n)$ of n-variate circuits of size and degree $\mathrm{poly}(n)$ has a hitting set of size $n^{O(k)}$.*

Note that the above statement is only true for algebraic circuits. This is because its proof crucially uses the fact that "factors of easy polynomials are easy", which was known only for circuits until the recent result of Sinhababu and Thierauf [ST20] who proved it for ABPs. However, some works have overcome this barrier and have proven similar results for some specialized settings.

Dvir, Shpilka and Yehudayoff [DSY09] explored the question of *closure under taking factors* for bounded depth circuits, and showed that hard-enough polynomials (of low individual degree) for a depth d yield sub-exponential hitting sets for depth $(d-5)$ circuits. Guo, Kumar, Saptharishi and Solomon [GKSS19] showed that a constant variate polynomial (over reals or complexes) that is almost optimally hard would lead to efficient hitting sets for algebraic circuits. Thus, their result shows that a constant variate polynomial that is optimally hard completely resolves Question 1.10. Another recent work of Andrews [And20] shows a similar result for fields of small characteristic, by building on the works on the technique of *bootstrapping hitting sets* [AGS19, KST19]. This phenomenon of *bootstrapping* is explored by one the works in this thesis, the details of which can be found in Chapter 5.

For a comprehensive and insightful discussion about hardness-randomness connections in algebraic complexity, refer to the recent survey by Kumar and Saptharishi [KS19].

1.4 Current Status

1.4.1 Lower Bounds

Since proving lower bounds for general circuits and formulas has proven to be hard, most of the research in algebraic circuit complexity has focused on more structured variants of these models. We shall now briefly survey some of those results that are relevant to rest of the thesis.

Constant Depth Models

Constant depth models are an interesting object of study mainly because of their simplicity. Further, the depth reduction results (see Section 1.3.2) only add to their importance. Nisan and Wigderson [NW97] proved an exponential lower bound on *homogeneous* depth-3 circuits for the *elementary symmetric polynomial*, which is known to have small circuits. Later, Shpilka and Wigderson [SW01] extended their techniques and showed a lower bound of $\Omega(n^2)$ on *non-homogeneous* depth-3 circuits for the same polynomial, which happens to be tight due to a construction that is attributed to Ben-Or. A further enhancement of the same technique was then used by Kayal, Saha, Tavenas [KST16] to give an almost cubic lower bound against depth-3 circuits for a polynomial in VNP. Later Balaji, Limaye and Srinivasan [BLS16] showed that same techniques could achieve essentially the same bound for a polynomial family in VP.

Note that the depth reduction results to depth-4 circuits yield *homogeneous* circuits; this saw a flurry of activity in homogeneous depth-4 circuit lower bounds. Using suitable variants of the measure of *shifted partial derivatives* introduced by Kayal [Kay12], a sequence of results showed exponential lower bounds against homogeneous depth-4 circuits. A lower bound of $\exp\left(\Omega(\sqrt{n})\right)$ for Det_n and Perm_n was shown by Gupta, Kamath, Kayal and Saptharishi [GKKS14], which was improved to a $\exp\left(\Omega(\sqrt{n}\log n)\right)$ lower bound for a polynomial in VNP by Kayal, Saha and Saptharishi [KSS14]. The same lower bound was later achieved for a polynomial in VP by Fournier, Limaye, Malod and Srinivasan [FLMS15]. All these results assumed an upper bound on the fan-in of the product gates in the bottom layer, which was the same as that in Theorem 1.20. For the setting of unrestricted bottom fan-in, Kayal, Limaye, Saha and Srinivasan [KLSS17] obtained an $n^{\Omega(\sqrt{n})}$ lower bound against homogeneous depth-4 circuits for a polynomial in VNP; this was also shown to be true for a polynomial in VP by Kumar and Saraf [KS14]. Note that among other things, this long line of work implies that the parameters in Theorem 1.20 are asymptotically tight. Recently, the work of Gupta, Saha and Thankey [GST20] gave a slightly super-quadratic lower bound against *non-homogeneous* depth-4 circuits, which is the best known lower bound in this setting.

Constant depth powering models. Replacing one or more product-layers in constant depth models with *powering* gates gives some interesting models; we will now see some results about them. The measure of *shifted partial derivatives* referred to in the above mentioned works was in fact developed by Kayal [Kay12] to study a fairly restricted variant of depth-4 circuits, that of *sums of powers of bounded-degree polynomials*, against which he showed an exponential lower bound for the monomial. A powering model that is almost fully understood in the context of lower bounds, is the *depth-3-powering* model which was first studied as a circuit model by [Sax08]. This model computes polynomials as *sums of powers of linear forms*. We now know tight *exponential* lower bounds on the size of a depth-3-powering circuit that computes a *monomial* [FS13, RS11]. This model is studied by one of the works in this thesis, which has been described in Chapter 4.

Multilinear Models

Since we are interested in the complexity of Det_n and Perm_n — which are both multilinear — *multilinear models* are also well studied. These models compute multilinear polynomials even in their intermediate computation, and thereby always output a multilinear polynomial. More formally, a circuit/formula is said to be a *syntactically multilinear circuit/formula* if for every product gate v in it, we have that the sets of variables, appearing in the left and the right subtrees of the tree rooted at v, are *disjoint*. Similarly, an ABP is said to be a *syntactically multilinear ABP*, if every s to t path in it computes a multilinear polynomial. We shall now see some of the results about multilinear models. We drop the word "syntactically" for this discussion for brevity.

The first lower bound of this kind came from a work of Raz [Raz06], where he showed a quasipolynomial ($n^{\Omega(\log n)}$) lower bound on the size of the syntactically multilinear formula computing a polynomial that has a poly(n) sized multilinear circuit. Note that this gives a tight separation between multilinear formulas and circuits due to the efficient depth reduction results (see Theorem 1.19). More results used variants of the techniques introduced by Raz [Raz06]. Exponential lower bounds were shown against constant depth multilinear circuits by Raz and Yehudayoff [RY09], Raz [Raz09] showed a quasipolynomial lower bound on the multilinear formulas computing Det_n and Perm_n, and Hrubeš and Yehudayoff [HY11b] gave a super-polynomial lower bound against *homogeneous* multilinear formulas for a polynomial that has $O(n^2)$ size depth-3 *non-homogeneous* formulas. We also know of super-polynomial separations between multilinear formulas and ABPs, from the work of Dvir, Malod, Perifel and Yehudayoff [DMPY12] who proved a $n^{\Omega(\log n)}$ separation, which is also known to be tight.

Non-commutative Models

A slightly different class of models that is well studied, is that of *non-commutative* algebraic models. In a non-commutative model, we forgo the assumption that the underlying variables commute under multiplication. In other words, we treat xy and yx as distinct monomials for all pairs of distinct variables x, y. A good use case for these models is polynomials over matrices. Non-commutativity makes it harder to compute a certain monomial in different ways, and therefore it is slightly easier to prove lower bounds against these models.

Nisan initiated the study of non-commutative models through his landmark work [Nis91], which proves exponential lower bounds against non-commutative ABPs computing the determinant and the permanent polynomials. He showed these results by providing an *exact characterization* for the non-commutative ABP complexity of a polynomial. This characterization has also led to some results on PIT and *reconstruction* of non-commutative ABPs. Lagarde, Malod and Perifel [LMP19] extended Nisan's characterization and showed finer separations between non-commutative ABPs and circuits; they introduced a new model called *Unique Parse Tree (UPT) circuits* and showed that ABPs, UPT circuits and circuits, were exponentially separated, in that very order. An intriguing result about non-commutative models due to Arvind and Srinivasan [AS18] is worth mentioning: they show that (the non-commutative version of) Det_n is as hard as (the non-commutative version of) Perm_n for non-commutative circuits, which in particular means that we expect a super-polynomial lower bound on the non-commutative circuit size of Det_n, a stark contrast from the commutative world.

1.4.2 Hitting Sets

Explicit hitting set constructions have always been preceded by strong explicit lower bounds against the same models. Thus, PIT is seen as a harder task than lower bounds, and very few strong lower bounds that we just saw have resulted in constructions of non-trivial hitting sets.

Unlike in the case of lower bounds, designing hitting sets even for depth-2 circuits (also called *sparse polynomials*) becomes a non-trivial question. Efficient hitting sets for the class of all sparse polynomials are known due to Klivans and Spielman [KS01], and Agrawal and Biswas [AB03]. This result can be seen as a building block for most non-trivial hitting set constructions that we know of today.

A slightly more powerful (yet simple) constant depth model is that of *depth-3-powering* circuits, for which we know of quasipolynomial ($s^{O(\log s)}$) sized hitting sets due to Forbes and Shpilka [FS12], and Agrawal, Saha and Saxena [ASS13]. Later, Forbes, Saptharishi and Shpilka [FSS14] improved this to a hitting set of size $s^{O(\log \log s)}$. Forbes, Ghosh and Saxena [FGS18] give an efficient hitting set for this model, when the number of variables depends logarithmically on the size ($n = O(\log s)$). Obtaining efficient hitting sets for depth-3-powering circuits in the general setting remains an interesting open question.

The commutative analogues of non-commutative ABPs, called Read-once Oblivious ABPs (ROABPs) have been extensively studied in the context of hitting sets, perhaps due to Nisan's characterization [Nis91] which extends to ROABPs. The first explicit construction of hitting sets for ROABPs came from the work of Forbes and Shpilka [FS13], who gave an explicit hitting set of size $s^{O(\log n)}$ in what is called a *known variable order* setting[7]. Agrawal, Gurjar, Korwar and Saxena [AGKS15] later achieved the same parameters in the *unknown variable order* setting. Gurjar, Korwar, Saxena and Theirauf [GKST17] constructed hitting sets of size $(nds)^{O(\log n)}$ for sums of constantly many ROABPs, and Gurjar, Korwar and Saxena [GKS17] gave efficient hitting sets for ROABPs of *constant width* in the *known variable order* setting. Recently, Guo and Gurjar [GG20] gave a construction that matches most of the currently known hitting sets for ROABPs and improves on some parameters in some restricted settings.

Apart from ROABPs and depth-3-powering circuits, non-trivial hitting sets are known for restricted subclasses of other well-studied models, like constant depth multilinear circuits with bounded top fan-in (cf. [SS10, AvMV15, ASSS16]), constant depth multilinear circuits (cf. [OSV16]), and read-once formulas (cf. [MV17]).

1.4.3 Analysing Lower Bound Techniques

As we saw, there has been significant progress on lower bounds against several structured algebraic models. However, this has not quite translated into any strong lower bounds against the general models like circuits or formulas. This disparity has recently attracted some attention from the community, and some notable results that comment on the efficacy of current lower bound techniques have been presented. The key observation in most of these results

[7]This setting is technically not "fully blackbox", but it still subsumes the hitting set question for non-commutative ABPs.

is that several lower bound techniques that have been successful against structured models have a particular structure. For instance, the techniques used in almost all the lower bounds against constant depth models can be seen as variants of *the method of partial derivatives*, which was first introduced by Nisan and Wigderson [NW97].

A significant result analysing the efficacy of lower bound techniques that followed a "common template" was that of Razborov and Rudich [RR97], in the world of *boolean circuits*. They observed that almost all the known lower bounds against boolean circuit models had been proved (or could be reproved) using a common template, which they called a *natural proof strategy*. They then showed that under a widely believed cryptographic assumption, no natural proof strategy would be able to show a super-polynomial lower bound against boolean circuits. Thus, they showed the existence of a *natural proofs barrier* towards proving boolean circuit lower bounds.

Considering that algebraic circuits seem like a fairly general and powerful model of computation, it is tempting to think that the *natural proofs barrier* of Razborov and Rudich [RR97] also extends to this setting. This problem turns out to be a non-trivial one, and indeed it is not known whether their results extend to algebraic circuits. This question is closely related to the question of whether cryptographically secure algebraic pseudorandom functions can be computed by small and low degree algebraic circuits and there does not seem to be substantial evidence one way or the other on this [8].

It was observed by various authors ([AD08, Gro15, FSV18, GKSS17]) that most of the currently known proofs of algebraic circuit lower bounds fit into a common unifying framework, not unlike that for boolean circuits ([RR97]), although of a more algebraic nature. These proofs implicitly go via defining a property for the set of all polynomials and using this property to separate the hard polynomial from the easy ones. Informally, any algebraic circuit lower bound which goes via defining a property that can be efficiently simulated by a polynomial, is an *algebraically natural proof* of a lower bound.

Analogous to the abstraction of *natural proofs* for Boolean circuit lower bounds, this framework of *algebraically natural proofs* turns out to be rich and general enough that almost all of our current proofs of algebraic circuit lower bounds are in fact algebraically natural, or can be viewed in this framework with a little work [Gro15]. Thus, this definition seems like an important first step towards understanding the strengths and limitations of many of our current lower bound techniques in algebraic complexity.

The recent works of Forbes, Shpilka and Volk [FSV18], and Grochow, Kumar, Saks and Saraf [GKSS17] argue that under an appropriate (but non-standard) pseudorandomness assumption of the existence of *succinct hitting sets*, the answer to the question above is negative. That is, algebraically natural proof techniques cannot be used to show strong lower bounds for algebraic circuits. Using this observation, [FSV18] showed that for various restricted circuit classes $\mathcal{C}$ and $\mathcal{D}$, lower bounds for $\mathcal{C}$ cannot be proved via properties that can be simulated by polynomials in $\mathcal{D}$. However, this question has remained unanswered for more general

[8]Refer to [AD08] and [FSV18] for a more detailed discussion on this issue.

circuit classes $\mathcal{C}$ and $\mathcal{D}$. In particular, we do not have any strong evidence for or against the existence of efficient equations for VP. We expand on this discussion in Chapter 6, where we also present some evidence *in favour of* the natural lower bound techniques.

1.5 Contributions of the thesis

This thesis studies hitting sets for algebraic circuits and other restricted models. In particular, the thesis provides two hitting set constructions for restricted models, and two interesting consequences of explicit hitting sets for algebraic models. We now briefly state these results before discussing them in more detail in the subsequent sections.

1.5.1 Hitting Sets for UPT circuits

The first result is a hitting set construction for the non-commutative model of *UPT circuits*, which was introduced by Lagarde, Malod and Perifel [LMP19]. In their work and a follow-up work by Lagarde, Limaye and Srinivasan [LLS19], several whitebox PITs which were known for non-commutative ABPs were lifted to UPT circuits. Thus an interesting open question was whether the hitting set constructions for non-commutative ABPs (and ROABPs) could be extended to UPT circuits. We answer this question in the affirmative and extend the known results for ROABPs [AGKS15, GKST17, GKS17] to the corresponding analogues of UPT circuits. Here are two of our main results about explicit hitting sets. We expand on our results on UPT circuits in Chapter 3.

Theorem 1.24 (Hitting sets for UPT circuits [ST18])**.** *For $n, d, s \in \mathbb{N}$, let $\mathcal{U}(n, d, s) \subset \mathbb{F}\langle \mathbf{x} \rangle$ be the class of all n-variate, degree d polynomials computed by UPT circuits of size at most s. Then there is a set of matrices $\mathcal{H}(n, d, s) \subset \mathbb{F}^{(d+1)\times(d+1)}$ of size $(nds)^{O(\log d)}$ that is a hitting set for $\mathcal{U}(n, d, s)$. Further, $\mathcal{H}(n, d, s)$ can be constructed in time $(nds)^{O(\log d)}$.*

Theorem 1.25 (Hitting sets for sums of UPT circuits [ST18])**.** *For $n, d, s, c \in \mathbb{N}$, let $\mathcal{U}(n, d, s, c) \subset \mathbb{F}\langle \mathbf{x} \rangle$ be the class of all n-variate, degree d polynomials computed by sums of at most c UPT circuits, each of size at most s. Then there is a set of matrices $\mathcal{H}(n, d, s, c) \subset \mathbb{F}^{(d+1)\times(d+1)}$ of size $(nds)^{\left(2^{O(c)} \log d\right)}$ that is a hitting set for $\mathcal{U}(n, d, s, c)$. The set $\mathcal{H}(n, d, s, c)$ is constructible in time $(nds)^{O(\log d)}$.*

Key Ideas

Our hitting set construction in Theorem 1.24 extends the ideas in the work of Agrawal, Gurjar, Korwar and Saxena [AGKS15], who use a *divide-and-conquer* approach to construct a *basis isolating weight assignment (BIWA)* for ROABPs. With some effort, one can see that essentially the same divide-and-conquer method when applied to UPT circuits, yields hitting sets of size $s^{O(r)}$ for UPT circuits of size s and depth r. At this point, if one shows that any UPT circuit for a degree d polynomial can be simulated by a UPT circuit of depth of $O(\log d)$ (similar to

Theorem 1.19), then we would get hitting sets of size $n^{O(\log n)}$ for UPT circuits of size $\mathrm{poly}(n)$, thus proving Theorem 1.24.

However, such a depth-reduction result is known to be false. Using the techniques in [LMP19] (or even [Nis91]), we can show that the palindrome polynomial Pal_d requires UPT circuits of depth $\omega(d/\log d)$, even though it can be computed by a $\mathrm{poly}(d)$ sized UPT circuit. However, even though Pal_d cannot be computed by small depth non-commutative circuits, we observe that a *shuffling* of the palindrome does indeed have circuits of small depth. For a permutation σ, we define the *σ-shuffling* of a degree d non-commutative homogeneous polynomial f, to be the polynomial that is obtained by permuting each monomial of f using σ. We prove our depth reduction result (Theorem 3.4) under a suitable shuffling, essentially by following the strategy of [VSBR83] and [AJMV98], while using the UPT structure based on the underlying shape of the parse trees. Intuitively, one can say that the argument is basically about "balancing" the underlying shape of all the parse trees of the given UPT circuit.

As mentioned earlier, with the depth-reduction statement at our disposal, we obtain hitting sets for UPT circuits by lifting the technique of *basis isolating weight assignments* of Agrawal, Gurjar, Korwar and Saxena [AGKS15] to this more general setting to obtain Theorem 1.24. In fact, once we prove the depth-reduction theorem, all our results about hitting sets UPT circuits and their variants follow from a careful translation of the existing works ROABPs [AGKS15, GKST17, GKS17] to the setting of UPT (or FewPT) circuits. Consequently, this also generalizes the respective hitting sets from ROABPs to *UPT (or FewPT) set-multilinear* circuits.

1.5.2 Isolating Log-variate Depth-3-Powering Circuits

Our other hitting set construction is for a special case of *depth-3-powering circuits* or *sums of powers of linear forms* (denoted by $\Sigma\wedge\Sigma$). The best known hitting sets for n-variate, degree d, size s depth-3-powering circuits ($\Sigma^s\wedge^d\Sigma^n$) have size $(nds)^{O(\log\log n)}$ [FSS14]. Forbes, Ghosh and Saxena [FGS18] considered a *low-variate* version of this model, where they restricted the number of variables to be $O(\log(sd))$. They provided a blackbox PIT for this low-variate variant that ran in time $\mathrm{poly}(2^n, s, d)$ which is efficient in the size of the circuit when $n = O(\log(sd))$. We construct an *explicit isolating weight assignment* for low-variate depth-3-powering circuits using *Newton polytopes*. A consequence of the weight assignment construction is a hitting set of size $\mathrm{poly}(s,d)$ for log-variate depth-3-powering circuits. We now state our result and briefly discuss the main proof ideas. Rest of the details of this result are discussed in Chapter 4.

Theorem 1.26 (Isolating weight assignment for log-variate $\Sigma\wedge\Sigma$). *Let $\mathbb{F}$ be a field of characteristic zero, and for $s,d\in\mathbb{N}$, let $r,m = \mathrm{poly}(s,d)$ be large enough. There exist weight assignments $w_1, w_2, \ldots, w_r$ with $w_i : [n] \to [m]$ for all $i\in[r]$, such that for any polynomial $f(\mathbf{x}) \in \Sigma^{[s]}\wedge^{[d]}\Sigma$ depending on $\ell = O(\log(sd))$ variables, there exists an $i\in[r]$ such that f has a* unique minimum weight monomial *according to w_i.*

Key Ideas

Our main task is to construct an explicit weight assignment that *isolates* (see Definition 2.16) polynomials computed by a $\Sigma \bigwedge \Sigma$ circuit. It is easy to see that it is impossible to come up with a single explicit assignment that isolates all such polynomials, so the goal is then to come up with a small set of assignments, such that for any fixed polynomial at least one of the assignments in the set works (see Definition 2.17).

We observe that for any polynomial f, a weight assignment wt is a *linear function* on the *exponent vectors* of the monomials in f, and hence it can be seen as a linear function on the *Newton polytope of f* (see definition Definition 4.5). Thus, if we can design a linear function $\ell(\mathbf{e})$ that is uniquely minimized at a vertex $\mathbf{a}$ of the Newton polytope of f, then we will be done.

One way to achieve this is to design a weight assignment that gives distinct weights to all the vertices of the polytope. Therefore, if we bound the number of vertices of the Newton polytope for any polynomial computable by a $\Sigma \bigwedge \Sigma$ circuit, by say r, then we can come up with a set of $\text{poly}(r)$ assignments using known techniques [KS01, AB03]. In order to prove such a bound, we then observe that for any polynomial f and any vertex $\mathbf{a}$ of the Newton polytope of f, the *dimension of partial derivatives* of f is lower bounded by the *cone-size* of $\mathbf{a}$, which extends an observation of Forbes [For14, Corollary 8.4.13]. We then apply previously known bounds on the dimension of partial derivatives for $\Sigma \bigwedge \Sigma$ [For14, Lemma 8.4.8] and the number of monomials of low-cone size [FGS18] to derive Theorem 4.2.

1.5.3 *Bootstrapping* Hitting Sets

Our next result is about an interesting consequence of obtaining mildly non-trivial hitting sets for constant-variate models. In a key step towards understanding hitting sets, Agrawal, Ghosh and Saxena [AGS19] showed a surprising relation between constant-variate hitting sets and the central hitting set question Question 1.10. They showed that for all large enough constants k, even a marginal improvement in the trivial exponential hitting sets, like $d^{k^{0.49}}$ (instead of d^k), will imply an almost-polynomial sized $(s^{\text{tiny}(s)})$[9] hitting sets for general size s, degree s circuits. Some questions that arise out of the statement above are: (a) what about smaller constants k? (b) does a similar conclusion follow from an even milder improvement like $s^{k^{0.99}}$? (c) does a similar statement hold for formulas/ABPs? Building on the techniques in [AGS19], we answer all these questions in the affirmative, and prove the following result. More details are discussed in Chapter 5.

Theorem 1.27 (Near-optimal bootstrapping of hitting sets [KST19]). *Let $k \geq 2$ and $\varepsilon > 0$ be any constants, and suppose for all large s there are explicit hitting sets of size $s^{k-\varepsilon}$ for the class of all k-variate, degree s polynomials that have circuits of size s.*

Then for all large s, the class of all s-variate, degree s polynomials that have circuits of size s has explicit hitting sets of size $s^{\text{tiny}(s)}$.

[9]For $\text{tiny}(s) = \exp(\exp(O(\log^* s)))$.

Furthermore, the above statement continues to be true when circuits are replaced by ABPs or formulas in both *the hypothesis and the conclusion.*

Key Ideas

Both the above bootstrapping theorems exploit the hardness-randomness connections due to Kabanets and Impagliazzo [KI04], Heintz and Schnorr [HS80] and Agrawal [Agr05] (see Theorem 1.22 and Theorem 1.23).

The core idea of *bootstrapping* in [AGS19] was to observe that for a careful setting of parameters, one could start with a marginally non-trivial low-variate hitting set, obtain a hard polynomial using Theorem 1.22 ([HS80, Agr05]), and use that hard polynomial to construct much better hitting sets for polynomials of a much larger arity using Theorem 1.23 ([KI04]). The above mentioned result in [AGS19] can then be proven by starting with constant variate hitting sets and repeating the above process several times till the arity saturates to s.

In order to obtain the stronger statement of Theorem 1.27, we instantiate the proof of Theorem 1.23 specifically for hard polynomials obtained via hitting sets using Theorem 1.22. This small but crucial change lets us handle seemingly weaker models of formulas and ABPs, and additionally allow for a wider range of parameters in the bootstrapping process. Thus, we can do a finer analysis of the core *bootstrapping* procedure to obtain Theorem 1.27, which turns out to be a significant strengthening of results in [AGS19].

1.5.4 Algebraically Natural Proofs

The final result in this thesis investigates the notion of *algebraically natural proofs* proposed by Forbes, Shpilka and Volk [FSV18] and Grochow, Kumar, Saks and Saraf [GKSS17] in order to assess the efficacy of lower bound techniques. They essentially showed that if the class VP had *succinct hitting sets* then there were no "natural lower bound techniques" that were useful against VP. No such hitting sets are known to (provably) exist for VP, but note that the hypothesis depends only on their existence, and not a proof of correctness.

However, due to the existence of strong *positive* connections between hitting sets and lower bounds (see Section 1.3.3), this statement seems a bit counter-intuitive. A natural question therefore is what sort of lower bound *proofs* can be obtained from hitting sets. We investigate this and essentially *algebraize* the proofs in these connections [HS80, Agr05] to prove the following result about *equations for* VP: an object closely connected with proofs for circuit lower bounds against VP. Since our theorem covers almost all well-studied polynomials, it provides some evidence *against* a barrier towards proving lower bounds using natural techniques. More details on the result are discussed in Chapter 6.

Theorem 1.28 (Defining equations for VP with small coefficients [CKR+20]). *Let* VP$'$ *be the class of all polynomial families in* VP *that involve polynomials with* small *coefficients. Then for all large $n, d \in \mathbb{N}$ and $N = \binom{n+d}{n}$, there exists an N-variate polynomial $P(\mathbf{Z}) \in \mathsf{VP}_N$[10] such that:*

[10]That is, $\mathrm{size}(P_N), \deg(P_N) = \mathrm{poly}(N)$.

- *For all* $f \in \mathsf{VP}'$, $P(\overline{\mathrm{coeff}}(f)) = 0$.
- *There exists an* n*-variate, degree* d *polynomial* $h(\mathbf{x})$ *for which* $P(\overline{\mathrm{coeff}}(h)) \neq 0$.

Key Ideas.

At a high level, the idea behind our results is to try and come up with a *non-trivial* property of polynomials which every polynomial with a small circuit satisfies. The hope is that once we have such a property (which is nice enough), one can try to transform this into a defining equation via an appropriate *algebraization*. The property that we end up using is the existence of (non-explicit) *hitting sets* for polynomials with small circuits.

Let us consider the map $\Phi_{\mathcal{H}}$, defined by the hitting set $\mathcal{H}$ of $\mathcal{C}$ on the set of all polynomials, that maps any given polynomial f to its evaluations over the points in $\mathcal{H}$. It is clear from the above observation that any nonzero polynomial in the kernel of $\Phi_{\mathcal{H}}$ is guaranteed to be outside $\mathcal{C}$. Thus, if there were a nonzero polynomial that vanishes on all polynomials $f \notin \ker(\Phi_{\mathcal{H}})$, we would have a defining equation for $\mathcal{C}$. Moreover, if such a polynomial happened to have its degree and circuit complexity polynomially bounded in its number of variables, we would be able to prove a statement like Theorem 1.28. However, note that *not* being in the kernel of a linear map seems to be a tricky condition to check via a polynomial (as opposed to the complementary property of *being* in the kernel, which can be easily checked via a polynomial). To prove our theorems, we get past this issue in the setting of small finite fields, and for polynomials over $\mathbb{C}$ with bounded integer coefficients.

Over a finite field $\mathbb{F}$, a univariate polynomial that maps every nonzero $x \in \mathbb{F}$ to zero and vice versa, already exists in $q(x) = 1 - x^{|\mathbb{F}|-1}$. Therefore, for a given polynomial f, the defining equation essentially outputs $\prod_{\mathbf{h}\in\mathcal{H}} q(f(\mathbf{h}))$. Clearly, for a polynomial f, $\prod_{\mathbf{h}\in\mathcal{H}} q(f(\mathbf{h}))$ is zero if and only if f evaluates to a nonzero value on at least one point in $\mathcal{H}$.

To generalize this to other fields, we wish to find a "low-degree" univariate $q(x)$ that maps nonzero values to 0, and zero to a nonzero value. We observe that when the polynomials in $\mathcal{C}$ have integer coefficients of bounded magnitude , we can still obtain such a univariate polynomial, and in turn a non-trivial defining equation. Indeed, if q were such a univariate, we essentially output $\prod_{\mathbf{h}\in\mathcal{H}} q(f(\mathbf{h}))$, for a given polynomial f. This step relies on a simple application of the Chinese Remainder Theorem. In order to show that the equations are *non-trivial*, we need to show that there are nonzero polynomials with bounded integer coefficients which vanish everywhere on the hitting set $\mathcal{H}$. We show this via a lemma of Siegel[11].

As it turns out, our proofs do not use much about the class VP except for the existence of small (non-explicit) hitting sets for the class. Since this property also holds for the seemingly larger class VNP, our results here also extend to VNP.

Recent update. Soon after the compilation of this thesis, in a joint work with Kumar, Ramya and Saptharishi [KRST20], we showed the following. Assuming that $\{\mathsf{Perm}_n\}$ requires circuits

[11]A statement of the lemma can be found here. Refer to [Sie14] for details.

of size $2^{n^{\varepsilon}}$ for some constant ε, the class VNP *does not* have efficiently computable equations. In the context of the results in Chapter 6, it might appear that these works contradict each other. It will be easier to address this issue after understanding the results in Chapter 6 in more detail, and hence we defer this discussion towards the end of that chapter.

1.6 Organization of the Thesis

In Chapter 2, we shall fix some notation for the rest of the thesis and provide some necessary technical background for the results mentioned above. The work on constructing hitting sets for UPT circuits and their variants is discussed in Chapter 3, and the result about hitting sets for log-variate depth-3-powering circuits is discussed in Chapter 4. The next two chapters discuss our works on the consequences of non-trivial hitting sets. The results on bootstrapping of hitting sets are discussed in Chapter 5, and Chapter 6 deals with our results surrounding the notion of algebraically natural proofs. Finally, we provide some concluding remarks and some directions for future work in Chapter 7.

2 | Preliminaries

In this chapter we will formally define and state some concepts and results that were alluded to in the previous chapter and those which will be useful for the rest of the thesis. We will also fix some notation that we use throughout the thesis.

2.1 Basic Notation

- We use lower-case boldface characters like $\mathbf{x}, \mathbf{y}, \mathbf{a}, \ldots$ to denote vectors (ordered sets) of variables and constants, and use indexed lower-case letters to refer to individual elements, e.g. $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$.

- For a positive integer k, we denote the set $\{1, 2, \ldots, k\}$ by $[k]$. We also use boldface letters with subscripts to denote the number of variables or constant in a vector, for instance $\mathbf{x}_{[4]}$ will mean $\{x_1, x_2, x_3, x_4\}$; we will drop the subscript whenever the size is irrelevant or clear from the context.

- For an arbitrary vector $\mathbf{a}_{[n]}$ and some $\mathbf{e} \in \mathbb{N}^n$, we use the shorthand $\mathbf{a}^{\mathbf{e}}$ for the "monomial" $a_1^{e_1} a_2^{e_2} \cdots a_n^{e_n}$.

- For a polynomial f, we denote by $\operatorname{supp}(f)$ the set of all monomials that appear in f with a nonzero coefficient (called the *support of f*), and we use the notation $\operatorname{coeff}_f(m)$ for the coefficient of m in f.

- We use $\deg(f)$ to denote the (total) degree of f. For a specific variable t, we sometimes use $\deg_t(f)$ to denote the degree of f in t, which is the maximum exponent that t appears with in any monomial in f.

  We naturally extend this notation to sets of variables, that is for $f(\mathbf{x}, \mathbf{y})$ we use $\deg_{\mathbf{x}}(f)$ to denote the degree of f in terms of the $\mathbf{x}$ variables.

- For a polynomial $f(\mathbf{x}_{[n]})$, when we say that the *individual degree of f* is at most k, we mean that for all $i \in [n]$, $\deg_{x_i}(f) \leq k$.

- We use $\mathbb{F}[\mathbf{x}]^{\leq d}$ to denote polynomials over the field $\mathbb{F}$ in variables $\mathbf{x}$ of degree at most d, and use $\mathbf{x}^{\leq d}$ to denote the set of all monomials in variables $\mathbf{x}$ of degree at most d.

- For a polynomial $f(\mathbf{x})$, we will write $f = 0$ to denote that f is *the zero polynomial*, or $f(\mathbf{x}) \equiv 0 \in \mathbb{F}[\mathbf{x}]$. On the other hand, for f being zero at a specific point, say $\mathbf{a} \in \mathbb{F}^n$, we write $f(\mathbf{a}) = 0$.

- For a substitution map $\Phi : \mathbf{x} \to \mathbb{F}[\mathbf{t}]$ and a polynomial $f(\mathbf{x})$ we use $\Phi(f)$ to refer to the polynomial $f(\Phi(\mathbf{x})) \in \mathbb{F}[\mathbf{t}]$.

2.2 Computing Polynomials

- For a certain polynomial $f(\mathbf{x})$, we will use $\mathrm{size}(f)$ to refer to the size of the smallest *algebraic circuit* (see Definition 1.1) that computes f.

- For an algebraic model $\mathcal{A}$, we sometimes use the phrase "$\mathcal{A}$-complexity of f" to talk about the size of the smallest $A \in \mathcal{A}$ that computes f (see also Remark 2.2).

- We study the circuit complexity of certain *polynomial families*; a polynomial family should be seen as a collection of polynomials of growing arity (number of variables) that are somehow interrelated.
  We use $\{f_n\}_{n\in\mathbb{N}}$ to denote families of polynomials. We drop the index set whenever it is clear from context.

Definition 2.1 (Family of low-degree polynomials). *A sequence of polynomials $\{f_n\}$ given by $\{f_1, f_2, \ldots, f_n, \ldots\}$ is called a family of low-degree polynomials if there exist constants c_1, c_2 such that for all large enough $n \in \mathbb{N}$ the nth polynomial f_n depends on $m(n) \leq n^{c_1}$ variables and has degree $d(n) \leq n^{c_2}$.* $\Diamond$

Remark 2.2. *While talking about the computability of a polynomial $f_n \in \mathbb{F}[\mathbf{x}]$, the underlying model is allowed to use constants from some extension field $\mathbb{K} \supset \mathbb{F}$.* $\Diamond$

Definition 2.3 (VP: Efficiently computable families). *A family of low-degree polynomials $\{f_n\}$ over a field $\mathbb{F}$ is said to be in $\mathsf{VP}_\mathbb{F}$ (or just VP when the field is clear from context) if there exists a constant c such that for all large enough n, the polynomial f_n is computable by a circuit of size at most n^c.* $\Diamond$

Definition 2.4 (VNP: Efficiently definable families). *A low-degree family of polynomials $\{f_n\}$ over a field $\mathbb{F}$ is said to be in $\mathsf{VNP}_\mathbb{F}$ (or just VNP when the field is clear from context) if there exists a constant c such that for all large n there is an $m \leq n^c$ and an $(n+m)$-variate polynomial $g_{n+m}(\mathbf{x}, \mathbf{y})$ of degree at most n^c which has an algebraic circuit of size at most n^c that satisfies*

$$f_n(\mathbf{x}) = \sum_{\mathbf{a}\in\{0,1\}^m} g_{n+m}(\mathbf{x}, \mathbf{a}).$$

$\Diamond$

2.2.1 Some Important Polynomial Families

We now formally define some of the important polynomial families that are studied extensively in algebraic complexity theory.

Definition 2.5 (Elementary symmetric polynomial). *For all $n, d \in \mathbb{N}$ with $d \leq n$, the n-variate symmetric polynomial of degree d is defined as follows.*

$$\mathsf{ESym}_{n,d} = \sum_{\substack{S \subseteq [n] \\ |S|=d}} \prod_{i \in S} x_i$$

$\Diamond$

Definition 2.6 (Determinant). *For all $n \in \mathbb{N}$, the nth (symbolic) determinant is the following n^2-variate polynomial of degree n.*

$$\mathsf{Det}_n = \sum_{\sigma \in s_n} \mathsf{sign}(\sigma) \prod_{i \in [n]} x_{i,\sigma(i)}$$

$\Diamond$

Definition 2.7 (Permanent). *For all $n \in \mathbb{N}$, the nth (symbolic) permanent is the following n^2-variate polynomial of degree n.*

$$\mathsf{Perm}_n = \sum_{\sigma \in s_n} \prod_{i \in [n]} x_{i,\sigma(i)}$$

$\Diamond$

Note that each of the above defined polynomials naturally define the corresponding polynomial families of the appropriate degree.

2.3 Hitting Set Basics

In this section we will go over some of the concepts that are frequently used while working with hitting sets for classes of polynomials. We start by stating the lemma of Heintz and Schnorr [HS80] showing that a random set of points is a hitting set.

Lemma 2.8 ([HS80] (Informal)). *For $n, d, s \in \mathbb{N}$ large enough, let $\mathcal{C}(n, d, s)$ be the class of all n-variate polynomials of total degree at most d that are computable by circuits of size at most s. Then for $w = (ds)^2$ and $r = (ns)^3$, a sequence of points $a_1, a_2, \ldots, a_r$ sampled uniformly from the grid $[w]^n$ forms a hitting set for $\mathcal{C}(n, d, s)$ with probability at least 0.99.*

A note on explicitness. We will be using the notion of "explicitness" of certain objects, especially hitting sets and functions between positive integers. Throughout this chapter, unless stated otherwise, we will use the following definition of explicitness.

Definition 2.9 (Explicit sets and functions). *A set of points $\mathcal{H} \in \mathbb{F}^n$ is said to be explicit, if there is an algorithm $A_{\mathcal{H}}$ which when given n as an input, outputs $\mathcal{H}$ in time* $\operatorname{poly}(|\mathcal{H}|, n)$.

A function $w : \mathbb{N} \to \mathbb{N}$ is said to be explicit, if there is an algorithm A_w, which for every k outputs $w(k)$ in time $\operatorname{poly}(k)$. $\Diamond$

2.3.1 Hitting Sets and Hitting Set Generators

Since the trivial hitting set for n-variate, degree d polynomials is of size $d^{O(n)}$, a very good way to get a smaller hitting set for a specific class $\mathcal{C}$ of circuits is to reduce the number of variables, while preserving the nonzeroness, for any circuit $C \in \mathcal{C}$. The obvious choice for such an operation, is a polynomial map that maps k-tuples in the underlying field $\mathbb{F}$ to n-tuples, for some $k \ll n$. A polynomial map that "preserves the nonzeroness" for any circuit in $\mathcal{C}$ is called a *variable reduction map* for $\mathcal{C}$.

Definition 2.10 (Variable reduction map). *Let $\mathcal{C} \subseteq \mathbb{F}[\mathbf{x}]$ be a class of n-variate, degree-d polynomials, and let $\mathbf{g}(\mathbf{t}_{[k]}) \in \mathbb{F}[\mathbf{t}_{[k]}]^n$ be a k-variate polynomial map given by a tuple of n polynomials $(g_1(\mathbf{t}), \ldots, g_n(\mathbf{t}))$ for some $k < n$. We call $\mathbf{g}(\mathbf{t})$ a* variable reduction map *for $\mathcal{C}$, if for every nonzero $C \in \mathcal{C}$ we have that $C(\mathbf{g}(\mathbf{t}))$ remains a nonzero polynomial in $\mathbb{F}[\mathbf{t}]$.*

We will say that a variable reduction map has degree at most r, if $\deg(g_i) \le r$ *for all $i \in [n]$.* $\Diamond$

One can then obtain a hitting set from a variable reduction map, which is stated in the following easy lemma. Hence we will now refer to variable reduction maps as *hitting set generators (HSGs)*, which is the more commonly used name.

Lemma 2.11. *Let $\mathcal{C}(n,d)$ be a class of n-variate, degree d polynomials, and suppose $\mathbf{g}(\mathbf{t})$ is a k-variate, degree r* hitting set generator *for $\mathcal{C}(n,d)$. Then $\mathcal{C}(n,d)$ has a hitting set of size at most $(dr)^{O(k)}$.*

Proof. We know that for any nonzero $C \in \mathcal{C}(n,d)$, $C(\mathbf{g}(\mathbf{t}))$ is also nonzero; which happens to be a k-variate polynomial of degree at most dr. Therefore $C(\mathbf{g}(\bar{\alpha})) \neq 0$ for some $\bar{\alpha} \in (dr+1)^k$, and thus the set $\mathcal{H} = \{\mathbf{g}(\bar{\alpha}) : \bar{\alpha} \in (dr+1)^k\}$ is a hitting set for $\mathcal{C}$. $\square$

Just as we can obtain hitting sets from HSGs, we can also obtain HSGs from hitting sets using elementary *polynomial interpolation*.

Lemma 2.12. *Suppose $\mathcal{H} \subseteq \mathbb{F}^n$ is a hitting set for some class $\mathcal{C}$ of n-variate polynomials, with $|\mathcal{H}| = r$. Then there is a (univariate) hitting set generator $\mathbf{g}(t) \in \mathbb{F}[t]^n$ for $\mathcal{C}$ of degree at most r.*

More generally, for any k, there is a k-variate hitting set generator $\mathbf{g}(\mathbf{t}_{[k]}) \in \mathbb{F}[\mathbf{t}_{[k]}]^n$ for $\mathcal{C}$ of individual *degree at most $\lceil r^{1/k} \rceil$.*

In both the cases, the corresponding hitting set generator $\mathbf{g}$ can be found in time $\operatorname{poly}(r)$.

Proof. Let the hitting set be $\{h_1, \ldots, h_r\}$. We will assume that the field has size at least r[1].

For the univariate HSG, we construct polynomials $g_1(t), \ldots, g_n(t)$, each of degree at most $(r-1)$, such that for each $j \in [n]$, $g_j(i) = h_i(j)$. The vector of polynomials $\mathbf{g}(t)$ therefore reads out $\mathcal{H}$ on inputs $1, 2, \ldots, r$. Now for each j, let $c_j \in \mathbb{F}^r$ be the (purported) coefficient vector for

[1] In case it does not, we can always work with an extension of $\mathbb{F}$ of degree $O(\log r)$.

$g_j(t)$, and let e_j be such that $e_j(i) = h_j(i)$. Then the required conditions on the evaluations of $g_j(t)$ can be written as the matrix-vector product $Vc_j = e_j$. Here V is the $r \times r$ *Van der Monde* matrix for the points in $[r]$, which is invertible because the underlying points are distinct (any distinct r points work). We can therefore just solve for v_j to obtain $g_j(t)$ in time $\text{poly}(r)$.

