Polynomial Identity Testing of Read-Once Oblivious Algebraic Branching Programs

by

Michael Andrew Forbes
S.B., Massachusetts Institute of Technology (2009)
S.M., Massachusetts Institute of Technology (2011)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science
at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2014

© Massachusetts Institute of Technology 2014. All rights reserved.
Polynomial Identity Testing of Read-Once Oblivious Algebraic Branching Programs

by

Michael Andrew Forbes

Submitted to the Department of Electrical Engineering and Computer Science on May 21, 2014, in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

We study the problem of obtaining efficient, deterministic, black-box polynomial identity testing algorithms (PIT) for algebraic branching programs (ABPs) that are read-once and oblivious. This class has an efficient, deterministic, white-box polynomial identity testing algorithm (due to Raz and Shpilka [RS05]), but prior to this work there was no known such black-box algorithm.

The main result of this work gives the first quasi-polynomial sized hitting set for size $S$ circuits from this class, when the order of the variables is known. As our hitting set is of size $\exp(\lg^2 S)$, this is analogous (in the terminology of boolean pseudorandomness) to a seed-length of $\lg^2 S$, which is the seed length of the pseudorandom generators of Nisan [Nis92] and Impagliazzo-Nisan-Wigderson [INW94] for read-once oblivious boolean branching programs. Thus our work can be seen as an algebraic analogue of these foundational results in boolean pseudorandomness.

We also show that several other circuit classes can be black-box reduced to read-once oblivious ABPs, including non-commutative ABPs and diagonal depth-4 circuits, and consequently obtain similar hitting sets for these classes as well.

To establish the above hitting sets, we use a form of dimension reduction we call a rank condenser, which maps a large-dimensional space to a medium-dimensional space, while preserving the rank of low-dimensional subspaces. We give an explicit construction of a rank condenser that is randomness efficient and show how it can be used as a form of oblivious Gaussian elimination.

As an application, we strengthen a result of Mulmuley [Mul12a], and show that derandomizing a particular case of the Noether Normalization Lemma is reducible to black-box PIT of read-once oblivious ABPs. Using our hitting set results, this gives a derandomization of Noether Normalization in that case.

Thesis Supervisor: Scott Aaronson
Title: Associate Professor

Thesis Supervisor: Amir Shpilka
Title: Associate Professor
Acknowledgements

Dad, I have a question.

Yes, Mike?

Why does \(\text{question}\)?

You know, that is an excellent question. \(\text{answer}\)

The acknowledgments section is, arguably, the hardest part to write of any thesis. For the debt one can feel to those who have made us who we are, that debt looms so large in our minds, but feels so small when written in such tiny fonts.

My biggest thanks goes to my father, who this thesis is dedicated to. As an academic himself, he has shared with me the joys of learning from a young age. Not only has he let my curiosity flourish by putting up with my (many) questions, he saw to it that my education was top-notch. Of course, there is also the personal touch: doing a Ph.D. (or an MIT undergraduate) is no trivial affair, and his support has meant a lot to me over the years. I really value the fact that despite our differing fields, we can talk about research with each other, and in these moments, I really feel I’ve earned the title of “Dr. Forbes” (which I mostly still associate with him).

For Sunny, my girlfriend, my Ph.D. has been a journey for both of us. I think she now knows more about conference reviewing then she ever wanted to. My extensive travel (around the world, for conferences and extended research visits) certainly only adds to the burden. But weighed against how much she makes me smile, it doesn’t seem so bad.

For the rest of my family, my mother and Greg, and Mary-Jo and Thomas, I appreciate the support you’ve given me over the years, especially when my thesis topic might as well be in Greek. When the end of fall-term madness ends each year, it is always comforting to go home and relax.

To all of my friends (near and far) through graduate school (Doug Adams, John Chai, David Chen, Peder Hals, Jeff Ketchersid, Danielle Le, Kendra Leigh, Randy Li, Joann Lin, Josh Monzon, Joey Zhou), thanks for all of the board games and good times.

My two research supervisors, Scott and Amir, guided my journey. I really have to thank Scott for latitude he gave me to explore my own topics in my early years, as well as the encouragement to do so. My first semester of grad school, when I was working on tensor rank lower bounds, I thought I proved a super-linear lower rank lower bound (which would have been a major result). It withstood scrutiny for only a
day or two (Boris and Jacob caught the bug), and in that time I had to announce — and retract — my claim to Scott via email. That he encouraged me to continue after this episode helped me keep my spirits up in ways that are hard to explain.

Many people often ask me how I got to working with Amir — it is all Madhu Sudan’s fault. Madhu taught me “advanced complexity theory”, and his class introduced me to the polynomial identity testing (PIT) problem (the subject of this thesis) for the second time (Mike Sipser’s was the first). This second time was enough to convince me that I wanted to work on this problem “eventually”. However, at the time the research on “ΣΠΣ(k) circuits” was all the rage, and it seemed too difficult to make a contribution. Thankfully, Madhu asked me after my first year of graduate school to proofread a survey by Amir (Shpilka) and Amir Yehudayoff [SY10], and this forced me to actually learn the subject. When Amir came to Microsoft Research New England (where Madhu was) during my second year of graduate school, this sealed my fate.

Amir has been a great guide throughout my research career. While it seemed completely natural at the time, I now fully appreciate the dedication he put in when we worked on our first research paper together essentially entirely over email. Subsequent visits to Israel on my part lessened the need for this, but the time he was able to devote to me when I visited is much appreciated (and was very fruitful).

To my other co-authors from my graduate years (Boris Alexeev, Alessandro Chiesa, Neeraj Kayal, Rajat Mittal, Chandan Saha, Ramprasad Saptharishi, Jacob Tsimerman, and Sergey Yekhanin), I owe you many thanks for your patience, as well as your insight.

In terms of research, I’d like to thank Peter Bürgisser for explaining the work of Mulmuley, and the Complexity Theory week at Mathematisches Forschungsinstitut Oberwolfach for facilitating that conversation. Without that conversation, the results of Chapter 9 could not have been discovered. Likewise, I’d like to thank Avi Wigderson for questions that led to the better results for constant-width roABPs. I’d also like to thank Omer Reingold and Raghu Meka for various helpful conversations.

To my office mates (Mohammed Bavarian, Elette Boyle, Jing Chen, Alessandro Chiesa, Ionna Ivan, Brendan Juba, Swastik Kopparty, Ben Rossman, Zeyuan Zhu): thanks for the good company, and for dealing with my eccentricities (eg, the standing desk).

I’ve found that the theoretical computer science community is a very friendly place, and I’ve made quite a few friends from my time at MIT and the Technion (Arnab Bhattacharyya, Eric Blais, Gil Cohen, Matt Cou drown, Alan Deckelbaum, Andy Drucker, Ariel Gabizon, Alan Guo, Ankit Gupta, Elad Haramaty, Piotr Indyk, Prithi Kamath, G Kamath, Yo hay Kaplan, Aleksander Madry, Ankur Moitra, Shay Moran, Jelani Nelson, Rotem Oshman, Eric Price, Noga Ron-Zewi, Shubhangi Saraf, Ludwig Schmidt, Aaron Sidford, Thomas Steinke, Justin Thaler, Thomas Vidick, Ben lee Volk, Ilya Volkovich, Amir Yehudayoff, Henry Yuen). Many thanks to them all (and those I forgot to add).

Academically, there are so many people to thank. I’ll start with my first brush with complexity theory (although, I didn’t know it then) just before high school at a summer camp run by the Center for Talented Youth (CTY). Dr. Tom Ottinger taught me about Eulerian and Hamiltonian graphs and exposed me to the idea that
while the former is well-characterized, the latter is not (this disparity is essentially that of P versus NP). This was startling to me at the time, and for some reason I decided to explore it thinking I could make progress. While my efforts were mostly computational (testing graphs for Hamiltonicity and looking for patterns) and I only figured out basic things (that every Hamiltonian graph is 1-tough), encouragement by Eric Walstein, Dr. Lawrence Washington, and Dr. Lawrence Somer really meant a lot to me.

During the summer before senior year in high school, I worked with Dr. Samir Khuller on combinatorial approximation algorithms for vehicle routing problems. Samir was an amazing mentor — while I knew various combinatorial algorithms already, he taught me even more algorithms one-on-one, and I’ll never cease to be amazed at the cleverness of Christofides’s algorithm for the Traveling Salesman Problem.

Turning to undergrad, I have to thank Prof. Alar Toomre for making differential equations actually exciting, and he is more or less the reason I was a mathematics major. This choice led me to Michael Goemans as an academic advisor, whose advice was gold. He pushed me into taking more advanced math classes than I thought I was ready for, and thankfully he saw my real potential. I also have to thank him for teaching me matroids, which make a cameo appearance in this thesis (Lemma 5.4.2).

Mike Sipser taught me complexity theory, and he is the reason I am a complexity theorist. As all who have taken his class know, Mike is extremely well-organized, his lectures are flawless, and his textbook is a good read. He bestowed upon me a sense of wonder that basic questions such as P versus NP can still be open. I also have to thank Mike for the opportunity to TA for him twice.

Dr. James Lawrence gave me my first research experience in constructing pseudorandom objects (though, I didn’t know to call it that), and I learned a fair bit of math during my time in my internship with him. Scott gave me my first research experience with complexity theory (in undergrad), and the small results I got that summer whetted my appetite for research in complexity. My two internships at Microsoft Research (under Neeraj Kayal and later Sergey Yekhanin) both left me with new research interests that continue to distract me (in the good way).

As someone who feels compelled to dig into the details, I also want to thank those authors whose textbooks have influenced me by their clarity. On a more basic level, Rudin’s Principles of Mathematical Analysis and Munkres’ Topology (I also had the pleasure of having Munkres as a teacher) were paragons of clarity, and I owe my mathematical maturity to them. More in terms of research, Salil Vadhan’s Vad12 monograph on pseudorandomness (and his class on the subject) taught me a great deal, and prepared me for many of the unexpected connections my work has with previous boolean pseudorandomness constructions. Victor Shoup’s Sho09 book on algebra and number theory was perfect for going into the intersection of algebra and computation, and I really appreciate him for answering my emails about it. Finally, the book by Cox, Little, and O’Shea CLO07 about computational algebraic geometry is simply a joy.
For my father, who always answers my questions.
Contents

Contents

1 Notation 13

List of Folklore Facts 17

List of Theorems Cited Without Proofs 19

2 Introduction 21

2.1 This Thesis 25

2.1.1 Forbes-Shpilka [FS13b] 26

2.1.2 Forbes-Shpilka [FS13a] 29

3 Polynomial Identity Testing (PIT) 31

3.1 Algebraic Computation 31

3.2 Polynomial Identity Testing (PIT) 33

3.2.1 Interpolation Sets 39

3.2.2 Generators 40

4 Algebraic Branching Programs (ABPs) and their Variants 43

4.1 Algebraic Branching Programs (ABPs) 43

4.2 Layered Algebraic Branching Programs 45

4.3 Trace of Matrix Powers 52

4.4 Read-Once Oblivious Algebraic Branching Programs (roABPs) 55

4.4.1 Set-Multilinear ABPs (smABPs) 57

4.5 Lower Bounds for roABPs 58

4.5.1 The Complexity of Variable Disjoint Inner Products 58

4.5.2 roABPs as Variable Disjoint Inner Products 62

4.5.3 Polynomials Requiring Large Variable Disjoint Inner Products 64

4.5.4 Evaluation Dimension 66

5 Rank Condensers 69

5.1 Definition of Rank Condensers 70

5.2 Rank Condensers — A Generator Version 73

5.3 Randomness Efficient Rank Condensers Exist 74

5.4 Explicit Rank Condensers 80
C Hasse Derivatives
  C.1 Hasse Derivatives ..................................................... 185
  C.2 Derivatives as a Complexity Measure ............................. 194

D Counting Subspaces
  D.1 $q$-Analogues .......................................................... 201
  D.2 Counting Subspaces .................................................. 202

Bibliography .......................... 209
Chapter 1

Notation

In this work, all rings will have an identity element. Further, a non-commutative ring $R$ is in general non-commutative, but allowed to be commutative. That a non-commutative ring $R$ is an $F$-algebra for some field $F$ means that $F \subseteq R$, that $F$ commutes with all elements in $R$, and that $R$ is an $F$-vector space structure so that scalar multiplication by $F$ is induced by the multiplication in $R$.

**Numbers:** $\mathbb{N}$ denotes the natural numbers $\mathbb{N} := \{0, 1, 2, \ldots\}$, which includes zero. Variables that are used without indication of their domain are to be considered natural numbers.

When a variable $i$ (or otherwise similarly named) is defined only by an inequality with respect to an integer, such as “$i \geq 0$”, that variable is will only assume integral values.

Denote $[n] := \{1, \ldots, n\}$ and $\mathbb{J}_n := \{0, \ldots, n - 1\}$. The latter will be useful when dealing with polynomials as $\mathbb{J}_n$ more naturally corresponds to the possible coefficients of a degree $< n$ polynomial.

**Norms:** For a vector $\mathbf{a} \in \mathbb{Z}^n$, we define $\text{Supp}(\mathbf{a}) = \{i \mid a_i \neq 0\}$, $\|\mathbf{a}\|_0 := \sum_{i \in [n]} |a_i|^0 = |\text{Supp}(\mathbf{a})|$, $\|\mathbf{a}\|_1 := \sum_{i \in [n]} |a_i|$, $\|\mathbf{a}\|_\infty := \max_{i \in [n]} |a_i|$, and $\|\mathbf{a}\|_\times := \prod_{i \in [n]} (a_i + 1)$. See Section 8.2 for more on the $\times$-norm.

**Strings:** We use $\lambda$ to denote the empty string. When $n = 0$, $\Sigma^n = \{\lambda\}$, and thus for $n \geq 0$, $|\Sigma^n| = |\Sigma|^n$. The lexicographic order on $\{0, 1\}^n$ is defined so that $\mathbf{x} < \mathbf{y}$ for $\mathbf{x}, \mathbf{y} \in \{0, 1\}^n$ if the first $i$ where $x_i \neq y_i$ it has that $x_i = 0 < 1 = y_i$. Thus, $\{0, 1\}^2$ in lexicographic order is $\{00, 01, 10, 11\}$.

**Functions:** Given a set $S$ in some universe $U$, the indicator function for $S$ is denoted $1_S : U \rightarrow \{0, 1\}$ so that

$$1_S(x) = \begin{cases} 
1 & x \in S \\
0 & x \notin S
\end{cases}.$$  

When $S$ is the singleton $\{\alpha\}$ we will simply use $1_\alpha$. 
A similar notation will be used for the **Lagrange interpolation polynomials**, so for \( \alpha \in S \subseteq \mathbb{F} \), \( 1_{\alpha,S}(x) \) will be a polynomial such that the function induced on \( S \) is the indicator function \( 1_{\alpha} : S \to \{0, 1\} \). See Section B.2 for the formal definition of these polynomials.

**Polynomials:** Polynomials will often be written out in their monomial expansion. When "taking the coefficient of \( \bar{y}^\bar{f} \) in \( f \in \mathbb{F}[\bar{x}, \bar{y}] \)" we mean that both \( \bar{x} \) and \( \bar{y} \) are treated as variables and thus the coefficient returned is a scalar in \( \mathbb{F} \). However, when "taking the coefficient of \( \bar{y}^\bar{f} \) in \( \bar{f} \in \mathbb{F}[\bar{x}][\bar{y}] \)" we mean that \( \bar{x} \) is now part of the ring of scalars, so the coefficient will be an element of \( \mathbb{F}[\bar{x}] \).

The **(total) degree** of a polynomial \( f(\bar{x}) = \sum_{\bar{c}} c_{\bar{c}} \bar{x}^\bar{c} \), is defined as \( \deg f := \max_{\bar{c}_{\bar{x}} \neq 0} \|\bar{c}\|_1 \). The **individual degree** of a polynomial \( f(\bar{x}) = \sum_{\bar{c}} c_{\bar{c}} \bar{x}^\bar{c} \), is defined as \( \operatorname{ideg} f := \max_{\bar{c}_{\bar{x}} \neq 0} \|\bar{c}\|_\infty \).

Given a polynomial \( f \) (in \( \mathbb{F}[\bar{x}] \) or in the non-commutative \( \mathbb{F}(\bar{x}) \)), we write \( H_k(f) \) for the homogeneous part of \( f \) of degree \( k \).

**Random Variables:** We use the font “\( X \)” for random variables.

**Operations on Vectors:** Operations with vectors will often be used, and this will be done coordinate-wise. Thus, if \( \bar{f} \in (\mathbb{F}[\bar{x}])^m \) is a vector of polynomials, and \( \varphi : \mathbb{F}[\bar{x}] \to \mathbb{F}[\bar{x}] \) is some map on polynomials, then \( \varphi(\bar{f}) \) is \( \varphi \) applied coordinate-wise to \( \bar{f} \). Similarly, for \( g \in \mathbb{F}[\bar{x}] \), \( g \cdot \bar{f} \) is \( g \) multiplied on coordinate-wise on \( \bar{f} \).

As another example, we will often take coefficients of matrix- or vector-valued polynomials. That is, given a matrix \( \bar{M}(\bar{x}) \in \mathbb{F}[\bar{x}]^{n \times n} \), we can think of this matrix as either a matrix with entries lying in \( \mathbb{F}[\bar{x}] \), or can think of this matrix as a polynomial in \( \bar{x} \) with coefficients in \( \mathbb{F}^{r \times r} \). In either equivalent viewpoint, we can consider \( \operatorname{Coeff}_{\bar{x}}(\bar{M}) \), which is the coefficient operator applied entry-wise to \( \bar{M} \) (in the first viewpoint), or is simply taking the matrix-valued coefficient (in the second viewpoint).

There are two notation exceptions. The first is the binomial coefficient \( \binom{\bar{a}}{\bar{b}} := \prod_i \binom{a_i}{b_i} \), so that the binomial operator is applied coordinate-wise and then the coordinates are multiplied. The second exception is similar. Let \( \bar{f} \in \mathbb{F}[\bar{x}]^n \) be a vector of polynomials and \( \bar{a} \in \mathbb{N}^n \) an exponent vector. Then \( \bar{f}^{\bar{a}} := \prod_{i=1}^n f_i^{a_i} \). This notation will be most often used as \( \bar{x}^{\bar{a}} \) to denote the monomial \( \prod_{i=1}^n x_i^{a_i} \).

**Matrices:** Block-matrix notation will occasionally be used, and \( 0_{n \times m} \) will denote the \( n \times m \) matrix of all zeros. When \( n = m \) the notation \( 0_n \) will be used. When \( n \) and \( m \) are determined by context and their exact value is not important, the notation \( 0 \) will be used.

Matrices (and vectors) will be given sizes (numbers of rows and columns). By default, if there are \( n \) rows then they are indexed by the set \([n]\), so that \( \bar{M} \in \mathbb{F}^{n \times m} \) is indexed by \([n] \times [m] \). In other cases, matrices (or vectors) will be explicitly be given indexing sets. Sometimes this set will be \([n]\), just to be explicit. Other times this set will be \([n]\) ([n] × [m]), so that \( \bar{M} \in \mathbb{F}^{[n] \times [m]} \) will be indexed by \([n]\) × \([m]\).

14
The above indexing will also be used for selecting submatrices. That is, given $M \in \mathbb{F}^{S \times T}$ for sets $S$ and $T$, we define $M|_{S',T'}$ for $S' \subseteq S$ and $T' \subseteq T$ to be the submatrix of $M$ indexed by the rows in $S'$ and the columns in $T'$. The notation `$\cdot$' will be used to denote that there is no restriction in the relevant coordinate, so that $M|_{S',\cdot}$ is the submatrix consisting of (all of) the rows of $M$ indexed by $S'$ and $M|_{\cdot,T'}$ is the submatrix of $M$ consisting of (all of) the columns of $M$ indexed by $T'$.

When matrices are multiplied as $\prod_{s \in S} M_s$, the product will be done in the natural order on the set $S$. Thus, when $S \subseteq \mathbb{N}$, the natural order on $\mathbb{N}$ will be used. When $S = \{0,1\}^n$, the lexicographic order on $\{0,1\}^n$ will be used, as defined above.

**Miscellaneous:** Set-valued functions will be in bold, and when sets are being added or multiplied the respective symbol will be in bold. Disjoint unions will be denoted with $\sqcup$. The exponential power series is defined as $\exp(x) = \sum_{n \geq 0} \frac{x^n}{n!}$ (defined in the power series $\mathbb{Q}[x]$ over $\mathbb{Q}$), and the truncated exponential by $|\exp|_k(x) = \sum_{n=0}^{k} \frac{x^n}{n!}$.
List of Folklore Facts

3.1.3  Fact (Circuits Compute in a Well-Defined Manner) . . . . . . . . 32
3.1.5  Fact (Converting Algebraic Circuits to Fan-in 2, Efficiently) . . 32
3.1.7  Fact (Algebraic Circuits can be Evaluated Efficiently) . . . . . 33
4.2.10 Fact (Layered ABPs can be Computed by Small Low-Depth Circuits) 52
## List of Theorems Cited Without Proofs

<table>
<thead>
<tr>
<th>Section</th>
<th>Theorem (Gaussian Elimination is in ( \text{NC}^2 ), from Borodin, von zur Gathen, and Hopcroft [BvzGH82], as derandomized by Mulmuley [Mul87])</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3.3</td>
<td></td>
</tr>
<tr>
<td>9.1.3</td>
<td>Theorem (Generators for the Ring of Invariants of Simultaneous Conjugation [Pro76, Raz74, For86])</td>
</tr>
<tr>
<td>9.1.12</td>
<td>Theorem (Separating invariants yield small integral subring, Theorem 2.3.12, Derksen and Kemper [DK02], stated by Mulmuley in Theorem 2.11)</td>
</tr>
<tr>
<td>9.1.14</td>
<td>Theorem (Separating invariants characterize orbit closures, [MFK94, Mul12a])</td>
</tr>
</tbody>
</table>
Chapter 2

Introduction

Complexity Theory: This thesis concerns computational complexity, a subfield of theoretical computer science which seeks to quantify the resources required to solve computational problems. While this subfield is quite broad, its various instantiations all tend to focus on some model of computation, the resources this model uses to compute, and the natural computational tasks this model can solve. The most common model used is that of a Turing machine (see any complexity theory book, such as Sipser [Sip06]), which is a model for general purpose computation. The most common resource studied in the context of Turing machines, is that of time, or the number of steps required to finish the computation. There is also space, which is the amount of intermediate storage needed for the computation. Most relevant to this thesis is the resource of randomness. A computation on some fixed input uses randomness to succeed with good probability (with probability only over the randomness of the algorithm, not of the input), and one can measure the number of random bits the computation produces. Randomness is a resource as it must “come from somewhere”.

Given the above model of a Turing machine, the central question of complexity theory is: which computational problems require large amounts of time/space/randomness to solve, and why? While generic methods show that there exist computational problems that require large amounts of time/space/randomness in any algorithm solving these problems, this question is asking for explicit examples. Such explicit examples not only address the pure scientific questions at hand, but also can be used for algorithmic purposes, such as cryptography.

The above question is that of obtaining lower bounds for the resources required for explicit problems. However, one can strengthen this question to ask for trade-offs between resources such as time, space and randomness. Indeed, the flag-bearing question of complexity theory, the P versus NP question, is not just about proving lower bounds but also about understanding how the two resources of time and non-determinism relate. In the context of the thesis, we will be more interested in another trade-off between resources: between randomness and time.

Specifically, there is the class P of computational problems that are solvable in deterministic polynomial time (poly(n) steps on a deterministic Turing machine, for size n inputs) and we wish to compare this class with BPP, the class of computational problems that are solvable in randomized polynomial time with good probability
\((\text{poly}(n))\) steps on a randomized Turing machine, for size \(n\) inputs, with success probability \(\geq \frac{2}{3}\). By definition, \(\text{P} \subseteq \text{BPP}\), and thus the question is: does \(\text{P} = \text{BPP}\)? Put another way, this question asks whether we can substitute the resource of randomness for the resource of time, without incurring a large (super-polynomial) penalty.

Currently, there are many computational problems that are solvable efficiently with randomness, but for which no efficient deterministic algorithm is known (see an example below). While this gap may lead to the suspicion of “\(\text{P} \neq \text{BPP}\)”, there is a long line of work showing that plausible complexity theoretic conjectures imply that \(\text{P} = \text{BPP}\). This line of work is known as hardness versus randomness. It stems from the observation of Blum-Micali [BM84] and Yao [Yao82] that if a boolean function \(f: \{0, 1\}^n \rightarrow \{0, 1\}\) is sufficiently hard to compute (on average) then the output of \(f\) seems random, so that the \((n + 1)\)-bit string \((\pi, f(\pi))\) cannot be efficiently distinguished from a random \((n + 1)\)-bit string. By iterating this procedure, Blum-Micali-Yao [BM84, Yao82] showed how to stretch \(n\) random bits into \(\text{poly}(n)\) pseudorandom bits, for any \(\text{poly}(n)\). These works spurred a long line of research, but we mention two natural milestones. The first is the work of Nisan-Wigderson [NW94], who took the idea from above and showed how to stretch \(n\)-bits of randomness to \(\exp(n)\) bits of pseudorandomness, so that the existence of sufficiently hard (on average) explicit functions would imply \(\text{P} = \text{BPP}\). This line of research culminated in the work of Impagliazzo-Wigderson [IW97] (with an alternate proof by Sudan, Trevisan and Vadhan [STV01]) who showed that if there is an explicit \(n\)-bit function computable in deterministic time \(2^{O(n)}\) that requires boolean circuits of size \(2^{O(n)}\) (a worst-case assumption) then \(\text{P} = \text{BPP}\). It is considered plausible that such functions exist, and thus the conjecture that \(\text{P} = \text{BPP}\) is now widely believed.

Thus, while trading-off randomness for time (\(\text{P} = \text{BPP}\)) is conjecturally plausible, this does not resolve how to unconditional derandomize any specific computations. Thus, we now turn to the question of obtaining deterministic efficient algorithms for specific computational problems by exploiting the structure of these problems, in particular when these problems are algebraic.

**Algebraic Complexity:** The above discussion was for computation over \(\{0, 1\}\) and as such used the Turing machine model. However, in many cases when one wants to compute an algebraic object (such as a polynomial), the most natural method for doing so is computing it in a model that is inherently algebraic. As an example consider the determinant, the omnipresent polynomial from linear algebra, defined as

\[
\det X = \sum_{\pi \in \text{S}_n} \text{sgn}(\pi) \prod_{i \in [n]} x_{i, \pi(i)},
\]

where \(X\) is an \(n \times n\) matrix with \((i, j)\)-th entry \(x_{i,j}\), \(\text{S}_n\) is the group of permutations \(\pi : [n] \rightarrow [n]\), and \(\text{sgn}(\pi)\) denotes the sign of the permutation \(\pi\). To compute this polynomial when the matrix \(X\) takes in rational values, it seems most natural to only access these rational numbers via algebra operations such as multiplication and addition. Further, standard Gaussian elimination shows that this restricted access to
the input suffices, as the determinant can be computed in \(\text{poly}(n)\) algebraic operations. One can ask for more: can the determinant be computed efficiently in parallel? Berkowitz [Ber84] showed that the determinant can be computed in \(\text{NC}^2\), and this result also fits in the model of only computing with numbers as numbers. Indeed, the result of Valiant, Skyum, Berkowitz and Rackoff [VSBR83] shows that this is a general phenomenon: any polynomial that can be computed efficiently by only using multiplication and addition can also be computed efficiently in parallel (something believed to be false for boolean computation). This demonstrates that understanding the model of algebraic computation often leads to stronger conclusions than those known for boolean computation.

Thus, as algebraic computation is sufficiently general for computing algebraic objects (and polynomials in particular), this motivates studying this model itself. This leads to the notion of an \emph{algebraic circuit}. This is a directed acyclic graph whose leaves are labelled with input variables \(x_i\) or entries from the field \(\mathbb{F}\). The gates of this circuit are \(\times\) and \(+\) and correspond to multiplication and addition respectively. The \emph{size} of such a circuit is the number of edges in its underlying graph. Algebraic circuits naturally compute polynomials, and natural polynomials (eg, the determinant) that can be computed efficiently also tend to be computable in this model with small circuits. Thus, as with boolean computation, one can measure the resources needed to compute polynomials, and the principal resource here is that of circuit size.

Computing algebraically is not just applicable to algebraic objects, as it turns out that many boolean computational tasks can be \emph{algebrized}. A typical instantiation of this paradigm takes a computational problem over \(\{0, 1\}\) and embeds it into a computational problem over some finite field \(\mathbb{F}_q\). One can then use all of the tools of algebra to attack this new problem. Some particularly powerful instantiations of this paradigm where used to establish that \(\text{IP} = \text{PSPACE}\) [LFKN92, Sha92] as well as the PCP theorem [AS98, ALM+98]. More in the context of randomized (and derandomized) algorithms are the \(\text{RNC}\)-algorithm for finding maximum matchings due to Mulmuley, Vazirani and Vazirani [MVV87] and the celebrated deterministic primality test of Agrawal, Kayal and Saxena [AKS04].

These results are all particularly interesting in the context of \(\text{P versus BPP}\), as they show that algebraic problems are naturally solved with randomness. That is, in boolean computation if we have some function \(f(\overline{x}) : \{0, 1\}^n \to \{0, 1\}\) and we know that there \emph{exists} some \(\overline{x}\) with \(f(\overline{x}) = 1\), we gain no further information as the number of such \(\overline{x}\) could be anywhere between 1 and \(2^n\). However, if \(f(\overline{x}) \in \mathbb{F}[\overline{x}]\) is a polynomial, then the existence of some \(\overline{x}\) with \(f(\overline{x}) \neq 0\) means that most \(\overline{x}\) have this property (as long as \(\deg f \ll |\mathbb{F}|\)), and thus is the content of the often-used Schwartz-Zippel Lemma [Sch80, Zip79]. Indeed, the Schwartz-Zippel Lemma [Sch80, Zip79] plays a role in most of the above results.

The above works algebrize questions in boolean computation, leading to non-trivial results. The resulting algebraic questions (and their solutions) often involve randomness, as they appeal to the Schwartz-Zippel Lemma [Sch80, Zip79]. Thus, in the context of trying to derandomize \emph{particular} algorithms, one can ask: when can we replace the Schwartz-Zippel Lemma [Sch80, Zip79] with determinism?
**Polynomial Identity Testing (PIT):** This thesis develops several algorithms for polynomial identity testing (PIT). This problem, in its fullest generality, is the problem mentioned above of replacing the Schwartz-Zippel Lemma [Sch80, Zip79] with a deterministic algorithm, whenever possible. More concretely, it asks: given an algebraic circuit $C$ computing a polynomial $f(\pi)$, is $f(\pi)$ the identically zero polynomial? Note that, over large fields, a polynomial is non-zero (as a polynomial) iff it has a non-zero evaluation point. The Schwartz-Zippel Lemma [Sch80, Zip79] then tells us that over large fields, a polynomial is non-zero iff most evaluation points are non-zero. This gives a very natural randomized algorithm for testing if a polynomial is non-zero: choose a random point $\alpha$ over the field $F$ and declare $f$ non-zero iff $f(\alpha) \neq 0$.

One important feature of the above randomized PIT algorithm is that it only uses the circuit $C$ as a method for evaluating the polynomial $f$ on given inputs. Such algorithms are called black-box algorithms, as they treat the circuit as merely a black-box that computes some low-degree polynomial (that admits some small circuit computing it). This is in contrast to white-box algorithms, that probe the structure of $C$ in some way. White-box algorithms are thus less restricted, whence deriving a black-box algorithm is a stronger result. A standard argument (Lemma 3.2.9 and Lemma 3.2.10) shows that deterministic black-box PIT algorithms are equivalent to the notion of a hitting set, which is a small set $H \subseteq F^n$ of evaluation points such that for any non-zero polynomial $f$ computed by a small algebraic circuit, $f$ must evaluate to non-zero on some point in $H$. A standard argument (Lemma 3.2.14) shows that small hitting sets exist. The main question is whether small explicit hitting sets exist. If $P = BPP$ then we know such explicit hitting sets exist, and the main challenge in PIT is to develop deterministic algorithms (white-box or black-box) that unconditionally determine whether algebraic circuits compute the non-zero polynomial.

Testing whether a given algebraic circuit is non-zero is in some sense the simplest computational task one could solve about circuits. However, PIT is actually a core algorithmic routine and thus derandomizing it would also derandomize lots of other algorithms. A basic example is that this would allow one to test the equivalence of two algebraic circuits, by testing if their difference computes a non-zero polynomial. Other examples include derandomizing the RNC-algorithm for maximum matchings by Mulmuley, Vazirani, and Vazirani [MVV87] mentioned above, testing if two polynomials $f(\pi)$ and $g(\pi)$ computed by small circuits are equivalent under some shift $\pi$ (so that $f(\pi+\pi) = g(\pi)$) (Dvir, Oliveira, Shpilka [DOS14]), and also decomposing a polynomial computed by a small circuit into small circuits computing its factors (Kopparty, Saraf, Shpilka [KSS14]).

Indeed, the problem of derandomizing PIT has received a lot of attention in the past few years. In particular, many works examine a particular class of circuits $\mathcal{C}$, and design PIT algorithms only for circuits in that class. Aside from algorithmic applications, one reason for this attention is the strong connection between deterministic PIT algorithms for a class $\mathcal{C}$ and lower bounds for $\mathcal{C}$, thus instantiating the above hardness versus randomness connection for algebraic circuits. This connection was first observed by Heintz and Schnorr [HS80] (and also by Agrawal [Agr05]) for the black-box model and by Kabanets and Impagliazzo [KI04] for the white-box model (see also Dvir, Shpilka and Yehudayoff [DSY09]). This connection was also recently strengthened by a line of
work (see [VSBR83, AV08, Koi12, Tav13, GKKS13b]) showing that algebraic circuits can be *depth-reduced*, and through the hardness versus randomness connection, shows that explicit polynomial-size hitting sets for depth-3 algebraic circuits (of exponential degree) imply explicit *quasipolynomial*-size hitting sets for general algebraic circuits. This suggests that studying PIT is important even for very limited classes of circuits. Finally, the PIT problem is, in some sense, the most general problem that we know today for which we have randomized $\text{BPP}$ (in fact, $\text{coRP}$) algorithms but no polynomial time algorithms, thus studying it is a natural step towards a better understanding of the relation between $\text{BPP}$ and $\text{P}$. For more on the PIT problem we refer to the survey by Shpilka and Yehudayoff [SY10].

Although the white-box model seems to be simpler than the black-box model, for most models for which a white-box PIT algorithm is known, a black-box PIT algorithm is also known, albeit sometimes with worse parameters and a more difficult analysis. Such examples include depth-2 circuits (also known as sparse polynomials) (see [BOT88, KS01] and references therein), depth-3 $\Sigma \Pi \Sigma (k)$ circuits (see [SS12] and the many references therein), Read-$k$ formulas [AvMV11] and depth-3 tensors (also known as depth-3 set-multilinear circuits) [RS05, FS12].

While the running time of the algorithms for depth-2 circuits and $\Sigma \Pi \Sigma (k)$ circuits are essentially the same in the white-box and black-box models, for Read-$k$ formulas and set-multilinear depth-3 circuits we encounter some loss that results in a quasipolynomial running time in the black-box model compared to a polynomial time algorithm in the white-box model. However, prior to this work, the white-box algorithm of Raz-Shpilka [RS05] had no black-box analogue. In this work, we give such an analogue, as we now discuss.

## 2.1 This Thesis

This thesis is the combination of two works, Forbes-Shpilka [FS13b, FS13a], which we now describe, as well as outlining this thesis. The thesis begins with Chapter 3, which gives formal definitions of algebraic circuits and the polynomial identity testing problem. It also defines the solution concepts for white- and black-box PIT algorithms, as well as discussing the relationships between hitting sets and *generators*.

The thesis ends with several appendices, which we briefly describe here. Appendix A describes the *unit cost model* for algebraic computation that all our algorithms use. This model, while not entirely realistic in all situations, is arguably the most natural model for performing computation over arbitrary fields. Appendix B gives the needed background on algebra, such as polynomial interpolation and the Cauchy-Binet formula. Appendix C will define the notion of Hasse derivatives, and give the needed properties for the hitting sets of Chapter 8. Finally, Appendix D gives a detailed estimates on the number of $k$-dimensional subspaces of $\mathbb{F}_q^n$, as well as the probability an $n \times n$ matrix over $\mathbb{F}_q$ is full rank, as these estimates will be needed for Chapter 5.
2.1.1 Forbes-Shpilka [FS13b]

This paper is the main focus of the thesis, and is the source for much of the material of Chapter 4, 5, 6 and 7 and some of Chapter 8. As stated above, until this work, the only model for which an efficient white-box algorithm was known without a (sub-exponential) black-box counterpart was the model of non-commutative algebraic formulas, or, more generally, the models of non-commutative algebraic branching programs (ABPs) and set-multilinear algebraic branching programs. This work gives quasipolynomial-size hitting sets for this model, and we review the breakup of this work into the respective chapters.

Chapter 4 — Algebraic Branching Programs (ABPs) and their Variants:

This chapter introduces the algebraic branching program model defined by Nisan [Nis91], and gives the definitions of the subclasses of this model that we will consider. In particular, we will define the restricted class of circuits called read-once oblivious algebraic branching program. As part of this introduction, we now summarize these models.

Definition (Nisan [Nis91], see also Definition 4.2.1). A layered algebraic branching program (ABP) is a directed acyclic graph with one vertex of in-degree zero, which is called the source, and one vertex of out-degree zero, which is called the sink. The vertices of the graph are partitioned into levels numbered 0, \ldots, d. Edges may only go from level \(i - 1\) to level \(i\) for \(i = 1, \ldots, d\). The source is the only vertex at level 0 and the sink is the only vertex at level \(d\). Each edge is labeled with a affine function in the input variables. The width of an ABP is the maximum number of nodes in any layer, and the size of the ABP is the number of vertices.

Each directed source-sink path in the ABP computes a polynomial, which is a product of the labels on the edges in that path. As this work concerns non-commutative computation, we specify that the product of the labels is in the order of the path, from source to sink. The ABP itself computes the sum of all such polynomials.

The restricted model we consider is defined as follows.

Definition (Read-Once Oblivious ABP, see also Definition 4.4.1). A read-once oblivious ABP (roABP) in the variable set \(x = \{x_1, \ldots, x_n\}\) is an ABP of depth \(n\), such that each edge between layer \(i - 1\) and layer \(i\) is labeled with a univariate polynomial in \(x_i\) of degree < \(d\).

Note that in read-once oblivious ABPs we allow edges to be labeled with arbitrary univariate polynomials (with a bound of \(d\) on the degree) and not just with linear forms. Further, note that the above definition has a known variable order \(x_1 < \cdots < x_n\).

Nisan [Nis91], along with defining ABPs, gave exponential lower bounds in the non-commutative ABP model for the determinant. We adapt his methods in this chapter to give such lower bounds for roABPs.

Theorem (Nisan [Nis91], see Corollary 4.5.10). There exists an explicit \(n\)-variable polynomial which requires \(\exp(n)\)-width to be computed by a roABP.
Finally, we conclude with a notion of Saptharishi [Sap12], called *evaluation dimension*. We show that this notion is equivalent to being computed by a small roABP in any variable order (not just $x_1 < \cdots < x_n$).

**Chapter 5 — Rank Condensers:** This chapter defines the main pseudorandom object that will power the hitting set results of Chapter 6, and this object is called a *rank condenser*. It is a particular form of dimension reduction that maps large dimensional spaces down to medium-sized spaces, all the while preserving the dimension of low-dimensional subspaces. We give here one definition of this object.

**Definition (Definition 5.1.3).** A set of matrices $E \subseteq \mathbb{F}^{m \times n}$ is said to be a **rank condenser for rank** $r$ if for every matrix $M \in \mathbb{F}^{n \times r}$ of rank $\leq r$, there is some $E \in E$ such that $\text{rank}(E \cdot M) = \text{rank}(M)$. ♦

The main way this object will be used will be as a form of *oblivious Gaussian elimination*. That is, given a set of vectors $v_1, \ldots, v_n \in \mathbb{F}^r$ where $n \gg r$, we want to find a set of vectors $\pi_1, \ldots, \pi_r \in \mathbb{F}^r$ such that $\text{span}\{v_i\} = \text{span}\{\pi_j\}$. Note that this is trivially done using Gaussian elimination, but rank condensers (with suitable parameters) allow this to be done in a manner *oblivious* to the input. This will be particularly important in Chapter 6, as we will take the algorithm of white-box PIT algorithm of Raz-Shpilka [RS05] and make it black-box. Noting that black-box algorithms are oblivious to their input, we obtain the result by replacing Gaussian elimination with rank condensers and use degree reduction step we call *reinterpolation*.

Along with defining the above notion, as well as some variants, this chapter studies the *seed-length* of the above rank condensers. That is, we seek to use rank condensers to *derandomize* PIT, and thus clearly need rank condensers that are *randomness efficient*. To begin, we show what the non-constructive probabilistic method yields.

**Theorem (Lemma 5.3.5).** Let $\mathbb{F}_q$ be a finite field with $q \geq \text{poly}(n)$. Then a random set $E \subseteq \mathbb{F}^{r \times n}$ with $|E| = r(n-r) + 1$ is a rank condenser $E \subseteq \mathbb{F}^{r \times n}$ for rank $r$.

Note that the naive rank condenser (choosing all matrices in $\mathbb{F}^{r \times n}$) would have $|E| = q^{nr}$, so the above is a substantial improvement.

Of course, for algorithmic purposes the above is not helpful. Thus, we develop an *explicit* rank condenser with the same parameters.

**Theorem (Theorem 5.4.3).** Let $\mathbb{F}_q$ be a finite field with $q \geq \text{poly}(n)$. Then there is an explicit set $E \subseteq \mathbb{F}^{r \times n}$ with $|E| = r(n-r) + 1$ that is a rank condenser $E \subseteq \mathbb{F}^{r \times n}$ for rank $r$.

**Chapter 6 — Polynomial Identity Testing of roABPs:** In this chapter we give PIT algorithms for roABPs. These algorithms will iteratively reduce the number of variables in the roABPs until one variable remains, so that brute-force methods apply. To do so, we need to understand what this reduction *preserves* throughout this process, so that the property of non-zeroness is maintained. In particular, the notion we use here is that of *coefficient span*, which we informally define here.
Definition (Definition 6.2.2). Given $f_1, \ldots, f_k \in \mathbb{F}[\overline{x}]$, their \textit{coefficient span} is a subspace of $\mathbb{F}^k$ defined by

$$\text{span}\{(\text{Coeff}_{\overline{x}i}f_1, \ldots, \text{Coeff}_{\overline{x}i}f_k)\}_{i=1}^{k}.$$

That is, we consider the polynomials $(f_i)_i$ as a single \textit{vector-valued polynomial}, and take the span of the coefficients of this vector-valued polynomial.

Once we have this definition in hand, we can show how this notion composes nicely under variable disjoint products. In particular, this leads naturally to the white-box PIT algorithm of Raz-Shpilka [RS05].

Theorem (Raz-Shpilka [RS05], Theorem 6.3.7). There is an efficient deterministic parallel algorithm for white-box PIT of $n$-variate, width-$w$ roABPs of individual degree $< d$, in the unit cost model over $\mathbb{F}$.

We then discuss ideas for how to translate the above result into a black-box PIT algorithm, and observe that the Gaussian elimination in the above algorithm can be replaced with any rank condenser that is \textit{compatible} with polynomial evaluation, as the above rank condenser happens to be. This leads into the main result of the thesis.

Theorem (Corollary 6.5.17.(1)). Let $C$ be the set of $n$-variate polynomials computable by width $w$, depth $n$, individual degree $< d$ roABPs. If $|\mathbb{F}| \geq \text{poly}(n, w, d)$, then $C$ has a $\text{poly}(n, w, d)$-explicit hitting set, of size $\leq \text{poly}(n, w, d)^{O(\lg n)}$.

The above construction can be seen (see Remark 6.5.4) to be analogous to Nisan’s [Nis92] pseudorandom number generator (PRG) for RL, as his result is really a PRG for read-once oblivious \textit{boolean} branching programs (roBPs). Unfortunately there are few results improving on Nisan’s [Nis92] in large generality, and the resistance of the subclass of \textit{constant width} roBPs to improvement has been particularly noticeable. Somewhat surprisingly, by tuning the parameters of our construction (as prompted by Wigderson [Wig12]) we can achieve an improvement for constant-width roABPs.

Theorem (Corollary 6.5.17.(2)). Let $C$ be the set of $n$-variate polynomials computable by width $O(1)$, depth $n$, individual degree $< d$ roABPs. If $|\mathbb{F}| \geq \text{poly}(n, d)$, then $C$ has a $\text{poly}(n, d)$-explicit hitting set, of size $\leq \text{poly}(n, d)^{O(\lg n/\lg \lg n)}$.

Chapter 7 — Non-Commutative ABPs: This chapter considers general ABPs that compute over non-commutative variables $x_i$, and thus work in the ring $\mathbb{F}(\overline{x})$. Nisan [Nis91] gave exponential lower bounds for this model, as mentioned above, and the white-box PIT algorithm of Raz-Shpilka [RS05] was originally phrased as working for this model (as opposed to roABPs). This chapter extends the above roABP hitting set to non-commutative ABPs, where the hitting set is now defined \textit{over matrices}. This is achieved by showing that our matrix construction naturally “homogenizes” the non-commutative ABP, and that each homogeneous part is naturally computed by an roABP, so that the hitting set can be applied to these roABPs.
Corollary (Corollary 7.2.7). Let $\mathcal{C}$ be the set of $n$-variate non-commutative polynomials computable by width $r$, depth $D$ ABPs. If $|\mathbb{F}| \geq \text{poly}(n, r, D)$, then $\mathcal{C}$ has a $\text{poly}(D, n, r)$-explicit hitting set over $(D + 1) \times (D + 1)$ matrices, of size $\leq \text{poly}(n, r, D)^{O(\lg D)}$.

The above thus gives the desired black-box analogue of the Raz-Shpilka [RS05] algorithm that, prior to this work, was unknown.

Chapter 8 — Diagonal Circuits: This chapter studies a different class of circuits known as diagonal circuits, first studied by Saxena [Sax08], who extended the algorithm of Raz-Shpilka [RS05] to get white-box PIT for this class. We give an informal definition of a subclass of this class.

Definition (Σ ∧ Σ Circuits, see Definition 8.3.1). A $\Sigma$ ∧ $\Sigma$ circuit in the variables $x_1, \ldots, x_n$ of top fan-in $s$ and degree $d$ is a polynomial computed as

$$f(\overline{x}) = \sum_{i \in [s]} (\alpha_{i,0} + \alpha_{i,1}x_1 + \cdots + \alpha_{i,n}x_n)^{d_i},$$

where $d_i \leq d$, and the $\alpha_{i,j}$ are constants.

In this chapter (specifically, in Section 8.6), we show how to strengthen Saxena’s [Sax08] reduction, and give a reduction from diagonal circuits directly to roABPs, so that we can then apply the above roABP result to give hitting sets for this class.

Theorem (Corollary 8.6.10). Let $\mathcal{C}$ be the set of polynomials $f(\overline{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ computable by a size-$s$ diagonal circuit. If $|\mathbb{F}| > \text{poly}(n, s)$, then $\mathcal{C}$ has a $\text{poly}(n, s)$-explicit hitting set $\mathcal{H} \subseteq \mathbb{F}^n$ of size $\text{poly}(n, s)^{O(\lg n)}$.

2.1.2 Forbes-Shpilka [FS13a]

This work contributes to the first half of Chapter 8, and to the entirety of Chapter 9.

Chapter 8 — Diagonal Circuits: While the above quasipolynomial-size hitting sets for diagonal circuits suffice, in this chapter we show how the partial derivative method of Nisan and Wigderson [NW96] yields a particularly simple alternate proof of the above claim, at least for $\Sigma$ ∧ $\Sigma$ circuits.

Chapter 9 — Explicit Noether Normalization for Simultaneous Conjugation: This comprises the bulk of Forbes-Shpilka [FS13a], and rather than developing PIT algorithms, this chapter applies them. In particular, we consider a certain computational question known as Explicit Noether Normalization, which is a problem in computational algebraic geometry which is easily solved by randomness. This problem is about developing a form of explicit dimension reduction that preserves the structure of low-dimensional varieties, and is particularly interesting when these varieties are explicit. Mulmuley [Mul12a] observed that this question is equivalent to a (strong) form of PIT, and raised the challenge of derandomizing Noether Normalization. He
even observed that some very simple explicit varieties lack derandomized Noether Normalization.

In this chapter, we derandomize Noether Normalization for the main example of a simple variety that Mulmuley [Mul12a] noted had no known derandomization. This is done by improving Mulmuley’s [Mul12a] reduction from Noether Normalization to PIT, and showing that in this case, black-box PIT of roABPs suffices, so that the above hitting set can be applied. As even stating this result is beyond the terminology of this introduction, we defer the statement of this result (even informally) to Chapter 9.
Chapter 3
Polynomial Identity Testing (PIT)

Abstract: In this chapter we give basic background on the polynomial identity testing (PIT) problem, as well as solution concepts (see the survey by Shpilka-Yehudayoff [SY10] for more). We define algebraic circuits and quote basic facts about them. We then give a formal definition of PIT, and explain why the problem should be defined in this manner. We then give the basic solution concepts to PIT: white-box and black-box PIT algorithms, where the latter (when deterministic) is shown to be equivalent to a hitting set. We give basic constructions of hitting sets, and then relate them to the stronger notion of an interpolation set. Finally, we give the notion of a generator from Shpilka-Volkovich [SV09], show its equivalence with hitting sets, and argue why they are more natural.

3.1 Algebraic Computation

We begin this chapter by giving basic definitions in algebraic computation. We begin with the definition of the underlying computation graph.

Definition 3.1.1. An ordered directed acyclic (multi-)graph (ordered DAG) is a directed acyclic multi-graph \( G = (V, E) \) such that each node \( v \in V \) imposes some total order on its children.

We use multigraph here to mean that self-loops and parallel edges are allowed. Of course, in a directed acyclic multigraph, there can be no self-loops. We allow multi-graphs so that the squaring operation can be most naturally represented as a multiplication gate with a repeated child.

We can then allow this graph to perform computation if we add computational primitives, known as gates.

Definition 3.1.2. Let \( S \) be a set. Let \( \mathcal{G} \) be a set of functions of varying arity, so \( \mathcal{G} = \{ g_i : S^{k_i} \to S, k_i \geq 0 \} \).

A circuit \( C \) with gates \( \mathcal{G} \) computing in \( S \) is an ordered directed acyclic multi-graph \( G = (V, E) \) with a labelling \( V \to \mathcal{G} \) such that a node with \( k \) children is labelled with a function \( g \in \mathcal{G} \) with arity \( k \).

The size of \( C \) is the number of edges \( |E| \). The depth of \( C \) is the length of the
longest path in the graph $G$. \hfill \Box

A standard inductive claim shows that such circuits induce a well-defined computation.

**Fact 3.1.3** (Circuits Compute in a Well-Defined Manner). Let $S$ be a set. Let $\mathcal{G}$ be a set of functions of varying arity, so $\mathcal{G} = \{g_i : S^{k_i} \rightarrow S, k_i \geq 0\}$. Let $C$ be a circuit with gates $\mathcal{G}$ computing in $S$. Each node in $C$ has a well-defined value in $S$.

Note that the depth of a circuit is a well-defined integer, as no path can be longer than $|E|$, as otherwise this would violate the acyclicity of the circuit.

Note that we allow arity 0 gates, which label the source-nodes of $G$ with elements of $S$.

When the gates of the circuit are algebraically natural (such as multiplication and addition), we then obtain an algebraic circuit.

**Definition 3.1.4.** Let $F$ be a field and $n \geq 1$. An algebraic circuit (over $F[x_1, \ldots, x_n]$) is a circuit computing in $F[x]$ with gates $\mathcal{G} = \{\times_k\}_{k \geq 2} \cup \{+k\}_{k \geq 2} \cup F \cup \{x_i\}_{i \in [n]}$, where

- $\times_k : F[x]^k \rightarrow F[x]$ is the $k$-ary multiplication function $(\alpha_1, \ldots, \alpha_k) \mapsto \alpha_1 \cdots \alpha_k$
- $+k : F[x]^k \rightarrow F[x]$ is the $k$-ary addition function $(\alpha_1, \ldots, \alpha_k) \mapsto \alpha_1 + \cdots + \alpha_k$
- $F$ and $x_i$ are 0-ary functions yielding the respective element of $F[x]$

Unless otherwise specified, a algebraic circuit has a unique node with out-degree 0, and the value of this node is the output of the algebraic circuit.

An algebraic circuit is $t(n)$-explicit if it can be computed in $t(n)$ steps in the unit-cost model over $F$. \hfill \Box

Note that as $F[x]$ is a commutative ring, the underlying orders imposed by the gates on their children does not alter the computation of the circuit. Also, note that $t(n)$-explicit algebraic circuits necessarily have size at most $t(n)$.

We now quote the fact stating that algebraic circuits can always be modified so their fan-in is 2.

**Fact 3.1.5** (Converting Algebraic Circuits to Fan-in 2, Efficiently). Suppose $f \in F[x]$ is computable by a size $\leq s$ algebraic circuit. Then $f$ is computable by a size $\leq \text{poly}(s)$ algebraic circuit where each node has a fan-in of at most 2.

Using this above fact, we can then count the number of small algebraic circuits, over a finite field.

**Lemma 3.1.6.** Let $F$ be a finite field and $n, s \geq 1$. There are $\leq (8n|F|s^2)^s$ polynomials in $F[x_1, \ldots, x_n]$ computable by (single-output) algebraic circuits of size $\leq s$ and fan-in $\leq 2$.

**Proof:** Consider a (single-output) algebraic circuit of size $\leq s$. It has $\leq s$ nodes, each of which can be a $\times$-gate, a $+$-gate, a constant in $F$, or a variable $x_i$. This constitutes $\leq |F| + n + 2$ choices. As each gate has at most 2 children, there are at most $s^2$
choices of these children. The choice for each gate of its type, and its children, uniquely determines the circuit. Thus, as each gate is specified by \( \leq |\mathbb{F}| + n + s^2 \) choices, and there are \( s \) gates, this yields \( \leq (|\mathbb{F}| + n + s^2 + 2)^s \leq (8|\mathbb{F}|s^2)^s \) such circuits (as \( |\mathbb{F}|, n, s \geq 1 \)).

Finally, we quote here that standard fact that algebraic circuits can be computed efficiently, as this will be needed later.

**Fact 3.1.7** (Algebraic Circuits can be Evaluated Efficiently). Suppose \( f \in \mathbb{F}[\mathbf{x}] \) is computable by a size \( \leq s \) algebraic circuit \( C \). Given \( C \), for any \( \mathbf{\alpha} \in \mathbb{F}^n \), \( f(\mathbf{\alpha}) \) can be computed in time \( \text{poly}(s, n) \) in the unit-cost model over \( \mathbb{F} \). Further, the algebraic circuit can be computed in \( \text{poly}(d, \text{polylog}(s, n)) \) parallel time.

### 3.2 Polynomial Identity Testing (PIT)

We now discuss the *polynomial identity testing (PIT) problem*, a fundamental problem in computational algebra.

**Definition 3.2.1.** Let \( \mathcal{C} \) be a non-trivial\(^1\) subset of algebraic circuits of size \( \leq s \) computing polynomials in \( \mathbb{F}[x_1, \ldots, x_n] \) of degree \( < d \). The **polynomial identity testing (PIT) problem** is to, given a circuit \( C \in \mathcal{C} \) computing a polynomial \( f_C \), decide whether \( f_C = 0 \). The size of an input circuit \( C \) will be considered to be \( \text{snd} \), so an algorithm is considered efficient if it runs in \( \text{poly}(\text{snd}) \) steps.

An algorithm solving PIT for \( \mathcal{C} \), in the unit-cost model over \( \mathbb{F} \), is **black-box** if it solves PIT only using \( C \) to evaluate \( f_C \) on points in \( \mathbb{F}^n \). Otherwise it is **white-box**. \( \Box \)

As an example to contrast white-box versus black-box algorithms, consider the PIT problem over univariate polynomials of degree \( < d \) computable by size \( \leq s \) circuits (and think of \( s \approx d \), so that this problem is somewhat trivial). PIT for this class of circuits can be solved in \( \text{poly}(sd) \) time in the white-box model simply by using the circuit to compute the complete expansion of the polynomial. As this algorithm computes polynomials evaluated not just by the output node of the circuit but also by all internal nodes, it is clearly white-box. In contrast, a black-box algorithm would use interpolation (see Proposition B.2.5) to reconstruct the output polynomial from any \( d \) evaluations.

**Remark 3.2.2** (On the degree of polynomials in PIT). Note that when the field \( \mathbb{F} \) is \( \mathbb{Q} \) or some large finite field, PIT can be defined where the size of a circuit is not “\( \text{snd} \)” but rather “\( \text{sn log } d \)” recognizing that size \( s \) circuits can efficiently compute polynomials of degree \( \exp(s) \).

We will not define the size of a circuit this way for several reasons, which mostly are because such a regime is “much harder”:

- Many “natural” instances of PIT only consider polynomials of low-degree, such as (projections of) the determinant. Another example is the application of PIT to explicit Noether Normalization constructions, as discussed in Chapter 9.

\(^1\)We say the subset \( \mathcal{C} \) is non-trivial if there are \( \text{poly}(s, n, d) \)-explicit circuits \( C_0, C_1 \in \mathcal{C} \) so that \( C_0 \) is a circuit computing the zero polynomial and \( C_1 \) is a circuit computing some non-zero polynomial.
Many “natural” restricted models of computation (such as formulas [circuits where each node has out-degree 1], and algebraic branching programs [see Chapter 4]) are syntactically guaranteed to compute polynomials which have degree bounded by the size of the circuit.

A size $s$ circuit can certainly compute the polynomial $x^{2^s}$ via repeated squaring. While evaluating this circuit can be done in $\text{poly}(s)$ steps in the unit-cost model, this is not true when considering the underlying complexity of these field operations. That is, computing $\alpha^{2^s}$ can be done in $\text{poly}(s,q)$ steps in a finite field of size $q$, but computing $2^{2^s}$ in $\mathbb{Q}$ would take $2^s$ steps simply because the output requires that many bits.

Thus, to consider circuits with such large degree one needs to pay attention to the underlying bit complexity of the operations, which would break our abstraction of treating operations in $\mathbb{F}$ as unit-cost.

One can deal with computations such as $x^{2^s}$ over $\mathbb{Q}$ by viewing them modulo a random prime $p$, taking $p \approx 2^s$. However, doing so leaves the purview of abstract algebra and enters number theory, which is usually more difficult.

When considering $\exp(s)$-degree polynomials, even univariate problems become difficult. That is, for a degree-$(< d)$ polynomial, one can use that it has at most $< d$ roots (Lemma B.2.10) so that any $d$ points must contain a non-root. While this is cheap for small $d$, when $d = \exp(s)$ this is expensive. As many known PIT algorithms proceed by reducing the multivariate case to the univariate case, this paradigm would be unavailable when $d = \exp(s)$, making the problem much more difficult.

The univariate PIT problem when $d = \exp(s)$ can also be seen to be formally hard, in that it captures the low-degree multivariate PIT problem entirely. That is, given an algebraic circuit of size $s$ computing a multivariate polynomial $f(x_1, \ldots, x_n)$ of total degree $< d$, one can use the Kronecker substitution to see that the polynomial $\hat{f}(x) := f(x, x^d, \ldots, x^{d^{n-1}})$ is nonzero iff $f(x)$ is non-zero. Further, $\hat{f}(x)$ can be computed in size $\text{poly}(s, n, \lg d)$. Thus, solving the PIT problem for this univariate polynomial implies solving PIT for the original multivariate polynomial, so that univariate PIT for $\exp(s)$-degree polynomials entirely captures the complexity of multivariate PIT for $\text{poly}(s)$-degree polynomials. As even tackling the low-degree PIT problem seems difficult, it seems worth restricting attention to this easier (but still difficult) problem.

Algorithms can also be classified as randomized or deterministic, following the usual notions of complexity theory. In particular, there is an efficient randomized black-box PIT algorithm, as this follows from the Schwartz-Zippel Lemma (Lemma B.2.11).

Lemma 3.2.3 (randomized black-box PIT). Let $\mathbb{F}$ be a field of size $\geq 2d$. Let $\mathcal{C}$ be the set of algebraic circuits of size $\leq s$ computing polynomials in $\mathbb{F}[x_1, \ldots, x_n]$ of degree $< d$. There is a randomized black-box PIT algorithm for $\mathcal{C}$ running in $\text{poly}(s, n, d)$ time, so that for $C \in \mathcal{C}$, if $f_C = 0$ then the algorithm always returns “$f_C = 0$” and if $f_C \neq 0$ then the algorithm returns “$f_C \neq 0$” with probability $\geq 1/2$. 

34
Proof: Let $S \subseteq \mathbb{F}$ be an explicit subset of size $2d$, computable in $\text{poly}(d)$ steps. Given a circuit $C \in \mathcal{C}$ The algorithm then picks a random point $\alpha \in S^n$ and evaluates $f_C(\alpha)$. It returns “$f_C = 0$” iff $f_C(\alpha) = 0$.

Correctness: It is clear that if $f_C = 0$ then “$f_C = 0$” is always returned. Suppose then $f_C \neq 0$. Then by the Schwartz-Zippel Lemma, $\Pr_{\alpha \sim S^n}[f_C(\alpha) = 0] < 1/2$, so that “$f_C \neq 0$” is returned with probability $\geq 1/2$.

Efficiency: As discussed in Appendix A, drawing the random element $\alpha$ can be done in $\text{poly}(n, d)$ steps. Evaluating $f_C(\alpha)$ can be done in $\text{poly}(s)$ steps, by Fact 3.1.7, and zero testing of $f_C(\alpha)$ can be done at unit cost.

\begin{remark}[On the size of the field in PIT] Note that in the above we restrict our attention to the case when $\mathbb{F}$ is large. When $\mathbb{F}$ is small, the Schwartz-Zippel Lemma no longer yields anything meaningful. Indeed, if we wish to solve black-box PIT over $\mathbb{F}_2$ (for $d = \text{poly}(n)$) only using evaluation access over $\mathbb{F}_2$ then this is equivalent to the Satisfiability problem, which is $\text{NP}$-complete. \hfill \qed

To use the above result when the field $\mathbb{F}_q$ is too small, one can embed $\mathbb{F}_q$ into an extension field $\mathbb{F}_{q^t}$, noting that a polynomial $f \in \mathbb{F}_q[x]$ is non-zero iff it is non-zero as a polynomial in $\mathbb{F}_{q^t}[x]$. While there are many non-trivial results about finding exponentially large extension fields (for example, see Adleman and Lenstra [AL86]), in PIT one typically only needs a field that is polynomially large. In this regime, finding such an extension is trivial, as seen by the following lemma.

**Lemma 3.2.5.** Let $\mathbb{F}_q$ be a field with $q < d$. Then an explicit representation of a field $\mathbb{F}_{q^t}$ (as $\mathbb{F}_q[x]/(f)$) extending $\mathbb{F}_q$ with $d^2 > q^t \geq d$ can be found deterministically in $\text{poly}(d)$ steps.

Proof: Let $t = \lceil \log_d q \rceil$ so that $d \leq q^t \leq d \cdot q < d^2$ and thus $t \leq \lg d + 1$ (as $q \geq 2$). By standard facts about finite fields (for example, see Shoup [Sho09]), there exists a finite field $\mathbb{F}_{q^t}$ with $|\mathbb{F}_{q^t}| = q^t$ and $\mathbb{F}_{q^t}$ can be represented by $\mathbb{F}_q[x]/(f)$ for some monic irreducible polynomial $f \in \mathbb{F}_q[x]$ of degree $t$. As a polynomial $f \in \mathbb{F}_q[x]$ of degree $t$ can be tested for irreducibility in $\text{poly}(q, t) \leq \text{poly}(d)$ steps (see Shoup [Sho09]), we can check all $q^t$ monic polynomials in $\mathbb{F}_q[x]$ of degree $t$ and find some irreducible polynomial $f$. As $q^t < d^2$, enumerating all $f$ takes $\text{poly}(d)$ steps, and thus yields the explicit representation of $\mathbb{F}_{q^t} \equiv \mathbb{F}_q[x]/(f)$. \hfill \qed

Thus, the above lemma reduces PIT over small fields to PIT over larger fields. In the white-box case, this reduction “computationally simulates” the large field using the isomorphism $\mathbb{F}_{q^t} \equiv \mathbb{F}_q[x]/(f)$ for some polynomial $f$. In the black-box case, this reduction means allowing evaluation access not just over the field $\mathbb{F}_q$ but also the extension field $\mathbb{F}_{q^t}$. As these reductions are straightforward and considered standard, we instead focus on PIT over large fields, and quantify “large” in each case. However, to make this explicit, we record that PIT admits a (white box) randomized algorithm over any field.

**Lemma 3.2.6** (randomized white-box PIT, any field). Let $\mathbb{F}$ be any field. Let $\mathcal{C}$ be the set of algebraic circuits of size $\leq s$ computing polynomials in $\mathbb{F}[x_1, \ldots, x_n]$ of degree
There is a randomized white-box PIT algorithm for $C$ running in $\text{poly}(s,n,d)$ time, so that for $C \in C$, if $f_C = 0$ then the algorithm always returns “$f_C = 0$” and if $f_C \neq 0$ then the algorithm returns “$f_C \neq 0$” with probability $\geq 1/2$.

Proof: $|F| \geq 2d$: We can use Lemma 3.2.3.

$|F| < 2d$: Using Lemma 3.2.5 we can find a field extension $K$ of $F$ such that $|K| \geq 2d$, in $\text{poly}(d)$ steps. As $K$ is explicitly represented as $F[x]/\langle f \rangle$ for some degree $O(\lg d)$ polynomial, it follows that operations over $K$ can be simulated in $\text{poly}(d)$ operations over $F$. Thus, given a circuit over $F$ we can treat it as a circuit over $K$ and run Lemma 3.2.3, using that evaluating this circuit over $K$ can be done in $\text{poly}(s,n,d)$ steps over $F$. This gives the runtime analysis, and the correctness is clear. □

We now discuss deterministic black-box PIT algorithms. The fundamental object in this study is that of a hitting set, which we now define.

**Definition 3.2.7.** Let $C \subseteq F[x_1, \ldots, x_n]$ be a set of polynomials. A multi-set $H \subseteq F^n$ is a hitting set for $C$ if for all $f \in C$,

$$f \equiv 0 \text{ iff } f|_H \equiv 0.$$  

That is, $f(\bar{x}) = 0$ in $F[\bar{x}]$ iff $f(\bar{x}) = 0$ for all $\bar{x} \in H$.

The hitting set $H$ is $t(n)$-explicit if there is an algorithm such that given an index into $H$, the corresponding element of $H$ can be computed in $t(n)$-time in the unit-cost computation model over $F$.

**Remark 3.2.8.** Note that we formally define a hitting set $H$ as a multi-set, even though duplicate points in $H$ can be removed without harming the property that $f \equiv 0$ iff $f|_H \equiv 0$. We allow multi-sets because removal of duplicates would decrease the explicitness of $H$ from $t(n)$ to $\text{poly}(t(n), n, |H|)$, which can be much worse if $t(n) \leq \text{poly}(n)$ and $H$ is super-polynomial. This phenomenon occurs in the hitting set for diagonal circuits (Theorem 8.5.6), as well as any hitting set arising from a generator (Lemma 3.2.21) with non-constant seed-length. □

It follows immediately that explicit hitting sets yield deterministic black-box PIT algorithms.

**Lemma 3.2.9** (hitting set $\implies$ black-box PIT). Let $C$ be a subset of algebraic circuits of size $\leq s$ computing polynomials in $F[x_1, \ldots, x_n]$ of degree $< d$. If there is a $t(s,n,d)$-explicit hitting set $H$ for $C$ then there is a deterministic $\text{poly}(|H|, t(s,n,d), s, n, d)$-time black-box PIT algorithm for $C$.

Proof: Given a circuit $C \in C$ the algorithm computes the hitting set $H$, then evaluates $f_C(\bar{x})$ for all $\bar{x} \in H$. It returns “$f_C = 0$” iff $f_C|_H = 0$.

**Correctness:** This follows from the definition of a hitting set.

**Efficiency:** Computing the hitting set takes $\text{poly}(|H|, t(s,n,d))$-time, and evaluating $f_C|_H$ takes $\text{poly}(|H|, s, n)$-time as evaluating each point can be done quickly (Fact 3.1.7). □
Note that the above algorithm is non-adaptive, in that the evaluations of $f_C$ do not depend on $C$. For some computational problems, adaptivity can be proven stronger than non-adaptivity. However, this not the case for black-box PIT, as we now show.

**Lemma 3.2.10** (black-box PIT $\implies$ hitting set). Let $C$ be a non-trivial subset of algebraic circuits of size $\leq s$ computing polynomials in $\mathbb{F}[x_1,\ldots,x_n]$ of degree $< d$. If there is a $t(s,n,d)$-time deterministic black-box PIT algorithm $A$ for $C$ then there is $\text{poly}(t(s,n,d), s, n, d)$-explicit hitting set $\mathcal{H}$ for $C$ with $|\mathcal{H}| \leq t(s,n,d)$.

**Proof:** Let $C_0 \in C$ be a circuit computing the zero polynomial, which exists as $C$ is non-trivial. Run the algorithm $A$ on $C_0$. Let $(\alpha_i)_{i \leq t(s,n,d)}$ be the sequence of points in $\mathbb{F}^n$ that $A$ evaluates $f_{C_0}$ on. Define $\mathcal{H} := \{ \alpha_i \}_i$.

- **Correctness:** Clearly if $C \in C$ has $f_C = 0$ then $f_C |_{\mathcal{H}} = 0$. Thus, suppose $f_C \neq 0$. Run $A$ on $C$, producing a sequence of evaluation points $\beta_i \in \mathbb{F}^n$. Let $i_0$ be the first $\beta_i$ so that $f_C(\beta_i) \neq 0$. We now have the following key claim.

**Claim 3.2.11.** For $i \leq i_0$, $\alpha_i = \beta_i$.

**Proof:** By induction on $i$. The base case is $i = 0$, which is vacuous.

- $i \geq 1$: As for all $j < i$, we have $\alpha_j = \beta_j$ and $j < i \leq i_0$ it follows that $f_C(\alpha_j) = f_C(\beta_j) = 0 = f_{C_0}(\beta_j)$. Thus, as $A$ is only allowed access to $C$ through an evaluation oracle to $f_C$, it follows that $C$ and $C_0$ induce the same state of $A$ (as $A$ is deterministic) after the first $i - 1$ queries. Thus, the $i$-th query that $A$ produces will be the same for $C$ and $C_0$, so $\alpha_i = \beta_i$. \qed

It follows then that $f_C(\alpha_{i_0}) = f_C(\beta_{i_0}) \neq 0$, so that $f_C |_{\mathcal{H}} \neq 0$, as desired.

- **size:** As $A$ runs for at most $t(s,n,d)$ steps it follows that $|\mathcal{H}|$ is as desired.

- **Explicitness:** Producing $C_0$ can be done in $\text{poly}(s,n,d)$-steps (by definition of “non-trivial”), and running $A$ takes $t(s,n,d)$-steps, so producing $\mathcal{H}$ can be done in the requisite time bound. \qed

Thus, the above shows that to develop deterministic black-box PIT algorithms, we can equivalently construct explicit hitting sets, and we will thus exclusively take this view. We now establish some basic hitting sets. We first show that explicit hitting sets exist, if one allows a large size.

**Lemma 3.2.12** (large explicit hitting sets). Let $\mathbb{F}$ be a field with $|\mathbb{F}| \geq d$. Let $C \subseteq \mathbb{F}[x_1,\ldots,x_n]$ be a set of polynomials of degree $< d$. Then there is $\text{poly}(n,d)$-explicit hitting set for $C$ of size $d^n$.

**Proof:** Let $S \subseteq \mathbb{F}$ be an explicit subset of size $d$, computable in time $\text{poly}(d)$. Define the hitting set as $\mathcal{H} := S^n$.

- **Correctness:** By interpolation (Corollary B.2.9).

- **Explicitness:** Indexing into $\mathcal{H}$ takes $\text{poly}(n, \lg d)$ time, and computing the respective element of $S^n$ take $\text{poly}(n,d)$ steps, as $S$ is explicit. \qed
We now show, over finite fields, that small hitting sets exist, if one allows non-explicit sets. One can show similar statements over infinite fields, see the survey of Shpilka and Yehudayoff [SY10] for more. We first show this finite-field result for any small set of polynomials, via a standard probabilistic method argument.

**Lemma 3.2.13.** Let $\mathbb{F}$ be a finite field with $|\mathbb{F}| \geq (1 + \epsilon)d$. Let $C \subseteq \mathbb{F}[x_1, \ldots, x_n]$ be a finite set of polynomials of degree $< d$. Then there is a non-explicit hitting set for $C$ of size $\leq \lceil \log_{1+\epsilon}|C| \rceil$.

**Proof:** Fix some polynomial $f \in C$, and consider the probability a random set fails to hit $\mathcal{H}$.

$$\Pr_{\mathcal{H} \subseteq \mathbb{F}^n}[f|_{\mathcal{H}} \equiv 0] = \Pr_{\mathcal{H}}[\forall \alpha \in \mathcal{H}, f(\alpha) = 0]$$

Appealing to the Schwartz-Zippel Lemma (Lemma B.2.11),

$$\leq \left( \frac{d}{|\mathbb{F}|} \right)^{|\mathcal{H}|} \leq \left( \frac{1}{1 + \epsilon} \right)^{|\mathcal{H}|}$$

Applying a union bound, we see that

$$\Pr_{\mathcal{H}}[\forall f \in C, f|_{\mathcal{H}} \equiv 0] < |C| \left( \frac{1}{1 + \epsilon} \right)^{|\mathcal{H}|}$$

Taking $|\mathcal{H}| = \lceil \log_{1+\epsilon}|C| \rceil$,

$$\leq 1$$

Thus, the probabilistic method shows that there exists a hitting set $\mathcal{H}$ for $C$ of the desired size. \qed

We now conclude that small hitting sets exist.

**Lemma 3.2.14 (small hitting sets exist).** Let $\mathbb{F}$ be a finite field with $|\mathbb{F}| \geq d^2$. Let $C \subseteq \mathbb{F}[x_1, \ldots, x_n]$ be the set of polynomials of degree $< d$ computable by algebraic circuits of size $\leq s$ and fan-in $\leq 2$. Then there is a non-explicit hitting set for $C$ of size $\leq \lceil 2s(\log n + 2 \log s + 4) \rceil$.

**Proof:** As $|\mathbb{F}| \geq d^2$ then $d \leq \sqrt{|\mathbb{F}|}$, so $|\mathbb{F}| \geq (1 + \epsilon)d$ with $1 + \epsilon = \sqrt{|\mathbb{F}|}$. Using this value of $\epsilon$ in Lemma 3.2.13 applied to the counting bound of Lemma 3.1.6 we get a hitting set of size

$$\lceil \log_{1+\epsilon}|C| \rceil \leq \lceil \log_{\sqrt{|\mathbb{F}|}}(8n|\mathbb{F}|s^2)^s \rceil$$

$$= \lceil s(2 + \log_{\sqrt{|\mathbb{F}|}}(8ns^2)) \rceil$$

$$= \lceil s(2 + 2 \cdot \log_{|\mathbb{F}|}(8ns^2)) \rceil$$

38
As $|\mathbb{F}| \geq 2$,

\[
\leq \lceil s(2 + 2\log(8ns^2)) \rceil \\
= \lceil 2s(\log n + 2 \log s + 4) \rceil
\]

One can view this chain of results as another path to showing that PIT has a randomized black-box PIT algorithm. That is, the probabilistic method yields that a random set of size $\approx s$ will be a hitting set for size $s$ circuits with high probability. Plugging this into Lemma 3.2.9 then yields the desired (randomized) black-box PIT algorithm.

### 3.2.1 Interpolation Sets

In this section we discuss a stronger notion than a hitting set, which is called an interpolation set. While a hitting set $\mathcal{H}$ requires that any non-zero $f$ has some non-zero amongst the evaluations $f|_{\mathcal{H}}$, an interpolation set $\mathcal{I}$ requires that $f$ is determined by the evaluations $f|_{\mathcal{I}}$. We begin with the definition.

**Definition 3.2.15.** Let $\mathcal{C} \subseteq \mathbb{F}[x_1, \ldots, x_n]$ be a set of polynomials. A set $\mathcal{I} \subseteq \mathbb{F}^n$ is an interpolation set for $\mathcal{C}$ if for all $f, g \in \mathcal{C}$,

$$f|_{\mathcal{I}} \equiv g|_{\mathcal{I}} \implies f = g .$$

The interpolation set $\mathcal{I}$ is $t(n)$-explicit if there is an algorithm such that given an index into $\mathcal{I}$, the corresponding element of $\mathcal{I}$ can be computed in $t(n)$-time in the unit-cost computation model over $\mathbb{F}$.

We now show that interpolation sets for $\mathcal{C}$ can be constructed from hitting sets for a slightly stronger class. Morally, this indicates that hitting sets and interpolation sets are roughly equivalent for natural classes of computation. This result can be seen as a generalization of the coding theory result showing that unique decoding of an error correcting code can be done up to half of its minimum distance.

**Lemma 3.2.16.** Let $\mathcal{C} \subseteq \mathbb{F}[x]$ be a set of polynomials. Define $\mathcal{C}' = \{ f - g : f, g \in \mathcal{C} \}$. If $\mathcal{H}$ is a hitting set for $\mathcal{C}'$, then $\mathcal{H}$ is an interpolation set for $\mathcal{C}$.

**Proof:** Let $f, g \in \mathcal{C}$ with $f|_{\mathcal{H}} \equiv g|_{\mathcal{H}}$, so that $(f - g)|_{\mathcal{H}} \equiv 0$. As $f - g \in \mathcal{C}'$ and $\mathcal{H}$ is a hitting set for $\mathcal{C}'$ it follows that $f - g = 0$ so that $f = g$ as desired.

While in the previous section we saw (non-explicit) upper bounds for the size of hitting sets, we now show how to get lower bounds for interpolation sets.

**Lemma 3.2.17.** Let $\mathbb{F}$ be a finite field. Let $\mathcal{C} \subseteq \mathbb{F}[x]$ be a finite set of polynomials. Then any interpolation set for $\mathcal{C}$ must have size $\geq \log_{|\mathbb{F}|} |\mathcal{C}|$.

**Proof:** Let $\mathcal{I}$ be an interpolation set. Consider the map from $\mathcal{C}$ to $\mathbb{F}^\mathcal{I}$ sending $f \mapsto f|_{\mathcal{I}}$. By definition this map is injective. Thus, $|\mathbb{F}|^{|\mathcal{I}|} = |\mathbb{F}^\mathcal{I}| \geq |\mathcal{C}|$, giving the result.
Unfortunately this lemma is difficult to use in practice as it is hard to lower bound the number of distinct polynomials computable within some class. However, we will use this result in Section 8.5 at an intuitive level.

### 3.2.2 Generators

While the above shows that small explicit hitting sets give good deterministic black-box algorithms for PIT, hitting sets are conceptually deficient in two respects. The first respect is their notion of explicitness. For hitting sets of super-polynomial size, the notion of being polynomial-time explicit suggests that the hitting set itself is somehow “uniform”, in the sense that there is a single object that “explains” the super-polynomial number of points in the set in a concise way. Thus, it more natural to consider the concise object as a solution concept to PIT. Second, hitting sets lack a natural composition. That is, when constructing PIT algorithms, it can be desirable to use a recursive paradigm boosting slightly non-trivial algorithms into powerful algorithms. However, producing a non-trivial (yet super-polynomially large) hitting set qualitatively changes the problem from asking if a polynomial is non-zero, to asking if a polynomial is non-zero on a given set. It is more natural to avoid this qualitative change. Motivated by the above, we now another concept called a generator, as introduced by Shpilka and Volkovich [SV09].

**Definition 3.2.18.** Let \( C \subseteq \mathbb{F}[x_1, \ldots, x_n] \) be a set of polynomials. A polynomial \( \mathcal{G} : \mathbb{F}^s \rightarrow \mathbb{F}^n \) is a **generator for \( C \) with seed-length \( s \)** if for all \( f \in C \),

\[
    f \equiv 0 \iff f \circ \mathcal{G} \equiv 0.
\]

That is, \( f(x) = 0 \) in \( \mathbb{F}[x] \) iff \( f(\mathcal{G}(\mathbb{F})) = 0 \) in \( \mathbb{F}[\mathbb{F}] \).

The generator \( \mathcal{G} \) is \( t(n) \)-**explicit** if there is an \( n \)-output algebraic circuit computing \( \mathcal{G} \) that can be constructed \( t(n) \)-time in the unit-cost computation model over \( \mathbb{F} \).

A generator is naturally favorable to composition, as it allows one to reduce PIT on \( n \) variables, to PIT on \( s \) variables. Further, generators remain generators under multiplication.

**Lemma 3.2.19 (Shpilka-Volkovich [SV09]).** Let \( C \subseteq \mathbb{F}[x_1, \ldots, x_n] \) be a set of polynomials. Let \( \mathcal{G} : \mathbb{F}^s \rightarrow \mathbb{F}^n \) be a generator for \( C \). Then \( \mathcal{G} \) is a generator for \( C \cdot C = \{ f \cdot g \mid f, g \in C \} \).

**Proof:** Consider \( f, g \in C \). Then as \( \mathbb{F}[x] \) is an integral domain, \( fg \neq 0 \) iff \( f, g \neq 0 \). Then as \( \mathcal{G} \) is a generator for \( C \), \( f, g \neq 0 \) iff \( f \circ \mathcal{G}, g \circ \mathcal{G} \neq 0 \). Thus, using the integral domain property again, \( f \circ \mathcal{G}, g \circ \mathcal{G} \neq 0 \) iff \( (fg) \circ \mathcal{G} = (f \circ \mathcal{G})(g \circ \mathcal{G}) \neq 0 \). \( \Box \)

**Example 3.2.20.** The above lemma shows that a generator for a class of polynomials is also a generator for the multiplication of polynomials from that class. The same is not true for hitting sets, for essentially trivial reasons — hitting sets need to become larger as the degree of the polynomials get larger (as under multiplication) simply because higher degree polynomials can have more roots. For example, consider
\( \mathcal{C} := \{x, x - 1\} \). Then \( \mathcal{H} := \{0, 1\} \) is a hitting set for \( \mathcal{C} \). But \( x(x - 1) \in \mathcal{C} \cdot \mathcal{C} \) has \( x(x - 1)|_{\mathcal{H}} = 0 \). By moving to generators this problem can be circumvented. \( \diamond \)

Another interesting feature of a generator is that it can concisely represent a hitting set of super-polynomial size, as we show in the following lemmas relating generators to hitting sets, as done in Shpilka-Volkovich [SV09]. The first lemma converts a generator to a hitting set, by evaluating the generator over a sufficiently large cube of inputs.

**Lemma 3.2.21** (Shpilka-Volkovich [SV09], generators \( \implies \) hitting sets). Let \( \mathbb{F} \) be a field with \( |\mathbb{F}| \geq dD \). Let \( \mathcal{C} \subseteq \mathbb{F}[x_1, \ldots, x_n] \) be a set of polynomials of degree \( < d \), and \( \mathcal{G} : \mathbb{F}^s \to \mathbb{F}^n \) a generator for \( \mathcal{C} \) that is \( t(n, d) \)-explicit and of individual degree \( < D \). Then there is a hitting set for \( \mathcal{C} \) of size \( (dD)^s \) that is \( \text{poly}(t(n, d), s, dD) \)-explicit.

**Proof:** Let \( S \subseteq \mathbb{F} \) be an explicit subset of size \( dD \), which is computable in time \( \text{poly}(dD) \) in the unit-cost model. Let \( \mathcal{H} := \mathcal{G}(S^s) \) be the desired hitting set. We now show the desired properties.

- **Correctness:** By the property of the generator, \( f = 0 \) iff \( f \circ \mathcal{G} = 0 \). Thus, the brute-force hitting set of **Lemma 3.2.12** provides (via interpolation) a hitting set \( \mathcal{H}' = S^s \) for \( f \circ \mathcal{G} \) of size \( (dD)^s \), so that \( f \circ \mathcal{G} = 0 \) iff \( (f \circ \mathcal{G})|_{S^s} = 0 \), which is equivalent to saying that \( f|_{\mathcal{G}(S^s)} = 0 \).

- **Explicitness:** Indexing into \( \mathcal{H} \) can be done using \( \lceil s \log(dD) \rceil \) bits, and from this index in \( \mathcal{H} \) we can produce the corresponding index in \( S^s \) in \( \text{poly}(s, dD) \) steps. As \( \mathcal{G} \) is \( t(n) \)-explicit, this means that in time \( t(n) \) the generator \( \mathcal{G} \) can be constructed and that \( \mathcal{G} \) is computed by a size \( t(n) \) algebraic circuit. It follows from **Fact 3.1.7** that for any \( \bar{x} \in S^s \), \( \mathcal{G}(|\bar{x}|) \) can be computed in \( \text{poly}(t(n)) \) time, producing the desired element of \( \mathcal{H} \). \( \square \)

The second lemma converts a hitting set to a generator by interpolating a polynomial through this hitting set.

**Lemma 3.2.22** (Shpilka-Volkovich [SV09], hitting sets \( \implies \) generators). Let \( \mathbb{F} \) be a field with \( |\mathbb{F}| \geq d \). Let \( \mathcal{C} \subseteq \mathbb{F}[x_1, \ldots, x_n] \) be a set of polynomials of degree \( < d \) where \( \mathcal{H} \) is a \( t(n, d) \)-explicit hitting set for \( \mathcal{C} \). Then there is a \( \text{poly}(t(n, d), |\mathcal{H}|, n, d) \)-explicit generator \( \mathcal{G} : \mathbb{F}^{|\log_d(|\mathcal{H}|)} \to \mathbb{F}^n \) for \( \mathcal{C} \), with individual degree \( < d \).

**Proof:** Let \( S \subseteq \mathbb{F} \) be an explicit set of size \( d \), computable in time \( \text{poly}(d) \). Let \( m := \lceil \log_d(|\mathcal{H}|) \rceil \) so that \( |\mathcal{H}| \leq |S^m| < d|\mathcal{H}| \). Let \( \varphi : S^m \to \mathcal{H} \subseteq \mathbb{F}^n \) be any explicit surjection (thus computable in time \( \text{poly}(d, |\mathcal{H}|) \)). The generator \( \mathcal{G} \) is defined by

\[
\mathcal{G}(\vec{y}) = \sum_{\vec{\pi} \in S^m} \varphi(\vec{\pi}) \mathbb{1}_{\vec{\pi}, S^m}(\vec{y}),
\]

where \( \mathbb{1}_{\vec{\pi}, S^m}(\vec{y}) \) is the Lagrange interpolation polynomial defined in **Construction B.2.1**. Note that the above polynomial is vector valued. We now verify the properties of \( \mathcal{G} \).

- **Correctness:** Let \( f \in \mathcal{C} \). If \( f = 0 \) then clearly \( f \circ \mathcal{G} = 0 \). Suppose \( f \neq 0 \). Then there is some \( \vec{\beta} \in \mathcal{H} \) so that \( f(\vec{\beta}) \neq 0 \) as \( \mathcal{H} \) is a hitting set for \( \mathcal{C} \). Thus, there is some \( \vec{\alpha} \in S^m \) so \( \varphi(\vec{\alpha}) = \vec{\beta} \), from which it follows from Lagrange interpolation (**Lemma B.2.4**).
that $G(\beta) = 0$. Thus, $(f \circ G)(\beta) = f(G(\beta)) = f(\beta) \neq 0$. It follows that $f \circ G \neq 0$ as a polynomial in $\mathbb{F}[y]$.

explicitness: This follows from the explicitness of Lagrange interpolation (see Lemma B.2.3), that $|S^m| < d|H|$, and the explicitness of $\varphi$.

individual degree: This follows from the degree of Lagrange interpolation.

Remark 3.2.23. The above show a rough equivalence of hitting sets and generators, where one can measure this roughness by using the above equivalences in a circle (for some fixed class $C$ of degree $< d$ polynomials) and studying how the parameters change.

If one starts with a hitting set $H \subseteq \mathbb{F}^n$, converts $H$ to a generator $\mathcal{G}: \mathbb{F}^{\lceil \log_d |H| \rceil} \to \mathbb{F}^n$ of individual degree $< d$, and then converts $\mathcal{G}$ to another hitting set $H'$, we have that $|H'| \leq (d \cdot d)^{\lceil \log_d |H| \rceil} < d^2 |H|^2$. Thus, these transformations are fairly efficient with respect to hitting set size.