In the k-variate setting, we work with the grid $[w]^k$ for $w = \lceil r^{1/k} \rceil$. Following the same steps, we obtain a matrix-vector product $Vc_j = e_j$ for each k-variate $\mathbf{g}(\mathbf{t})$ with *individual* degree $(w-1)$. Here the matrix $V \in \mathbb{F}^{r \times r}$ is a "tensor product" of k copies of the (univariate) $w \times w$ Van der Monde matrix for the points in $[w]$, which is again invertible. The hitting set generator $\mathbf{g}(\mathbf{t})$ is therefore again computable in time $\text{poly}(r)$. □

Although moving to an HSG from a hitting set may seem counter-intuitive, it is easier to work with HSGs for some arguments. A canonical example for this is when we have a hitting set H for a class $\mathcal{C}$, and we wish to find a hitting set for the class $\mathcal{C}^2 := \{(p \cdot q) : p, q \in \mathcal{C}\}$. Arguably the simplest way to do this is to obtain an HSG $\mathbf{g}$ from H, and the same HSG then works for fg.

As we will soon see, it is sometimes easier to design a *family* of candidate HSGs for a class $\mathcal{C}$ such that for any nonzero polynomial in $\mathcal{C}$ at least one of the HSGs works. The following lemma gives us an easy way of obtaining a single HSG and a corresponding hitting set, in such cases.

Lemma 2.13 (Combining HSGs)**.** *Let $\mathcal{C}$ be a class of n-variate, degree d polynomials and suppose $\mathcal{G} = \{\mathbf{g}_1(\mathbf{t}_{[k]}), \ldots, \mathbf{g}_s(\mathbf{t}_{[k]})\}$ is a family of k-variate, degree r polynomial maps, such that for every nonzero $C \in \mathcal{C}$, there exists $j \in [s]$ for which $C(\mathbf{g}_i(\mathbf{t}))$ is nonzero. Then for a fresh variable u, there exists a $(k+1)$-variate HSG $\tilde{\mathbf{g}}(\mathbf{t}, u)$ of degree at most $\max(r, s)$ for $\mathcal{C}$. Furthermore, if $\mathcal{G}$ was explicit then $\tilde{\mathbf{g}}$ is also explicit.*

Proof. Again, the proof will use interpolation. Let $\mathbf{g}_j(\mathbf{t})$ be the tuple $(\mathbf{g}_{j,1}(\mathbf{t}), \mathbf{g}_{j,2}(\mathbf{t}), \ldots, \mathbf{g}_{j,n}(\mathbf{t}))$ for every $j \in [s]$. For each $i \in [n]$, we will define $\tilde{\mathbf{g}}(\mathbf{t}, u)$ to be the unique polynomial of degree at most s in u, such that $\tilde{\mathbf{g}}(\mathbf{t}, \alpha) = \mathbf{g}_\alpha(\mathbf{t})$ for all $\alpha \in [s]$. It is now easy to see that $\tilde{\mathbf{g}}(\mathbf{t})$ is an (explicit) HSG of degree $\max(s, r)$ for $\mathcal{C}$. □

Note that in the above lemma, we also get a hitting set of size at most $(srd)^{O(k+1)}$ using Lemma 2.11.

2.3.2 Designing Hitting Set Generators

We now list some explicit HSGs that we refer to in the thesis. On our way, we will also describe a commonly used template for designing HSGs.

Kronecker map. Suppose we have a polynomial map $\varphi_{(n,d)}(t)$ such that for any n-variate, degree-d polynomial $f(\mathbf{x})$, we have that (i) for any monomial $m \in \text{supp}(f)$, $\varphi_{(n,d)}(m)$ is a monomial in t, and (ii) for any two *distinct* monomials m_1, m_2, $\varphi_{(n,d)}(m_1) \neq \varphi_{(n,d)}(m_2)$. In other words, $\varphi_{(n,d)}$ maps distinct monomials to distinct monomials. Observe that such a map will be an HSG for the class of all n-variate, degree d polynomials. One way to achieve this is to use the following map that maps every variable in $\mathbf{x}$ to monomials in t.

Definition 2.14 (Kronecker map). *For any $n, d \in \mathbb{N}$, the* Kronecker map *for n-variate, degree d polynomials is given as follows.*
$\kappa_{(n,d)}(x_1) := t$, $\kappa_{(n,d)}(x_2) := t^{(d+1)}, \cdots, \kappa_{(n,d)}(x_i) := t^{(d+1)^{(i-1)}}, \cdots, \kappa_{(n,d)}(x_n) := t^{(d+1)^{(n-1)}}$. ◊

Note that $\kappa_{(n,d)}$ basically views *the exponent vectors* of degree d monomials as numbers in base $(d+1)$, and maps every $\mathbf{x}^{\mathbf{e}}$ to $t^{\mathrm{val}(\mathbf{e})}$, where $\mathrm{val}(\mathbf{e})$ is the value of the "$(d+1)$-ary number" given by $\mathbf{e}$, and therefore $\kappa_{(n,d)}$ maps distinct monomials in $\mathbf{x}$ to distinct monomials in t. However, despite being an HSG for all n-variate, degree d polynomials, the Kronecker map is very unwieldy as it has degree that is exponential in n (which gives another explicit hitting set of size $d^{O(n)}$).

Isolating Assignments

We saw that the Kronecker map just *separates all monomials* by mapping each monomial to the value of its exponent vector, but requires very large degree. There are several (equally easy) ways to see that any map that *separates all n-variate, degree d monomials* will never result in an "efficient HSG". Nevertheless, we can ask for an appropriate weakening of the above requirement, such that it yields an efficient HSG for some non-trivial class.

First, let us see how we can build polynomial maps from what are called *weight assignments* to variables.

Definition 2.15 (Weight assignment on monomials). *For any $n \in \mathbb{N}$, a function* $\mathrm{wt} : [n] \to \mathbb{N}$ *can be seen as a* weight assignment on a set of n-variate monomials *by setting* $\mathrm{wt}(x_i) = \mathrm{wt}(i)$ *and* $\mathrm{wt}(\mathbf{x}^{\mathbf{e}}) = \sum_{i \in [n]} (e_i \cdot \mathrm{wt}(x_i))$. ◊

Let us now define a weakening of the requirement of *separating all monomials* that we saw in the Kronecker map.

Definition 2.16 (Isolating weight assignment). *Let $\mathcal{M}$ be a set of n-variate monomials and let* wt *be a weight assignment on $\mathcal{M}$. Then we say that* wt isolates $\mathcal{M}$ *if there is a* unique minimum weight monomial. *That is there exists an $m \in \mathcal{M}$ such that for any $m' \in \mathcal{M}$, if $m' \neq m$ then* $\mathrm{wt}(m) < \mathrm{wt}(m')$.
Similarly, we say that wt isolates $f(\mathbf{x})$, *if* wt *isolates* $\mathrm{supp}(f)$. ◊

As it turns out, in most cases it is impossible to design a single *explicit* assignment even with this weaker property. We will therefore allow for a family of assignments that fulfil our purpose.

Definition 2.17 (Family of isolating weight assignments). *For a class of polynomials $\mathcal{C}$ and a family of weight assignments $W = \{w_1, \ldots, w_r\}$, we say that W* isolates $\mathcal{C}$ *if for all $f \in \mathcal{C}$, there exists an $i \in [r]$ such that w_i isolates f.* ◊

Let us now return to our goal of designing potential HSGs, and see how families of isolating weight assignments naturally give polynomial maps.

Definition 2.18 (Substitution map of an assignment). *Let $w : [n] \to \mathbb{N}$ be a weight assignment. Then the* substitution given by w, *say φ_w, is the map given by $\varphi_w(x_i) = t^{w(i)}$ for all $i \in [n]$. Note that $\varphi_w(\mathbf{x}^{\mathbf{e}}) = t^{w(\mathbf{x}^{\mathbf{e}})}$ for all $\mathbf{e} \in \mathbb{N}^n$, and the degree of φ_w is just $\max_{i \in [n]} w(i)$.* $\Diamond$

We are now ready to see how *isolating families* suffice for designing HSGs, despite the significantly weaker property (compared to *separation*).

Lemma 2.19. *Let $\mathcal{C}$ be a class of n-variate polynomials and suppose $W = \{w_1, \ldots, w_r\}$ is a family of weight assignments that isolates $\mathcal{C}$. Then for the substitution maps $\varphi_1, \ldots, \varphi_r$, each given by $\varphi_j(x_i) = t^{w_j(x_i)}$, we have that for any* nonzero *$f \in \mathcal{C}$, there exists a $j \in [r]$ such that $\varphi_j(f) \neq 0$.*

Proof. Suppose $0 \neq f \in \mathcal{C}$; hence there exists a w_j that isolates $\operatorname{supp}(f)$. That is, under the assignment w_j, there is a unique minimum weight monomial $m \in \operatorname{supp}(f)$. Hence, for $f'(t) = f(\varphi_j(\mathbf{x}))$, $\operatorname{coeff}_{f'}(t^{\operatorname{wt}(m)}) = \operatorname{coeff}_f(m) \neq 0$, which tells us that $f' \neq 0$. $\square$

Note that this lemma yields an explicit HSG using Lemma 2.13 and Lemma 2.11.

HSG for sparse polynomials. We are now ready to define the following weight assignment (and the corresponding polynomial map) attributed to Klivans and Spielman [KS01], and Agrawal and Biswas [AB03], that combines the Kronecker map with a "hashing argument" to obtain an HSG for polynomials computed by depth-2 circuits (also called *sparse polynomials*).

Definition 2.20 ([KS01, AB03]). *Let $n, d, A \in \mathbb{N}$, and let $r = A^2 \cdot n \cdot (d+1)$. For the first r primes $p_1, \ldots, p_r$, the family $W(n, d, A)$ consists of r weight assignments $w_1, \ldots, w_r$; where for each $i \in [n]$ and $j \in [r]$, $w_i(j) = (d+1)^{(j-1)} \bmod p_i$.* $\Diamond$

Lemma 2.21 (Separating a set of monomials [KS01, AB03]). *For all large enough $n, d, A \in \mathbb{N}$, let $W(n, d, A) = \{w_1, \ldots, w_r\}$ be as defined in Definition 2.20. Then for any set $\mathcal{M}$ of n-variate monomials of total degree at most d with $|\mathcal{M}| \leq A$, there exists an $i \in [r]$ for which w_i assigns distinct weights to all monomials in $\mathcal{M}$.*

Let $\Sigma^{[s]}\Pi^{[d]}(\mathbf{x}_n)$ be the class of n-variate, degree d polynomials that have a depth-2 circuit of top fan-in *at most s. Then the family of polynomial maps $\Phi_{(n,d,s)}$ obtained from $W(n, d, s)$ as per Definition 2.18 gives an HSG for $\Sigma^s\Pi^{\leq d}(n)$ using Lemma 2.19.*

3 | Hitting Sets for Circuits with Restricted Parse Trees

The main object of study in this chapter is the non-commutative model of *Unique Parse Tree (UPT) circuits* introduced by Lagarde, Malod and Perifel [LMP19]. We construct quasipolynomial hitting sets for this model and its closely related variants, essentially by extending the scope of hitting sets for weaker models of non-commutative ABPs, and ROABPs[1].

3.1 Non-commutative and Unique Parse Tree (UPT) circuits

Nisan [Nis91] introduced the non-commutative model, specifically the non-commutative algebraic branching programs (ABP). In his seminal paper, he showed that the non-commutative versions of the determinant and permanent polynomials (among others) require exponential sized non-commutative ABPs to compute them. Limaye, Malod and Srinivasan [LMS16] extended Nisan's lower bound to non-commutative skew circuits, which are circuits where every multiplication gate has at most one child that is a non-leaf. Lagarde, Malod and Perifel [LMP19] initiated the study of non-commutative *unambiguous circuits*, or *Unique Parse Tree (UPT)* circuits.

A parse tree of a circuit is obtained by starting at the root, and at every $+$ gate choosing exactly one child, and at every $\times$ gate choosing all its children (formally defined in Definition 3.6). Informally, a parse tree of a circuit is basically a *certificate* of computation of a monomial in a circuit. Lagarde, Malod and Perifel [LMP19] introduced a subclass of non-commutative circuits called *Unique Parse Tree (UPT) circuits* or *unambiguous circuits* where all parse trees of the circuit have the same shape (formally defined in Definition 3.7). The class of non-commutative UPT circuits subsumes the class of non-commutative ABPs as any ABP can be expressed as a left-skew circuit.

Lagarde, Malod and Perifel [LMP19] extended the techniques of Nisan [Nis91] to give exponential lower bounds for UPT circuits. Later, a lower bound against the class of circuits with parse trees of not-too-many shapes (at most $2^{o(n)}$ shapes) was shown by Lagarde, Limaye and Srinivasan [LLS19].

In Figure 3.1, (a) is an example of a UPT circuit with (b) being the underlying parse tree

[1]The results in this chapter have appeared in [ST18].

shape; (c) is an example of a circuit with two distinct parse tree shapes.

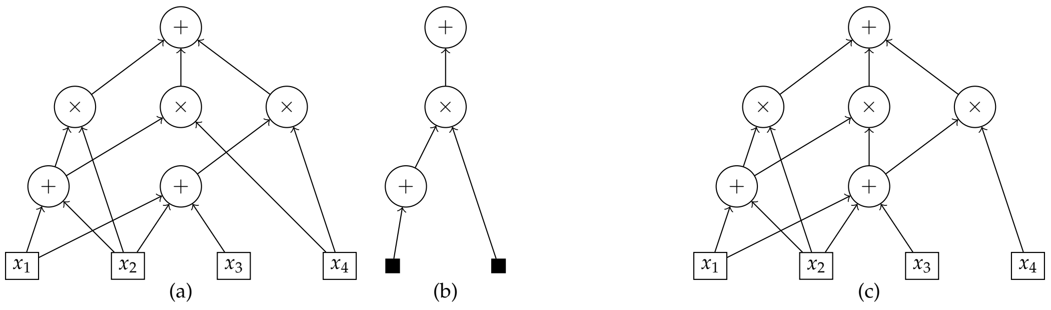


Figure 3.1: Examples of circuits with restricted parse trees

A related model of *set-depth-Δ formulas* was studied by Agrawal, Saha and Saxena [ASS13] that is a subclass of UPT circuits where the underlying parse trees are extremely regular[2]. Arvind and Raja [AR16] also studied lower bounds for various subclasses of commutative set-multilinear circuits. Some of the models they study include analogues of UPT and FewPT circuits. They also proved lower bounds for UPT and FewPT set-multilinear circuits, and for some other subclasses of set-multilinear circuits.

3.1.1 Polynomial identity testing

In the non-commutative world, Raz and Shpilka [RS05] gave the first deterministic polynomial time white-box PIT for the class of non-commutative ABPs, building on the characterization given by Nisan [Nis91]. Forbes and Shpilka [FS13] gave a quasipolynomial ($n^{O(\log n)}$) size hitting set for non-commutative ABPs. This was achieved by studying a natural commutative analogue of non-commutative ABPs, and this was the class of *Read-Once Oblivious Algebraic Branching Programs (ROABPs)* where the variables are read in a "known order".

The class of ROABPs is interesting in its own right owing to the connection with the "RL vs L" question. In fact, much of the hitting set constructions for ROABPs has been inspired by Nisan's [Nis92] pseudorandom generator for RL (which has seed length $O(\log^2 n)$). As mentioned earlier, Forbes and Shpilka gave a hitting set of size $n^{O(\log n)}$ for polynomial sized ROABPs when the order in which variables are read was known. Agrawal, Gurjar, Korwar and Saxena [AGKS15] presented a different hitting set for the class of ROABPs that did not need the knowledge of the order in which the variables were read. Subsequently, Gurjar, Korwar, Saxena and Thierauf [GKST17] studied polynomials that can be computed as a sum of constantly many ROABPs (of possibly different orders) and presented a polynomial time white-box PIT, and also a quasipolynomial time black-box PIT for this class.

Lagarde, Malod and Perifel [LMP19], besides presenting lower bounds against UPT circuits, also gave a polynomial time white-box PIT for this class. This was extended by Lagarde, Limaye and Srinivasan [LLS19] to a white-box algorithm for non-commutative circuits with

[2]the formula is levelled, and all nodes at a level have the same fan-in

constantly many parse tree shapes (analogous to the result of [GKST17]). The question of constructing black-box PITs was left open by them, and we answer this in our work.

3.1.2 Results in this chapter

Polynomial Identity Testing

Our main results are hitting sets for the class of polynomials computed by UPT circuits and related classes.

Theorem 3.1 (Hitting sets for UPT circuits)**.** *There is an explicit hitting set $\mathcal{H}_{d,n,s}$ of at most $(snd)^{O(\log d)}$ size for the class of degree d n-variate homogeneous non-commutative polynomials in $\mathbb{F}\langle \mathbf{x} \rangle$ that are computed by UPT circuits of size at most s.*

This result builds on the technique of *basis isolating weight assignments* first introduced by [AGKS15] for constructing hitting sets for ROABPs. Furthermore, we can also extend the hitting set to the class of non-commutative circuits that have *few shapes* (analogous to [GKST17]'s hitting set for sum of few ROABPs).

Theorem 3.2 (Hitting sets for circuits with few parse tree shapes)**.** *There is an explicit hitting set $\mathcal{H}_{d,n,s,k}$ of size at most $(s^{2^k} nd)^{O(\log d)}$ for the class of n-variate degree d homogeneous non-commutative polynomials in $\mathbb{F}\langle \mathbf{x} \rangle$ that are computed by non-commutative circuits of size at most s consisting of parse trees of at most k shapes.*

Both the above theorems are fully black-box in the sense that it is not required to know the underlying shape(s). For the case of non-commutative ABPs (and more generally, ROABPs in a known order), Gurjar, Korwar and Saxena [GKS17] presented a more efficient hitting set when the width of the ABP is small. For UPT circuits, there is a natural notion of *preimage-width* of a UPT circuit (formally defined in Definition 3.8) that corresponds to the notion of width of an ABP. We show an analogue of the hitting set of Gurjar, Korwar and Saxena for the class of UPT circuits of small *preimage-width* if the underlying shape of the parse trees is known.

Theorem 3.3 (Hitting sets for known-shape low-width UPT circuits)**.** *Let $\mathcal{C}_{n,d,T,w}$ be the class of n-variate degree d non-commutative polynomials that are computable by UPT circuits of preimage-width at most w and underlying parse-tree shape as T. Over any field of zero or large characteristic, there is an explicit hitting set $\mathcal{H}_{n,d,T,w}$ of size $w^{O(\log d)}\operatorname{poly}(nd)$ for $\mathcal{C}_{n,d,T,w}$.*

These hitting sets also translate to the natural commutative analogues: *UPT set-multilinear circuits* etc. (formally defined in Definition 3.25).

On our way towards extending the previously known hitting set constructions for ROABPs [AGKS15, GKS17, GKST17] to the setting of UPT circuits, we prove some structural results about UPT circuits and non-commutative ABPs. We shall now go over some of those results briefly.

Structural results

If f is a non-commutative polynomial of degree d and if $\sigma \in S_d$ is a permutation on d letters, we define the *shuffling* of f by σ (denoted by $\Delta_\sigma(f)$) as the natural operation of permuting each *word* of f according to σ.

The three PIT statements stated above begin with the following depth reduction statement about UPT circuits.

Theorem 3.4 (Depth reduction for UPT circuits)**.** *Let f be an n-variate degree d polynomial that is computable by a UPT circuit of preimage-width w. Then, there is some $\sigma \in S_d$ such that $\Delta_\sigma(f)$ can be computed by a UPT circuit of $O(\log d)$ depth and preimage-width at most $O(w^2)$.*

The above theorem implies that $\Delta_\sigma(f)$ is computable by an ABP of quasipolynomial size. We also show that this blow-up of quasipolynomial size is tight.

Theorem 3.5 (Separating UPT circuits and ABPs, under shuffling)**.** *There is an explicit n-variate degree d non-commutative polynomial f that is computable by UPT circuits of preimage-width $w = \mathrm{poly}(n, d)$ such that for every $\sigma \in S_d$, the polynomial $\Delta_\sigma(f)$ requires non-commutative ABPs of size $(nd)^{\Omega(\log nd)}$ to compute it.*

We also extend the lower bound of [LMP19] to give a polynomial computed by a *skew circuit* that requires exponential sized UPT circuits under any shuffling. Details are in Section 3.8. Additionally, we also compare the model of constant width UPT circuits with non-commutative formulas and ABPs, both with and without shufflings. We discuss those results in Section 3.11.

3.1.3 Proof ideas

As mentioned, the starting point of all these results is the depth reduction. From a result of Nisan [Nis91], the palindrome polynomial Pal_d[3] is known to require ABPs of size $2^{\Omega(d)}$ even though it can be computed by a polynomial sized UPT circuit. Therefore, Pal_d cannot be computed by a $\mathrm{poly}(d)$ sized circuit of depth $o(d/\log d)$. The key insight here is that even though Pal_d cannot be computed by small depth non-commutative circuits, a shuffling of the palindrome is

$$\sum_{w_1,\ldots,w_d \in [n]} x_{w_1} x_{w_1} x_{w_2} x_{w_2} \cdots x_{w_d} x_{w_d} = \prod_{i=1}^{d} (x_1 x_1 + \cdots + x_n x_n),$$

which is of course computable by even an $O(\log d)$ depth UPT formula. Hence we attempt to reduce the depth under a suitable shuffling (see Definition 3.11).

In order to establish the depth reduction (Theorem 3.4) we follow the strategy of Valiant, Skyum, Berkowitz and Rackoff [VSBR83] and Allender, Jiao, Mahajan and Vinay [AJMV98] but make use of the UPT structure (work with different *frontier nodes* and *gate quotients*) based

[3] A definition can be found in Section 3.8.1.

on the underlying shape of the parse trees. The key ideas in our proof of depth reduction were also used by Arvind and Raja [AR16] for a commutative analogue of UPT circuits[4].

This depth reduction immediately yields that there is a quasipolynomial sized ABP computing a shuffling of f. We show that this blow-up is tight (Theorem 3.5) by essentially following the proof of Hrubeš and Yehudayoff [HY16] to separate monotone ABPs and monotone circuits in the commutative world.

In order to obtain hitting sets for UPT circuits, one could potentially just use the fact that there is a quasipolynomial sized ABP computing a shuffling of f and just use the known hitting sets for non-commutative ABPs [FS13] to obtain a hitting set of $\text{poly}(ndw)^{O(\log^2 d)}$. However, we directly work with the UPT circuit and lift the technique of *basis isolating weight assignments* of Agrawal, Gurjar, Korwar and Saxena [AGKS15] to this more general setting to obtain Theorem 3.1. Theorem 3.3 is a straightforward generalization of the ideas of Gurjar, Korwar and Saxena [GKS17] once we observe that the depth reduction keeps the preimage-width small.

Theorem 3.2 essentially follows the same ideas as those in the work of Gurjar, Korwar, Saxena and Thierauf [GKST17]. The techniques of [GKST17] are general enough that once a circuit class has a *characterizing set of dependencies* and a *basis isolating weight assignment*, there is a natural method to lift the techniques to work with the sum of few elements from this class. [GKST17] use this for ROABPs and we use this for UPT circuits.

To summarize, once we obtain the depth reduction, much of the results in this chapter is a careful translation of prior work of [HY16], [AGKS15], [GKST17], [GKS17] to the setting of UPT (or FewPT) circuits. Consequently, this also generalises the hitting sets of [AGKS15, GKST17, GKS17] from ROABPs to *UPT (or FewPT) set-multilinear* circuits. Such a generalization was unknown prior to this work.

3.2 Background

We would sometimes shift between the commutative and the non-commutative computation; we therefore use $\mathbf{x}$ to talk about non-commutative variables, and $\mathbf{y}, \mathbf{z}$ for commuting variables.

We use $\mathbb{F}\langle \mathbf{x}_n \rangle$ to refer to the non-commutative ring of polynomials over $\mathbf{x} = \{\mathbf{x}\}$. For a parameter d, we use $\mathbb{F}\langle \mathbf{x} \rangle_{\deg = d}$ to refer to the set of polynomials in $\mathbb{F}\langle \mathbf{x} \rangle$ that are homogeneous and of degree d. Similarly, $\mathbb{F}\langle \mathbf{x} \rangle_{\deg \leq d}$ refers to the set of polynomials of degree at most d.

We now provide some essential background that is specific to the results discussed in this chapter.

3.2.1 Basic definitions

[4]This was pointed out to us after we had published a preprint of our work on arXiv.

UPT and FewPT circuits

Definition 3.6 (Parse trees). *A parse tree T of a circuit C is a tree obtained as follows:*

- *the root of C is the root of T,*
- *if $v \in T$ is a $\times$ gate, then all the children in C are the children of v in T in the same order,*
- *if $v \in T$ is a $+$ gate, then exactly one child of v in C is a child of v in T.*

The value of the parse tree T, denoted by $[T]$, is just the product of the leaf labels in T. $\Diamond$

Intuitively, a parse tree is a *certificate* that a monomial was produced in the computation of C (though it could potentially be cancelled by other parse trees computing the same monomial). Therefore, if f is the polynomial computed by C, then

$$f = \sum_{T \text{ is a parse tree}} [T].$$

Definition 3.7. *(UPT and FewPT circuits) A circuit C computing a homogeneous polynomial is said to be a* Unique Parse Tree (UPT) circuit *if all parse trees of C have the same shape (that is, they are identical except perhaps for the gate names).*

A circuit C that computes a homogeneous polynomial is said to be a FewPT(k) circuit if the parse trees of C have at most k distinct shapes. $\Diamond$

Definition 3.8 (Preimage-width). *Suppose C is a UPT circuit and say T is the shape of the underlying parse trees. For a node $\tau \in T$ and a gate $g \in C$, we shall say that g is a* preimage *of τ, denoted by $g \sim \tau$, if and only if there is some parse tree T' of C where the gate g appears in position τ.*

The preimage-width *of a UPT circuit C is the largest size of preimages of any node $\tau \in T$. That is,*

$$\text{preimage-width}(C) = \max_{\tau \in T} \left|\{g \in C \ : \ g \sim \tau\}\right|. \qquad \Diamond$$

It is clear that if C is a UPT circuit of preimage-width w computing a homogeneous degree d polynomial, then the size of C is at most dw. The preimage-width of a UPT circuit is a more useful measure to study than the size of the circuit. A simple concrete example of this is that the standard conversion of homogeneous ABPs to homogeneous circuits in fact yields UPT circuits. Furthermore, the width of the ABP is directly related to the preimage-width of the resulting UPT circuit.

Observation 3.9. *If f is computable by a width w homogeneous algebraic branching program, then f can be equivalently computed by UPT circuits of preimage-width w^2.* $\square$

Infixions

Definition 3.10 (Infixion). *For any $d_1, d_2 \geq 0$ and p satisfying $0 \leq p \leq d_2$, define $\times_p$ as the unique bilinear map $\times_p : \mathbb{F}\langle \mathbf{x} \rangle_{\deg = d_1} \times \mathbb{F}\langle \mathbf{x} \rangle_{\deg = d_2} \to \mathbb{F}\langle \mathbf{x} \rangle_{\deg = d_1 + d_2}$ that satisfies*

$$x_{w_1} \cdots x_{w_{d_1}} \times_p x_{v_1} \cdots x_{v_{d_2}} = x_{v_1} \cdots x_{v_p} x_{w_1} \cdots x_{w_{d_1}} x_{v_{p+1}} \cdots x_{v_{d_2}}.$$

*We shall refer to this operation as a p-*infixion [5]. $\Diamond$

For instance, the usual multiplication (or concatenation) operation is just a 0-infixion (or $\times_0$).

Shuffling of a polynomial

Definition 3.11 (Shuffling of a non-commutative polynomial). *Let $P_d(\mathbf{x}) \in \mathbb{F}\langle\mathbf{x}\rangle_{\deg=d}$ be a homogeneous degree d non-commutative polynomial. Given any permutation $\sigma \in S_d$ over d-letters, we can define the* shuffling of P_d via σ *as the unique linear map $\Delta_\sigma : \mathbb{F}\langle\mathbf{x}\rangle_{\deg=d} \to \mathbb{F}\langle\mathbf{x}\rangle_{\deg=d}$ that is obtained by linearly extending*

$$\Delta_\sigma(x_{w_1}\cdots x_{w_d}) = x_{w_{\sigma(1)}}\cdots x_{w_{\sigma(d)}}. \qquad \Diamond$$

3.2.2 Basic lemmas

Canonical UPT circuits, and types of gates

We shall say that a UPT circuit C with underlying parse tree shape T is *canonical* if for every gate $g \in C$ there is some node $\tau \in T$ such that every parse tree of C involving g has g only in position τ. In other words, every gate of the circuit has a unique type associated with it.

Lemma 3.12 ([LMP19]). *Suppose if $f \in \mathbb{F}\langle\mathbf{x}\rangle$ is a homogeneous, degree d, non-commutative polynomial computed by a non-commutative UPT circuit of preimage-width w. Then, f can be equivalently computed by a canonical UPT circuit of preimage-width w as well.*

For a canonical UPT circuit where the parse trees have shape T, we shall say that g has type τ if $\tau \in T$ is the unique node in T such that $g \sim \tau$.

Fix a $\tau \in T$ and let i be the number of leaves of the subtree rooted at τ, and let p be the number of leaves to the left of τ in the inorder traversal of T. We shall then say that τ (or a gate $g \in C$ of type τ) has *position-type* (i, p). The following lemma allows us to write the polynomial computed by the circuit as a small sum of p-infixions.

Lemma 3.13 ([LMP19]). *Let f be a polynomial computed by a canonical UPT circuit C of preimage-width w and say T is the shape of the underlying parse trees. If $\tau \in T$ with position-type (i, p), then we can write f as*

$$f(\mathbf{x}) = \sum_{r=1}^{w} g_r(\mathbf{x}) \times_p h_r(\mathbf{x}),$$

where $\deg g_r = i$ and $\deg h_r = \deg(f) - i$ for all $r = 1, \ldots, w$.

3.3 Depth reduction for UPT circuits

This section shall address Theorem 3.4, which we recall below.

[5] An infix is defined as "a formative element inserted in a word". E.g. The plural of 'spoonful' is 'spoonsful', obtained by adding the infix 's'.

Theorem 3.4 (Depth reduction for UPT circuits). *Let f be an n-variate degree d polynomial that is computable by a UPT circuit of preimage-width w. Then, there is some $\sigma \in S_d$ such that $\Delta_\sigma(f)$ can be computed by a UPT circuit of $O(\log d)$ depth and preimage-width at most $O(w^2)$.*

It was pointed out to us that a very similar depth reduction was also proved by Arvind and Raja [AR16]. They showed that a commutative UPT set-multilinear circuit can be depth-reduced to a corresponding quasi-polynomial sized $O(\log d)$ depth UPT set-multilinear formula via Hyafil's [Hya79] depth reduction. Using techniques similar to [VSBR83], one can obtain a polynomial sized circuit of depth $O(\log d)$ while maintaining the UPT property. Though this can be inferred from the results in [AR16], we state and prove it in the form needed for the non-commutative setting.

3.3.1 UPT infixion-circuits

To prove the depth reduction, we will move to an intermediate model of *UPT infixion-circuits*.

Definition 3.14 (UPT infixion-circuits). *The class of* UPT infixion-circuits *is a generalization of homogeneous non-commutative circuits in that the internal gates are $+$ gates and $\times_p$ gates instead of the usual $+$ and $\times$ gates. We shall also say that the circuit is* semi-unbounded *if all $\times_p$ gates have fan-in bounded by 2 (with no restriction on $+$ gates).*

A parse tree *for an infixion-circuit is similar to parse trees in a general non-commutative circuit but the internal nodes of the parse tree are labelled by $+$ and $\times_p$ (with the p specified at each gate).*

We shall say that an infixion-circuit C is UPT if every parse tree is of the same shape. That is, two parse trees in C can differ only in the gate names. $\diamondsuit$

To prove Theorem 3.4, we shall first depth reduce the circuit to obtain an infixion-circuit computing f of $O(\log d)$ depth. Then, we will convert that to a UPT circuit that computes a shuffling of f.

Lemma 3.15 (Depth reducing to infixion-circuits). *Let $f \in \mathbb{F}\langle \mathbf{x} \rangle$ be a homogeneous degree d polynomial that is computable by a UPT circuit of preimage-width s. Then, f can be equivalently be computed by a semi-unbounded UPT infixion-circuit of preimage-width $O(s^2)$ and depth $O(\log d)$.*

Proof. Let C be the UPT circuit computing $f(x_1, \ldots, x_n)$ and say T is the shape of the parse trees of C. For any node $\tau \in T$, let $\mathcal{F}_\tau$ be the set of all gates in C whose position in T is τ. For two gates $u, v \in C$, we shall say that $u \succeq v$ if the place of u in T is an ancestor of the place of v in T. We shall abuse notation and use $u \succeq \tau$ to mean that u's position in T is an ancestor of $\tau \in T$. For a gate $u \in C$, let $[u]$ refer to the polynomial computed at that gate. Similar to [VSBR83, AJMV98], we define inductively the following notion of a *gate quotient* for any pair of gates $u, v \in C$:

$$[u : v] = \begin{cases} 0 & \text{if } u \not\succeq v, \\ 1 & \text{if } u = v, \\ [u_1 : v] + [u_2 : v] & \text{if } u = u_1 + u_2, \\ [u_1 : v] \cdot [u_2] & \text{if } u = u_1 \times u_2 \text{ and } u_1 \succeq v, \\ [u_1] \cdot [u_2 : v] & \text{if } u = u_1 \times u_2 \text{ and } u_2 \succeq v. \end{cases}$$

Claim 3.16. *For any $u \in C$, if $\tau \in T$ such that $u \succeq \tau$, then*

$$[u] = \sum_{\substack{w \in C \\ w \sim \tau}} [w] \times_p [u : w] \tag{3.17}$$

for a suitable p depending just on τ and the type of u. Furthermore, suppose $u, v \in C$ with v being a multiplication gate and if $\tau \in T$ such that $u \succeq \tau \succeq v$ then

$$[u : v] = \sum_{\substack{w \in C \\ w \sim \tau}} [w : v] \times_p [u : w]. \tag{3.18}$$

for a suitable p depending just on τ and the type of u and v.

We'll defer this proof to later and first finish the proof of Lemma 3.15. With (3.17) and (3.18), we can construct the infixion-circuit C' for f just as in [VSBR83, AJMV98]. The circuit C' would have gates computing each $[u]$ and $[u : v]$ for nodes $u, v \in C$ with $u \succeq v$ and v being a multiplication gate. The wirings in C' is built by appropriate applications of (3.17) and (3.18).

Let $u \in C$ and say $\deg[u] = d_u$. The plan would be to set up the computation in C' so that using an $O(1)$ depth computation, we can compute $[u]$ using gates whose degrees are a constant factor smaller than d_u. Consider any parse tree rooted at u, and starting from u follow the *higher degree* child. Let τ be the last point on the path with degree $\geq d_u/2$ (degree of its children will be $< d_u/2$). Applying (3.17),

$$\begin{aligned} [u] &= \sum_{w \sim \tau} [w] \times_p [u : w] \\ &= \sum_{w \sim \tau} ([w_1] \times [w_2]) \times_p [u : w] && \text{where } w = w_1 \times w_2. \end{aligned}$$

Now observe that each of the terms on the RHS, $[u : w], [w_1], [w_2]$ have degree at most $d_u/2$, as we wanted. Furthermore, all coordinates of the tuple $([u : w], [w_1], [w_2])$ are all of the same *type* as we run over all $w \sim \tau$.

We now need to show how to compute $[u : v]$ for a pair $u \succ v$. Say $\deg[u] = d_u$ and $\deg[v] = d_v$. For this, start with some parse tree rooted at u and walk down the path leading to the place of v, and let τ be the last point on this path such that $\deg \tau \geq \frac{d_u+d_v}{2}$. Using (3.18),

$$\begin{aligned}
[u:v] &= \sum_{w\sim\tau}[w:v] \times_p [u:w] \\
&= \sum_{w\sim\tau}([w_1] \times [w_2:v]) \times_p [u:w]
\end{aligned}$$

where $w = w_1 \times w_2$ and $w_2 \succeq v$ (the other possibility is identical). By the choice of τ, we have $\deg[u:w], \deg[w_2:v] \leq \frac{d_u-d_v}{2}$. However, the best bound we can give on $\deg[w_1]$ is $d_u - d_v$. Nevertheless, we can apply (3.17) again on $[w_1]$ by finding a suitable $\tau' \prec w_1$ satisfying $\deg \tau' \geq \frac{\deg w_1}{2}$ and write

$$\begin{aligned}
[u:v] &= \sum_{w\sim\tau}([w_1] \times [w_2:v]) \times_p [u:w] \\
&= \sum_{w\sim\tau}\left(\left(\sum_{w'\sim\tau'}[w'] \times_{p'} [w_1:w']\right) \times [w_2:v]\right) \times_p [u:w] \\
&= \sum_{w\sim\tau}\sum_{w'\sim\tau'}\left(\left(\left([w_1'] \times [w_2']\right) \times_{p'} [w_1:w']\right) \times [w_2:v]\right) \times_p [u:w]
\end{aligned}$$

By the choice of τ and τ', each of the *factors* on the RHS have degree at most $\frac{(d_u-d_v)}{2}$ as we wanted. Furthermore, once again, all of the summands consists of similarly typed factors.

This naturally yields an infixion-circuit computing f of depth $O(\log d)$ and size $\text{poly}(s)$. Since all summands consist of similarly typed factors, it follows that the circuit is UPT as well. $\square$

Proof of Claim 3.16. The proof is by induction. As a base case, suppose $u \sim \tau$. Then, $[u]$ is just the sum of the values of parse trees. Some of the parse trees use u. Of all nodes $w \in C$ such that $w \sim \tau$, only $[u:u] = 1$ and every other $[u:w] = 0$. Therefore, clearly $[u] = \sum_{w\sim\tau}[w] \cdot [u:w]$.

Now suppose $u \succ \tau$ and say we already know that $[u'] = \sum_{w\sim\tau}[w] \times_p [u':w]$ for every $u \succ u' \succeq \tau$. If $u = u_1 + u_2$, then

$$\begin{aligned}
[u] &= [u_1] + [u_2] \\
&= \left(\sum_{w\sim\tau}[w] \times_p [u_1:w]\right) + \left(\sum_{w\sim\tau}[w] \times_p [u_2:w]\right) \\
&= \sum_{w\sim\tau}[w] \times_p ([u_1:w] + [u_2:w]) \\
&= \sum_{w\sim\tau}[w] \times_p [u:w].
\end{aligned}$$

Similarly, suppose $[u] = [u_1] \times [u_2]$. We have two cases depending on whether $u_1 \succeq \tau$ or $u_2 \succeq \tau$.

If $u_1 \succeq \tau$, then

$$\begin{aligned}[u] &= [u_1] \times [u_2] \\ &= \left(\sum_{w \sim \tau} [w] \times_p [u_1 : w]\right) \times [u_2] \\ &= \sum_{w \sim \tau} [w] \times_p ([u_1 : w] \times [u_2]) \\ &= \sum_{w \sim \tau} [w] \times_p [u : w].\end{aligned}$$

If $u_2 \succeq \tau$, then

$$\begin{aligned}[u] &= [u_1] \times [u_2] \\ &= [u_1] \times \left(\sum_{w \sim \tau} [w] \times_p [u_2 : w]\right) \\ &= \sum_{w \sim \tau} [w] \times_{p+\deg u_1} ([u_1] \times [u_2 : w]) \\ &= \sum_{w \sim \tau} [w] \times_{p+d_1} [u : w].\end{aligned}$$

Essentially the same proof works for (3.18) as well. □

Lemma 3.19 (Infixion-circuits to circuits for a shuffling). *Let $f \in \mathbb{F}\langle \mathbf{x} \rangle$ be a homogeneous degree d polynomial that is computable by a UPT infixion-circuit C' of size s. Consider the circuit C'' obtained by replacing all $\times_p$ (infixion) gates in C' by $\times$ gates. Then, C'' computes $\Delta_\sigma(f)$ for some $\sigma \in S_d$.*

Proof. We shall prove this by induction. We need a slightly stronger inductive hypothesis which is that the choice of permutation σ depends only on the shape of the parse trees in C'.

Say u is the root of C'. Suppose u is a $+$ gate and say $u = u_1 + u_2 + \cdots + u_r$. If $u' = u'_1 + \cdots + u'_r$ is the resulting computation in C'' then by the inductive hypothesis, we know that there is a $\sigma \in S_d$ such that $[u'_i] = \Delta_\sigma([u_i])$. Therefore,

$$[u'] = \sum_{i=1}^{r} \Delta_\sigma([u_i]) = \Delta_\sigma([u]).$$

Suppose $u = u_1 \times_p u_2$ with $\deg[u_1] = d_1$ and $\deg[u_2] = d_2$. Say $u_1 = \sum_{\alpha \in [n]^{d_1}} a_\alpha x_\alpha$ and $\sum_{\beta \in [n]^{d_2}} b_\beta x_\beta$. Then, $[u] = \sum_{\alpha,\beta} a_\alpha b_\beta \cdot x_\alpha \times_p x_\beta$. If u', u'_1 and u'_2 is the resulting computation in C'', then

$$\begin{aligned}[u'] &= [u'_1] \times [u'_2] \\ &= \Delta_{\sigma_1}([u_1]) \times \Delta_{\sigma_2}([u_2]) && \text{for some } \sigma_1 \in S_{d_1}, \sigma_2 \in S_{d_2}, \\ &= \sum_{\alpha,\beta} a_\alpha b_\beta \cdot (\Delta_{\sigma_1}(x_\alpha) \times \Delta_{\sigma_2}(x_\beta)) \\ &= \sum_{\alpha,\beta} a_\alpha b_\beta \cdot \Delta_\sigma(x_\alpha \times_p x_\beta) && \text{for some } \sigma \in S_d, \\ &= \Delta_\sigma([u]) \end{aligned}$$ □

Together, Lemma 3.15 and Lemma 3.19 yield Theorem 3.4. □(Theorem 3.4)

The following corollary is immediate from the fact that any circuit of depth D and size s can be computed by a formula of size $s^{O(d)}$ and hence an ABP of size $s^{O(d)}$.

Corollary 3.20. *If $f \in \mathbb{F}\langle \mathbf{x} \rangle$ is a homogeneous degree d polynomial that is computable by a UPT circuit of size s, then there is some $\sigma \in S_d$ such that $\Delta_\sigma(f)$ is computable by a non-commutative algebraic branching program of size $s^{O(\log d)}$.*

Furthermore, the shuffling σ that permits this can also be efficiently computed given the underlying shape for the circuit computing f.

3.3.2 UPT circuits of constant width

For a UPT circuit C, we shall say that its *width* is w if for every node τ in the shape T, there are at most w gates of C that have type τ. The following observation is evident from the proof of the above depth reduction.

Observation 3.21. *If C is a UPT circuit of width w, then the depth reduced circuit C' as obtained in Theorem 3.4 has width $O(w^2)$.*

This observation would allow us to yield a more efficient hitting set for the class of *small width known shape* UPT circuits. Details are present in Section 3.9.2.