However, if the original hitting set $H$ is $t(n,d)$-explicit, then the above only yield that $H'$ is $\text{poly}(t(n,d), |H|, n,d)$-explicit. This can be a large loss, especially if $|H|$ is super-polynomial in $n,d$. In particular, the conversion from a hitting set to a generator is not efficient with respect to explicitness.

Similarly, if one starts with a generator $\mathcal{G}: \mathbb{F}^s \to \mathbb{F}^n$ of individual degree $< D$, converts this to a hitting set $H$ of size $(dD)^s$, and then converts this to a generator $\mathcal{G}': \mathbb{F}^{s'} \to \mathbb{F}^n$ of individual degree $< d$, then $s' = \lceil \log_d (dD)^s \rceil = \lceil s(\log_d D + 1) \rceil$. Thus, as above, these transformations are fairly efficient with respect to the seed-length of the generator.

Turning to explicitness, if the original generator $\mathcal{G}$ is $t(n,d)$-explicit, then the above only yield that $\mathcal{G}$ is $\text{poly}(t(n,d), d^s, n)$-explicit. This can be a large loss, especially if $s$ is super-constant. Again, this loss arises from the conversion of a hitting set to a generator.

Remark 3.2.24. Note that in the above “hitting set to generator to hitting set”, there is a quadratic loss in the hitting set size. One can decrease this loss by using a larger individual degree in the “hitting set to generator” construction. In particular, if $H$ is of polynomial-size, one can use a univariate generator, so that $|H'| \leq d|H|$. However, this only suffices when the field $\mathbb{F}$ is large enough.  

\[ \diamond \]

42
Chapter 4

Algebraic Branching Programs (ABPs) and their Variants

Abstract: In this chapter we define algebraic branching programs (ABPs), which are a subclass of algebraic circuits that naturally correspond to the power of iterated matrix multiplication, as introduced by Nisan [Nis91]. We show how these ABPs can be put into a normal form, called a layered ABP, that makes this connection more evident. We define ABPs in broad generality, over any non-commutative ring, but will mostly be interested in ABPs over $\mathbb{F}[x]$ or non-commutative ABPs over $\mathbb{F}(x)$ (ncABPs). We also show how these models are equivalent to a trace of matrix power model.

Finally, we introduce subclasses of layered ABPs by restricting different layers to have certain sorts of labels, and this naturally yields the read-once oblivious ABP (roABP) and set-multilinear (smABP) model. We then discuss lower bounds for roABPs. We give exponential lower bounds, essentially due to Nisan [Nis91] (who phrased this result for ncABPs), by characterizing the width needed to represent a polynomial as roABPs in terms of the dimension of the partial derivative matrix. Finally, we show the equivalence (due to Forbes-Shpilka [FS13b]) of the notion of evaluation dimension of Saptharishi [Sap12] with the notion of roABP width in any variable order.

4.1 Algebraic Branching Programs (ABPs)

In this section we define the model of computation known as an algebraic branching program (ABP), due to Nisan [Nis91]. We begin with the definition of the computational device. We present this definition in generality over any non-commutative ring, and will specialize it later.

Definition 4.1.1 (R-ABPs). Let $R$ be a non-commutative ring. Let $\mathcal{L} \subseteq R$. An $R$-algebraic branching program (R-ABP) $B$ with labels $\mathcal{L}$ is an (unordered) directed acyclic graph $G = (V, E)$ with a labelling $E \to \mathcal{L}$.

The size of $B$ is the number of edges $|E|$. The depth of $B$ is the length of the longest path in the graph $G$. ♦
Note that, unlike algebraic circuits, we require here that the underlying graph-theoretic structure is a graph, not a multi-graph. This will simplify some of the technicalities, and also does not affect the computational power in any way, as parallel edges can be merged (summing their labels) while preserving the output of the program.

Note that the depth is a well-defined finite number as the underlying graph is a directed acyclic graph, so each path can only visit node in $G$ at most once, so that the depth is $\leq |V| - 1$ (as we count edges).

That the underlying directed acyclic graph is unordered is a reference to the notion of an ordered directed acyclic graph (Definition 3.1.1) where the in-edges of every vertex are ordered. This order is important for algebraic circuits computing in a non-commutative ring. However, even in non-commutative rings this order will not be needed for algebraic branching programs.

We now define how $R$-ABPs compute.

**Definition 4.1.2 (R-ABP Computation).** Let $R$ be a non-commutative ring. Let $L \subseteq R$. An $R$-ABP with labels $L$ with underlying graph $G = (V, E)$. Let $\{s_i\}_i \subseteq V$ be the source nodes in $G$, and let $\{t_j\}_j$ be the sinks.

Define the output of the $R$-ABP to be the set elements of $R$, for all sources $s_i$ and sinks $t_j$ of

$$f_{i,j} = \sum_{\text{path } p} \prod_{e \in p} \text{label}(e),$$

where $\text{label}(e)$ denotes the label in $L$ of the edge $e$, and the multiplication $\prod_{e \in p} \text{label}(e)$ is done in order of the path. That a matrix $F \in R^{n \times m}$ is computed by an $R$-ABP means that $F$ is a submatrix of the matrix $(f_{i,j})_{i,j}$. ♦

Notice that this summation is well-defined as there are only a finite number of paths between a source $s_i$ and a sink $t_j$ as the underlying graph $G$ is a directed acyclic graph. The primary case of interest is when the $R$-ABP is single-output, and this will be assumed unless otherwise specified.

We now specialize the definition of an $R$-ABP to cases of particular interest, the commutative and non-commutative polynomial rings, that is $R = \mathbb{F}[x]$ or $R = \mathbb{F}(x)$. In particular, the edge labels will be degree $\leq 1$ polynomials.

**Definition 4.1.3 (ABPs).** Let $\mathbb{F}$ be a field and $n \geq 1$. An algebraic branching program (ABP) (over $\mathbb{F}[x_1, \ldots, x_n]$ (respectively $\mathbb{F}(x_1, \ldots, x_n)$)) is an $\mathbb{F}[x]$-ABP (respectively $\mathbb{F}(x)$-ABP) with labels $L$ consisting of all degree $\leq 1$ polynomials.

The size is $(n + 1)|E|$, where $E$ is the set of underlying edges. The ABP is $t(n)$-explicit if it can be computed in $t(n)$ steps in the unit-cost model over $\mathbb{F}$. ♦

We define the size to be $(n + 1)|E|$ as each edge label is determined by $n + 1$ coefficients. One could imagine a more refined notion of size where zero-coefficients are not counted, but we do not consider that here.
4.2 Layered Algebraic Branching Programs

While algebraic branching programs offer a natural mode of computation, it is often much more convenient to assume they fit a normal form in the sense that they are layered. This means that the vertices of the underlying graph are partitioned into an ordered set of layers and edges only go from the \((i-1)\)-st layer to the \(i\)-th layer. We now present the formal definition for general non-commutative rings.

**Definition 4.2.1 (Layered \(R\)-ABPs).** Let \(R\) be a non-commutative ring. Let \(d \geq 1\). Let \(\mathcal{L}_1, \ldots, \mathcal{L}_d \subseteq R\). A layered \(R\)-ABP with labels \(\mathcal{L}_1, \ldots, \mathcal{L}_d\) is an \(R\)-ABP with labels in \(\bigcup_{i \in [d]} \mathcal{L}_i\) where

- The vertices are partitioned into \(d+1\) layers, \(V = V_0 \sqcup V_1 \sqcup \cdots \sqcup V_d\), so that \(V_0\) is a set of sources and \(V_d\) is a set of sinks.
- Each edge \(e\) goes from \(V_{i-1}\) to \(V_i\) for some \(i \in [d]\), so that \(E \subseteq \bigcup_{i \in [d]} V_{i-1} \times V_i\)
- An edge \(e\) from \(V_{i-1}\) to \(V_i\) is labelled with an element in \(\mathcal{L}_i\).

The width is \(\max_i |V_i|\). The size is \(w^2d\). The output of a layered ABP is the set of \(|V_0| \cdot |V_d|\) outputs associated with the paths from the sources in \(V_0\) to the sinks in \(V_d\). That a matrix \(F \in R^{n \times m}\) is computed by a layered ABP means that \(F\) is a submatrix of the matrix in \(R^{V_0 \times V_d}\) of outputs. 

Note that we can always assume that \(F\) equals the matrix of outputs, as we can discard sources and sinks appropriately.

Note that the above definition allows sources and sinks to be present in an intermediate layer, but these are not relevant for the output of the layered \(R\)-ABP. One could prune these intermediate sources and sinks, but this would sacrifice some amount of explicitness.

It follows that the number of edges in a layered width-\(w\) \(R\)-ABP is at most \(w^2d\), so we use that quantity to define its size (as an upper bound of its actual complexity). The depth is exactly \(d\) (as all sources are in the 0-th layer and all sinks are in the \(d\)-th layer).

We now observe that algebraic branching programs can be made layered with a small penalty. As an \(R\)-ABP \(B\) on \(m\) nodes can have paths of length \(\leq m^1\) this means that if we take \(m + 1\) copies of \(B\), where now the edges go from the \((i-1)\)-st copy to the \(i\)-th copy, it follows that all source-sink paths in \(B\) are now represented by some path starting at the 0-th layer and ending in some sink in some copy of \(B\). However, we need these paths to terminate at a sink in the last layer, so we must add edges to ensure that these paths reach the last layer. These edges will be labelled with value 1 so that their weight is preserved.

**Lemma 4.2.2.** Let \(R\) be a non-commutative ring. Let \(\mathcal{L} \subseteq R\). Given an \(R\)-ABP \(B\) with labels \(\mathcal{L}\) and \(\leq m\) nodes, there is an equivalent-output layered \(R\)-ABP \(B'\) with labels \((\mathcal{L}_i)_{i \in [m]}\) computing in \(R\) of width \(m\) and depth \(m\), where \(\mathcal{L}_i = \mathcal{L} \cup \{1\}\).

\(^1\)Actually, length \(\leq m - 1\).
Further, $B'$ can be computed from $B$ using $\text{poly}(m)$ operations, where the function labelling the edges of $B'$ is computed only by copying edge labels from $B'$ (without knowing their contents) or by assigning the edge label 1.

Proof: Let $G = (V, E)$ denote the underlying directed acyclic graph for the ABP $B$, where $S \subseteq V$ are the sources and $T \subseteq V$ are the sinks. Define the graph $G' = (V', E')$ as follows. Let $V' := V_0' \cup \cdots \cup V_m'$ where $V_i' := V$ for $0 < i < m$ and $V_0' := S \times \{0\}$ and $V_m' := T \times \{m\}$, so that $V' \subseteq V \times \{0, \ldots, m\}$. Now define $E'$ by

\[
E' := \{(u \times \{i - 1\}, v \times \{i\}) \mid (u, v) \in E, 0 < i < m\} \\
\cup \{(u \times \{0\}, v \times \{1\}) \mid (u, v) \in E, u \in S\} \\
\cup \{(u \times \{m - 1\}, v \times \{m\}) \mid (u, v) \in E, v \in T\} \\
\cup \{(t \times \{i - 1\}, t \times \{i\}) \mid t \in T, i \in [m]\}
\]

where edges of the first three types inherit their label from $B$ and edges of the last type get label 1. Now note that the four edge types yield distinct edges. That the first three are distinct is clear. That the fourth is distinct from the first three is also true, because there can be no edges $(u, u) \in E$ as these are self-loops and thus forbidden in a directed acyclic graph.

Further, by construction we see that the output of the ABP $B'$ is some matrix in $R^{(S \times \{0\}) \times (T \times \{m\})} \equiv R^{S \times T}$. As the output of $B$ is a matrix in $R^{S \times T}$ by definition of $S$ and $T$, we simply need to show that the corresponding entries are the same.

Now consider any path in $B'$ from $V_0'$ to $V_m'$, $p : s \times \{0\} \rightsquigarrow t \times \{m\}$ for $s \in S$ and $t \in T$. Notice that by construction it must have the form $(v_i \times \{i\})^{m}_{i=0}$ for $v_i \in V$. Further, note that once the path lands in $T$ it must remain there, that is $v_i \in T \Rightarrow v_j = v_i$ for $j \geq i$. This is because the vertices in $T$ are sinks in $B$ so that there are no edges in $E$ leading out of any vertex in $T$. Consequently, the only edges in $B'$ leading out of some vertex in $T \times \{i\} \subseteq V_i'$ are those of the fourth type in the above definition of $E'$.

Thus, any path $p : s \times \{0\} \rightsquigarrow t \times \{m\}$ must take the form $s \times \{0\} = v_0 \times \{0\} \rightarrow v_1 \times \{1\} \rightarrow \cdots \rightarrow v_k \times \{k\} = t \times \{k\} \rightarrow t \times \{k + 1\} \rightarrow \cdots \rightarrow t \times \{m\}$, where $v_0 \rightarrow v_1 \rightarrow \cdots \rightarrow v_k$ is a path $s \rightsquigarrow t$ in $B$, as the edges used come from the first three types in $E'$, which all arise from $E$.

Further, note that this map from paths in $B'$ to paths in $B$ is in fact a bijection. It is injective as for two paths in $B'$ to yield the same path in $B$, they must follow that path in $B$ (increasing the layer in $B'$ at each edge) until edges of the fourth type are used. However, these is only one way to use these edges as they are stationary points. The map is surjective as any such $s \rightsquigarrow t$ path in $B$ is of length $\leq m$, one can use the fourth-type edges to pad this to a path $s \times \{0\} \rightsquigarrow t \times \{m\}$ in $B'$.

Finally, note that an $s \times \{0\} \rightsquigarrow t \times \{m\}$ path in $B'$ and the corresponding $s \rightsquigarrow t$ path in $B$ have the same weight (product of their labels) as the edges $t \times \{i - 1\}$ to $t \times \{i\}$ in $B'$ have label 1.

Thus, there is an explicit weight-preserving bijection between the $V_0'$ to $V_m'$ paths in $B'$ and source-sink paths in $B$. As such, the outputs are the same.
explicitness: It is clear that the above construction is efficient, and that the labels of $E'$ are computed only by copying the labels of $E$ or setting the label to 1.

Thus, as layered $R$-ABPs are more structured than $R$-ABPs and the conversion to a layered $R$-ABP is only of a modest penalty, we shall mostly discuss layered $R$-ABPs from now on.

While layered $R$-ABPs are already more structured than arbitrary $R$-ABPs, we will want an even more structured way of understanding layered $R$-ABPs. In particular, the next result observes that the output of a layered $R$-ABP naturally corresponds to the result of iterated matrix multiplication applied to the adjacency matrices of the layers. As such, the $i$-th matrix will have entries coming from the $i$-th label set of the layered ABP (the set $L_i$) or zero.

**Lemma 4.2.3.** Let $R$ be a non-commutative ring. Let $d \geq 1$. Let $L_1, \ldots, L_d \subseteq R$.

Let $B$ be a layered $R$-ABP (with underlying graph $G = (\sqcup_{0 \leq i \leq d} V_i, E)$) with labels $L_1, \ldots, L_d$ computing in $R$. For $i \in [d]$ define $M_i \in (L_i \cup \{0\})^{V_{i-1} \times V_i}$ such that the $(u, v)$-th entry in $M_i$ is the label on the edge from $u \in V_{i-1}$ to $v \in V_i$, or 0 if no such edge exists. Then

$$F := \prod_{i \in [d]} M_i := M_1 M_2 \cdots M_d \in R^{V_0 \times V_d}$$

has

$$F_{s_i, t_j} = \text{output of $B$ from source } s_i \text{ to sink } t_j,$$

for $s_i \in V_0$ and $t_j \in V_d$.

Further, the matrices $M_i$ can be computed from $B$ in $\text{poly}(\sum_i |V_i|)$ operations in $O(\log(\sum_i |V_i|))$ parallel time, where we only allow the entries of the $M_i$ to be copies of labels of the edges of $B$ (copying without knowing their contents) or by assigning the entry 0.

**Proof:** That $M_i \in L_i^{V_{i-1} \times V_i}$ is clear from construction. We now prove the properties of $F$ by induction on $d$.

$d = 1$: Then the output of $B$ from source $s_i$ to sink $t_j$ is simply the label of the edge from $s_i$ to $t_j$ (or zero if no edge exists). This is exactly the entry of $(M_1)_{s_i, t_j}$, and as $F = M_1$ this yields the claim.

$d > 1$: Let $V_1 = \{r_k\}_k$. So then the output of $B$ from source $s_i$ to sink $t_j$ is

$$f_{i, j} := \sum_{p: s_i \rightarrow t_j} \prod_{e \in p} \text{label}(e)$$

As $B$ is layered, all paths from $V_0$ to $V_d$ must pass through exactly one node in $V_1$, so we can partition paths along their intersection with $V_1$,

$$= \sum_{r_k \in V_1} \sum_{p: s_i \rightarrow r_k \rightarrow t_j} \prod_{e \in p} \text{label}(e)$$

47
and breaking these paths into two paths,

\[
\sum_{r_k \in V_1} \left( \sum_{\text{path } p' \atop p': s_i \rightarrow r_k} \prod_{e \in p'} \text{edge } e \prod_{e \in p''} \text{label } (e) \right) \cdot \left( \sum_{\text{path } p'' \atop p'': r_k \rightarrow t_j} \prod_{e \in p'} \text{edge } e \prod_{e \in p''} \text{label } (e) \right)
\]

appealing to induction, as taking contiguous subsets of layers of a layered $R$-ABP yields a layered $R$-ABP

\[
\sum_{r_k \in V_1} (M_1)_{i,k} \cdot (M_2 \cdots M_d)_{k,j}
\]

\[
= (M_1 M_2 \cdots M_d)_{i,j}
\]

**explicitness:** The bound on the number of operations is clear from construction. That this can also be done in parallel is also clear, as each label can be copied in parallel.

The above result transforms a depth-$d$ layered $R$-ABP into $d$ matrices of varying sizes. We now observe that by padding these matrices with zeroes, we can assume they all have the same size (equalling the width of the original layered $R$-ABP).

**Lemma 4.2.4.** Let $R$ be a non-commutative ring. Let $d \geq 1$. Let $F \in R^{n \times m}$. Then the following are equivalent.

1. The matrix $F$ can be computed by a depth-$d$, width $\leq w$ layered $R$-ABP, with label sets $\mathcal{L}_1, \ldots, \mathcal{L}_d \subseteq R$.

2. For $i \in [d]$, there are matrices $M_i \in (\mathcal{L}_i \cup \{0\})^{(\leq w) \times (\leq w)}$ such that

\[
F = \left( \prod_{i \in [d]} M_i \right)_{[n] \times [m]}
\]

3. For $i \in [d]$, there are matrices $M_i \in (\mathcal{L}_i \cup \{0\})^{w \times w}$ such that in block notation,

\[
\begin{bmatrix}
F & 0 \\
0 & 0
\end{bmatrix} = \prod_{i \in [d]} M_i,
\]

where $0$ is the zero matrix with the appropriate dimensions in each case.
Further, the conversion between these different forms can be done in \(\text{poly}(n, w, d)\) operations in \(O(\log nwd)\) parallel time, where labels in \(R\) can be manipulated only by copying (without inspection), by setting to the constant 0, or testing of a label is 0.

**Proof:** (1) \(\Rightarrow\) (2): Lemma 4.2.3 furnishes \(M_i\) such that \(F\) is a submatrix of \(\prod_i M_i\), where \(M_i \in (\mathcal{L}_i \cup \{0\})^{V_i \times V_i}\), where \(|V_i| \leq w\). By redefining \(M_1 \leftarrow PM_1\) and \(M_d \leftarrow M_dQ\) for appropriate permutation matrices, we get that \(F = (\prod_i M_i)_{[n] \times [m]}\) as desired.

(2) \(\Rightarrow\) (3): Define \(M_i' \in (\mathcal{L}_i \cup \{0\})^{w \times w}\) to be \(M_i\) padded with zeros to become \(w \times w\) sized, that is,

\[
M_i' := \begin{bmatrix} M_i & 0 \\ 0 & 0 \end{bmatrix},
\]

where we use block-matrix notation. By the properties of block-matrix multiplication, it follows that

\[
\left( \prod_i M_i' \right)_{n \times m} = \left( \prod_i \begin{bmatrix} M_i & 0 \\ 0 & 0 \end{bmatrix} \right)_{n \times m} = \left( \begin{bmatrix} \prod_i M_i & 0 \\ 0 & 0 \end{bmatrix} \right)_{n \times m} = F,
\]

as desired.

Now define \(M_i'' \in (\mathcal{L}_i \cup \{0\})^{w \times w}\) by \(M_i'' := M_i'\) for \(0 < i < d\) and

\[
M_0'' = \begin{bmatrix} I_n & 0 \\ 0 & 0 \end{bmatrix} M_0',
\]

and

\[
M_d'' = M_d' \begin{bmatrix} I_m & 0 \\ 0 & 0 \end{bmatrix},
\]

so that \(M_0''\) zeros-out all but the \(n\) rows of \(M_0'\) and \(M_d''\) zeros-out all but the first \(m\) columns of \(M_d'\). Thus,

\[
\prod_i M_i'' = \begin{bmatrix} I_n & 0 \\ 0 & 0 \end{bmatrix} \prod_i M_i' \begin{bmatrix} I_m & 0 \\ 0 & 0 \end{bmatrix} = \left( \prod_i M_i' \right)_{[n] \times [m]} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} F & 0 \\ 0 & 0 \end{bmatrix},
\]

as desired.

(3) \(\Rightarrow\) (1): For \(1 < i < d\) define \(M_i' := M_i\). Define \(M_1' := (M_1)_{[n] \times \bullet}\) and \(M_d' := (M_d)_{\bullet \times [m]}\), so that \(M_i'\) takes the first \(n\) rows of \(M_1\) and \(M_d'\) takes the first \(m\) columns of \(M_d\). It follows then that

\[
F = \prod_i M_i.'
\]

Now define the layered \(R\)-ABP so that \(V_0 = [n], V_d = [m]\) and \(V_i = [w]\) for \(0 < i < d\). Define the edges between \(V_{i-1}\) and \(V_i\) by the non-zero entry pattern of \(M_i'\), where the labels of these entries are defined by the entries in \(M_i'\). That is, \(M_i'\) is the adjacency matrix of \(V_{i-1}\) and \(V_i\). Note that even though \(M_i'\) has entries in \(\mathcal{L}_i \cup \{0\}\), we have not introduced any edges with zero labels, so the resulting layered \(R\)-ABP...
will have the $i$-th layer only using labels from $\mathcal{L}_i$.

Thus the layered $R$-ABP has the desired syntactic properties, and computes $F$ as seen by applying Lemma 4.2.3.

**explicitness:** That the above three forms can be efficiently constructed in few total operations from each other is clear from the above arguments. In particular, while some matrix multiplications over $R$ are performed, this is simply a convenient notation for selecting certain matrix entries, which can be performed with copying or zero-setting operations. In the conversion from matrices to an layered $R$-ABP, some zero-testing of labels is required to prune out edges with zero labels.

That these operations are also computable in logarithmic parallel time is also clear, as they are simply copying/renaming which can all be done in parallel.

We now observe that computation by small width layered $R$-ABPs are closed under addition. The intuition for this proof is that two layered $R$-ABPs can be run “in parallel” so that there are exactly two source-sink pairs, one computing $f$ and one computing $g$. By then merging these two sources and sinks, one computes $f + g$.

**Lemma 4.2.5.** Let $R$ be a non-commutative ring. Let $d \geq 1$. Let $\mathcal{L}_1, \ldots, \mathcal{L}_d \subseteq R$. Let $f, g$ be computed by layered $R$-ABPs with labels $\mathcal{L}_1, \ldots, \mathcal{L}_d$, where $f$ is computed in width $\leq w$ and $g$ is computed in width $\leq s$. Then $f + g$ is computable by a layered $R$-ABP with labels $\mathcal{L}_1, \ldots, \mathcal{L}_d$ in width $\leq w + s$.

Further, the layered $R$-ABP for $f + g$ can be computed from the layered $R$-ABPs for $f$ and $g$ using $\text{poly}(d, w, s)$ operations in $O(\log dw s)$ parallel time, where labels in $R$ can be copied, zero-tested and set to 0 in unit cost.

Proof: By Lemma 4.2.4, we have $M_i \in (\mathcal{L}_i \cup \{0\})^{w_i-1 \times w_i}$ with $w_i \leq w$ and $N_i \in (\mathcal{L}_i \cup \{0\})^{s_i-1 \times s_i}$ with $s_i \leq s$, such that

$$f = \prod_i M_i, \quad g = \prod_i N_i.$$  

Thus, consider $P_i \in (\mathcal{L}_i \cup \{0\})^{(w_i-1+s_i-1) \times (w_i+s_i)}$ defined by

$$P_i := \begin{bmatrix} M_i & 0 \\ 0 & N_i \end{bmatrix},$$

so that by block matrix multiplication,

$$\begin{bmatrix} f \\ 0 \end{bmatrix} = \begin{bmatrix} \Pi_i M_i & 0 \\ 0 & \Pi_i N_i \end{bmatrix} = \prod_i P_i.$$

Thus, defining $P_1' := P_i$ for $1 < i < d$ and

$$P_1' := \begin{bmatrix} M_1 & N_1 \end{bmatrix}, \quad P_d' := \begin{bmatrix} M_d \\ N_d \end{bmatrix},$$

we have

$$f + g = \prod_i P_i.$$
it follows from more block matrix multiplication that

\[
\prod_i P'_i = P'_1 \cdot \prod_{1<i<d} P'_i \cdot P'_d
\]

\[
= \begin{bmatrix} M_1 & N_1 \end{bmatrix} \cdot \prod_{1<i<d} P'_i \cdot \begin{bmatrix} M_d \\ N_d \end{bmatrix}
\]

\[
= \begin{bmatrix} M_1 & N_1 \end{bmatrix} \begin{bmatrix} \prod_{1<i<d} M_i & 0 \\ 0 & \prod_{1<i<d} N_i \end{bmatrix} \begin{bmatrix} M_d \\ N_d \end{bmatrix}
\]

\[
= \begin{bmatrix} \prod_{1\leq i<d} M_i & \prod_{1\leq i<d} N_i \end{bmatrix} \begin{bmatrix} M_d \\ N_d \end{bmatrix}
\]

\[
= \begin{bmatrix} \prod_{1\leq i\leq d} M_i + \prod_{1\leq i\leq d} N_i \end{bmatrix}
\]

\[
= f + g
\]

Appealing again to Lemma 4.2.4 to construct the layered R-ABP yields the claim, as \( P'_i \in (\mathcal{L}_i \cup \{0\})^{(\leq w+s)\times(\leq w+s)} \) and \( \prod_i P'_i = f + g \).

explicitness: First note that the conversion of a layered R-ABP to its matrix form (Lemma 4.2.4) is constructive. The rest of the proof is also easily seen to be constructive. That this process is constructive in parallel logarithmic time is also immediate as the new matrices are computed simply by copying entries. \( \square \)

We now observe that layered R-ABPs can also be scaled by constants, assuming that the constants are “cheap”. This proof is rather simple, as the labels in the first layer can all be rescaled uniformly by the desired constant.

Lemma 4.2.6. Let \( R \) be a non-commutative ring that is also an \( \mathbb{F} \)-algebra, so that \( \mathbb{F} \subseteq R \). Let \( d \geq 1 \). Let \( \mathcal{L}_1, \ldots, \mathcal{L}_d \subseteq R \) where \( \mathcal{L}_1 \) is also a \( \mathbb{F} \)-vector space. Let \( f \) be computed by a layered R-ABP with labels \( \mathcal{L}_1, \ldots, \mathcal{L}_d \) in width \( \leq w \). Then for any \( \alpha \in \mathbb{F} \), \( \alpha f \) is computable by a layered R-ABP with labels \( \mathcal{L}_1, \ldots, \mathcal{L}_d \) in width \( w \).

Further, the layered R-ABP for \( \alpha f \) can be computed from the layered R-ABPs for \( f \) using \( \text{poly}(d,w) \) operations in \( O(\log dw) \) parallel time, where labels in \( \mathcal{L}_1 \) can added and multiplied by scalars in \( \mathbb{F} \) in unit cost, and labels in \( R \) can be copied, zero-tested and set to 0 in unit cost.

Proof: By Lemma 4.2.4, there exist \( M_i \in (\mathcal{L}_i \cup \{0\})^{(\leq w)\times(\leq w)} \) with \( f = \prod_i M_i \). Now define \( M'_i := \alpha \cdot M_i \) and \( M'_i := M_i \) for \( i > 1 \), where \( M'_i \in (\mathcal{L}_i \cup \{0\})^{(\leq w)\times(\leq w)} \) as \( \mathcal{L}_1 \) is an \( \mathbb{F} \)-vector space. It follows that \( \alpha f = \prod_i M'_i \), and appealing to Lemma 4.2.4 this shows that \( \alpha f \) is computed as desired.

explicitness: As in Lemma 4.2.5, the explicitness of this construction is clear. \( \square \)

We now discuss a specific instantiation of layered R-ABPs, where \( R = \mathbb{F}[x] \) or \( R = \mathbb{F}(x) \), and the labels are degree \( \leq 1 \) polynomials. Thus yields a subclass of ABPs (Definition 4.1.3).

Definition 4.2.7 (Layered ABPs). Let \( \mathbb{F} \) be a field and \( n,d \geq 1 \). An \textbf{layered ABP (over} \( \mathbb{F}[x_1, \ldots, x_n] \) \textbf{respectively} \( \mathbb{F}(x_1, \ldots, x_n) \)) is an layered R-ABP with \( R = \mathbb{F}[x] \)
(respectively $R = \mathbb{F}(\pi)$) with labels $\mathcal{L}_i$ consisting of all degree $\leq 1$ polynomials. The size is $(n + 1)w^2d$.\qedhere

As with non-layered ABPs, we now incorporate the complexity of the labels into the size of the ABP. Note that the depth $d$ is also a bound on the degree of the computed polynomials, so is consistent with the convention of using “$d$” for the degree.

We can instantiate the above results (Lemma 4.2.4, Lemma 4.2.5, Lemma 4.2.6) for $R$-ABPs for this new notion, observing that labels for layered ABPs are just affine forms so can be copied/zero-tested in logarithmic parallel time.

**Corollary 4.2.8.** Consider a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ (or in $\mathbb{F}(x_1, \ldots, x_n)$). Then the following are equivalent.

- The polynomial $f$ is computed by a width-$w$ ($\leq w$), depth-$d$ layered ABP.

- There exist matrices $M_i(\pi) \in \mathbb{F}[\pi]^{w \times w}$ (respectively $M_i(\pi) \in \mathbb{F}(\pi)^{w \times w}$) of degree $\leq 1$ such that $f = \bigoplus_{i=1}^{d} M_i(\pi)$.

- There exist matrices $M_i(\pi) \in \mathbb{F}[\pi]^{(\leq w) \times (\leq w)}$ (respectively $M_i(\pi) \in \mathbb{F}(\pi)^{(\leq w) \times (\leq w)}$) of degree $\leq 1$ such that $f = \bigoplus_{i=1}^{d} M_i(\pi)$, treating $f$ as a $1 \times 1$ matrix.

Further, the conversion between these different forms can be done in $\text{poly}(n, w, d)$ operations in $O(\log nwd)$ parallel time in the unit-cost model over $\mathbb{F}$.

**Corollary 4.2.9.** If polynomials $f, g \in \mathbb{F}[x_1, \ldots, x_n]$ (or in $\mathbb{F}(x_1, \ldots, x_n)$) are computed by depth-$d$ layered ABPs where $f$ is computed by a width $\leq w$ layered ABP that is $t(n, w, d)$-explicit, and $g$ is computed by a width $\leq s$ layered ABP that is $r(n, s, d)$-explicit, then for every $\alpha, \beta \in F$, $\alpha f + \beta g$ is computed by a depth-$d$, width-$w+s$, $\text{poly}(t(n, w, d), r(n, s, d), n, w, s, d)$-explicit layered ABP. \qedhere

We conclude this section by stating that layered ABPs can be efficiently simulated circuits.

**Fact 4.2.10** (Layered ABPs can be Computed by Small Low-Depth Circuits). A polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ (or in $\mathbb{F}(x_1, \ldots, x_n)$) computed by a width-$w$, depth-$d t(n, w, d)$-explicit layered ABP can be computed by a $\text{poly}(t(n, w, d), n, w, d)$-explicit algebraic circuit of size $\text{poly}(n, w, d)$ and depth $\text{polylog}(n, w, d)$.

### 4.3 Trace of Matrix Powers

In this section we study a slightly different model of computation, that of a **trace of a $R$-matrix power**, and show that it is equivalent to a layered $R$-ABP, as first proved by Malod and Portier [MP08] but independently proved in [FS13a]. We do this as this model was used in the results of Mulmuley [Mul12a] on explicit Noether Normalization, which we improve upon in Chapter 9, where we also discuss more of the relevance of the trace of a matrix power model in that context.

We first define the model.
Definition 4.3.1 (Trace of a R-Matrix Power). Let $R$ be a non-commutative ring. Let $d \geq 1$. Let $\mathcal{L} \subseteq R$. A trace of a R-matrix power with labels $\mathcal{L}$ is an expression of the form $\text{tr} M^d$ for $M \in \mathcal{L}$. The width is $w$, the depth is $d$ and the size is $w^2d$.

This model naturally computes the element of $R$ given by $\text{tr} M^d$.

We now establish that traces of $R$-matrix powers can efficiently simulate layered $R$-ABPs. To do this, we first study matrix powers and how they interact with traces.

Lemma 4.3.2. Let $x_0, \ldots, x_{d-1}$ be formally non-commuting variables, and let $R$ be any non-commutative ring. Define $A(\mathcal{X}) \in R[x_0, \ldots, x_{d-1}]^{J \times K}$

$$A(\mathcal{X})_{i,j} := \begin{cases} x_i & \text{if } j = i + 1 \pmod{d} \\ 0 & \text{else} \end{cases}.$$ 

Then $\text{tr}(A(\mathcal{X})^d) = \sum_{i \in [d]} x_ix_{i+1} \cdots x_{(i-1) \pmod{d}}$.

Proof: The matrix $A$ defines an adjacency matrix on a $d$-vertex graph, which is the directed cycle with weights $x_0, x_1, \ldots, x_{d-1}$ ordered cyclically. Raising $A$ to the $d$-power corresponds to the adjacency matrix for the length-$d$ walks on the length-$d$ cycle. The only such walks are single traversals of the cycle, starting and ending at some vertex $i$, and these walks have weight $x_ix_{i+1} \cdots x_{(i-1) \pmod{d}}$. Taking the trace of $A^d$ corresponds to summing the weights of these walks, giving the desired formula.

Alternate proof of Lemma 4.3.2. By definition,

$$(A^d)_{i,i} = \sum_{i_1, \ldots, i_{d-1}} A_{i,i_1}A_{i_1,i_2} \cdots A_{i_{d-1},i}.$$ 

It follows that the only nonzero contribution is when $i_j = i_{j-1} + 1$ for all $j$, when defining $i_0 = i_d = i$ and working modulo $d$, and that this yields $x_ix_{i+1} \cdots x_{(i-1) \pmod{d}}$. The claim follows by summing over $i$.

Corollary 4.3.3. Let $R$ be a non-commutative ring, and let $M_1, \ldots, M_d \in R^{[n] \times [n]}$ be matrices. Define the larger matrix $A \in R^{[nd] \times [nd]}$ by treating $A$ as a block matrix in $(R^{[n] \times [n]})^{[d] \times [d]}$, so that

$$A_{i,j} = \begin{cases} M_i & \text{if } j = i + 1 \pmod{d} \\ 0 & \text{else} \end{cases}.$$ 

Then $\text{tr}(A^d) = d\text{tr}(M_1 \cdots M_d)$.

Proof: The properties of block-matrix multiplication imply that we can treat the $M_i$ as lying in the non-commutative ring of matrices, and thus we apply Lemma 4.3.2 to
the trace of $A^d$ to see that
\[
\text{tr}(A^d) = \sum_{i=1}^{d} \text{tr}(M_i M_{i+1} \cdots M_{(i-1 \mod d)}) = d \text{tr}(M_1 M_2 \cdots M_d)
\]
where the second equality uses that trace is cyclic.

This lemma shows that the trace of a matrix power can embed the trace of a matrix product (up to the factor $d$), and Lemma 4.2.3 shows that traces of matrix products capture layered $R$-ABPs. This leads to the equivalence of traces of matrix powers and layered $R$-ABPs.

**Lemma 4.3.4.** Let $R$ be a non-commutative ring. If $f \in R$ is computable by a width-$w$, depth-$d$ layered $R$-ABP with labels $\mathcal{L}_1, \ldots, \mathcal{L}_d \subseteq R$, then for any $d' \geq d$ such that $\text{char}(R) \nmid d'$, $f$ can be computed by a width-$w d'$, depth-$d'$ trace of a $R$-matrix power, where the matrix has entries coming from $(\frac{1}{d'} \cdot \mathcal{L}_1) \cup (\cup_{i > 1} \mathcal{L}_i) \cup \{0, 1\}$, where $\frac{1}{d'} \cdot \mathcal{L}_1$ is the set of labels from $\mathcal{L}_1$ each multiplied by $\frac{1}{d'}$. In particular, $d' \in \{d, d+1\}$ suffices.

Further, the trace of $R$-matrix power can be computed using $\text{poly}(w, d)$ operations over $R$, where we copy labels, zero-test labels, divide labels by $d'$, or set labels to 0 or 1.

Conversely, if $f \in R$ is computable by a width-$w$, depth-$d$ trace of a $R$-matrix power with entries in $\mathcal{L}$, then $f$ can also be computed by a width-$w^2$, depth $d$ layered $R$-ABP with labels in $\mathcal{L}$.

Further, the layered $R$-ABP can be computed using $\text{poly}(w, d)$ operations over $R$, where we copy labels, zero-test labels, or set labels to 0 or 1.

**Proof:** \(\text{ABP} \implies \text{trace of matrix power:} \) By Lemma 4.2.4 there are matrices $M_i \in (\mathcal{L}_i \cup \{0\})^{[w] \times [w]}$ so that
\[
\begin{bmatrix}
  f \\
  0 \\
  0
\end{bmatrix} = \prod_i M_i.
\]
In particular, we find that $f = \text{tr}\prod_i M_i$. For $i \in [d']$, define new matrices $M'_i$ such that for $i = 1$, $M'_i := M_i / d'$, and for $1 < i \leq d$, we set $M'_i := M_i$, and for $i > d$, define $M'_i := I_w$, the $[w] \times [w]$ identity matrix.

By linearity of the trace, it follows that $f = \frac{1}{d'} \cdot \text{tr}(M'_1 \cdots M'_d M'_{d+1} \cdots M'_{d'})$. Applying Corollary 4.3.3 to these $M'_i$, we get a single matrix $M \in R^{[wwd] \times [wwd]}$ so that $f = \text{tr} M^d$. Note further that $M$ is explicit as desired, and has the appropriate entries.

Finally note that $d$ and $d + 1$ cannot both be divisible by the characteristic of $R$, so one of them works as $d'$.

\(\text{trace of matrix power} \implies \text{ABP:} \) Suppose $f = \text{tr} M^d$, where $M \in \mathcal{L}^{w \times w}$. It follows by Lemma 4.2.4 that each $(i, i)$ entry of $M^d$ is computable by a width-$w$, depth-$d$ layered $R$-ABP. By Lemma 4.2.5 we can then compute their sum in width-$w^2$, and can explicitly compute the layered ABP with the desired labels. \(\square\)

Thus the above shows that, up to polynomial factors in size, ABPs and traces of matrix powers compute the same polynomials. We note that there is also an equivalent
computational model, defined by the determinant, which we mention for completeness.

**Definition 4.3.5.** A polynomial \( f(x_1, \ldots, x_n) \) is a width \( w \) projection of a determinant if there exists a matrix \( A(\pi) \in \mathbb{F}[x_1, \ldots, x_n]^{[w] \times [w]} \) whose entries are affine functions in \( \pi \) such that \( f(\pi) = \det(A(\pi)) \). The size of a projection of a determinant is \( nw \). ◊

Valiant [Val79] (see also [MP08]) showed any small ABP can be simulated by a small projection of a determinant. Conversely, phrased in the language of this paper, Berkowitz [Ber84] gave a small ABP for computing a small projection of a determinant. Thus, the projection of determinant model is also equivalent to the ABP model. We summarize these results in the following theorem.

**Theorem 4.3.6.** The computational models of algebraic branching programs, traces of matrix powers, and projections of determinants are equivalent, up to polynomial blow up in size.

In particular, this implies that derandomizing black-box PIT is equally hard for all three of these computational models.

### 4.4 Read-Once Oblivious ABPs (roABPs)

We next define the ABP-variant that will comprise much of our attention, that is, the read-once oblivious algebraic branching program, an instantiation of layered \( \mathbb{F}[\pi] \)-ABPs where the edge labels are univariate polynomials. While these variables are commutative, the structure of computation does not exploit this as it reads each variable in a fixed order (so is oblivious). After discussing roABPs and their properties, we also define the set-multilinear ABP (smABP) model, which is morally equivalent to the roABP model.

**Definition 4.4.1 (Read-Once (Oblivious) Algebraic Branching Program).** Let \( \mathbb{F} \) be a field, \( n \geq 1 \), and let \( \pi : [n] \to [n] \) be a permutation. An read-once (oblivious) algebraic branching program (roABP) with variable order \( \pi \) is a depth-\( n \) layered \( \mathbb{F}[\pi] \)-ABP with labels

\[
\mathcal{L}_i := \{ \text{univariate polynomials in } x_{\pi(i)} \}
\]

The program computes in a **known order** if \( \pi \) is a fixed known permutation (such as the identity permutation), and computes in an **unknown order** if \( \pi \) is unknown.

The program has **individual degree** \( < d \) if the labels in all \( \mathcal{L}_i \) are also restricted to be polynomials with individual degree \( < d \). The size of a width-\( w \), individual-degree-\( (< d) \) roABP is \( nw^2d \). ◊

Again, we define size to capture the complexity of the labels (which are univariate polynomials of degree \( < d \)).

Recall that an ABP computes by summing over all the paths the weight of that path, where the weight of that path is the product of the labels on the edges. An ABP is read-once if in each path each variable appears in at most one label. An
read-once ABP is \textit{oblivious} if in each path the variables always appear in the same order (where this order is defined by some permutation $\pi$ of $[n]$). In this work we will always consider read-once ABPs that are also oblivious, so will henceforth drop this qualifier.

Frequently, we will have results that work with any fixed variable order $\pi$. Instead of referencing $\pi$ explicitly, we will assume that $\pi$ is the identity permutation and thus the roABP computes “in the variable order $x_1 < \cdots < x_n$.” Achieving these results for arbitrary $\pi$ can then be done using the appropriate renaming of the variables.

We can now specialize Lemma 4.2.4 to roABPs, noting that operations on the labels of roABPs (low degree univariate polynomials) can be done efficiently in the unit cost model over $\mathbb{F}$.

\textbf{Corollary 4.4.2.} Consider a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ and let $\pi : [n] \to [n]$ be a permutation. Then the following are equivalent.

\begin{itemize}
  \item The polynomial $f$ is computed by a individual degree-($< d$), width-($\leq w$) roABP in variable order $\pi$.
  \item There exist matrices $M_i(x_{\pi(i)}) \in \mathbb{F}[x_{\pi(i)}]_{w \times w}$ of (individual) degree $< d$ such that $f = (\prod_{i=1}^{n} M_i(x_{\pi(i)}))_{1,1}$.
  \item There exist matrices $M_i(x_{\pi(i)}) \in \mathbb{F}[x_{\pi(i)}]_{(\leq w) \times (\leq w)}$ of (individual) degree $< d$ such that $f = \prod_{i=1}^{n} M_i(x_{\pi(i)})$, treating $f$ as a $1 \times 1$ matrix.
\end{itemize}

Further, the conversion between these different forms can be done in $\text{poly}(n, w, d)$ operations in $O(\log nwd)$ parallel time in the unit-cost model over $\mathbb{F}$. \hfill $\Box$

Note that the above matrices are univariate, so their individual degree equals their (total) degree.

We now use this normal form to observe that when an roABP computes a polynomial $f$, it never needs to use computation of a higher degree than the individual degree of $f$. This follows as one can discard higher-degree terms without increasing the size of the computation, and observe that the result is unaffected.

\textbf{Lemma 4.4.3.} Let $f \in \mathbb{F}[x]$ be a polynomial computed by a $t(n, w)$-explicit roABP of width $\leq w$ in variable order $x_1 < \cdots < x_n$, so that $f = (\prod_{i=1}^{n} M_i(x_i))_{1,1}$, for matrices $M_i \in \mathbb{F}[x_i]_{w \times w}$. Then there are matrices $M_i' \in \mathbb{F}[x_i]_{w \times w}$ with $\deg_{x_i} M_i' \leq \deg_{x_i} f$ with $f = (\prod_{i=1}^{n} M_i'(x_i))_{1,1}$, so that $f$ is computed by a $\text{poly}(t(n, w), n, w, \deg f)$-explicit roABP in variable order $x_1 < \cdots < x_n$ of individual degree $\leq \deg f$ and width $w$.