3.4 Separating ROABPs and UPT circuits

Theorem 3.5 (Separating UPT circuits and ABPs, under shuffling)**.** *There is an explicit n-variate degree d non-commutative polynomial f that is computable by UPT circuits of preimage-width $w = \mathrm{poly}(n,d)$ such that for every $\sigma \in S_d$, the polynomial $\Delta_\sigma(f)$ requires non-commutative ABPs of size $(nd)^{\Omega(\log nd)}$ to compute it.*

The polynomial and the proof technique described here were introduced by Hrubeš and Yehudayoff [HY16] to separate monotone circuits and monotone ABPs in the commutative regime. The polynomial described here is a non-commutative analogue of the polynomial used by [HY16]. Much of the proof is also the argument of [HY16] tailored to the non-commutative setting.

3.4.1 The polynomial

Let T_d denote the complete binary tree of depth d (with 2^d leaves) and let $D = 2^{d+1} - 1$ refer to the number of nodes in T_d. We shall say that a colouring $\gamma : T_d \to \mathbb{Z}_m$ is *legal* if for every node $u \in T$, if v and w are the children of u then $\gamma(u) = \gamma(v) + \gamma(w) \bmod m$.

Let $v_1, \ldots, v_D$ be the vertices of T_d listed in an *in-order* manner (left-subtree listed inductively, then the root, and then the right-subtree listed inductively). We now define the non-commutative polynomial $P_d(x_1, \ldots, x_m) \in \mathbb{F}\langle x_1, \ldots, x_m \rangle$ of degree $D = 2^{d+1} - 1$ as

$$P_d(x_1, \ldots, x_m) = \sum_{\substack{\gamma \in [m]^D \\ \gamma \text{ is legal}}} x_{\gamma(v_1)} x_{\gamma(v_2)} \cdots x_{\gamma(v_D)}. \tag{3.22}$$

Lemma 3.23 (Upper bound)**.** *For every $m, d > 0$, the polynomial $P_d(y_1, \ldots, y_m)$ can be computed by a non-commutative UPT circuit of size $O(m^2 d)$ and preimage-width $O(m^2)$.*

(Refer to Section 3.7 for a proof).

Theorem 3.24 (Lower bound)**.** *For every permutation $\sigma \in S_D$, any non-commutative ABP computing the polynomial $\Delta_\sigma(P_d)$ has width $m^{\Omega(d)}$.*

Hence for $d = \log m$, we have that $P_d(x_1, \ldots, x_m)$ is computable by a UPT circuit of size $O(m^2 \log m)$ but for every $\sigma \in S_D$ the above theorem tells us that $\Delta_\sigma(P_d)$ requires ABPs of width $m^{\Omega(\log m)}$ to compute it. The lower bound follows on exactly same lines as the [HY16]. A proof is present in Section 3.7.

3.5 Hitting sets for non-commutative models

Commutative brethren of non-commutative models

This reduction to an appropriate commutative case was used by Forbes and Shpilka [FS13] to reduce constructing hitting sets for non-commutative ABPs to hitting sets for commutative ROABPs (more precisely, to set-multilinear ABPs). They studied the image of the non-commutative polynomial under the map $\Psi : \mathbb{F}\langle \mathbf{x} \rangle_{\deg = d} \to \mathbb{F}[y_{1,1}, \ldots, y_{d,n}]$ which is the unique $\mathbb{F}$-linear map given by $\Psi : x_{w_1} \cdots x_{w_d} \mapsto y_{1,w_1} \cdots y_{d,w_d}$.

For the model of non-commutative UPT circuits, the appropriate commutative model is a restriction of set-multilinear circuits that we call UPT set-multilinear (UPT-SML) circuits.

Definition 3.25 (Set-multilinear circuits)**.** *Let $\mathbf{y} = \mathbf{y}_1 \sqcup \cdots \sqcup \mathbf{y}_d$ be a partition of the variables. A circuit C computing a polynomial $f \in \mathbb{F}[\mathbf{y}]$ is said to be a* set-multilinear circuit *with respect to the above partition if:*

- *each gate $g \in C$ is labelled by a subset $S_g \subseteq [d]$ and g computes a polynomial over variables $\bigcup_{i \in S_g} \mathbf{y}_i$ where every monomial of $[g]$ is divisible by exactly one variable in $\mathbf{y}_i$ for each $i \in S_g$,*
- *if g is a $+$ gate, then the subset that labels g also labels each of its children,*
- *if g is a $\times$ gate with g_1 and g_2 being its children, then the subsets S_{g_1} and S_{g_2} labelling g_1 and g_2 respectively is a partition of S_g. That is, $S_g = S_{g_1} \sqcup S_{g_2}$.* $\Diamond$

We shall say the circuit C is UPT set-multilinear *if every parse tree of C is of the same shape and identically labelled. That is, if g and g' are $\times$ gates labelled by a set $S \subseteq [d]$, and if $g = g_1 \times g_2$ with S_1 and S_2 labelling g_1 and g_2, then the children of g' are also labelled by S_1 and S_2 respectively.*

We shall say the set-multilinear circuit C is FewPT(k) set-multilinear *if the circuit consists of parse trees of at most k different shapes.*

A natural generalization that will be useful later is a *multi-output UPT set-multilinear* circuit, which is a UPT set-multilinear circuit that potentially has multiple output gates, which are all labelled with the same subset.

Forbes and Shpilka [FS13] showed that constructing hitting sets for these commutative models suffices for the corresponding non-commutative models by a simple reduction (details in Section 3.9.1). We shall therefore focus on these commutative models for the hitting set constructions. And since we have already seen that such circuits can be depth reduced[6]

[6]the shuffling just reorders the partition of the set-multilinear circuit

to $O(\log d)$ depth, it suffices to construct a hitting set for $O(\log d)$-depth UPT and FewPT set-multilinear circuits.

3.5.1 Preliminaries for PIT

Weight assignments and basis isolation

To construct hitting sets for ROABPs, Agrawal, Gurjar, Korwar and Saxena [AGKS15] defined the notion of *basis isolating weight assignments* for associated vector spaces of polynomials. The description presented here is an adaptation of the approach of [AGKS15] to set-multilinear circuits of small depth.

Definition 3.26 (Basis Isolating Weight Assignment (BIWA)). *A* weight assignment *is a function* $\mathrm{wt} : \mathbf{y} \to [M]^k$*, for some positive integer* M*, that can then be extended to all multilinear monomials over* $\mathbf{y}$ *via*

$$\mathrm{wt}\left(\prod_{i \in S} y_i\right) = \sum_{i \in S}^{n} \mathrm{wt}(y_i). \qquad \Diamond$$

Let V *be a vector space of polynomials in* $\mathbb{F}[\mathbf{y}]$*, which can also be thought of as a matrix with a generating set of polynomials listed out as rows (with each column being indexed by a monomial in* $\mathbf{y}$*).*

Such a weight assignment wt *is said to be a* basis isolating weight assignment *for* V *if there exists a basis of its column space, indexed by* $B \subseteq \mathrm{Mons}(\mathbf{y})$*, such that*

1. *if* $m_1, m_2 \in B$ *and* $m_1 \neq m_2$*, then* $\mathrm{wt}(m_1) \neq \mathrm{wt}(m_2)$*,*
2. *for every* $m \notin B$*,*

   $$V_m \in \mathrm{span}\left\{V_{m'} \;:\; m' \in B \,,\, \mathrm{wt}(m') \prec \mathrm{wt}(m)\right\}$$

   where by V_m *we mean the column of* V *indexed by the monomial* m *and* $\prec$ *is the lexicographic ordering on* $M^k \subset \mathbb{N}^k$*.*

Lemma 3.27 ([AGKS15]). *Let* V *be a vector space of polynomials in* $\mathbb{F}[\mathbf{y}]$ *and say* $f \in V$*. If* $\mathrm{wt} : \mathbf{y} \to [M]^k$ *is a BIWA for* V*, then if* $\mathbf{t} = \{t_1, \ldots, t_k\}$

$$f(y_1, \ldots, y_n) \neq 0 \Longleftrightarrow f(\mathbf{t}^{\mathrm{wt}(y_1)}, \cdots, \mathbf{t}^{\mathrm{wt}(y_n)}) \neq 0$$

(where $\mathbf{t}^{(\alpha_1,\ldots,\alpha_k)}$ *is short-hand for* $t_1^{\alpha_1} \cdots t_k^{\alpha_k}$ *).*

If $f \neq 0$ and $\deg(f) \leq d$, then $f(\mathbf{t}^{\mathrm{wt}(y_1)}, \ldots, \mathbf{t}^{\mathrm{wt}(y_n)})$ is a non-zero k-variate polynomial of degree at most dM. Hence, the polynomial identity lemma [Ore22, DL78, Zip79, Sch80] would present a $(dM+1)^k$ sized hitting set.

Definition 3.28 (Separating small sets of monomials). *Let* S *be an arbitrary set of monomials over* $\mathbf{y}$*. We shall say that a weight assignment* $\mathrm{wt} : \mathbf{y} \to \mathbb{N}$ separates S *if for every distinct* $m, m' \in S$ *we* *have* $\mathrm{wt}(m) \neq \mathrm{wt}(m')$. $\Diamond$

Lemma 3.29 ([AB03]). *Let S be an arbitrary set of r multilinear monomials of degree at most d over variables $\mathbf{y} = \{y_{ij} \,:\, i \in [d], j \in [n]\}$. For a prime p, let $w_p : \mathbf{y} \to \mathbb{N}$ be a weight assignment given by*

$$w_p(y_{i,j}) = 2^{(i-1)n+(j-1)} \bmod p.$$

Then for all but at most $\binom{r}{2} \cdot n^2$ primes p, the weight assignment w_p separates S.

BIWAs for subspaces and products

Agrawal, Gurjar, Korwar and Saxena [AGKS15] constructed BIWAs for polynomials computed by ROABPs. The following two lemmas are slight abstractions of the key ideas in [AGKS15], so that they can also be applied in our setting. For the sake of completeness, the proofs are provided in Observation 3.9.1.

Lemma 3.30 (BIWA for subspaces). *Say V is a vector space of polynomials and suppose* wt *is a BIWA for V. Then, if V' is a subspace of V, then* wt *is a BIWA for V' as well.*

Lemma 3.31 (BIWA for variable disjoint products). *Say $V_1 \subseteq \mathbb{F}[\mathbf{y}]$ and $V_2 \subseteq \mathbb{F}[\mathbf{z}]$ are two vector spaces of polynomials over disjoint sets of variables, and of dimension at most s. Suppose*

$$\mathrm{wt}_1 : \mathbf{y} \to \mathbb{N}^k$$
$$\mathrm{wt}_2 : \mathbf{z} \to \mathbb{N}^k$$

are BIWAs for V_1 and V_2 isolating bases B_1 and B_2 respectively. If $w : \mathbf{y} \cup \mathbf{z} \to \mathbb{N}$ is a weight assignment that separates *$B_1 \cdot B_2 = \{m_1 m_2 \,:\, m_1 \in B_1 \,,\, m_2 \in B_2\}$. Then the weight assignment defined by*

$$\begin{aligned}
&\mathrm{wt} : \mathbf{y} \cup \mathbf{z} \to \mathbb{N}^{k+1}\\
&\mathrm{wt} : y_i \mapsto (\mathrm{wt}_1(y_i), w(y_i)) \quad \textit{for all } y_i \in \mathbf{y},\\
&\mathrm{wt} : z_i \mapsto (\mathrm{wt}_2(z_i), w(z_i)) \quad \textit{for all } z_i \in \mathbf{z},
\end{aligned}$$

is a BIWA for $V = V_1 \cdot V_2 = \mathrm{span}\,\{f \cdot g \,:\, f \in V_1 \,,\, g \in V_2\}$.

3.5.2 Hitting sets for UPT set-multilinear circuits

Theorem 3.32 (Hitting sets for UPT set-multilinear circuits). *Let $\mathcal{C}$ be the class of n-variate degree d set-multilinear polynomials (with respect to $\mathbf{y} = \mathbf{y}_1 \sqcup \cdots \sqcup \mathbf{y}_d$) that are computable by UPT set-multilinear circuits of preimage-width w and depth r. Then, for $M = \left(\binom{w}{2} n^2 d + 1\right)^2$, the set*

$$\mathcal{H} = \left\{ (b_{11}, \ldots, b_{dn}) \;:\; \mathbf{p} \in [M]^r \,,\; a_k \in A \,,\; b_{ij} = \prod_{k=1}^{r+1} a_k^{2^{(i-1)n+(j-1)} \bmod p_i} \right\}$$

is a hitting set for $\mathcal{C}$ of size $\operatorname{poly}(ndw)^r$.

The proof of this theorem is obtained by constructing what is called a *basis isolating weight assignment* for polynomials simultaneously computed by a multi-output UPT-SML circuit, heavily borrowing from the ideas in [AGKS15].

Proof. Suppose $f(\mathbf{y})$ is a polynomial that is computable by a UPT set-multilinear circuit C with respect to $\mathbf{y} = \mathbf{y}_1 \sqcup \cdots \sqcup \mathbf{y}_d$ and say C is of preimage-width size w and depth r.

Since C is a UPT set-multilinear circuit, let T be the shape of the parse tree. For each $\tau \in T$, we define the vector space

$$V_\tau = \operatorname{span}\{[g] \ : \ g \in C\,,\ g \sim \tau\}\,.$$

The following claim relates the vector space corresponding to nodes in T to the vector spaces corresponding to the children.

Claim 3.33. *If $\tau \in T$ labels a $+$ gate and if τ' is the unique child of τ, then $V_\tau \subseteq V_{\tau'}$.*

If $\tau \in T$ labels a $\times$ gate and has children τ_1 and τ_2, then V_τ is a subspace of $V_{\tau_1} \cdot V_{\tau_2}$.

Proof. Suppose $\tau \in T$ labels a $+$ gate and say τ' is the unique child of τ in T. Pick an arbitrary $g \in C$ such that $g \sim \tau$. If $[g] = [g_1] + \cdots + [g_s]$, then each $g_i \sim \tau'$. Therefore, $[g_i] \in V_{\tau'}$ and $[g] = [g_1] + \cdots + [g_s]$ implies that $[g] \in V_{\tau'}$. Since the choice of g was an arbitrary gate of type τ, it follows that V_τ is a subspace of $V_{\tau'}$.

Say τ labels a $\times$ gate, and say τ_1 and τ_2 are the children of τ. Pick an arbitrary gate $g \in C$ with $g \sim \tau$. If $[g] = [g_1] \times [g_2]$ then $g_1 \sim \tau_1$ and $g_2 \sim \tau_2$. But that implies that $[g_1] \in V_{\tau_1}$ and $[g_2] \in V_{\tau_2}$ and therefore $[g] \in V_{\tau_1} \cdot V_{\tau_2}$. Once again, since the choice of g was arbitrary, we get V_τ is a subspace of $V_{\tau_1} \cdot V_{\tau_2}$. □ (Claim 3.33)

Define the *multiplication height* of any gate g, denoted by $|g|_\times$, as the largest number of $\times$ gates encountered on a path from g to a leaf. Starting with the leaves, we shall build towards a BIWA for V_{root}, which by Lemma 3.27 also yields a hitting set.

Let P be the set of the first $(dn^2\binom{w}{2} + 1)$ primes. For each $0 \le k \le r$ and $\mathbf{p} = (p_1, \ldots, p_k) \in P^k$, define the function

$$\Omega_{\mathbf{p}}^{(k)} : \mathbf{y} \to \mathbb{N}^{k+1}$$
$$\Omega_{\mathbf{p}}^{(k)} : y_{ij} \mapsto (j, 2^{(i-1)n+(j-1)} \bmod p_1, \ldots, 2^{(i-1)n+(j-1)} \bmod p_k).$$

The plan is to use $\Omega_{\mathbf{p}}^{(k)}$ to build BIWAs for each V_τ. For a $\tau \in T$ with $|\tau|_\times = k$, let $S_\tau \subseteq [d]$ be the subset of indices labelling τ. Define $\operatorname{wt}_{\mathbf{p}}^{(\tau)}$ to be the restriction of $\Omega_{\mathbf{p}}^{(k)}$ to $\cup_{i \in S_\tau} \mathbf{y}_i$:

$$\operatorname{wt}_{\mathbf{p}}^{(\tau)} : \bigcup_{i \in S_\tau} \mathbf{y}_i \to \mathbb{N}^{k+1}$$
$$\operatorname{wt}_{\mathbf{p}}^{(\tau)}(y_{ij}) = \Omega_{\mathbf{p}}^{(k)}(y_{ij}).$$

We shall prove, by induction, that for each $0 \leq k \leq r$ there is a $\mathbf{p} \in P^k$ such that for every $\tau \in T$ with $|\tau|_\times \leq k$, the weight assignment $\mathrm{wt}_{\mathbf{p}}^{(k)}$ is a BIWA for V_τ.

If τ was a leaf of T, then any such node just computes a variable. Clearly, $\mathrm{wt}_{\mathbf{p}}^{(\tau)} : (y_{ij}) \mapsto j$ is a BIWA as it gives distinct weights to all variables of a partition. Hence, $\mathrm{wt}_{\mathbf{p}}^{(\tau)}$ is a BIWA for all V_τ whenever τ is a leaf.

If τ is not a leaf but $|\tau|_\times = 0$, then neither τ nor its descendants are $\times$ gates. Hence, the subtree at τ has a unique leaf ℓ and all the nodes along this path are $+$ gates. By Claim 3.33, V_τ is a subspace of V_ℓ and hence, by Lemma 3.30, $\mathrm{wt}_{\mathbf{p}}^{(\tau)} = \mathrm{wt}_{\mathbf{p}}^{(\ell)}$ is a BIWA for V_τ. That finishes the base case of $k = 0$.

Suppose we have proved the claim up to $k-1$. Let T_k be the set of all nodes of multiplication height at most k that are $\times$ gates. By the inductive hypothesis, there exists $\mathbf{p} \in P^{k-1}$ such that $\mathrm{wt}_{\mathbf{p}}^{(\tau')}$ is BIWA for all $V_{\tau'}$ with $|\tau'|_\times < k$. Fix such a $\mathbf{p}$. For each $\tau \in T_k$, its children τ_1, τ_2 must have multiplication height at most $k-1$. Since C is set-multilinear, the subset of indices that label τ_1 and τ_2 must be disjoint. Say S_1 and S_2 are the subsets of indices labelling τ_1 and τ_2 respectively.

Hence, by Claim 3.33, V_τ is a subspace of $V_{\tau_1} \cdot V_{\tau_2}$. By our inductive hypothesis, we know that $\mathrm{wt}_{\mathbf{p}}^{(\tau_1)}$ and $\mathrm{wt}_{\mathbf{p}}^{(\tau_2)}$ are BIWAs for V_{τ_1} and V_{τ_2} respectively. Observe that $\Omega_{\mathbf{p}}^{(k-1)}$ restricted to the appropriate subset of variables is a refinement of the weight assignments $\mathrm{wt}_{\mathbf{p}}^{(\tau_1)}$ and $\mathrm{wt}_{\mathbf{p}}^{(\tau_2)}$ (as $|\tau_1|_\times$ or $|\tau_2|_\times$ could have been smaller than $k-1$). Nevertheless, if $\mathrm{wt}_{\mathbf{p}}^{(\tau_1)}$ and $\mathrm{wt}_{\mathbf{p}}^{(\tau_2)}$ are BIWAs for V_{τ_1} and V_{τ_2} respectively, then the following weight assignments

$$\mathrm{wt}_1 : \bigcup_{i \in S_1} \mathbf{y}_i \to \mathbb{N}^k \qquad\qquad \mathrm{wt}_2 : \bigcup_{i \in S_2} \mathbf{y}_i \to \mathbb{N}^k$$
$$\mathrm{wt}_1 : y_{ij} \mapsto \Omega_{\mathbf{p}}^{(k-1)}(y_{ij}) \qquad\qquad \mathrm{wt}_2 : y_{ij} \mapsto \Omega_{\mathbf{p}}^{(k-1)}(y_{ij})$$

are also BIWAs for V_{τ_1} and V_{τ_2} respectively. It follows from Lemma 3.31, Lemma 3.30 and Lemma 3.29, that besides perhaps $\binom{w}{2}n^2$ primes $p \in P$, the weight assignment defined by

$$\begin{aligned}
\mathrm{wt} &: \bigcup_{i \in S_1 \cup S_2} \mathbf{y}_i \to \mathbb{N}^{k+1} \\
\mathrm{wt}(y_{ij}) &= \begin{cases} (\mathrm{wt}_1(y_{ij}), 2^{(i-1)n+(j-1)} \bmod p) & \text{if } i \in S_1, \\ (\mathrm{wt}_2(y_{ij}), 2^{(i-1)n+(j-1)} \bmod p) & \text{if } i \in S_2, \end{cases} \\
&= (\Omega_{\mathbf{p}}^{(k-1)}(y_{ij}), 2^{in+j} \bmod p)
\end{aligned}$$

is a BIWA for V_τ. For different τs in T_k there may a different set of $\binom{w}{2}n^2$ primes that we should exclude. But since the set P of primes is at least $\binom{w}{2}n^2 d + 1$, there is a prime $p \in P$ for which $\mathrm{wt}(y_{ij}) = (\Omega_{\mathbf{p}}^{(k-1)}, 2^{(i-1)n+(j-1)} \bmod p)$ is a BIWA for every V_τ where $\tau \in T_k$. By extending $\mathbf{p}$ by p in the last coordinate, this shows that there is a $\mathbf{p}' \in P^k$ such that for each $\tau \in T_k$, the weight assignment $\mathrm{wt}_{\mathbf{p}'}^{(\tau)}$ is a BIWA for V_τ.

To complete the inductive step, we also need to prove the same for $\tau \in T$ that are $+$ gates with $|\tau|_\times = k$. Hence, there must be a $\times$ gate $\tau' \in T_k$ that is a descendant of τ such that the path from τ to τ' consists only of $+$ gates. Once again, this forces $\mathrm{wt}_{\mathbf{p}}^{(\tau)} = \mathrm{wt}_{\mathbf{p}}^{(\tau')}$ and V_τ is a subspace of $V_{\tau'}$. Hence, by Claim 3.33 and Lemma 3.30, it follows that $\mathrm{wt}_{\mathbf{p}}^{(\tau)} = \mathrm{wt}_{\mathbf{p}}^{(\tau')}$ is a BIWA for V_τ as well. And that completes the proof of the inductive step.

Hence, if f is a polynomial computed by a preimage-width w UPT set-multilinear circuit of depth r, $\Omega_{\mathbf{p}}^{(r)}$ is a BIWA for V_{root}. Furthermore, by the prime number theorem, we know that the $\left(\binom{w}{2}n^2d+1\right)$-th prime cannot be bigger than $\left(\binom{w}{2}n^2d+1\right)^2$. Hence, the constructed BIWA is in fact a map

$$\Omega_{\mathbf{p}}^{(r)} : \mathbf{y} \to [M]^{r+1}$$

where $M \leq \left(\binom{w}{2}n^2d+1\right)^2$. Therefore, by Lemma 3.27 and the polynomial identity lemma, if we pick a set $A \subseteq \mathbb{F}$ with $|A| > d \cdot \left(\binom{w}{2}n^2d+1\right)^2$, then

$$\mathcal{H} = \left\{ (b_{11}, \ldots, b_{dn}) \;:\; \mathbf{p} \in [M]^r \;,\; a_k \in A \;,\; b_{ij} = \prod_{k=1}^{r+1} a_k^{2^{(i-1)n+(j-1)} \bmod p_i} \right\}$$

is a hitting set for UPT set-multilinear circuits of preimage-width w and depth r, such that $|\mathcal{H}| = \mathrm{poly}\left((ndw)^r\right)$. □

3.5.3 Poly-sized hitting sets for constant width UPT circuits

Theorem 3.3 (Hitting sets for known-shape low-width UPT circuits)**.** *Let $\mathcal{C}_{n,d,T,w}$ be the class of n-variate degree d non-commutative polynomials that are computable by UPT circuits of preimage-width at most w and underlying parse-tree shape as T. Over any field of zero or large characteristic, there is an explicit hitting set $\mathcal{H}_{n,d,T,w}$ of size $w^{O(\log d)}\,\mathrm{poly}(nd)$ for $\mathcal{C}_{n,d,T,w}$.*

The proof is an easy extension of the ideas from [GKS17], the details of which are in Section 3.9.2.

3.6 FewPT circuits

In this section we describe the black-box identity test for FewPT(k) circuits. The following lemma from [LLS19] shows that this class is equivalent to polynomials computed by sum of k UPT circuits (of possibly different shapes).

3.6.1 Preliminaries

Lemma 3.34. ([LLS19, Lemma 16]) *Let $f(\mathbf{x})$ be a polynomial computed by FewPT(k) circuit of preimage-width w. Then f can be equivalently computed by a sum of k UPT circuits of preimage-width w each.*

Like in [LLS19], we'll refer to this class by $\mathrm{FewPT}(k)$. We shall further qualify this notation to use $\mathrm{FewPT}(k)(w)$ to denote the class of circuits that is a sum of k UPT circuits of preimage-width w.

From this lemma, we can just work with Σ^k UPT-SML circuits. The proof largely follows the ideas of Gurjar, Korwar, Saxena and Thierauf [GKST17][7].

Notation

Let $\mathbf{y} = \mathbf{y}_1 \sqcup \cdots \sqcup \mathbf{y}_d$ be a partition of the variables and let $S = \{s_1, \ldots, s_p\}$ be a subset of $[d]$. Define the set of variables $\mathbf{y}_S = \mathbf{y}_{s_1} \cup \cdots \cup \mathbf{y}_{s_p}$ and the set of monomials $\mathbf{y}^S = \mathbf{y}_{s_1} \times \cdots \times \mathbf{y}_{s_p}$. Also, define $\mathbf{y}_{-S} = \mathbf{y} \setminus \mathbf{y}_S$ and $\mathbf{y}^{-S} = \mathbf{y}^{[d]\setminus S}$.

Definition 3.35 (Coefficient operator)**.** *Given a set-multilinear polynomial $f = \sum_{m \in \mathbf{y}^{[d]}} \alpha_m m$ of degree d, for $S \subseteq [d]$ and a monomial $m \in \mathbf{y}^S$, define $\mathrm{coeff}_m : \mathbb{F}[\mathbf{y}] \to \mathbb{F}[\mathbf{y}_{-S}]$ to be as follows.*

$$\mathrm{coeff}_m(f) = \sum_{m' \in \mathbf{y}^{-S}} \alpha_{(m \cdot m')} m'$$

where $\alpha_{(m \cdot m')}$ is the coefficient of mm' in f. $\Diamond$

Lemma 3.36. *Let $\mathbf{y} = \mathbf{y}_1 \sqcup \ldots \sqcup \mathbf{y}_d$ be a partition and $f(\mathbf{y})$ be a set-multilinear polynomial (with respect to the above partition) computed by a* UPT-SML *circuit of preimage-width w and underlying parse-tree shape T. Suppose $g(\mathbf{y})$ is another set-multilinear polynomial (under the same partition) that* cannot *be computed by a* UPT-SML *circuit of preimage-width w with the same shape T.*

Then, there exists $S \subseteq [d]$ and $R \in \mathbb{F}[\mathbf{y}_S]^{1 \times w'}$, and $P, Q \in \mathbb{F}[y_{-S}]^{w' \times 1}$ with $w' \leq w^2$ such that:

- *For each $i \in [w']$, there is a monomial $m_i \in \mathbf{y}^S$ such that the i-th element of P and Q is $\mathrm{coeff}_{m_i}(f)$ and $\mathrm{coeff}_{m_i}(g)$ respectively,*
- *there is a vector $\Gamma \in \mathbb{F}^{1 \times w'}$ of support size at most $w+1$ such that $\Gamma P = 0$ and $\Gamma Q \neq 0$,*
- *the coefficient space of R is full-rank. That is, if we interpret R as a matrix over $\mathbb{F}$ by listing each of its w' entries as a column vector of coefficients, then this matrix has full column-rank.*
- *the vector of polynomials R is simultaneously computable by a* UPT-SML *circuit of preimage-width at most w'.*

This lemma is a fairly natural and straightforward generalization of [GKST17, Lemma 4.5] and a proof of this is provided in Section 3.10.

Lemma 3.37. *Suppose $f(\mathbf{y})$ is a non-zero polynomial computed by a Σ^k* $\mathrm{UPT\text{-}SML}(w)$ *circuit. Suppose $\mathrm{wt} : \mathbf{y} \to M^r$ is a weight assignment that satisfies the following properties:*

- *wt is a BIWA for spaces of polynomials simultaneously computed by* UPT-SML *circuits of preimage-width at most $w(w+1)$,*

[7] [GKST17] constructed hitting sets for sums of ROABPs and we use similar techniques for sums of UPT circuits. Roughly speaking, if we have a class $\mathcal{C}$ that has a *characterizing set of dependencies* for which we know how to construct BIWAs, then we can also construct hitting sets for $\Sigma^k\mathcal{C}$.

- *For any g in Σ^{k-1}* UPT-SML$(w(w+1))$*, the polynomial $g(\mathbf{y}+\mathbf{t}^{\mathrm{wt}}) \in \mathbb{F}(\mathbf{t})[\mathbf{y}]$ has a monomial with non-zero coefficient that depends on at most ℓ distinct variables in $\mathbf{y}$.*

Then, the polynomial $f(\mathbf{y}+\mathbf{t}^{\mathrm{wt}})$ has a monomial, depending on at most $\log(w(w+1))+\ell$ distinct variables in $\mathbf{y}$, with a non-zero coefficient.

This is essentially a restatement of [GKST17, Lemma 4.6, Lemma 4.8] and follows from their proof. Unravelling the recursion, we get the following corollary.

Corollary 3.38. *Let $f(\mathbf{y})$ be a non-zero polynomial that can computed by a Σ^k* UPT-SML(w) *circuit. Suppose $\mathrm{wt}: \mathbf{y} \to M^r$ is a BIWA for the class of polynomials simultaneously computed by* UPT-SML *circuits of preimage-width at most $w^{2^{O(k)}}$. Then, the polynomial $f(\mathbf{y}+\mathbf{t}^{\mathrm{wt}}) \in \mathbb{F}(\mathbf{t})[\mathbf{y}]$ has a monomial with a non-zero coefficient that depends on at most $2^{O(k)}\log w$ variables in $\mathbf{y}$.*

Once we are guaranteed to retain a monomial of small-support, we can construct a hitting set by enumerating over all possible supports and applying the polynomial identity lemma (or apply standard generators such as the Shpilka-Volkovich generator [SV15]). This completes the proof of Theorem 3.2, which we restate below for convenience.

Theorem 3.2 (Hitting sets for circuits with few parse tree shapes)**.** *There is an explicit hitting set $\mathcal{H}_{d,n,s,k}$ of size at most $(s^{2^k} nd)^{O(\log d)}$ for the class of n-variate degree d homogeneous non-commutative polynomials in $\mathbb{F}\langle \mathbf{x}\rangle$ that are computed by non-commutative circuits of size at most s consisting of parse trees of at most k shapes.*

3.7 Separating ABPs from UPT circuits

This section contains the proofs of the separation between ABPs and UPT circuits. Recall the definition of the polynomial P_d (of degree $D = 2^{d+1}-1$).

$$P_d(x_1,\ldots,x_m) = \sum_{\substack{\gamma\in[m]^D \\ \gamma \text{ is legal}}} x_{\gamma(v_1)}x_{\gamma(v_2)}\cdots x_{\gamma(v_D)}.$$

Upper bound

Lemma 3.23 (Upper bound)**.** *For every $m, d > 0$, the polynomial $P_d(y_1,\ldots,y_m)$ can be computed by a non-commutative UPT circuit of size $O(m^2 d)$ and preimage-width $O(m^2)$.*

Proof. Let $\mathcal{G}(d,\alpha)$ be the set of all legal colourings γ with v_{2^d} (root of T_d) satisfying $\gamma(v_{2^d}) = \alpha$. Now we define $P_{d,\alpha}(x_1,\ldots,x_m)$ as

$$P_{d,\alpha}(x_1,\ldots,x_m) = \sum_{\gamma\in\mathcal{G}(d,\alpha)} x_{\gamma(v_1)}x_{\gamma(v_2)}\cdots x_{\gamma(v_D)}.$$

Clearly, $P_d(x_1,\ldots,x_m) = \sum_{\alpha\in[m]} P_{d,\alpha}(x_1,\ldots,x_m)$. Therefore we can now recursively write

$$P_{d,\alpha}(x_1,\ldots,x_m) = \sum_{\beta\in[m]} P_{d-1,\beta}(x_1,\ldots,x_m)\cdot x_\alpha \cdot P_{d-1,(\alpha-_m\beta)}(x_1,\ldots,x_m), \tag{3.39}$$

where $\alpha -_m \beta = (\alpha - \beta) \bmod m$.

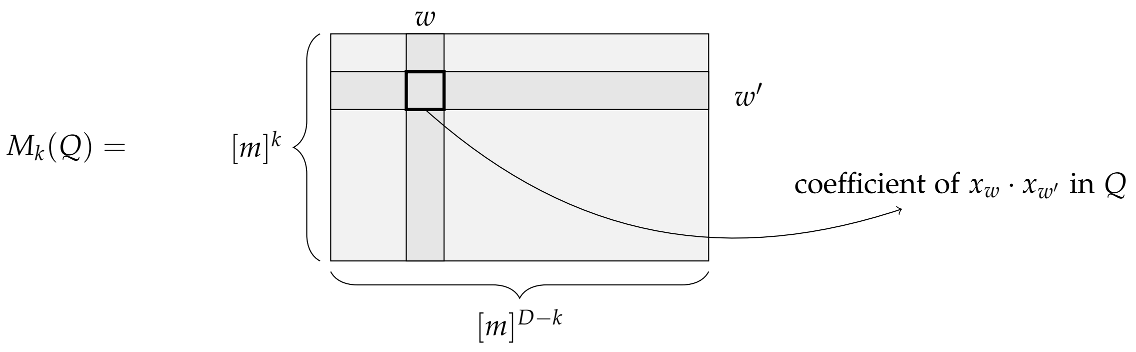
Now using (3.39) it is easy to see that if we have UPT circuits with $O(m^2k)$ gates *simultaneously* computing the polynomials $P_{k,\alpha}(x_1, \ldots, x_m)$ for any $k \le d-1$ and all $\alpha \in [m]$, then a UPT circuit with $O(m^2d)$ gates computing $P_d(x_1, \ldots, x_m)$ can be obtained and this follows directly by induction. Hence, repeated application of (3.39) yields a UPT circuit computing P_d of size $O(m^2d)$. It is also easy to see that the preimage-width of such a UPT circuit is at most $O(m^2)$, as there are no more than $O(m^2)$ computing polynomials of the same degree. □

Lower bound

As mentioned earlier, much of the lower bound argument is exactly along the lines of the proof of [HY16]. The modifications required from their proof are quite minor but we present the proof here for completeness.

Theorem 3.24 (Lower bound). *For every permutation $\sigma \in S_D$, any non-commutative ABP computing the polynomial $\Delta_\sigma(P_d)$ has width $m^{\Omega(d)}$.*

Proof. Let us fix some $\sigma \in S_D$ and let $Q(x_1, \ldots, x_m) = \Delta_\sigma(P_d)$. In order to show that Q requires ABPs of large width, it suffices to show that there exists some $0 \le k \le D$ for which the partial derivative matrix, given by



has rank at least $m^{\Omega(d)}$. We shall prove this by exhibiting an $r \times r$ identity matrix as a submatrix in $M_k(Q)$ with $r = m^{\Omega(d)}$. The k that we will work with would be the number whose binary expansion is $10101\cdots$. The relevance for this comes from the fact that the *edge boundary* of any subset $V_0 \subseteq T_d$ is with $|V_0| = k$ for such a k is reasonably large.

Definition 3.40 (Isoperimetric profile of graphs). *Given a graph $G = (V(G), E(G))$ and a subset of vertices $A \subseteq V(G)$, edge isoperimetric profile of G is given by the following function* eip(k) *defined by*

$$\mathrm{eip}_G(k) = \min\left\{ \left|E(A, \overline{A})\right| \ : \ A \subseteq V(G), |A| = k \right\},$$

where $E(A, \overline{A})$ is the set of edges with one end-point in A and the other outside. ◊

Lemma 3.41. *[HY16] If $k \le D$ is the number whose binary expansion is $1010\cdots$, then* $\mathrm{eip}_{T_d}(k) \ge \frac{d}{4}$.

The relevance for this would become apparent shortly, but let us proceed for now. If there is indeed an ABP for a shuffling of f, then the rows of $M_k(Q)$ is just a partial colouring of a subset $V_0 \subset T_d$ of size exactly k. Similarly, the columns of $M_k(Q)$ are partial colourings of $V_1 := T_d \setminus V_0$. Therefore $M_k(Q)_{(x_w, x_{w'})}$ is 1 only if the colouring of V_0 given by x_w and that of V_1 given by $x_{w'}$ together form a legal colouring of T_d. Hence the task of finding an $r \times r$ submatrix of $M_k(Q)$ reduces to finding colourings $C_1, C_2, \ldots, C_r$ of V_0 and colourings $C'_1, C'_2, \ldots, C'_r$ of V_1 such that the colouring $C_i \circ C'_j$ is legal if and only if $i = j$, for all $i, j \in [r]$.

We will need the notion of *pure nodes* (as defined by [HY16]).

Definition 3.42. (Pure *nodes*). *For $i \in \{0,1\}$, a non-leaf node v in V_i is called said to be* pure *if there is a path $\Pi = (v, v_1, v_2, \ldots, v_k)$ in T_d where v_k is a leaf that is a descendant of v, and $\Pi \cap V_i = \{v\}$.* $\Diamond$

There may be multiple witnesses v_k for the fact that v is a pure node. For each pure node, we shall assign one leaf arbitrarily as its *pure leaf*. It is easy to see that the pure leaves are distinct for each pure node.

Let the pure nodes in V_0 be P_0 and those in V_1 be P_1 and say $P := P_0 \cup P_1$. Let $\ell(P)$, $\ell(P_0)$ and $\ell(P_1)$ be the pure leaves of P, P_0 and P_1 respectively.

Lemma 3.43. ([HY16, Claim 11]) $|P| \geq \frac{|E(V_0, V_1)|}{4}$.

Without loss of generality, we may assume that P_0 is bigger than P_1 and the above lemma, in conjunction with Lemma 3.41, gives that $|P_0| \geq d/32$. We are now ready to define our colourings $C_1, \ldots, C_r$ and $C'_1, \ldots, C'_r$ for $r = m^{|P_0|} \geq m^{d/32}$.

Let L be the set of all leaves in T_d. For each $\mathbf{c}_i \in [m]^{|P_0|}$, define $\tilde{C}_i : T_d \to \mathbb{Z}_m$ obtained by assigning colour 1 to all leaves in $L \setminus \ell(P_0)$, assigning $\mathbf{c}_i$ to the leaves in $\ell(P_0)$ and extending it uniquely to the other vertices of T_d in order to make it legal. The partial colourings C_i and C'_i be the restriction of $\tilde{C}_i$ to V_0 and V_1 respectively.

Clearly, $C_i \circ C'_i = \tilde{C}_i$ and hence is a valid colouring. Now consider C_i and C'_j for $i \neq j$. There must exist some leaf $v \in \ell(P_0)$ that gets different colours in C_i and C_j and let u be the node in P_1 that v was a pure leaf of. We shall assume that u is *minimal* in the sense that any pure node $u' \in P_1$ that is a descendant has all its leaves identically coloured in C_i and C_j. But then, the colour of u in $\tilde{C}_i$ and in $\tilde{C}_j$ cannot be the same as exactly one leaf if u has a different colour in $\tilde{C}_i$ and $\tilde{C}_j$ respectively. This would then imply that C_i forces u to be given a colour different than what C'_j assigns and hence $C_i \circ C'_j$ is not legal.

Therefore, this shows that the matrix $M_k(Q)$ has an $r \times r$ identity submatrix with $r \geq m^{d/32}$. Therefore, any ABP computing Q must have width at least $m^{\Omega(d)}$. $\square$

3.8 Exponential lower bound under any shuffling

Here we give an explicit polynomial that has polynomial sized arithmetic circuits but requires exponential sized UPT circuits under any shuffling. A version of the hard polynomial appears in [LMP19]. They show that the polynomial requires exponential sized UPT circuits and that it is efficiently computable by what are known as *skew circuits* (see [LMP19] for a formal definition). Here we extend the lower bound and show that it applies to any *shuffling* of the polynomial.

3.8.1 The polynomial

The hard polynomial we discuss is called the *moving palindrome* which is a variant of the *palindrome* polynomial. The palindrome polynomial of degree d on n variables, as known, is defined as follows.

$$\mathrm{Pal}_d(x_1,\dots,x_n) := \sum_{w \in \{x_1,\dots,x_n\}^{d/2}} w \cdot w^R$$

where w^R denotes the reverse of the word w.
Using this definition, we define the $(n+1)$-variate moving palindrome of degree D as follows.

$$\mathrm{Pal}_D^{\mathrm{mov}}(x_1,\dots,x_n,z) := \sum_{0\le \ell \le D/2} z^\ell \cdot \mathrm{Pal}_{\frac{D}{2}}(x_1,\dots,x_n) \cdot z^{\frac{D}{2}-\ell}$$

3.8.2 The lower bound

Similar to the matrix M_k defined in Section 3.7 for a commutative polynomial, define a *partial derivative* matrix $M_{(i,p)}$ for a non-commutative polynomial g. Here the (w,w') entry of $M_{(i,p)}$ will be the coefficient of $w \times_p w'$ in g, where $\deg(w) = i$. We will show that $M_{(i,p)}$ for $\mathrm{Pal}_D^{\mathrm{mov}}$ has rank $n^{\Omega(D)}$ for a *range of types* (i,p), such that any UPT circuit computing any shuffling of $\mathrm{Pal}_D^{\mathrm{mov}}$ must admit at least one of those types. Then using the characterization from [LMP19], we will conclude the following theorem.

Theorem 3.44. *For any $\sigma \in S_D$, a UPT circuit computing $\Delta_\sigma(\mathrm{Pal}_D^{\mathrm{mov}})$ has $n^{\Omega(D)}$ gates.*

Proof. Let $2d$ be the degree of the palindrome, so that $\mathrm{Pal}_D^{\mathrm{mov}}$ has degree $D = 4d$. Also, let $P_\ell(\mathbf{x},z) = z^\ell\, \mathrm{Pal}_{2d}(\mathbf{x}) z^{2d-\ell}$. Therefore $\mathrm{Pal}_D^{\mathrm{mov}} = \sum_{\ell=0}^{2d} P_\ell(\mathbf{x},z) = f(\mathbf{x},z)$ (say). For P_ℓ, and for $\ell < j_1, j_2 \le 4d-\ell$, we will say that j_1 and j_2 are *dependent* with respect to P_ℓ if all monomials in P_ℓ contain the same variable in positions j_1 and j_2. It is easy to see that the criterion $j_1 + j_2 = 2(d+\ell)+1$ captures this relation. Define a *dependency* graph $G_\ell = (V, E_\ell)$ with $V = \{1,2,\dots,4d\}$ such that $(j_1,j_2) \in E_\ell$ if and only if j_1 and j_2 are dependent with respect to P_ℓ. Let $G = (V,E)$ with $E = \cup_\ell E_\ell$.

If $[4d] = V_0 \sqcup V_1$ is a partition, let us define a matrix $\tilde{M}_{V_0,V_1}(f)$ to be the one where rows and columns are indexed by a partial assignment to the positions V_0 and V_1 respectively.

Claim 3.45. *Let $[4d] = V_0 \sqcup V_1$ be a partition of the positions, and suppose that for some $\ell \in \{0,\dots,2d\}$ we have t edges in E_ℓ crossing the cut (V_0,V_1) in G_ℓ. Then,* $\mathrm{rank}\left(\tilde{M}_{V_0,V_1}(f)\right) \ge n^t$.

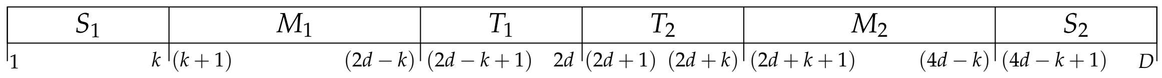
Proof. In the polynomial P_ℓ, let $Z_\ell \subseteq [4d]$ be the positions that are fixed to z. Consider the submatrix of $\tilde{M}_{V_0,V_1}$ where $V_0 \cap Z_\ell$ and $V_1 \cap Z_\ell$ are assigned to z. Observe that this submatrix is precisely $\tilde{M}_{V_0',V_1'}(P_\ell)$ where $V_0' = V_0 \cap \overline{Z_\ell}$ and $V_1' = V_1 \cap \overline{Z_\ell}$.

If we have t edges crossing the cut (V_0', V_1') (none of the cut edges can be adjacent on Z_ℓ), then we have a size t matching in (V_0', V_1'). This means that fixing the variables in their V_0' end-points uniquely fixes their V_1' end-points. Hence, it is clear that we have an $n^t \times n^t$ identity submatrix and hence that the rank of $\tilde{M}_{V_0,V_1}(f)$ is at least n^t. $\square$

The next claim shows that for any V_0 in a fairly wide range of sizes, there will always be some ℓ with G_ℓ exhibiting a large cut.

Claim 3.46. *For any set $V_0 \subseteq [4d]$ of size k with $\frac{d}{6} \leq k \leq \frac{d}{3}$, there is some $\ell \in \{0, \ldots, 2d\}$ such that $\Omega(d)$ edges in E_ℓ cross the cut (V_0, V_1).*

Proof. Let V_0 be a set of k positions with $k \leq \frac{d}{3}$. Let us partition the set of positions $V = \{1, 2, \ldots, 4d\}$ into blocks of size k,$(2d - 2k)$, and k, and k,$(2d-2k)$, and k, labelled S_1, M_1 and T_1, and T_2, M_2 and S_2, respectively. We illustrate such a partition below.



Now the possible choices for V_0 can be split into the following (possibly overlapping) cases:

1. $V_0 \cap T_1 \geq \frac{k}{8}$:
   Note that the degree of any vertex in T_1 is at least $(2d-k)$, and that every even (or odd) vertex in M_1 is connected to every odd (or even) vertex in S_2. Now $V_1 \cap M_1$ is at least $2d - k - (k - \frac{k}{8}) \geq 2(d-k)$. Total number of edges crossing (V_0, V_1) is therefore $\geq |(V_0 \cap T_1, V_1 \cap M_1)| \geq 2\left(\frac{1}{4} \times (d-k) \times \frac{k}{8}\right) = \Omega(dk)$. Therefore there exists an E_i that achieves the average $\Omega(k) = \Omega(d)$ edges crossing the cut (V_0, V_1).

2. $V_0 \cap S_1 \geq \frac{k}{4}$:
   Consider the neighbourhood of $V_0 \cup S_1$ due to E_0. All these positions are in T_1. If more than $\frac{k}{8}$ of them are in V_0 then case 1 applies. Else we get that $\geq \frac{k}{8}$ edges from E_0 cross (V_0, V_1).

3. $V_0 \cap M_1 \geq \frac{k}{4}$:
   Again, every even (or odd) position in M_1 is connected to every odd (or even) position in T_1, the degree of every position in M_1 is at least k, and $|V_1 \cap T_1| \geq \frac{k}{8}$. Therefore a total of $\Omega(k^2)$ edges cross (V_0, V_1), thereby again giving us that some E_i achieves $\Omega(d)$ edges crossing (V_0, V_1).

Since the other cases (with T_2, S_2, M_2) are symmetric to those discussed above, we can conclude the statement of the claim. $\square$

In order to complete the proof, we just need to show that any UPT circuit computing a homogeneous degree d polynomial, there will be a gate of position-type (i, p) with $\frac{d}{6} \leq i \leq \frac{d}{3}$.

Lemma 3.47. *For all $0 < \alpha < \frac{1}{2}$, any UPT circuit (with fan-in 2 $\times$ gates) computing a polynomial of degree D contains a gate computing a degree i polynomial for some $\alpha D \le i \le 2\alpha D$.*

Sketch of Proof. Let C be a UPT circuit computing a degree D polynomial with multiplication gates of fan-in 2. Starting from the root of C, choose an arbitrary child at every addition gate and the child computing a higher degree polynomial at every multiplication gate. As the degree never drops to a fraction less than half in any step, we eventually reach an appropriate gate. $\square$

Now Lemma 3.47 tells us that for any UPT circuit computing $\Delta_\sigma(\mathrm{Pal}_D^{\mathrm{mov}})$, will have a gate of position-type (i, p) with $\frac{D}{24} \le i \le \frac{D}{12}$. We can then apply Claim 3.46 and then Claim 3.45 to obtain an $n^{\Omega(D)}$ lower bound on the number of gates in C. $\square$

3.9 Hitting sets for UPT circuits

3.9.1 Commutative analogue of UPT circuits

Consider substitution map $\Phi : \{\mathbf{x}\} \to \mathbb{F}[y_{1,1}, \ldots, y_{d,n}]^{(d+1)\times(d+1)}$ given by

$$\Phi(x_i) = \begin{bmatrix} 0 & y_{1,i} & 0 & \ldots & 0 & 0 \\ 0 & 0 & y_{2,i} & \ldots & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 0 & y_{d,i} \\ 0 & 0 & 0 & \ldots & 0 & 0 \end{bmatrix}, \quad \text{for all } i \in [n].$$

To understand the effect of Φ on a homogeneous non-commutative polynomial $f(\mathbf{x})$ of degree d, define $\Psi : \mathbb{F}\langle \mathbf{x} \rangle_{\deg = d} \to \mathbb{F}[y_{1,1}, \ldots, y_{d,n}]$ as the unique $\mathbb{F}$-linear map given by
$\Psi : x_{w_1} \cdots x_{w_d} \mapsto y_{1,w_1} \cdots y_{d,w_d}$.

Lemma 3.48 ([FS13]). *Let $f = \sum_w a_w x_w \in \mathbb{F}\langle \mathbf{x} \rangle$ be a homogeneous degree d non-commutative polynomial. Then, f under the substitution map Φ (defined above) is given by*

$$f \circ \Phi = f(\Phi(x_1), \ldots, \Phi(x_n)) = \begin{bmatrix} 0 & \cdots & 0 & \Psi(f) \\ 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \end{bmatrix}_{(d+1)\times(d+1)}$$ $\square$

Similar to the above definition of Ψ, we define a *shifted* version of it called Ψ_a (for a parameter $a \in \mathbb{N}$) as $\Psi_a : x_{w_1} \cdots x_{w_d} \mapsto y_{a+1,w_1} \cdots y_{a+d,w_d}$.

Observation 3.49. *If $f \in \mathbb{F}\langle \mathbf{x} \rangle_{\deg = d_1}$ and $g \in \mathbb{F}\langle \mathbf{x} \rangle_{\deg = d_2}$, then for any $a \in \mathbb{N}$, we have $\Psi_a(f \cdot g) = \Psi_a(f) \cdot \Psi_{a+d_1}(g)$.*

In the case of [FS13], when f was computable by non-commutative ABPs, they showed that $\Psi(f)$ is computable by an ROABP. In our setting of non-commutative UPT circuits, the following is the commutative analogue.

Observation 3.50. *Let C be a UPT circuit computing a polynomial $f \in \mathbb{F}\langle \mathbf{x} \rangle$ of size s and depth r. Consider the commutative circuit C' where each leaf variable of type $(1, p)$ that is labelled by x_i is replaced by $y_{p+1,i}$. Then the circuit C' computes $\Psi(f)$ and is UPT and set-multilinear with respect to $\mathbf{y} = \mathbf{y}_1 \sqcup \cdots \sqcup \mathbf{y}_d$ where $\mathbf{y}_i = \{y_{i,j} \ : \ j \in [n]\}$.*

BIWAs for subspaces and products

Lemma 3.30 (BIWA for subspaces)**.** *Say V is a vector space of polynomials and suppose* wt *is a BIWA for V. Then, if V' is a subspace of V, then* wt *is a BIWA for V' as well.*

Proof. If B is a monomial basis of V that is isolated by wt, then the columns indexed by B span the column space of V' as well. Starting with the columns of V' indexed by B, pick a *minimum weight basis* B' according to wt, so that any column of V' that is outside B' is spanned by lower weight monomials in B'. By definition wt is a BIWA of V' isolating B', as all columns in B' get distinct weights and every column outside B' is spanned by lower weight columns in B'. $\square$

Lemma 3.31 (BIWA for variable disjoint products)**.** *Say $V_1 \subseteq \mathbb{F}[\mathbf{y}]$ and $V_2 \subseteq \mathbb{F}[\mathbf{z}]$ are two vector spaces of polynomials over disjoint sets of variables, and of dimension at most s. Suppose*

$$\begin{aligned}
&\mathrm{wt}_1 : \mathbf{y} \to \mathbb{N}^k \\
&\mathrm{wt}_2 : \mathbf{z} \to \mathbb{N}^k
\end{aligned}$$

are BIWAs for V_1 and V_2 isolating bases B_1 and B_2 respectively. If $w : \mathbf{y} \cup \mathbf{z} \to \mathbb{N}$ is a weight assignment that separates *$B_1 \cdot B_2 = \{m_1 m_2 \ : \ m_1 \in B_1 \,,\, m_2 \in B_2\}$. Then the weight assignment defined by*

$$\begin{aligned}
&\mathrm{wt} : \mathbf{y} \cup \mathbf{z} \to \mathbb{N}^{k+1} \\
&\mathrm{wt} : y_i \mapsto (\mathrm{wt}_1(y_i), w(y_i)) \quad \textit{for all } y_i \in \mathbf{y}, \\
&\mathrm{wt} : z_i \mapsto (\mathrm{wt}_2(z_i), w(z_i)) \quad \textit{for all } z_i \in \mathbf{z},
\end{aligned}$$

is a BIWA for $V = V_1 \cdot V_2 = \operatorname{span}\{f \cdot g \ : \ f \in V_1 \,,\, g \in V_2\}$.

Proof. Observe that by the definition of wt, $\mathrm{wt}(m_1 \cdot m_2) = (\mathrm{wt}_1(m_1) + \mathrm{wt}_2(m_2), w(m_1 \cdot m_2))$ for any $m \in \mathrm{Mons}(\mathbf{y})$ and $m' \in \mathrm{Mons}(\mathbf{z})$.

If V_1 and V_2 are expressed as matrices (with the generators listed as rows), then the matrix corresponding to V is just $V_1 \otimes V_2$, the tensor product. Let $B_1 = \{m_1, \ldots, m_r\}$ and $B_2 = \{m'_1, \ldots, m'_s\}$. We shall prove that the weight assignment wt is a BIWA that isolates the natural spanning set $B = B_1 \cdot B_2 = \left\{m_i m'_j \ : \ i \in [r] \,,\, j \in [s]\right\}$. Firstly, note that all the elements of B

have distinct weights due to the presence of the last coordinate from wt, which separates the rs monomials in $B_1 \cdot B_2$.

Now suppose $\tilde{m} = m \cdot m' \notin B$ for $m \in \mathrm{Mons}(\mathbf{y})$ and $m' \in \mathrm{Mons}(\mathbf{z})$ and say without loss of generality $m \notin B_1$. The column indexed by $\tilde{m}$ in $V_1 \cdot V_2$ is just the tensor product of the columns indexed by m in V_1 and the column indexed by m' in V_2. But since wt_1 is basis isolating for V_1, the column of V_1 indexed by m can be expressed as a linear combination of lower weight terms.

$$\begin{aligned} V_{1,m} &= \sum_{\mathrm{wt}_1(m_i) \prec \mathrm{wt}_1(m)} a_i \cdot V_{1,m_i} \\ \implies V_{\tilde{m}} = V_{1,m} \otimes V_{2,m'} &= \sum_{\mathrm{wt}_1(m_i) \prec \mathrm{wt}_1(m)} a_i \cdot (V_{1,m_i} \otimes V_{2,m'}) \\ &= \sum_{\mathrm{wt}_1(m_i) \prec \mathrm{wt}_1(m)} a_i \cdot V_{m_i m'} \end{aligned}$$

But notice that $\mathrm{wt}_1(m_i) \prec \mathrm{wt}_1(m)$ also implies that $\mathrm{wt}(m_i m') \prec \mathrm{wt}(mm')$. Therefore, (repeating this argument on m' if $m' \notin B_2$) we can write any column with index outside B as a linear combination of columns of smaller weight in B. Hence, wt is indeed a BIWA for V that isolates B. □

3.9.2 Constant width UPT circuits

In this subsection we prove the existence of a $\mathrm{poly}(n,d)$ hitting set for UPT circuits of constant preimage-width computing n-variate degree-d polynomials, when the *shape* of the circuit is known. The proof is an easy extension of the ideas of [GKS17] to the UPT-SML circuits regime. We will construct a univariate substitution map that preserves its nonzeroness and has degree $\mathrm{poly}(n,d)$, which will imply a hitting set naturally.

Say $\mathbf{y} = \mathbf{y}_1 \sqcup \cdots \sqcup \mathbf{y}_d$ and let $f(\mathbf{y})$ be an nd-variate degree d polynomial computable by a UPT-SML circuit (with respect to the above partition) of constant preimage-width. From Observation 3.21, we may assume that the circuit has depth $\log d$. We will need the following lemma for bivariate polynomials over *large* fields.

Lemma 3.51. ([GKS17, Lemma 3.2]) *Let $f(y_1,y_2) = \sum_{i=1}^{w} u_i(y_1)v_i(y_2)$ be a nonzero bivariate polynomial of degree d over $\mathbb{F}$. If $\mathrm{char}(\mathbb{F}) = 0$ or $\mathrm{char}(\mathbb{F}) > d$, then $f(t^w, t^{w-1} + t^w) \neq 0$.*

Suppose $f(\mathbf{y})$ is computable by a circuit C that has shape T. Define the set of variables $\mathbf{t} = \{t_\tau : \tau \in T\}$. We will begin by substituting $t_{\tau_i}^j$ for every y_{ij} where the leaf in C computing polynomials over $\mathbf{y}_i$ corresponds to τ_i in T. As long as we can, we will pick a multiplication gate τ that has its left and right children (say τ_L and τ_R) computing univariate polynomials in t_{τ_L} and t_{τ_R} respectively; and then substitute $t_{\tau_L} \leftarrow t_\tau^w$ and $t_{\tau_R} \leftarrow t_\tau^w + t_\tau^{w-1}$. Let us call this substitution Φ_τ.

Lemma 3.52. *Consider the above iterative process of substituting some of the $\mathbf{y}_i$'s by suitable polynomials in $\mathbf{t}$. Let $\tilde{\Phi}(f) = \tilde{f}(\mathbf{t}, \mathbf{y}) \neq 0$ be the polynomial just before applying the substitution Φ_τ. Then*

$\tilde{f}' = \Phi_\tau(\tilde{f}) := \tilde{f}(t_{\tau_L} \leftarrow t_\tau^w, t_{\tau_R} \leftarrow t_\tau^w + t_\tau^{w-1}) \neq 0.$

Proof. From (3.17), we have

$$\begin{aligned} f &= \sum_{u \sim \tau} [u] \cdot [\text{root} : u] \\ &= \sum_{u \sim \tau} [u_L] \cdot [u_R] \cdot [\text{root} : u], \\ \implies \tilde{\Phi}(f) &= \sum_{u \sim \tau} a_u(t_{\tau_L}) \cdot b_u(t_{\tau_R}) \cdot h_u(\mathbf{y}, \mathbf{t} \setminus t_{\tau_L}, t_{\tau_R}) \neq 0. \end{aligned}$$

We may treat $\tilde{\Phi}(f)$ as a bivariate polynomial in t_{τ_L}, t_{τ_R} over the field $\mathbb{F}(\mathbf{t} \setminus \{t_{\tau_L}, t_{\tau_R}\})$ and apply Lemma 3.51 to conclude that $\Phi_\tau(\tilde{\Phi}(f))$ will be nonzero if and only if $\tilde{\Phi}(f)$ was nonzero. $\square$

Now for every leaf node in T, create a sequence which we will call its *signature*, by walking down from the root to the leaf. Every time we pick the left child, we append L to the signature and every time we pick the right child, we append R. For $\tau \in T$, call the sequence $\text{sig}_\tau = (a_1\ a_2\ \cdots\ a_r)$. Let t be a fresh variable and τ_i be the node corresponding to $\mathbf{y}_i$. Define

$$\begin{aligned} &\Phi_L : t \mapsto t^w \quad , \quad \Phi_R : t \mapsto t^w + t^{w-1} \\ &\Psi : \mathbf{y} \to \mathbb{F}[t] \\ &\Psi : y_{ij} \mapsto \Phi_{a_1} \circ \Phi_{a_1} \circ \cdots \circ \Phi_{a_r}(t^j) \end{aligned}$$

where $(a_1 \cdots a_r) = \text{sig}_{\tau_i}$. Observe that the procedure described above essentially executes the substitution Ψ on $\mathbf{y}$. We can then infer from Lemma 3.52 that for any $f(\mathbf{y})$ computable by UPT-SML circuits, $f(\mathbf{y}) \neq 0 \iff f(\Psi(\mathbf{y})) \neq 0$. This gives us the following theorem.

Theorem 3.53. *Let $f(\mathbf{y})$ be a polynomial computed by an* UPT-SML *circuit of width w and depth r. Consider the following substitution $\Psi : \mathbf{y} \to \mathbb{F}[t]$ given by*

$$\Psi : y_{ij} \mapsto \Phi_{a_1} \circ \Phi_{a_2} \circ \cdots \circ \Phi_{a_r}(t^j),$$

where the signature *of the part $\mathbf{y}_i$ is $a_1 a_2 \cdots a_r$. Then $f(\mathbf{y})$ is non-zero if and only if $f(\Psi(\mathbf{y}))$ is non-zero.*

Now since the depth of the circuit is at most $O(\log d)$, if the width is constant, then the final degree of $f(\Psi(\mathbf{y}))$ is at most $O(nw^{O(\log d)})$, which is $\text{poly}(n, d)$ if $w = O(1)$. This finishes the proof of Theorem 3.3.

3.10 Hitting sets for FewPT circuits

We will need the following fact about coefficient operators (defined in Definition 3.35).

Observation 3.54 (Coefficients of UPT circuits are also UPT circuits)**.** *Suppose $f(\mathbf{y})$ is a homogeneous degree d polynomial that is computable by a UPT set-multilinear circuit with respect to*

$\mathbf{y} = \mathbf{y}_1 \sqcup \cdots \sqcup \mathbf{y}_d$ *of preimage-width* w*. If* $S \subseteq [d]$ *and* m *is any monomial in* $\mathbf{y}^S$*, then the polynomial* $\mathrm{coeff}_m(f)$ *can also be computed by a UPT set-multilinear circuit of preimage-width* w*.*

Sketch of Proof. Since the UPT-SML circuit C can be made canonical without loss of generality, we only need to set the corresponding leaves in $\mathbf{y}_S$ as 0 or 1 depending on whether the variable appears in m. □

The following is an analogue of [GKST17, Lemma 4.5].

Lemma 3.36. *Let* $\mathbf{y} = \mathbf{y}_1 \sqcup \ldots \sqcup \mathbf{y}_d$ *be a partition and* $f(\mathbf{y})$ *be a set-multilinear polynomial (with respect to the above partition) computed by a* UPT-SML *circuit of preimage-width* w *and underlying parse-tree shape* T*. Suppose* $g(\mathbf{y})$ *is another set-multilinear polynomial (under the same partition) that* cannot *be computed by a* UPT-SML *circuit of preimage-width* w *with the same shape* T*.*

Then, there exists $S \subseteq [d]$ *and* $R \in \mathbb{F}[\mathbf{y}_S]^{1\times w'}$*, and* $P, Q \in \mathbb{F}[y_{-S}]^{w'\times 1}$ *with* $w' \leq w^2$ *such that:*

- *For each* $i \in [w']$*, there is a monomial* $m_i \in \mathbf{y}^S$ *such that the i-th element of* P *and* Q *is* $\mathrm{coeff}_{m_i}(f)$ *and* $\mathrm{coeff}_{m_i}(g)$ *respectively,*
- *there is a vector* $\Gamma \in \mathbb{F}^{1\times w'}$ *of support size at most* $w+1$ *such that* $\Gamma P = 0$ *and* $\Gamma Q \neq 0$*,*
- *the coefficient space of* R *is full-rank. That is, if we interpret* R *as a matrix over* $\mathbb{F}$ *by listing each of its* w' *entries as a column vector of coefficients, then this matrix has full column-rank.*
- *the vector of polynomials* R *is simultaneously computable by a* UPT-SML *circuit of preimage-width at most* w'*.*

Proof. For an $S \subseteq [d]$, let $\mathbf{y}^S = \{m_1, \ldots, m_r\}$ and $\mathbf{y}^{-S} = \{n_1, \ldots, n_t\}$ in some order. Define $M_{f,S} \in \mathbb{F}^{r\times t}$ such that $M_{f,S}(i,j)$ is the coefficient of n_j in $\mathrm{coeff}_{m_i}(f)$. Note that the i^{th} row of $M_{f,S}$ is the polynomial $\mathrm{coeff}_{m_i}(f)$ written in the coefficient vector form.

For a type τ in a tree T, S_τ will denote the set of leaves of the node τ in T. Consequently, we will also use just $M_{f,\tau}$ to mean M_{f,S_τ}. We will denote by $B_{f,\tau}$ a set of monomials from $\mathbf{y}^{S_\tau}$ such that the rows indexed by them in $M_{f,S}$ will form a basis of the rows of $M_{f,S}$. Note that if τ has children τ_1, τ_2, then we can ensure that our choice of $B_{f,\tau}$ satisfies $B_{f,\tau} \subseteq B_{f,\tau_1} \times B_{f,\tau_2}$ as the latter is clearly a spanning set. Using such a basis $B_{f,\tau}$, we can then write down a set of dependencies as below corresponding to f and τ.

$$\forall m \in \mathbf{y}^{S_\tau} : \mathrm{coeff}_m(f) = \sum_{m' \in B_{f,\tau}} \gamma_{m,m'}\, \mathrm{coeff}_{m'}(f). \tag{3.55}$$

Using this, we can rewrite f in the following way for any $\tau \in T$.

$$f = \sum_{m_k \in \mathbf{y}^{S_\tau}} m_k \left(\sum_{m_i' \in B_{f,\tau}} \gamma_{i,k}\, \mathrm{coeff}_{m_i'}(f) \right) = \sum_{m_i' \in B_{f,\tau}} \left(\sum_{m_k \in \mathbf{y}^{S_\tau}} \gamma_{i,k} m_k \right) \mathrm{coeff}_{m_i'}(f)$$
$$f = \sum_{m_i' \in B_{f,\tau}} u_i(\mathbf{y}^{S_\tau})\, \mathrm{coeff}_{m_i'}(f) \quad \text{for some } u_i \in \mathbb{F}[\mathbf{y}_{S_\tau}]. \tag{3.56}$$

Suppose $\tau \in T$ has two children τ_1 and τ_2 that share the same dependencies for g as well. That is,

$$f = \sum_{m_i' \in B_{f,\tau_1}} u_i(\mathbf{y}^{S_{\tau_1}})\,\mathrm{coeff}_{m_i'}(f), \qquad f = \sum_{n_j' \in B_{f,\tau_2}} v_j(\mathbf{y}^{S_{\tau_2}})\,\mathrm{coeff}_{n_j'}(f),$$

$$g = \sum_{m_i' \in B_{f,\tau_1}} u_i(\mathbf{y}^{S_{\tau_1}})\,\mathrm{coeff}_{m_i'}(g), \qquad g = \sum_{n_j' \in B_{f,\tau_2}} v_j(\mathbf{y}^{S_{\tau_2}})\,\mathrm{coeff}_{n_j'}(g).$$

Combining them (and renaming the variables by dropping the $'$s), we get

$$f = \sum_{(m_i,n_j) \in B_{f,\tau_1} \times B_{f,\tau_2}} u_i(\mathbf{y}^{S_{\tau_1}}) v_j(\mathbf{y}^{S_{\tau_2}}) \cdot \mathrm{coeff}_{m_i \cdot n_j}(f),$$

$$g = \sum_{(m_i,n_j) \in B_{f,\tau_1} \times B_{f,\tau_2}} u_i(\mathbf{y}^{S_{\tau_1}}) v_j(\mathbf{y}^{S_{\tau_2}}) \cdot \mathrm{coeff}_{m_i \cdot n_j}(g).$$

Observe that if for all $m \in B_{f,\tau_1} \times B_{f,\tau_2}$ we have

$$\mathrm{coeff}_m(f) = \sum_{m' \in B_{f,\tau}} \gamma_{m,m'}\,\mathrm{coeff}_{m'}(f) \quad , \quad \mathrm{coeff}_m(g) = \sum_{m' \in B_{f,\tau}} \gamma_{m,m'}\,\mathrm{coeff}_{m'}(g),$$

then this also forces that by (3.56), for τ:

$$f = \sum_{m_i \in B_{f,\tau}} u_i'(\mathbf{y}^{S_\tau})\,\mathrm{coeff}_{m_i}(f) \quad , \quad g = \sum_{m_i \in B_{f,\tau}} u_i'(\mathbf{y}^{S_\tau})\,\mathrm{coeff}_{m_i}(g).$$

Since g is *not* computable by a UPT-SML circuit with underlying shape T this cannot happen for all $\tau \in T$. Let us pick the lowest τ (closest to the leaves; and say its children are τ_1, τ_2) such that for some $m \in B_{f,\tau_1} \times B_{f,\tau_2}$ we have

$$\begin{aligned} \mathrm{coeff}_m(f) &= \sum_{m' \in B_{f,\tau}} \gamma_{m,m'}\,\mathrm{coeff}_{m'}(f), \\ \mathrm{coeff}_m(g) &\neq \sum_{m' \in B_{f,\tau}} \gamma_{m,m'}\,\mathrm{coeff}_{m'}(g). \end{aligned} \tag{3.57}$$

The choice of the vector of polynomials is now clear. If $w' = |B_{f,\tau_1}| \cdot |B_{f,\tau_2}| \leq w^2$, then

$$R := \left(u_i(\mathbf{y}^{S_{\tau_1}}) v_j(\mathbf{y}^{S_{\tau_2}}) \ : \ (m_i, n_j) \in B_{f,\tau_1} \times B_{f,\tau_2}\right) \in \mathbb{F}[\mathbf{y}_{S_\tau}]^{1 \times w'}$$

$$P := \left(\mathrm{coeff}_{m_i \cdot n_j}(f) \ : \ (m_i, n_j) \in B_{f,\tau_1} \times B_{f,\tau_2}\right)^T \in \mathbb{F}[\mathbf{y}_{-S_\tau}]^{w' \times 1}$$

$$Q := \left(\mathrm{coeff}_{m_i \cdot n_j}(g) \ : \ (m_i, n_j) \in B_{f,\tau_1} \times B_{f,\tau_2}\right)^T \in \mathbb{F}[\mathbf{y}_{-S_\tau}]^{w' \times 1}.$$

It is clear from the definition that the vectors P and Q are made up of coefficients of f and g. Also, (3.57) provides a suitable vector Γ of support at most $w+1$ such that $\Gamma P = 0$ but $\Gamma Q \neq 0$.

It follows that the coefficient space of R is full-rank as the sets of polynomials $\{u_i : i \in B_{f,\tau_1}\}$ and $\{v_j : j \in B_{f,\tau_2}\}$ are linearly independent and are on disjoint sets of variables.
We only need to show that every entry of R can also be computed by a UPT-SML circuit of preimage-width at most w^2. To see this, observe that the set $\left\{\text{coeff}_m(f) : m \in \mathbf{y}^{-S_{\tau_1}}\right\}$ is spanned by the set $\left\{u_i(\mathbf{y}^{S_{\tau_1}}) : i \in B_{f,\tau_1}\right\}$, and similarly the set $\left\{\text{coeff}_n(f) : n \in \mathbf{y}^{-S_{\tau_2}}\right\}$ is spanned by $\left\{v_j(\mathbf{y}^{S_{\tau_2}}) : j \in B_{f,\tau_2}\right\}$. Since the dimension of these spaces is at most w, it follows that each $u_i(\mathbf{y}^{S_{\tau_1}})$ can be written as a linear combination of at most w many $\text{coeff}_m(f)$'s, and similarly each $v_j(\mathbf{y}^{S_{\tau_2}})$. Observation 3.54 shows that each of the coefficient polynomials can also be computed by UPT-SML circuits of preimage-width at most w. Thus, by computing each of the u_i's and v_j's separately, and then taking all w^2 products, we have a UPT-SML circuit of preimage-width at most w^2 that simultaneously computes all the entries of R. $\square$

3.11 Finer analysis of constant width UPT circuits

We now present some results comparing the power of constant width UPT circuits to variants of non-commutative ABPs and formulas. We recall the following statements that we shall use in the rest of this section.

- Nisan [Nis91] showed that for any homogeneous non-commutative polynomial f the *width* of a (homogeneous) ABP computing it, in the layer i, is exactly equal to the rank of the partial derivative matrix of f for degree i.
- Lagarde et al. [LMP19] show that in the smallest UPT circuit of shape T computing a polynomial f, the number of gates of a type $\tau \in T$ is equal to the rank of the *generalised partial derivative matrix* (formally defined in the proof of Theorem 3.5) for the type τ.

Thus, the ranks of the appropriate generalised partial derivative matrices for f characterize the ABP complexity and the UPT complexity of f. We will drop the term 'generalised' for the remainder of this discussion.

3.11.1 Constant width ABPs

Note that an ABP of width w can directly be converted to a UPT circuit of preimage-width $O(w^2)$. Also, for an $m = 2$ the polynomial family $\{P_d(y_1, \ldots, y_m)\}$ from Section 3.7 yields the following lemma using in Lemma 3.23 and Theorem 3.24.

Lemma 3.58. *There is a family of non-commutative bivariate degree n polynomials $\{Q_n\}$[8] such that for all large enough n, there is a UPT circuit for Q_n with size $O(\log n)$ and preimage-width $O(1)$, and for any shuffling of Q_n, an ABP computing it has width $n^{\Omega(1)}$.* $\square$

Putting these observations together we get that constant width ABPs are strictly weaker than constant width UPT circuits, even under shufflings.

[8]The polynomial Q_n is P_d for $d = \lfloor \log n \rfloor$.

3.11.2 General non-commutative ABPs and formulas

Constant width UPT circuits are weaker. For a large enough constant c, consider the following family of a non-commutative variant of the elementary symmetric polynomial family.

$$\mathrm{NESym}_n(x_1,\ldots,x_n) = \sum_{1\le i_1<\cdots<i_c\le n} x_{i_1}x_{i_2}\cdots x_{i_c}$$

Lemma 3.59. *For all large enough n, the polynomial NESym_n is computable by a* formula *of size $O(n^c)$. Also, any shuffling of NESym_n requires UPT circuits of width $n^{\Omega(1)}$.*

Proof Sketch. Note that for any bipartition of the positions, the corresponding *partial derivative matrix* of $\mathrm{NESym}_n(\mathbf{x})$ is a *set disjointness* matrix. Therefore it has full row rank, which is at least $n^{\Omega(1)}$ for any bipartition with the smaller part having size between $c/3$ and $2c/3$. Once we have this, it is clear that any shuffling of NESym_n continues to have this property. Thus, we have that no shuffling of NESym_n has a UPT circuit of constant preimage-width.

Also, NESym_n has exactly $\binom{n}{c}$ monomials and therefore has $\mathrm{poly}(n)$ sized formulas. □

ABPs are weaker (without shufflings). Next, let us consider family of the bivariate palindrome polynomials of degree $2n$.

$$\mathrm{Pal}_n(x_1,x_2) = \sum_{(i_1,\ldots,i_n)\in[2]^n} (x_{i_1}x_{i_2}\cdots x_{i_n})\cdot(x_{i_n}\cdots x_{i_2}x_{i_1})$$

The following lemma is now easy to verify using the discussions in rest of this chapter about the palindrome family.

Lemma 3.60. *For all large enough n, $P_n(x_1,x_2)$ has a UPT circuit of size $\mathrm{poly}(n)$ and constant preimage-width. Also, any ABP computing $P_n(x_1,x_2)$ requires size $2^{\Omega(d)} = n^{\omega(1)}$.* □

Formulas simulate shufflings of constant width UPT circuits. The following easy lemma tells us that any polynomial that has a constant width UPT circuit has a shuffling that is efficiently computable by a formula.

Lemma 3.61. *If f_n is an n-variate degree d polynomial that has a UPT circuit of size s and preimage-width w, then there is a shuffling $\Delta_\sigma(f)$ of f_n that has formulas of size $\mathrm{poly}(s, w^{O(\log d)})$.*

Proof Sketch. We know from Theorem 3.4 that there is a shuffling of f, say f', that has a UPT circuit of preimage-width $O(w^2)$ and depth $O(\log d)$. Let the shape of the new circuit be T. Now for any type $\tau \in T$ and any gate $g \sim \tau$, the number of paths from g to the root is at most $w^{\text{depth}} = w^{O(\log d)}$. Thus a careful replication of gates in the depth-reduced UPT circuit will give us a formula of the required size. □

4 | Isolating Log-variate Polynomials

In this chapter we give a construction of an explicit isolating family of weight assignments for depth-3 powering circuits of size s that depend on $O(\log s)$ variables. We achieve this by studying the Newton polytopes of polynomials that have low dimension of partial derivatives. Our construction extends to all log-variate polynomials with low dimension of partial derivatives, and also gives an efficient explicit hitting set for such polynomials, reproving a recent result of Forbes, Ghosh and Saxena [FGS18].

4.1 Introduction

The model of depth-3 powering circuits is perhaps the most well-understood *complete* model in the context of lower bounds, after depth-2 circuits. We know of tight exponential lower bounds against this model for polynomials as simple as the monomial $x_1 x_2 \cdots x_n$. However, when it comes to explicit hitting sets, the best known result is an $n^{O(\log\log n)}$ time construction due to Forbes, Saptharishi and Shpilka [FSS14]. Therefore constructing polynomial-sized explicit hitting sets for this model remains a highly interesting open question.

Expressing polynomials as sums of powers of linear forms has been studied in connection to the notion of *Waring rank*, which is a classical problem in algebraic geometry with works dating as far back as the late 19th century (see [IK99] for details). However, the model of depth-3 powering circuits was first considered in the algebraic complexity literature in the work of Saxena [Sax08]. He analysed the question of PIT for various structured constant depth models and gave efficient *whitebox PITs* for those models. He also proved an exponential lower bound against depth-3 powering circuits in the same work. In the context of lower bounds, this model is quite well understood, and we now know of tight exponential lower bounds against depth-3 powering circuits computing a monomial [RS11].

In the context of hitting sets, independent quasipolynomial sized ($s^{O(\log s)}$) constructions were shown by Forbes and Shpilka [FS12], and Agrawal, Saha and Saxena [ASS13]. Later, Forbes, Saptharishi and Shpilka [FSS14] improved this to a hitting set of size $s^{O(\log\log s)}$ by using a technique of Shpilka and Volkovich [SV15] in combination with a result of Forbes and Shpilka [FS13] on hitting sets for ROABPs. A different construction achieving the same parameters is also known due to Gurjar, Korwar and Saxena [GKS17]. Forbes, Ghosh and Saxena [FGS18] give an efficient ($\mathrm{poly}(s,d)$) hitting set for this model, when the number of variables depends logarithmically on the size ($n = O(\log s)$). Obtaining efficient hitting sets for depth-3 powering circuits in the general setting remains an open question.

4.1.1 Isolating Weight Assignments

The main tool in designing explicit hitting sets is the notion of *hitting set generators (HSGs)* which are polynomial maps that preserve nonzeroness (see Definition 2.10). One way to design an HSG is to come up with an *efficient weight assignment* on the underlying set of variables, and to then construct a polynomial map that essentially carries out this assignment (see Definition 2.18). This idea of using weight assignment for identity testing was introduced by Mulmuley, Vazirani and Vazirani [MVV87], in the context of obtaining fast parallel algorithms for perfect matching. They proposed the concept of an *isolating weight assignment* via a lemma that is known as the *isolation lemma*.

Lemma 4.1 (Isolation Lemma [MVV87] (Informal))**.** *Suppose A is an arbitrary subset of $2^{[n]}$. Then for some $m = \mathrm{poly}(n)$, a random weight assignment $w : [n] \to [m]$ is such that the element $a \in A$ that minimizes $w(a) := \sum_{i \in a} w(i)$, is unique, with probability at least $2/3$.*

Note that the bound on m is independent of the size of the set A. Suppose A is the set of monomials of some n-variate multilinear polynomial, and suppose that $w : [n] \to [m]$ is an assignment that has a unique minimum weight monomial. Note that such as assignment naturally gives us an HSG using Definition 2.18 and Lemma 2.19. Thus, coming up with isolating weight assignments or isolating families is an effective technique for designing hitting sets[1]. Some notable examples of hitting sets based on isolating weight assignments are the hitting sets for depth-2 circuits [KS01, AB03], and the hitting sets for ROABPs [AGKS15, GG20].

4.1.2 Results in this chapter

The main contribution of this work is an explicit isolating family for the class of log-variate polynomials with small dimension of partials.

Theorem 4.2 (Main Theorem)**.** *Let $\mathbb{F}$ be a field of characteristic zero. Let $d, k \in \mathbb{N}$ be large enough, and for some $n = O(\log k)$, let $\mathcal{C}(k, d)$ be the class of n-variate, degree d polynomials over $\mathbb{F}$, such that $\partial^*(f) \le k$ (see Definition 4.8), for all $f \in \mathcal{C}$. Then there exists an explicit family $W(k, d)$ consisting of $\mathrm{poly}(k, \log d)$ weight assignments, which isolates $\mathcal{C}(k, d)$.*

This theorem yields an explicit hitting set for the same class using Lemma 2.19, reproving a result of Forbes, Ghosh and Saxena [FGS18].

Corollary 4.3 (Hitting Set for Low Partials)**.** *Let $\mathbb{F}$ be a field of characteristic zero. For all large enough $k, d \in \mathbb{N}$ and all $n = O(\log k)$, the class $\mathcal{C}(k, d)$ of n-variate, degree d polynomials over $\mathbb{F}$ with dimension of partials at most k, has a hitting set of size $\mathrm{poly}(k, d)$.*

[1] Although Lemma 4.1 has been stated for a family of subsets of $[n]$, it easily extends to multisets, thereby being applicable for non-multilinear PIT.

4.1.3 Proof Idea

Our main result is the construction of an explicit weight assignment family that *isolates* (see Definition 2.16) log-variate polynomials that have a low dimension of partial derivatives.

We observe that for any polynomial f, a weight assignment wt is a *linear function* on the *exponent vectors* of the monomials in f, and hence it can be seen as a linear function on the *Newton polytope of* f (see Definition 4.5). Thus, if we can design a linear function $\ell(\mathbf{e})$ that is uniquely minimised at a vertex $\mathbf{a}$ of the Newton polytope of f, then we will be done. One way to achieve this is to design a weight assignment that gives distinct weights to all the vertices of the polytope. Therefore, if we bound the number of vertices of the Newton polytope for any polynomial computable by a depth-3 powering circuit, by say r, then we can come up with a set of $\text{poly}(r)$ assignments using known techniques [KS01, AB03].

In order to prove such a bound, we then observe that for any polynomial f and any vertex $\mathbf{a}$ of the Newton polytope of f, the dimension of partial derivatives of f is lower bounded by the *conesize* of $\mathbf{a}$, which extends an observation of Forbes [For14, Corollary 8.4.13] (see Definition 4.4). We then apply the known bounds on the dimension of partial derivatives for depth-3 powering [For14, Lemma 8.4.8] and the number of monomials of low-cone size [FGS18], to derive Theorem 4.2.

4.1.4 Related work

Newton polytopes. In algebraic complexity, polynomials have been studied previously by analysing the structure of their Newton polytopes, although not in the context of PIT. Koiran, Portier, Tavenas and Thomassé [KPTT15] studied the Newton polytopes of bivariate polynomials. Among other things, they formulated a *τ-conjecture for Newton polygons* which they showed implies that the Permanent requires exponential sized algebraic circuits. The recent work of Hrubeš and Yehudayoff [HY20] studies *shadows of Newton polytopes* and shows that it is connected to the monotone formula, and in some cases the monotone circuit complexity, of multivariate polynomials. The works of Fenner, Gurjar and Thierauf [FGT16], and Gurjar, Thierauf and Vishnoi [GTV18] are some notable examples of constructing isolating families for polytopes arising from combinatorial problems. In [FGT16], a *quasi-explicit* isolating family is constructed for the bipartite matching polytope. In [GTV18], the techniques used in [FGT16] are generalised to polytopes with *totally unimodular faces*.

Comparison with [FGS18]. The main result of this work reproves a result of Forbes, Ghosh and Saxena [FGS18], as mentioned earlier, and also largely resembles their approach. Both the works crucially use an upper bound in the number of monomials of "low conesize" (refer Lemma 4.13). However, [FGS18] show that the coefficients of monomials of low conesize can be efficiently computed, and directly obtain a blackbox PIT. On the other hand, we obtain hitting sets from this upper bound, by constructing an *isolating weight assignment* for the underlying class, by analysing the structure of the Newton polytopes of polynomials in it.

4.2 Background

We now fix some notation and state some known results that we will be using in this chapter.

We use the shorthand $\sum^{[s]} \wedge^{[d]} \sum^{[n]}$ to denote n-variate depth-3 powering circuits with degree $\leq d$ and top fan-in $\leq s$. We drop the superscripts whenever they are clear from the context or are not irrelevant.