\textbf{Proof:} Define $M_i' := \sum_{j=0}^{\deg f} \text{Coeff}_{x_j}^f M_i \cdot x_j^j$. It is clear that these have the desired size and degree. That their product yields $f$ is also clear because by construction any non-zero monomial in $(\prod_{i=1}^{n} M_i(x_i))_{1,1} - (\prod_{i=1}^{n} M_i'(x_i))_{1,1}$ must have some variable $x_i$ that appears with a power $> \deg_{x_i} f$. However, there can be no such non-zero monomials as $(\prod_{i=1}^{n} M_i(x_i))_{1,1}$ cannot contribute any as it equals $f$, and $(\prod_{i=1}^{n} M_i'(x_i))_{1,1}$ cannot contribute any by construction. Thus $(\prod_{i=1}^{n} M_i(x_i))_{1,1} - (\prod_{i=1}^{n} M_i'(x_i))_{1,1} = 0$, so that $f = (\prod_{i=1}^{n} M_i'(x_i))_{1,1}$ as desired.

Further, the explicitness of this construction is clear. \hfill $\Box$
The above claim shows that when computing a polynomial with an roABP, the individual degree of this roABP is not important, as it can always be assumed to be at most \( \text{ideg} f \) while preserving all of the other relevant properties (width, explicitness, variable order). As such, we will sometimes drop the discussion of the individual degree unless it is somehow relevant.

Similarly, we can specialize Lemma 4.2.5 and Lemma 4.2.6 to roABPs, noting that the labels for roABPs (univariate polynomials) can be multiplied by constants in \( \mathbb{F} \) efficiently.

**Corollary 4.4.4.** If polynomials \( f, g \in \mathbb{F}[x_1, \ldots, x_n] \) are computed by a individual degree-\((< d)\) roABPs in a variable order \( \pi : [n] \to [n] \), where \( f \) is computed by a width \( \leq w \) roABP that is \( t(n, w, d) \)-explicit, and \( g \) is computed by a width \( \leq s \) roABP that is \( r(n, s, d) \)-explicit, then for every \( \alpha, \beta \in \mathbb{F} \), \( \alpha f + \beta g \) is computed by an individual degree-\((< d)\), width-\((w + s)\) roABP in variable order \( \pi \). Further, this roABP can be computed in \( \text{poly}(t(n, w, d) \cdot r(n, s, d) \cdot nwsd) \) operations in \( O(\log(t(n, w, d) \cdot r(n, s, d) \cdot nwsd)) \) parallel time.

We now show that the above results give an easy way to show that any polynomial can be computed by a (possibly large) roABP. The construction follows by taking any polynomial and expanding it in its complete expansion of monomials. Each monomial can be trivially computed by a roABP of width 1 (in any variable order). Appealing to Corollary 4.4.4 to show how roABPs can be added yields the claim. Thus, the roABP model is complete.

**Lemma 4.4.5.** Let \( f \in \mathbb{F}[x_1, \ldots, x_n] \) be a polynomial that is the sum of \( \leq s \) non-zero monomials. Then for any permutation \( \pi : [n] \to [n] \), \( f \) can be computed by a roABP of width \( s \) in variable order \( \pi \). In particular, any \( f \) with \( \text{ideg} f < d \) can be computed in width \( d^n \).

**Proof:** First consider \( f = \bar{\pi} \). Define \( M_i(x_{\pi(i)}) \in \mathbb{F}[x_{\pi(i)}]^{1 \times 1} \) by
\[
M_i(x_{\pi(i)}) := \begin{bmatrix} x_{\pi(i)}^{a_{\pi(i)}} \end{bmatrix}.
\]
Thus, \( \bar{\pi} = \prod_i M_i(x_{\pi(i)}) \), so that any monomial \( \bar{\pi} \) can be computed by a width-1 roABP in the variable order \( \pi \).

Now consider a polynomial \( f = \sum_{\pi \in S} c_{\pi} \bar{\pi} \) where \( |S| \leq s \). It follows from Corollary 4.4.4 that \( f \) can be computed by a width-\( s \) roABP, as desired. Further, for any \( \text{ideg} f < d \) we note that \( s \leq d^n \).

### 4.4.1 Set-Multilinear ABPs (smABPs)

We give here a variant of roABPs, called **set-multilinear ABPs**. If we have variables \( \bar{x} = x_1 \sqcup \cdots \sqcup x_d \), a polynomial \( f(\bar{x}) \) is **set-multilinear**, if \( \deg_{x_i} f \leq 1 \) for each \( i \). For an ABP to be set-multilinear, it should naturally respect this variable partition, and this leads to the following definition.

**Definition 4.4.6** (Set-Multilinear Algebraic Branching Program). Let \( \mathbb{F} \) be a field, \( n, d \geq 1 \). Let \( \bar{x} \) be a set of \( nd \) variables. A **set-multilinear algebraic branching program**
program (smABP) with partition \( \pi = \pi_1 \sqcup \cdots \sqcup \pi_d \) is a depth-\( d \) layered \( \mathbb{F}[\pi] \)-ABP with labels
\[
\mathcal{L}_i := \{ \text{affine polynomials in } \pi_i \},
\]
where each \( \pi_i \) contains \( n \) variables.

The program has partition size \( n \), and size of a width-\( w \) smABP is \( nw^2d \).

It follows that smABPs naturally compute set-multilinear polynomials. Their relation with roABPs is also evident as each partition is read-once, and more on this connection will be given in Theorem 6.5.21.

### 4.5 Lower Bounds for roABPs

In this section we discuss lower bounds for the roABP model, by translating the arguments of [Nis91] for non-commutative ABPs into this model. We first describe the measure.

#### 4.5.1 The Complexity of Variable Disjoint Inner Products

We now define a complexity measure of a polynomial \( f(\pi, \gamma) \) that attempts to measure the “correlation” between the variables \( \pi \) and the variables \( \gamma \).

**Definition 4.5.1.** Let \( \text{Coeff}_\gamma : \mathbb{F}[\pi, \gamma] \to 2^{\mathbb{F}[\pi]} \) be the space of \( \mathbb{F}[\pi][\gamma] \) coefficients (function), defined by
\[
\text{Coeff}_\gamma(f) := \text{span} \left\{ \text{Coeff}_{\pi_i}(f) \right\}_{i \in \mathbb{N}^n},
\]
where coefficients of \( f \) are taken in \( \mathbb{F}[\pi][\gamma] \).

Similarly, define \( \text{Coeff}_\pi : \mathbb{F}[\pi, \gamma] \to 2^{\mathbb{F}[\gamma]} \) by taking coefficients in \( \mathbb{F}[\gamma][\pi] \).

Slightly abusing notation, for \( f \in \mathbb{F}[\pi, \gamma] \) define
\[
\dim \text{Coeff}_{\pi|\gamma}(f) := \max\{ \dim \text{Coeff}_\pi(f), \dim \text{Coeff}_\gamma(f) \}.
\]

This complexity measure thus partitions, or cuts, the variables into two parts and measures the complexity of their interaction. Note that not all cuts are equally powerful. As an example, consider a multilinear polynomial \( f(\pi, \gamma) \) on \( n \) variables \( \pi, \gamma \). If \( \gamma \) has \( k \) variables then \( \text{Coeff}_\gamma(f) \) has size at most \( 2^k \) as there can only be \( 2^k \) such coefficients in \( \gamma \). At the same time, this space can have size at most \( 2^{n-k} \), as there can only be \( 2^{n-k} \) such coefficients in the \( \pi \) variables. By symmetry, the same bounds apply to \( \text{Coeff}_\pi(f) \). Thus, for the above measure to yield large complexity we must ensure that \( \min\{k, n-k\} \) is large.

Note that we have two spaces in the above, one for \( \pi \) and one for \( \gamma \), each with their associated dimensions. The above discussion suggests a symmetry between them. The vectors spaces are certainly different though (one is a subspace of \( \mathbb{F}[\pi] \), the other a subspace of \( \mathbb{F}[\gamma] \)). However, the next lemma shows that their dimensions are the same, so that this justifies the (symmetric) quantity \( \dim \text{Coeff}_{\pi|\gamma}(f) \) as the “right” notion. This is done by making concrete the above vector spaces by choosing a basis, so that
the vector spaces are now row/column spaces of a particular matrix (sometimes called the partial derivative matrix).

**Lemma 4.5.2.** Consider \( f \in \mathbb{F}[x, y] \). Define the scalar matrix

\[
(C_f)_{\alpha, \beta} := \text{Coeff}_{x^\alpha y^\beta}(f),
\]

where coefficients are taken in \( \mathbb{F}[x, y] \), for \( \|\alpha\|_1, \|\beta\|_1 \leq \deg f \). Then

\[
\dim \text{Coeff}_x(f) = \dim \text{Coeff}_y(f) = \text{rank} C_f,
\]

and thus

\[
\dim \text{Coeff}_x(f) = \dim \text{Coeff}_y(f) = \dim \text{Coeff}_{x|y}(f).
\]

**Proof:** Consider the matrix \( C_f \) ranging over all \( \alpha, \beta \). While this matrix is infinite, not that it only has a finite number of non-zero entries, and all of these entries occur when \( \|\alpha\|_1, \|\beta\|_1 \leq \deg f \).

Now consider \( \text{Coeff}_y(f) \), which is spanned by \( \{\text{Coeff}_y^\beta(f)\}_{\|\beta\|_1 \leq \deg f} \) where again we can restrict to a finite set of \( \beta \) as the discarded \( \beta \) yield coefficients which are zero polynomials. Note that these polynomials lie in \( \mathbb{F}[x] \) and have \( \deg_x \leq \deg f \). Thus, we may write them out as vectors in the monomial basis, so that

\[
\text{Coeff}_x(\text{Coeff}_y^\beta(f)) = \text{Coeff}_{x|y}(f).
\]

Thus, the vectors \( \{\text{Coeff}_y^\beta(f)\}_{\|\beta\|_1 \leq \deg f} \), when written in the monomial basis, exactly correspond to the columns of the matrix \( C_f \). Thus, \( \dim \text{Coeff}_y(f) = \dim \text{col-span}(C_f) \).

In a symmetric manner, \( \dim \text{Coeff}_x(f) = \dim \text{row-span}(C_f) \). As \( \text{row-span}(C_f), \text{col-span}(C_f) \) have equal dimension by basic linear algebra, the claim follows.

We now show that the dimension of these vector spaces for a polynomial \( f \) is a lower bound on the size of certain variable-disjoint inner-product for \( f \). We do this using the basis-free definition of these vector spaces, although working with matrices is equally valid (and we will do so later).

**Lemma 4.5.3.** Let \( f \in \mathbb{F}[x, y] \). If \( f(x, y) \) admits the decomposition \( f(x, y) = \sum_{i \in [r]} g_i(x)h_i(y) \), then \( r \geq \dim \text{Coeff}_{x|y}(f) \).

**Proof:** Taking the coefficients in \( \mathbb{F}[x]|y] \) of the above expression,

\[
\text{Coeff}_y(f) = \text{Coeff}_{x|y} \left( \sum_{i \in [r]} g_i(x)h_i(y) \right)
\]

so as \( \text{Coeff}_y^x \) is a linear operator (Lemma B.1.1) and polynomials in \( \mathbb{F}[x] \) are scalars in this context,

\[
= \sum_{i \in [r]} g_i(x) \cdot \text{Coeff}_{x|y}^x(h_i(y))
\]
and as \( \text{Coeff}_{\overline{c}}(h_i(\overline{g})) \in \mathbb{F}, \)
\[
\in \text{span}\{g_i(\overline{x})\}_{i \in [r]}.
\]
As this holds for all \( \overline{b} \), it follows that \( \text{Coeff}_{\overline{c}}(f) \) is contained in a subspace spanned by \( r \) polynomials, so thus \( \dim \text{Coeff}_{\overline{c}}(f) = \dim \text{Coeff}_{\overline{c}}(f) \leq r \), as desired.

We now work to show that not only is the dimension of the coefficients a lower bound for such variable-disjoint inner-products, it is also an upper bound. We first show that whenever such a inner-product does not meet the dimension bound, we can simplify the inner-product. We will take the matrix viewpoint, as the following lemma will facilitate the proof.

Lemma 4.5.4. Let \( A \in \mathbb{F}^{n \times r} \) and \( B \in \mathbb{F}^{r \times m} \) with \( r \leq n, m \). If \( \text{rank}(A), \text{rank}(B) = r \) then \( \text{rank}(AB) = r \).

Proof: As \( \text{rank}(A) = r \) there are \( r \) linearly independent rows. Let \( P \in \mathbb{F}^{n \times n} \) be a permutation matrix that permutes these rows to the top of \( A \). That is, top \( r \times r \) submatrix \( PA \) is full rank, and in particular
\[
PA = \begin{bmatrix} A' \\ A'' \end{bmatrix},
\]
with \( A' \in \mathbb{F}^{r \times r}, A'' \in \mathbb{F}^{(n-r) \times r} \) and \( \det(A') \neq 0 \). Similarly, we can find a permutation matrix \( Q \in \mathbb{F}^{m \times m} \) so that
\[
BQ = \begin{bmatrix} B' \\ B'' \end{bmatrix},
\]
with \( B' \in \mathbb{F}^{r \times r}, B'' \in \mathbb{F}^{r \times (m-r)} \) and \( \det(B') \neq 0 \). Thus, by block-matrix multiplication
\[
PABQ = \begin{bmatrix} A'B' & A'B'' \\ A''B' & A''B'' \end{bmatrix}.
\]
As the determinant is multiplicative, and \( A' \) and \( B' \) are both \( r \times r \) matrices, it follows that \( \det(A'B') \neq 0 \), so that \( \text{rank}(PABQ) \geq \text{rank}(A'B') = r \). However, as \( P \) and \( Q \) are permutation matrices, \( \text{rank}(PABQ) = \text{rank}(ABQ) = \text{rank}(AB) \), showing that \( \text{rank}(AB) \geq r \). That \( \text{rank}(AB) \leq r \) follows from that \( \text{rank}(AB) \leq \text{rank}(A) \) for any \( A \) and \( B \).

Example 4.5.5. Note that one cannot hope to prove such a claim when “\( \text{rank}(A), \text{rank}(B) = r \)” is relaxed to “\( \text{rank}(A), \text{rank}(B) \approx r \)”.

For example, one could take
\[
A := \begin{bmatrix} I_r & 0_r \\ 0_r & 0_r \end{bmatrix}, \quad B := \begin{bmatrix} 0_r & 0_r \\ 0_r & I_r \end{bmatrix},
\]
where \( I_r \) is the \( r \times r \) identity matrix and \( 0_r \) is the \( r \times r \) zero matrix. Then both \( A \) and \( B \) are \( 2r \times 2r \) matrices with rank \( r \), but their product \( AB = 0_{2r} \) (by block matrix multiplication) has rank zero.

\begin{proof}[\Box]

60
\end{proof}
We now use this lemma to show that non-optimal variable-disjoint inner-products can be simplified, using a matrix approach. The intuition is that if \( f(x, y) \) has the inner-product \( f(x, y) = \sum_{i \in [r]} g_i(x)h_i(y) \), where the \( g_i \) are linearly independent and the \( h_j \) are linearly independent, then the above lemma shows that \( f \) itself must have “rank \( r \)”, which in this case means that \( \dim \text{Coeff}_{x \| y}(f) = r \). Thus, when \( r > \dim \text{Coeff}_{x \| y}(f) \) it means there is some redundancy in either the \( g_i \) or \( h_j \), and by removing this redundancy we can get a smaller formula.

**Lemma 4.5.6.** Let \( f \in \mathbb{F}[\overline{x}, \overline{y}] \). Suppose \( f(x, y) = \sum_{i \in [r]} g_i(x)h_i(y) = \langle \overline{g}, \overline{h} \rangle \) where \( \overline{g} \in \mathbb{F}[\overline{x}]^r \) and \( \overline{h} \in \mathbb{F}[\overline{y}]^r \). If \( r > \dim \text{Coeff}_{x \| y}(f) \), then there exist matrices \( B, C \in \mathbb{F}^{r \times r'} \) for \( r' < r \) so that \( f(x, y) = \langle B\overline{g}, C\overline{h} \rangle \) so that \( B\overline{g} \in \mathbb{F}[\overline{x}]^{r'} \) and \( C\overline{h} \in \mathbb{F}[\overline{y}]^{r'} \).

**Proof:** As in **Lemma 4.5.2**, consider the scalar matrix \( C_f \) defined by

\[
(C_f)_{\overline{a}, \overline{b}} := \text{Coeff}_{x \| y}(f).
\]

Define the scalar matrix \( G \) by

\[
G_{\overline{a}, i} := \text{Coeff}_{x \| y}(g_i),
\]

and similarly define the scalar matrix \( H \) by

\[
H_{j, \overline{b}} := \text{Coeff}_{y \| y}(h_j).
\]

As above, these matrices are finite as we can truncate them to only by indexed by monomials of degree at most \( \deg f \). Then it follows that, taking coefficients in \( \mathbb{F}[\overline{x}, \overline{y}] \),

\[
(C_f)_{\overline{a}, \overline{b}} = \text{Coeff}_{x \| y}(f) = \sum_i \text{Coeff}_{x \| y}(g_i(x)h_i(y)) = \sum_i \text{Coeff}_{x \| y}(g_i(x)) \cdot \text{Coeff}_{y \| y}(h_i(y)) = \sum_i G_{\overline{a}, i}H_{j, \overline{b}} = (GH)_{\overline{a}, \overline{b}},
\]

so that \( C_f = GH \).

As \( G \) has \( r \) columns, and \( H \) has \( r \) rows, we can apply **Lemma 4.5.4** to see that if \( \text{rank}(G) = \text{rank}(H) = r \) then \( r = \text{rank}(C_f) = \dim \text{Coeff}_{x \| y}(f) \) (where this second equality uses **Lemma 4.5.2**). However, as \( r > \dim \text{Coeff}_{x \| y}(f) \) by hypothesis, it follows that one of \( G \) or \( H \) has rank \( < r \).

Assume \( G \) has rank \( < r \). Then there is some \( i_0 \in [r] \) so that \( G_{\bullet, i_0} \in \text{span}\{G_{\bullet, i}\}_{i \neq i_0} \).

As the summation \( f = \sum_i g_i h_i \) can be permuted freely, we can assume without loss of generality that \( i_0 = r \) so that \( G_{\bullet, r} \in \text{span}\{G_{\bullet, i}\}_{i < r} \), or equivalently, \( g_r(x) = \sum_{i < r} c_{i} g_i(x) \)
for some \( c_i \in \mathbb{F} \). Thus,
\[
 f(x, y) = \sum_i g_i(x) h_i(y) \\
= g_r(x) h_r(y) + \sum_{i<r} g_i(x) h_i(y) \\
= \left( \sum_{i<r} c_i g_i(x) \right) h_r(y) + \sum_{i<r} g_i(x) h_i(y) \\
= \sum_{i<r} g_i(x) (h_i(y) + c_i h_r(y)) \\
= \langle B\overline{g}, C\overline{h} \rangle ,
\]
where the matrices \( B, C \in \mathbb{F}^{[r-1] \times [r]} \) are defined to make the above equation true, that is,
\[
 (B)_{i,j} = \begin{cases} 1 & i = j \\ 0 & \text{else} \end{cases},
\]
and
\[
 (C)_{i,j} = \begin{cases} 1 & i = j < r \\ c_i & j = r \\ 0 & \text{else} \end{cases}.
\]

The case when \( H \) has rank \( < r \) is symmetric. \( \square \)

Note that one can recover Lemma 4.5.2 in a matrix formalism using the above, by seeing that \( \dim \text{Coeff}_{x|y}(f) = \text{rank}(C_f) = \text{rank}(GH) \leq \text{rank}(G), \text{rank}(H) \leq r \).

Also, note that it is tempting in the above proof to assert that \( G \) and \( H \) must have rank \( \text{dim} \text{Coeff}_{x|y}(f) = \text{rank}(C_f) \) so that one could take \( r' = \text{dim} \text{Coeff}_{x|y}(f) \) in the first step. However, the example Example 4.5.5 shows that this can be false. That is, as we are only guaranteed that \( \text{rank}(GH) = \text{rank}(C_f) < r \), that \( G \) has \( r \) columns and \( H \) has \( r \) rows means that \( G, H \) could have rank \( > \text{rank}(C_f) \).

Applying induction to the above lemma allows to conclude (along with the lower bound Lemma 4.5.3) that \( \dim \text{Coeff}_{x|y}(f) \) characterizes the minimal size of a variable-disjoint inner-product of \( f(x, y) \).

**Corollary 4.5.7.** Let \( f \in \mathbb{F}[\overline{x}, \overline{y}] \) with \( s = \dim \text{Coeff}_{x|y}(f) \). If \( f(x, y) \) admits the decomposition \( f(x, y) = \sum_{i \in [r]} g_i(x) h_i(y) = \langle g(x), h(y) \rangle \), then \( r \geq s \), and there exist matrices \( B, C \in \mathbb{F}^{s \times r} \) so that \( f(x, y) = \langle B\overline{g}, C\overline{h} \rangle \) so that \( B\overline{g} \in \mathbb{F}^{[x]^s} \) and \( C\overline{h} \in \mathbb{F}^{[y]^s} \).

In particular, \( \dim \text{Coeff}_{x|y}(f) = \min \{ r \mid \exists \overline{g} \in \mathbb{F}^{[x]^r}, \overline{h} \in \mathbb{F}^{[y]^r}, f = \langle \overline{g}, \overline{h} \rangle \} \). \( \square \)

### 4.5.2 roABPs as Variable Disjoint Inner Products

The previous section gave a complexity measure of a polynomial \( f \) that characterizes (Corollary 4.5.7) the size of any variable disjoint inner product that yields \( f \). We now observe that small-width roABPs yield variable disjoint inner-products with small size. This is because \( f \) is a \( 1 \times 1 \) matrix, and when writing \( f \) as a product of variable
disjoint matrices, one can “cut” this product between any two adjacent variables and collapse the matrices to yield a variable disjoint inner-product, to which Corollary 4.5.7 can be applied. This result gives matrices that then compress this product at the variables where the product was cut, and touches no other variables. By applying this compression to each possible cut point, we can compress each layer of the roABP to the dimension bound of the associated variable partition. This allows us to give lower bounds for the width of any roABP computing a polynomial $f$ in terms of this complexity measure, and also shows a matching upper bound.

Note that in this section we will only consider roABPs in the variable order $x_1 < \cdots < x_n$, as a suitable permutation can be applied to the variables to get results for roABPs in other orders. Further, we will ignore the individual-degree of the roABP, as it can always be assumed to be small (Lemma 4.4.3).

**Lemma 4.5.8.** Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be computed by a roABP in variable order $x_1 < \cdots < x_n$, so that $f = \prod_{i=1}^n M_i(x_i)$ for matrices $M_i(x_i) \in \mathbb{F}[x_i]^{w_{i-1} \times w_i}$, for some $w_i$ (treating $f$ as a $1 \times 1$ matrix).

Then there exists $M'_i(x_i) \in \mathbb{F}[x_i]^{w'_{i-1} \times w'_i}$ so that $f = \prod_{i=1}^n M'_i(x_i)$ and for $0 < i < n$, $w'_i := \dim \text{Coeff}_{x_{\leq i} | x_{> i}}(f)$,

and $w'_0, w'_n := 1$, where $x_{\leq i}$ is the vector of variables $(x_1, \ldots, x_i)$ and $x_{> i}$ is the vector of variables $(x_{i+1}, \ldots, x_n)$.

In particular, $f$ is computed by an roABP with width $\max_i \dim \text{Coeff}_{x_{\leq i} | x_{> i}}(f)$.

Further, in any roABP computing $f$, so that $f = \prod_{i=1}^n M_i(x_i)$ with $M_i(x_i) \in \mathbb{F}[x_i]^{w_{i-1} \times w_i}$, it must be that $w_i \geq \dim \text{Coeff}_{x_{\leq i} | x_{> i}}(f)$ for $0 < i < n$. In particular, the roABP must have width $\geq \max_i \dim \text{Coeff}_{x_{\leq i} | x_{> i}}(f)$.

**Proof:** We first show how, for any $0 < i_0 < n$, to transform $f = \prod_{i=1}^n M_i(x_i)$ into an expression of the form $f = \prod_{i=1}^{i_0} M'_i(x_i)$ for matrices $M'_i(x_i) \in \mathbb{F}[x_i]^{w'_{i-1} \times w'_i}$, where $w'_{i_0} = \dim \text{Coeff}_{x_{\leq i_0} | x_{> i_0}}(f)$ and $w'_i = w_i$ for $i \neq i_0$.

As $f = \prod_{i=1}^n M_i(x_i)$ is a $1 \times 1$ matrix, it follows that $w_0 = w_n = 1$. Now cut this product into

$$f = \prod_{i \leq i_0} M_i(x_i) \cdot \prod_{i > i_0} M_i(x_i).$$

It follows that $\overline{g}(x_{\leq i_0}) := \prod_{i \leq i_0} M_i(x_i)$ is a row-vector, thus in $\mathbb{F}[x_{\leq i_0}]^{1 \times w_{i_0}}$. Similarly, it follows that $\overline{h}(x_{> i_0}) := \prod_{i > i_0} M_i(x_i)$ is a column-vector, thus in $\mathbb{F}[x_{> i_0}]^{w_{i_0} \times 1}$. Rewriting the above expression, we can write $f$ as the variable disjoint inner-product (and taking the transpose of $\overline{g}$ so it becomes a column vector)

$$f(\overline{x}_{\leq i_0}, \overline{x}_{> i_0}) = \langle \overline{g}(x_{\leq i_0}), \overline{h}(x_{> i_0}) \rangle.$$

Taking $w'_{i_0} = \dim \text{Coeff}_{x_{\leq i_0} | x_{> i_0}}(f)$, Corollary 4.5.7 provides matrices $B, C \in \mathbb{F}^{w'_{i_0} \times w_{i_0}}$ so that

$$f = \langle B\overline{g}^t, C\overline{h} \rangle.$$
Translating this back to the matrix equation, we see that

\[
\begin{align*}
f &= \langle B\vec{g}^t, C\vec{h} \rangle = (B\vec{g}^t)^t \cdot C\vec{h} \\
&= \left( \prod_{i \leq i_0} M_i(x_i) \cdot B^t \right) \cdot \left( C \cdot \prod_{i > i_0} M_i(x_i) \right) \\
&= \prod_{i < i_0} M_i(x_i) \cdot \left( M_{i_0}(x_{i_0}) \cdot B^t \right) \cdot \left( C \cdot M_{i_0+1}(x_{i_0+1}) \right) \cdot \prod_{i > i_0+1} M_i(x_i) .
\end{align*}
\]

Thus, defining

\[
M'_{i_0} := M_{i_0}(x_{i_0}) \cdot B^t \in \mathbb{F}[x_{i_0}]^{w_{i_0}-1 \times w'_{i_0}} ,
\]

and

\[
M'_{i_0+1} := C \cdot M_{i_0+1}(x_{i_0+1}) \in \mathbb{F}[x_{i_0+1}]^{w'_{i_0} \times w_{i_0+1}} ,
\]

and \( M'_i := M_i \) for \( i \notin \{i_0, i_0 + 1\} \), we get the desired expression \( f = \prod_{i=1}^n M'_i(x_i) \), as the dimensions are correct.

Thus, the above argument shows that given any product \( f = \prod_{i=1}^n M_i(x_i) \) and any \( 0 < i_0 < n \) we can modify it so that \( w_{i_0} \) meets the appropriate dimension bound without modifying any \( w_i \) for \( i \neq i_0 \). Thus, by applying this argument to each of \( 1, \ldots, n - 1 \) in order, we get the desired expression \( f = \prod_{i=1}^n M'_i(x_i) \) where each \( w'_i \) meets the dimension bound.

Appealing to Corollary 4.4.2, we see that the above expression yields the desired upper bound on the width of a roABP computing \( f \). That the lower bound holds is also clear (from Corollary 4.4.2 and Corollary 4.5.7), as the above argument shows how any width-\( w \) roABP induces variable disjoint inner-products of size \( w \), and \( w \) must be at least the dimension of the relevant coefficient space.

4.5.3 Polynomials Requiring Large Variable Disjoint Inner Products

We now exhibit a polynomial (essentially the palindrome polynomial of Nisan [Nis91]) that cannot be computed by a small roABP in a particular order. This polynomial also has superficial similarities to the inner-product function, at least when considering their communication complexity.

**Lemma 4.5.9.** Consider \( f(\vec{x}, \vec{y}) \in \mathbb{F}[x_1, \ldots, x_n, y_1, \ldots, y_n] \) defined by

\[
f(\vec{x}, \vec{y}) = \prod_i (x_i + y_i) .
\]

Then \( \dim \text{Coeff}_{\vec{x}\vec{y}}(f) = 2^n \).
Proof: $\geq$: By Lemma 4.5.2, $\dim \text{Coeff}_{\pi|y}(f) = \dim \text{Coeff}_{\pi}(f)$. For $\overline{b} \in \{0, 1\}^n$, consider taking coefficients in $\mathbb{F}[\overline{x}][\overline{y}]$,

$$\text{Coeff}_{\overline{y}^b}(f) = \text{Coeff}_{\overline{y}^b} \left( \prod_i (x_i + y_i) \right) = \text{Coeff}_{\overline{y}^b} \left( \prod_i (x_i^1 y_i^0 + x_i^0 y_i^1) \right)$$

$$= \text{Coeff}_{\overline{y}^b} \left( \sum_{\pi \in \{0, 1\}^n} \overline{x}^{\pi} \overline{y}^{\overline{\pi}} \right)$$

$$= \overline{x}^{\pi} \overline{b}.$$ 

Thus, every $\text{Coeff}_{\overline{y}^b}(f)$ for $\overline{b} \in \{0, 1\}^n$ yields a distinct monomial, so these coefficients are linearly independent in $\mathbb{F}[\overline{x}]$, so $\dim \text{Coeff}_{\overline{y}^b}(f) \geq 2^n$.

$\leq$: This follows from observing that if $\overline{b} \notin \{0, 1\}^n$ then $\text{Coeff}_{\overline{y}^b}(f) = 0$ as $\deg_{y_i}(f) = 1$ for every $i \in [n]$, so the only coefficients contributing to $\dim \text{Coeff}_{\overline{y}}(f)$ are those with $\overline{b} \in \{0, 1\}^n$ of which there are $2^n$. \(\square\)

One can also see this result by considering the matrix $C_f$ of Lemma 4.5.2 and observing that it contains a large permutation matrix.

We now use this result to show that the palindrome polynomial requires large roABPs in some variable orders. We also give a matching upper bound for those variable orders, and show that there are other variable orders where the palindrome polynomial is extremely easy. Note that because roABPs are without loss of generality low-degree (Lemma 4.4.3) we ignore this parameter in the below statement.

Corollary 4.5.10. Consider $f(\overline{x}, \overline{y}) \in \mathbb{F}[x_1, \ldots, x_n, y_1, \ldots, y_n]$ defined by

$$f(\overline{x}, \overline{y}) = \prod_i (x_i + y_i).$$

Then

1. For all permutations $\pi$ of the variables $\overline{x}, \overline{y}$, $f$ can be computed by a roABP in variable order $\pi$ in width $\leq 2^n$.

2. There exist permutations $\pi$ on the variables $\overline{x}, \overline{y}$ (those where $\pi(\overline{x}) < \pi(\overline{y})$), so that any roABP computing $f$ in variable order $\pi$ must be of width $\geq 2^n$.

3. There exist permutations $\pi$ on the variables $\overline{x}, \overline{y}$ (the variable order $x_1 < y_1 < x_2 < y_2 < \cdots < x_n < y_n$), where there is a $\text{poly}(n)$-explicit roABP of width 2 for computing $f$ in the variable order $\pi$. $x_1 < y_1 < x_2 < y_2 < \cdots < x_n < y_n$.

Proof: (1): This follows from the completeness of roABPs (Lemma 4.4.5) that $f$ has ideg $f < 2$.

(2): This follows from the lower bound for the dimension of the coefficients of $f$ (Lemma 4.5.9) and how this dimension characterizes the width of roABPs (Lemma 4.5.8).
Consider
\[ M_i(x_i) = \left[ \begin{array}{cc} x_i & 1 \end{array} \right] \in F[x_i]^{1 \times 2}, \quad N_i(y_i) = \left[ \begin{array}{c} 1 \\ y_i \end{array} \right] \in F[y_i]^{2 \times 1}. \]

Thus, \( M_i(x_i)N_i(y_i) = x_i + y_i \), so that \( \prod_{i=1}^n M_i(x_i)N_i(y_i) = f \). This is thus an roABP computation in the desired order and width (appealing to Corollary 4.4.2). That this construction is explicit is also clear.

\[ \square \]

4.5.4 Evaluation Dimension

In the previous subsections we defined a complexity measure of a polynomial \( f \) in terms of the dimension of its coefficients when we partition the variables into \( \overline{x}, \overline{y} \). In this subsection we give another complexity measure, due to Saptharishi [Sap12], in terms of evaluations, and show that it equals the complexity measure in terms of coefficients, via interpolation (over large fields).

We begin with the definition of the complexity measure.

Definition 4.5.11. Let \( \text{Eval}_{\overline{y}} : F[\overline{x}, \overline{y}] \to 2^{F[\overline{x}]} \) be the space of \( F[\overline{x}] \) evaluations (function), defined by
\[
\text{Eval}_{\overline{y}}(f) := \text{span}_F \{ f|_{y \leftarrow \beta} \}_{\beta \in F[\overline{y}]}.
\]

Similarly, define \( \text{Eval}_{\overline{x}} : F[\overline{x}, \overline{y}] \to 2^{F[\overline{y}]} \) by replacing \( \overline{x} \) with all possible evaluations \( \overline{\alpha} \in F[\overline{\alpha}] \).

This measure thus partially evaluates a polynomial on all substitutions of \( \overline{y} \), whereas the complexity measure from before would take all coefficients of \( \overline{y} \). Using the equivalence of coefficients and evaluations in the \( F \)-algebra \( F[\overline{x}] \) (Corollary 6.2.10), one can see these two measures are equal over large enough fields.

Corollary 4.5.12. Let \( f \in F[\overline{x}, \overline{y}] \). If \( |F| > \text{ideg} f \) then \( \text{Eval}_{\overline{y}}(f) = \text{Coeff}_{\overline{y}}(f) \).

Proof: By the definitions,
\[
\text{Coeff}_{\overline{y}} = \text{span}_F \{ \text{Coeff}_{\overline{y}}(f) \}_{\overline{y} \in F[\overline{y}]}, \quad \text{Eval}_{\overline{y}} = \text{span}_F \{ f(\overline{x}, \overline{\alpha}) \}_{\overline{\alpha} \in F[\overline{\alpha}]}.
\]

Thus, by the equivalence of coefficients and evaluations in the \( F \)-algebra \( F[\overline{x}] \) (Corollary 6.2.10) over fields larger than the individual degree of the target polynomial, it follows these two spans are the same.

\[ \square \]

We now define the notion of evaluation dimension, where we partial evaluate a polynomial \( f \) in all possible ways.

Definition 4.5.13. Let \( \dim \text{Eval} : F[\overline{x}] \to \mathbb{N} \) be defined as
\[
\max_{S \subseteq \overline{x}} \dim \text{Eval}_{\overline{x} \mid S}(f).
\]

\[ \diamond \]
We now observe that by the previous section, we can relate the evaluation dimension of \( f \) to the width of roABPs computing \( f \). However, as we now consider all partial evaluations, we now get a bound on the width in any order. As before, we will ignore the individual degree aspect of the roABP, as without loss of generality it matches the individual degree of the polynomial \( f \) (Lemma 4.4.3).

**Corollary 4.5.14.** Let \( f \in \mathbb{F}[\pi] \). Then \( \dim \text{Eval}(f) \leq w \) iff for all permutations \( \pi \) of the variables \( \pi \), \( f \) can be computed by an roABP of width \( \leq w \) in variable order \( \pi \).

**Proof:** From the above equivalence of \( \text{Eval}_{\pi|_S}(f) \) and \( \text{Coeff}_{\pi|_S}(f) \) (Corollary 4.5.12), it follows that \( \dim \text{Eval}(f) \leq w \) iff for all partitions \( \pi = (\gamma, \zeta) \), \( \dim \text{Coeff}_{\pi|_S}(f) \leq w \).

Now consider a permutation \( \pi \) of \( \pi \). By Lemma 4.5.8, \( f \) can be computed in width \( \leq w \) in variable order \( \pi \) iff \( \dim \text{Coeff}_{\pi|_S}(f) \leq w \) for all \( \pi \), where the order \( \leq \pi \) is the order induced by the permutation \( \pi \).

\[ \implies \] Putting the above together, it follows that the bound \( \dim \text{Eval}(f) \leq w \) implies that \( \dim \text{Coeff}_{\pi|_S}(f) \leq w \) for all \( i \), so that \( f \) is computable in order \( \pi \).

\[ \iff \] Now consider any partition \( \pi = (\gamma, \zeta) \). It follows that there is then a permutation \( \pi \) of \( \pi \) so that \( \gamma = \pi_{\leq \pi} \) and \( \zeta = \pi_{> \pi} \) for some \( i \). As \( f \) is computable by a width \( \leq w \) roABP in variable order \( \pi \), it follows that \( \dim \text{Coeff}_{\gamma|\pi}(f) = \dim \text{Coeff}_{\pi|_S}(f) \leq w \). As this was for any partition, the bound follows. \[ \square \]
Chapter 5

Rank Condensers

Abstract: In this chapter we define the main pseudorandom object that powers the hitting set construction of Chapter 6. This object we dub a seeded rank condenser. This object maps a large dimensional ambient space to a smaller dimensional ambient space, while preserving the rank of all low-dimensional subspaces. We will want a rank condenser that is randomness efficient, so that few seeds are required. In this chapter, we define this object and explore its variants. We also show non-explicitly that randomness efficient rank condensers exist, as well as explicitly match the non-explicit construction.

In this chapter, we define and construct (seeded) rank condensers, with a special emphasis on getting rank condensers that are randomness efficient. A rank condenser in a type of dimension reduction. Generally speaking, dimension reduction is the process of mapping a large dimensional ambient space to a medium dimensional space such that all small objects are preserved. In most dimension reduction instances there must be a distribution of such maps as no single map can preserve all small objects, so that rather we ask that for any fixed small object that it is preserved with high probability over the distribution of maps. In the case of rank condensers, we will consider a set of linear maps $\mathbb{F}^n \rightarrow \mathbb{F}^m$ such that for any low-dimensional subspace one of the maps in the set preserves the rank of the subspace.

We will first give the definition of a rank condenser, in both a hitting-set version and a generator version. We then discuss some variations of the definition and show equivalences between some of these variations. We then give an existential result showing that randomness efficient rank condensers exist, through a probabilistic method argument that uses the careful estimates given in Appendix D for counting subspaces. Finally, we conclude with an explicit randomness efficient rank condenser, that matches the parameters of the non-explicit construction. This explicit rank condenser will be crucial for the black-box PIT results of Section 6.5.

We briefly remark here that a rank condenser, like many dimension reduction instances, is an information theoretic object. That is, while there is a notion of being large or small, these are not viewed from the perspective of computational complexity but rather the amount of actual information present. However, we stress that such objects are still widely applicable to computational pseudorandomness as often computation can be understood by breaking it into small parts such that each
part contains only a small amount of information. Information theoretic reasoning can then be applied on each small part, and the parts can be recombined using the understanding of how the computation was decomposed. The usage of rank condensers in this work (Section 6.5) will follow this paradigm.

5.1 Definition of Rank Condensers

We begin with the motivation for rank condensers, which is the following question.

**Question 5.1.1.** Given a matrix $M \in \mathbb{F}^{n \times m}$, can one perform Gaussian elimination on $M$ without knowing $M$?

The above question is, of course, not well-defined. But it is motivated by the observation that many algorithms crucially rely on Gaussian elimination, and that Gaussian elimination is a process that is non-oblivious in that its actions are highly input-dependent. In some situations this input-dependence is undesirable, for example in situations where communication is a bottleneck this input-dependence would imply many rounds of communication as every action depends on an input that is potentially expensive to inspect.

In the context of this work, we shall see a white-box PIT algorithm for read-once (oblivious) algebraic branching programs in Section 6.3, and this algorithm will crucially use Gaussian elimination as a subroutine. In particular, the use of Gaussian elimination is the essentially only reason this algorithm yields a white-box PIT algorithm instead of a black-box PIT algorithm. Thus, the above question can naturally be interpreted as asking how to turn this algorithm into a black-box PIT algorithm.

The usage of Gaussian elimination in the above-mentioned algorithm is particularly simple. That is, we do not need to find a reduced row-echelon form, but rather only seek to prune a large set of vectors living in a low-dimensional space to a much smaller set (ideally of number at most the dimension bound).

**Question 5.1.2.** Given $v_1, \ldots, v_n \in \mathbb{F}^r$ where $n \gg r$, how can one find a spanning set $\{u_1, \ldots, u_m\}$ for $\text{span} \{v_i\}$ with $m \lesssim r$, where the algorithm is oblivious to the input $\{v_i\}$?

Note that Gaussian elimination has the particular feature that $\{u_i\} \subseteq \{v_i\}$. However, in the above-mentioned algorithm, this requirement is actually unnecessary and thus we phrase the question without this restriction. Without this relaxation the problem seems quite difficult. In particular, if we plant $r$ linearly independent vectors amongst $n - r$ zero vectors, then finding a small spanning set would seemingly require trying all $\binom{n}{r}$ possible subsets.

The key to answering the above question comes from two ideas. The first is randomness. That is, when the input $\{v_i\}$ is essentially unknown, no fixed algorithm can always succeed as it simply does not have enough information. Thus the “only” way an algorithm can succeed is by doing something random. The second idea is to pick linear combinations of the $\{v_i\}$ as the $u_j$, as this allows more flexibility than just sub-sampling the $\{v_i\}$. Putting these two ideas together, we get the solution concept

70
we call a rank condenser, which we now define.

**Definition 5.1.3.** Let \( F \) be a field, and \( n \geq r \). A multi-set \( E \subseteq F^{m \times n} \) is said to be a (seeded) rank condenser (hitting set) for rank \( \leq r \) if for every matrix \( M \in F^{n \times k} \) of rank \( \leq r \), there is some \( E \in E \) such that \( \text{rank}_F(E \cdot M) = \text{rank}_F(M) \).

The rank condenser \( E \) is \( t(n,m) \)-explicit if there is an algorithm such that given an index into \( E \), the corresponding element of \( E \) can be computed in \( t(n,m) \)-time in the unit-cost computation model over \( F \).

We will often drop “seeded” and “hitting set” as it will be clear from context.

Note that \( \text{rank}(EM) \leq \text{rank}(M) \) always, so the only non-trivial part of fulfilling the definition is showing that \( \text{rank}(EM) \geq \text{rank}(M) \).

**Remark 5.1.4.** Note that in the above we only require some \( E \in E \) to give the desired rank bound. One could also imagine scenarios when most \( E \) work. In this more general notion one can consider \( \lg |E| \) to be the seed length, or number of random bits, that the rank condenser uses. However, this more general notion is not needed in this work, as we will always end up enumerating all \( E \in E \) as opposed to picking a random \( E \).

**Remark 5.1.5.** The above definition can be considered lossless in that there is some \( E \in E \) where the map \( E \) exactly preserves the rank of \( M \). One can imagine a lossy version, where only \( \text{rank}(E \cdot M) \approx \text{rank}(M) \) is required for some \( E \).

**Remark 5.1.6.** The above definition can also considered to be one-sided in that there is some \( E \in E \) has that \( \text{rank}(EM) \) is large enough, in that it equals \( \text{rank}(M) \). As \( \text{rank}(EM) \leq \text{rank}(M) \), there is no concern that \( \text{rank}(EM) \) is too large. That we consider a one-sided notion here is typical for algebraic complexity as there is no sufficiently general notion of distance in general fields, so we simply rely on the difference between zero values and non-zero values.

As this notion is one-sided, one could object to the use of the word “condenser” as that word is a two-sided notion when used in boolean pseudorandomness, so that “disperser” might be a better word to give an analogy to the boolean case. However, we feel that due to the inherently one-sided nature of algebraic complexity, “condenser” is appropriate as there would never be a weaker notion of “disperser” to consider.