Definition 4.4 ($\mathbb{F}$-cone of a monomial)**.** *Let $\mathbb{F}$ be some field and m be a monomial over variables $\mathbf{x} = \{x_1, \ldots, x_n\}$. We define the* $\mathbb{F}$-conesize *of the monomial m as follows.*

$$\mathbb{F}\text{-cone}(m) = \{\partial_{\mathbf{e}} m : \mathbf{e} \in \mathbb{N}^n, \partial_{\mathbf{e}} m \neq 0\}$$

Here $\partial_{\mathbf{e}} m$ denotes the derivative of m with respect to $\mathbf{x}^{\mathbf{e}}$ over $\mathbb{F}$. ♢

Newton Polytopes

We build our family of isolating weight assignments by analysing the *Newton polytopes* of polynomials that are computed by depth-3 powering circuits. We will therefore need the following definition of the *Newton polytope* of a polynomial, and also the definition of a *vertex of a polytope*.

Definition 4.5 (Newton Polytope)**.** *For a polynomial $f(\mathbf{x})$, the* Newton Polytope *of f, denoted by $\mathcal{P}(f)$ is defined as $\mathcal{P}(f) := \text{conv-hull}\left(\{\mathbf{e} \in \mathbb{N}^n : \text{coeff}_f(\mathbf{x}^{\mathbf{e}}) \neq 0\}\right) \subseteq \mathbb{R}^n$.* ♢

Definition 4.6 (Vertex of a Polytope)**.** *Let $\mathcal{P} \subseteq \mathbb{R}^n$ be a convex polytope. A point $\mathbf{v} \in \mathcal{P}$ is said to be a vertex of $\mathcal{P}$, if it can not be expressed as a* non-trivial *convex combination of other points in $\mathcal{P}$. In other words, there is no choice of points $\mathbf{a}_1, \mathbf{a}_2, \ldots \in \mathcal{P}$ and $\lambda_1, \lambda_2, \ldots \in (0,1)$ with $\sum_i \lambda_i = 1$, that satisfies $\sum_i \lambda_i \mathbf{a}_i = \mathbf{v}$.* ♢

We will also need the following results about the behaviour of linear functions on convex polytopes, which can be found in any text on linear or combinatorial optimization (e.g. [PS82]).

Lemma 4.7 (Vertices and Minima of Linear Functions)**.** *For any convex polytope $\mathcal{P} \subseteq \mathbb{R}^n$, the following hold.*

- *For any $\mathbf{w} \in \mathbb{R}^n$ and a point $\mathbf{e} \in \mathcal{P}$, there exists a vertex $\mathbf{v}$ of $\mathcal{P}$ such that $\mathbf{w}^\mathsf{T}\mathbf{v} \leq \mathbf{w}^\mathsf{T}\mathbf{e}$.*
- *If $\mathbf{v}$ is a vertex of $\mathcal{P}$ and if $\mathbf{c}^\mathsf{T}\mathbf{v} < \mathbf{c}^\mathsf{T}\mathbf{v}'$ for all* vertices *$\mathbf{v}' \neq \mathbf{v}$ of $\mathcal{P}$, then the linear function $\ell_{\mathbf{c}} : \mathbf{y} \mapsto \mathbf{c}^\mathsf{T}\mathbf{y}$ is* uniquely *minimised on $\mathcal{P}$ at $\mathbf{v}$.*
- *If $\mathbf{v}$ is a vertex of $\mathcal{P}$, then there exists a vector $\mathbf{c} \in \mathbb{R}^n$ such that the linear function $\ell_{\mathbf{c}} : \mathbf{y} \mapsto \mathbf{c}^\mathsf{T}\mathbf{y}$ is* uniquely *minimised on $\mathcal{P}$ at $\mathbf{v}$.*

Dimension of Partial Derivatives

A key measure that we use to analyse polynomials computed by depth-3 powering circuits is the *dimension of partial derivatives* of polynomials, which is defined as follows.

Definition 4.8 (Dimension of Partial Derivatives [NW97]). *For a polynomial* $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ *the* dimension of the partial derivatives of f, *denoted by* $\partial^*(f)$, *is defined as follows.*

$$\partial^*(f) = \dim\left(\operatorname{span}_{\mathbb{F}}\left\{\frac{\partial^{|\mathbf{e}|} f}{\partial \mathbf{x}^{\mathbf{e}}} : \mathbf{e} \in \mathbb{N}^n\right\}\right)$$

◊

We will use the following lemma about the dimension of partial derivatives of depth-3 powering circuits, which first appeared in Forbes's thesis [For14], to apply Theorem 4.2 to the case of depth-3 powering circuits.

Lemma 4.9 (Dimension of Partial Derivatives of $\Sigma \bigwedge \Sigma$ [Folklore]). *Suppose* $f(\mathbf{x})$ *is a degree* d *polynomial that is computable by a* $\Sigma \bigwedge \Sigma$ *circuit of size* s. *Then* $\partial^*(f) \leq s(d+1)$.

4.3 Isolating Family for Structured Log-variate Polynomials

The key ingredient of our result is the following structural lemma relating the dimension of partial derivatives of a polynomial, to the vertices in its Newton polytope.

Lemma 4.10 (Partial Derivatives of Vertices). *Let* $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ *be an* n*-variate polynomial of degree* d *and let* $\mathcal{P}(f)$ *be it's Newton Polytope. Then for any vertex* $\mathbf{a}$ *of* $\mathcal{P}(f)$ *we have that* $\dim(\partial^*(f)) \geq \dim(\partial^*(\mathbf{x}^{\mathbf{a}})) = |\mathbb{F}\text{-cone}(\mathbf{a})|$.

Proof. Let $\mathbf{w} \in \mathbb{R}^n$ be such that the linear function $\ell_{\mathbf{w}} : \mathbf{y} \mapsto \mathbf{w}^{\mathsf{T}}\mathbf{y}$ is uniquely minimised at $\mathbf{a}$. Such a vector $\mathbf{w}$ exists because $\mathbf{a}$ is a vertex of $\mathcal{P}(f)$, by Lemma 4.7.

Claim 4.11. *If* $\mathbf{b} \in \mathbb{N}^n$ *is such that* $\partial_{\mathbf{b}}(\mathbf{x}^{\mathbf{a}}) \neq 0$, *then the function* $\ell_{\mathbf{w}}$ *is uniquely minimised at* $(\mathbf{a}-\mathbf{b})$ *on the polytope* $\mathcal{P}' = \mathcal{P}(\partial_{\mathbf{b}}(f))$.

Proof. Let $f' = \partial_{\mathbf{b}}(f)$ and note that $\operatorname{supp}(f') = \left\{\mathbf{x}^{(\mathbf{e}-\mathbf{b})} : \mathbf{x}^{\mathbf{e}} \in \operatorname{supp}(f), \partial_{\mathbf{b}}(\mathbf{x}^{\mathbf{e}}) \neq 0\right\}$. Now suppose that the function $\ell_{\mathbf{w}}$ is minimised at some $\mathbf{e}' \in \mathcal{P}'$. We can assume that $\mathbf{e}'$ is a vertex of $\mathcal{P}'$ due to Lemma 4.7, which means $\mathbf{x}^{\mathbf{e}'} \in \operatorname{supp}(f')$. Therefore, there exists a monomial $\mathbf{x}^{\mathbf{e}} \in \operatorname{supp}(f)$ such that $\mathbf{e} = \mathbf{e}' + \mathbf{b}$. But then $\ell_{\mathbf{w}}(\mathbf{e}) = \ell_{\mathbf{w}}(\mathbf{e}'+\mathbf{b}) \leq \ell_{\mathbf{w}}(\mathbf{a}-\mathbf{b}) + \ell_{\mathbf{w}}(\mathbf{b}) = \ell_{\mathbf{w}}(\mathbf{a})$, which contradicts the definition of $\mathbf{w}$. □

Now, let $M(f)$ be a matrix with rows and columns indexed by monomials in $\mathbf{x}$ of degree at most d, such that the mth row of $M(f)$ reads out the *coefficient vector* of $\partial_m(f)$. Note that $\operatorname{rank}(M(f)) = \dim(\partial^*(f))$, and therefore it is enough show that $\operatorname{rank}(M(f)) \geq |\mathbb{F}\text{-cone}(\mathbf{x}^{\mathbf{a}})|$. For that purpose, consider the submatrix $M_{\mathbf{a}}(f)$ of $M(f)$ which contains the rows of $M(f)$ that are indexed by monomials that divide $\mathbf{x}^{\mathbf{a}}$.

We shall order the columns of $M_{\mathbf{a}}(f)$ *non-decreasingly* with respect to $\mathbf{w}^{\mathsf{T}}\mathbf{y}$; we break ties arbitrarily. Note that the leading entries of all the rows in $M_{\mathbf{a}}(f)$ are indexed precisely by the sub-monomials of $\mathbf{x}^{\mathbf{a}}$, as their exponent vectors uniquely minimise $\ell_{\mathbf{w}}$ in the Newton polytopes of the respective partial derivatives of f (Claim 4.11). Therefore all the nonzero rows of

$M_{\mathbf{a}}(f)$, indexed by monomials from $\mathbb{F}\text{-cone}(\mathbf{x}^{\mathbf{a}})$, are linearly independent. This finishes the proof. □

Thus, an upper bound on the dimension of partials of a polynomial translates to an upper bound on the conesize of the vertices of its polytope, as stated below.

Corollary 4.12. *For a polynomial $f(\mathbf{x})$ with $\partial^*(f) \le k$ and any vertex $\mathbf{a}$ of $\mathcal{P}(f)$, we have that $\mathbb{F}\text{-cone}(\mathbf{a}) \le k$.*

This further lets us bound the number of vertices of $\mathcal{P}(f)$ for an n-variate degree-d polynomial $f(\mathbf{x})$ whose dimension of partials is at most k, using the following lemma from [FGS18][2].

Lemma 4.13 (Counting Monomials with Small Cone Size [FGS18])**.** *For a field $\mathbb{F}$ of characteristic zero, let $n, d, k \in \mathbb{N}$ be large enough. The number of n-variate, degree d monomials that have $\mathbb{F}$-conesize $\le k$ is at most $\binom{n}{\log k} \cdot k^2$.*

We can now prove Theorem 4.2, which we first restate.

Theorem 4.2 (Main Theorem)**.** *Let $\mathbb{F}$ be a field of characteristic zero. Let $d, k \in \mathbb{N}$ be large enough, and for some $n = O(\log k)$, let $\mathcal{C}(k,d)$ be the class of n-variate, degree d polynomials over $\mathbb{F}$, such that $\partial^*(f) \le k$ (see Definition 4.8), for all $f \in \mathcal{C}$. Then there exists an explicit family $W(k,d)$ consisting of $\mathrm{poly}(k, \log d)$ weight assignments, which isolates $\mathcal{C}(k,d)$.*

Proof. Fix an arbitrary polynomial $f \in \mathcal{C}(k,d)$ and let $\mathcal{P} = \mathcal{P}(f)$ be its Newton polytope. First, Lemma 4.10 tells us that all the vertices of $\mathcal{P}$ have $\mathbb{F}$-conesize at most k. We can then use Lemma 4.13 to conclude that $\mathcal{P}$ has at most $\left(\binom{n}{\log k} \cdot k^2\right)$ vertices, which is $\mathrm{poly}(k)$ for any $n = O(\log k)$. Hence, from Lemma 4.7 and Lemma 2.21, it follows that for some $A = \mathrm{poly}(k)$, the family of assignments $W(n,d,A)$ defined in Definition 2.20 isolates f. Moreover, since the same bound on number of vertices of $\mathcal{P}(f)$ holds for all $g \in \mathcal{C}(k,d)$, we get that $W(n,d,A)$ isolates $\mathcal{C}$. □

The following corollaries, which first appeared in the work of Forbes, Ghosh and Saxena [FGS18], then follow immediately using Lemma 2.19 and Lemma 4.9.

Corollary 4.3 (Hitting Set for Low Partials)**.** *Let $\mathbb{F}$ be a field of characteristic zero. For all large enough $k, d \in \mathbb{N}$ and all $n = O(\log k)$, the class $\mathcal{C}(k,d)$ of n-variate, degree d polynomials over $\mathbb{F}$ with dimension of partials at most k, has a hitting set of size $\mathrm{poly}(k,d)$.*

Corollary 4.14 (Hitting Sets for Log-variate $\Sigma \wedge \Sigma$)**.** *Let $\mathbb{F}$ be a field of characteristic zero. For all large enough $s, d \in \mathbb{N}$ and all $n = O(\log k)$, the class $\Sigma^{[s]} \wedge^{[d]} \Sigma$ of n-variate depth-3 powering circuits over $\mathbb{F}$, has a hitting set of size $\mathrm{poly}(s,d)$.*

[2]Forbes, Ghosh and Saxena cite a private communication with Saptharishi for this lemma.

4.4 Discussion

Apart from proving a slightly stronger version of the hitting set result in [FGS18], we believe that our technique of studying Newton polytopes for the task of PIT is of independent interest. More specifically, our technique can possibly be combined with previous works on isolating assignments for polytopes ([FGT16, GTV18]) to give better parameters for general depth-3 powering circuits.

Moreover, to the best of our knowledge, this work is the first to analyse the Newton polytopes of polynomials for constructing explicit hitting sets. This approach could find applications in blackbox PITs for other structured models.

5 | Near-optimal Bootstrapping of Hitting Sets for Algebraic Models

This chapter discusses some findings centred around the question *"what would happen if we had marginally non-trivial hitting sets for constant variate circuits or formulas?"*. This question was first considered by Agrawal, Ghosh and Saxena [AGS19]. Our results are essentially obtained by optimizing their methods using some crucial observations, the details of which we now present[1].

5.1 Bootstrapping of Hitting Sets

A result of Agrawal, Ghosh and Saxena [AGS19] shows, among other things, the following surprising result: blackbox PIT algorithms for size s, degree s and n-variate circuits with running time as bad as $\left(s^{n^{0.5-\delta}}\right)$, where $\delta > 0$ is a constant, can be used to construct blackbox PIT algorithms for size s circuits with running time $s^{\exp(\exp(O(\log^* s)))}$. Note that $\log^* n$ refers to the smallest i such that the i-th iterated logarithm $\log^{\circ i}(n)$ is at most 1. This shows that certain mild derandomisation of PIT would be sufficient to get a nearly complete derandomisation. Their proof uses a novel *bootstrapping* technique where they use the connections between hardness and derandomisation repeatedly so that by starting with a weak hitting set we can obtain better and better hitting sets.

One of the open questions of Agrawal, Ghosh and Saxena [AGS19] was whether the hypothesis can be strengthened to a *barely* non-trivial derandomisation. That is, suppose we have a blackbox PIT algorithm, for the class of size s, degree s and n-variate circuits, that runs in time $s^{o(n)}$, can we use this to get a nearly complete derandomisation? Note that we have a trivial $(s+1)^n \cdot \operatorname{poly}(s)$ algorithm from the *polynomial identity lemma* [Ore22, DL78, Zip79, Sch80]. Our main result is an affirmative answer to this question in a very strong sense. Furthermore, our result holds for *typical* subclasses that are reasonably well-behaved under composition. Formally, we prove the following theorem.

Theorem 5.1 (Bootstrapping PIT for algebraic formulas, branching programs and circuits)**.** *Let $\varepsilon > 0$ and $n \geq 2$ be constants. Suppose that, for all large enough s, there is an explicit hitting set of size $s^{n-\varepsilon}$ for all degree s, size s algebraic formulas (algebraic branching programs or circuits*

[1]The results in this chapter have appeared in [KST19].

respectively) over n variables. Then, there is an explicit hitting set of size $s^{\exp(\exp(O(\log^* s)))}$ for the class of degree s, size s algebraic formulas (algebraic branching programs or circuits respectively) over s variables.

Note that $(s+1)^{n-\varepsilon} = s^{n-\varepsilon} \cdot \left(1+\frac{1}{s}\right)^{n-\varepsilon} < e \cdot s^{n-\varepsilon} < s^{n-\varepsilon'}$ for some other constant $\varepsilon' > 0$ since s is large enough. Hence, for this theorem, there is no qualitative difference if the hitting set had size $(s+1)^{n-\varepsilon}$ instead of $s^{n-\varepsilon}$. We also note that as far as we understand, such a statement for classes such as algebraic branching programs or formulas, even with the stronger hypothesis of there being a $s^{O(n^{(1/2)-\varepsilon})}$, did not follow from the results of [AGS19]. We elaborate more on this, and the differences between our proof and theirs in the next subsection.

An interesting, albeit simple corollary of the above result is the following statement.

Corollary 5.2 (From slightly non-trivial PIT to lower bounds). *Let $\varepsilon > 0$ and $n \geq 2$ be constants. Suppose that, for all large enough s, there is an explicit hitting set of size $(s^{n-\varepsilon})$ for all degree s, size s algebraic formulas (algebraic branching programs or circuits respectively) over n variables. Then, for every function $d : \mathbb{N} \to \mathbb{N}$, there is a polynomial family $\{f_n\}$, where f_n is n variate and degree $d(n)$, and for every large enough n, f_n cannot be computed by algebraic formulas (algebraic branching programs or circuits respectively) of size smaller than $\binom{n+d}{d}^{1/\exp(\exp(O(\log^* nd)))}$. Moreover, there is an algorithm which when given as input an n variate monomial of degree d, outputs its coefficient in f_n in deterministic time $\binom{n+d}{d}$.*

Thus, a slightly non-trivial blackbox PIT algorithm leads to hard families with near optimal hardness. A recent work of Carmosino, Impagliazzo, Lovett and Mihajlin [CILM18] that has a similar theme shows that given an explicit polynomial family of constant degree which requires super linear sized non-commutative circuits, one can obtain explicit polynomial families of exponential hardness. Besides the obvious differences in the statements, one important point to note is that the notions of explicitness in the conclusions of the two statements are different from each other. In [CILM18], the final exponentially hard polynomial family is in VNP provided the initial polynomial family is also in VNP. On the other hand, for our result, we can say that the hard polynomial family obtained in the conclusion is explicit in the sense that its coefficients are computable in deterministic time $\binom{n+d}{d}$. Another difference between Corollary 5.2 and the main result of [CILM18] is in the hypothesis. From a non-trivial hitting set, we can obtain a large class of lower bounds by varying parameters appropriately (see Theorem 5.3), however the main result of [CILM18] starts with a lower bound for a single family. In that regard, our hypothesis appears to be much stronger and slightly non-standard. We discuss this issue in some detail at the end of the next section.

In another relevant result, Jansen and Santhanam [JS12] showed that marginal improvements to known hitting set constructions imply lower bounds for the permanent polynomial. In particular, they show that a "sufficiently succinct" hitting set of size d, for univariates of degree d that have *constant-free* algebraic circuits of small size, would imply that the permanent polynomial requires super-polynomial sized *constant-free* algebraic circuits. Note that even though their hypothesis needs a much weaker improvement in the size of the hitting set when compared to ours, the hitting set is additionally required to be "succinct"[2], which makes it difficult to compare the two hypotheses.

5.1.1 Proof overview

The basic intuition for the proofs in this chapter, and also for the proofs of the results in [AGS19], comes from the results of Kabanets and Impagliazzo [KI04], and those of Heintz and Schnorr [HS80] and Agrawal [Agr05]. We start by informally stating these results.

Theorem 5.3 (Informal, Heintz and Schnorr [HS80], Agrawal [Agr05]). *Let $H(n,d,s)$ be an explicit hitting set for circuits of size s, degree d in n variables. Then, for every $k \leq n$ and d' such that $d'k \leq d$ and $(d'+1)^k > |H(n,d,s)|$, there is a nonzero polynomial on n variables and individual degree d' that vanishes on the hitting set $H(n,d,s)$, and hence cannot be computed by a circuit of size s.*

In a nutshell, given an explicit hitting set, we can obtain hard polynomials. In fact, playing around with the parameters d' and $k \leq n$, we can get a hard polynomial on k variables, degree kd' for all k, d' satisfying $d'k < d$ and $(d'+1)^k > |H(n,d,s)|$.

We now state a result of Kabanets and Impagliazzo [KI04] that shows that hardness can lead to derandomisation.

Theorem 5.4 (Informal, Kabanets and Impagliazzo [KI04]). *A super-polynomial lower bound for algebraic circuits for an explicit family of polynomials implies a deterministic blackbox PIT algorithm for all algebraic circuits in n variables and degree d of size $\mathrm{poly}(n)$ that runs in time $\mathrm{poly}(d)^{n^{\varepsilon}}$ for every $\varepsilon > 0$.*

Now, we move on to the main ideas in our proof. Suppose we have non-trivial hitting sets for size s, degree $d \leq s$ circuits on n variables. The goal is to obtain a blackbox PIT for circuits of size s, degree s on s variables with a much better dependence on the number of variables.

Observe that if the number of variables was much much smaller than s, say at most a constant, then the hitting set in the hypothesis has a polynomial dependence on s, and we are done. We will proceed by presenting *variable reductions* to eventually reach this stage. With this in mind, the hitting sets for s variate circuits in the conclusion of Theorem 5.1 are designed iteratively starting from hitting sets for circuits with very few variables. In each iteration, we start with a hitting set for size s, degree $d \leq s$ circuits on n variables with some dependence on n and obtain a hitting set for size s, degree $d \leq s$ circuits on $m = 2^{n^{\delta}}$ variables (for some $\delta > 0$), that has a *much* better dependence on m. Then, we repeat this process till the number of variables increases up to s, which takes $O(\log^* s)$ iterations. We now briefly outline the steps in each such iteration.

[2]They require their hitting sets to be encoded by uniform TC^0 circuits of appropriately small size. See [JS12] for details.

- **Obtaining a family of hard polynomials :** The first step is to obtain a family of explicit hard polynomials from the given hitting sets. This step is done via Theorem 5.3, which simply uses interpolation to find a nonzero polynomial Q on k variables and degree d that vanishes on the hitting set for size s', degree d' circuits on n variables, for some s', d' to be chosen appropriately.

- **Variable reduction using Q :** Next, we take a *combinatorial design* (see Definition 5.7) $\{S_1, S_2, \ldots, S_m\}$, where each S_i is a subset of size k of a universe of size $\ell = \text{poly}(k)$, and $|S_i \cap S_j| \ll k$. Consider the map $\Gamma : \mathbb{F}[x_1, x_2, \ldots, x_m] \to \mathbb{F}[y_1, y_2, \ldots, y_\ell]$ given by the substitution $\Gamma(C(x_1, x_2, \ldots, x_m)) = C\left(Q(\mathbf{y}\mid_{S_1}), Q(\mathbf{y}\mid_{S_2}), \ldots, Q(\mathbf{y}\mid_{S_m})\right)$. As Kabanets and Impagliazzo show in the proof of Theorem 5.4, Γ preserves the nonzeroness of all algebraic circuits of size s on m variables, provided Q is hard enough.
  We remark that our final argument for this part is slightly simpler than that of Kabanets and Impagliazzo, and hence our results also hold for algebraic branching programs and formulas. In particular, we do not need Kaltofen's seminal result that algebraic circuits are closed under polynomial factorization, whereas the proof in [KI04] crucially uses Kaltofen's result [Kal89]. This come from the simple, yet crucial, observation that if Q vanishes on some hitting set, then so does any multiple of Q. This allows us to use the hardness of *low-degree* multiples of Q, and so, we do not need any complexity guarantees on factors of polynomials.

- **Blackbox PIT for m-variate circuits of size s and degree s :** We now take the hitting set given by the hypothesis for the circuit $\Gamma(C)$ (invoked with appropriate size and degree parameters) and evaluate $\Gamma(C)$ on this set. From the discussion so far, we know that if C is nonzero, then $\Gamma(C)$ cannot be identically zero, and hence it must evaluate to a nonzero value at some point on this set. The number of variables in $\Gamma(C)$ is at most $\ell = \text{poly}\log m$, whereas its size turns out to be *not too much* larger than s. Hence, the size of the hitting set for C obtained via this argument turns out to have a better dependence on the number of variables m than the hitting set in the hypothesis.

To prove Corollary 5.2, we let $t(n) = \exp(\exp(O(\log^* n)))$. Now, we invoke the the conclusion of Theorem 5.1 with $s = \binom{n+d}{d}^{1/10t(n)}$. Thus, we get an explicit hitting set H of size $\binom{n+d}{d}^{1/10}$ for n variate circuits of size s and degree d. We now use Theorem 5.3 to get a nonzero polynomial of degree d and n which vanishes on the set H and hence cannot be computed by circuits of size at most s.

Why does bootstrapping work? As far as we understand, the primary reason that makes such bootstrapping results feasible is the following observation from the results of Heintz-Schnorr and Agrawal [HS80, Agr05]. Given a single hitting set, we can obtain a *family* of lower bounds by varying the degree and the number of variables in the interpolating polynomial. In particular, one can obtain a k-variate polynomial that has hardness 2^{2^k} by appropriately choosing the parameter k and the individual degree of the interpolating polynomial. This is not true, for instance, in the boolean world. It turns out that in the result of Kabanets and Implagliazzo [KI04] that converts a hard polynomial P into a hitting set, the proof of this conversion has different sensitivities to the degree of P and the number of variables it depends on. This allows one to start with a moderately non-trivial hitting set, obtaining a hard polynomial from it of the *right* degree and number of variables, and use that to obtain a hitting set which is significantly better than what we started with. This, in our opinion, is a high level picture of why bootstrapping works in the algebraic world.

Similarities and differences with the proof of Agrawal, Ghosh and Saxena [AGS19]. The high level outline of our proof is essentially the same as that in [AGS19]. However, there are some differences that make our final arguments shorter, simpler and more robust than those from [AGS19] thus leading to a stronger and near optimal bootstrapping statement in Theorem 5.1. Moreover, as we already alluded to, our proof extends to formulas and algebraic branching programs as well, whereas, to the best of our understanding, those in [AGS19] do not. We now elaborate on the differences.

One of the main differences between the proofs in this work and those from [AGS19] is in the use of the the result of Kabanets and Impagliazzo [KI04]. Agrawal, Ghosh and Saxena [AGS19] use this result as a blackbox to get deterministic PIT using hard polynomials. The result in [KI04] crucially relies on a result of Kaltofen, which shows that low degree algebraic circuits are closed under polynomial factorization. That is, if a degree d, n variate polynomial P has a circuit of size at most s, then any factor of P has a circuit of size at most $(snd)^e$ for a constant e. Such a closure result is not known to be true for algebraic formulas, and hence the results in [AGS19] do not seem to extend to these settings. Additionally, the analysis of bootstrapping in [AGS19] seems to depend crucially on the fact that the circuit size grows additively with substitution ($\text{size}(f \circ \mathbf{g}) \leq \text{size}(f) + \sum_i \text{size}(g_i)$). This is not true for algebraic branching programs and hence it is not clear if their analysis holds for ABPs despite the recent result [ST20] about closure of ABPs under factoring. Also, the removal of any dependence on the "factorization exponent" e is crucial in our proof as it allows us to start with a hypothesis of a barely non-trivial hitting set. The other main difference between our proof and that in [AGS19] is rather technical but we try to briefly describe it. This is in the choice of combinatorial designs. The designs used in our work are based on the standard Reed-Solomon code and they yield larger set families than the designs used in [AGS19][3].

Also, their proof is quite involved and we are unsure if there are other constraints in their proof that force such choices of parameters. Our proof, though along almost exactly the same lines, appears to be more transparent and more malleable with respect to the choice of parameters.

The strength of the hypothesis. The hypothesis of Theorem 5.1 and also those of the results in [AGS19] is that we have a non-trivial explicit hitting set for algebraic circuits of size s,

[3]However, even without these improved design parameters, our proof can be used to provide the same conclusion when starting off with a hitting set of size $s^{n^{1-\delta}}$, instead of the hypothesis of Theorem 5.1.

degree d on n variables where d and s could be arbitrarily large as functions of n. This seems like an extremely strong assumption, and also slightly non-standard in the following sense. In a typical setting in algebraic complexity, we are interested in PIT for size s, degree d circuits on n variables where d and s are polynomially bounded in the number of variables n. A natural open problem here, which would be a more satisfying statement to have, would be to show that one can weaken the hypothesis in Theorem 5.1 to only hold for circuits whose degree and size are both polynomially bounded in n. It is not clear to us if such a result can be obtained using the current proof techniques, or is even true.

Having noted that our hypothesis is very strong, and perhaps even slightly unnatural with respect to the usual choice of parameters in the algebraic setting, we remark that our hypothesis does in fact follow from the assumptions that the Permanent is hard for Boolean circuits, and the Generalized Riemann Hypothesis (GRH). The proof is essentially the same as that of Corollary 1 in the work of Jansen and Santhanam [JS12]. The only difference is that while Jansen and Santhanam show that there are non-trivial explicit [4] hitting sets for univariate polynomials with small circuits assuming the hardness of Permanent for Boolean circuits and the GRH, here we have to work with circuits computing multivariate polynomials. At a high level, the proof in [JS12] proceeds by constructing a pseudorandom generator for Boolean circuits of appropriate size assuming the hardness of permanent for Boolean circuits. Then, the set of binary strings in the output of this generator is interpreted in a natural way as an integer. This gives us a small set of integer points, which can be constructed deterministically. Then they argue that there is no constant free algebraic circuit of small size which vanishes on all these integer points. The proof of this step is via contradiction, where they assume the existence of such a constant free algebraic circuit to construct a Boolean circuit of small size which is not fooled by the aforementioned Boolean pseudorandom generator. For algebraic circuits which are not constant free and are allowed to use arbitrary field constants and hence cannot be efficiently simulated by a Boolean circuit, they assume the GRH to reduce to the case of constant free circuits in a fairly standard way. For our setting, we interpret the output of the Boolean pseudorandom generator as not just a single integer point, but a k tuple of integers points. These set of points in $\mathbb{Z}^k$ form our candidate hitting set. The rest of the proof carries over without any changes. We refer the interested reader to [JS12] for further details.

Remark. *Throughout the chapter, we shall assume that there are suitable $\lfloor \cdot \rfloor$'s or $\lceil \cdot \rceil$'s if necessary so that certain parameters chosen are integers. We avoid writing this purely for the sake of readability.*

All results in this work continue to hold for the underlying model of algebraic formulas, algebraic branching programs or algebraic circuits. In fact, the results also extend to the model of border *of algebraic formulas, algebraic branching programs or algebraic circuits. That is, if there is a slightly non-trivial hitting set for polynomials in the border of these classes, then our main theorem gives a highly non-trivial explicit hitting set for these polynomials. Since our proofs extend as it is to this setting with essentially no changes, we skip the details for this part, and confine our discussions in the rest of the chapter to just standard algebraic formulas.* $\Diamond$

[4] In fact their notion of explicitness is stronger than ours.

5.2 Preliminaries

We now define some standard notions we work with, and state some of the known results that we use in this chapter.

We will use the following folklore algorithm for computing univariate polynomials, often attributed to Horner[5]. We also include a proof for completeness.

Proposition 5.5 (Horner rule). *Let $P(x) = \sum_{i=0}^{d} p_i x^i$ be a univariate polynomial of degree d over any field $\mathbb{F}$. Then, P can be computed by an algebraic formula of size $2d+1$.*

Proof. Follows from the fact that $P(x) = (\cdots((p_d x + p_{d-1})x + p_{d-2})\cdots)x + p_0$, which is a formula of size $2d+1$. $\square$

The following observation shows that the classes of algebraic formulas/ABPs/circuits are *robust* under some very natural operations. These are precisely the properties of the underlying models that we rely on in this work. Any circuit model that satisfies these properties would be sufficient for our purposes but we shall focus on just the standard models of formulas, ABPs and circuits.

Observation 5.6. *The class of polynomials computed by formulas/ABPs/circuits satisfy the following properties:*

- *Any polynomial of degree d with at most s monomials can be computed by a formula/ABP/circuit of size $s \cdot d$. In the specific setting when the polynomial is a univariate, it can be computed by a formula/ABP/circuit of size $O(d)$.*
- *Partial substitution of variables does not increase the size of the formula/ABP/circuit.*
- *If each of $Q_1, \ldots, Q_k$ is computable by size s formulas/ABPs/circuits, then $\sum Q_i$ is computable by size sk formula/ABP/circuit respectively.*
- *Suppose $P(x_1, \ldots, x_n)$ is computable by a size s_1 formula/ABP/circuit and say $Q_1, \ldots, Q_n$ are polynomials each of which can be computed by formulas/ABPs/circuits of size s_2. Then the composition $P(Q_1, \ldots, Q_n)$ can be computed by a formula/ABP/circuit of size at most $s_1 \cdot s_2$ respectively.* $\square$

5.2.1 Combinatorial designs

Definition 5.7 (Combinatorial designs [NW94]). *A family of sets $\{S_1, \ldots, S_m\}$ is said to be an (ℓ, k, r) design if*

- *$S_i \subseteq [\ell]$,*
- *$|S_i| = k$,*
- *$|S_i \cap S_j| < r$ for any $i \neq j$.* $\lozenge$

[5]Though this method was discovered at least 800 years earlier by Iranian mathematician and astronomer Sharaf al-Dīn Ṭūsī (cf. Hogendijk [Hog89]).

The following is a standard construction of such designs based on the *Reed-Solomon* code.

Lemma 5.8 (Construction of designs)**.** *Let $c \geq 2$ be any positive integer. There is an algorithm that, given parameters ℓ, k, r satisfying $\ell = k^c$ and $r \leq k$ with k being a power of 2, outputs an (ℓ, k, r) design $\{S_1, \ldots, S_m\}$ for $m \leq k^{(c-1)r}$ in time $\operatorname{poly}(m)$.*

Proof. Since k is a power of 2, we can identify $[k]$ with the field $\mathbb{F}_k$ of k-elements and $[\ell]$ with $\mathbb{F}_k \times \mathbb{F}_{k^{c-1}}$. For each univariate polynomial $p(x) \in \mathbb{F}_{k^{c-1}}[x]$ of degree less than r, define the set S_p as

$$S_p = \{(i, p(i)) \ : \ i \in \mathbb{F}_k\}.$$

Since there are $k^{(c-1)r}$ such polynomials we get $k^{(c-1)r}$ subsets of $\mathbb{F}_k \times \mathbb{F}_{k^{c-1}}$ of size k each. Furthermore, since any two distinct univariate polynomials cannot agree at r or more places, it follows that $\left|S_p \cap S_q\right| < r$ for $p \neq q$. $\square$

5.2.2 Hardness-randomness connections

Observation 5.9. *Let H be a hitting set for the class $\mathcal{C}(n, d, s)$ of n-variate polynomials of degree at most d that are computable by formulas of size s. Then, for any nonzero polynomial $Q(x_1, \ldots, x_n)$ such that $\deg(Q) \leq d$ and $Q(\mathbf{a}) = 0$ for all $\mathbf{a} \in H$, we have that Q cannot be computed by formulas of size s.*

Proof. If Q was indeed computable by formulas of size at most s, then Q is a member of $\mathcal{C}(n, d, s)$ for which H is a hitting set. This would violate the assumption that H was a hitting set for this class as Q is a nonzero polynomial in the class that vanishes on all of H. $\square$

From this observation, it is easy to see that explicit hitting sets can be used to construct lower bounds.

Lemma 5.10 (Hitting sets to hardness [HS80, Agr05])**.** *Let H be an explicit hitting set for $\mathcal{C}(n, d, s)$. Then, for any $k \leq n$ such that $k|H|^{1/k} \leq d$, there is a polynomial $Q(z_1, \ldots, z_k)$ of individual degree smaller than $|H|^{1/k}$ that is computable in time $\operatorname{poly}(|H|)$ that requires formulas of size s to compute it. Furthermore, given the set H, there is an algorithm to output a formula of size $|H| \cdot d$ for Q in time $\operatorname{poly}(|H|)$.*

Proof. This is achieved by finding a nonzero k-variate polynomial, for $k \leq n$, of individual degree $d' < |H|^{1/k}$, that vanishes on the hitting set H; this can be done by interpreting it as a homogeneous linear system with $(d'+1)^k$ "variables" and at most $|H|$ "constraints". Such a Q_k can then be found via interpolation by solving a system of linear equations in time $\operatorname{poly}(|H|)$. The degree of Q_k is at most $k \cdot |H|^{1/k} \leq d$ from the hypothesis and the hardness of Q_k follows from Observation 5.9. $\square$

Remark 5.11. *(Bit complexity of Q_k) Note that we can obtain*[6] *a hard polynomial Q_k such that its coefficients have bit complexities that are at most polynomially large in terms of the bit complexities of the points in the given hitting set.* $\Diamond$

It is also known that we can get non-trivial hitting sets from suitable hardness assumptions. For a fixed (ℓ, k, r) design $\{S_1, \dots, S_m\}$ and a polynomial $Q(z_1, \dots, z_k) \in \mathbb{F}[\mathbf{x}]$ we shall use the notation $Q[\![\ell, k, r]\!]_{\mathrm{NW}}$ to denote the vector of polynomials

$$Q[\![\ell, k, r]\!]_{\mathrm{NW}} := (Q(\mathbf{y}\mid_{S_1}), Q(\mathbf{y}\mid_{S_2}), \dots, Q(\mathbf{y}\mid_{S_m})) \in (\mathbb{F}[y_1, \dots, y_\ell])^m .$$

Kabanets and Impagliazzo [KI04] showed that, if $Q(\mathbf{z}_{[k]})$ is hard enough, then the composed polynomial $P(Q[\![\ell, k, r]\!]_{\mathrm{NW}})$ is nonzero if and only if $P(\mathbf{x}_{[m]})$ is nonzero. However, their proof crucially relies on a result of Kaltofen [Kal89] (or even a non-algorithmic version due to Bürgisser [Bür00]) about the complexity of factors of polynomials. Hence, this connection is not directly applicable while working with other subclasses of circuits such as algebraic formulas or algebraic branching programs as we do not know if they are closed under factorization. The following lemma can be used in such settings and this work makes heavy use of this.

Lemma 5.12 (Hardness to randomness without factor complexity). *Let $Q(z_1, \dots, z_k)$ be an arbitrary polynomial of individual degree smaller than d. Suppose there is an (ℓ, k, r) design $\{S_1, \dots, S_m\}$ and a nonzero polynomial $P(x_1, \dots, x_m)$, of degree at most D, that is computable by a formula of size at most s such that $P(Q[\![\ell, k, r]\!]_{\mathrm{NW}}) \equiv 0$. Then there is a polynomial $\tilde{P}(z_1, \dots, z_k)$, whose degree is at most $k \cdot d \cdot D$ that is divisible by Q and computable by formulas of size at most $s \cdot (r-1) \cdot d^r \cdot (D+1)$.*

Moreover, if $r = 2$, then this upper bound can be improved to $4 \cdot s \cdot d \cdot (D+1)$

If the polynomial $Q(z_1, \dots, z_k)$ in the above lemma was chosen such that Q vanished on some hitting set H for the class of size s', n-variate, degree d' polynomials where $s' \geq s \cdot (r-1) \cdot d^r \cdot (D+1)$, then so does $\tilde{P}$ since Q divides it. If it happens that $\deg(\tilde{P}) \leq d'$, then Observation 5.9 immediately yields that $\tilde{P}$ cannot be computed by formulas of size s', contradicting the conclusion of the above lemma. Hence, in such instances, we would have that $P(Q[\![\ell, k, r]\!]_{\mathrm{NW}}) \not\equiv 0$, without appealing to any factorization closure results.

Proof of Lemma 5.12. Borrowing the ideas from Kabanets and Impagliazzo [KI04], we look at the m-variate substitution $(x_1, \dots, x_m) \mapsto Q[\![\ell, k, r]\!]_{\mathrm{NW}}$ as a sequence of m univariate substitutions. We now introduce some notation to facilitate this analysis.

Given the (ℓ, k, r) design $\{S_1, \dots, S_m\}$, let $\mathbf{y}_i = \mathbf{y}\mid_{S_i}$, for each $i \in [m]$. The tuple $Q[\![\ell, k, r]\!]_{\mathrm{NW}}$ can therefore be written as $(Q(\mathbf{y}_1), Q(\mathbf{y}_2), \dots, Q(\mathbf{y}_m)) \in (\mathbb{F}[y_1, \dots, y_\ell])^m$. For each $0 \leq i \leq m$, let $P_i = P(Q(\mathbf{y}_1), Q(\mathbf{y}_2), \dots, Q(\mathbf{y}_i), x_{i+1}, \dots, x_m)$, which is P after substituting for the variables $x_1, \dots, x_i$. Since $P_0 = P$ is a nonzero polynomial and $P_m = P(Q[\![\ell, k, r]\!]_{\mathrm{NW}}) \equiv 0$, let t be the unique integer with $1 \leq t \leq m$, for which $P_{t-1} \not\equiv 0$ and $P_t \equiv 0$.

[6]via a cleverer variant of Gaussian elimination, e.g. Bareiss algorithm [Bar68].

Since $P_t(\mathbf{y}, x_t, \ldots, x_m)$ is a nonzero polynomial, there exist values that can be substituted to the variables besides x_t and $\mathbf{y}_t$ such that it remains nonzero; let this polynomial be $P'_t(\mathbf{y}_t, x_t)$. Also, for each $j \in [t-1]$, let $Q^{(t)}(\mathbf{y}_j \cap \mathbf{y}_t)$ be the polynomial obtained from $Q(\mathbf{y}_j)$ after this substitution, which is a polynomial of individual degree less than d on at most $(r-1)$ variables. We can now make the following observations about $P'(\mathbf{y}_t, x_t)$:

- Each $Q^{(t)}(\mathbf{y}_j \cap \mathbf{y}_t)$ has a formula of size at most $(d(r-1)) \cdot d^{r-1}$, and thus $P'(\mathbf{y}_t, x_t)$ has a formula of size at most $(s \cdot (r-1) \cdot d^r)$,
- $\deg(P') \leq D \cdot \deg(Q) \leq D \cdot (kd)$, and $\deg_{x_t}(P') \leq D$,
- $P'(\mathbf{y}_t, Q(\mathbf{y}_t)) \equiv 0$.

The last observation implies that the polynomial $(x_t - Q(\mathbf{y}_t))$ divides P'. Therefore we can write $P' = (x_t - Q(\mathbf{y}_t)) \cdot R$, for some polynomial R. Consider P' and R as univariates in x_t with coefficients as polynomials in $\mathbf{y}_t$:

$$P' = \sum_{i=0}^{D} P'_i \cdot x_t^i \quad , \quad R = \sum_{i=0}^{D-1} R_i \cdot x_t^i.$$

If a is the smallest index such that $P'_a \neq 0$, then $P'_a = R_a \cdot Q(\mathbf{y}_t)$ and hence $Q(\mathbf{y}_t)$ divides P'_a. Any coefficient P'_i can be obtained from P' using interpolation from $(D+1)$ evaluations of x_t. Hence, $\tilde{P} = P'_a$ can be computed in size $(s \cdot (r-1) \cdot d^r \cdot (D+1))$.

For the case of $r = 2$, observe that the polynomial $Q^{(t)}(\mathbf{y}_j \cap \mathbf{y}_t)$ is a univariate of degree at most d. Thus, by Proposition 5.5, $Q^{(t)}(\mathbf{y}_j \cap \mathbf{y}_t)$ can be computed by a formula of size $2d + 1 \leq 4d$. So, we get an upper bound of $(4 \cdot s \cdot d)$ on the formula complexity of $P'(\mathbf{y}_t, x_t)$ (instead of $O(sd^2)$ that we would get by invoking the general bound for $r = 2$) and after interpolation as above, we get a bound of $4 \cdot s \cdot d \cdot (D+1)$ on the formula complexity of P'_a as defined above. □

5.3 Bootstrapping Hitting Sets

The following are the main bootstrapping lemmas to yield our main result. These lemmas follow the same template as in the proof from [AGS19] but with some simple but crucial new ideas that avoid any requirement on bounds on factor complexity, and also permitting a result starting from a barely non-trivial hitting set.

Lemma 5.13 (Barely non-trivial to moderately non-trivial hitting sets). *Let $\varepsilon > 0$ and $n \geq 2$ be constants. Suppose that for all large enough s there is an explicit hitting set of size $s^{n-\varepsilon}$, for all degree s, size s algebraic formulas over n variables.*

Then for a large enough m and for all $s \geq m$, there is an explicit hitting set of size $s^{m/50}$ for all degree s, size s algebraic formulas over m variables.

Lemma 5.14 (Bootstrapping moderately non-trivial hitting sets). *Let n_0 be large enough, and n be any power of two that is larger than n_0. Suppose for all $s \geq n$ there are explicit hitting sets of size $s^{g(n)}$ for $\mathcal{C}(n,s,s)$, the class of n-variate degree s polynomials computed by size s formulas.*

1. *Suppose $g(n) \leq \frac{n}{50}$, then for $m = n^{10}$ and all $s \geq m$, there are explicit hitting sets of size $s^{h(m)}$ for $\mathcal{C}(m,s,s)$ where $h(m) \leq \left(\frac{1}{10}\right) \cdot m^{1/4}$.*

2. *Suppose $g(n) \leq \left(\frac{1}{10}\right) \cdot n^{1/4}$, then for $m = 2^{n^{1/4}}$ and all $s \geq m$, there are explicit hitting sets of size $s^{h(m)}$ for $\mathcal{C}(m,s,s)$ where $h(m) = 20 \cdot \left(g(\log^4 m)\right)^2$.*

   Furthermore, $h(m)$ also satisfies $h(m) \leq \left(\frac{1}{10}\right) \cdot m^{1/4}$.

We will defer the proofs of these lemmas to the end of this section and complete the proof of Theorem 5.1.

Theorem 5.1 (Bootstrapping PIT for algebraic formulas, branching programs and circuits). *Let $\varepsilon > 0$ and $n \geq 2$ be constants. Suppose that, for all large enough s, there is an explicit hitting set of size $s^{n-\varepsilon}$ for all degree s, size s algebraic formulas (algebraic branching programs or circuits respectively) over n variables. Then, there is an explicit hitting set of size $s^{\exp(\exp(O(\log^* s)))}$ for the class of degree s, size s algebraic formulas (algebraic branching programs or circuits respectively) over s variables.*

Proof. Observe that the statements of Lemma 5.13 and Lemma 5.14 are worded to ensure that the conclusion of Lemma 5.13 is precisely the hypothesis of Lemma 5.14(1), the conclusion of Lemma 5.14(1) is precisely the hypothesis of Lemma 5.14(2), and Lemma 5.14(2) admits repeated applications as its conclusion also matches the requirements in the hypothesis. Thus, we can use one application of Lemma 5.13 followed by one application of Lemma 5.14(1) and repeated applications of Lemma 5.14(2) to get hitting sets for polynomials depending on larger and larger sets of variables, until we can get a hitting set for the class $\mathcal{C}(s,s,s)$.

Let n_0 be large enough so as to satisfy the hypothesis of Lemma 5.13, and the two parts of Lemma 5.14. We start with an explicit hitting set of size $s^{n_0-\varepsilon}$ for $\mathcal{C}(n_0,s,s)$ and one application of Lemma 5.13 gives an explicit hitting set of size $s^{n_1/50}$ for $\mathcal{C}(n_1,s,s)$ for $n_1 \geq n_0^8$ and all $s \geq n_1$. Using Lemma 5.14(1) we obtain an explicit hitting set of size $s^{(1/10)\cdot m_0^{1/4}}$ for the class $\mathcal{C}(m_0,s,s)$ for all $s \geq m_0 = n_1^{10}$. We are now in a position to apply Lemma 5.14(2) repeatedly. We now set up some basic notation to facilitate this analysis.

Suppose after i applications of Lemma 5.14(2) we have an explicit hitting set for the class $\mathcal{C}(m_i,s,s)$ of size s^{t_i}. We wish to track the evolution of m_i and t_i. Recall that $m_i = 2^{m_{i-1}^{1/4}}$ after one application of Lemma 5.14(2).

Let $\{b_i\}_i$ be such that $b_0 = \log m_0$ and, for every $i > 0$, let $b_i = 2^{(b_{i-1}/4)}$ so that $b_i = \log m_i$. Similarly to keep track of the complexity of the hitting set, if s^{t_i} is the size of the hitting set for $\mathcal{C}(m_i,s,s)$, then by Lemma 5.14(2) we have $t_0 = \left(\frac{1}{10}\right) m_0^{1/4}$ and $t_i = 20 \cdot t_{i-1}^2$ for all $i > 0$. The following facts are easy to verify.

- $m_i \geq s$ or $b_i \geq \log s$ for $i = O(\log^* s)$,

- for all j, we have $t_j = 20^{(2^j-1)} \cdot t_0^{2^j} = \exp(\exp(O(j)))$.
- the exponent of s in the size of the final hitting set is $t_{O(\log^* s)} = \exp(\exp(O(\log^* s)))$.

Therefore we have an explicit hitting set of size $s^{\exp(\exp(O(\log^* s)))}$ for $\mathcal{C}(s,s,s)$. An explicit algorithm describing the hitting set generator is presented in Section 5.4. □

5.3.1 Proofs of the bootstrapping lemmas

Here we prove the two main lemmas used in the proof of Theorem 5.1. We restate the lemmas here for convenience. The proofs follow a very similar template but with different settings of parameters and minor adjustments.

Lemma 5.13 (Barely non-trivial to moderately non-trivial hitting sets). *Let $\varepsilon > 0$ and $n \geq 2$ be constants. Suppose that for all large enough s there is an explicit hitting set of size $s^{n-\varepsilon}$, for all degree s, size s algebraic formulas over n variables.*

Then for a large enough m and for all $s \geq m$, there is an explicit hitting set of size $s^{m/50}$ for all degree s, size s algebraic formulas over m variables.

Proof. Let $a = \max(n, 250/\varepsilon)$. We begin by fixing the design parameters, $k = n$, $\ell = a \cdot k^4 = a \cdot n^4$ and $r = 2$.

Constructing a suitably hard polynomial: For $B = 5k/\varepsilon$, we construct $Q_k(z_1, \dots, z_k)$ that vanishes on the hitting set for all size s^B degree s^B formulas over k variables, that has size $s^{B(k-\varepsilon)}$ using Lemma 5.10. The polynomial $Q_k(\mathbf{z})$ has the following properties.

- Q_k has individual degree $d < s^{B(k-\varepsilon)/k}$, and total degree $< k \cdot s^{B(k-\varepsilon)/k}$.
- Q_k is not computable by formulas of size s^B.
- Q_k has a formula of size $\leq (kd) \cdot s^{B(k-\varepsilon)}$.

Building the NW design: Using Lemma 5.8, we now construct an (ℓ, k, r) design $\{S_1, \dots, S_m\}$ with $m := \left(\frac{\ell}{k}\right)^r = \left(ak^{(4-1)}\right)^2 = a^2k^6$.

Variable reduction: Let $P(x_1, \dots, x_m)$ be a nonzero m-variate polynomial of degree s that is computable by a formula of size s, and let $P(Q_k[\![\ell, k, r]\!]_{\mathrm{NW}}) \equiv 0$. Then, from the 'moreover' part of Lemma 5.12 (since $r = 2$), we get that there is a polynomial $\tilde{P}(z_1, \dots, z_k)$ that vanishes on a hitting set for formulas of size s^B and degree s^B, and is computable by a formula of size at most

$$\begin{aligned}
\mathrm{size}(\tilde{P}) &\leq 4 \cdot s \cdot d \cdot (s+1) \\
&\leq 4s(s+1) \cdot s^{B(k-\varepsilon)/k} \\
&\leq s^{\left(5 + \frac{B(k-\varepsilon)}{k}\right)} = s^{5 + \frac{5k}{\varepsilon} - 5} = s^B.
\end{aligned}$$

Moreover, note that the degree of $\tilde{P}(z_1, \ldots, z_k)$ is at most $(k \cdot d) \cdot s \leq s^{\left(2+\frac{B(k-\varepsilon)}{k}\right)} < s^B$. Since $\tilde{P}$ vanishes on the hitting set for formulas of size s^B and degree s^B, we get a contradiction due to Observation 5.9. Therefore it must be the case that $P(Q_k[\![\ell, k, r]\!]_{\mathrm{NW}})$ is nonzero.

Construction of the hitting set: Therefore, starting with a nonzero formula of degree s, size s, over m variables, we obtain a nonzero ℓ-variate polynomial of degree at most $s \cdot (kd) \leq s^B$. At this point we can just use the trivial hitting set given by the polynomial identity lemma [Ore22, DL78, Zip79, Sch80], which has size at most $s^{B\ell}$.

Therefore what remains to show is that our choice of parameters ensures that $B\ell < \frac{m}{50}$. This is true, as $\frac{m}{50} = \frac{a^2k^6}{50} = \frac{ak}{50} \cdot ak^5 = \left(\frac{5k}{\varepsilon}\right) \cdot \ell \cdot k > B\ell$.

The construction runs in time that is polynomial in the size of the hitting set in the conclusion, and the *bit-size* of the points in it. See Section 5.4 for a more elaborate discussion. □

Lemma 5.14 (Bootstrapping moderately non-trivial hitting sets). *Let n_0 be large enough, and n be any power of two that is larger than n_0. Suppose for all $s \geq n$ there are explicit hitting sets of size $s^{g(n)}$ for $\mathcal{C}(n, s, s)$, the class of n-variate degree s polynomials computed by size s formulas.*

1. *Suppose $g(n) \leq \frac{n}{50}$, then for $m = n^{10}$ and all $s \geq m$, there are explicit hitting sets of size $s^{h(m)}$ for $\mathcal{C}(m, s, s)$ where $h(m) \leq \left(\frac{1}{10}\right) \cdot m^{1/4}$.*
2. *Suppose $g(n) \leq \left(\frac{1}{10}\right) \cdot n^{1/4}$, then for $m = 2^{n^{1/4}}$ and all $s \geq m$, there are explicit hitting sets of size $s^{h(m)}$ for $\mathcal{C}(m, s, s)$ where $h(m) = 20 \cdot \left(g(\log^4 m)\right)^2$.*

   Furthermore, $h(m)$ also satisfies $h(m) \leq \left(\frac{1}{10}\right) \cdot m^{1/4}$.

Proof. The proofs of both parts follow the same template as in the proof of Lemma 5.13 but with different parameter settings. Hence, we will defer the choices of the parameters ℓ, k, r towards the end to avoid further repeating the proof. For now, let ℓ, k, r be parameters that satisfy $r \leq k$, $\ell = k^2$ and $5r \cdot g(n) \leq k$.

Constructing a hard polynomial: The first step is to construct a polynomial $Q_k(z_1, \ldots, z_k)$ that vanishes on the hitting set for the class $\mathcal{C}(n, s^5, s^5)$, where[7] $k \leq n$. This can be done by using Lemma 5.10. The polynomial $Q_k(\mathbf{z})$ will therefore have the following properties.

- Q_k has individual degree d smaller than $s^{5g(n)/k}$, and degree at most $k \cdot s^{5g(n)/k}$.
- Computing Q_k requires formulas of size more than s^5.
- Q_k has a formula of size at most $s^{10g(n)}$.

Building the NW design: Using the parameters ℓ, k, r, and the construction from Lemma 5.8, we now construct an (ℓ, k, r) design $\{S_1, \ldots, S_m\}$ with $m \leq k^r$.

[7]that is, Q_k is a k-variate polynomial that is just masquerading as an n-variate polynomial that does not depend on the last $n - k$ variables.

Variable reduction using Q_k: Let $P(x_1, \ldots, x_m) \in \mathcal{C}(m, s, s)$ be a nonzero polynomial. Suppose $P(Q_k[\![\ell, k, r]\!]_{\mathrm{NW}}) \equiv 0$, then Lemma 5.12 states that there is a nonzero polynomial $\tilde{P}(z_1, \ldots, z_k)$ of degree at most $s \cdot k \cdot d$ such that Q_k divides $\tilde{P}$, and that $\tilde{P}$ can be computed by a formula of size at most

$$\begin{aligned} s \cdot (r-1) \cdot d^r \cdot (s+1) &\leq s^4 \cdot d^r \\ &\leq s^4 \cdot s^{5r \cdot g(n)/k} \\ &\leq s^5. && \text{(since } k, r \text{ satisfy } 5r \cdot g(n) \leq k\text{)} \end{aligned}$$

Furthermore, the degree of $\tilde{P}$ is at most $s \cdot r \cdot s^{5g(n)/k} \leq s^5$. Hence, $\tilde{P}$ is a polynomial on $k \leq n$ variables, of degree at most s^5 that vanishes on the hitting set of $\mathcal{C}(n, s^5, s^5)$ since Q_k divides $\tilde{P}$. But then, Observation 5.9 states that $\tilde{P}$ must require formulas of size more than s^5, contradicting the above size bound. Hence, it must be the case that $P(Q_k[\![\ell, k, r]\!]_{\mathrm{NW}}) \not\equiv 0$.

Hitting set for $\mathcal{C}(m, s, s)$: At this point, we set the parameters k and r depending on how quickly $g(n)$ grows.

Part (1) $\left(g(n) \leq \frac{n}{50}\right)$**:** In this case, we choose $k = n$ and $r = 10$ (so we satisfy $5r \cdot g(n) \leq n = k$). From Lemma 5.8, we have an explicit (ℓ, k, r) design $\{S_1, \ldots, S_m\}$ with $m = k^r = n^{10}$.

For any nonzero $P \in \mathcal{C}(m, s, s)$, we have that $P(Q_k[\![\ell, k, r]\!]_{\mathrm{NW}})$ is a nonzero ℓ-variate polynomial of degree at most $s \cdot k \cdot s^{5g(n)/k} \leq s^3$. Hence, by just using the trivial hitting set via the polynomial identity lemma [Ore22, DL78, Sch80, Zip79], we have an explicit hitting set of size $s^{3\ell} \leq s^{3m^{1/5}}$. Since $m \geq n_0$ and n_0 is large enough, we have that

$$h(m) := 3m^{1/5} \leq \left(\frac{1}{10}\right) \cdot m^{1/4}.$$

Part (2) $\left(g(n) \leq \left(\frac{1}{10}\right) n^{1/4}\right)$**:** In this case, we choose $k = \sqrt{n}$ and $r = n^{1/4}$, so that $5r \cdot g(n) \leq 10r \cdot g(n) \leq k$ and $\ell = n$. Using Lemma 5.8, we now construct an explicit (ℓ, k, r) design $\{S_1, \ldots, S_m\}$ with $m = 2^{n^{1/4}} \leq k^r$.

We have a formula computing the n-variate polynomial $P(Q_k[\![\ell, k, r]\!]_{\mathrm{NW}})$ of size at most $s \cdot s^{10g(n)} \leq s^{20g(n)} =: s'$. Using the hypothesis for hitting sets for $\mathcal{C}(n, s', s')$, we have an explicit hitting set for $\mathcal{C}(m, s, s)$ of size at most

$$(s')^{g(n)} = s^{20g(n)^2} = s^{h(m)},$$

where $h(m) = 20\left(g((\log m)^4)\right)^2$. Since n_0 is large enough, we have that

$$\begin{aligned} 10 \cdot h(m) &\leq 20 \cdot 10 \cdot \left(g((\log m)^4)\right)^2 \\ &\leq 2(\log m)^2 && \left(\text{since } g(n) \leq \left(\frac{1}{10}\right) n^{1/4}\right) \\ &\leq m^{1/4}. && (\text{since } m \geq n_0 \text{ and } n_0 \text{ is large enough}) \end{aligned}$$

This completes the proof of both parts of the lemma. □

5.4 Algorithm for generating the hitting set

We now give an algorithm to generate an explicit hitting set for $\mathcal{C}(s,s,s)$, for all large s, using the hypothesis of Theorem 5.1. Let n_0 be the initial threshold from the hypothesis and let n_1 be a constant that satisfies the "large enough" requirements of Lemma 5.14.

$$\begin{aligned} & n_0 \geq 2 && t_0 = (n_0 - \varepsilon) \\ & n_1 \text{ is large enough} && t_1 = n_1/50 \\ & n_2 = n_1^{10} && t_2 = (1/10)\, n_2^{1/4} \\ \text{For all } i \geq 3, \quad & n_i = 2^{n_{i-1}^{1/4}} && t_i := 20t_{i-1}^2 \end{aligned}$$

We are provided an algorithm Initial-Hitting-Set(s) that outputs a hitting set for $\mathcal{C}(n_0,s,s)$ of size at most $s^{n_0-\varepsilon}$. Algorithm 1 describes a function Hitting-Set which, given inputs i and s, outputs a hitting set for $\mathcal{C}(n_i,s,s)$ of size at most s^{t_i} in time poly(s^{t_i}).

From the growth of n_i, it follows that $n_b \geq s$ for $b = O(\log^* s)$ and $t_i = 20^{2^i-1}t_0^{2^i}$. Unfolding the recursion for Hitting-Set(j,s), for any j, the algorithm makes at most 2^j calls to Initial-Hitting-Set(s') for various sizes s' satisfying

$$s' \leq s^{B\cdot 20^{j-1}\prod_{i=1}^{j-1} t_i} \leq s^{Bt_{j-1}^2} = s^{O(t_j)}.$$

Thus for Hitting-Set(b,s), the algorithm makes at most 2^b calls to Initial-Hitting-Set(s'), for sizes s' that are at most $s^{\exp(\exp(O(\log^* s)))}$. The overall running time is polynomial time in the size of the final hitting set which is $s^{t_b} = s^{\exp(\exp(O(\log^* s)))}$.

Bit complexity of the hitting sets. We will now discuss the bit complexity of the hitting sets that are generated during the bootstrapping procedure. We will analyse Algorithm 1 and show that any hitting set H for n-variate formulas that the algorithm outputs, will have a bit complexity that is at most $|H|^{f(n)}$, for $f(n) = \exp(O(\log^* n))$.

Let us first consider the case when $i \geq 3$. Suppose each evaluation point in $H_0 = H_{i-1}(s^5)$ is at most h^a bits long, where $h = |H_0|$ and $a = f(n_{i-1})$. Using Remark 5.11, we get that each coefficient of Q is at most h^{ca} bits long, for some other constant c. The output of Algorithm 1 for this case will be evaluations of Q on $H_1 = H_{i-1}(s^{20t_{i-1}})$. Since $|H_1| = h^{4t_{i-1}}$,

```
Algorithm 1: Hitting-Set
Input  : Parameter $i$ and a size $s$.
Output: A hitting set of size $s^{t_i}$ size for $\mathcal{C}(n_i, s, s)$.
1  if $i = 1$ then
2      Let $A = \max(n_0, 250/\varepsilon)$, $B = 3n_0/\varepsilon$
3      Let $H_0(s^B) :=$ Initial-Hitting-Set$(s^B)$          // size at most $s^{B(n_0 - \varepsilon)}$
4      Compute a nonzero polynomial $Q$ on $k = n_0$ variables of individual degree
         smaller than $s^{Bt_0/k}$ that vanishes on $H_0(s^B)$.          // takes poly$(s^{Bt_0})$ time
5      Compute an $(A \cdot n_0^4, n_0, 2)$-design $\{S_1, \ldots, S_{n_1}\}$.
6      Let $S \subseteq \mathbb{F}$ be of size at least $s^B$.
7      return $\left\{ (Q[\![\ell, k, r]\!]_{\mathrm{NW}})(\mathbf{a}) \ : \ \mathbf{a} \in S^{An_0^5} \right\}$          // size at most $s^{BAn_0^5} \leq s^{n_1/50} = s^{t_1}$
8  else if $i = 2$ then
9      $H_1(s^5) :=$ Hitting-Set$(1, s^5)$          // size at most $s^{5t_1}$
10     Compute a nonzero polynomial $Q$ on $k = n_1$ variables of individual degree
         smaller than $s^{5t_1/k}$ that vanishes on $H_1(s^5)$.          // takes poly$(s^{5t_1})$ time
11     Compute an $(n_1^2, n_1, 10)$-design $\{S_1, \ldots, S_{n_2}\}$.
12     Let $S \subseteq \mathbb{F}$ be of size at least $s^3$.
13     return $\left\{ (Q[\![\ell, k, r]\!]_{\mathrm{NW}})(\mathbf{a}) \ : \ \mathbf{a} \in S^{n_1^2} \right\}$          // size at most $s^{3n_1^2} \leq s^{0.1 \cdot n_2^{1/4}} = s^{t_2}$
14 else if $i \geq 3$ then
15     $H_{i-1}(s^5) :=$ Hitting-Set$(i-1, s^5)$          // size at most $s^{5t_{i-1}}$
16     Compute a nonzero polynomial $Q$ on $k = \sqrt{n_{i-1}}$ variables of individual degree
         smaller than $s^{5t_{i-1}/k}$ that vanishes on $H_{i-1}(s^5)$.          // takes poly$(s^{5t_{i-1}})$ time
17     Compute an $(n_{i-1}, \sqrt{n_{i-1}}, \sqrt[4]{n_{i-1}})$-design $\{S_1, \ldots, S_{n_i}\}$
18     $H_{i-1}(s^{20t_{i-1}}) :=$ Hitting-Set$(i-1, s^{20t_{i-1}})$          // size at most $s^{20t_{i-1}^2}$
19     return $\{ (Q[\![\ell, k, r]\!]_{\mathrm{NW}})(\mathbf{a}) \ : \ \mathbf{a} \in H_{i-1}(s^{20t_{i-1}}) \}$          // size at most $s^{20t_{i-1}^2} = s^{t_i}$
```

the bit complexity of H_1 is at most $h^{4at_{i-1}}$. Now Q is a degree $< s^5$ polynomial with $O(h)$ monomials. As a result any evaluation of Q on a point from H_1 will have at most $O(\log h) \cdot \left(h^{ca} + s^5 \cdot h^{4at_{i-1}}\right) \leq h^{2a\cdot(4t_{i-1})} = |H_1|^{2a}$, as t_{i-1} is large enough. Since $n_i = 2^{n_{i-1}^{1/4}}$, we have that $f(n) = \exp(O(\log^* n))$. For the cases when $i = 1$ or $i = 2$, since we use the trivial hitting sets in place of H_1 in the above discussion, the above bounds will continue to hold.

We want to remark that although the bit complexity of the output of Hitting-Set(i, s) is not polynomial in the size s^{t_i}, the exponent t_i is always larger than $f(n_i)$. As a result the time taken to generate the final hitting set in the conclusion of Theorem 5.1 can still be bounded by $s^{\exp(\exp(O(\log^* s)))}$, albeit for a slightly larger constant.

Note that the successive outputs of the procedure are obtained by repeatedly composing hard polynomials on a hitting set. As a result, with each composition the size of the hitting set stays the same whereas the same cannot be said about the bit complexity. It is therefore unlikely that the bit complexity will depend solely on the size of the hitting set.

6 | On the Existence of Algebraically Natural Proofs

This chapter deals with the question *"do strong circuit lower bounds admit easy proofs?"*, a question that has received much attention in the recent years, due to the lack of strong circuit lower bounds despite significant progress in more structured settings[1].

6.1 Introduction

In the context of proving algebraic circuit lower bounds, perhaps the only thing more frustrating than the inability to prove such lower bounds is the inability to come up with plausible approaches towards them. This lack of progress on the problem and a dearth of potential approaches towards it has spurred some work towards understanding the viability of some of the current lower bound approaches; the idea being that a good sense of what approaches will *not* work would aid in the search of approaches that *might* work.

In the broader context of lower bounds in computational complexity, there are various results of this flavour which establish that various families of techniques cannot be used for proving very strong lower bounds, e.g., the barrier of *Relativisation* due to Baker, Gill and Solovay [BGS75], that of *Algebraization* due to Aaronson and Wigderson [AW09] and that of *Natural Proofs* due to Razborov and Rudich [RR97] [2]. While none of these barrier results are directly applicable to the setting of algebraic computation, there have been recent attempts towards generalizing these ideas to the algebraic set up. A key notion in this line of work is the notion of *algebraically natural proofs* alluded to and defined in the works of Aaronson and Drucker [AD08], Forbes, Shpilka and Volk [FSV18], and Grochow, Kumar, Saks and Saraf [GKSS17].

We now discuss this notion, starting with a discussion of *Natural Proofs* which motivated the definition.

[1]The results in this chapter have appeared in [CKR+20].
[2]Sometimes, these results are conditional, as in [RR97].

6.1.1 The Natural Proofs framework of Razborov and Rudich

Razborov and Rudich [RR97] noticed that underlying many of the lower bound proofs known in Boolean circuit complexity, there was some common structure. They formalized this common structure via the notion of a *Natural Property*, which we now define.

Definition 6.1. *A subset $\mathcal{P} \subseteq \{f : \{0,1\}^n \to \{0,1\}\}$ of Boolean functions is said to be a natural property useful against a class $\mathcal{C}$ of Boolean circuits if the following are true.*

- **Usefulness.** *Any Boolean function $f : \{0,1\}^n \to \{0,1\}$ that can be computed by a Boolean circuit in $\mathcal{C}$ does not have the property $\mathcal{P}$.*

- **Constructivity.** *Given the truth table of a Boolean function $f : \{0,1\}^n \to \{0,1\}$, whether or not it has the property $\mathcal{P}$ can be decided in time polynomial in the length of the input, that is in time $2^{O(n)}$.*

- **Largeness.** *For all large enough n, at least a $2^{-O(n)}$ fraction of all n variate Boolean functions have the property $\mathcal{P}$.* $\Diamond$

A proof that a certain family of Boolean functions cannot be computed by circuits in $\mathcal{C}$ is said to be a *natural lower bound proof* if the proof (perhaps implicitly) proceeds via establishing a natural property useful against $\mathcal{C}$, and showing that the candidate hard function has this property. Razborov and Rudich then showed that most of the Boolean circuit lower bound proofs that we know, e.g., lower bounds for AC^0 circuits [FSS84, Hås86] or lower bounds for $\mathsf{AC}^0[\oplus]$ circuits [Raz87, Smo87] fit into this framework (maybe with some work) and hence are *natural* in this sense. Further, they argue that under standard cryptographic assumptions, the proof of a lower bound against any sufficiently rich circuit class (such as the class P/poly) cannot be natural! Thus, under standard cryptographic assumptions, most of the current lower bound techniques are not strong enough to show super-polynomial lower bounds for general Boolean circuits.

We now move on to discuss a relatively recent analogue of the notion of Natural Proofs, formalized in the context of algebraic computation.

6.1.2 Algebraically Natural Proofs

Considering that algebraic circuits seem like a fairly general and powerful model of computation, it is tempting to think that the *natural proofs barrier* of Razborov and Rudich [RR97] also extends to this setting. This problem turns out to be a non-trivial one, and indeed it is not known whether their results extend to algebraic circuits. This question is closely related to the question of whether cryptographically secure algebraic pseudorandom functions can be computed by small and low degree algebraic circuits and there does not seem to be substantial evidence one way or the other on this [3].

[3]Refer to [AD08] and [FSV18] for a more detailed discussion on this issue.

In the last few years, this question of trying to find an algebraic analogue of the barrier results in boolean circuits [RR97] has received substantial attention. It was observed by various authors ([AD08, Gro15, FSV18, GKSS17]) that most of the currently known proofs of algebraic circuit lower bounds fit into a common unifying framework, not unlike that in the boolean setting [RR97], although of a more algebraic nature. Indeed, these proofs also implicitly go via defining a property for the set of all polynomials and using this property to separate the hard polynomial from the easy ones. Moreover, the notions of *largeness* and *constructivity* in Definition 6.1 also seem to extend to these proofs.

We now discuss this framework in a bit more detail. The key notion here is that of an equation of polynomials in a complexity class.

Definition 6.2 (Equations for a Class of Polynomials)**.** *For some $n, d \in \mathbb{N}$, let $\mathcal{C}_{n,d}$ be a class of n-variate polynomials of* total *degree at most d; i.e.* $\mathcal{C}_{n,d} \subseteq \mathbb{F}[\mathbf{x}]^{\leq d}$.

Then for $N = \binom{n+d}{n}$, a nonzero polynomial $P_N(\mathbf{Z})$ is said to be an equation *for $\mathcal{C}_{n,d}$ if for all $f(\mathbf{x}) \in \mathcal{C}_{n,d}$, we have that $P_N(\overline{\mathrm{coeff}}(f)) = 0$, where $\overline{\mathrm{coeff}}(f)$ is the coefficient vector of f.* $\diamondsuit$

The definition naturally extends to a class of polynomial families, as opposed to just a class of polynomials as defined above. In particular, suppose that $\mathcal{C}$ is a class of polynomial families $\{\{f_n\} : f_n \in \mathcal{C}_{n,d_n}\}$, and $\{P_N\}$ is a polynomial family. Then, the family $\{P_N\}$ is said to be a family of equations for $\mathcal{C}$ if there is an n_0, such that for all $n \geq n_0$ the polynomial P_N is an equation for $\mathcal{C}_{n,d_n}$ where $N = \binom{n+d_n}{n}$. That is, P_N is an equation for $\mathcal{C}_{n,d_n}$ for all large enough n.

Intuitively, non-vanishing of an equation (for a class $\mathcal{C}$) on the coefficient vector of a given polynomial f is a proof that f is not in $\mathcal{C}$. We note that the equations for a class $\mathcal{C}$ evaluate to zero not just on the coefficient vectors of polynomials in $\mathcal{C}$ but also on the coefficient vectors of polynomials in the Zariski closure of $\mathcal{C}$. This framework comes up very naturally in the context of algebraic geometry (and geometric complexity theory), where it is often geometrically nicer to work with the variety obtained by taking the Zariski closure of a complexity class.

Getting our hands on an equation of a variety gives us a plausible way to test and certify non-membership in the variety, in other words, to prove a lower bound for the corresponding complexity class. Thus, equations for a class gives an algebraic analogue of the notion of *natural properties useful against a class* in [RR97]. Moreover, since a nonzero polynomial does not vanish very often on a random input from a large enough grid, it follows that a nonzero equation for a class $\mathcal{C}$ will be nonzero on the coefficient vector of a "random polynomial". Here by a random polynomial we mean a polynomial whose coefficients are independent and uniformly random elements from some large enough set in the underlying field. With appropriate quantitative bounds, this observation can be formalized to give an appropriate algebraic analogue of the notion of *largeness*. Lastly, the algebraic circuit complexity of the equation gives a natural algebraic analogue of the notion of *constructivity*. Intuitively, any algebraic circuit lower bound which goes via defining a nonzero proof polynomial of polynomially bounded degree that can be efficiently computed by an algebraic circuit is an

Algebraically Natural Proof of a lower bound.

We now formally define an algebraically natural proof.

Definition 6.3 (Algebraically natural proofs [FSV18, GKSS17]). *Let $\mathcal{C}$ be a class of polynomial families $\{\{f_{n,d}\} : f_{n,d} \in \mathcal{C}_{n,d}\}$.*

Then, for a class $\mathcal{D}$ of polynomial families, we say that $\mathcal{C}$ has $\mathcal{D}$-natural proofs *if there is a family $\{P_N\} \in \mathcal{D}$ which is a non-trivial family of equations for $\mathcal{C}$.* $\Diamond$

In the rest of this chapter, whenever we say a natural proof, without specifying the class $\mathcal{D}$, we mean a VP-natural proof.

Analogous to the abstraction of *natural proofs* for Boolean circuit lower bounds, this framework of *algebraically natural proofs* turns out to be rich and general enough that almost all of our current proofs of algebraic circuit lower bounds are in fact algebraically natural, or can be viewed in this framework with a little work [Gro15]. Thus, this definition seems like an important first step towards understanding the strengths and limitations of many of our current lower bound techniques in algebraic complexity.

The immediate next question is whether algebraically natural proofs are rich enough to give strong algebraic circuit lower bounds. This can naturally be worded in terms of the complexity of equations for the class VP as follows.

Question 6.4. *For every constant $c > 0$, does there exist a nonzero polynomial family $\{P_{N,c}\}$ in* VP *such that for all large enough n, the following is true?*

> *For every family of polynomials $\{f_n\}_n$ in* VP*, such that f_n is an n variate polynomial of degree n^c, $P_{N,c}$ vanishes on the coefficient vector of f_n for $N = \binom{n+n^c}{n}$.*

Forbes, Shpilka and Volk [FSV18], and Grochow, Kumar, Saks and Saraf [GKSS17] argue that under an appropriate (but non-standard) pseudorandomness assumption, the answer to the question above is negative, that is, algebraically natural proof techniques cannot be used to show strong lower bounds for algebraic circuits. To discuss this pseudorandomness assumption formally, we need the following definition of *succinct hitting sets*.

Definition 6.5 (Succinct hitting sets for a class of polynomials (Informal)). *For some $n, d \in \mathbb{N}$, let $\mathcal{C}_{n,d}$ be a class of n-variate polynomials of* total *degree at most d; that is, $\mathcal{C}_{n,d} \subseteq \mathbb{F}[\mathbf{x}]^{\leq d}$.*

Then for $N = \binom{n+d}{n}$, we say that a class of N variate polynomials $\mathcal{D}_N$ has $\mathcal{C}_{n,d}$-succinct hitting sets *if for all $0 \not\equiv P(\mathbf{Z}) \in \mathcal{D}_N$, there exists some $f \in \mathcal{C}_{n,d}$ such that $P_N(\overline{\text{coeff}}(f)) \neq 0$.* $\Diamond$

As with Definition 6.2, this definition naturally extends to polynomial families.

It immediately follows from the definitions that non-existence of $\mathcal{D}$-natural proofs against a class $\mathcal{C}$ is equivalent to the existence of $\mathcal{C}$-succinct hitting sets for the class $\mathcal{D}$. Forbes, Shpilka and Volk [FSV18] showed that for various restricted circuit classes $\mathcal{C}$ and $\mathcal{D}$, the class $\mathcal{D}$ has $\mathcal{C}$ succinct hitting sets. Or equivalently, lower bounds for $\mathcal{C}$ cannot be proved via proof polynomial families in $\mathcal{D}$. However, this question has remained unanswered for more general circuit classes $\mathcal{C}$ and $\mathcal{D}$. In particular, if we take both $\mathcal{C}$ and $\mathcal{D}$ to be VP, we do not seem to have significant evidence on the existence of VP succinct hitting sets for VP. In [FSV18], the authors observed that showing VP succinct hitting sets for VP would immediately imply non-trivial deterministic algorithms for polynomial identity testing, which via well known connections between algebraic hardness and derandomisation will in turn imply new lower bounds [HS80, KI04]. Thus, the problem of proving an unconditional barrier result for algebraically natural proof techniques via this route seems as hard as proving new circuit lower bounds! It is, however, conceivable that one can show such a barrier conditionally. And in some more structured settings, such as for the case of matrix completion, such results are indeed known [BIJL18]. However, Question 6.4 continues to remain open. In particular, even though many of the structured subclasses of VP have low degree equations which are very efficiently computable, perhaps hoping that this extends to richer and more general circuit classes is too much to ask for?

We are now ready to state our results.

6.1.3 Results in this chapter

In our main results, we make progress towards answering Question 6.4 in the affirmative. We prove the following theorems.

Equations for polynomials in VP with coefficients of small complexity

Theorem 6.6. *Let $c > 0$ be any constant. There is a polynomial family $\{P_{N,c}\} \in \mathsf{VP}_{\mathbb{Q}}$ such that for all large n and $N = \binom{n+n^c}{n}$, the following are true.*

- *For every family $\{f_n\} \in \mathsf{VP}_{\mathbb{C}}$, where f_n is an n variate polynomial of degree at most n^c and coefficients in $\{-1, 0, 1\}$, we have*

$$P_{N,c}(\overline{\mathrm{coeff}}(f_n)) = 0\,,$$

  where $\overline{\mathrm{coeff}}(f_n)$ is the coefficient vector of f_n.

- *There exists a family $\{h_n\}$ of n variate polynomials and degree at most n^c with coefficients in $\{-1, 0, 1\}$ such that*

$$P_{N,c}(\overline{\mathrm{coeff}}(h_n)) \neq 0\,.$$

Here for a field $\mathbb{F}$, $\mathsf{VP}_{\mathbb{F}}$ denotes the class VP where the coefficients of the polynomials are from the field $\mathbb{F}$.

We remark that even though Theorem 6.6 is stated for polynomials with coefficients in $\{-1, 0, 1\}$, the theorem holds for polynomials with coefficients as large as $\mathrm{poly}(N)$. However, for brevity, we will confine the discussion in this chapter to polynomials with $\{-1, 0, 1\}$ coefficients.

We also prove an analogous theorem for finite fields of small size.

Theorem 6.7. *Let $\mathbb{F}$ be any finite field of constant size and $c > 0$ be any constant. There is a polynomial family $\{P_{N,c}\} \in \mathsf{VP}_{\mathbb{F}}$ such that for all large enough n and $N = \binom{n+n^c}{n}$, the following are true.*

- *For every $\{f_n\} \in \mathsf{VP}_{\mathbb{F}}$, where f_n is an n variate polynomial of degree at most n^c, we have*

$$P_{N,c}(\overline{\mathrm{coeff}}(f_n)) = 0\,.$$

- *There exists a family $\{h_n\}$ of n variate polynomials and degree at most n^c with coefficients in $\mathbb{F}$ such that*

$$P_{N,c}(\overline{\mathrm{coeff}}(h_n)) \neq 0\,.$$

Furthermore, we also prove analogous statements for the larger class VNP, which we now state.

Equations for polynomials in VNP with coefficients of small complexity

Theorem 6.8. *Let $c > 0$ be any constant. There is a polynomial family $\{Q_{N,c}\} \in \mathsf{VP}_{\mathbb{Q}}$ such that for all large n and $N = \binom{n+n^c}{n}$, the following are true.*

- *For every family $\{f_n\} \in \mathsf{VNP}_{\mathbb{C}}$, where f_n is an n variate polynomial of degree at most n^c and coefficients in $\{-1, 0, 1\}$, we have*

$$Q_{N,c}(\overline{\mathrm{coeff}}(f_n)) = 0\,.$$

- *There exists a family $\{h_n\}$ of n variate polynomials and degree at most n^c with coefficients in $\{-1, 0, 1\}$ such that*

$$Q_{N,c}(\overline{\mathrm{coeff}}(h_n)) \neq 0\,.$$

Theorem 6.9. *Let $\mathbb{F}$ be any finite field of constant size and $c > 0$ be any constant. There is a polynomial family $\{Q_{N,c}\} \in \mathsf{VP}_{\mathbb{F}}$ such that for all large enough n and $N = \binom{n+n^c}{n}$, the following are true.*

- *For every family $\{f_n\} \in \mathsf{VNP}_{\mathbb{F}}$, where f_n is an n variate polynomial of degree at most n^c, we have*

$$Q_{N,c}(\overline{\mathrm{coeff}}(f_n)) = 0\,.$$

- *There exists a family $\{h_n\}$ of n variate polynomials and degree at most n^c with coefficients in $\mathbb{F}$ such that*

$$Q_{N,c}(\overline{\mathrm{coeff}}(h_n)) \neq 0\,.$$

6.1.4 Discussion and relations to prior work

As is evident from the statements, our main theorems make some progress towards answering Question 6.4 in the affirmative, at least in the setting of small finite fields and for polynomials with small integer coefficients, in a fairly strong sense. In fact, as Theorem 6.8 and Theorem 6.9 show, in the context of polynomials with coefficients of low complexity, not just VP but even the seemingly larger class VNP has efficiently computable low degree[4] equations.

Many of the families of polynomials commonly studied in algebraic complexity have integer coefficients with absolute values bounded by 1, and fall in the setting of the results here. Moreover, the condition of computing polynomials with bounded coefficients is a semantic condition on a model, in the sense that even though the final output of the circuit is required to have bounded coefficients, the circuit is free to use arbitrary constants from $\mathbb{C}$ in the intermediate computation. Thus, it is conceivable that we might be able to prove a super-polynomial lower bound on the algebraic circuit size for the permanent polynomial via an algebraically natural proof constructible in VP, thereby separating VP and VNP. However, since analogues of Theorem 6.6 and Theorem 6.7 are also true for VNP, any such separation of VNP and VP will have to rely on more fine grained information on the equations, and not just their degree and algebraic circuit size. Unfortunately, our proofs are all existential and do not give a sense of what the polynomial families $\{P_{N,c}\}$ (or $\{Q_{N,c}\}$) might look like.

We also note that in the light of some of the prior work, the results here are perhaps a bit surprising. The classes of polynomials in VP and VNP with small coefficients (or over small finite fields), are seemingly rich and complex sets, and the main theorems here show (unconditionally) that they have equations which are also efficiently computable. As discussed earlier in this introduction, this property is known to be true for many structured subclasses of algebraic circuits (for example, homogeneous circuits of depth 3 and 4, multilinear formulas, polynomials of small Waring rank). However, it is unclear if this property extends to more general circuit classes, in particular VP (or VNP).

Indeed, following the [FSV18] and [GKSS17], much of the research on this problem (e.g. [FSV18, BIJL18, BIL$^+$19]) has focused on proving the *non-existence* of efficiently computable equations for VP, and this line of work has made interesting progress in this direction for many structured and special instances of problems of this nature. The results in [BIJL18, BIL$^+$19] draw connections between the existence of efficiently constructible equations characterising a variety and the problem of testing (non)membership in it and use the conditional hardness of the (non)membership testing problem for certain varieties to rule out the existence of efficiently computable equations for them. More precisely, Bläser, Ikenmeyer, Jindal and Lysikov [BIJL18] show that if all the defining equations for the variety of matrices with zero permanent are constructible by small constant-free algebraic circuits, then the non-membership problem for this variety can be decided in the class $\exists$BPP. Thus, unless $\mathsf{P}^{\#\mathsf{P}} \subseteq \exists\mathsf{BPP}$, the defining equations of this variety do not have small, low degree constant

[4]Throughout this chapter, by a *low degree* polynomial family, we mean a polynomial family whose degree is polynomially bounded in its number of variables.

free algebraic circuits. In a subsequent work [BIL$^+$19], the results of [BIJL18] are generalized to *min-rank* or *slice-rank* varieties. However, in the bounded coefficient setting (and over small finite fields), our results show that the contrary is true, and VP does have efficiently computable low degree equations. We also remark that because of the setting of bounded integer coefficients or small finite fields in this work, this natural connection between variety non-membership and defining equations of varieties discussed in [BIJL18, BIL$^+$19] appears to break down.