Notice how the above definition instantiates dimension reduction. The maps in \( E \) map \( F^n \to F^m \). Further, for every \( V \subseteq F^n \) of rank \( \leq r \) we can find a matrix \( M \in F^{n \times r} \) such that \( \text{col-span}(M) = V \), and \( E \) ensures that some \( E \in E \) has \( \text{rank}(EM) = \text{rank}(M) \). In the language of vector spaces, this means that \( E : F^n \to F^m \) and \( \dim V = \dim E(V) \). Thus, a rank condenser induces dimension reduction on column spaces of matrices. This also somewhat motivates the name, as a rank condenser “condenses” \( n \)-dimensional space to \( m \)-dimensional space, while preserving rank (where the word “condenser” is used in analogy to condensers in boolean pseudorandomness).

**Remark 5.1.7.** Notice as \( \text{rank}(M) \) could be as big as \( r \), and that \( \text{rank}(E \cdot M) \leq m \) by construction, it follows that \( m \geq r \) always. If \( m = r \) then one might more properly call \( E \) an extractor.

In the context of this work, the quantitative difference in output size between an extractor and a condense will not be important. None the less, we will construct an non-explicit rank extractor (Lemma 5.3.3) as well as an explicit rank extractor.
The above discussion focused on dimension reduction of column spaces of matrices. We now note that rank condensers also can be thought of as dimension reduction on row spaces of matrices.

**Lemma 5.1.8.** Let $\mathbb{F}$ be a field, and $n \geq r$. Let $E \subseteq \mathbb{F}^{m \times n}$ be rank condenser for rank $\leq r$. Let $M \in \mathbb{F}^{n \times k}$ be of rank $\leq r$. Then there is some $E \in E$ such that row-span $EM = \text{row-span } M$.

**Proof:** Note that row-span $EM \subseteq \text{row-span } M$ for any $E$, as $EM$ is formed from $M$ via row operations. That there is an $E \in E$ such that rank $EM = \text{rank } M$ implies that row-span $EM = \text{row-span } M$ for that $E$. \hfill $\Box$

Observe now that this result shows how rank condensers give a positive answer to Question 5.1.2. That is, if we take vectors $v_1, \ldots, v_n \in \mathbb{F}^r$ with $n \gg r$, and define $M \in \mathbb{F}^{n \times r}$ such that $M|_{i^*} = v_i$, then row-span $M = \text{span}_\mathbb{F}\{v_i\}_i$. That $E \in E$ has row-span $EM = \text{row-span } M$ means that defining $v_i := (EM)|_{i^*}$ yields $\text{span}_\mathbb{F}\{v_i\}_i = \text{span}_\mathbb{F}\{(EM)|_{i^*}\}_i$. Further, there are only $m$ such $v_i$. Thus, we see that to get a randomness efficient oblivious Gaussian elimination, it suffices to find a rank condenser for rank $\leq r$ with $m \approx r$ and $|E|$ small.

We now further contrast rank condensers to their boolean analogue. In particular, in the boolean case a condenser would typically take the form “if the random variable $\overline{X}$ has min-entropy $\geq k$ then $\text{Cond}(\overline{X}, \overline{y})$ has min-entropy $\gtrsim k$ for most values of the seed $\overline{y}$”. Thus, one might more naturally define a rank condenser as saying that “if a matrix $M \in \mathbb{F}^{n \times k}$ has rank $\geq r$ then there is some $E \in E \subseteq \mathbb{F}^{m \times n}$ so that rank$(EM) \geq r$”. However, we show that this property is exactly equivalent to the definition given.

**Lemma 5.1.9.** Let $\mathbb{F}$ be a field, and $n \geq r$. Then the following are equivalent properties of $E \subseteq \mathbb{F}^{m \times n}$.

1. $E$ is a rank condenser for rank $\leq r$.
2. For any $M \in \mathbb{F}^{n \times k}$ of rank $\geq r$, there exists an $E \in E$ such that rank $EM \geq r$.
3. For any $M \in \mathbb{F}^{n \times r}$ of rank $r$, there exists an $E \in E$ such that rank $EM = r$.

**Proof:** (1) $\implies$ (2): Consider a $M \in \mathbb{F}^{n \times k}$ of rank $\geq r$. Then there is an $S \subseteq [k]$ with $|S| = r$ such that $M|_{\bullet, S}$ has rank $M|_{\bullet, S} = r$. Thus there is some $E \in E$ such that rank$(EM) = \text{rank } M|_{\bullet, S} = r$. Thus rank $EM \geq \text{rank } (EM|_{\bullet, S}) = r$.

(2) $\implies$ (3): Consider a matrix $M \in \mathbb{F}^{n \times r}$ of rank $r$. Then there exists an $E \in E$ such that rank $EM \geq r$. But as rank $EM \leq \text{rank } M = r$, it follows that rank $EM = r$ as desired.

(3) $\implies$ (1): Consider a matrix $M \in \mathbb{F}^{n \times k}$ with rank $s \leq r$. Then there is a set $S \subseteq [k]$ with $|S| = s$ such that $M|_{\bullet, S}$ has rank $s$. Now consider a matrix $N \in \mathbb{F}^{n \times r}$ formed by adding $r - s$ columns to $M|_{\bullet, S}$, so that $N|_{\bullet, S} = M|_{\bullet, S}$. Note that we can create $N$ so that rank $N = r$ by adding columns to $M|_{\bullet, S}$ that are linearly independent.

Now find an $E \in E$ so that rank $EN = \text{rank } N = r$, so that $E \cdot (N|_{\bullet, S})$ has rank $s$. As $E \cdot (N|_{\bullet, S}) = E \cdot (M|_{\bullet, S})$, it follows that $E \cdot (M|_{\bullet, S})$ has rank $s$, so that rank $EM \geq s$. But as rank $EM \leq \text{rank } M = s$, it follows that rank $EM = \text{rank } M = s$ as desired. \hfill $\Box$
5.2 Rank Condensers — A Generator Version

In the previous section we gave a definition of a rank condenser that is the analogue of a hitting set, in that there is a set of matrices such that one matrix is guaranteed to condense rank. However, as argued in Subsection 3.2.2, hitting sets often lack the robustness desired in some applications, and that generators are a more natural perspective. While both notions will actually suffice for the application to black-box PIT for roABPs (Section 6.5), the generator version is particularly clean so we will work with that notion, which we now define.

**Definition 5.2.1.** Let $F$ be a field, and $n \geq r$. A matrix $E(\bar{t}) \in F[\bar{t}]^{n \times m}$ is said to be a *(seeded) rank condenser (generator)* for rank $\leq r$ if for every matrix $M \in F^{m \times k}$ of rank $\leq r$, we have that rank$_F(E(\bar{t}) \cdot M) = \text{rank}_F(M)$.

The rank condenser $E(\bar{t})$ is *(n,m)-explicit* if there is an nm-output algebraic circuit computing $E(\bar{t})$ that can be constructed in $t(n,m)$-time in the unit-cost computation model over $F$.

In the above definition we now have a *single* matrix that is parameterized by the seed $\bar{t}$. That a generator is randomness-efficient then translates into the seed $\bar{t}$ using few variables. Though it is somewhat subtle (for example, linear dependencies over $F(\bar{t})$ may not translate to linear dependencies over $F$, as one might divide by zero when plugging in concrete values for $\bar{t}$), it is not difficult to relate the seed-lengths of the hitting set version and generator version of rank condensers, using similar ideas to that of Subsection 3.2.2. However, because of the above subtlety, we will redo some of the properties establish in the last section for rank-condenser hitting sets, thus establishing them for generators as well.

We begin with again stating how rank condensers can finding small spanning sets for the row-span of tall and skinny matrices.

**Lemma 5.2.2.** Let $F$ be a field. Let $E(\bar{t}) \in F[\bar{t}]^{m \times n}$ and $M \in F^{n \times k}$ be matrices so that rank$_F(E(\bar{t})) M = \text{rank}_F M$. Then row-span$_{F[\bar{t}]} E(\bar{t}) M = \text{row-span}_{F[\bar{t}]} M$.

*Proof:* As $M \in F^{n \times k}$ we can also consider $M$ as a matrix in $F(\bar{t})^{n \times k}$, so that row-span$_{F(\bar{t})} M$ is well-defined. Further, rank$_F M = \text{rank}_{F(\bar{t})} M$, as matrix rank is unchanged by passing to an extension field.

Now note that row-span$_{F(\bar{t})} E(\bar{t}) M \subseteq \text{row-span}_{F(\bar{t})} M$ for any $E(\bar{t})$, as $E(\bar{t}) M$ is formed from $M$ via row operations. By the property of the rank condenser, we have that the rank is preserved, so that rank$_{F(\bar{t})} E(\bar{t}) M = \text{rank}_F M = \text{rank}_{F(\bar{t})} M$. This implies that row-span$_{F(\bar{t})} E(\bar{t}) M = \text{row-span}_{F(\bar{t})} M$ as these vector spaces have the same dimension. \qed

Applying this in particular to rank condenser generators yields the following.

**Corollary 5.2.3.** Let $F$ be a field, and $n \geq r$. Let $E(\bar{t}) \in F[\bar{t}]^{m \times n}$ be rank condenser for rank $\leq r$. Let $M \in F^{n \times k}$ be of rank $\leq r$. Then row-span$_{F(\bar{t})} E(\bar{t}) M = \text{row-span}_{F(\bar{t})} M$.

*Proof:* Note that the above statement is subtle in that one must carefully pick which field ($F$ versus $F(\bar{t})$) the row-span is defined over. That is, while $M$ is defined over $F$, one
must consider its row-span over the larger field \( \mathbb{F}(\bar{t}) \) so that it can be compared with the row-span of \( E(\bar{t})M \).

Similarly, we now get an equivalence for various definitions of generators, as in Lemma 5.1.9, using the same proof. We repeat it here because of the subtleties of moving to the fraction field, and also because we will need this property later (in Theorem 5.4.3).

**Lemma 5.2.4.** Let \( \mathbb{F} \) be a field, and \( n \geq r \). Then the following are equivalent properties of \( E \in \mathbb{F}(\bar{t})^{m \times n} \).

1. \( E(\bar{t}) \) is a rank condenser for rank \( \leq r \).
2. For any \( M \in \mathbb{F}^{n \times k} \) of rank \( \geq r \), \( \text{rank}_{\mathbb{F}(\bar{t})} E(\bar{t})M \geq r \).
3. For any \( M \in \mathbb{F}^{n \times r} \) of rank \( r \), \( \text{rank}_{\mathbb{F}(\bar{t})} E(\bar{t})M = r \).

**Proof:**

1. \( \Rightarrow \) 2: Consider a matrix \( M \in \mathbb{F}^{n \times k} \) of rank \( \geq r \). Then there is an \( S \subseteq [k] \) with \( |S| = r \) such that \( M|_{\bullet , S} \) has rank \( M|_{\bullet , S} = r \). Thus \( \text{rank}_{\mathbb{F}(\bar{t})}(E(\bar{t}) \cdot (M|_{\bullet , S})) = \text{rank}_{\mathbb{F}} M|_{\bullet , S} = r \). Thus \( \text{rank}_{\mathbb{F}(\bar{t})} E(\bar{t})M \geq \text{rank}_{\mathbb{F}(\bar{t})}(E(\bar{t}) \cdot (M|_{\bullet , S})) = r \).

2. \( \Rightarrow \) 3: Consider a matrix \( M \in \mathbb{F}^{n \times r} \) of rank \( r \). Then \( \text{rank}_{\mathbb{F}(\bar{t})} E(\bar{t})M \geq r \). But as \( \text{rank}_{\mathbb{F}(\bar{t})} E(\bar{t})M \leq \text{rank}_{\mathbb{F}(\bar{t})} M = \text{rank}_{\mathbb{F}} M = r \), it follows that \( \text{rank}_{\mathbb{F}(\bar{t})} E(\bar{t})M = r \) as desired.

3. \( \Rightarrow \) 1: Consider a matrix \( M \in \mathbb{F}^{n \times k} \) with rank \( s \leq r \). Then there is a set \( S \subseteq [k] \) with \( |S| = s \) such that \( M|_{\bullet , S} \) has rank \( s \). Now consider a matrix \( N \in \mathbb{F}^{n \times r} \) formed by adding \( r - s \) columns to \( M|_{\bullet , S} \), so that \( N|_{\bullet , S} = M|_{\bullet , S} \). Note that we can create \( N \) so that \( \text{rank}_N N = r \) by adding columns to \( M|_{\bullet , S} \) that are linearly independent.

It follows that \( \text{rank}_{\mathbb{F}(\bar{t})} E(\bar{t})N = \text{rank}_E N = r \), so that \( E(\bar{t}) \cdot (N|_{\bullet , S}) \) has rank \( s \) over \( \mathbb{F}(\bar{t}) \). As \( E(\bar{t}) \cdot (N|_{\bullet , S}) = E(\bar{t}) \cdot (M|_{\bullet , S}) \), it follows that \( E(\bar{t}) \cdot (M|_{\bullet , S}) \) has rank \( s \) over \( \mathbb{F}(\bar{t}) \), so that \( \text{rank}_{\mathbb{F}(\bar{t})} E(\bar{t})M \geq s \). But as \( \text{rank}_{\mathbb{F}(\bar{t})} E(\bar{t})M \leq \text{rank}_{\mathbb{F}(\bar{t})} M = \text{rank}_{\mathbb{F}} M = s \), it follows that \( \text{rank}_{\mathbb{F}(\bar{t})} E(\bar{t})M = \text{rank}_{\mathbb{F}} M = s \) as desired.

\( \square \)

### 5.3 Randomness Efficient Rank Condensers Exist

In this section we establish that rank condenser hitting sets of small size exist over finite fields, and the output size for condensing rank \( \leq r \) matrices will be \( r(n - r) + 1 \), so that this could be considered to be an extractor. When the ambient space is \( n \)-dimensional, and the finite field \( \mathbb{F}_q \) has \( q \geq \text{poly}(n) \), the number of seeds in the hitting set will be \( r(n - r) + 1 \). This non-constructive result will follow from a standard probabilistic method argument. Note that while the general argument is standard, for some of the probability/counting estimates we will need a slightly better-than-naive counting, so we will use some estimates proven in Appendix D.

A constructive result matching this bound will be presented in Theorem 5.4.3.

We begin with understanding the effect of multiplying a full-rank matrix by a random matrix.
Lemma 5.3.1. Let $\mathbb{F}$ be a finite field and $M \in \mathbb{F}^{m \times r}$ be of rank $r$. Let $E$ be a random variable uniformly distributed over matrices in $\mathbb{F}^{m \times n}$. Then $E \cdot M$ is uniformly distributed over $\mathbb{F}^{m \times r}$.

Proof: $m = 1$: As $M$ has rank $r$, there is an $S \subseteq [n]$ with $|S| = r$ so that $M|_{S \times \bullet}$ has rank $r$. Now consider $EM$, so that by block matrix multiplication $EM = E|_S M|_{S \times \bullet} + E|_{[n] \setminus S} M|_{[n] \setminus S \times \bullet}$, where we index $E$ only using a single coordinate since $E$ is a row vector. As $M|_{S \times \bullet}$ has rank $r$, it follows that $\pi \mapsto \pi M|_{S \times \bullet}$ is a bijection on $\mathbb{F}^{1 \times r}$, so that $E|_S M|_{S \times \bullet}$ is uniformly distributed on $\mathbb{F}^{1 \times r}$. Noting that $E|_S$ is independent of $E|_{[n] \setminus S}$, and that the sum of a uniform random variable with an independent arbitrarily-distributed random variable is also uniform, it follows that $E|_{[n] \setminus S} M|_{[n] \setminus S \times \bullet} = EM$ is also uniformly distributed over $\mathbb{F}^{1 \times r}$.

$m > 1$: Note that the rows of $E$ are completely independent random vectors uniformly distributed over $\mathbb{F}^{1 \times n}$. Thus, appealing to the $m = 1$ case, it follows that each row $(EM)|_{i \times \bullet}$ is uniformly distributed over $\mathbb{F}^{1 \times r}$, and as these distributions are independent it follows that $EM$ is uniformly distributed $\mathbb{F}^{m \times r}$. \qed

Given that we will be working with random matrices, we need to understand the probability that they have full rank.

Lemma 5.3.2. Let $n \geq k$. The probability a random matrix in $\mathbb{F}^{n \times k}_q$ has rank $< k$ is

$$\frac{q^{k-1}}{q^n} < \frac{1}{q^{n-k(q-1)}} \leq \frac{1}{q^{r-1}}.$$ 

Proof: Let $M$ be a random variable, uniformly distributed over matrices in $\mathbb{F}^{n \times k}_q$. Then $M$ is full rank iff row-span $M = \mathbb{F}^k_q$ iff (row-span $M$)$^\perp = \{0\}$. Equivalently, $M$ is full rank iff (row-span $M$)$^\perp$ contains no 1-dimensional subspace. There are $(n \choose k)_q = \frac{q^k - 1}{q - 1}$ such subspaces (Lemma D.2.8), and they are parameterized by $\mathbb{P}^{k-1}(\mathbb{F}_q) = (\mathbb{F}^k_q \setminus \{0\})/(\mathbb{F}^k_q \setminus \{0\})$ — that is, non-zero vectors in $\mathbb{F}^k_q$ under equivalence by non-zero scalars.

Thus, $M$ is full rank iff for every $\tau \in \mathbb{P}^{k-1}(\mathbb{F}_q)$ we have $M\tau \neq 0 \in \mathbb{F}^n_q$. As $\tau \in \mathbb{F}^k_q \setminus \{0\}$ (once we choose representatives for $\mathbb{P}^{k-1}(\mathbb{F}_q)$) is non-zero, it is a rank-1 matrix, so in particular, $M\tau$ is uniformly distributed in $\mathbb{F}^n_q$ (Lemma 5.3.1). It follows that

$$\Pr_M[M\tau = 0] = q^{-n}.$$ 

Thus, by a union bound,

$$\Pr[\text{rank } M = k] = \Pr_M[\exists \tau \in \mathbb{P}^{k-1}(\mathbb{F}_q), M\tau = 0]$$

$$\leq |\mathbb{P}^{k-1}(\mathbb{F}_q)| \cdot q^{-n}$$

$$= \frac{q^k - 1}{q - 1} \cdot q^{-n}.$$ 

The second part of the claim simply uses that $k \leq n$. \qed

Unfortunately the above gives poor results for the probability a random matrix in $\mathbb{F}^{k \times k}_q$ is full rank (only with probability $\geq \frac{1}{2^n}$), so we cite the following better result, proven in the appendix.
Lemma (Corollary D.2.7). The probability a random matrix in $\mathbb{F}_2^{k \times k}$ has rank $< k$ is $< \frac{3}{4}$.

We now use these estimates to get the desired non-explicit rank condenser.

Lemma 5.3.3. Let $\mathbb{F}_q$ be the finite field of size $q$, and $n \geq r \geq 1$. For $q \geq 3$, there exists a rank condenser hitting set $\mathcal{E}$ for rank $\leq r$, with $\mathcal{E} \subseteq \mathbb{F}^{r \times n}$ and

$$|\mathcal{E}| \leq \left\lceil \frac{\ln q}{\ln(q - 1)} \cdot r(n - r) + \frac{1}{(q - 1) \ln(q - 1)} \cdot r \right\rceil.$$ 

For $q = 2$, there exists a rank condenser hitting set $\mathcal{E}$ for rank $\leq r$, with $\mathcal{E} \subseteq \mathbb{F}^{r \times n}$ and

$$|\mathcal{E}| \leq \left\lceil \frac{\ln 2}{\ln 4/3} \cdot r(n - r) + \frac{1}{\ln 4/3} \cdot r \right\rceil.$$ 

Proof: We first consider $q > 2$, then obtain the result for $q = 2$.

$q \geq 3$: Let $E_1, \ldots, E_k$ be independently distributed matrices over $\mathbb{F}_q^{r \times n}$. By Lemma 5.1.9, $\mathcal{E} := \{E_1, \ldots, E_k\}$ is a rank condenser for rank $\leq r$ iff for all $M \in \mathbb{F}_q^{n \times r}$ with rank$(M) = r$ it has that some $i \in [k]$ yields rank$(E_i M) = r$. Noticing that rank$(E_i M)$ is invariant to column operations on $M$, it follows that $\mathcal{E}$ being a rank condenser is really a statement about the column spaces of the matrices $M$ as opposed to the matrices themselves. Thus, $\mathcal{E}$ is a rank condenser for rank $\leq r$ iff for all subspaces $V \in \mathbb{F}_q^n$ of dimension $r$, there is some $i \in [k]$ so that the map $E_i : \mathbb{F}_q^n \to \mathbb{F}_q^r$ has dim$E_i(V) = \dim V = r$. We now bound the probability this occurs using the union bound.

Let $V \subseteq \mathbb{F}_q^n$ be a subspace of dimension $r$ that we wish $\mathcal{E}$ to preserve. Instead of working with $V$ directly, we can represent $V$ via the column space of the rank $r$ matrix $M \in \mathbb{F}_q^{n \times r}$. Then as random matrices times full rank matrices are random (Lemma 5.3.2),

$$\Pr_{E_i} [\text{rank}(E_i M) < r] < \frac{1}{q - 1},$$

so as the $\{E_i\}_i$ are independent,

$$\Pr_{\{E_i\}_i} [\forall i, \text{rank}(E_i M) < r] < \frac{1}{(q - 1)^k},$$

and thus by the union bound over all dimension $r$ subspaces of $\mathbb{F}_q^n$, represented by the column space of some matrix $M \in \mathbb{F}_q^{n \times r}$,

$$\Pr_{\{E_i\}_i} [\exists M, \forall i, \text{rank}(E_i M) < r] < \binom{n}{r} \frac{1}{(q - 1)^k},$$

invoking Lemma D.2.13,

$$\leq e^{\frac{r^2}{(q - 1)^k}} \cdot q^{r(n - r)} \frac{1}{(q - 1)^k}.$$
Now take \( k \) so that
\[
\begin{align*}
k & \geq r(n - r) \cdot \frac{\ln q}{\ln(q - 1)} + r \cdot \frac{1}{(q - 1) \ln(q - 1)} \\
& = r(n - r) \cdot \log_{q-1} q + r \cdot \frac{\log_{q-1} e}{q - 1} \\
& = \log_{q-1} q^{r(n-r)} \cdot e^{\frac{r}{q-1}}.
\end{align*}
\]
Thus, plugging this into the above it follows that
\[
\Pr[ E \text{ is a rank condenser for rank } \leq r] = \Pr[ \exists M, \forall i, \text{rank}(E_i M) < r] < e^{\frac{r}{q-1} \cdot q^{r(n-r)}} \cdot \frac{1}{(q - 1)^k} \leq 1.
\]
Thus, for \( q \geq 3 \), the probabilistic method shows that there exists such a rank condenser \( E \subseteq \mathbb{F}_q^{r \times n} \) for rank \( \leq r \), with \( |E| = \lceil r(n - r) \cdot \ln q \cdot q^{r(n-r)} \cdot \log_{q-1} e \cdot \frac{1}{(q-1) \ln(q-1)} \rceil \).

\( q = 2 \): The above strategy is the same, we only need to replace the estimate that an \( r \times r \) matrix is full rank from Lemma 5.3.2 to Corollary D.2.7, so that we replace \( \frac{1}{q-1} \) with \( 3/4 \). Thus, we get,
\[
\Pr[ E \text{ is a rank condenser for rank } \leq r] < e^{\frac{r}{q-1} \cdot 2^{r(n-r)} \cdot (3/4)^k} = 2^{\frac{r}{q-1} \cdot 2^{r(n-r)} \cdot (3/4)^k} = e^r \cdot 2^{r(n-r)} \cdot (3/4)^k.
\]
Thus, taking \( k \) with
\[
\begin{align*}
k & \geq \log_{3/4} 2^{r(n-r)} \cdot e^r \\
& = r(n - r) \cdot \ln 2 \cdot \ln 4/3 + r \cdot \ln e \cdot \ln 4/3 \\
& = \frac{\ln 2}{\ln 4/3} \cdot r(n - r) + r \cdot \frac{1}{\ln 4/3} \cdot r,
\end{align*}
\]
will then yield
\[
\Pr[ E \text{ is a rank condenser for rank } \leq r] < e^r \cdot 2^{r(n-r)} \cdot (3/4)^k \leq 1,
\]
which as above, by the probabilistic method, establishes the desired rank condenser.

We now seek to refine this bound for polynomially-large \( q \), and to do so need the following approximation.
Lemma 5.3.4. For \((q - 1) \ln(q - 1) \geq k\),
\[
\frac{\ln q}{\ln(q - 1)} \leq 1 + \frac{1}{k}.
\]

Proof:
\[
\frac{\ln q}{\ln(q - 1)} \leq 1 + \frac{1}{k},
\]
is equivalent to,
\[
q \leq (q - 1)^{1 + \frac{1}{k}},
\]
which is equivalent to,
\[
1 + \frac{1}{q - 1} = \frac{q}{q - 1} \leq (q - 1)^{1/k},
\]
or equivalently,
\[
\left(1 + \frac{1}{q - 1}\right)^k \leq q - 1.
\]
As \(\left(1 + \frac{1}{q - 1}\right)^k \leq e^{\frac{k}{q - 1}}\), this is implied by
\[
\frac{k}{q - 1} \leq \ln(q - 1),
\]
yielding the claim. \(\square\)

We now conclude with a non-explicit rank condenser for large finite fields.

Lemma 5.3.5. Let \(\mathbb{F}_q\) be the finite field of size \(q\), and \(n \geq r \geq 1\). For \(q \geq 2n^2 + 1\), there exists a rank condenser hitting set \(E\) for rank \(\leq r\), with \(E \subseteq \mathbb{F}^{r \times n}\) and
\[
|E| \leq r(n - r) + 1.
\]

Proof: By Lemma 5.3.3 there exists such a \(E\) with
\[
|E| \leq \left\lceil \frac{\ln q}{\ln(q - 1)} \cdot r(n - r) + \frac{1}{(q - 1)\ln(q - 1)} \cdot r \right\rceil.
\]
As \(q \geq 2n^2 + 1\) it follows from Lemma 5.3.4 that
\[
\frac{\ln q}{\ln(q - 1)} \leq 1 + \frac{1}{2n^2},
\]

78
so that,

\[
\frac{\ln q}{\ln(q-1)} \cdot r(n-r) + \frac{1}{(q-1) \ln(q-1)} \cdot r \\
\leq \left(1 + \frac{1}{2n^2}\right) \cdot r(n-r) + \frac{1}{(q-1) \ln(q-1)} \cdot r \\
= r(n-r) + \frac{1}{2n^2} \cdot r(n-r) + \frac{1}{(q-1) \ln(q-1)} \cdot r \\
\leq r(n-r) + \frac{1}{2n^2} \cdot r(n-r) + \frac{1}{q-1} \cdot r
\]

as \( r \leq n \),

\[
\leq r(n-r) + \frac{1}{2n^2} \cdot n^2 + \frac{1}{q-1} \cdot n
\]

as \( q \geq 2n^2 + 1 \),

\[
\leq r(n-r) + \frac{1}{2n^2} \cdot n^2 + \frac{1}{2n^2} \cdot n
\]

\[
\leq r(n-r) + \frac{1}{2} + \frac{1}{2} = r(n-r) + 1.
\]

Thus there exists such a \( E \) with

\[
|\mathcal{E}| \leq \left[ \frac{\ln q}{\ln(q-1)} \cdot r(n-r) + \frac{1}{(q-1) \ln(q-1)} \cdot r \right] \leq [r(n-r) + 1] = r(n-r) + 1.
\]

\[\Box\]

**Remark 5.3.6.** The above non-explicit construction was slightly more involved than it needed to be, simply because we wanted extremely tight parameters. In particular, there are two places where a simpler argument could have been used.

First, if we relaxed the output size of \( E \) from \( r \) to \( r+1 \), then **Lemma 5.3.2** would imply that for a random \( \mathcal{E} \), \( E \mathcal{M} \) is rank deficient probability \(< \frac{1}{q(q-1)} \leq 1/q \). In particular, this would have meant that we would not need a separate argument for \( q = 2 \), and that there would be no \( \frac{\ln q}{\ln(q-1)} \) term multiplying \( r(n-r) \) in the above. However, as taking the output size to be \( r \) is the minimum possible output size (so that we get an “extractor”) it seems worthwhile to understand this regime.

Second, instead of using the bound of \( e^{\frac{n}{r-1}} q^{r(n-r)} \) on the number of rank \( r \) subspaces of \( \mathbb{F}_q^n \) (**Lemma D.2.13**) we could use the weaker bound \( q^r \) (**Lemma D.2.11**). This would avoid the additive term of \( \approx r/q \) in the above bound. However, as the above result for large \( q \) shows, the number of seeds in a rank condenser should really be \( r(n-r) \) (plus one) as opposed to \( r(n-r) \).

In particular, this is important as our constructive rank condenser (**Theorem 5.4.3**) will achieve this. Perhaps more fundamentally is that \( r(n-r) \) seems to be optimal. In that sense, we believe that morally this construction should be compared to
Reed-Solomon codes (and indeed that construction uses them), in that Reed-Solomon codes meet the Singleton Bound (when they exist over large fields), and morally our construction meets an analogous bound (see Forbes and Shpilka [FS12] for how this construction can be used to construct rank-metric codes meeting the rank-metric Singleton bound).

\[\Box\]

5.4 Explicit Rank Condensers

In the previous section, we saw that rank condenser hitting sets for rank \(\leq r\) exist where the output size is \(r\) (which is optimal), and the number of seeds is \(r(n-r) + 1\) (over large fields). In this section, we meet this bound with an explicit construction from Forbes-Shpilka [FS12].

We first present the Gabizon-Raz lemma ([GR08], Lemma 6.1), stated in the language of this paper. This lemma does yield a rank condenser, but fails to match seed length of the existential result of Lemma 5.3.5.

**Lemma** (Gabizon-Raz ([GR08], Lemma 6.1)). Let \(n \geq r \geq 1\). Define \(E(t) \in F[t]^{r \times r}\) by \((E(t))_{i,j} = t^{ij}\). Then \(E(t)\) is a poly\((n)\)-explicit rank condenser generator for rank \(\leq r\).

Further, for any \(M \in F^{n \times k}\) with rank \(\leq r\), there are \(\leq nr^2\) values \(\alpha \in F\) such that \(\text{rank}(E(\alpha)M) < r\). Thus, for any \(S \subseteq F\) with \(|S| > nr^2\), \(\{E(\alpha)\}_{\alpha \in S}\) is a rank condenser hitting set for rank \(\leq r\). In particular, when \(|F| \geq nr^2 + 1\), there is a poly\((n)\)-explicit rank condenser hitting set for rank \(r\) using \(nr^2 + 1\) seeds when taking the set \(S\) to be poly\((n)\)-explicit and of size \(nr^2 + 1\).

The proof of the above lemma is conceptually similar to what we do below, but our analysis is simpler. This simpler analysis (via the Cauchy-Binet formula) is tied to our construction and does not apply as easily to the above.

We remark here that in the application of rank condensers to black-box PIT of roABPs (Corollary 6.5.17), the above result will qualitatively suffice, but in other applications (such as to constructing rank-metric codes as in Forbes-Shpilka [FS12]) the better construction given below is important.

We begin with noting that a rank condenser generator \(E(\overline{t})\) is inherently a short-and-fat matrix, and the matrix \(M\) to be condensed is a tall-and-skinny matrix. As we will aim to have \(E(\overline{t})\) with \(r\) rows, and \(M\) has \(r\) columns, to show that the rank of \(E(\overline{t})M\) is equal to \(r\) is equivalent to showing that the determinant is non-zero polynomial. To do so, we use the Cauchy-Binet formula, which we cite here and give a proof of in the appendix.

**Lemma 5.4.1** (Cauchy-Binet Formula, Lemma B.3.1). Let \(m \geq n \geq 1\). Let \(M \in F^{n \times m}\), \(N \in F^{m \times n}\). Then,

\[
\det(MN) = \sum_{S \in \binom{[m]}{n}} \det(M|_{\bullet,S}) \det(N|_{S,\bullet}).
\]
While the above allows us to separate \( \det(E(\overline{i} \cup i)M) \) into the \( E(\overline{i}) \) and \( M \) parts, the above sum is of exponentially many terms. As such, it is a-priori difficult to show that it is non-zero. However, in the below we will use a very naive idea to do so. That is, as \( \det(E(\overline{i} \cup i)M) \) is a polynomial in \( \overline{i} \), we can hope to show that there is some term that is *not cancelled out*. In our case, each term in the above sum will be a monomial, so no-cancellation just means that the corresponding monomial appears just once in the summation. If such a monomial can be established, it follows that the summation is a non-zero polynomial in \( \overline{i} \).

The above strategy seems, in general, not a viable approach. However, it turns out that there is a matroid/greedy structure to the above sum, so that uniqueness can be derived. In particular, we use the following uniqueness lemma for optimizing a weight function over linearly independent columns in a matrix.

**Lemma 5.4.2.** Let \( m \geq n \geq 1 \). Let \( M \) be a \( n \times m \) matrix of rank \( n \). Let \( w : [m] \to \mathbb{Z} \) be a weight function on the columns of \( M \), which we extend linearly to subsets of columns via \( w(S) := \sum_{i \in S} w(i) \) for \( S \subseteq [m] \). If \( w \) is injective, then there is a unique set \( S \in \binom{[m]}{n} \) maximizing

\[
\max \{ w(S) \mid \det(M \setminus \bullet, S) \neq 0 \}.
\]

**Proof:** The proof uses the ideas of the Steinitz Exchange Lemma. That is, recall the following facts in linear algebra. If sets \( S_1, S_2 \) are both sets of linearly independent vectors, and \( |S_1| > |S_2| \), then there is some \( \overline{v} \in S_1 \setminus S_2 \) such that the vectors in \( S_2 \cup \{ \overline{v} \} \) are linearly independent. Thus, if \( S_1, S_2 \) are both sets of linearly independent vectors and \( |S_1| = |S_2| \) then for any \( \overline{v} \in S_2 \setminus S_1 \) there is a vector \( \overline{v} \in S_1 \setminus S_2 \) such that \( (S_2 \setminus \{ \overline{w} \}) \cup \{ \overline{v} \} \) is linearly independent.

Now suppose that there are two different sets \( S_1, S_2 \subseteq [m] \) that with \( w(S_1) = w(S_2) \) and \( \det(M \setminus \bullet, S_1), \det(M \setminus \bullet, S_2) \neq 0 \), so that \( |S_1| = |S_2| = n \). We wish to show that they do not have the maximum weight. Consider the (non-empty) symmetric difference \( (S_2 \setminus S_1) \cup (S_1 \setminus S_2) \), and pick \( k \in (S_2 \setminus S_1) \cup (S_1 \setminus S_2) \) that minimizes \( w(k) \). There is a unique such \( k \) as \( w \) is injective. Suppose that \( k \in S_2 \setminus S_1 \), as the case when \( k \in S_1 \setminus S_2 \) is symmetric. By the Steinitz Exchange Lemma, it follows that there is an index \( \ell \in S_1 \setminus S_2 \) such that the columns in \( S_3 := (S_2 \setminus \{ k \}) \cup \{ \ell \} \) are linearly independent, and thus \( \det(M \setminus \bullet, S_3) \neq 0 \) as \( |S_3| = n \) by construction.

By choice of \( k \) and construction of \( \ell \in (S_2 \setminus S_1) \cup (S_1 \setminus S_2), k \neq \ell \) and thus \( w(k) < w(\ell) \) as \( w \) is injective. Thus, \( w(S_3) = w(S_2) + w(\ell) - w(k) > w(S_2) = w(S_1) \). Thus, whenever there are distinct \( n \)-sets of linearly independent columns with the same weight, we can find such a set of larger weight. It follows by contrapositive that there is a unique maximum. \( \square \)

Note that the above result can be stated more generally as a result about matroids, where the above is simply the uniqueness of optimization over sets in a linear matroid (with distinct weights). Further, when the weight function \( w \) is positive, there is a unique maximizer when we simply enforce \( \operatorname{rank}(M \setminus \bullet, S) = |S| \), as enlarging \( S \) by adding linearly independent columns will always increase \( w(S) \). However, in the application below we will only have \( w \) be non-negative, so this situation will not arise.
The above result shows that the above maximization leads to a unique maximum. We now use this to obtain the rank condenser, by showing that the Cauchy-Binet decomposition leads to a sum of terms, each of which is a monomial, so that there is a unique maximum monomial by applying the above result.

We now give the explicit rank condenser. To do so, we will need an element of large multiplicative order which by Lemma A.0.5 can be found efficiently.

**Theorem 5.4.3.** Let \( n \geq r \geq 1 \). Let \( \mathbb{F} \) be a field with \( |\mathbb{F}| > n \). Let \( \omega \in \mathbb{F} \) be an \( \text{poly}(n) \)-explicit element of order \( \geq n \). Define \( E(t) \in \mathbb{F}[[t]]^{[r] \times [n]} \) by \( (E(t))_{ij} = (\omega^t)^j \). Then \( E(t) \) is a \( \text{poly}(n) \)-explicit rank condenser generator for rank \( \leq r \).

Further, for any \( M \in \mathbb{F}^{n \times k} \) with rank \( \leq r \), there are \( \leq r(n-r) \) non-zero values \( \alpha \in \mathbb{F} \) such that \( \text{rank}(E(\alpha)M) < r \). Thus, for any \( S \subseteq \mathbb{F} \setminus \{0\} \) with \( |S| > r(n-r) \), \( \{E(\alpha)\}_{\alpha \in S} \) is a rank condenser hitting set for rank \( \leq r \). In particular, when \( |\mathbb{F}| \geq r(n-r) + 2 \), there is a \( \text{poly}(n) \)-explicit rank condenser hitting set for rank \( r \) using \( r(n-r) + 1 \) seeds when taking the set \( S \) to be \( \text{poly}(n) \)-explicit and of size \( r(n-r) + 1 \).

**Proof:** generator: By Lemma 5.2.4, it follows that proving the generator claim is equivalent to showing that for any \( M \in \mathbb{F}^{n \times r} \) with rank \( r \) that \( \text{rank}_{\mathbb{F}(t)} E(t)M = r \). Equivalently, as \( E(t)M \) is an \( r \times r \) matrix, the claim is equivalent to showing that \( \det E(t)M \) is a non-zero polynomial in \( t \).

To analyze this determinant, we invoke the Cauchy-Binet formula (Lemma B.3.1), so that

\[
\det E(t)M = \sum_{S \subseteq \{1, \ldots, n\}} \det(E(t)|_{\bullet, S}) \det(M|_{S, \bullet}).
\]

For \( S = \{k_1, \ldots, k_r\} \subseteq [n] \),

\[
\det(E(t)|_{\bullet, S}) = \det \begin{bmatrix} (t)^{k_1} & \cdots & (t)^{k_r} \\
(\omega t)^{k_1} & \cdots & (\omega t)^{k_r} \\
\vdots & \ddots & \vdots \\
(\omega^{r-1} t)^{k_1} & \cdots & (\omega^{r-1} t)^{k_r} \end{bmatrix}
\]

\[
= t \sum_{i=1}^r k_i \cdot \det \begin{bmatrix} 1 & \cdots & 1 \\
\omega^{k_1} & \cdots & \omega^{k_r} \\
\vdots & \ddots & \vdots \\
(\omega^{k_1} t)^{r-1} & \cdots & (\omega^{k_r} t)^{r-1} \end{bmatrix}
\]

\[
= t \sum_{i=1}^r k_i \prod_{1 \leq i < j \leq r} (\omega^{k_j} - \omega^{k_i})
\]

By assumption that the order of \( \omega \) is \( \geq n \), so the elements \( (\omega^k)_{k \in [n]} \) are distinct, implying that the above Vandermonde determinant is non-zero. Putting the above more succinctly, we have that

\[
\det(E(t)|_{\bullet, S}) = \text{(non-zero coefficient)} \cdot t \sum_{i=1}^r k_i.
\]

We now show \( \det E(t)M \) is not identically zero, as a polynomial in \( t \). We show this by showing that there is no cancellation of terms at the highest degree of \( \det E(t)M \).
That is, there is a unique set \( S \in \binom{[n]}{r} \) maximizing \( \sum_{k \in S} k \) subject to \( \det(M|_{S, \bullet}) \neq 0 \), as shown by the above lemma. Thus,

\[
\det E(t)M = \sum_{S \in \binom{[n]}{r}} \det(E(t)|_{\bullet, S}) \det(M|_{S, \bullet}) \\
= \sum_{S \in \binom{[n]}{r}} (\text{non-zero coefficient}) \cdot t^{\sum_{i=1}^{r} k_i} \det(M|_{S, \bullet}) \\
= \sum_{S \in \binom{[n]}{r}, \det(M|_{S, \bullet}) \neq 0} (\text{non-zero coefficient}) \cdot t^{\sum_{i=1}^{r} k_i} \cdot (\text{non-zero coefficient})
\]

letting \( S_0 \) maximize \( \sum_{k \in S} k \) subject to \( \det(M|_{S, \bullet}) \neq 0 \),

\[
= (\text{non-zero coefficient}) \cdot t^{\sum_{k \in S_0} k} + (\text{terms of degree } < \sum_{k \in S_0} k) \\
\neq 0
\]

Thus, \( \det E(t)M \neq 0 \) as a polynomial in \( t \), so that \( \operatorname{rank}_{\mathbb{F}(t)} E(t)M = r \) as desired.

**generator explicitness:** This is clear from construction, as \( \omega \) can be found explicitly by Lemma A.0.5.

**hitting set:** Consider the above polynomial \( \det E(t)M \). Notice that the degree of this polynomial is \( \sum_{k \in S_0} k \) for some \( S_0 \in \binom{[n]}{r} \), so that in particular the degree is at most \( \sum_{k=n-r}^{n-1} k = nr - \binom{r+1}{2} \). Further, each term in the Cauchy-Binet expansion of \( \det E(t)M \) takes the form \( ct^{\sum_{i \in |S|} i} \) for \( c \neq 0 \), so that it follows that the minimal degree of any coefficient is at least \( \sum_{i \in |S|} i = \binom{r}{2} \). Thus, \( \det E(t)M = t^{\binom{r}{2}} \cdot f(t) \) for \( 0 \neq f \in \mathbb{F}[t] \) with \( \deg f \leq nr - \binom{r+1}{2} - \binom{r}{2} = nr - r^2 = r(n-r) \).

Note then that for any \( M \in \mathbb{F}^{n \times r} \) with rank \( r \) and \( \alpha \in \mathbb{F} \), \( \operatorname{rank} E(\alpha)M = r \) iff \( \det E(\alpha)M \neq 0 \). Thus, if we exclude \( \alpha = 0 \), then \( \det E(\alpha)M = 0 \) iff \( f(\alpha) = 0 \), but as \( f \) has degree \( \leq r(n-r) \) there can be at most \( r(n-r) \) such \( \alpha \).

It follows that taking any set of non-zero \( S \subseteq \mathbb{F} \) with \( |S| > r(n-r) \) will have that for any such \( M \) some \( \alpha \) will have \( \det E(\alpha)M \neq 0 \) so that \( \operatorname{rank} E(\alpha)M = r \), showing the desired hitting set property. In particular, if \( |\mathbb{F}| \geq r(n-r) + 2 \) then there are at least \( r(n-r) + 1 \) non-zero values to create such an \( S \) explicitly by enumerating \( \mathbb{F} \).

**hitting set explicitness:** This is also clear from construction. \( \square \)

In the above discussion of rank condensers, we focused on the case when the rank bound \( r \) is below the dimension of the vectors \( n \). This is always the case, but in some cases we will have a dimension bound that exceeds \( n \) and will not wish to do case analysis on whether \( r \leq n \) or \( r > n \). Thus, we state the following corollary of the above construction, that gives a rank-condensing property in this regime.

**Corollary 5.4.4.** Let \( n, r \geq 1 \). Let \( \mathbb{F} \) be a field with \( |\mathbb{F}| > n \). Let \( \omega \in \mathbb{F} \) be an \( \text{poly}(n) \)-explicit element of order \( \geq n \). Define \( E(t) \in \mathbb{F}[t]^{[r] \times [n]} \) by \( (E(t))_{i,j} = (\omega^t)^j \). Then \( E(t) \) is \( \text{poly}(n) \)-explicit, and for any \( M \in \mathbb{F}^{n \times k} \) with rank \( \leq r \), \( \operatorname{rank}_{\mathbb{F}(t)} E(t)M = \operatorname{rank}_{\mathbb{F}} M \).
Proof: There are two cases, depending on whether \( r \leq n \).

\( r \leq n \): In this case, the statement is just that \( E(t) \) is a rank condenser for rank \( \leq r \), which is established in the above Theorem 5.4.3.

\( r > n \): In this case, observe that \( E(t) |_{[n]} \) is just the rank condenser for rank \( \leq n \) defined by Theorem 5.4.3, and that \( (E(t) |_{[n]})M \) is a submatrix of \( E(t)M \) so that \( \text{rank}_{F(t)} E(t)M \geq \text{rank}_{F(t)}((E(t) |_{[n]})M) \). As \( \text{rank}_F M \leq n \), it follows that

\[
\text{rank}_F M = \text{rank}_{F(t)} M \geq \text{rank}_{F(t)} E(t)M \geq \text{rank}_{F(t)}((E(t) |_{[n]})M) = \text{rank}_F M,
\]

from which case all inequalities are equalities, yielding the claim. \( \square \)
Chapter 6

Polynomial Identity Testing of roABPs

Abstract: This chapter contains the main result of this thesis: the construction of quasipolynomial-size hitting set for roABPs. For completeness, we will take the long route in explaining the motivation and ideas behind this algorithm. As such, we first explain what polynomial identity testing means in the context of having a vector of polynomials, as opposed to having a single polynomial. In this setting, there is a linear algebraic structure to preserve (as opposed to just non-zeroness), and we call this structure the coefficient span. In exploring the notion of coefficient span, we show that it is simple to merge the coefficient spans of variable disjoint products. This leads naturally into the algorithm by Raz-Shpilka [RS05] for white-box PIT of roABPs, where one uses the additional idea of compressing coefficient spans via Gaussian elimination. To obtain the desired hitting sets from Forbes-Shpilka [FS13b], we study how to compress coefficient span in a way compatible with black-box PIT, and show that the oblivious Gaussian elimination given by the rank condensers of Chapter 5 suffice. By replacing Gaussian elimination with rank condensers, we then obtain the desired hitting sets, once we use the additional idea we call reinterpolation.

6.1 Introduction

In this chapter, we yet again consider the read-once oblivious algebraic branching program (roABP), that is, polynomials computable by

\[ f(x_1, \ldots, x_n) = (M_1(x_1) \cdots M_n(x_n))_{1,1}, \]

where the \( M_i \) are \( w \times w \) matrices with low-degree univariate polynomials as entries. In the previous chapter we gave structural results for roABPs and in particular gave an explicit polynomial that requires \( w \geq \exp(n) \) to be computed as above. In this chapter, we will turn to PIT for this model, in both the white-box and black-box model.
6.1.1 Related work

Boolean Pseudorandomness: This work fits into the research program of derandomizing PIT, in particular derandomizing black-box PIT. However, for many of the models of algebraic circuits studied, there are corresponding boolean circuit models for which derandomization questions can also be asked. In particular, for a class $C$ of boolean circuits, we can seek to construct a pseudorandom generator $G : \{0,1\}^s \to \{0,1\}^n$ for $C$, such that for any circuit $C \in C$ on $n$ inputs, we have the $\epsilon$-closeness of distributions $C(G(U_s)) \approx C(U_n)$, where $U_k$ denotes the uniform distribution on $\{0,1\}^k$. Nisan [Nis92] studied pseudorandom generators for space-bounded computation, and for space $S$ computation gave a generator with seed length $s = \mathcal{O}(\lg^2 S)$. Impagliazzo, Nisan and Wigderson [INW94], and Ganor-Raz [GR13] later gave a different constructions with the same seed length. Randomized space-bounded computation can be modeled with read-once oblivious (boolean) branching programs, and these generators are established by fooling this model of computation.

This chapter studies read-once oblivious (algebraic) branching programs, and we achieve a quasi-polynomial-sized hitting set, corresponding to a seed length of $\mathcal{O}(\lg^2 S)$ for roABPs of size $S$. Aside from the similarities in the statements of these results, there are some similarities in the high-level techniques as well. Indeed, an interpretation by Raz and Reingold [RR99] argues that the [INW94] generator for space-bounded computation can be seen as recursively partitioning the branching program, using an (boolean) extractor to recycle random bits between the partitions. Similarly, in our work, we use an (algebraic rank) extractor between the partitions. However, in reality, our proof technique is closer to that of Nisan [Nis92], as we use the rank condenser as an efficient sampler (replacing Nisan’s [Nis92] use of pairwise independent hash functions).

However, despite the similarities to the boolean regime, our results improve when the branching programs have bounded width. Specifically, our hitting sets (Corollary 6.5.17) achieve a seed length of $\mathcal{O}(\lg^2 S/\lg \lg S)$ in this regime, and it has been a long-standing open problem (see [Vad12, Open Problem 8.6]) to achieve a seed length (for pseudorandom generators, or even hitting sets) of $o(\lg^2 S)$ for boolean read-once oblivious branching programs of constant width. Despite much recent work (see [BDVY13, SZ11, GMR+12, BRRY10, BV10, KNP11, De11, Ste12]), such seed-lengths are only known for branching programs that are restricted even further, such as regular or permutation branching programs. It is an interesting question as to whether any insights of this work can achieve a similar seed length in the boolean regime, or in general whether there are any formal connections between algebraic and boolean pseudorandomness for read-once oblivious branching programs.

Algebraically Natural Proofs: Razborov and Rudich [RR97] defined the notion of a natural proof, and showed that no natural proof can yield strong lower bounds for many interesting boolean models of computation, assuming widely believed conjectures in cryptography about the existence of pseudorandom functions. Further, they showed that this barrier explains the lack of progress in obtaining such lower bounds, as essentially all of the lower bound techniques known are natural in their sense.
In algebraic complexity, there is also a notion of an algebraically natural proof, and essentially all known lower bounds are natural in this sense (see [Aar08] and [SY10, §3.9]). However, there is no formal evidence to date that algebraically natural proofs cannot prove strong lower bounds, as there are no candidate constructions for (algebraic) pseudorandom functions. The main known boolean construction, by Goldreich, Goldwasser and Micali [GGM86], does not work in the algebraic regime as the construction uses repeated function composition, which results in a polynomial of exponential degree and as such does not fit in the framework of algebraic complexity theory, which only studies polynomials of polynomial degree.

In this work, we give limited informal evidence that some variant of the Goldreich-Goldwasser-Micali [GGM86] construction could yield an algebraically natural proofs barrier. That is, Naor (see [Rei13]) observed that the Goldreich, Goldwasser, and Micali [GGM86] construction is superficially similar to Nisan’s [Nis92] pseudorandom generator, in that they both use recursive trees of function composition. In this work, we give an algebraic analogue of Nisan’s [Nis92] boolean pseudorandom generator, and as such use repeated function composition. As in the naive algebraization of the Goldreich-Goldwasser-Micali [GGM86] construction, a naive implementation of our proof strategy would incur a degree blow-up. While the blow-up would only be quasi-polynomial, it would ultimately result in a hitting set of size \( \exp(\lg^3 S) \) for read-once oblivious ABPs of size \( S \), instead of the \( \exp(\lg^2 S) \) that we achieve. To obtain this better result, we introduce an \textit{reinterpolation trick} that allows us to control the degree blow-up, even though we use repeated function composition. While this trick alone does not yield the desired algebraic analogue of the Goldreich-Goldwasser-Micali [GGM86] construction, it potentially removes the primary obstacle to constructing an algebraic analogue.

The Partial Derivative Method: Besides the natural goal of obtaining the most general possible PIT result, the problem of obtaining a black-box version of the algorithm of Raz and Shpilka [RS05] is interesting because of the technique used there. Roughly, the PIT algorithm of Raz-Shpilka [RS05] works for any model of computation that outputs polynomials whose space of partial derivatives is (relatively) low-dimensional. Set-multilinear ABPs, non-commutative ABPs, low-rank tensors, and so called pure algebraic circuits (defined in Nisan and Wigderson [NW96]) are all examples of algebraic models that compute polynomials with that property, and so the algorithm of Raz-Shpilka [RS05] works for them. In some sense, using information on the dimension of partial derivatives of a given polynomial is the most applicable technique when trying to prove lower bounds for algebraic circuits (see e.g., [Nis91, NW96, Raz06, Raz09, RSY08, RY09]) and so it was an interesting problem to understand whether this powerful technique could be carried over to the black-box setting. Prior to this work it was not known how to use this low-dimensional property in order to obtain a small hitting set, and the results of this chapter achieve this goal. Earlier, in Forbes-Shpilka [FS12], we obtained a black-box algorithm for the model of low-rank tensors, but could not prove that it also works for the more general case of set-multilinear ABPs (we still do not know whether this is the case or not — we discuss the
similarities and differences between this prior work and the algorithms of this chapter below), so this work closes a gap in our understanding of such low-dimensional models.

### 6.1.2 Previous and Concurrent Work

**Forbes-Shpilka [FS12]:** In the earlier work Forbes-Shpilka [FS12] we gave a hitting set of quasi-polynomial size for the class of depth-3 set-multilinear polynomials, also known as tensors, which are a subclass of roABPs. That result is mostly subsumed by the hitting sets in this chapter (Corollary 6.5.17), but the previous work has a better dependence on some of the parameters. More importantly, it is interesting to note that the proof in Forbes-Shpilka [FS12] is also based on the same intuitive idea of preserving dimension of linear spaces while reducing the number of variables, but in order to prove the hitting set results in this chapter we had to take a different approach. At a high level the difference can be described as follows. We now summarize the algorithm of Forbes-Shpilka [FS12]. First the case of degree 2 is solved. In this case the tensor is simply a bilinear map. For larger \( D \), the algorithm works by first reducing to the bivariate case using the Kronecker substitution \( x_i \leftarrow x^{a_i} \) for \( i \leq D/2 \) and \( x_{i+D/2} \leftarrow y^{a_i} \) for \( 1 \leq i \leq D/2 \). Appealing now to the bivariate case, we can take \( y \) to be some multiple of \( x \), and then undo the Kronecker substitution (and applying some degree reduction) to recover a tensor on \( D/2 \) variables. If we try to implement this approach with roABPs then we immediately run into problems, as the previous work requires that the layers of the roABP are commutative, in that we can re-order the layers. While this is true for depth-3 set-multilinear computation, it is not true for general roABPs. To generalize the ideas to general roABPs, this work respects the ordering of the layers in the roABP. In particular, while the previous work had a top-down recursion, this work follows a bottom-up recursion. We merge variables in adjacent levels of the roABP and then reduce the degree of the resulting polynomial using an (algebraic rank) extractor. We do this iteratively until we are left with a univariate polynomial. Perhaps surprisingly, this gives a set that is simpler to describe as compared to the hitting set in Forbes-Shpilka [FS12]. On the other hand, if we restrict ourselves to set-multilinear depth-3 circuits then the hitting set of Forbes-Shpilka [FS12] is polynomially smaller than the set that we construct here.

**Agrawal, Saha and Saxena [ASS13]:** Independently and simultaneously with this work, Agrawal, Saha and Saxena [ASS13] obtained results on black-box derandomization of PIT for small-depth set-multilinear formulas, when the partition of the variables into sets is unknown. They obtained a hitting set of size \( \exp((2h^2 \lg(s))^{h+1}) \), for size \( s \) formulas of multiplicative-depth \( h \). We note that their model is both stronger and weaker compared to the models that we consider here. On the one hand, this model does not assume knowledge of the partition of the variables, whereas our model assumes this knowledge. On the other hand, they only handle small-depth formulas, and indeed, the size of their hitting grows doubly-exponentially in the depth, whereas our results handle arbitrary set-multilinear formulas, according to their definition, if we know the partition, and the size of the hitting set grows quasi-polynomially with
the depth\\(^1\).

**Jansen, Qiao and Sarma [JQS09, JQS10]:** We also mention that in Jansen, Qiao and Sarma [JQS09, JQS10] studied black-box PIT in various models related to algebraic branching programs. Essentially, all these models can be cast as a problem of obtaining black-box PIT for read-once oblivious branching programs where each variable appears on a small number of edges. Their result gives a hitting set of size (roughly) \(n^{O(k \lg(k) \lg(n))}\) when \(k\) is an upper bound on the number of edges that a variable can label and the roABP is of polynomial size. In comparison, our hitting sets (Corollary 6.5.17) are of size \(n^{O(\lg n)}\) and works for \(k = \text{poly}(n)\) (as long as the size of the roABP is polynomial in \(n\)). Our techniques are very different from those of Jansen-Qiao-Sarma [JQS09], which follows the proof technique of Shpilka-Volkovich [SV09]. The later paper Jansen-Qiao-Sarma [JQS10] use a (weak) algebraic analogue of the Impagliazzo-Nisan-Wigderson [INW94] generator.

### 6.1.3 This Chapter

In this chapter we use the structural results of Chapter 4 to get polynomial identity testing algorithms for read-once oblivious algebraic branching programs (roABPs). The algorithms will iteratively repeat a merge phase and a compress phase. Initially, the roABP will be split into univariate parts. In the merge phase, variable disjoint parts will be glued together into a univariate computation via a brute-force expansion. The compress phase will then reduce the complexity of this univariate computation. By iteratively repeating these phases, the entire computation will be reduced to a univariate computation, where this univariate computation is of low-degree, so that brute-force methods can then be applied.

In the first section of this chapter, we explain how to understand the decomposition of an roABP into sub-computations. That is, the above scheme will need to break up an roABP into parts and manipulate these parts, all the while preserving the “essential” properties of this roABP. Choosing which properties to preserve is a delicate balancing act. Preserving too many properties of the roABP will make compression impossible. Preserving too few properties (such as just preserving non-zeroness alone) may not provide a strong enough inductive hypothesis for repeated compression to succeed. Thus, the key to these algorithms is understanding what properties to preserve. In this first section, we define the notion of *coefficient span* and establish its basic properties, as well as arguing its importance as the natural notion of what to preserve in an roABP.

In the second section, we give the (parallel) polynomial-time deterministic PIT of Raz and Shpilka [RS05] for roABPs (as rephrased in the language used here), which introduced the above merge-and-compress scheme. We explain how to implement the above merge-and-compress scheme in a white-box manner. The merge step will be

---

\(^1\)Their definition of set-multilinear formulas is actually that of a *pure* formula (see Nisan-Wigderson [NW96]). It is not hard to see that pure formulas form a sub-model of the set-multilinear ABPs. That is, the usual transformation from formulas to ABPs can be done preserving set-multilinearity.
a simple brute-force operation, roughly corresponding to the tensor product. The compress step will be inherently white-box, as it will use Gaussian elimination to prune linearly dependent vectors from a spanning set of vectors. Finally, we argue that this algorithm is the linear algebraic analogue of Savitch’s theorem from complexity theory (which shows that \( \text{NSPACE}(S) \subseteq \text{DSPACE}(S^2) \)), see Remark 6.3.8.

In the third section, we give the quasi-polynomial-size hitting set for roABPs of Forbes and Shpilka [FS13b]. This hitting set takes the merge-and-compress scheme of Raz and Shpilka [RS05] from the white-box regime and makes this idea black-box. The main white-box part of the Raz-Shpilka [RS05] algorithm was Gaussian elimination. The Forbes-Shpilka [FS13b] hitting set replaces Gaussian elimination with a randomness efficient seeded rank condenser, which were defined and constructed in Chapter 5. As the rank condenser uses a random seed, the hitting set will crucially use the merge-and-compress paradigm can be implemented in parallel, so that there are few stages of compression so that few random seeds are needed. There are also further details involved, such as implementing recursion via generators (as opposed to hitting sets) as well as using a degree reduction step so that the composition of generators does not lead to super-polynomial degree. Finally, we draw parallels between this construction and the pseudorandom generator of Nisan [Nis92] for RL (see Remark 6.5.4).

### 6.1.4 Subsequent Work

**Forbes, Saptharishi and Shpilka [FSS14]:** The results of this chapter focus on PIT for roABPs that are in a known variable order. In the white-box model, this is not a restriction, as one can read off the variable order from the roABP in a white-box fashion. However, in the black-box model, one could hope to have hitting sets that work for any variable order. Indeed, the work of Agrawal-Saha-Saxena [ASS13] discussed above succeeded in giving such hitting sets for “constant depth” versions of roABPs (whereas the full model more corresponds to logarithmic depth). To do so, they introduced a new idea called rank concentration, which is a notion of complexity so that if it is small than there are small hitting sets. Further, they showed that one can shift a polynomial \( f(x) \) to \( f(x + \alpha) \) so that \( g(x) := f(x + \alpha) \) has the desired rank concentration for random \( \alpha \). When \( f \) was in the subclass of roABPs they considered, they were able to construction pseudorandom \( \alpha \) with this property.

The paper of Forbes-Saptharishi-Shpilka [FSS14] took this idea further, giving constructions of such pseudorandom \( \alpha \) for (multilinear) roABPs in any variable order, leading to quasipolynomial-size hitting sets for this class. They drew heavily on the merge and recurse paradigm given in this chapter, as well as the ideas of rank condensers from Chapter 5.

We note that this work is particularly interesting as the known pseudorandom generators for read-once boolean branching programs ([Nis92, INW94, GR13]) also need to know the order that the program reads variables. There are some results for this unknown-order boolean setting ([BPW11, IMZ12, RSV13]) but their results are much more limited and do not achieve polylogarithmic seed length for polynomial size branching programs.
6.2 Vector-Valued Polynomials

In this section we consider polynomials whose coefficients are vectors over \( \mathbb{F} \) as opposed to elements of \( \mathbb{F} \) itself. As mentioned in the introduction to this chapter, we do this so we can understand sub-computations of roABPs. That is, in the end we wish to understand whether a roABP computing a single polynomial has this single polynomial being non-zero. However, to understand this single polynomial we need to understand the many intermediate polynomials the roABP computation produces. Thus, it becomes natural for inductive purposes to consider “polynomial identity testing” of several polynomials jointly. From another perspective, a vector of polynomials is then a single polynomial with vector output. In usual PIT we simply seek to preserve non-zeroness, but with vector output we now have possibly more structure to consider. Because of the nature of roABPs (their computation is variable disjoint) it suffices to understand the linear algebraic structure of a vector-valued polynomial, called its coefficient span, which we now work to define.

We first define vector-valued (commutative) polynomials.

**Definition 6.2.1.** Let \( V \) be an \( \mathbb{F} \)-vector space. The space of polynomials in \( \overline{x} \) with coefficients in \( V \), denoted \( V[\overline{x}] \) is the space of formal sums

\[
\overline{f}(\overline{x}) = \sum_{\overline{a}} v_{\overline{a}} \overline{x}^\overline{a},
\]

where \( v_{\overline{a}} \in V \) and only finitely many \( v_{\overline{a}} \neq 0 \).

The (total) degree of \( \overline{f} \) is defined just as with scalar polynomials, so that we denote the total degree as \( \deg \overline{f} \), and \( \deg \overline{f} = \max_{\text{Coeff}_{\overline{x}^\overline{a}}(\overline{f}) \neq 0} \|\overline{a}\|_1 \). The degree with respect to any variable is defined likewise as in the scalar case, as is the notion of individual degree (maximum degree with respect to any individual variable).

It is not hard to see that \( V[\overline{x}] \) itself is an \( \mathbb{F} \)-vector space. However, it is not a ring as \( V \) is not necessarily endowed with a multiplication. When \( R \) is a non-commutative \( \mathbb{F} \)-algebra (so by our conventions (Chapter 1) it is a non-commutative ring, with \( \mathbb{F} \subseteq R \), so that the \( \mathbb{F} \)-vector space structure of \( R \) is compatible with multiplication of \( R \) and \( \mathbb{F} \) commutes with all elements in \( R \)), it follows that the non-commutative polynomial ring \( R[\overline{x}] \) has the same vector space structure as \( V[\overline{x}] \) when \( V = R \). The main examples are \( R = \mathbb{F}^{n \times n} \) and \( R = \mathbb{F}[\overline{y}] \), and we will primarily be interested in the first.

Our principal interest will be vector spaces that are also \( \mathbb{F} \)-algebras, but we will state some results for vector spaces when that level of generality suffices. In particular, the following definition of coefficient span works for polynomials over a vector space.

**Definition 6.2.2.** Let \( V \) be an vector space over \( \mathbb{F} \). Let \( \overline{f} \in V[\overline{x}] \). Define the coefficient span of \( f \), denotes \( \text{coeff-span}_\mathbb{F}(\overline{f}) \), to be

\[
\text{coeff-span}_\mathbb{F}(\overline{f}) := \text{span}_\mathbb{F}\{\text{Coeff}_{\overline{x}^\overline{a}}(\overline{f})\}_{\overline{a}}.
\]

When the field \( \mathbb{F} \) is understood from context, we will simply write “coeff-span(\( \overline{f} \))”.

Note that the polynomial only has a finite number of non-zero coefficients, it
follows that the coefficient span is always finite dimensional (this is trivially true when $V$ itself is finite dimensional).

We now observe that any processes that preserves the coefficient span necessarily preserves non-zeroness.

**Lemma 6.2.3.** Let $V$ be a vector space over $\mathbb{F}$. Let $\overline{f} \in V[\overline{\pi}]$. Then $\text{coeff-span}(\overline{f}) = \{0\}$ iff $\overline{f} = 0$.

*Proof:* The polynomial $\overline{f}$ is zero iff all of its coefficients are zero iff the span of all its coefficients are zero. \hfill $\square$

We now give lemmas relating the coefficient span when we map coefficients in $V$ to coefficients in $W$.

**Lemma 6.2.4.** Let $V, W$ be vector spaces over $\mathbb{F}$. Let $\varphi : V \to W$ be a linear map. Then $\varphi$ induces a linear map $\varphi : V[\overline{\pi}] \to W[\overline{\pi}]$ by

$$\sum_{\overline{\pi}} \overline{v_{\overline{\pi}}} \overline{\pi} \mapsto \sum_{\overline{\pi}} \varphi(\overline{v_{\overline{\pi}}}) \overline{\pi}.$$ 

*Proof:* By definition of $\varphi$ on $V[\overline{\pi}]$,

$$= \sum_{\overline{\pi}} \varphi(\alpha \overline{v_{\overline{\pi}}} + \beta \overline{u_{\overline{\pi}}}) \overline{\pi}$$

as $\varphi$ is linear on $V$,

$$= \sum_{\overline{\pi}} (\alpha \varphi(\overline{v_{\overline{\pi}}}) + \beta \varphi(\overline{u_{\overline{\pi}}})) \overline{\pi}$$

$$= \alpha \sum_{\overline{\pi}} \varphi(\overline{v_{\overline{\pi}}}) \overline{\pi} + \beta \sum_{\overline{\pi}} \varphi(\overline{u_{\overline{\pi}}}) \overline{\pi}$$

$$= \alpha \varphi \left( \sum_{\overline{\pi}} \overline{v_{\overline{\pi}}} \overline{\pi} \right) + \beta \varphi \left( \sum_{\overline{\pi}} \overline{u_{\overline{\pi}}} \overline{\pi} \right) \quad \square$$

Using that maps $V \to W$ lift to $V[\overline{\pi}] \to W[\overline{\pi}]$, we can now show that this map preserves equalities of coefficient span. We first state a simple linear algebraic fact.

**Lemma 6.2.5.** Let $V, W$ be vector spaces over $\mathbb{F}$, and let $\varphi : V \to W$ be a linear map. Suppose $\overline{v} \in \text{span}_F \{\overline{v_i}\}_i$. Then $\varphi(\overline{v}) \in \text{span}_F \{\varphi(\overline{v_i})\}_i$.

*Proof:* That $\overline{v} \in \text{span}_F \{\overline{v_i}\}_i$ means that

$$\overline{v} = \sum_i c_i \overline{v_i},$$

92
for some \( c_i \in F \). As \( \varphi \) is linear, it follows that
\[
\varphi(\mathbf{v}) = \sum_i c_i \varphi(v_i),
\]
so that \( \varphi(\mathbf{v}) \in \text{span}_F \{ \varphi(v_i) \}_i \).

Recalling that coefficient spans are just the span of the coefficients, the following is then immediate.

**Corollary 6.2.6.** Let \( V, W \) be vector spaces over \( F \) and let \( \overline{f} \in V[\overline{x}] \). Let \( \varphi : V \to W \) be a linear map. If \( \mathbf{v}_1, \ldots, \mathbf{v}_k \in F^m \) have \( \text{span}_F \{ \mathbf{v}_i \}_i = \text{coeff-span}(\overline{f}) \), then \( \text{coeff-span}(\varphi(\overline{f})) = \text{span}_F \{ \varphi(\mathbf{v}_i) \}_i \).

In particular, if \( \overline{f}, \overline{g} \in V[\overline{x}] \) have \( \text{coeff-span}(\overline{f}) \) equalling \( \text{coeff-span}(\overline{g}) \), then \( \text{coeff-span}(\varphi(\overline{f})) \) equals \( \text{coeff-span}(\varphi(\overline{g})) \).

The best example of vector-valued polynomials is when \( V = F^m \). However, in this case we can more easily think of such polynomials as a vector of polynomials, as seen by the following lemma.

**Lemma 6.2.7.** As vector spaces over \( F \), \( (F^m)[\overline{x}] \) and \( (F[\overline{x}])^m \) are isomorphic.

**Proof:** Consider the map \( \pi_i : F^m \to F \) that projects \( F^m \) to its \( i \)-th coordinate. As it is linear, it lifts to a linear map \( \pi_i : (F^m)[\overline{x}] \to F[\overline{x}] \) (Lemma 6.2.4). Now consider the linear map \( \pi : (F^m)[\overline{x}] \to (F[\overline{x}])^m \) given by \( \overline{f} \to (\pi_i(\overline{f}))_{i \in [m]} \). This map is injective as any non-zero \( \overline{f} \) has some non-zero coefficient \( \text{Coeff}_{\overline{x}^i}(\overline{f}) \), which has some non-zero coordinate \( i \). It follows then that \( \text{Coeff}_{\overline{x}^i}(\pi_i(f)) \neq 0 \), so that \( (\pi_i(\overline{f}))_i \neq 0 \). Similarly, the map is surjective as \( \pi_i(x^a) \) maps to the \( m \)-tuple of polynomials where the \( i \)-th coordinate contains \( x^a \) and all other coordinates are zero, and these vectors span \( (F[\overline{x}])^m \).

**Alternate Proof:** Recall that for any \( \overline{x}^{\mathbf{v}} \), the map \( \text{Coeff}_{\overline{x}^i} : F[\overline{x}] \to F \) is linear (Lemma B.1.1). It follows that applying this map coordinate-wise to \( (F[\overline{x}])^m \) also yields a linear map. Now consider the map \( (F[\overline{x}])^m \to (F^m)[\overline{x}] \) defined by
\[
\overline{f} \mapsto \sum_{\mathbf{v}} \text{Coeff}_{\overline{x}^i}(\overline{f}) \mathbf{v}^{\overline{x}}. 
\]
It follows that this is a linear map. It is also injective, as any non-zero \( \overline{f} \) will have some non-zero coefficient, thus yielding a non-zero image under the map. It is also surjective, as the image is easily seen to contain all \( \mathbf{v}^{\overline{x}} \) for \( \mathbf{v} \in F^m \) and these span \( (F^m)[\overline{x}] \). Thus, the spaces are isomorphic.

The above isomorphism allows us to switch between \( (F^m)[\overline{x}] \) and \( (F[\overline{x}])^m \) freely. As such, we will write “\( F[\overline{x}]^m \)” to denote both. Similarly, we will write “\( F[\overline{x}]^{m \times m} \)” when the coefficients are actually \( n \times m \) matrices.

The above shows that coefficient-span being zero exactly captures the entire tuple of polynomials being zero. Now we observe that knowing the coefficient-span also determines whether any individual polynomial in the tuple is zero or not. We give two proofs of this fact.
Lemma 6.2.8. Let $\overline{f} \in \mathbb{F}[\overline{x}]^m$. Suppose $\overline{v}_1, \ldots, \overline{v}_k \in \mathbb{F}^m$ have span$_\mathbb{F}\{\overline{v}_i\}_i$ equaling coeff-span($\overline{f}$). Then $(\overline{f})_j = 0$ iff for all $i$, $(\overline{v}_i)_j = 0$.

Proof: Consider the linear map $\pi_j : \mathbb{F}^m \to \mathbb{F}$ that projects $\mathbb{F}^m$ to its $j$-th coordinate, so that by Lemma 6.2.4 this map lifts to a linear map $\pi_j : \mathbb{F}[\overline{x}]^m \to \mathbb{F}[\overline{x}]$. As $\pi_j(\overline{f}) = (\overline{f})_j$, it follows from Corollary 6.2.6 that coeff-span($(\overline{f})_j$) = span$_\mathbb{F}\{\pi_j(\overline{v}_i)\}_i = \text{span}_\mathbb{F}\{(\overline{v}_i)_j\}_i$. In particular, these spans are subspaces of $\mathbb{F}$ as an $\mathbb{F}$-vector space, so are either both $\{0\}$ or both $\mathbb{F}$.

Dual Proof: Taking the dual view, the polynomial $(\overline{f})_j$ is zero iff all coefficients of $\overline{f}$ are vectors with their $j$-th coordinate zero, which equivalent to saying that $\overline{v}_j \in \text{coeff-span}(\overline{f})^\perp$. As span$_\mathbb{F}\{\overline{v}_i\}_i = \text{coeff-span}(\overline{f})$, it follows this is equivalent to saying that $\overline{v}_j \in (\text{span}_\mathbb{F}\{\overline{v}_i\}_i)^\perp$, which is equivalent to $(\overline{v}_i)_j = 0$ for all $i$. □

Remark 6.2.9. Studying the span of such polynomials can be considered a primal view. Other works, such as Kayal [Kay10], Saha-Saptharishi-Saxena [SSS13] and Agrawal-Saha-Saxena [ASS13] focus on the dual view. That is, they studying the linear dependencies of the coefficients, thus formally study the dual space of coeff-span($\overline{f}$).

The above isomorphism between $(\mathbb{F}^m)[\overline{x}]$ and $(\mathbb{F}[\overline{x}])^m$ shows that the dual space is a particularly natural notion. That is, given $\overline{f} \in \mathbb{F}[\overline{x}]^m$,

$$\text{coeff-span}(\overline{f})^\perp = \{\overline{\alpha} \mid \langle \overline{f}, \overline{\alpha} \rangle = 0 \}.$$ 

That is, $\overline{\alpha} \in \text{coeff-span}(\overline{f})^\perp$ are those vectors whose inner product is zero (in $\mathbb{F}$) with every coefficient of $\overline{f}$, so its inner product with $\overline{f}$ is zero (in $\mathbb{F}[\overline{x}]$).

The primal view and the dual view are, of course, equivalent. While it can often be useful to use both views in conjunction, we shall favor the primal view but still use the dual when appropriate (such as the Lemma 6.4.3). ◊

We now observe that the equivalence of coefficients and evaluations given via polynomial interpolation (Proposition B.2.5) extends to vector-valued polynomials.

Corollary 6.2.10. Let $\mathbb{F}$ be a field, let $S_1, \ldots, S_n \subseteq \mathbb{F}$, and denote $S := S_1 \times \cdots \times S_n$. Let $V$ be a $\mathbb{F}$-vector space. Then for $\overline{f} \in V[\overline{x}]$ with deg$_x \overline{f} < |S_1|$, for any subset $T \subseteq \mathbb{F}^n$,

$$\text{span}_\mathbb{F}\{\text{Coeff}_{\overline{x}}(\overline{f})\}_{\overline{x}} \supset \text{span}_\mathbb{F}\{\overline{f}(\overline{\alpha})\}_{\overline{\alpha} \in T},$$

and in particular,

$$\text{span}_\mathbb{F}\{\text{Coeff}_{\overline{x}}(\overline{f})\}_{\overline{x}} = \text{span}_\mathbb{F}\{\overline{f}(\overline{\alpha})\}_{\overline{\alpha} \in S}.$$

Proof: Write $\overline{f}$ out as $\overline{f} = \sum_{\overline{x} \in V} \overline{v}_i \overline{v}^\overline{x}$ where $\overline{v}_i \in V$.

$\supset$: Then $\overline{f}(\overline{\alpha}) = \sum_{\overline{x} \in V} \overline{v}_i \overline{v}^\overline{x} \in \text{span}_\mathbb{F}\{\overline{v}_i\}_{\overline{x}} = \text{span}_\mathbb{F}\{\text{Coeff}_{\overline{x}}(\overline{f})\}_{\overline{x}}$.

$\subset$: For any $\overline{\alpha}$ with $a_i < |S_i|$, polynomial interpolation (in particular, Lemma B.2.6 applied to the monomials $\overline{x}^\overline{b}$ with $b_i < |S_i|$) implies that there are coefficients $(c_{\overline{\alpha}, \overline{b}})_{\overline{\alpha} \in S} \in \mathbb{F}^S$ so that

$$\sum_{\overline{\alpha} \in S} c_{\overline{\alpha}, \overline{b}} \overline{v}^\overline{\alpha} = \begin{cases} 1 & \overline{b} = \overline{\alpha} \\ 0 & \text{else} \end{cases}.$$
It follows that for any $\overline{a}$ with $a_i < |S_i|$ and $f \in V[\overline{x}]$ of the above form, that
\[
\text{span}_F \{ f(\overline{x}) \}_{\overline{x} \in S} \ni \sum_{\overline{x} \in S} c_{\overline{x},\overline{b}} \overline{x} \overline{b}
= \sum_{\overline{x},\overline{b}} c_{\overline{x},\overline{b}} \sum_{\overline{b}} v_{\overline{x}} \overline{b}
= \sum_{\overline{b}} v_{\overline{b}} \sum_{\overline{x} \in S} c_{\overline{x},\overline{b}} \overline{x}
\]
using the above expression producing an indicator function,
\[
\sum_{\overline{b}} v_{\overline{b}} \mathbb{1}(\overline{b})
= v_{\overline{a}} = \text{Coeff}_F(\overline{a})
\]
Thus, $\text{span}_F \{ f(\overline{x}) \}_{\overline{x} \in S}$ contains $\text{Coeff}_F(\overline{a})$ for all $\overline{a}$ with $a_i < |S_i|$. However, as $f$ only has non-zero coefficients of this form, it follows that $\text{span}_F \{ \text{Coeff}_F(\overline{a}) \}_{\overline{a} \in S} \subseteq \text{span}_F \{ f(\overline{x}) \}_{\overline{x} \in S}$ as desired.

We now consider polynomials with coefficients in an $\mathbb{F}$-algebra, so that we can multiply these polynomials and study how the coefficient span interacts with multiplication. We begin with understanding how span of elements in $R$ interact with multiplication, in general.

**Lemma 6.2.11.** Let $R$ be an $\mathbb{F}$-algebra. Let $A, A' \subseteq R$ with $\text{span}_R A = \text{span}_R A'$, and $B, B' \subseteq R$ with $\text{span}_R B = \text{span}_R B'$. Then $\text{span}_R (A \cdot B) = \text{span}_R (\text{span}_R A \cdot \text{span}_R B)$ and thus $\text{span}_R (\text{coeff-span}(A \cdot B)) = \text{span}_R (\text{coeff-span}(A', B'))$ where $\text{coeff-span}(A \cdot B) := \{ A \cdot B \mid A \in A, B \in B \}$.

**Proof:** As $A \cdot B \subseteq \text{span}(A) \cdot \text{span}(B)$ it follows by linearity that $\text{span}(A \cdot B) \subseteq \text{span}(\text{span}(A) \cdot \text{span}(B))$. To show the other direction, consider any $C \in \text{span}(\text{span}(A) \cdot \text{span}(B))$. Thus $C = \sum_i \gamma_i (\sum_j \alpha_{i,j} A_j) (\sum_k \beta_{i,k} B_k)$ for $\gamma_i, \alpha_{i,j}, \beta_{i,k} \in \mathbb{F}$, $A_j \in A$ and $B_k \in B$ (note that even if $A$ or $B$ are infinite sets, the linear span consists of finite linear combinations, so the above summations are finite). As $R$ is an $\mathbb{F}$-algebra, $\mathbb{F} \subseteq R$ and this inclusion is compatible with the vector space structure of $R$ (and by our convention (Chapter 1), $\mathbb{F}$ commutes with all elements in $R$), so that we can rearrange the above expression to $C = \sum_{i,j,k} \gamma_i \alpha_{i,j} \beta_{i,k} A_j B_k \in \text{span}(A \cdot B)$, as desired.

The second claim follows from the first, by observing that $\text{span}(A \cdot B)$ can be rewritten as $\text{span}(\text{span}(A) \cdot \text{span}(B)) = \text{span}(\text{span}(A') \cdot \text{span}(B')) = \text{span}(A' \cdot B')$.

We now specialize the above to polynomials over $R$. To do so, the above lemma needs a product structure. In the context of polynomials this means that we only multiply variable disjoint polynomials.

**Corollary 6.2.12.** Let $R$ be an $\mathbb{F}$-algebra. Let $\overline{f}, \overline{f} \in R[\overline{x}]$ be $R$-valued polynomials with equal coefficient spans, so that $\text{coeff-span}(\overline{f}) = \text{coeff-span}(\overline{f'})$, and similarly let $\overline{g}, \overline{g'} \in R[\overline{y}]$ with $\text{coeff-span}(\overline{g}) = \text{coeff-span}(\overline{g'})$. Then $\text{coeff-span}(\overline{f} \overline{g}) = \text{coeff-span}(\overline{f'} \overline{g'})$. 

95
Proof: Recall that coeff-span($f$) = span$_{\pi}(\text{Coeff}_{\pi}(f))_\pi$. As $\pi$ and $\gamma$ are disjoint sets of variables, it follows that

$$\text{coeff-span}(fg) = \text{span}\{\text{Coeff}_{\pi}(f)\text{Coeff}_{\gamma}(g)\}_{\pi,\gamma}$$

invoking the above Lemma 6.2.11, and the hypothesis that $f \approx f'$ and $g \approx g'$,

$$= \text{span}\{\{\text{Coeff}_{\pi}(f')\}_{\pi} \cdot \{\text{Coeff}_{\gamma}(g')\}_{\pi}\}$$

and applying the same logic in reverse,

$$= \text{coeff-span}(f'g') \quad \square$$

Example 6.2.13. The above lemma only applies variable disjoint products, and otherwise fails in general. For example, consider $R = F^{2\times 2}$. Omitting zero entries for readability, let

$$F(x) := \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \end{bmatrix} x$$

and

$$G(x) := \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \end{bmatrix} x.$$ 

It is then clear that coeff-span $F = \text{coeff-span} G$. Now observe that

$$F(x)^2 = \left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \end{bmatrix} x\right)\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \end{bmatrix} x\right)$$

$$= \begin{bmatrix} 1 \\ 1 \end{bmatrix}^2 + \left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right)x + \begin{bmatrix} 1 \\ 1 \end{bmatrix}^2 x^2$$

$$= \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} x^2.$$ 

However,

$$F(x)G(x) = \left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \end{bmatrix} x\right)\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \end{bmatrix} x\right)$$

$$= \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right)x + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} x^2$$

$$= \begin{bmatrix} 1 \\ 1 \end{bmatrix} x.$$ 

Thus, coeff-span $F(x)^2$ has dimension 2, but coeff-span $F(x)G(x)$ has dimension 1, so
the spans are thus different, despite that coeff-span $F(x) = \text{coeff-span } G(x)$.

\section*{Remark 6.2.14.}

To understand the complexity of vector-valued polynomials, it is helpful to contrast the behavior of coefficient span under variable disjoint multiplication to the behavior of (not necessarily variable disjoint) multiplication of scalar-valued polynomials. That is, if we have scalar-valued polynomials $f(\overline{x})$ and $g(\overline{y})$, Lemma 3.2.19 shows that a generator for $f, g$ separately is also a generator for $f \cdot g$. However, this fact crucially uses that the multiplication of non-zero polynomials is non-zero, which is not true in general $\mathbb{F}$-algebras.

Another perspective for this difficulty is that the coefficient-span of a vector-valued polynomial necessarily contains “several” pieces of information as the span of coefficients typically has dimension $> 1$. To then “merge” two such spans somehow needs to pair coefficient spans in a clever way. The natural manner is to consider the spanning vectors as a product set (as done in Corollary 6.2.12). However, this squares the size of the spanning set. When the dimension of the $\mathbb{F}$-algebra is 1 (that is, the algebra is $\mathbb{F}$) this incurs no penalty. However, in general this can be a large penalty.

Thus, the above results show that the coefficient span of vector-valued polynomials preserves non-zeroness and also behaves well under (variable disjoint) multiplication. In particular, this behavior shows that variable disjoint products can be divided in half, have their underlying polynomials modified while preserving coefficient spans, and then recombined, and the total coefficient span would still be preserved. Note that this is not true of the property of being non-zero.

\section*{6.3 White-Box PIT for roABPs}

In this section we give an exposition of an efficient parallel deterministic algorithm to decide whether a given roABP computes the zero polynomial. This algorithm is due to Raz and Shpilka [RS05], who originally phrased this algorithm as performing white-box PIT for non-commutative ABPs. Here, we will be slightly revisionist and present the algorithm for roABPs, and later (Chapter 7) show how non-commutative ABPs reduce to roABPs and thus essentially recovering the white-box PIT result for non-commutative ABPs. Further, the original algorithm was not presented as a parallel (NC$^3$) algorithm, but was only observed to be so by the later work of Arvind, Joglekar and Srinivasan [AJS09], and this observation requires the use of parallel linear algebra. Arvind, Joglekar and Srinivasan [AJS09] also gave an alternate parallel algorithm, specific to characteristic-zero, for white-box PIT of non-commutative ABPs (within NC$^2$), but we do not discuss their algorithm here.

\section*{Remark 6.3.1.}

We note here that the results of this section can also be (morally) obtained by boolean-izing the roABP model. That is, the Schwartz-Zippel lemma (Lemma B.2.11) shows that a roABP is non-zero iff it is significantly non-zero on a large enough cube of evaluation points. Thus, by replacing the input variables of a width-$w$ roABP with values from this cube, it then follows that the output of the roABP can be computed by a width-$\approx w$ finite state machine (where the width will blow-up as now the computation must store field elements in a bit representation), when the
computation is properly normalized so that the edge weights become probabilities and
the issue of bit length are dealt with.

Once this is done, testing non-zeroness is then a question in boolean derandom-
ization of read-once (oblivious) boolean branching programs. This can be done in
polynomial time using a dynamic-programming argument (the same argument that
shows that BPL \( \subseteq \) P), and at its core is the same as the Raz-Shpilka [RS05] algorithm
from above.

To test non-zeroness, one can also use the boolean PRG of Nisan [Nis92], or
the polytime algorithm of Nisan [Nis94] which showed that the running PRG of
Nisan [Nis92] can be done in polytime in a non-oblivious manner. The PRG approach
does a hitting set for roABPs, but only when the bit-length of the coefficients are
bounded, which is not algebraic in spirit.

We note that just as Nisan [Nis94] showed how the PRG of Nisan [Nis92] can
be enumerated in polytime in a non-oblivious fashion, the hitting sets presented in
Section 6.5 can be made poly-size if one is granted white-box access to the roABP.
However, doing so is only more complicated than the whitebox algorithm presented
above.

We first give an intuitive explanation of the algorithm. By the structural results
of Section 4.4 a polynomial is computed by a roABP of width \( w \) iff it is an entry in
the product of \( w \times w \) univariate matrices. Notice that if we consider sub-products of
this matrix product, they retain the same structure as products of \( w \times w \) univariate
matrices. Thus, a natural approach is to use divide and conqueror to split this
product into sub-products, somehow compress these sub-products, and then recombine
them. As argued in the introduction to this chapter, it is then crucial to decide what
properties the compression of the sub-products will retain. The entirety of the last
section was devoted to the coefficient span, and indeed that is the property that will
be used here, where we work over the \( F \)-algebra of matrices. Thus, the algorithm will
take a roABP, divide it into left and right halves, compute a smaller roABP for each
halve by recursion, recombine, and then compress the new roABP.

As a basic example, consider a bivariate roABP \( M(x)N(y) \in F[x, y]^{w \times w} \), where
\( M \) and \( N \) are \( w \times w \) and degree \( < d \). We wish to reduce this roABP to a univariate
roABP, while preserving the matrix span. A trivial first idea for doing so is to use the
Kronecker substitution, so that \( P(z) := M(z)N(z^d) \). This does preserve the coefficient
span as this map induces a bijection between monomials so that no coefficients are
lost. However, this has not simplified the roABP in any way, as while there is now
only a single variable, the degree of the roABP has squared.