A positive result on the complexity of equations of naturally occurring varieties in algebraic complexity appears in a recent work of Kumar and Volk [KV20] where they show polynomial degree bounds on the equations of the Zariski closure of the set of non-rigid matrices and small linear circuits over all large enough fields. However, we do not know if any of these low degree equations can be *efficiently* computed by an algebraic circuit.

As alluded to in the previous paragraphs, most of the prior work related to Question 6.4 has focused on looking for evidence that the answer to it is negative, that is VP does not have efficiently computable and low degree equations. We hope that the results in this chapter also highlight the possibility of there being interesting upper bounds for the equations for rich and powerful algebraic complexity classes; a line of research that hasn't received much attention so far.

Other related work. Many of the algebraic circuit lower bounds (e.g. lower bounds for depth-3 and depth-4 circuits, and lower bounds for multilinear models) are obtained by considering the rank of certain matrices as a complexity measure. In their recent works, Efremenko, Garg, Oliveira and Wigderson [EGOW18] and Garg, Makam, Oliveira and Wigderson [GMOW19], discuss limitations of *rank based methods* towards proving lower bounds. In particular, [EGOW18] shows that some of these rank based methods cannot prove lower bounds better than $\Omega_d(n^{\lfloor d/2 \rfloor})$ on tensor rank (respectively Waring rank) for an order d tensor of side-length n. Building on [EGOW18], in [GMOW19], the authors demonstrate that one *cannot* hope to significantly improve the known lower bounds for tensor rank for order d tensors by merely lifting lower bounds on tensors of lower order. However, we note that a general algebraically natural proof of a lower bound does not necessarily fit into the framework of [EGOW18, GMOW19], and so these limitations for the so called *rank methods* do not seem to immediately extend to algebraically natural proofs in general. As discussed earlier, in the light of the results here, it is conceivable that we might be able to improve the state of the art for general algebraic circuit lower bounds, using techniques that are algebraically natural.

For Boolean circuits, Chow [Cho11] circumvents the natural proofs barrier in [RR97] by providing (under standard cryptographic assumptions) an explicit *almost natural proof* that is useful against P/ poly as well as constructive in nearly linear time, but compromises on the largeness condition. Furthermore, Chow [Cho11] shows the unconditional existence of a natural property useful against P/ poly (infinitely often) constructive in linear size that has a weakened largeness condition. In some sense, Theorem 6.7 and Theorem 6.6 are analogous to the work of Chow [Cho11], albeit in the algebraic world.

On the largeness criterion. It has been observed in the definitions of algebraically natural proofs [GKSS17, FSV18] that in the algebraic setting, an analogue of the *largeness* criterion in Definition 6.1 is often available for free. The reason being that a nonzero equation for any class of polynomials vanishes on a very small fraction of all polynomials over any sufficiently large field. However, this trade-off becomes a bit subtle when considering polynomials over finite fields of small size, or polynomials with bounded integer coefficients. In particular, as we observe in the course of the proofs of our results, we still have a large number of polynomials whose coefficients will keep $\{P_{N,c}\}$ (and $\{Q_{N,c}\}$) nonzero, although this set is no longer a significant fraction of the set of all polynomials.

6.1.5 An overview of the proof

At a high level, the idea behind our results is to try and come up with a *non-trivial* property of polynomials which every polynomial with a small circuit satisfies. By a non-trivial property, we mean that there should exist (nonzero) polynomials which do not have this property. The hope is that once we have such a property (which is nice enough), one can try to transform this into an equation via an appropriate *algebraization*. The property that we finally end up using is the existence of (non-explicit) *hitting sets* for polynomials with small circuits.

A hitting set for a class $\mathcal{C}$ of polynomials over a field $\mathbb{F}$ is a set of points $\mathcal{H}$, such that every nonzero polynomial in $\mathcal{C}$ evaluates to a nonzero value on at least one point in $\mathcal{H}$. We then turn this property of *not-vanishing-everywhere on* $\mathcal{H}$ into an equation in some settings to get our main theorems.

To make things a bit more formal, let us consider the map $\Phi_{\mathcal{H}}$, defined by the hitting set $\mathcal{H}$ of $\mathcal{C}$ on the set of all polynomials, that maps any given polynomial f to its evaluations over the points in $\mathcal{H}$. It is clear from the above observation that any nonzero polynomial in the kernel of $\Phi_{\mathcal{H}}$ is guaranteed to be outside $\mathcal{C}$. Thus, if there were a nonzero polynomial that vanishes on all polynomials $f \notin \ker(\Phi_{\mathcal{H}})$, we have an equation for $\mathcal{C}$.

Moreover, if such a polynomial happened to have its degree and circuit complexity polynomially bounded in its number of variables, we would have our main theorems. However, note that *not* being in the kernel of a linear map seems to be a tricky condition to check via a polynomial (as opposed to the complementary property of *being* in the kernel, which can be easily checked via a polynomial). To prove our theorems, we get past this issue in the setting of small finite fields, and for polynomials over $\mathbb{C}$ with bounded integer coefficients.

Over a finite field $\mathbb{F}$, a univariate polynomial that maps every nonzero $x \in \mathbb{F}$ to zero and vice versa, already exists in $q(x) = 1 - x^{|\mathbb{F}|-1}$. Therefore, for a given polynomial f, the equation essentially outputs $\prod_{\mathbf{h}\in\mathcal{H}} q(f(\mathbf{h}))$. Clearly, for a polynomial f, $\prod_{\mathbf{h}\in\mathcal{H}} q(f(\mathbf{h}))$ is zero if and only if f evaluates to a nonzero value on at least one point in $\mathcal{H}$.

To generalize this to other fields, we wish to find a "low-degree" univariate $q(x)$ that maps nonzero values to 0, and zero to a nonzero value. We observe that in the setting when the polynomials in $\mathcal{C}$ have integer coefficients of bounded magnitude , we can still obtain such a univariate polynomial, and in turn a non-trivial defining equation. Indeed, if q were such a univariate, we essentially output $\prod_{\mathbf{h} \in \mathcal{H}} q(f(\mathbf{h}))$, for a given polynomial f. This step relies on a simple application of the Chinese Remainder Theorem.

In order to show that the equations are non-trivial in the sense that there exist polynomials with bounded integer coefficients which do not pass this test, we need to show that there are nonzero polynomials with bounded integer coefficients which vanish everywhere on the hitting set $\mathcal{H}$. We show this via a well known lemma of Siegel[5], which uses a simple pigeon hole argument to show that an under-determined system of homogeneous linear equations where the constraint matrix has small integer entries has a nonzero solution with small integer entries.

As it turns out, our proofs do not use much about the class VP except for the existence of small hitting sets for polynomials in the class. It is not hard to observe that this property is also true for the seemingly larger class VNP and hence the results here also extend to VNP.

We remark that given the hitting set $\mathcal{H}$ explicitly, the construction of the equation is completely explicit. In other words, the non-explicitness in our construction comes only from the fact that we do not have explicit constructions of hitting sets for algebraic circuits. Also, using the techniques in this work, one can prove a general theorem of the form *if a class $\mathcal{C}$ has "small" hitting sets of "low" bit complexity, then there are efficient equations for the polynomials in $\mathcal{C}$ that have coefficients of "moderate" size*. We shall skip the formal statement of such a general theorem to avoid cumbersome notation and only focus on the two interesting consequences of this statement, for the classes VP and VNP.

Organization. We begin with some notations and preliminaries in Section 6.2 before moving on to prove Theorem 6.7 in Section 6.3 and Theorem 6.6 in Section 6.4. In Section 6.5, we observe that these results also generalize to VNP.

6.2 Some Background

We begin by defining "a slice" of VP (and VNP), which is basically the subset of VP (and VNP) of polynomials whose degrees are bounded from above by a specific polynomial in their arity.

Definition 6.10 ($\mathsf{VP}^{[c]}$ and $\mathsf{VNP}^{[c]}$). *For a fixed constant c, we will define $\mathsf{VP}_{\mathbb{F}}^{[c]}$ to denote*

$$\mathsf{VP}_{\mathbb{F}}^{[c]} := \left\{ \{f_n\} \in \mathsf{VP}_{\mathbb{F}} \ : \ f_n \in \mathbb{F}[x_1, \ldots, x_n]^{\leq n^c} \right\},$$

and $\mathsf{VNP}_{\mathbb{F}}^{[c]}$ to denote

$$\mathsf{VNP}_{\mathbb{F}}^{[c]} := \left\{ \{f_n\} \in \mathsf{VNP}_{\mathbb{F}} \ : \ f_n \in \mathbb{F}[x_1, \ldots, x_n]^{\leq n^c} \right\},$$

[5]A statement of the lemma can be found here. Refer to [Sie14] for details.

We also consider polynomials $\{f_n\}$ over integers whose coefficients are in $\{-1,0,1\}$. However, it is important to note that even in this setting the bound is only on the coefficients of f_n; the circuit computing f_n may use arbitrary constants from the underlying field, or an extension. $\Diamond$

Throughout this chapter, we use $\overline{\mathrm{coeff}}(f)$ to denote the vector[6] of coefficients of f.

We will need the following notion of *universal circuits* defined by Raz [Raz10]. A universal circuit is such that any polynomial computed by a small circuit is a simple projection of it. For the sake of completeness, we also include a proof sketch.

Lemma 6.11 (Universal circuit, [Raz10])**.** *Let $\mathbb{F}$ be any field and $n, s \geq 1$ and $d \geq 0$. Then there exists an algebraic circuit $\mathcal{U}$ of size $\mathrm{poly}(n,d,s)$ computing a polynomial in $\mathbb{F}[x_1,\ldots,x_n,y_1,\ldots,y_r]$ with $r \leq \mathrm{poly}(n,d,s)$ such that:*

- $\deg_{\mathbf{x}}(\mathcal{U}(\mathbf{x},\mathbf{y})), \deg_{\mathbf{y}}(\mathcal{U}(\mathbf{x},\mathbf{y})) \leq \mathrm{poly}(d)$*;*
- *for any $f \in \mathbb{F}[x_1,\ldots,x_n]$ with $\deg_{\mathbf{x}}(f) \leq d$ that is computable by an algebraic circuit of size s, there exists an $\mathbf{a} \in \mathbb{F}^r$ such that $f(\mathbf{x}) = \mathcal{U}(\mathbf{x},\mathbf{a})$.*

Sketch. Let f be an n-variate degree d polynomial computable by a circuit C of size s. Using the classical depth reduction result due to Valiant, Skyum, Berkowitz and Rackoff [VSBR83], f has a circuit C' of size $s' = \mathrm{poly}(n,d,s)$ and depth $\ell = O(\log d)$ with the following properties (see e.g. [Sap15] for a complete proof).

- All the product gates have fan-in at most 5.
- C' is *layered*, with alternating layers of sum and product gates.
- The layer above the leaves is of product gates, and the root is an addition gate.

We can therefore construct a *layered* universal circuit $\mathcal{U}$ for the given parameters n, d, s. The circuit will have ℓ layers, with $V_1, V_2, \ldots, V_\ell$ being the layers indexed from leaves to the root. So V_ℓ has a single gate, which is the output gate of the circuit, and V_1 has $n+1$ gates, labelled with the variables $x_1, \ldots, x_n$ and with the constant 1. All the gates in $\mathcal{U}$ are then connected using auxiliary variables $\mathbf{y}$, as follows.

- V_2 has $\leq (n+1)^5$ product gates, with each gate computing a unique monomial of degree at most 5 in the variables $\mathbf{x}$.
- For every odd i with $2 < i < \ell$, the layer V_i has s' addition gates that are all connected to all the gates in the layer V_{i-1}, with each of the wires being labelled by a fresh $\mathbf{y}$-variable.
- For every even i with $2 < i < \ell$, the layer V_i has $\binom{s'}{5}$ product gates, each one multiplying a unique subset of 5 gates from V_{i-1}.

It is now easy to see that $\mathcal{U}$ has at most $\ell(ns')^5$ gates, which is $\mathrm{poly}(n,d,s)$. Also, $\deg(\mathcal{U}) \leq 5^\ell$, which is $\mathrm{poly}(d)$; and $|\mathbf{y}| = r \leq \ell \cdot (ns')^6$, which is $\mathrm{poly}(n,d,s)$. Further, by the depth reduction result [VSBR83], the circuit C' for f can be obtained by setting the auxiliary variables $\mathbf{y}$ appropriately. Since the choice of f was arbitrary, this finishes the proof. $\square$

[6]We do not explicitly mention the monomial ordering used for this vector representation, since all our statements work for any monomial ordering.

6.3 Constructible equations for VP over small finite fields

In this section, we prove our main theorem for finite fields. As mentioned in the introduction, our proof uses the existence of non-explicit hitting sets for small circuits. This fact appears to be folklore but we state below the version that can be found in Forbes' thesis [For14].

Lemma 6.12 (Folklore (cf. Lemma 3.2.14 in [For14])). *Let $\mathbb{F}$ be a finite field with $|\mathbb{F}| \geq d^2$. Let $\mathcal{C}(n, d, s)$ be the class of polynomials in $\mathbb{F}[x_1, \ldots, x_n]$ of degree at most d that are computable by fan-in 2 algebraic circuits of size at most s. Then, there is a non-explicit hitting set for $\mathcal{C}$ of size at most $\lceil 2s \cdot (\log n + 2\log s + 4) \rceil$.*

The above lemma shows that over large enough finite fields, there are non-explicit hitting sets of size $O(s^2)$ (when $n, d \leq s$). We now use this to prove Theorem 6.7 which we first restate below.

Theorem 6.7. *Let $\mathbb{F}$ be any finite field of constant size and $c > 0$ be any constant. There is a polynomial family $\{P_{N,c}\} \in \mathsf{VP}_{\mathbb{F}}$ such that for all large enough n and $N = \binom{n+n^c}{n}$, the following are true.*

- *For every $\{f_n\} \in \mathsf{VP}_{\mathbb{F}}$, where f_n is an n variate polynomial of degree at most n^c, we have*
$$P_{N,c}(\overline{\mathsf{coeff}}(f_n)) = 0\,.$$
- *There exists a family $\{h_n\}$ of n variate polynomials and degree at most n^c with coefficients in $\mathbb{F}$ such that*
$$P_{N,c}(\overline{\mathsf{coeff}}(h_n)) \neq 0\,.$$

Proof. Let $d_n = n^c$ and $s_n = n^{\log n}$ (in fact any $s_n = n^{\omega(1)}$ would work). Since $\mathbb{F}$ has constant size, and we need fields of sufficiently large size for invoking Lemma 6.12, we work over an extension $\mathbb{K}_n$ of $\mathbb{F}$ of size at least n^{2c} and at most $O(n^{2c})$. Let $r_n = [\mathbb{K}_n : \mathbb{F}] = O(\log n)$. Note that the elements of $\mathbb{K}_n$ can also be interpreted as vectors over $\mathbb{F}$ via an $\mathbb{F}$-linear map $\Phi : \mathbb{K}_n \to \mathbb{F}^{r_n}$. We can then define for any $i \in [r_n]$, $\Phi_i : \mathbb{K}_n \to \mathbb{F}$ to be its projection to the i-th co-ordinate. That is, $\Phi_i : \alpha \mapsto (\Phi(\alpha))_i$ for every $i \in [r_n]$.

By Lemma 6.12, there are hitting sets in $\mathbb{K}_n^n$ for $\mathcal{C}(n, d_n, s_n)$ of size at most $O(s_n^2)$; let $\mathcal{H}_n$ be such a hitting set.

For $N = \binom{n+d_n}{n}$, let us index the set $[N]$ by the set $\mathbf{x}^{\leq d_n}$ of n-variate monomials of degree at most d_n. For a point $\mathbf{a} \in \mathcal{H}_n$, we define the vector $\mathsf{eval}(\mathbf{a}) \in \mathbb{K}_n^N$ as $\mathsf{eval}(\mathbf{a})_m = m(\mathbf{a})$ where $m \in \mathbf{x}^{\leq d_n}$ (that is, the m-th coordinate is the evaluation of the monomial m at $\mathbf{a}$). To get vectors over $\mathbb{F}$ instead, for each $i \in [r_n]$, we shall define $\mathsf{eval}(\mathbf{a})^{(i)} \in \mathbb{F}^N$ as $\mathsf{eval}(\mathbf{a})^{(i)}_m = \Phi_i(m(\mathbf{a}))$.

We are now ready to define the polynomial family $\{P_N\}$.

$$P_N(z_m \ : \ m \in \mathbf{x}^{\leq d_n}) := \mathrm{OR}(\mathbf{z}) \cdot \prod_{\mathbf{a} \in \mathcal{H}_n} \left(\prod_{i=1}^{r_n} \left(1 - \left(\sum_m z_m \cdot \mathrm{eval}(\mathbf{a})_m^{(i)} \right)^{|\mathbb{F}|-1} \right) \right),$$
$$\text{where } \mathrm{OR}(\mathbf{z}) = \left(1 - \prod_{m \in \mathbf{x}^{\leq d_n}} \left(1 - z_m^{|\mathbb{F}|-1} \right) \right)$$

Constructivity. Note that $\deg(P_N) \leq |\mathbb{F}| \cdot (N + (|\mathcal{H}_n| \cdot r_n)) = O(N + s_n^2 \cdot \log n) = O(N)$ and the above expression immediately yields an $O(N^2)$-sized circuit for P_N. Therefore, the above family $P_N \in \mathsf{VP}_{\mathbb{F}}$.

Usefulness. Now consider any family $\{f_n\} \in \mathsf{VP}_{\mathbb{F}}^{[c]}$; let k be an integer such that for all large enough n we have that f_n is computable by size n^k circuits. We need to show that $P_N(\overline{\mathrm{coeff}}(f_n)) = 0$ for all large enough n.

For any polynomial $g \in \mathbb{F}[x_1, \ldots, x_n]$ with $\deg g \leq n^c$, we have

$$\begin{aligned} P(\overline{\mathrm{coeff}}(g)) &= \mathrm{OR}(\overline{\mathrm{coeff}}(g)) \cdot \prod_{\mathbf{a} \in \mathcal{H}_n} \left(\prod_{i=1}^{r_n} \left(1 - \left(\sum_m \overline{\mathrm{coeff}}(g)_m \cdot \mathrm{eval}(\mathbf{a})_m^{(i)} \right)^{|\mathbb{F}|-1} \right) \right), \\ &= \mathrm{OR}(\overline{\mathrm{coeff}}(g)) \cdot \prod_{\mathbf{a} \in \mathcal{H}_n} \left(\prod_{i=1}^{r_n} \left(1 - (\Phi_i(g(\mathbf{a})))^{|\mathbb{F}|-1} \right) \right), \\ &= \begin{cases} 1 & \text{if } g \neq 0 \text{ and } g(\mathbf{a}) = 0 \text{ for all } \mathbf{a} \in \mathcal{H}_n, \\ 0 & \text{if } g = 0 \text{ or } g(\mathbf{a}) \neq 0 \text{ for some } \mathbf{a} \in \mathcal{H}_n. \end{cases} \end{aligned}$$

If $f_n = 0$, then $\mathrm{OR}(\overline{\mathrm{coeff}}(f_n)) = 0$. Else, if n is chosen large enough, then f_n is computable by circuits of size at most $s_n = n^{\log n}$ and the set $\mathcal{H}_n$ is a hitting set for f_n. Therefore, there is some point $\mathbf{a} \in \mathcal{H}_n$ such that $f_n(\mathbf{a}) \neq 0$. Hence, $\{P_N\}$ vanishes on the coefficient vector of every polynomial in $\mathsf{VP}_{\mathbb{F}}^{[c]}$.

A remark on the largeness. From the definition of P_N, any nonzero $g \in \mathbb{F}[x_1, \ldots, x_n]^{\leq d_n}$ such that $g(\mathbf{a}) = 0$ for all $\mathbf{a} \in \mathcal{H}_n$ will satisfy $P_N(\overline{\mathrm{coeff}}(g)) \neq 0$. If we interpret the coefficients of g as indeterminates, each equation of the form $g(\mathbf{a}) = 0$ introduces one homogeneous linear constraint in these N indeterminates, over the extension $\mathbb{K}_n$. Each such constraint can be interpreted as $r_n = O(\log n)$ homogeneous linear constraints, over $\mathbb{F}$. Since $|\mathcal{H}_n| \ll N$, the set of g's that are not annihilated by P_N form a subspace of dimension at least $N - O(|\mathcal{H}_n| \log n)$. Thus, there are at least $\left(|\mathbb{F}|^{N - O(|\mathcal{H}_n| \log n)} - 1 \right)$ many g's such that $P_N(\overline{\mathrm{coeff}}(g)) \neq 0$. In fact, the coefficient vector of such a polynomial g can easily be found in time $\mathrm{poly}(N)$ using *Gaussian elimination.* □

6.4 Constructible equations for VP with coefficients in $\{-1, 0, 1\}$

In this section, we prove Theorem 6.6. As before, our proof uses the existence of non-explicit hitting sets for circuits of small size. When the underlying field is $\mathbb{C}$, their existence is known due to the results of Heintz and Schnorr [HS80] as stated below.

Lemma 6.13 (Hitting sets for efficiently computable polynomials [HS80]). *For all $n, d, s \in \mathbb{N}$ large enough, there exist (non-explicit) hitting sets $\mathcal{H}$ for $\mathcal{C}(n, d, s)$ (the set of all n-variate polynomials with degree at most d that are computable by algebraic circuits of size at most s), such that $\mathcal{H} \subset [(sd)^2]^n$ and $|\mathcal{H}| = \mathrm{poly}(s)$.*

We now prove our main theorem which we restate below.

Theorem 6.6. *Let $c > 0$ be any constant. There is a polynomial family $\{P_{N,c}\} \in \mathsf{VP}_{\mathbb{Q}}$ such that for all large n and $N = \binom{n+n^c}{n}$, the following are true.*

- *For every family $\{f_n\} \in \mathsf{VP}_{\mathbb{C}}$, where f_n is an n variate polynomial of degree at most n^c and coefficients in $\{-1, 0, 1\}$, we have*

  $$P_{N,c}(\overline{\mathrm{coeff}}(f_n)) = 0\,,$$

  where $\overline{\mathrm{coeff}}(f_n)$ is the coefficient vector of f_n.

- *There exists a family $\{h_n\}$ of n variate polynomials and degree at most n^c with coefficients in $\{-1, 0, 1\}$ such that*

  $$P_{N,c}(\overline{\mathrm{coeff}}(h_n)) \neq 0\,.$$

As mentioned earlier, the proof would also generalise in a straightforward manner for polynomial families $\{f_n\} \in \mathsf{VP}_{\mathbb{Z}}^{[c]}$ whose coefficients are bounded by N. We state this for polynomials whose coefficients are in $\{-1, 0, 1\}$ just to avoid cumbersome notation.

Proof. The proof will proceed similar to the proof of Theorem 6.7, with a careful use of the Chinese Remainder Theorem.

Let $d_n = n^c$ and $s_n = n^{\log n}$ (again, s_n needs to be barely super-polynomial in n). For $N = \binom{n+d_n}{n}$, let us index the set $[N]$ by the set $\mathbf{x}^{\leq d_n}$ of n-variate monomials of degree at most d_n. For a point $\mathbf{a} \in \mathbb{Z}^n$, we define the vector $\mathrm{eval}(\mathbf{a}) \in \mathbb{Q}^N$ as $\mathrm{eval}(\mathbf{a})_m = m(\mathbf{a})$ where $m \in \mathbf{x}^{\leq d_n}$ (that is, the m-th coordinate is the evaluation of the monomial m at $\mathbf{a}$). Therefore, for any n-variate polynomial f of degree at most d_n, we have $f(\mathbf{a}) = \left\langle \overline{\mathrm{coeff}}(f), \mathrm{eval}(\mathbf{a}) \right\rangle$.

Let $B_n = (s_n \cdot d_n)^2$. By Lemma 6.13, there are hitting sets in $[B]^n$ of size $\mathrm{poly}(s_n)$ for the class $\mathcal{C}(n, d_n, s_n)$ (of n-variate polynomials, of degree at most d_n that are computable by circuits of size s_n) with coefficients in $\Delta = \{-1, 0, 1\}$. Let $\mathcal{H}_n$ be one such set. Note that for any n-variate polynomial f of degree at most d_n and coefficients in Δ, and any $\mathbf{a} \in \mathcal{H}_n$, we have $|f(\mathbf{a})| \leq N \cdot B^{d_n}$, which unfortunately is not $\mathrm{poly}(N)$. However, we can work with some "proxy evaluations" by simulating Chinese Remaindering.

For any $\mathbf{a} \in \mathcal{H}_n$ and a positive integer r, define the vector $\widetilde{\mathrm{eval}}_r(\mathbf{a})$ as follows:

$$\widetilde{\mathrm{eval}}_r(\mathbf{a})_m := (m(\mathbf{a}) \bmod r) \quad \text{for all } m \in \mathbf{x}^{\leq d_n}.$$

It is to be stressed that $\widetilde{\mathrm{eval}}_r(\mathbf{a})$ is a vector over $\mathbb{Q}$, whose coordinates are integers from the set $\{0, \ldots, r-1\}$.

Claim 6.14. *Suppose f is a polynomial with integer coefficients, and $\mathbf{a} \in \mathbb{Z}^n$. If $f(\mathbf{a}) \neq 0$ and $|f(\mathbf{a})| \leq M$, then there is some $r \leq O((\log M)^2)$ such that*

$$\left\langle \overline{\mathrm{coeff}}(f), \widetilde{\mathrm{eval}}_r(\mathbf{a}) \right\rangle \neq 0 \bmod r.$$

Proof of claim. Let $\ell = \log(M+1)$, note that the LCM of the set $[\ell^2]$ is at least $2^\ell > M$. Since $f(\mathbf{a})$ is a nonzero integer with $|f(\mathbf{a})| \leq M$, by the Chinese Remainder Theorem there is some prime $r \leq \ell^2$ such that $f(\mathbf{a}) \neq 0 \bmod r$.

$$\begin{aligned}\left\langle \overline{\mathrm{coeff}}(f), \widetilde{\mathrm{eval}}_r(\mathbf{a}) \right\rangle &= \left\langle \overline{\mathrm{coeff}}(f), \mathrm{eval}_r(\mathbf{a}) \right\rangle \bmod r \\ &= f(\mathbf{a}) \bmod r \\ &\neq 0 \bmod r\end{aligned}$$

□

Let $M = N \cdot B^{d_n}$ and $\ell = \log(M+1)$. For any $r \in [\ell^2]$, any $\mathbf{a} \in \mathcal{H}_n$ and n-variate polynomial f of degree at most d_n and coefficients from Δ, we have

$$\left| \left\langle \overline{\mathrm{coeff}}(f), \widetilde{\mathrm{eval}}_r(\mathbf{a}) \right\rangle \right| \leq N \cdot \ell^2 =: R.$$

We are now ready to define the polynomial family $\{P_N\}$.

$$\begin{aligned} P_N(z_m \;:\; m \in \mathbf{x}^{\leq n}) &= \mathrm{OR}(\mathbf{z}) \cdot \prod_{\mathbf{a} \in \mathcal{H}_n} \prod_{r=2}^{\ell^2} Q_r\left(\left\langle \mathbf{z}, \widetilde{\mathrm{eval}}_r(\mathbf{a}) \right\rangle\right), \\ \text{where } Q_r(x) &= \prod_{\substack{i \in [-R, \ldots, R] \\ i \bmod r \neq 0}} (x - i), \\ \mathrm{OR}(\mathbf{z}) &= 1 - \prod_{m \in \mathbf{x}^{\leq d_n}} (1 - z_m)\end{aligned}$$

Constructivity. For our setting of the underlying parameters, $|\mathcal{H}_n| \leq n^{O(\log n)}, B_n \leq n^{O(\log n)}$, $M \leq N \cdot n^{O(d_n \log n)}$ and $\ell = \mathrm{poly}(n)$; and $R \leq O(N \mathrm{poly}(n)) = \tilde{O}(N)$. Therefore, P_N is a polynomial of degree at most $\tilde{O}(N^2)$. Moreover, the above expression also shows that P_N is computable by a circuit of size $\tilde{O}(N^3)$ and hence $\{P_N\} \in \mathsf{VP}$.

Usefulness. Fix a polynomial family $\{f_n\} \in \mathsf{VP}^{[c]}$ such that the coefficients of f_n are in $\{-1,0,1\}$ for all n. Let k be an integer such that for all large enough n we have that f_n is computable by size n^k circuits. We need to show that $P_N(\overline{\mathrm{coeff}}(f_n)) = 0$ for all large enough n. Note that we have $\mathrm{OR}(\overline{\mathrm{coeff}}(f_n)) \neq 0$ if f_n is nonzero, and 0 if $f_n = 0$. Hence, it suffices to show that $P_N(\overline{\mathrm{coeff}}(f_n)) = 0$ for nonzero f_n.

For any large enough n so that $0 \neq f_n$ is computable by circuits of size at most $s_n = n^{\log n}$ and the set $\mathcal{H}_n$ is a hitting set for f_n, we know that $f_n(\mathbf{a}) \neq 0$ for some $\mathbf{a} \in \mathcal{H}_n$. Therefore, for some $r \in [\ell^2]$, we have that $\left\langle \overline{\mathrm{coeff}}(f), \widetilde{\mathrm{eval}}_r(\mathbf{a}) \right\rangle$ is a nonzero integer in $\{-R,\ldots,R\}$ that is not divisible by r. Hence, we have

$$\begin{aligned} Q_r\left(\left\langle \overline{\mathrm{coeff}}(f), \widetilde{\mathrm{eval}}_r(\mathbf{a}) \right\rangle\right) &= 0, \\ \implies P(\overline{\mathrm{coeff}}(f)) &= 0. \end{aligned}$$

A remark on the largeness. From the definition of P_N, any nonzero $g \in \mathbb{F}[x_1,\ldots,x_n]^{\leq d_n}$ such that $g(\mathbf{a}) = \left\langle \overline{\mathrm{coeff}}(g), \mathrm{eval}(\mathbf{a}) \right\rangle = 0$ for all $\mathbf{a} \in \mathcal{H}_n$ will satisfy $P_N(\overline{\mathrm{coeff}}(g)) \neq 0$. In order to show that there are many such g's with coefficients in $\{-1,0,1\}$, we use a pigeon-hole argument, which is essentially an instance of a well known lemma of Siegel [Sie14]. For completeness, we include a sketch of the argument here.

Consider the map $\Gamma : \mathbb{Z}^N \to \mathbb{Z}^{|\mathcal{H}_n|}$ defined as

$$\Gamma(z_m \ : \ m \in \mathbf{x}^{\leq d_n}) := (\langle \mathbf{z}, \mathrm{eval}(\mathbf{a}) \rangle \ : \ \mathbf{a} \in \mathcal{H}_n)$$

The map Γ is linear in the sense that $\Gamma(\mathbf{z} + \mathbf{z}') = \Gamma(\mathbf{z}) + \Gamma(\mathbf{z}')$. Consider the restriction of Γ on just $\{0,1\}^N$; the range of Γ under this restriction is $\{-M,\ldots,M\}^{|\mathcal{H}_n|}$. Hence, by the pigeon-hole-principle there must be some $\mathbf{b} \in \{-M,\ldots,M\}^{|\mathcal{H}_n|}$ with at least $2^N/(2M+1)^{|\mathcal{H}_n|}$ pre-images inside $\{0,1\}^N$. If $\mathbf{h}_0$ is any fixed preimage, then

$$\left\{\mathbf{h} - \mathbf{h}_0 \in \{-1,0,1\}^N \ : \ \mathbf{h} \in \Gamma^{-1}(\mathbf{b}) \cap \{0,1\}^N\right\}$$

are all coefficient vectors of polynomials $g \in \mathbb{Z}[x_1,\ldots,x_n]^{\leq d_n}$ with coefficients in $\{-1,0,1\}$ whose coefficient vectors are not zeroes of P_N. $\square$

It is worth mentioning that there are 3^N possible polynomials in $\mathbb{Z}[x_1,\ldots,x_n]^{\leq d_n}$ with coefficients in $\{-1,0,1\}$. The above remark on the largeness shows that there are $2^{N-q(n)}$ many polynomials g such that $P_N(\overline{\mathrm{coeff}}(g)) \neq 0$; for some $q(n) = n^{O(\log n)}$. Also note that unlike Theorem 6.7, finding such a polynomial g would probably require time $\exp(N)$, since it encodes the *minimum subset sum problem*.

6.5 Equations for VNP

We shall now state and prove the VNP analogues of Theorem 6.6 and Theorem 6.7. First, we have the following definition.

Definition 6.15 (Definability of Polynomials). *For $s \geq 1$, a polynomial f_n is said to be* s-definable *if there exists a polynomial $g_s \in \mathcal{C}(s,s,s)$ such that for $m = s - n$,*

$$f_n(\mathbf{x}) = \sum_{\alpha \in \{0,1\}^m} g_s(\mathbf{x}, \alpha)\,.$$

Further, let us denote by $\mathcal{D}(n,d,s)$ the class of all n-variate polynomials of degree d that are s-definable. $\lozenge$

Remark 6.16. *Note that for every family $\{f_n\} \in$ VNP, there is a polynomially bounded function $s(n) > n, d(n)$ such that f_n is $s(n)$-definable, for all large n.* $\lozenge$

6.5.1 VNP over Small Finite Fields

As in the VP case, we will need the existence of non-explicit hitting sets. A slight modification to the proof of Lemma 3.2.14 in [For14]) gives us the following statement.

Lemma 6.17. *Let $\mathbb{F}$ be a finite field with $|\mathbb{F}| \geq d^2$. Let $\mathcal{D}(n,d,s)$ be the class of polynomials in $\mathbb{F}[x_1, \ldots, x_n]$ of degree at most d that are s-definable. Then, there is a non-explicit hitting set $\mathcal{H}$ for $\mathcal{D}(n,d,s)$ of size at most $\lceil 2s \cdot (3\log s + 4) \rceil$.*

Proof. In order to prove the existence of a hitting set for the class $\mathcal{D}(n,d,s)$, we will need a bound on the number of polynomials in the class $\mathcal{D}(n,d,s)$ as well as a bound on the size of an explicit hitting set for the class of n-variate degree at most d polynomials. These two bounds are summarized in the following claims, proofs of which can be found in [For14].

Claim 6.18 (Lemma 3.1.6 in [For14]). *Let $\mathbb{F}$ be a finite field and $n, s \geq 1$. There are at most $(8n\,|\mathbb{F}|\,s^2)^s$ n-variate polynomials in $\mathbb{F}[\mathbf{x}]$ computable by (single-output) algebraic circuits of size $\leq s$ and fan-in ≤ 2.*

Claim 6.19 (Lemma 3.2.13 in [For14]). *Let $\mathbb{F}$ be a finite field with $|\mathbb{F}| \geq (1+\varepsilon)d$. Let $\mathcal{C} \subseteq \mathbb{F}[\mathbf{x}]$ be a finite set of n-variate polynomials of degree $< d$. Then there is a non-explicit hitting set for $\mathcal{C}$ of size $\leq \lceil \log_{1+\varepsilon} |\mathcal{C}| \rceil$.*

Note that by definition, the number of n-variate polynomials that are s-definable is at most the number of polynomials in $\mathcal{C}(s,s,s)$; the class of s-variate polynomials of degree $\leq s$ computable by size s algebraic circuits of fan-in ≤ 2. Thus, by Claim 6.18, $|\mathcal{D}(n,d,s)| \leq (8\,|\mathbb{F}|\,s^3)^s$.

The rest of the proof follows exactly along the lines of the proof of Lemma 3.2.14 in [For14].

As $|\mathbb{F}| \geq d^2$, we have $d \leq |\mathbb{F}|$, and so $|\mathbb{F}| \geq (1+\varepsilon)d$ for $(1+\varepsilon) = \sqrt{|\mathbb{F}|}$. Thus, using $\varepsilon = \sqrt{|\mathbb{F}|} - 1$ in Claim 6.19, we get that there is a non-explicit hitting set $\mathcal{H}$ for $\mathcal{D}(n,d,s)$ of size at most

$$\left\lceil \log_{\sqrt{|\mathbb{F}|}} |\mathcal{D}(n,d,s)| \right\rceil \leq \left\lceil \log_{\sqrt{|\mathbb{F}|}} (8\,|\mathbb{F}|\,s^3)^s \right\rceil = \left\lceil s(2+2\log_{|\mathbb{F}|}(8s^3)) \right\rceil = \left\lceil s(2+6\log_{|\mathbb{F}|}(2s)) \right\rceil$$

Finally, as $|\mathbb{F}| \geq 2$, we have

$$|\mathcal{H}| \leq \lceil s \cdot (2+6\log(2s)) \rceil = \lceil 2s \cdot (1+3\log(2s)) \rceil = \lceil 2s \cdot (3\log s + 4) \rceil .$$

This completes the proof. □

By Remark 6.16, we have that over large enough finite fields, there are non-explicit hitting sets of size $O(s^2)$ for the polynomials in VNP that are s-definable. Since the proof of Theorem 6.7 does not use any property of VP except for the existence of non-trivial hitting sets, we get the following theorem. The proof is omitted, since it is exactly the same except we use Lemma 6.17 instead of Lemma 6.12.

Theorem 6.9. *Let $\mathbb{F}$ be any finite field of constant size and $c > 0$ be any constant. There is a polynomial family $\{Q_{N,c}\} \in \mathsf{VP}_{\mathbb{F}}$ such that for all large enough n and $N = \binom{n+n^c}{n}$, the following are true.*

- *For every family $\{f_n\} \in \mathsf{VNP}_{\mathbb{F}}$, where f_n is an n variate polynomial of degree at most n^c, we have*

$$Q_{N,c}(\overline{\mathrm{coeff}}(f_n)) = 0 .$$

- *There exists a family $\{h_n\}$ of n variate polynomials and degree at most n^c with coefficients in $\mathbb{F}$ such that*

$$Q_{N,c}(\overline{\mathrm{coeff}}(h_n)) \neq 0 .$$

6.5.2 Polynomials in VNP with Small Integer Coefficients

Our argument will be identical to that in Section 6.4, for which we will need a statement analogous to Lemma 6.13 showing the existence of non-explicit hitting sets for VNP with small bit-complexity. We will first give a universal map for the polynomials in VNP, analogous to Lemma 6.11; for which we need the following lemma.

Lemma 6.20 (Coefficient Vectors of Definable Polynomials)**.** *Let $f \in \mathbb{C}[\mathbf{x}]$ be an n-variate polynomial of degree d that is s-definable. Then there exists an s-variate polynomial g and a* linear *map $L_{n,d,s}$, such that $\overline{\mathrm{coeff}}(f) = L(\overline{\mathrm{coeff}}(g))$. Furthermore, the map L depends* solely *on n, d and s.*

Proof. Let $m = s - n$. Since f is s-definable, there is an s-variate polynomial $g(\mathbf{x}, \mathbf{w})$ of degree at most s as follows.

$$f(\mathbf{x}) = \sum_{\alpha \in \{0,1\}^m} g(\mathbf{x}, \mathbf{w} = \alpha)$$

Now observe that for any monomial $\mathbf{x}^{\mathbf{e}} \in \mathbf{x}^{\leq d}$,

$$\begin{aligned}
\mathrm{coeff}_{\mathbf{x}^{\mathbf{e}}}(f) = \mathrm{coeff}_{\mathbf{x}^{\mathbf{e}}}\left(\sum_{\alpha \in \{0,1\}^m} g(\mathbf{x}, \alpha)\right) \\
&= \sum_{\alpha \in \{0,1\}^m} \mathrm{coeff}_{\mathbf{x}^{\mathbf{e}}}\left(g(\mathbf{x}, \alpha)\right) \\
&= \sum_{\alpha \in \{0,1\}^m} \mathrm{coeff}_{\mathbf{x}^{\mathbf{e}}}\left(\sum_{\mathbf{w}^{\mathbf{a}} \in \mathbf{w}^{\leq s}} \alpha^{\mathbf{a}}\, \mathrm{coeff}_{\mathbf{w}^{\mathbf{a}}}(g(\mathbf{x}, \mathbf{w}))\right) \\
&= \sum_{\mathbf{w}^{\mathbf{a}} \in \mathbf{w}^{\leq s}} \left(\sum_{\alpha \in \{0,1\}^m} \alpha^{\mathbf{a}}\right) \mathrm{coeff}_{\mathbf{x}^{\mathbf{e}}\mathbf{w}^{\mathbf{a}}}(g) \\
&= \sum_{\mathbf{w}^{\mathbf{a}} \in \mathbf{w}^{\leq s}} 2^{(m - |\mathrm{supp}(\mathbf{a})|)}\, \mathrm{coeff}_{\mathbf{x}^{\mathbf{e}}\mathbf{w}^{\mathbf{a}}}(g)
\end{aligned}$$

Now we can define the desired map $L : \mathbb{C}^M \to \mathbb{C}^N$ for $M = \binom{s+s}{s}$ and $N = \binom{n+d}{n}$, as follows.

$$L_{\mathbf{e}}(\overline{\mathrm{coeff}}(g)) = \sum_{\mathbf{w}^{\mathbf{a}} \in \mathbf{w}^{\leq s}} 2^{(m - |\mathrm{supp}(\mathbf{a})|)}\, \mathrm{coeff}_{\mathbf{x}^{\mathbf{e}}\mathbf{w}^{\mathbf{a}}}(g) \qquad \forall \mathbf{e} \in [N]$$

□

Lemma 6.21 (Universal Map for Definable Polynomials)**.** *Let $s \geq n \geq 1$ and $d \geq 0$. Then for $N = \binom{n+d}{n}$ there exists a polynomial map $\mathcal{U}(\mathbf{y}) : \mathbb{C}^r \to \mathbb{C}^N$ with $r \leq \mathrm{poly}(n, d, s)$ such that:*

- $\deg(\mathcal{U}(\mathbf{y})) \leq \mathrm{poly}(s)$*;*
- *for any $f \in \mathbb{C}[x_1, \ldots, x_n]$ with $\deg_{\mathbf{x}}(f) \leq d$ that is s-definable, there exists an $\mathbf{a} \in \mathbb{C}^r$ such that $\overline{\mathrm{coeff}}(f) = \mathcal{U}(\mathbf{a})$.*

Proof. Let $\mathcal{D}(n, d, s)$ be the class of all n-variate, degree d polynomials that are s-definable and suppose $f_n(\mathbf{v}) \in \mathcal{D}(n, d, s)$. Then by Lemma 6.20 there exists an s-variate, degree s polynomial $g_s \in \mathcal{C}(s, s, s)$ such that the coefficients of f_n are obtained by taking suitable *linear* combinations of the coefficients of g_s. Therefore we will now shift our focus to the coefficient vectors of polynomials from $\mathcal{C}(s, s, s)$.