Thus, the approach of Raz and Shpilka [RS05] is to use Gaussian elimination to
compress roABPs. That is, while the coefficient span of \( M(x)N(y) \) is by definition
the span of \( \leq d^2 \) matrices (the number of coefficients), each of these matrices is
\( w \times w \) and thus live in a \( w^2 \) dimensional space. Thus, when \( w \ll d \) the roABP
has redundancy that can be removed by using Gaussian elimination to prune the
coefficients to a basis of the coefficient space. This would yield \( \leq w^2 \) coefficients. By
packing these coefficients into a new polynomial \( P(z) \), one obtains that \( \deg_z P(z) < w^2 \)
and coeff-span \( P(z) = \text{coeff-span } M(x)N(y) \). Thus, this process turned a bivariate

98
roABP into a degree < \( w^2 \) univariate roABP, while preserving the coefficient span.

More generally, the algorithm will in a single stage halve the number of variables while preserving the coefficient span, and the individual degree will always be < \( w^2 \). This will be done by iteratively applying the above bivariate analysis, observing that each compression can be done independently as coefficient span composes nicely for variable disjoint products (Corollary 6.2.12). At the end of the algorithm, there will be a low-degree univariate matrix whose coefficient span equals the coefficient span of the original roABP, and this can be used to read-off whether the desired polynomial is non-zero (Lemma 6.2.8).

Expanding out these \( \leq d^2 \) matrices can be seen as a brute-force merging step of two variables to one variables, and pruning down to \( w^2 \) matrices can be seen as a compression step. The pruning, via Gaussian elimination, is inherently white-box. In a later section (Section 6.5) we will see how to replace Gaussian elimination with a seeded rank condenser which will allow for black-box pruning (at the expense of yielding a quasi-polynomial-time algorithm).

We now give the pseudocode of the algorithm. To best highlight the recursive structure of the algorithm, we consider an roABP on \( 2^n \) variables, and index the variables by bitstrings in \( \{0, 1\}^n \), so that the roABP will read the variables in lexicographic order. Upon halving the number of variables, we will then index the variables by \( \{0, 1\}^{n-1} \), and so on. In particular, for \( \sigma \in \{0, 1\}^{n-1} \), the variable \( x_{\sigma} \) will be the result of merging the two variables \( x_{\sigma 0} \) and \( x_{\sigma 1} \), which can be thought of as the children of \( \sigma \) in the complete binary tree.

**Algorithm 1** Raz-Shpilka [RS05] algorithm for computing coefficient span of roABP

1: **procedure** get-span

2: \( N = 2^n, w, d, \{M_{\sigma}(x_{\tau})\}_{\sigma \in \{0, 1\}^n}, M_{\sigma}(x_{\tau}) \in \mathbb{F}[x_{\tau}]^{w \times w} \) has deg < \( d \)

3: for \( k = n - 1, \ldots, 0 \) do

4: for \( \sigma \in \{0, 1\}^k \) do

5: Compute \( M_{\sigma} := \{\text{Coeff}_{x_{\tau 0}}(M_{\sigma 0}) \text{Coeff}_{x_{\tau 1}}(M_{\sigma 1})\}_{i, j} \subseteq \mathbb{F}^{w \times w} \)

6: Compute a spanning set \( \{M_{\sigma,i}\}_{i \in [w^2]} \subseteq M_{\sigma} \) for \( M_{\sigma} \)

7: Define \( M_{\sigma}(x_{\sigma}) := \sum_{i \in [w^2]} M_{\sigma,i} x_{\sigma}^i \)

8: end for

9: return \( \{\text{Coeff}_{x_{\lambda}}(M_{\lambda})\}_{i \in [w^2]} \) where \( \lambda \) is the empty string

10: **end procedure**

To establish the runtime properties of the above algorithm, we need some facts from the literature about parallel linear algebra. We first review the fact that matrix multiplication is parallelizable.

**Lemma 6.3.2.** Given \( M \in \mathbb{F}^{m \times n} \) and \( N \in \mathbb{F}^{n \times p} \), the product \( MN \in \mathbb{F}^{m \times p} \) can be computed in \( \text{NC}^1 \), in the unit cost model over \( \mathbb{F} \).

**Proof:** The \((i, j)\)-th entry of \( AB \) is expressible as \( (MN)_{i,j} = \sum_{k \in [m]} M_{i,k} B_{k,j} \). Each such product can be computed by a single processor, and then the summation can be done in a binary tree using \( \mathcal{O}(nmp) \) processors in \( \mathcal{O}(\lg(nmp)) \) steps. \( \square \)
We now cite a theorem about parallel linear algebra, showing that one can compute a basis of a set of vectors quickly.

**Theorem 6.3.3** (Gaussian Elimination is in $\mathbf{NC}^2$, from Borodin, von zur Gathen, and Hopcroft [BvzGH82], as derandomized by Mulmuley [Mul87]). Let $\overline{v}_1, \ldots, \overline{v}_m \in \mathbb{F}^n$. Then in $\mathbf{NC}^2$ ($O(\log^2(nm))$ parallel time, and $\text{poly}(n,m)$ processors) one can compute $i_1, \ldots, i_k \in [m]$ with $k \leq n$ such that \{\overline{v}_{i_j}\}_{j \in [k]}$ are a basis for $\text{span}_\mathbb{F}\{\overline{v}_i\}_{i \in [m]}$.

Furthermore, given a matrix $M \in \mathbb{F}^{n \times m}$ and a vector $\overline{b} \in \mathbb{F}^n$, one can determine in $\mathbf{NC}^2$ whether $M \overline{x} = \overline{b}$ has a solution in $\mathbb{F}^m$, and if so, output a basis of the affine subspace of solutions.

We now establish a series of lemmas to show the correctness and efficiency of the above algorithm. We begin with showing that the polynomials defined in the above algorithm all remain low degree.

**Lemma 6.3.4.** Assume the setup of Algorithm 1. Then for $\sigma \in \{0,1\}^n$, $\deg M_{\sigma} < d$ and for $\sigma \in \{0,1\}^k$ for $k \in [n]$, $\deg M_{\sigma} < w^2$.

*Proof:* Note that for $\sigma \in \{0,1\}^n$ this degree bound is what is promised as input. For $\sigma \in \{0,1\}^k$ this is by construction, as $M_{\sigma}$ is constructed to have degree $w^2 - 1$ in Line (6).

We now show that the above algorithm is efficient, and even runs in parallel.

**Lemma 6.3.5.** Algorithm 1 runs in $\mathbf{NC}^3$ (that is, in $O(\log^2(Nwd))$ parallel time, and $\text{poly}(N,w,d)$ processors) in the unit cost model over $\mathbb{F}$.

*Proof:* Line (4): This line takes $\mathbf{NC}^1$ time, as each matrix multiplication is done separately and each matrix multiplication can be done in $\mathbf{NC}^1$ (Lemma 6.3.2). Note that the number of multiplications to be done is the product of the degrees of the matrix-valued polynomials. When $k = n - 1$ this product is at most $d^2$, and when $k < n - 1$ this product is at most $w^4$, as seen from Lemma 6.3.4.

Line (5): By Theorem 6.3.3 we can prune the set of matrices $M_{\sigma}$ to a basis within $\mathbf{NC}^2$. By padding this basis with redundant vectors, we can then construct a spanning set for $M_{\sigma}$ of exactly $w^2$ matrices. Thus, the entire step is computable in $\mathbf{NC}^2$.

Line (6): This step is just memory management, so runs in $\mathbf{NC}^1$.

Line (9): This is also just memory management, so runs in $\mathbf{NC}^1$.

Line (3): Each run of this loop is completely independent, so as each run is $\mathbf{NC}^2$ and there are $\text{poly}(N)$ such steps, the total work is still $\mathbf{NC}^2$.

Line (2): Each run of this loop is $\mathbf{NC}^2$ work, and there are $n = \lg N$ such runs, so the total work of the loop is $\mathbf{NC}^3$.

As the above lines comprise the entire work of the algorithm, the entire algorithm is $\mathbf{NC}^3$.

We now establish that the above algorithm correctly computes the coefficient span of a roABP.
Lemma 6.3.6. Let $F(\overline{x}) = \prod_{\sigma \in \{0,1\}^n} M_\sigma(x_\sigma)$, for $M_\sigma(x_\sigma) \in \mathbb{F}[x_\sigma]^{w \times w}$ of degree $< d$, where the above product is in lexicographic order (following Chapter 1).

Then, Algorithm 1 on input $(N = 2^n, w, d, \{M_\sigma(x_\sigma)\}_{\sigma \in \{0,1\}^n})$ will have

1. For $0 \leq k \leq n$, coeff-span $\prod_{\sigma \in \{0,1\}^k} M_\sigma(x_\sigma) = \text{coeff-span} \prod_{\sigma \in \{0,1\}^k} M_\sigma(x_\sigma)$.

2. The output will be $w^2$ matrices $\{M_{\lambda,i}\}_{i \in [w^2]}$ capturing the coefficient span of $F(\overline{x})$, so that coeff-span $F(\overline{x}) = \text{span}_\mathbb{F}\{M_{\lambda,i}\}_{i \in [w^2]}$.

Proof: (1): By induction on $k$, from $k = n$ down to $k = 0$.

$k < n$: Note that the output of Algorithm 1 is just $\{\text{Coeff}_{x^\sigma_{\pi_0}}(M_{\sigma_0})\text{Coeff}_{x^\sigma_{\pi_1}}(M_{\sigma_1})\}_{i,j}$. Thus, it follows that
\[
\text{coeff-span } M_\sigma(x_\sigma) = \text{coeff-span } M_{\sigma_0}(x_{\sigma_0})M_{\sigma_1}(x_{\sigma_1}).
\]

Thus,
\[
\text{coeff-span } \prod_{\sigma \in \{0,1\}^{k+1}} M_\sigma(x_\sigma) = \text{coeff-span } \prod_{\sigma \in \{0,1\}^k} M_{\sigma_0}(x_{\sigma_0})M_{\sigma_1}(x_{\sigma_1}),
\]
so that using the above property of $M_\sigma(x_\sigma)$ and appealing to Corollary 6.2.12,
\[
= \text{coeff-span } \prod_{\sigma \in \{0,1\}^k} M_\sigma(x_\sigma).
\]

(2): Note that the output of Algorithm 1 is just $\{\text{Coeff}_{x^\sigma_{\pi_0}}(M_{\sigma_0})\text{Coeff}_{x^\sigma_{\pi_1}}(M_{\sigma_1})\}_{i,j}$, the list of coefficients of $M_\lambda$, which by definition is the spanning set of $\prod_{\sigma \in \{0,1\}^n} M_\sigma(x_\sigma)$. By (1), these coefficients have the same span as coeff-span $F(\overline{x})$, as $F(\overline{x})$ equals coeff-span $\prod_{\sigma \in \{0,1\}^n} M_\sigma(x_\sigma)$. \hfill \Box

We now put this together to get the desire white-box PIT algorithm for roABPs.

Theorem 6.3.7 (Raz-Shpilka [RS05]). Let $\mathbb{F}$ be a field. There is an efficient deterministic parallel (NC$^3$, so $\text{poly}(n, w, d)$ operations in $O(\log^3 (nwd))$ parallel time) algorithm for white-box PIT of $n$-variate, width-$w$ roABPs of individual degree $< d$, in the unit cost model over $\mathbb{F}$.

Proof: Given a width-$w$ roABP of individual degree $< d$, computing a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$, one can in parallel construct matrices $M_i(x_i) \in \mathbb{F}[x_i]^{w \times w}$ such that $f(\overline{x}) = (\prod_i M_i(x_i))_{1,1}$ by Corollary 4.4.2. By introducing new variables $x_{n+1}, \ldots, x_{2^{\lceil \log n \rceil}}$ and taking $M_i(x_i) := I_w$ for $i \in \{n+1, \ldots, 2^{\lceil \log n \rceil}\}$, it follows that $f(\overline{x}) = (\prod_{i \in [2^{\lceil \log n \rceil}]} M_i(x_i))_{1,1}$, so that $f$ is computable by an $N$-variate roABP where $N = 2^{\lceil \log n \rceil}$ is a power of two, and this construction is within the needed time bounds.

Then, by using Algorithm 1 we can compute in NC$^3$ a set of $w^2$ matrices $\{M_i\}_{i \in [w^2]} \subseteq \mathbb{F}^{w \times w}$ such that $\text{span}_\mathbb{F}\{M_i\}_{i=1} = \text{coeff-span} \prod_{i \in [2^{\lceil \log n \rceil}]} M_i(x_i)$. We can then return that $f$ is non-zero iff some $(M_i)_{1,1}$ is non-zero. This is clearly efficient, and correctness follows from Lemma 6.2.8. \hfill \Box
Remark 6.3.8. We now briefly discuss the similarity of the Raz-Shpilka [RS05] algorithm to algorithms from boolean complexity. The most similar algorithmic paradigm is that of Savitch’s theorem (see any standard complexity textbook, such as Sipser [Sip06]) which shows that $\text{NSPACE}(S) \subseteq \text{DSPACE}(S^2)$, or similarly, any sort of repeated squaring of matrices. That is, Savitch’s theorem seeks to solve a directed $s - t$ connectivity problem on a directed layered graph with width $w$ and depth $d$, where all of the layers have the same adjacency matrix. Note that the assumption that the adjacency matrix is the same throughout the layers is essentially without loss of generality, as can be seen via a result similar to Lemma 4.3.2 but using paths instead of cycles.

Whether there is such an $s - t$-path can be phrased as asking whether a certain adjacency matrix $A \in \{0, 1\}^{w \times w}$ has a certain entry in the power $A^d \in \mathbb{Z}^{w \times w}$ being non-zero. While the obvious algorithm to compute $A^d$ would take $d$ multiplications, one can be more clever and use repeated squaring so that only $\approx \lg d$ multiplications are needed.

The $\approx \lg d$ squarings thus correspond to the $\lg d$ number of rounds of the outer loop in the Raz-Shpilka [RS05] algorithm. That each round in the Raz-Shpilka [RS05] algorithm compresses the matrices to only preserve their coefficient span is analogous to truncating the integer matrix $A^d$ after each squaring to be in boolean again by replacing any non-zero value with 1. This operation thus results in only determining whether there is a $s - t$-path as opposed to returning the number of such paths. While doing this truncation/compression is not needed in the boolean case, it is essential in the algebraic problem as returning the entire “number” of paths would correspond to returning the original roABP. Rather, we only want to decide whether there “is an algebraic $s - t$-path”, so we follow the boolean algorithm and truncate between repeated squarings.

6.4 PIT of Vector-Valued Polynomials

In the previous sections we introduced vector-valued polynomials and discussed their coefficient span. Here, we ask the obvious computational polynomial question that seeks to generalize (black-box) polynomial identity testing to this regime: given an algebraic circuit with multiple outputs, what is the coefficient-span? Note that we now seek to determine the entire coefficient span (as opposed to simply testing non-zeroness). This is a natural generalization of polynomial identity testing of a single polynomial (where we get “zero” or “non-zero”), as for a single polynomial the coefficient span is just a subspace of $\mathbb{F}$, which is just either $\{0\}$ or $\mathbb{F}$, and PIT determines which of these two subspaces is the case.

Here, we will introduce a solution hitting set concept for coefficient span and then give a non-explicit construction. We then observe the deficiency in this concept, propose an alternate concept. We then observe that this alternate concept is equivalent to a rank condenser that is compatible with polynomial evaluation. Finally, we observe that the explicit rank condenser of Theorem 5.4.3 is compatible with (univariate) polynomial evaluation, thus gives the desired explicit construction for hitting coefficient span.
We now provide the first solution concept for this question, based on the notion of a hitting set.

**Definition 6.4.1.** Let $V$ be a vector space over $\mathbb{F}$. Let $C \subseteq V[x_1, \ldots, x_n]$ be a set of polynomials. A multi-set $H \subseteq \mathbb{F}^n$ is a hitting set for the coefficient span of $C$ if for all $f \in C$,

$$\text{coeff-span}(f) = \text{span}_\mathbb{F}\{f(\alpha)\}_{\alpha \in H}.$$  

Note that when $V = \mathbb{F}$ we recover the usual definition of a hitting set. That is, for $f \in \mathbb{F}[x]$, $\text{coeff-span}(f) = \{0\}$ iff $f = 0$ (Lemma 6.2.3), and $\text{span}_\mathbb{F}\{f(\alpha)\}_{\alpha \in H} = 0$ iff $f|_H = 0$.

One can easily get explicit hitting sets for $V[\bar{x}]$ if one allows a large size, just as with scalar-valued polynomials (Lemma 3.2.12).

**Lemma 6.4.2.** Let $\mathbb{F}$ be a field with $|\mathbb{F}| \geq d$. Let $V$ be a vector space over $\mathbb{F}$. Let $C \subseteq V[x_1, \ldots, x_n]$ be a set of polynomials of degree $< d$. Then there is $\text{poly}(n, d)$-explicit hitting set for the coefficient span of polynomials in $C$ of size $d^n$.

**Proof:** Let $S \subseteq \mathbb{F}$ be an explicit subset of size $d$, computable in time $\text{poly}(d)$. Define the hitting set as $H := S^n$.

- **correctness:** By the equivalence of coefficient span and evaluation span (Corollary 6.2.10), and appealing to total degree as opposed to individual degree.

- **explicitness:** Indexing into $H$ takes $\text{poly}(n, \lg d)$ time, and computing the respective element of $S^n$ take $\text{poly}(n, d)$ steps, as $S$ is explicit. 

We now derive existential results for hitting sets for coefficient span, along the lines of the analogous results for scalar-valued polynomials (Lemma 3.2.13). As this result will have a dependence on the dimension, we explicitly take $V = \mathbb{F}^k$.

**Lemma 6.4.3.** Let $\mathbb{F}$ be a finite field with $|\mathbb{F}| \geq (1 + \epsilon)d$. Let $C \subseteq \mathbb{F}[x_1, \ldots, x_n]^k$ be a finite set of polynomials of degree $< d$. Then there is a non-explicit hitting set for the coefficient span of $C$ of size $\leq \lfloor \log_{1+\epsilon} |C| + k \log_{1+\epsilon} |\mathbb{F}| \rfloor$.

**Proof:** Recall that we can treat polynomials in $C$ as a vector of $k$ polynomials (Lemma 6.2.7). By the relation between coefficient span and evaluation span (Corollary 6.2.10), for any set $H$ it will be that the evaluation span of $\bar{f} \in C$ on $H$ is contained in the coefficient span of $\bar{f}$, that is,

$$\text{span}_\mathbb{F}\{\bar{f}(\pi)\}_{\pi \in H} \subseteq \text{coeff-span}(\bar{f}).$$

By examining the dual spaces, it follows that this containment is equality iff

$$(\text{span}_\mathbb{F}\{\bar{f}(\pi)\}^\perp)_{\pi \in H} = \text{coeff-span}(\bar{f})^\perp.$$ 

That is, for every $\tau \in \mathbb{F}^k$ so that

$$\langle \tau, \bar{f} \rangle \neq 0 \iff \{\langle \tau, \bar{f}(\pi) \rangle\}_{\pi \in H} \neq 0.$$
Equivalently, \( H \) is a hitting set for the coefficient span of \( C \) iff \( H \) is a hitting set for the class of polynomials \( C' := \{ (\tau, \overline{f}) \mid \tau \in \mathbb{F}^k, \overline{f} \in C \} \). As \( |C'| \leq |\mathbb{F}|^k \cdot |C| \), appealing to existential results for hitting sets of scalar-valued polynomials (Lemma 3.2.13) yields the claim.

**Remark 6.4.4.** Note that the above uses the dual view of coefficient span, and it is not clear how to get such a proof by relying on the primal view alone.

Notice that the above result shows that hitting sets for coefficient spans existentially have a fairly mild \( k \) on the dimension of the vector space. However, there is something definitively unsatisfying about the above solution concept. That is, the coefficient span of \( \overline{f} \in \mathbb{F}[\overline{x}]^k \) is only a space of dimension \( \leq k \) but the above construction gives a set of size \( \gg k \) that spans this space. Thus, to isolate some \( m \) points so that \( \overline{f} \) on these points spans the coefficient span would require examining the entire hitting set. Thus, we will seek the following stronger solution concept.

**Definition 6.4.5.** Let \( C \subseteq \mathbb{F}[x_1, \ldots, x_n]^k \) be a set of vector-valued polynomials. A multi-set of \( m \)-tuples \( H \subseteq (\mathbb{F}^m)^m \) is a \( m \)-sample hitting set for the coefficient span of \( C \) if for all \( \overline{f} \in C \),

\[
\text{coeff-span}(\overline{f}) = \text{span}_\mathbb{F}\{\overline{f}(\overline{\alpha}_1), \ldots, \overline{f}(\overline{\alpha}_k)\}
\]

for some \((\overline{\alpha}_1, \ldots, \overline{\alpha}_m) \in H\).

That is, a \( m \)-sample hitting set for coefficient span gives a list of \( m \)-tuples such that for any \( \overline{f} \in C \), some single \( m \)-tuple of evaluations will capture the coefficient span of \( \overline{f} \). Clearly \( m \geq \text{dim coeff-span}(\overline{f}) \) in the above, and because \( \text{dim coeff-span}(\overline{f}) \leq k \), one can hope to take \( k = m \). Getting such a small number of samples that entirely capture the coefficient span will be quite important for the black-box PIT for roABPs in Section 6.5.

We now show that this indeed possible, in particular by converting hitting sets for coefficient span into \( k \)-sample hitting sets.

**Lemma 6.4.6.** Let \( C \subseteq \mathbb{F}[x_1, \ldots, x_n]^k \) be a set of vector-valued polynomials. If \( H \) is a hitting set for the coefficient span of \( C \), then \( H^k \) is a \( k \)-sample hitting set for the coefficient span of \( C \).

**Proof:** Let \( \overline{f} \in C \). By the hypothesis on \( H \),

\[
\text{coeff-span}(\overline{f}) = \text{span}_\mathbb{F}\{\overline{f}(\overline{\alpha})\}_{\overline{\alpha} \in H}.
\]

As \( \text{coeff-span}(\overline{f}) \) has dimension \( \leq k \), there are some \( \overline{\alpha}_1, \ldots, \overline{\alpha}_k \in H \) so that

\[
\text{coeff-span}(\overline{f}) = \text{span}_\mathbb{F}\{\overline{f}(\overline{\alpha}_1), \ldots, \overline{f}(\overline{\alpha}_k)\}.
\]

As \((\overline{\alpha}_1, \ldots, \overline{\alpha}_k) \in H^k\), this gives the desired property. \( \square \)

Note that this lemma highlights the difference between the strong and weak notions of a hitting set for coefficient span, as the conversion from weak to strong significantly increases the size of the set. Put another way, the above conversion does not give a
randomness efficient hitting set for coefficient span. We now observe that \( m \)-sample hitting sets for coefficient span (for the class of all low-degree polynomials) are rank condensers of a specific form, so that we can hope to obtain randomness efficient hitting sets via randomness efficient rank condensers.

**Lemma 6.4.7.** Let \( \mathcal{C} \subseteq \mathbb{F}[x_1, \ldots, x_n] \) be the class of \( n \)-variate degree \( \leq d \) polynomials with coefficients in \( \mathbb{F}^k \). Then a multi-set of \( m \)-tuples \( \mathcal{H} \subseteq (\mathbb{F}^n)^m \) is a \( m \)-sample hitting set for the coefficient span of \( \mathcal{C}^k \) (\( k \)-tuples of such polynomials) iff the set of matrices \( \{ E_{(\alpha_1, \ldots, \alpha_m)}^{(\beta_1, \ldots, \beta_m)} \}_{(\alpha_1, \ldots, \alpha_m) \in \mathcal{H}} \subseteq \mathbb{F}^{m \times \binom{n+d}{d}} \) is a rank condenser hitting set for rank \( \leq \min \{ k, \binom{n+d}{d} \} \), where,

\[
(E_{(\alpha_1, \ldots, \alpha_m)})_{i, \bar{\alpha}} = \bar{\alpha}^T ,
\]

so that we index \( \binom{n+d}{d} \) by exponent vectors \( \bar{\alpha} \in \mathbb{N}^n \) with \( \| \bar{\alpha} \|_1 \leq d \).

**Proof:** Note that \( \mathcal{C} \) is a vector space of dimension \( \binom{n+d}{d} \), so we can identify \( \mathcal{C} \equiv \mathbb{F}^{\binom{n+d}{d}} \), where we use the standard monomial basis. For a fixed \( (\alpha_1, \ldots, \alpha_m) \in (\mathbb{F}^n)^m \), consider the map \( \varphi_{(\alpha_1, \ldots, \alpha_m)} : \mathcal{C} \to \mathbb{F}^m \) given by \( f \mapsto (f(\alpha_1), \ldots, f(\alpha_m)) \). Note that this is a linear map, and using that \( \mathcal{C} \equiv \mathbb{F}^{\binom{n+d}{d}} \) it follows that \( \varphi_{(\alpha_1, \ldots, \alpha_m)} \) as a map \( \mathbb{F}^{\binom{n+d}{d}} \to \mathbb{F}^m \) is exactly given by the matrix \( E_{(\alpha_1, \ldots, \alpha_m)} \).

Thus, consider any \( \bar{f} \in \mathcal{C}^k \), so that using \( \mathcal{C} \equiv \mathbb{F}^{\binom{n+d}{d}} \) this corresponds to a matrix \( M \in \mathbb{F}^{\binom{n+d}{d} \times k} \) and clearly \( M \) has rank \( \leq \min \{ k, \binom{n+d}{d} \} \). Equivalently, it is clear that for any \( M \) of this form we can create such a \( \bar{f} \). It follows that coeff-span \( \bar{f} = \text{row-span} \ M \), and that \( \text{span} \{ \bar{f}(\alpha_1), \ldots, \bar{f}(\alpha_m) \} = \text{row-span} \ E_{(\alpha_1, \ldots, \alpha_m)} M \).

That \( \mathcal{H} \) is an \( m \)-sample hitting set for the coefficient-span of \( \mathcal{C}^k \) is saying that for any such \( \bar{f} \), that

\[
\text{coeff-span} \ \bar{f} = \text{span} \{ f(\alpha_1), \ldots, f(\alpha_m) \} ,
\]

for some \( (\alpha_1, \ldots, \alpha_m) \in \mathcal{H} \). Unwrapping this, we see that this is equivalent to saying that

\[
\text{row-span} \ M = \text{coeff-span} \ \bar{f} = \text{span} \{ \bar{f}(\alpha_1), \ldots, \bar{f}(\alpha_m) \} = \text{row-span} \ E_{(\alpha_1, \ldots, \alpha_m)} M ,
\]

for some \( (\alpha_1, \ldots, \alpha_m) \in \mathcal{H} \). As \( \text{row-span} \ M \supseteq \text{row-span} \ EM \) for any \( E \), so that \( \text{row-span} \ M = \text{row-span} \ EM \) iff \( \text{rank} \ M = \text{rank} \ EM \), it follows that the above is equivalent to saying that

\[
\text{rank} \ M = \text{rank} \ E_{(\alpha_1, \ldots, \alpha_m)} M ,
\]

for some \( (\alpha_1, \ldots, \alpha_m) \in \mathcal{H} \), which is exactly the desired property that the above matrices \( \{ E_{(\alpha_1, \ldots, \alpha_m)} \}_{(\alpha_1, \ldots, \alpha_m) \in \mathcal{H}} \) are a rank condenser hitting set for rank \( \leq \min \{ k, \binom{n+d}{d} \} \). \( \square \)

The above result shows that \( m \)-sample hitting sets for coefficient span are a very
specific type of rank condenser — one that is compatible with the map taking a polynomial to its evaluations. We now observe that the randomness efficient rank condenser given by Theorem 5.4.3 is compatible with polynomial evaluation, so gives a way to preserve coefficient span in few samples. For convenience, we will state this from a generator-viewpoint, which we first define.

**Definition 6.4.8.** Let $C \subseteq \mathbb{F}[x_1, \ldots, x_n]^k$ be a set of vector-valued polynomials. A polynomial map $(G_1(t), \ldots, G_m(t)) \in \mathbb{F}[t]^m$ is a $m$-sample generator for the coefficient span of $C$ if for all $f \in C$,

$$\text{coeff-span}_{\mathbb{F}(t)}(f) = \text{span}_{\mathbb{F}(t)}\{\overline{f}(G_1(t)), \ldots, \overline{f}(G_m(t))\}.$$  

Now, we use the explicit rank condenser of Theorem 5.4.3 to get an efficient way to sample the coefficient span of univariate polynomials, as this rank condenser has the required form for the above equivalence to go through.

**Corollary 6.4.9.** Let $\mathbb{F}$ be a field with $|\mathbb{F}| > d$. Let $\omega \in \mathbb{F}$ be a poly($d$)-explicit element of order $\geq d$. Let $\overline{f} \in \mathbb{F}[x]^n$ be a vector-valued univariate polynomial of degree $< d$. Then

$$\text{coeff-span}_{\mathbb{F}(t)}(\overline{f}) = \text{span}_{\mathbb{F}(t)}\{\overline{f}(t\omega^i)\}_{i \in [n]}.$$  

That is, the polynomial map $(t, t\omega, \ldots, t\omega^{n-1}) \in \mathbb{F}[t][n]$ is a poly($n, d$)-explicit $n$-sample generator for the coefficient span of univariate polynomials of degree $< d$.

**Proof:** Define the matrix $M \in \mathbb{F}[d \times n]$ by placing the coefficients of $\overline{f}$ as rows of $M$, so that,

$$M_{i,j} := \text{Coeff}_{x^i}(f_j),$$

so that $M_{i,\bullet} = \text{Coeff}_{x^i}(\overline{f})$. It follows that row-span$_{\mathbb{F}} M = \text{coeff-span}_{\mathbb{F}} \overline{f}$, and that by viewing $\overline{f}$ as having coefficients in $\mathbb{F}(t)$, row-span$_{\mathbb{F}(t)} M = \text{coeff-span}_{\mathbb{F}(t)} \overline{f}$.

Now consider the matrix $E(t) \in \mathbb{F}[t][n \times d]$ defined by Corollary 5.4.4 that yields a (relaxed) rank condenser generator for rank $\leq n$ in $d$-dimensional space (that it is “relaxed” means that we allow $d < n$). As

$$(E(t))_{i,j} = (t\omega^i)^j,$$

it follows that,

$$(E(t)M)_{i,j} = \sum_{k \in [d]} (t\omega^i)^k \cdot \text{Coeff}_{x^k}(f_j) = f_j(t\omega^i),$$

so that $(E(t)M)_{i,\bullet} = \overline{f}(t\omega^i)$. By the definition of the rank condenser, and using its application to row-spans (Lemma 5.2.2), it follows that

$$\text{row-span}_{\mathbb{F}(t)} E(t)M = \text{row-span}_{\mathbb{F}(t)} M.$$
However, by unravelling the definitions of these row-spans, it follows that,

\[
\text{span}_{F(t)} \{ \mathcal{T}(t \omega^i) \}_{i \in [n]} = \text{span}_{F(t)} \{ (E(t)M)_{i,\bullet} \}_{i \in [n]}
\]

\[
= \text{row-span}_{F(t)} E(t)M
\]

\[
= \text{row-span}_{F(t)} M
\]

\[
= \text{coeff-span}_{F(t)} \mathcal{T}
\].

That the polynomial map \((t, t\omega, \ldots, t\omega^{n-1})\) is a generator for coefficient span then follows from the definition. \(\square\)

6.5 Black-Box PIT for roABPs

In a previous section, we discussed the Raz-Shpilka [RS05] white-box PIT algorithm for roABPs. At its core, this algorithm gives a way to take two variable disjoint roABPs, expand them into their coefficient span, prune this coefficient span to a smaller spanning set via Gaussian elimination, and then repack this spanning set into a single new roABP on fewer variables.

In this section, we now implement the above strategy in a black-box PIT algorithm. As discussed in introduction to this chapter, the main difficulty in this task is converting Gaussian elimination into an oblivious method for pruning many vectors to a smaller spanning set. As any such oblivious process will need randomness, it is crucial that the Raz-Shpilka [RS05] paradigm operates in parallel, so that there are only few rounds of pruning so that each round of pruning can use the same randomness via a union bound.

In this section, we show how the randomness efficient rank condenser construction in Chapter 5 can be used to give a randomness efficient method for implementing the Raz-Shpilka [RS05] merge step. Specifically, we create span preserving curves that are low degree and show how repeated composition can yield a quasi-polynomial-size hitting set for roABPs. While this is already interesting, this hitting set is of size \(\text{poly}^{\log^2}\) instead of the \(\text{poly}^{\log}\) we desire. This blow-up is because repeated composition of the span preserving curves yields a high degree generator as the degree of a polynomial multiplies under composition. Thus, we then show how one can keep the generator low degree by reinterpolating the span preserving curves, so that we do not simply treat these curves as black-boxes. Having argued this, we then present the generator in full detail and derive its properties. In particular, we show that by tuning the parameters we can achieve a hitting set of size \(\text{poly}^{\log / \log \log}\) when the roABP is of constant width, of which there is no analogue known in the setting of boolean pseudorandomness.

6.5.1 Span Preserving Curves

In the previous section (Section 6.4), we discussed how one can prune a vector-valued polynomial into a small number of evaluations that capture the entire coefficient span, and instantiated the rank condenser of Theorem 5.4.3 as Corollary 6.4.9, yielding a randomness efficient way to do this pruning. We now argue that by interpolating
through these points, we can efficiently merge two variables in an roABP into one. To do so, let us first recall the Raz-Shpilka [RS05] paradigm.

Let $M(x) = \sum_{i \in [d]} M_i x^i \in \mathbb{F}[x]^{w \times w}$ and $N(y) = \sum_{j \in [d]} N_j y^j \in \mathbb{F}[y]^{w \times w}$ be of individual degree $< d$. Then we can represent the Raz-Shpilka [RS05] algorithm by the below equivalences, where we use “$\approx$” to denote equivalent (coefficient) spans.

$$M(x) N(y) \approx \{ M_i N_j \}_{i,j \in [d]}$$

via Gaussian elimination we find an $S = \{(i_k, j_k)\} \subseteq [d]^2$ with $|S| = w^2$,

$$\approx \{ M_i N_j \}_{(i,j) \in S}$$

repacking these coefficients into $P(z) := \sum_{k=1}^{w^2} M_{ik} N_{jk} z^k$,

$$\approx P(z).$$

Thus, what we are seeking is a black-box manner for establishing a matrix $P(z)$ so that

$$M(x) N(y) \approx P(z).$$

However, as we are only allowed to use $M$ and $N$ as black-boxes, it is clear that they only way to achieve this is to create curves $f(z), g(z) \in \mathbb{F}[z]$, so that

$$M(x) N(y) \approx M(f(z)) N(g(z)) =: P(z).$$

Indeed, one such pair of curves is $(f, g) = (z, z^d)$, as by the Kronecker substitution, it follows that there is a bijection between the coefficients of $M(x) N(y)$ and $M(z) N(z^d)$ as $M, N$ are of degree $< d$, so that

$$M(x) N(y) \approx M(z) N(z^d).$$

However, while this does yield the desired curves, note that the degree of these curves has a dependence on the degree of the original matrices. As such, if we try to recurse this scheme to preserving the coefficient span of four matrices, we get

$$M(x) N(y) P(z) Q(w) \approx M(x) N(x^d) P(z) Q(z^d)$$

$$\approx M(x) N(x^d) P(x^{d^2}) Q(x^{d^3}),$$

as the second use of the curves needs to use degree $d^2$, as $M(x) N(x^d)$ and $P(z) Q(z^d)$ have degree $< d^2$.

It should then be clear that recursing using the Kronecker substitution approach for $n$-variate roABPs of individual degree $< d$ will yield a generator of degree $d^{n-1}$, which would yield a hitting set of size $\approx d^n$, and thus be as bad as the trivial hitting set (Lemma 3.2.12).
The principal failure of the above approach is that the Kronecker substitution preserves too much about the roABP, in that it induces a bijection between coefficients. In particular, this approach works regardless of the width \( w \) of the roABP. As roABPs can compute any \( n \)-variate polynomial when \( w = \exp(n) \) (Lemma 4.4.5), it follows from a counting lower bound (Lemma 3.2.17, as roABPs are closed under subtraction (Corollary 4.4.4)) that any approach that ignores the width of the roABP must produce hitting sets of size \( \exp(n) \). Thus, as argued in the introduction to this chapter, we must be more careful about what we preserve, and only seek to preserve the coefficient span of the roABP as we merge variables.

We now give the construction of the span-preserving curves that allow us to merge variables without the degree blow-up that occurred in the Kronecker substitution. However, to do so, we must modify the above approach so that it is seeded. That is, so that the curves are \( f_\ell(z), g_\ell(z) \in \mathbb{F}[t, z] \) where \( t \) is considered a seed. We then seek to preserve the coefficient span for most values of \( t \), or equivalently, preserve the coefficient span over \( \mathbb{F}(t) \). With this seeded approach in mind, we now present the construction.

**Lemma 6.5.1** (Span Preserving Curves). Let \( w, d \geq 1 \). Let \( \mathbb{F} \) be a field with \( |\mathbb{F}| \geq w^4, d \). Let \( \omega \in \mathbb{F} \) be a \( \text{poly}(d) \)-explicit element of order \( \geq d \). Let \( S \subseteq \mathbb{F} \) be a \( \text{poly}(w) \)-explicit subset with \( |S| = w^4 \) where \( S = \{ \xi_{\ell, \ell'} \}_{\ell, \ell' \in [w^2]} \).

Let
\[
f_\ell(z) := \sum_{(\ell, \ell') \in [w^2]^2} t_\omega^{\ell} \cdot \mathbb{1}_{\xi_{\ell, \ell'}, S}(z),
\]
and
\[
g_\ell(z) := \sum_{(\ell, \ell') \in [w^2]^2} t_\omega^{\ell'} \cdot \mathbb{1}_{\xi_{\ell, \ell'}, S}(z),
\]
where we use the notation for the Lagrange interpolation polynomials from Construction B.2.1.

Then \( f_\ell(z), g_\ell(z) \) are \( \text{poly}(w, d) \)-explicit curves in \( \mathbb{F}[z, t] \) with degree \( \leq w^4 \), such that for any \( M(x) \in \mathbb{F}[x]^{w \times w} \) and \( N(y) \in \mathbb{F}[y]^{w \times w} \) of degree \( < d \),
\[
\text{coeff-span}_{\mathbb{F}(t)} M(x) N(y) = \text{coeff-span}_{\mathbb{F}(t)} M(f_\ell(z)) N(g_\ell(z)).
\]

**Proof:** correctness: Recalling that the coefficient-span always contains the span of any set of evaluations (Corollary 6.2.10), and this is equality over infinite fields (and \( \mathbb{F}(t) \) is infinite),
\[
\text{coeff-span}_{\mathbb{F}(t)} M(x) N(y) = \text{span}_{\mathbb{F}(t)} \{ M(\alpha) N(\beta) \}_{\alpha, \beta \in \mathbb{F}(t)} \\
\supseteq \text{span}_{\mathbb{F}(t)} \{ M(f_\ell(\gamma)) N(g_\ell(\gamma)) \}_{\gamma \in \mathbb{F}(t)} \\
= \text{coeff-span}_{\mathbb{F}(t)} M(f_\ell(z)) N(g_\ell(z))
\]

Thus, it suffices to exhibit the reverse containment.

For this, we again go between coefficients and evaluations,
\[
\text{coeff-span}_{\mathbb{F}(t)} M(f_\ell(z)) N(g_\ell(z)) \supseteq \text{span}_{\mathbb{F}(t)} \{ M(f_\ell(\xi_{\ell, \ell'})) N(g_\ell(\xi_{\ell, \ell'})) \}_{(\ell, \ell') \in [w^2]^2}
\]
using the definition of these curves,

$$= \text{span}_{\bar{F}(t)} \{ M( t \omega^\ell ) N( t \omega^{\ell'} ) \}_{(\ell,\ell') \in \mathbb{F}_2^2}$$

as Corollary 6.4.9 ensures these evaluations \( \{ t \omega^i \}_{i \in \mathbb{F}_2^2} \) capture the coefficient span of \( M \) and \( N \) (as \( M \) and \( N \) are \( \mathbb{F}_2 \)-dimensional vector-valued polynomials), we can use Lemma 6.2.11 as this set of evaluations has a product structure, so that,

$$= \text{coeff-span}_{\bar{F}(t)} M(x) N(y) ,$$

given the desired containment.

**degree**: This is clear from construction.

**explicitness**: This is clear from construction. \( \square \)

**Remark 6.5.2.** Note that the above result at its core uses Corollary 6.4.9 to find a set of evaluations that preserve the coefficient span. That is, in the language of Section 6.4, we are using a \( \mathbb{F}_2 \)-sample generator for coefficient span of univariate polynomials. However, by itself these sets of evaluation points do not lend themselves to recursion, as we went from a polynomial to a set of evaluations. The key observation of the above lemma is that we can repack these evaluations into a polynomial by passing a curve through the evaluations.

Note that the above construction gives curves that have degree \(< w^4 \) regardless of the degree of \( M \) and \( N \), so that there will be no exponential blow-up in the degree when using recursion (as happened above with the Kronecker substitution).

**Remark 6.5.3.** One could allow a slight dependence (eg, logarithmic) of the degree of the curves on the degree \( d \) of \( M \) and \( N \) and potentially get similar results.

However, the curves we present have no dependence at all on the degree of \( M \) and \( N \), and only depend on their size as matrices. Rather, the dependence on degree is pushed into the number of bad values of the seed \( t \). We have essentially hidden this dependence as we use generators, but this dependence will re-emerge when we interpolate the generator to yield a hitting set (Corollary 6.5.17). One could of course make the number of bad values of \( t \) explicit throughout, but this seems only more tedious. \( \diamond \)

We now conclude this subsection by showing that these span-preserving curves are enough to give quasi-polynomial size hitting sets. We will show the recursion paradigm for 4-variate roABPs only, as we will achieve a better result later in this section.

Consider \( M(x), N(y), P(z), Q(w) \) to all be \( w \times w \) matrix-valued univariate polynomials of degree \(< d \). Then using a seed \( t \), the above lemma shows that

$$M(x) N(y) \approx M( f_t(x) ) N( g_t(x) ) ,$$

where, as above, we use \( \approx \) to denote equality of coefficient span (but now over the
appropriate fraction field). Similarly, we have that

$$P(z)Q(w) \approx P(f_t(z))Q(g_t(z)) .$$

It follows from how coefficient span acts nicely with variable disjoint products (Corollary 6.2.12), that

$$M(x)N(y)P(z)Q(w) \approx M(f_t(x))N(g_t(x))P(f_t(z))Q(g_t(z)) .$$

Notice that this reduces a 4-variate roABP to a 2-variate roABP, while keeping the coefficient span the same.

From a hitting set perspective (as opposed to a generator perspective), the above shows that for most values of $t$ the curves preserve the coefficient span of $M(x)N(y)$, and similarly for $P(z)Q(w)$. A union bound argument then would show that for most values of $t$ simultaneously yield curves preserving both $M(x)N(y)$ and $P(z)Q(w)$, so that this seed will preserve the entire $M(x)N(y)P(z)Q(w)$.

We now go down to a 1-variate roABP. That is, we use a new seed $s$, and see that the same logic as above shows that, using $M'$ and $N'$ as intermediate names,

$$M(f_t(x))N(g_t(x))P(f_t(z))Q(g_t(z))$$

$$= M'(x)N'(z)$$

$$\approx M'(f_s(x))N'(g_s(x))$$

$$= M((f_t \circ f_s)(x))N((g_t \circ f_s)(x))P((f_t \circ g_s)(x))Q((g_t \circ g_s)(x)).$$

It should be then clear that we can continue this process, so that we can take an $N = 2^n$-variate roABP and reduce it to a univariate roABP with the same coefficient span, only using $n$ seeds. Each coordinate in the generator is then a composition of $n$ of the curves $f$ and $g$, where the $i$-th curve uses the $i$-th seed, and the order of the $f, g$ in the $j$-th output of the generator corresponds to the binary representation of $j$.

Once one reduces to a univariate roABP, one can interpolate-out all of the variables used in the generator. There are $n + 1 = 1 + \lg N$ — $n$ seeds and 1 remaining variable. However, note that this generator has quasipolynomial degree, as each curve has degree $(w^d)^n = w^{d \lg N}$ as we have composed $n$ curves of degree $w^d$. Thus, the interpolation step would yield a hitting set of size $\text{poly}^{\log^2}$, as opposed to the desired $\text{poly}^\log$.