Using Lemma 6.21 for number of variables, degree and size, all bounded by s, we get a universal circuit $\mathcal{U}(\mathbf{x}, \mathbf{y})$ for $\mathcal{C}(s, s, s)$ with $|\mathbf{y}| = r \leq s^k$ for some constant k. We will assume without loss of generality that $\deg_{\mathbf{y}}(\mathcal{U}) \leq s^k$. Now for $M = \binom{s+s}{s}$, we can view $\mathcal{U}(\mathbf{x}, \mathbf{y})$ as a polynomial map $\mathcal{U} : \mathbb{C}^r \to \mathbb{C}^M$ given by $\mathcal{U}(\mathbf{y}) = (\mathcal{U}(\mathbf{y}_1), \ldots, \mathcal{U}(\mathbf{y}_M))$, where $\mathcal{U}_m(\mathbf{y})$ is the coefficient of the monomial $m \in \mathbf{x}^{\leq s}$ in the polynomial $\mathcal{U}(\mathbf{x}, \mathbf{y})$. Note that the degree of every such $\mathcal{U}_m(\mathbf{y})$ is at most s^k.

Now by Lemma 6.21, for every $g_s \in \mathcal{C}(s, s, s)$ the coefficient vector of g_s is in the image of $\mathcal{U}$. Therefore, for $N = \binom{n+d}{n}$, let $L : \mathbb{C}^M \to \mathbb{C}^N$ be the linear map given by Lemma 6.20. Then for every $f_n \in \mathcal{D}(n, d, s)$ the coefficient vector of f_n is in the image of $(L \circ \mathcal{U}) : \mathbb{C}^r \to \mathbb{C}^N$. Further, since L is a linear map, the degree of $(L \circ \mathcal{U})$ is also bounded by $s^k = \mathrm{poly}(s)$. □

We now prove the existence of non-explicit hitting sets of small bit-complexity even for the class of efficiently definable polynomials with small integer coefficients.

Number of efficiently definable polynomials with small coefficients

We first need to bound the number of definable polynomials with small coefficients. The lemma below is a slight modification of a result of Hrubeš and Yehudayoff [HY11a, Claim 3.6]. The proof uses some basic algebraic geometry notions such as *dimension and degree of varieties* and also employs Bézout's theorem, which may be found in most algebraic geometry texts (e.g. [DS13]).

Lemma 6.22 ([HY11a]). *Let $V \in \mathbb{C}^n$ be an irreducible algebraic variety of dimension k and degree r. Suppose $F = (F_1, \ldots, F_m)$ with $F_i \in \mathbb{F}[x_1, \ldots, x_n]^{\leq d}$ is a polynomial map. Then, for $\Delta \subset \mathbb{Z}$,*

$$|F(V) \cap \Delta^m| \leq r \cdot (|\Delta| \cdot d)^k.$$

Proof. The proof is by induction on the dimension k. For the base case of $k = 0$, we would have $|V| = 1$ as V is irreducible and hence $|F(V)| = 1$.

For each $i \in [m]$ and $b \in \Delta$, define $V_{i,b} = V \cap F_i^{-1}(b)$. Suppose for every $i \in [m]$ there is just a single $b \in \Delta$ such that $V_{i,b} \neq \emptyset$, then clearly $|F(V) \cap \Delta^m| \leq 1$. Otherwise, let i be such that at least two of $\{V_{i,b} \ : \ b \in \Delta\}$ are non-empty. Since at least two of them are non-empty, each $V_{i,b}$ is a proper sub-variety of V and hence $\dim(V_{i,b}) < \dim(V)$. Let the non-empty varieties be decomposed into irreducible varieties as

$$V_{i,b} = V_{i,b}^{(1)} \cup \cdots \cup V_{i,b}^{(t_b)}.$$

By Bézout's theorem (see e.g., [DS13]), we also have $\sum_j \deg(V_{i,b}^{(j)}) \leq d \cdot \deg(V_{i,b})$. Then,

$$\begin{aligned} F(V) \cap \Delta^m &\subseteq \bigcup_{b \in \Delta} F(V_{i,b}) \cap \Delta^m \\ &= \bigcup_{b \in \Delta} \bigcup_{j \in [t_b]} F(V_{i,b}^{(t)}) \cap \Delta^m \\ \implies |F(V) \cap \Delta^m| &\leq \sum_{b \in \Delta} \sum_{j \in [t_b]} \deg(V_{i,b}^{(j)}) (|\Delta| \cdot d)^{k-1} \\ &\leq \deg(V) \cdot (|\Delta| \cdot d)^k. \end{aligned}$$

□

Corollary 6.23. *The number of polynomials with coefficients in $\Delta \subset \mathbb{Z}$ that are s-definable is at most $(|\Delta| \cdot s)^{\mathrm{poly}(s)}$.*

Proof. By Lemma 6.21, we know that any s-definable polynomial can be seen as an image of a universal map $\mathcal{U} : \mathbb{F}[\mathbf{x}, \mathbf{y}] \to \mathbb{F}[\mathbf{x}]$. Thus, if we view $\mathcal{U}$ as a polynomial map of the form $\mathcal{U} = (\mathcal{U}_1(\mathbf{y}), \ldots, \mathcal{U}_N(\mathbf{y}))$, then any polynomial of the type we wish to count is contained in the set $(\mathcal{U}(\mathbb{C}^{|\mathbf{y}|}) \cap \Delta^N)$. Here $\mathcal{U}_m$ computes the coefficient of the m-th monomial in $\mathbf{x}$, and by Lemma 6.21, $|\mathbf{y}| = \mathrm{poly}(s)$ and $\deg(\mathcal{U}_m) = \mathrm{poly}(s)$ for every $m \in [N]$.

Finally, note that $\mathbb{C}^{|\mathbf{y}|}$ is an irreducible variety which has degree 1 and dimension $|\mathbf{y}|$. Thus using Lemma 6.22, we have that the number of polynomials with coefficients in Δ that are s-definable is at most $(|\Delta| \cdot \mathrm{poly}(s))^{\mathrm{poly}(s)} \leq (|\Delta| \cdot s)^{\mathrm{poly}(s)}$. □

Existence of hitting sets with low bit-complexity

Lemma 6.24 (Hitting sets for efficiently definable polynomials)**.** *Let $\Delta \subset \mathbb{Z}$. There are (non-explicit) hitting sets $\mathcal{H}$ for $\mathcal{D}(n,d,s)$ (the set of all n-variate polynomials with degree at most d that are s-definable) with coefficients in Δ, such that $\mathcal{H} \subset [ds\,|\Delta|]^n$ and $|\mathcal{H}| = \mathrm{poly}(s)$.*

Proof. Let $\mathcal{H}$ be a uniformly random subset of size $t = \mathrm{poly}(s)$ of the grid $[ds\,|\Delta|]^n$. For any nonzero polynomial $f(\mathbf{x}) \in \mathcal{D}(n,d,s)$, by the Polynomial Identity Lemma (Lemma 1.7) we know that the number of zeroes of any n-variate degree d polynomial f on the grid $[ds\,|\Delta|]^n$ is upper bounded by $d(ds\,|\Delta|)^{n-1} = \frac{1}{s|\Delta|}(ds\,|\Delta|)^n$. Thus, the probability that $\mathcal{H}$ is not a hitting set for a fixed $f \in \mathcal{D}(n,d,s)$ is equal to the ratio $\left(\binom{(ds|\Delta|)^n/s|\Delta|}{t} / \binom{(ds|\Delta|)^n}{t}\right)$, which can be upper bounded by $(1/s\,|\Delta|)^{\Omega(t)}$.

Let $\mathcal{D}'$ be the set of all polynomials in $\mathcal{D}(n,d,s)$ whose coefficients are from Δ. Therefore, the probability that $\mathcal{H}$ is *not* a hitting set for $\mathcal{D}'$ is upper bounded by:

$$\begin{aligned}\Pr_{\mathbf{a}_1,\ldots,\mathbf{a}_t \in [ds|\Delta|]^n}\left[\{\mathbf{a}_1,\ldots,\mathbf{a}_t\} \text{ not a hitting set for } \mathcal{C}'\right] &\leq \left|\mathcal{C}'\right| \cdot \left(\frac{1}{s\,|\Delta|}\right)^{\Omega(t)} \\ &\leq (s\,|\Delta|)^{\mathrm{poly}(s)-\Omega(t)} \quad \text{(Corollary 6.23)} \\ &\ll 1. \quad \text{(if } t = \mathrm{poly}(s) \text{ large enough)}\end{aligned}$$

Hence, there exist $\mathrm{poly}(s)$-sized hitting sets $\mathcal{H} \subset [ds\,|\Delta|]^n$ for $\mathcal{D}'$. □

We can now prove the following theorem along the lines of Theorem 6.6. The proof of Theorem 6.6 almost directly extends here, as the proof does not assume anything about VP except for the existence of non-explicit hitting sets of small bit-complexity, which here is given by Lemma 6.24. We omit the proof to avoid repetition.

Theorem 6.8. *Let $c > 0$ be any constant. There is a polynomial family $\{Q_{N,c}\} \in \mathsf{VP}_{\mathbb{Q}}$ such that for all large n and $N = \binom{n+n^c}{n}$, the following are true.*

- *For every family $\{f_n\} \in \mathsf{VNP}_{\mathbb{C}}$, where f_n is an n variate polynomial of degree at most n^c and coefficients in $\{-1,0,1\}$, we have*

$$Q_{N,c}(\overline{\mathrm{coeff}}(f_n)) = 0\,.$$

- *There exists a family $\{h_n\}$ of n variate polynomials and degree at most n^c with coefficients in $\{-1,0,1\}$ such that*

$$Q_{N,c}(\overline{\mathrm{coeff}}(h_n)) \neq 0\,.$$

Remark 6.25. *In the setting of small integer coefficients (or over small finite fields), there exist constructible low degree equations for both* VP *and* VNP*. However, this does not mean that the framework of algebraically natural proofs cannot be used for separating* VP *and* VNP*. It is worth noting that the equations for* VP *and* VNP *that are constructible in* VP *seem to be different from each other as they use different universal map constructions. This also highlights the fact that any separation of* VP *and* VNP *(in the bounded coefficient setting) cannot rely solely on the degree and circuit size of their equations, but might need to look more carefully at the structure and properties of these equations.* $\Diamond$

6.6 Discussion

In the context of proving circuit lower bounds, and in relation to the notion of *algebraically natural proofs*, an interesting question that emerges from the results in this chapter is whether the condition of "small coefficients" is necessary for efficiently constructible equations to exist, especially for the class VP. While this question remains open for VP, a recent joint work with Kumar, Ramya and Saptharishi [KRST20] shows that if the family $\{\mathsf{Perm}_n\}$ is 2^{n^ε}-hard, then VNP does not have efficiently computable equations. This means that the additional restriction on the coefficients is essentially vital for the existence of efficiently constructible equations for the class VNP, and therefore provides strong evidence *against* the existence of efficient equations for VNP.

In light of Theorem 6.8 and the result of [KRST20] for VNP, one could make a case that equations for VP might also incur a super-polynomial blow up, without the restriction on coefficients. On the other hand, it could also be argued that an analogue of [KRST20] may not be true for VP, since their proof crucially uses the fact that VNP is "closed under exponential sums". In fact, the proofs in [KRST20] essentially algebraises the intuition that coefficient vectors of polynomials in VNP "look random" to a polynomial in VP, *provided that* VNP was exponentially more powerful than VP, using a result of Kabanets and Impagliazzo [KI04] (see Theorem 1.23).

A crucial point to note however, is that the result in [KRST20] also shows that if VNP was indeed exponentially more powerful than VP, then any algebraically natural proof for VP is an algebraically natural proof that separates VP and VNP!

These varied perspectives on the recent results on efficient equations for polynomials in VP with bounded coefficients highlight that the existence of such equations for VP in general continues to remain an intriguing mystery.

7 | Future Directions

This thesis attempts to gain a better understanding of hitting sets for algebraic models. On the one hand, we provide a new technique for constructing hitting sets and extend a previously known construction to a strictly more powerful model, thus contributing to the set of tools for PIT. On the other hand, our study about the consequences of non-trivial explicit hitting sets provides some important insights into blackbox PIT, and also possibly into the question of proving strong lower bounds.

Naturally, each of the works that we have seen lead to interesting open questions. We now list some immediate directions for improving the works that have been discussed.

7.1 Questions about Structured Models

Extending the results in Chapter 3. An interesting open problem is whether we can give non-trivial hitting sets for the class of *non-commutative skew circuits*. Lagarde, Limaye and Srinivasan [LLS19] provide a white-box PIT in some restricted settings when the skew circuits are somewhat closer to UPT (with some restriction on what sort of parse trees they can have) but removing this restriction would be a great step forward.

Another issue is that the current construction of hitting sets for FewPT circuits (which build on the work of Gurjar, Korwar, Saxena and Thierauf [GKST17]) incurs quasipolynomial losses at two different places. The first is in the construction of the *basis isolating weight assignment (BIWA)*, and we only know to construct that using quasipolynomially large weights. The other is in a brute-force enumeration of all monomials of support $O(\log s)$. As a result, even if at a later day we have a construction of a BIWA with polynomially large weights, this proof would still only yield a quasipolynomially large hitting set for FewPT circuits. It would be interesting to see if this brute-force enumeration could be circumvented.

Extending the results in Chapter 4. A natural question is whether we can exploit the structure of Newton polytopes of polynomials computed by other structured models, like ROABPs or other well-understood models. For example, one can already see that Lemma 4.10 applies to any measure that uses an operator that "acts linearly on the Newton polytope", e.g. shifted partials. It would therefore be interesting to see if any lower bounds (and hitting sets) based on such measures can be extended or even reproved, using the method of Newton polytopes.

Another possible direction is to extend the results of Forbes, Ghosh and Saxena [FGS18] to log-variate analogues of slightly more powerful models. Obtaining efficient hitting sets for log-variate ROABPs is already known to imply efficient hitting sets for depth-3-powering circuits in the general setting. A relatively easier, and famously interesting question is that of constructing efficient hitting sets for depth-3-powering circuits in the general case. The state-of-the-art for hitting sets of depth-3-powering circuits is $n^{O(\log\log n)}$, and this model is arguably the best understood model for which we do not have explicit efficient hitting sets.

7.2 Questions about the General Models

Based on Chapter 5. A natural question in the spirit of the results from Chapter 5, and those in the work of Agrawal, Ghosh and Saxena [AGS19] is *"Can we hope to bootstrap lower bounds"*? In particular, can we hope to start from a mildly non-trivial lower bound for general algebraic circuits (e.g. super-linear or just super-polynomial), and hope to amplify it to get a stronger lower bound (super-polynomial or truly exponential respectively). In the context of non-commutative algebraic circuits, Carmosino, Impagliazzo, Lovett and Mihajlin [CILM18] recently showed such results, but no such result appears to be known for commutative algebraic circuits.

Another interesting open question is to see which of the *bootstrapping* results, or even the results about the "hardness-randomness connections" can be interpreted to the setting of *whitebox PIT*. However, such a result would perhaps require a more unified understanding of whitebox PIT, which could then be used as an analogue for hitting sets.

Recent developments. A recent work of Guo, Kumar, Saptharishi and Solomon [GKSS19] answers some natural questions that follow from the results in Chapter 5 over the fields of characteristic zero. They show that (over characteristic zero fields) for any constant k, the saving of *a single point* from the trivial hitting set (hitting sets of size $(s+1)^k - 1$) for k-variate, degree s circuits of size s *for all large* s, gives us hitting sets of size poly(s) for all degree s circuits of size s. Moreover, they can get to their hypothesis by assuming a strong enough lower bound against constant variate circuits. Their result uses a novel algebraic *pseudorandom generator (PRG)* construction that yields an almost optimal conversion of "hardness to randomness" in the relevant settings over fields of characteristic zero.

For fields of positive characteristic, the recent work of Andrews [And20] allows us to go from strong enough constant variate lower bounds to the hypothesis of the main theorem in Chapter 5. This means that such a lower bound would lead to almost-polynomial sized hitting sets for multivariate algebraic circuits. Therefore, a natural and interesting open question here is to design a strong algebraic PRG (analogous to [GKSS19]) that works over fields of positive characteristic.

Based on Chapter 6. The most natural question here is to extend the results from Chapter 6 to the entire class VP over all fields. Our proofs crucially use the complexity of the coefficients and it is not clear if the same ideas can be used for such an extension. As discussed towards the end of the chapter, the recent work on equations for VNP ([KRST20]) only makes this question more interesting, for both VP and VNP. A first line of attack could be to understand the complexity of defining equations for constant-free versions of the classes VP and VNP, namely VP^0 and VNP^0. Perhaps we can unconditionally say something interesting about the equations for these restricted classes.

In general, proving non-trivial upper (and lower) bounds on the circuit complexity and degree of equations of varieties associated with natural algebraic models is an interesting question. In addition to proving such bounds for VP as mentioned above, it is also of great interest to prove such bounds for other models, like formulas or algebraic branching programs.

Bibliography

[AB03] Manindra Agrawal and Somenath Biswas. Primality and identity testing via Chinese remaindering. *Journal of the ACM*, 50(4):429–443, 2003. Preliminary version in the *40th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1999)*. (13), (17), (27), (43), (62), (63)

[AD08] Scott Aaronson and Andrew Drucker. Arithmetic natural proofs theory is sought. Shtetl Optimized: Scott Aaronson's Blog, 2008. (14), (87), (88), (89)

[AGKS15] Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Hitting-Sets for ROABP and Sum of Set-Multilinear Circuits. *SIAM Journal of Computing*, 44(3):669–697, 2015. Pre-print available at `arXiv:1406.7535`. (13), (15), (16), (30), (31), (33), (42), (43), (44), (62)

[Agr05] Manindra Agrawal. Proving Lower Bounds Via Pseudo-random Generators. In *Proceedings of the 25th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2005)*, pages 92–105, 2005. (9), (18), (71), (72), (76)

[AGS19] Manindra Agrawal, Sumanta Ghosh, and Nitin Saxena. Bootstrapping variables in algebraic circuits. *Proceedings of the National Academy of Sciences*, 116(17):8107–8118, 2019. Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC 2018). `eccc:TR18-035`. (10), (17), (18), (69), (70), (71), (73), (78), (110)

[AJMV98] Eric Allender, Jia Jiao, Meena Mahajan, and V. Vinay. Non-Commutative Arithmetic Circuits: Depth Reduction and Size Lower Bounds. *Theoretical Computer Science*, 209(1-2):47–86, 1998. `eccc:TR95-043`. (8), (16), (32), (36), (37)

[And20] Robert Andrews. Algebraic Hardness versus Randomness in Low Characteristic. In *Proceedings of the 35th Annual Computational Complexity Conference (CCC 2020)*, pages 37:1–37:32, 2020. `arXiv:2005.10885`. (10), (110)

[AR16] Vikraman Arvind and S. Raja. Some Lower Bound Results for Set-Multilinear Arithmetic Computations. *Chicago Journal of Theoretical Computer Science*, 2016. (30), (33), (36)

[AS18] Vikraman Arvind and Srikanth Srinivasan. On the Hardness of the Noncommutative Determinant. *Computational Complexity*, 27(1):1–29, 2018. Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC 2010). (12)

[ASS13] Manindra Agrawal, Chandan Saha, and Nitin Saxena. Quasi-polynomial hitting-set for set-depth-Δ formulas. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC 2013)*, pages 321–330, 2013. `eccc:TR12-113`. (13), (30), (61)

[ASSS16] Manindra Agrawal, Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena. Jacobian Hits Circuits: Hitting Sets, Lower Bounds for Depth-D Occur-k Formulas and Depth-3 Transcendence Degree-k Circuits. *SIAM Journal of Computing*, 45(4):1533–1562, 2016. Preliminary version in the *44th Annual ACM Symposium on Theory of Computing (STOC 2012)*. `arXiv:1111.0582`. (13)

[AV08] Manindra Agrawal and V. Vinay. Arithmetic Circuits: A Chasm at Depth Four. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008)*, pages 67–75, 2008. `eccc:TR08-062`. (8)

[AvMV15] Matthew Anderson, Dieter van Melkebeek, and Ilya Volkovich. Deterministic polynomial identity tests for multilinear bounded-read formulae. *Computational Complexity*, 24(4):695–776, 2015. Preliminary version in the *26th Annual IEEE Conference on Computational Complexity (CCC 2011)*. (13)

[AW09] Scott Aaronson and Avi Wigderson. Algebrization: A New Barrier in Complexity Theory. *ACM Transactions on Computation Theory*, 1(1), February 2009. Preliminary version in the *40th Annual ACM Symposium on Theory of Computing (STOC 2008)*. (87)

[Bar68] Erwin H Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Mathematics of computation*, 22(103):565–578, 1968. (77)

[BCPS18] Anurag Bishnoi, Pete L. Clark, Aditya Potukuchi, and John R. Schmitt. On Zeros of a Polynomial in a Finite Grid. *Combinatorics, Probability and Computing*, 27(3):310–333, 2018. (4)

[BGS75] Theodore Baker, John Gill, and Robert Solovay. Relativizations of the $\mathcal{P} =?\mathcal{NP}$ Question. *SIAM Journal on Computing*, 4(4):431–442, 1975. (87)

[BIJL18] Markus Bläser, Christian Ikenmeyer, Gorav Jindal, and Vladimir Lysikov. Generalized matrix completion and algebraic natural proofs. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC 2018)*, pages 1193–1206, 2018. (91), (93), (94) [BIL+19] Markus Bläser, Christian Ikenmeyer, Vladimir Lysikov, Anurag Pandey, and Frank-Olaf Schreyer. Variety Membership Testing, Algebraic Natural Proofs, and Geometric Complexity Theory. *CoRR*, abs/1911.02534, 2019. (93), (94)

[BLS16] Nikhil Balaji, Nutan Limaye, and Srikanth Srinivasan. An Almost Cubic Lower Bound for $\Sigma\Pi\Sigma$ Circuits Computing a Polynomial in VP. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:143, 2016. (11)

[Bre74] Richard P. Brent. The Parallel Evaluation of General Arithmetic Expressions. *Journal of the ACM*, 21(2):201–206, April 1974. (7)

[Bür00] Peter Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*, volume 7 of *Algorithms and Computation in Mathematics*. Springer, 2000. (77)

[Cho11] Timothy Y. Chow. Almost-natural proofs. *Journal of Computer and System Sciences*, 77(4):728–737, 2011. Preliminary version in the *49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008)*. (94)

[CILM18] Marco L. Carmosino, Russell Impagliazzo, Shachar Lovett, and Ivan Mihajlin. Hardness Amplification for Non-Commutative Arithmetic Circuits. In *Proceedings of the 33rd Annual Computational Complexity Conference (CCC 2018)*, pages 12:1–12:16, 2018. `eccc:TR18-095`. (70), (110)

[CKR+20] Prerona Chatterjee, Mrinal Kumar, C. Ramya, Ramprasad Saptharishi, and Anamay Tengse. On the Existence of Algebraically Natural Proofs. *CoRR*, abs/2004.14147, 2020. Pre-print available at `arXiv:2004.14147`. (18), (87)

[DL78] Richard A. DeMillo and Richard J. Lipton. A Probabilistic Remark on Algebraic Program Testing. *Information Processing Letters*, 7(4):193–195, 1978. (4), (42), (69), (81), (82)

[DMPY12] Zeev Dvir, Guillaume Malod, Sylvain Perifel, and Amir Yehudayoff. Separating multilinear branching programs and formulas. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 615–624. ACM, 2012. Pre-print available at `eccc:TR11-134`. (12)

[DS13] Vladimir Ivanovich Danilov and Vyacheslav V Shokurov. *Algebraic Geometry I: Algebraic Curves, Algebraic Manifolds and Schemes*, volume 23. Springer Science & Business Media, 2013. (106)

[DSY09] Zeev Dvir, Amir Shpilka, and Amir Yehudayoff. Hardness-Randomness Tradeoffs for Bounded Depth Arithmetic Circuits. *SIAM J. Comput.*, 39(4):1279–1293, 2009. Preliminary version in the *40th Annual ACM Symposium on Theory of Computing (STOC 2008)*. (10)

[EGOW18] Klim Efremenko, Ankit Garg, Rafael Oliveira, and Avi Wigderson. Barriers for Rank Methods in Arithmetic Complexity. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 1:1–1:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. (94)

[FGS18] Michael A. Forbes, Sumanta Ghosh, and Nitin Saxena. Towards Blackbox Identity Testing of Log-Variate Circuits. In *Proceedings of the 45th International Colloquium on Automata, Languages and Programming (ICALP 2018)*, volume 107 of *LIPIcs*, pages 54:1–54:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. (13), (16), (17), (61), (62), (63), (66), (67), (110)

[FGT16] Stephen A. Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite perfect matching is in quasi-NC. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC 2016)*, pages 754–763, 2016. `eccc:TR15-177`. (63), (67)

[FLMS15] Hervé Fournier, Nutan Limaye, Guillaume Malod, and Srikanth Srinivasan. Lower Bounds for Depth-4 Formulas Computing Iterated Matrix Multiplication. *SIAM Journal of Computing*, 44(5):1173–1201, 2015. Preliminary version in the *46th Annual ACM Symposium on Theory of Computing (STOC 2014)*. `eccc:TR13-100`. (11)

[For14] Michael Forbes. *Polynomial Identity Testing of Read-Once Oblivious Algebraic Branching Programs*. PhD thesis, Massachusetts Institute of Technology, 2014. (3), (17), (63), (65), (98), (103)

[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. ACM, 2012. Pre-print available at `eccc:TR11-147`. (13), (61)

[FS13] Michael A. Forbes and Amir Shpilka. Quasipolynomial-Time Identity Testing of Non-commutative and Read-Once Oblivious Algebraic Branching Programs. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2013)*, pages 243–252, 2013. Full version at `arXiv:1209.2408`. (11), (13), (30), (33), (41), (53), (54), (61)

[FSS84] Merrick Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical systems theory*, 17(1):13–27, 1984. (88)

[FSS14] Michael A. Forbes, Ramprasad Saptharishi, and Amir Shpilka. Hitting sets for multilinear read-once algebraic branching programs, in any order. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 2014)*, pages 867–875, 2014. (13), (16), (61) [FSV18] Michael A. Forbes, Amir Shpilka, and Ben Lee Volk. Succinct Hitting Sets and Barriers to Proving Lower Bounds for Algebraic Circuits. *Theory of Computing*, 14(1):1–45, 2018. Preliminary version in the *49th Annual ACM Symposium on Theory of Computing (STOC 2017)*. arXiv:1701.05328. (14), (18), (87), (88), (89), (90), (93), (95)

[GG20] Zeyu Guo and Rohit Gurjar. Improved Explicit Hitting-Sets for ROABPs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*, volume 176 of *LIPIcs*, pages 4:1–4:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. (13), (62)

[GKKS14] Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Approaching the Chasm at Depth Four. *Journal of the ACM*, 61(6):33:1–33:16, 2014. Preliminary version in the *28th Annual IEEE Conference on Computational Complexity (CCC 2013)*. Pre-print available at eccc:TR12-098. (11)

[GKKS16] Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Arithmetic Circuits: A Chasm at Depth 3. *SIAM Journal of Computing*, 45(3):1064–1079, 2016. Preliminary version in the *54th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2013)*. eccc:TR13-026. (9)

[GKS17] Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Identity Testing for Constant-Width, and Commutative, Read-Once Oblivious ABPs. *Theory of Computing*, 13(1):1–21, 2017. Preliminary version in the *31st Annual Computational Complexity Conference (CCC 2016)*. arXiv:1601.08031. (13), (15), (16), (31), (33), (46), (55), (61)

[GKSS17] Joshua A. Grochow, Mrinal Kumar, Michael E. Saks, and Shubhangi Saraf. Towards an algebraic natural proofs barrier via polynomial identity testing. *CoRR*, abs/1701.01717, 2017. Pre-print available at arXiv:1701.01717. (14), (18), (87), (89), (90), (93), (95)

[GKSS19] Zeyu Guo, Mrinal Kumar, Ramprasad Saptharishi, and Noam Solomon. Derandomization from Algebraic Hardness: Treading the Borders. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 147–157. IEEE Computer Society, 2019. (10), (110)

[GKST17] Rohit Gurjar, Arpita Korwar, Nitin Saxena, and Thomas Thierauf. Deterministic Identity Testing for Sum of Read-once Oblivious Arithmetic Branching Programs. *Computational Complexity*, 26(4):835–880, 2017. Preliminary version in the *30th Annual Computational Complexity Conference (CCC 2015)*. arXiv:1411.7341. (13), (15), (16), (30), (31), (33), (47), (48), (57), (109) [GMOW19] Ankit Garg, Visu Makam, Rafael Oliveira, and Avi Wigderson. More Barriers for Rank Methods, via a "Numeric to Symbolic" Transfer. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 824–844. IEEE Computer Society, 2019. (94)

[Gro13] Joshua Grochow, 2013. `http://cstheory.stackexchange.com/questions/19261/degree-restriction-for-polynomials-in-mathsfvp/19268#19268`. (3)

[Gro15] Joshua A. Grochow. Unifying Known Lower Bounds via Geometric Complexity Theory. *Computational Complexity*, 24(2):393–475, 2015. (14), (89), (90)

[GST20] Nikhil Gupta, Chandan Saha, and Bhargav Thankey. A Super-Quadratic Lower Bound for Depth Four Arithmetic Circuits. In *Proceedings of the 35th Annual Computational Complexity Conference (CCC 2020)*, volume 169 of *LIPIcs*, pages 23:1–23:31. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. Pre-print available at `eccc:TR20-028`. (11)

[GTV18] Rohit Gurjar, Thomas Thierauf, and Nisheeth K. Vishnoi. Isolating a Vertex via Lattices: Polytopes with Totally Unimodular Faces. In *Proceedings of the 45th International Colloquium on Automata, Languages and Programming (ICALP 2018)*, pages 74:1–74:14, 2018. (63), (67)

[Hås86] Johan Håstad. Almost Optimal Lower Bounds for Small Depth Circuits. In *Proceedings of the −1882th Annual ACM Symposium on Theory of Computing (STOC 86)*, page 6–20, New York, NY, USA, 1986. Association for Computing Machinery. (88)

[Hog89] Jan P. Hogendijk. Sharaf al-Dīn Ṭūsī on the number of positive roots of cubic equations. *Historia Mathematica*, 16(1):69 – 85, 1989. (75)

[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 (STOC 1980)*, pages 262–272, 1980. (9), (18), (23), (71), (72), (76), (91), (100)

[HY11a] Pavel Hrubeš and Amir Yehudayoff. Arithmetic Complexity in Ring Extensions. *Theory of Computing*, 7(8):119–129, 2011. (106)

[HY11b] Pavel Hrubeš and Amir Yehudayoff. Homogeneous Formulas and Symmetric Polynomials. *Computational Complexity*, 20(3):559–578, 2011. (12)

[HY16] Pavel Hrubeš and Amir Yehudayoff. On Isoperimetric Profiles and Computational Complexity. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP 2016)*, pages 89:1–89:12, 2016. `eccc:TR15-164`. (33), (40), (41), (49), (50)

[HY20] Pavel Hrubeš and Amir Yehudayoff. Shadows of Newton polytopes. 2020. eccc:TR20-189. (63)

[Hya77] Laurent Hyafil. The Power of Commutativity. In *Proceedings of the 18th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1977)*, page 171–174, USA, 1977. IEEE Computer Society. (2)

[Hya79] Laurent Hyafil. On the Parallel Evaluation of Multivariate Polynomials. *SIAM Journal of Computing*, 8(2):120–123, 1979. Preliminary version in the *10th Annual ACM Symposium on Theory of Computing (STOC 1978)*. (36)

[IK99] Anthony Iarrobino and Vassil Kanev. *Power Sums, Gorenstein Algebras, and Determinantal Loci*. Springer-Verlag Berlin Heidelberg, 1999. (61)

[JS12] Maurice J. Jansen and Rahul Santhanam. Marginal hitting sets imply super-polynomial lower bounds for permanent. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ICTS 2012)*, pages 496–506, 2012. (70), (71), (74)

[Kal89] Erich Kaltofen. Factorization of Polynomials Given by Straight-Line Programs. In *Randomness and Computation*, pages 375–412. JAI Press, 1989. (72), (77)

[Kay12] Neeraj Kayal. An exponential lower bound for the sum of powers of bounded degree polynomials. In *Electronic Colloquium on Computational Complexity (ECCC)TR12-081*, 2012. (11)

[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)*. (9), (10), (18), (71), (72), (73), (77), (91), (108)

[KLSS17] Neeraj Kayal, Nutan Limaye, Chandan Saha, and Srikanth Srinivasan. An Exponential Lower Bound for Homogeneous Depth Four Arithmetic Circuits. volume 46, pages 307–335, 2017. Preliminary version in the *55th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2014)*. eccc:TR14-005. (11)

[Koi12] Pascal Koiran. Arithmetic Circuits: The Chasm at Depth Four Gets Wider. *Theoretical Computer Science*, 448:56–65, 2012. Pre-print available at arXiv:1006.4700. (8)

[KPTT15] Pascal Koiran, Natacha Portier, Sébastien Tavenas, and Stéphan Thomassé. A τ-Conjecture for Newton Polygons. *Foundations of Computational Mathematics*, 15:185–197, 2015. (63)

[KRST20] Mrinal Kumar, C. Ramya, Ramprasad Saptharishi, and Anamay Tengse. If VNP is hard, then so are equations for it. *CoRR*, abs/2012.07056, 2020. Pre-print available at arXiv:2012.07056. (19), (108), (111)

[KS01] Adam Klivans and Daniel A. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC 2001)*, pages 216–223, 2001. (13), (17), (27), (62), (63)

[KS14] Mrinal Kumar and Shubhangi Saraf. On the power of homogeneous depth 4 arithmetic circuits. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2014)*, 2014. Pre-print available at `eccc:TR14-045`. (11)

[KS19] Mrinal Kumar and Ramprasad Saptharishi. Hardness-Randomness Tradeoffs for Algebraic Computation. *Bulletin of EATCS*, 1(129), 2019. (10)

[KSS14] Neeraj Kayal, Chandan Saha, and Ramprasad Saptharishi. A super-polynomial lower bound for regular arithmetic formulas. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 2014)*, pages 146–153, 2014. Pre-print available at `eccc:TR13-091`. (11)

[KST16] Neeraj Kayal, Chandan Saha, and Sébastien Tavenas. An Almost Cubic Lower Bound for Depth Three Arithmetic Circuits. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 33:1–33:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. (11)

[KST19] Mrinal Kumar, Ramprasad Saptharishi, and Anamay Tengse. Near-optimal Bootstrapping of Hitting Sets for Algebraic Circuits. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 639–646. SIAM, 2019. `eccc:TR18-132`. (10), (17), (69)

[KV20] Mrinal Kumar and Ben Lee Volk. A Polynomial Degree Bound on Defining Equations of Non-rigid Matrices and Small Linear Circuits. *CoRR*, abs/2003.12938, 2020. Pre-print available at `arXiv:2003.12938`. (94)

[LLS19] Guillaume Lagarde, Nutan Limaye, and Srikanth Srinivasan. Lower Bounds and PIT for Non-commutative Arithmetic Circuits with Restricted Parse Trees. *Computational Complexity*, 28(3):471–542, 2019. Preliminary version in the *42nd Internationl Symposium on the Mathematical Foundations of Computer Science (MFCS 2017)*. `eccc:TR17-077`. (15), (29), (30), (46), (47), (109)

[LMP19] Guillaume Lagarde, Guillaume Malod, and Sylvain Perifel. Non-commutative computations: lower bounds and polynomial identity testing. *Chicago Journal of Theoretical Computer Science*, 2019. `eccc:TR16-094`. (12), (15), (16), (29), (30), (32), (35), (50), (51), (59) [LMS16] Nutan Limaye, Guillaume Malod, and Srikanth Srinivasan. Lower Bounds for Non-Commutative Skew Circuits. *Theory of Computing*, 12(1):1–38, 2016. eccc:TR15-22. (29)

[MV97] Meena Mahajan and V. Vinay. A Combinatorial Algorithm for the Determinant. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1997)*, pages 730–738, 1997. Available on citeseer:10.1.1.31.1673. (7)

[MV17] Daniel Minahan and Ilya Volkovich. Complete Derandomization of Identity Testing and Reconstruction of Read-Once Formulas. *ACM Transactions on Computation Theory*, 10(3):10:1–10:11, 2017. Preliminary version in the *32nd Annual Computational Complexity Conference (CCC 2017)*. eccc:TR16-171. (13)

[MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987. Preliminary version in the *19th Annual ACM Symposium on Theory of Computing (STOC 1987)*. (62)

[Nis91] Noam Nisan. Lower bounds for non-commutative computation. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC 1991)*, pages 410–418, 1991. Available on citeseer:10.1.1.17.5067. (12), (13), (16), (29), (30), (32), (59)

[Nis92] Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992. (30)

[NW94] Noam Nisan and Avi Wigderson. Hardness vs Randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994. Available on citeseer:10.1.1.83.8416. (9), (75)

[NW97] Noam Nisan and Avi Wigderson. Lower bounds on arithmetic circuits via partial derivatives. *Computational Complexity*, 6(3):217–234, 1997. Available on citeseer:10.1.1.90.2644. (10), (14), (65)

[Ore22] Øystein Ore. Über höhere Kongruenzen. *Norsk Mat. Forenings Skrifter*, 1(7):15, 1922. (4), (42), (69), (81), (82)

[OSV16] Rafael Oliveira, Amir Shpilka, and Ben Lee Volk. Subexponential Size Hitting Sets for Bounded Depth Multilinear Formulas. *Computational Complexity*, 25(2):455–505, 2016. Preliminary version in the *30th Annual Computational Complexity Conference (CCC 2015)*. arXiv:1411.7492. (13)

[PS82] Christos Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982. (64)

[Raz87] Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987. (88)

[Raz06] Ran Raz. Separation of Multilinear Circuit and Formula Size. *Theory of Computing*, 2(1):121–135, 2006. Preliminary version in the *45th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2004)*. Pre-print available at `eccc:TR04-042`. (12)

[Raz09] Ran Raz. Multi-Linear Formulas for Permanent and Determinant are of Super-Polynomial Size. *Journal of the ACM*, 56(2), 2009. Preliminary version in the *36th Annual ACM Symposium on Theory of Computing (STOC 2004)*. Pre-print available at `eccc:TR03-067`. (12)

[Raz10] Ran Raz. Elusive Functions and Lower Bounds for Arithmetic Circuits. *Theory of Computing*, 6(1):135–177, 2010. (97)

[RR97] Alexander A. Razborov and Steven Rudich. Natural Proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997. Preliminary version in the *26th Annual ACM Symposium on Theory of Computing (STOC 1994)*. (14), (87), (88), (89), (94)

[RS05] Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005. Preliminary version in the *19th Annual IEEE Conference on Computational Complexity (CCC 2004)*. (30)

[RS11] Kristian Ranestad and Frank-Olaf Schreyer. On the rank of a symmetric form. *Journal of Algebra*, 346(1):340–342, 2011. (11), (61)

[RY09] Ran Raz and Amir Yehudayoff. Lower Bounds and Separations for Constant Depth Multilinear Circuits. *Computational Complexity*, 18(2):171–207, 2009. Preliminary version in the *23rd Annual IEEE Conference on Computational Complexity (CCC 2008)*. Pre-print available at `eccc:TR08-006`. (12)

[Sap15] Ramprasad Saptharishi. A survey of lower bounds in arithmetic circuit complexity. Github survey, 2015. (97)

[Sax08] Nitin Saxena. Diagonal Circuit Identity Testing and Lower Bounds. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP 2008)*, pages 60–71, 2008. Pre-print available at `eccc:TR07-124`. (11), (61)

[Sch80] Jacob T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *Journal of the ACM*, 27(4):701–717, 1980. (4), (42), (69), (81), (82)

[Sie14] Carl L Siegel. Über einige anwendungen diophantischer approximationen. In *On Some Applications of Diophantine Approximations*, pages 81–138. Springer, 2014. (19), (96), (102)

[Smo87] Roman Smolensky. Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC 1987)*, pages 77–82, 1987. (88)

[SS10] Nitin Saxena and C. Seshadhri. From Sylvester-Gallai Configurations to Rank Bounds: Improved Black-Box Identity Test for Depth-3 Circuits. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS 2010)*, pages 21–29, 2010. arXiv:1002.0145. (13)

[ST18] Ramprasad Saptharishi and Anamay Tengse. Quasipolynomial Hitting Sets for Circuits with Restricted Parse Trees. In *Proceedings of the 38th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018)*, volume 122 of *LIPIcs*, pages 6:1–6:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. eccc:TR17-135. (15), (29)

[ST20] Amit Sinhababu and Thomas Thierauf. Factorization of Polynomials Given By Arithmetic Branching Programs. In *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 33:1–33:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. Pre-print available at eccc:TR20-077. (10), (73)

[Str69] V. Strassen. Gaussian Elimination is not Optimal. *Numerische Mathematik*, 13(3):354–356, 1969. (2)

[Str73] Volker Strassen. Die Berechnungskomplexität Von Elementarsymmetrischen Funktionen Und Von Interpolationskoeffizienten. *Numerische Mathematik*, 20(3):238–251, 1973. (2)

[SV15] Amir Shpilka and Ilya Volkovich. Read-once polynomial identity testing. *Computational Complexity*, 24(3):477–532, 2015. Preliminary version in the *40th Annual ACM Symposium on Theory of Computing (STOC 2008)*. (48), (61)

[SW01] Amir Shpilka and Avi Wigderson. Depth-3 arithmetic circuits over fields of characteristic zero. *Computational Complexity*, 10(1):1–27, 2001. Preliminary version in the *14th Annual IEEE Conference on Computational Complexity (CCC 1999)*. eccc:TR99-023. (10)

[Tav15] Sébastien Tavenas. Improved bounds for reduction to depth 4 and depth 3. *Inf. Comput.*, 240:2–11, 2015. Preliminary version in the *38th Internationl Symposium on the Mathematical Foundations of Computer Science (MFCS 2013)*. arXiv:1304.5777. (8)

[Val79] Leslie G. Valiant. Completeness Classes in Algebra. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC 1979)*, pages 249–261, 1979. (2), (3), (7)

[Val82] Leslie G. Valiant. Reducibility by algebraic projections. *L'Enseignement Mathematique: Logic and Algorithmic, Geneva*, 2:365–380, 1982. (3)

[VSBR83] Leslie G. Valiant, Sven Skyum, S. Berkowitz, and Charles Rackoff. Fast Parallel Computation of Polynomials Using Few Processors. *SIAM Journal of Computing*, 12(4):641–644, 1983. Preliminary version in the *6th Internationl Symposium on the Mathematical Foundations of Computer Science (MFCS 1981)*. (8), (16), (32), (36), (37), (97)

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979. (4), (42), (69), (81), (82)