**Remark 6.5.4.** The above composition is very analogous to the PRG for BPL given by Nisan [Nis92]. In that work, Nisan considers functions computed as $f(\pi) = (\Pi_i M_i(x_i))_{1,1}$, where the $M_i$ are now functions $M_i : \Sigma \rightarrow [0,1]^{w \times w}$ from some alphabet $\Sigma$ to $[0,1]$-valued matrices (where the outputs are actually stochastic matrices). His approach can be interpreted in the merge and reduce framework used in the span-preserving curves of this section. That is, using a pairwise independent hash function $h : \Sigma' \rightarrow \Sigma$, one can show that $M_i(U_{\Sigma'}) \approx M_i(h(U_{\Sigma'}))$ when $\Sigma'$ is large enough as a function of $w$. This can be thought of as an analogue of Corollary 6.4.9, which gives a small set of samples that preserves most information about a target matrix $M_i$, where “small” only depends on $w$. One then uses this sampling in a merge phase, using that $M(U_{\Sigma'})N(U_{\Sigma'}) \approx M(h(U_{\Sigma'}))N(h(U_{\Sigma'}))$, where the hash function $h$ is random. One
then has reduced the alphabet from $\Sigma^2$ to $(\Sigma')^2$. However, as $|\Sigma'|$ is only a function of $w$, this can then be recursively repeated using the same hash function in each level of recursion. This alphabet reduction can be seen as an analogue of our span-preserving curves.

The key difference in Nisan’s [Nis92] work and ours (aside from the different method of sampling), is that this naive composition/recursive yields suboptimal results in algebraic complexity because the degree of the polynomials grows rapidly — a phenomenon that is absent in boolean complexity. That this degree blow-up can be avoided is shown in the next section.

6.5.2 Degree Reduction via Reinterpolation

In this subsection we highlight the idea that allows us keep the generator low degree, while still using the ideas of repeatedly using span-preserving curves, using an idea we call reinterpolation, we we formalize in the below lemma. In the next subsection we give the full details of the generator.

The main idea in reinterpolation is to break open the construction of the span-preserving curves, so that we interpolate a curve through the output of previously constructed curves instead of interpolating a curve through their inputs. That is, the span-preserving curves use Corollary 6.4.9, which gives a set of $w^2$ points $H_t \subseteq \mathbb{F}(t)$ so that for a univariate $w \times w$ matrix, $\text{coeff-span}_{\mathbb{F}(t)} M(x) = \text{span}_{\mathbb{F}(t)} \{M(\alpha_t)\}_{\alpha_t \in H_t}$. The curves were constructed by passing a Lagrange interpolation polynomial through these $\alpha_t$, so that for $M(x), N(y)$

$$M(x)N(y) \approx M(f_t(x))N(g_t(x)),$$

and

$$P(z)Q(w) \approx P(f_t(z))Q(g_t(z)).$$

Now, instead of using the curves again, let us return to using Corollary 6.4.9, so that we get an $H_s$ of size $w^2$, depending on a new seed $s$, so that,

$$M(x)N(y) \approx \{M(f_t(\alpha_s))N(g_t(\alpha_s))\}_{\alpha_s \in H_s},$$

and

$$P(z)Q(w) \approx \{P(f_t(\alpha'_s))Q(g_t(\alpha'_s))\}_{\alpha'_s \in H_s}.$$

A recursive use of the span-preserving curves would pass a Lagrange interpolation polynomial through the points $(\alpha_s, \alpha'_s)$, which are the inputs to the above curves. However, we observe here that the while the $M, N, P, Q$ are black-box, the curves $f_t, g_t$ are white-box, as we constructed them. Thus, instead of passing curves through the inputs $(\alpha_s, \alpha'_s)$ to the above curves, we can pass a new curve through the outputted 4-tuples of the above curves

$$\{(f_t(\alpha_s), g_t(\alpha_s), f_t(\alpha'_s), g_t(\alpha'_s))\}_{\alpha_s, \alpha'_s \in H_s},$$

as these 4-tuples, of which there are $w^4$, will capture the coefficient span of the 4-variate polynomial $M(x)N(y)P(z)Q(w)$ by the above analysis. This will yield degree $w^4$. 

112
curves instead the degree $w^8$ that would arise from composing the span-preserving curves with themselves. It should then be clear that when merging two roABPs the curves will always be degree $w^4$, so that the generator is always low-degree.

While we will formally give the details of the recursion in the next section, we give here the main lemma stating the reinterpolation idea. As we will use this lemma in a $b$-ary merge (instead of merging two variables, as in the above discussion), we will state the result in this more general setting.

**Lemma 6.5.5.** Let $w, d, D \geq 1$ and $b \geq 2$. Let $\mathbb{F}$ be a field with $|\mathbb{F}| \geq w^{2b}$, $kdD + 1$. Let $\omega \in \mathbb{F}$ be an element of order $\geq kdD$. Let $S \subseteq \mathbb{F}$ be a set with $|S| = w^{2b}$ as indexed by $S = \{\xi_\ell\}_{\ell \in [w^2]^b}$.

Let $M_1(\overline{x}_1), \ldots, M_b(\overline{x}_b)$ be $k$-variate variable-disjoint matrices with $M_i(\overline{x}_i) \in \mathbb{F}[\overline{x}_i]^{w \times w}$ with individual degree $< d$. Suppose $\overline{f}_{1,\overline{x}}(y_1), \ldots, \overline{f}_{b,\overline{x}}(y_b)$ are curves with $\overline{f}_{i,\overline{x}}(y_i) \in (\mathbb{F}[y_i, \overline{s}])^k$ so that $\deg_y \overline{f}_{i,\overline{x}}(y_i) < D$, and that

$$\text{coeff-span}_{\mathbb{F}(\overline{x})} M_i(\overline{x}_i) = \text{coeff-span}_{\mathbb{F}(\overline{x})} M_i(\overline{f}_{i,\overline{x}}(y_i)).$$

Then the curve $\overline{g}_{\overline{s},\overline{t}}(z) \in (\mathbb{F}[z, \overline{s}, \overline{t}])^{bk}$ defined (using that notation for Lagrange interpolation polynomials from Construction B.2.1) by

$$\overline{g}_{\overline{s},\overline{t}}(z) := \left( \sum_{\ell \in [w^2]^b} \overline{f}_{1,\overline{s}}(t\omega^{\ell_1}) \cdot 1_{\xi_\ell, S}(z), \ldots, \sum_{\ell \in [w^2]^b} \overline{f}_{b,\overline{s}}(t\omega^{\ell_b}) \cdot 1_{\xi_\ell, S}(z) \right),$$

has

$$\text{coeff-span}_{\mathbb{F}(\overline{s},\overline{t})} \prod_{i \in [b]} M_i(\overline{x}_i) = \text{coeff-span}_{\mathbb{F}(\overline{s},\overline{t})} \left( \prod_{i \in [b]} M_i \right) \circ \overline{g}_{\overline{s},\overline{t}}(z).$$

**Proof:** We follow the proof of correctness of the span preserving curves (Lemma 6.5.1), but we now pay careful attention to composition.

Just as in Lemma 6.5.1, the coefficient span always contains the evaluation span (Corollary 6.2.10), so that plugging in the curve $\overline{g}_{\overline{s},\overline{t}}(z)$ cannot increase the coefficient span, that is,

$$\text{coeff-span}_{\mathbb{F}(\overline{s},\overline{t})} \prod_{i \in [b]} M_i(\overline{x}_i) \supseteq \text{coeff-span}_{\mathbb{F}(\overline{s},\overline{t})} \left( \prod_{i \in [b]} M_i \right) \circ \overline{g}_{\overline{s},\overline{t}}(z),$$

so it suffices to prove the reverse containment.

Proceeding again as in Lemma 6.5.1, we start with unrolling the definition of $\overline{g}_{\overline{s},\overline{t}}$,

$$\text{coeff-span}_{\mathbb{F}(\overline{s},\overline{t})} \left( \prod_{i \in [b]} M_i \right) \circ \overline{g}_{\overline{s},\overline{t}}(z) \quad (6.5.6)$$

$$= \text{coeff-span}_{\mathbb{F}(\overline{s},\overline{t})} \prod_{i \in [b]} M_i \left( \sum_{\ell \in [w^2]^b} \overline{f}_{i,\overline{s}}(t\omega^{\ell_i}) \cdot 1_{\xi_\ell, S}(z) \right).$$
using again that coefficient-span contains evaluation-span (Corollary 6.2.10),

\[
\supset \text{span}_{F(s,t)} \left\{ \prod_{i \in [b]} M_i \left( \sum_{\ell \in [w^2]^b} \mathcal{F}_{i,\pi}(t\omega^{k_i}) \cdot \mathbb{1}_{\xi_{\pi,S}(\xi_{\pi})} \right) \right\}_{\xi_{\pi} \in S}
\]

using that the Lagrange interpolation polynomials are indicator functions on evaluations in \( S \),

\[
= \text{span}_{F(s,t)} \left\{ \prod_{i \in [b]} M_i \left( \mathcal{F}_{i,\pi}(t\omega^{k_i}) \right) \right\}_{\xi_{\pi} \in S}
\]

using that the \( \bar{k} \in [w^2]^b \),

\[
= \text{span}_{F(s,t)} \left\{ \prod_{i \in [b]} M_i \left( \mathcal{F}_{i,\pi}(t\omega^{k_i}) \right) \right\}_{k_1, \ldots, k_b \in [w^2]^b}
\]

and making the product structure more explicit,

\[
= \text{span}_{F(s,t)} \prod_{i \in [b]} \left\{ M_i \left( \mathcal{F}_{i,\pi}(t\omega^{k_i}) \right) \right\}_{k_i \in [w^2]^b} \quad (6.5.7)
\]

Now, we observe that Corollary 6.4.9 ensures that

\[
\text{coeff-span}_{F(s,t)} M_i(\mathcal{F}_{i,\pi}(y_i)) = \text{span}_{F(s,t)} \left\{ M_i \left( \mathcal{F}_{i,\pi}(t\omega^{k_i}) \right) \right\}_{k_i \in [w^2]^b} \quad (6.5.8)
\]

as the \( M_i(\mathcal{F}_{i,\pi}(y_i)) \) are \( w \times w \) matrices of degree \( < kdD \) in \( y_i \), so that \( \omega \) has the desired multiplicative order.

Now, using the hypothesis on the \( \mathcal{F}_{i,\pi}(y_i) \) gives that

\[
\text{coeff-span}_{F(s,t)} M_i(\mathcal{F}_{i,\pi}(y_i)) = \text{coeff-span}_{F(s,t)} M_i(\mathcal{F}_{i,\pi}(y_i))
\]

so that \( \text{dim}_{F(s,t)} \text{coeff-span}_{F(s,t)} M_i(\mathcal{F}_{i,\pi}(y_i)) = \text{dim}_{F(s,t)} \text{coeff-span}_{F(s,t)} M_i(\mathcal{F}_{i,\pi}(y_i)) \). In the larger field \( F(s,t) \), we have that

\[
\text{coeff-span}_{F(s,t)} M_i(x_i) \supset \text{coeff-span}_{F(s,t)} M_i(\mathcal{F}_{i,\pi}(y_i))
\]

as coefficient span always contains the evaluation span (Corollary 6.2.10). Thus, as dimension is always determined over the base field, we conclude that the dimension \( \text{dim}_{F(s,t)} \text{coeff-span}_{F(s,t)} M_i(x_i) \) equals that of \( \text{dim}_{F(s,t)} \text{coeff-span}_{F(s,t)} M_i(\mathcal{F}_{i,\pi}(y_i)) \), from which the above containment becomes the equality

\[
\text{coeff-span}_{F(s,t)} M_i(x_i) = \text{coeff-span}_{F(s,t)} M_i(\mathcal{F}_{i,\pi}(y_i))
\]
Combining this with (6.5.8) yields

\[ \text{coeff-span}_F \left( \prod_{i \in [b]} M_i(x) \right) = \text{span}_F \left( \prod_{i \in [b]} \left\{ M_i \left( \tilde{f}_i(t, z_{k_i}) \right) \right\}_{k_i \in [w^2]} \right) . \]

Finally, if we continue (6.5.6),

\[ \text{coeff-span}_F \left( \prod_{i \in [b]} M_i(x) \right) = \text{coeff-span}_F \left( \prod_{i \in [b]} \left\{ M_i \left( \tilde{f}_i(t, z_{k_i}) \right) \right\}_{k_i \in [w^2]} \right) . \]

and using that this set of evaluations has a product structure, we can apply the result of Lemma 6.2.11 and the above equality, to obtain

\[ = \text{coeff-span}_F \left( \prod_{i \in [b]} M_i(x) \right) , \]

as desired. \( \square \)

6.5.3 The Generator for roABPs

In this section, we finally give the generator for roABPs, that yields \( \text{poly}(n, w, d) \)-explicit \( \text{poly}(n, w, d) \log n \)-size hitting sets for \( n \)-variate width-\( w \) roABPs, of individual degree < \( d \), and constitutes the main result of this work. The previous two subsections have essentially gave all of the ideas of the construction. Here, we mostly focus on the details, with one additional idea. This new idea is to allow recursion where we merge more than two variables at a time. This idea leads to an improvement when \( w \) is constant (see Corollary 6.5.17. (2)), and was prompted by a question by Wigderson [Wig12], as no analogous result is known for PRGs for read-once boolean branching programs (see the chapter introduction for discussion).

**Construction 6.5.9.** Let \( w, d \geq 1, n \geq 0, b \geq 2 \) and \( N = b^n \). Let \( F \) be a field of size \( |F| > dw^{2b} \cdot \max\{b^{n-1}, 1\} \), and let \( \omega \in F \) be a \( \text{poly}(d, N, w^{2b}) \)-explicit element of order \( \geq d b^{n-1} w^{2b} \) (e.g., from Lemma A.0.5). Let \( \Gamma \) be a set with \( |\Gamma| = b \). Let \( S \subseteq F \) be a \( \text{poly}(w^{2b}) \)-explicit set with \( |S| = w^{2b} \) as indexed by \( S = \{ \xi_{\ell} \}_{\ell \in [w^2]} \).

Define the polynomial map \( \mathcal{G}_n : F^{[n+1]} \rightarrow F^{[\Gamma]} \) as follows. For \( \sigma \in \Gamma^n \),

\[
(\mathcal{G}_n(t_0, \ldots, t_{n-1}, t_n))_{\sigma} := \sum_{\tilde{t}_0, \ldots, \tilde{t}_{n-1} \in [w^2]} \left( \prod_{i \in [n]} p_{i-1, \tilde{t}_{i-1}} \left( t_i \omega(\tilde{t}_i) \right) \right) \cdot p_{n-1, \tilde{t}_{n-1}}(t_n)
\]

where for \( i \in [n] \), \( p_{i, \tilde{t}_i} \) is defined via the Lagrange interpolation polynomial (Construction B.2.1),

\[ p_{i, \tilde{t}_i}(x) := \mathbb{1}_{\tilde{t}_i}(x) , \]

and where we abuse notation (as \( \tilde{t}_{-1} \) is undefined) and define, for \( i = -1 \),

\[ p_{-1, \tilde{t}_{-1}}(x) := x . \]

115
That is, for \( n = 0 \), we have that an output indexed by \( \Gamma^{[0]} = \{ \lambda \} \), where \( \lambda \) is the empty string, so that,
\[
(\mathcal{G}_0(t_0))_{\lambda} = t_0.
\]

In this case, the generator does not depend on \( \omega \). For \( n = 1 \), we have an output indexed by \( \Gamma^{[1]} = \Gamma \), so that for \( \sigma \in \Gamma \),
\[
(\mathcal{G}_1(t_0, t_1))_{\sigma} = \sum_{\tau_0 \in \mathbb{F}^{[w^2]^\Gamma}} \left( t_0 \omega(\tau_0)_{\sigma_0} \right) \cdot \mathbb{1}_{\xi_{\tau_0}} S(t_1).
\]

Finally, for \( n > 1 \), we have that for \( \bar{\sigma} \in \Gamma^{[n]} \), that
\[
(\mathcal{G}_n(\bar{\sigma}))_{\bar{\sigma}} = \sum_{\tau_0, \ldots, \tau_{n-1} \in \mathbb{F}^{[w^2]^\Gamma}} \left( t_0 \omega(\tau_0)_{\sigma_0} \right) \cdot \mathbb{1}_{\xi_{\tau_0}} S \left( t_1 \omega(\tau_1)_{\sigma_1} \right) \cdots \mathbb{1}_{\xi_{\tau_{n-2}}} S \left( t_{n-1} \omega(\tau_{n-1})_{\sigma_{n-1}} \right) \cdot \mathbb{1}_{\xi_{\tau_{n-1}}} S(t_n)
\]

We now establish properties of this construction. We first prove an easy lemma that gives an upper bound on the degrees of the variables in \( \mathcal{G} \).

**Lemma 6.5.10 (Bounding the degree).** Assume the setup of Construction 6.5.9. Let \( \sigma \in \Gamma^{[n]} \). Then for \( 0 \leq i \leq n \), \( \deg_{t_i} \left( \mathcal{G}_n(t_0, \ldots, t_n) \right)_{\sigma} < w^{2b} \).

**Proof:** By the construction of the Lagrange interpolation polynomials (Construction B.2.1), \( \mathbb{1}_{\xi_{\tau_i}} S(y) \) is a polynomial of degree \( < |S| = w^{2b} \) (Lemma B.2.2). It follows that for all \( -1 \leq i \leq n \), \( \deg(p_{i, \tau_i}(y)) < w^{2b} \) (even for \( i = -1 \), as then \( p_{i, \tau_i}(y) = y \), and \( w \geq 1 \)). Thus, as \( t_i \) appears in \( \mathcal{G} \) only via an evaluation point of \( p_{i-1, \tau_{i-1}} \), and \( \omega \) is a scalar, the claim follows.

We now show that the generator can be efficiently computed.

**Lemma 6.5.11 (Efficiency).** Assume the setup of Construction 6.5.9. For each \( \sigma \in \Gamma^{[n]} \), \( \mathcal{G}_n(t_0, \ldots, t_n)_{\sigma} \in \mathbb{F}[t_0, \ldots, t_n] \) can be computed by a \( \text{poly}(n, w^{2b}, b) \)-explicit width-(\( w^{2b} \)) roABP, of individual degree \( < w^{2b} \), in variable order \( t_0 < \cdots < t_n \).

**Proof:** construction: For \( 0 \leq i \leq n \), define \( N_i(t_i) \in \mathbb{F}[t_i]^{[w^2]^\Gamma} \times [w^2]^\Gamma \), by
\[
(N_i(t_i))_{\tau_{i-1}, \tau_i} := p_{i-1, \tau_{i-1}} \left( t_i \omega(\tau_i)_{\sigma_i} \right),
\]
following the definition of the \( p_{i, \tau_i} \) given in Construction 6.5.9 (and again abusing notation for \( i = -1 \)), where we take \( \sigma_n \in \Gamma \) to be arbitrary. It follows then that the \( N_i \) have individual degree \( < w^{2b} \), as argued in Lemma 6.5.10. Let the roABP be
\[
\prod_{i=0}^{n} N_i(t_i)_{\bar{\sigma}}
\]
where \( \bar{\sigma} \) to be the vector in \( [w^2]^\Gamma \) consisting of \( b \) zeroes. This roABP is width \( w^{2b} \), as seen by the equivalence of variable disjoint matrix products and roABPs (Corollary 4.4.2).
**Correctness:** We can then expand the product of the \( N_i \) as
\[
\left( \prod_{0 \leq i \leq n} N_i(t_i) \right)_{\overline{0}, \overline{\sigma}} = \sum_{\overline{t}_0, \ldots, \overline{t}_n-1 \in \llbracket w^2 \rrbracket} (N_0(t_0))_{\overline{0}, \overline{t}_0} \cdot (N_1(t_1))_{\overline{t}_0, \overline{t}_1} \cdots (N_{n-1}(t_{n-1}))_{\overline{t}_{n-2}, \overline{t}_{n-1}} \cdot (N_n(t_n))_{\overline{t}_{n-1}, \overline{t}_n}
\]
unpakcing the definition of the \( N_i \),
\[
= \sum_{\overline{t}_0, \ldots, \overline{t}_n-1 \in \llbracket w^2 \rrbracket} p_{-1, \overline{0}} \left( t_0 \omega(\overline{t}_0)_{\sigma_0} \right) \cdots p_{n-2, \overline{t}_{n-2}} \left( t_{n-1} \omega(\overline{t}_{n-1})_{\sigma_{n-1}} \right) \cdot p_{n-1, \overline{t}_{n-1}} \left( t_n \omega(\overline{0})_{\sigma_n} \right)
\]
using that \((\overline{0})_{\sigma_n} = 0\), so that \( \omega(\overline{0})_{\sigma_n} = 1\), and that \( p_{-1, \overline{t}_{-1}} \) does not depend on \( \overline{t}_{-1}\),
\[
= \sum_{\overline{t}_0, \ldots, \overline{t}_n-1 \in \llbracket w^2 \rrbracket} p_{-1, \overline{t}_{-1}} \left( t_0 \omega(\overline{t}_0)_{\sigma_0} \right) \cdots p_{n-2, \overline{t}_{n-2}} \left( t_{n-1} \omega(\overline{t}_{n-1})_{\sigma_{n-1}} \right) \cdot p_{n-1, \overline{t}_{n-1}} (t_n)
\]
were we in the last step invoke the construction of \( \mathcal{G}_n \).

**Explicitness:** First note that the desired \( \omega \) can be found in \( \text{poly}(d, N, w^{2b}) \) steps as \( \mathbb{F} \) is large enough (Lemma A.0.5). Similarly, the set \( S \) be constructed by taking the first \( w^{2b} \) elements in any enumeration of \( \mathbb{F} \), as \( \mathbb{F} \) is large enough.

Finally, there are \( n + 1 \) such \( N_i \), and each of the \( (w^{2b})^2 \) entries is a Lagrange interpolation polynomial over the set \( S \), which can be computed in \( \text{poly}(w^{2b}) \) steps (Lemma B.2.3). Then, evaluating these polynomials on the \( t_i \omega(\overline{t}_i)_{\sigma_i} \), and expressing them as a univariate in \( t_i \) takes \( \text{poly}(w^{2b}) \) steps, as \( (\overline{t}_i)_{\sigma_i} \in \llbracket w^2 \rrbracket \). Thus, we can compute all of the \( N_i \) in \( \text{poly}(n, w^{2b}) \) steps.

We now establish the recursive structure of this generator, which will allow us to plug this generator into the reinterpolation scheme presented in Lemma 6.5.5.

**Lemma 6.5.12** (The recursive structure). Assume the setup of Construction 6.5.9. For \( \sigma \in \Gamma^{[n]} \), \( \sigma_n \in \Gamma \), and \( \overline{t} \) denoting the variables \((t_i)_{i \in [n]} \),
\[
(\mathcal{G}_{n+1}(\overline{t}, t_n, t_{n+1}))_{\sigma_{n+1}} = \sum_{\overline{\ell}_n \in \llbracket w^2 \rrbracket} (\mathcal{G}_n(\overline{t}, t_n \omega(\overline{t}_n)_{\sigma_n}))_{\sigma} \cdot 1_{(\overline{t}_n)_{\sigma_n}}(t_{n+1})
\]
Proof: We expand $\mathcal{G}_{n+1}$ by the definition and do some re-arrangement,

$$(\mathcal{G}_{n+1}(\vec{t}, t_n, t_{n+1}))_{\sigma_{n+1}}$$

$$= \sum_{\vec{\tau}_0, \ldots, \vec{\tau}_n \in [w^2]^P} \left( \prod_{i \in [n+1]} p_{i-1, \vec{\tau}_{i-1}} \left( t_i \omega(\vec{\tau}_i) \right) \right) \cdot p_n, \vec{\tau}_n \left( t_{n+1} \right)$$

$$= \sum_{\vec{\tau}_n \in [w^2]^P} \sum_{\vec{\tau}_0, \ldots, \vec{\tau}_{n-1} \in [w^2]^P} \left( \prod_{i \in [n]} p_{i-1, \vec{\tau}_{i-1}} \left( t_i \omega(\vec{\tau}_i) \right) \right) \cdot p_{n-1, \vec{\tau}_{n-1}} \left( t_n \omega(\vec{\tau}_n) \sigma_n \right) \cdot p_n, \vec{\tau}_n \left( t_{n+1} \right)$$

invoking the definition of $\mathcal{G}_n$,

$$= \sum_{\vec{\tau}_n \in [w^2]^P} \left( \mathcal{G}_n \left( \vec{t}, t_n \omega(\vec{\tau}_n) \sigma_n \right) \right)_{\sigma} \cdot p_n, \vec{\tau}_n \left( t_{n+1} \right)$$

invoking the definition of $p_n, \vec{\tau}_n$ as a Lagrange interpolation polynomial, as $n \geq 0$,

$$= \sum_{\vec{\tau}_n \in [w^2]^P} \left( \mathcal{G}_n \left( \vec{t}, t_n \omega(\vec{\tau}_n) \sigma_n \right) \right)_{\sigma} \cdot 1_{\xi_n} \cdot S(t_{n+1})$$

as desired. $\square$

We now use the above recursive structure to recursively apply the reinterpolation lemma (Lemma 6.5.5) showing that coefficient span is preserved.

**Lemma 6.5.13** (Span preserving). Assume the setup of Construction 6.5.9.

Let $M_\sigma(x_\sigma) \in \mathbb{F}[x_\sigma]^{w \times w}$ for $\sigma \in \Gamma^n$ be of degree $< d$. Then, denoting $\vec{t}$ for the vector of variables $(t_i)_{i \in [n]}$,

$$\text{coeff-span}_{\mathbb{F}(\vec{t})} \prod_{\sigma \in \Gamma^{[n]}} M_\sigma \left( x_\sigma \right) = \text{coeff-span}_{\mathbb{F}(\vec{t})} \prod_{\sigma \in \Gamma^{[n]}} M_\sigma \left( \left( \mathcal{G}_n(\vec{t}, t_n) \right)_{\sigma} \right), \quad (6.5.14)$$

so the left-hand-side is coefficients in $\pi$, and the right-hand-side is coefficients of $t_n$, all over the fraction field $\mathbb{F}(\vec{t}) = \mathbb{F}(t_0, \ldots, t_{n-1})$.

**Proof:** We proceed by induction on $n$.

$n = 0$: For $n = 0$, $\Gamma^0 = \{ \lambda \}$, where $\lambda$ is the empty string, so that there is only the variable $x_\lambda$. Thus, by construction, $(\mathcal{G}_0(t_0))_\lambda = t_0$ as $\vec{t}$ contains no variables, so that

$$\text{coeff-span}_\pi M_\lambda \left( \left( \mathcal{G}_0(t_0) \right)_\lambda \right) = \text{coeff-span}_\pi M_\lambda \left( t_0 \right),$$

which trivially equals coeff-span$_\pi M_\lambda (x_\lambda)$.

$n \to n + 1$: We now assume the claim for $n$ and establish it for $n + 1$.

We will split the multiplication $\prod_{\sigma \in \Gamma^{[n+1]}} M_\sigma \left( x_\sigma \right)$ into $b$ parts, one for each symbol in $\Gamma$, then use induction and reinterpolation (Lemma 6.5.5) to argue that span is...
coefficient of \( n \) in the coefficients of \( t_n \). Then, appealing to the recursive structure of the generator (Lemma 6.5.12),

\[
\begin{align*}
= \text{coeff-span}_{F[\bar{t}, t_n]} \prod_{\sigma_n \in \Gamma} \prod_{\pi \in \Gamma^{n+1}} M_{\pi \sigma_n} \left( \left( G_n(\bar{t}, t_n) \right)_{\pi} \right) \\
= \text{coeff-span}_{F[\bar{t}, t_n]} \prod_{\sigma_n \in \Gamma} \prod_{\pi \in \Gamma^{n+1}} M_{\pi \sigma_n} \left( \left( G_{n+1}(\bar{t}, t_n, t_{n+1}) \right)_{\pi} \right),
\end{align*}
\]

as desired.

We now conclude that the construction yields an explicit hitting set for small roABPs.

**Theorem 6.5.15** (PIT for read-once oblivious ABPs). Let \( n, w, d \geq 1, b \geq 2 \). Let \( C \) be the set of polynomials \( \mathcal{f}(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n] \) computable by a width-(\( \leq w \)) roABP, with individual degree \( < d \), in the variable order \( x_1 < \ldots < x_n \).

If \( |\mathbb{F}| > d^2w^{2b} \), then there is a \( \text{poly}(n, w, d, b) \)-explicit generator \( \mathcal{G} : \mathbb{F}^{\log_b n + 1} \rightarrow \mathbb{F}^n \) for \( C \), of individual degree \( < w^{2b} \).

**Proof:** Consider Construction 6.5.9, where we take \( w, d \) as above, and set \( N = b^{\log_b n} \leq bn \), so that we need a field of size \( > d^{2b} \cdot \max\{N/b, 1\} \), which in particular holds as \( |\mathbb{F}| > d^{2b} \) and \( n \geq N/b, 1 \). This yields a map from \( \mathcal{G} : \mathbb{F}^{\log_b n + 1} \rightarrow \mathbb{F}^N \), which we can truncate to yield the map \( \mathcal{G} : \mathbb{F}^{\log_b n + 1} \rightarrow \mathbb{F}^n \) by taking the first \( n \) outputs. By Lemma 6.5.11, each coordinate of \( \mathcal{G} \) is \( \text{poly}(\log_b n, b, w^{2b}) \)-explicit, so that the entire map is \( \text{poly}(n, b, w^{2b}) \)-explicit.

Now, observe that any \( f \in \mathcal{C} \) can be expressed (by Corollary 4.4.2) as a polynomial of the form \( f(\mathbf{x}) = (\prod_{i \in [n]} M_i(x_i))_{1,1} \), where \( M_i(x_i) \in \mathbb{F}[x_i]^{w \times w} \) is of individual degree
< d. By defining $M_i(x_i) = 1_w$ for $i$ in $n < i \leq N$, we can pad $f$ to be on $N$-variables, and thus can express $f$ as $f(\bar{x}) = (\prod_{i\in[N]} M_i(x_i))_{1,1}$. It follows then from the span-preserving properties of this generator (Lemma 6.5.13) that for $\bar{t} = (t_0, \ldots, t_{\lceil \log_2 n \rceil - 1})$,

$$\text{coeff-span}_{\mathcal{F}(\bar{t})} \prod_{i\in[N]} M_i(x_i) = \text{coeff-span}_{\mathcal{F}(\bar{t})} \prod_{i\in[N]} M_i \left( (\mathcal{G}'(\bar{t}, t_{\lceil \log_2 n \rceil}))_i \right).$$

Thus, as $f$ is simply an entry in the above matrix product, we can use that preserving coefficient-span of a vector-valued polynomial preserves the non-zeroness of any coordinate (Corollary 6.2.6 plus Lemma 6.2.3), so thus,

$$f(x_1, \ldots, x_n, x_{n+1}, \ldots, x_N) \equiv 0 \iff (f \circ \mathcal{G}')(t_0, \ldots, t_{\lceil \log_2 n \rceil}) \equiv 0,$$

and thus,

$$f(x_1, \ldots, x_n) \equiv 0 \iff f(x_1, \ldots, x_n, x_{n+1}, \ldots, x_N) \equiv 0$$

$$\iff (f \circ \mathcal{G}')(t_0, \ldots, t_{\lceil \log_2 n \rceil}) \equiv 0$$

$$\iff (f \circ \mathcal{G}')(t_0, \ldots, t_{\lceil \log_2 n \rceil}) \equiv 0,$$

where in the last step we used that $f$ only depends on the first $n$ outputs of $\mathcal{G}'$, which is the output of $\mathcal{G}$, thus showing that $\mathcal{G}$ is the desired generator.

We now turn this generator into a hitting set, using the generic conversion of Lemma 3.2.21.

**Corollary 6.5.16.** Let $n, w, d \geq 1, b \geq 2$. Let $\mathcal{C}$ be the set of polynomials $f(\bar{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ computable by a width-$(\leq w)$ roABP, with individual degree $< d$, in the variable order $x_1 < \ldots < x_n$.

If $|\mathbb{F}| > dnw^{2b}$, then there is a poly($n, w^{2b}, d, b$)-explicit hitting set $\mathcal{H} \subseteq \mathbb{F}^n$ of size $(ndw^{2b})^{\lceil \log_2 n \rceil + 1}$.

Finally, we conclude by instantiating this with particular values of $b$. That is, in the initial versions of this work, only $b = 2$ was considered, corresponding to a divide-and-conqueror scheme where we break the roABP into two pieces. Clearly, taking $b \gg 2$ will lead to poor results as the $w^{2b}$ term will become too large. However, Wigderson [Wig12] asked how the result scales when $b > 2$, in particular when $w = \mathcal{O}(1)$. In the above hitting set, one can see that when $w = \mathcal{O}(1)$, one can choose $b$ so that $w^{2b} \approx nd$, so that $b \approx \lg nd$, and in this case the hitting set becomes of size poly$^{\log_{\log_{\log 2}}}$, which is an improvement over the case when $b = 2$ (which yields poly$^{\log_2}$). As discussed in the chapter introduction, no such analogous result is known in the setting of boolean pseudorandomness. We now turn to the exact parameters.

**Corollary 6.5.17.** Let $n, w, d \geq 1$. Let $\mathcal{C}$ be the set of polynomials $f(\bar{x}) \in \mathbb{F}[\bar{x}]$ computable by a width-$(\leq w)$ roABP, with individual degree $< d$, in the variable order $x_1 < \ldots < x_n$. Then,

1. If $|\mathbb{F}| > dnw^4$, then there is a poly($n, w, d$)-explicit hitting set $\mathcal{H} \subseteq \mathbb{F}^n$ for $\mathcal{C}$ of size $(ndw^4)^{\lceil \log_2 n \rceil + 1}$.
2. For \( w \geq 2 \), if \(|F| > n^3 d w^4\), then there is a \( \text{poly}(n, w, d) \)-explicit hitting set \( \mathcal{H} \subseteq \mathbb{F}^n \) for \( \mathcal{C} \) of size \( \leq \left( n^3 d w^4 \right)^{\frac{\lg n}{\lg \lg n}} + 2 \). In particular, if \( w = O(1) \), then \(|\mathcal{H}| \leq \text{poly}(n,d)^{O\left( \frac{\lg n}{\lg \lg n} \right)}\).

Proof: (1): This follows from Corollary 6.5.16 with taking \( b = 2 \).

(2): In Corollary 6.5.16, take \( b := \lceil \log_w n \rceil + 2 \). As \( n \geq 1 \), it follows that \( b \geq 2 \).

Thus,

\[
\begin{align*}
  w^{2b} &= w^{2(\lceil \log_w n \rceil + 2)} \\
           &= w^4 \cdot w^{2\lceil \log_w n \rceil} \\
           &\leq w^4 \cdot w^{2\log_w n} \\
           &= w^4 \cdot n^2,
\end{align*}
\]

and,

\[
\begin{align*}
  \lceil \log_b n \rceil + 1 &\leq \log_b n + 2 \\
                            &= \frac{\lg n}{\lg b} + 2 \\
                            &= \frac{\lg n}{\lg(\lceil \log_w n \rceil + 2)} + 2
\end{align*}
\]

as \( \lceil \log_w n \rceil + 2 \geq \log_w n \),

\[
\leq \frac{\lg n}{\lg \log_w n} + 2.
\]

Thus, the hitting set of Corollary 6.5.16 is of size \( (ndw^{2b})^{\lceil \log_b n \rceil + 1} \leq (n^3 d w^4)^{\frac{\lg n}{\lg \lg n}} + 2 \).

We make two remarks on this construction.

Remark 6.5.18. In the above generator, we have a polynomial map \( \mathbb{F}^{[n+1]} \rightarrow \mathbb{F}^{[n]} \) where each coordinate is computable by a \( \text{poly}(n, w^{2b}, d, b) \)-explicit roABP. A priori, this means that each coordinate is explicit separately, but says nothing about the uniformity between them. However, one can modify the above construction so that we get a polynomial \( \mathcal{G}(\vec{t}, \vec{r}) \in \mathbb{F}[\vec{t}, \vec{r}] \) so that when we choose the indexing set \( \Gamma \subseteq \mathbb{F} \) we get that

\[
\left( \mathcal{G}(\vec{t}) \right)_{\vec{r}} = \mathcal{G}(\vec{t}, \vec{r}),
\]

for \( \vec{r} \in \Gamma^{[n]} \subseteq \mathbb{F}^{[n]} \). Further, the polynomial \( \mathcal{G} \) is computable by \( \text{poly}(n, w^{2b}, d, b) \)-size circuit, and in particular, is computable as a small roABP in the \( \vec{t} \) when \( \vec{r} \) is fixed, and is also computable as a small roABP in the \( \vec{r} \) when the \( \vec{t} \) are fixed. To achieve this construction, one uses the input \( \sigma_i \) to pick out the constant \( \omega^{(\vec{t})}_{\sigma_i} \), via a Lagrange interpolation polynomial instead of via computation.

When phrased as above, this highlights that the computation of a coordinate depends on the output size as \( \text{polylog}(|\Gamma^{[n]}|) \) instead of \( \text{poly}(|\Gamma^{[n]}|) \), so in a sense is extremely
explicit, and could be considered not just as a pseudorandom generator but also as the stronger notion of a pseudorandom function from cryptography. However, this is not exactly true, as the explicitness of the generator depends polynomially on \( w \) instead of polylogarithmically.

However, these viewpoint seem to offer no quantitative advantage, and complicates notation, so we do not do so here.

\[ \text{Remark 6.5.19. The generator we established for roABPs is also computable by an roABP, but on logarithmically many variables. It is a interesting question to try to apply recursion to the composition of roABPs. However, the composition of roABPs is very not read-once, so this model seems hard to understand.} \]

6.5.4 Hitting Sets for Small Eval-Dimension and smABPs

In the previous section, we gave black-box PIT results for roABPs in any fixed variable order, which are a syntactic model. However, in Subsection 4.5.2 we saw that Nisan’s [Nis91] rank arguments gave a semantic characterization of roABPs by the dimension of certain associated vector spaces. In particular, the notion of evaluation dimension from Subsection 4.5.4, due to Saptharishi [Sap12], is a particularly clean characterization of when a polynomial can be computed by an roABP in any variable (over large fields). In this section, we record as corollaries hitting sets for small evaluation dimension by invoking the equivalence with roABPs. Similarly, we also obtain hitting sets for smABPs.

\[ \text{Corollary 6.5.20 (Saptharishi [Sap12]). Let } n, w, d \geq 1. \text{ Let } C \text{ be the set of polynomials } f(\overline{x}) \in \mathbb{F}[x_1, \ldots, x_n] \text{ with } \dim \text{Eval}(f) \leq w, \text{ with individual degree } < d. \text{ Then, if } |\mathbb{F}| > dnw^4, \text{ then there is a poly}(n, w, d)-explicit hitting set } H \subseteq \mathbb{F}^n \text{ for } C \text{ of size } (ndw^4)^{\lceil \lg n \rceil} + 1. \]

We note that the above was first observed by Saptharishi [Sap12], as he showed the proof technique of Forbes-Shpilka [FS13b] for constructing hitting sets for roABPs can be applied also to polynomials with small evaluation dimension. The formal connection between roABP width and evaluation dimension given in Subsection 4.5.4 was then made in a subsequent version of Forbes-Shpilka [FS13b], and thus gives a more modular proof of the above.

We note that hitting sets for smABPs follows likewise.

\[ \text{Theorem 6.5.21 (PIT for set-multilinear ABPs). Let } \overline{x} = \bigsqcup_{i \in [d]} \overline{x}_i \text{ where } \overline{x}_i = \{x_{i,j}\}_{j \in [n]} \text{ is a known partition. Let } C \text{ be the class of set-multilinear polynomials } f \in \mathbb{F}[\bigsqcup_{i \in [d]} \overline{x}_i], \text{ computable by a width } w, \text{ depth } d, \text{ smABP, with partition size } n. \text{ If } |\mathbb{F}| \geq \text{poly}(n, w, d), \text{ then } C \text{ has a poly}(n, w, d)-explicit hitting set, of size poly}(n, w, d)^{O(\lg d)). \]

\[ \text{Proof: By the degree bounds, the Kronecker map } x_{i,j} \leftarrow x_j \text{ induces a bijection among the monomials, and takes the original smABP to a roABP, where we replace the set of variables } \overline{x}_i \text{ with the single (new) variable } x_i. \text{ Structurally, this is still an width } w, \text{ depth } d \text{ ABP, and as we index from zero, the individual degrees are } < n. \text{ Appealing to Corollary 6.5.17 gives the result.} \]
Chapter 7

Non-Commutative ABPs

Abstract: In this chapter we discuss non-commutative ABPs (ncABPs), as defined in Chapter 4. We start with the observation (from Nisan [Nis91] and Raz-Shpilka [RS05]) that ncABPs efficiently homogenized, and that homogeneous non-commutative ABPs are equivalent to roABPs. We then follow Forbes-Shpilka [FS13b] and show that one can obtain this homogenization via evaluation of the ncABP on a carefully chosen set of matrices, which gives a reduction from ncABPs to set-multilinear ABPs (smABPs) in the black-box setting, so that the quasipolynomial-size hitting sets of Theorem 6.5.21 for smABPs likewise imply such hitting sets for ncABPs.

7.1 Introduction

While the primary focus of algebraic complexity are polynomials over commutative rings, it has proved challenging to prove lower bounds in sufficiently powerful circuit models because any lower bound would have to exclude any possible “massive cancellation”. Thus, in recent years, attention has turned to non-commutative computation, in the hopes that strong lower bounds would be easier to obtain (see [Nis91, CS07, AJS09, AS10, HWY11, CHSS11, HWY10, HY13]). This hope lies with the fact that non-commutative computation has less structure (lacking commutativity). While at first this would seemingly imply that non-commutative computation is harder to understand (because there is less structure to work with), non-commutative computation is actually easier to understand because this lack of structure makes algorithm design harder as the “tool” of commutativity is missing.

Further, one can give formal reductions, showing that non-commutative circuits are less powerful than commutative circuits. That is, given a commutative polynomial $f(\bar{x}) \in \mathbb{F}[\bar{x}]$ one can consider any non-commutative polynomial $\hat{f}(\bar{x}) \in \mathbb{F}\langle \bar{x} \rangle$ that recovers $f(\bar{x})$ when we allow the variables to commute again. It follows then that any non-commutative algebraic circuit for $\hat{f}$ is also an algebraic circuit for $f(\bar{x})$, once we let the variables commute again. As a trivial example, if we have $f = x^2y$, then we can take $\hat{f} = xyx$ or $\hat{f} = yxx$. The converse is, of course, false, as one can consider $f = x^2 - y^2$. This can be computed commutatively via $(x - y)(x + y)$, but this computation does not yield $f$ non-commutatively. Thus, we see that getting
lower bounds (and thus understanding) non-commutative computation is easier than commutative computation (though is still difficult).

7.1.1 Previous Work

In parallel to the work on non-commutative circuit lower bounds, the PIT problem has also been studied in the non-commutative model. While the work of Raz and Shpilka [RS05] (presented in Section 6.3 in the language of roABPs) established a white-box PIT algorithm for non-commutative ABPs, the black-box PIT question for this model is more intricate since one cannot immediately apply the usual Schwartz-Zippel lemma over non-commutative domains. However, Bogdanov and Wee [BW05] showed how, leveraging the ideas in the Amitsur-Levitzki theorem [AL50], one can reduce non-commutative black-box PIT questions to commutative black-box PIT questions. By then appealing to Schwartz-Zippel, they give the first randomized algorithm for non-commutative PIT. They also discussed the possibility of derandomizing their result and raise a conjecture that if true would lead to a hitting set of size $s^{\log s}$ for non-commutative ABPs of size $s$. Our Theorem 7.2.6 gives a hitting set of size $s^{\log s}$ and does not require unproven assumptions.

In particular, their conjecture asked about the minimal size of an ncABP required to computed a polynomial identity of $n \times n$ matrices. Recall that a polynomial identity of matrices is a non-zero polynomial $f(x)$ such that no matter which $n \times n$ matrices we substitute for the variables $x_i$, $f$ evaluates to the zero matrix. Our work bypasses this conjecture, as we instead give an improved reduction from non-commutative to commutative computation, such that for ABPs the resulting computation is set-multilinear. Our construction consists of $(D + 1) \times (D + 1)$ strictly-upper-triangular matrices for depth $D$ non-commutative ABPs. It is also not hard to see that there is a non-commutative formula of depth $D + 1$ which is a polynomial identity for the space of $(D + 1) \times (D + 1)$ strictly-upper-triangular matrices. Thus, our result is tight in that it also illustrates that if we go just one dimension up above what is obviously necessary then we can already construct a hitting set.

Arvind, Mukhopadhyay and Srinivasan [AMS10] gave a deterministic black-box algorithm for identity testing of sparse non-commutative polynomials. The algorithm runs in time polynomial in the number of variables, degree and sparsity of the unknown polynomial. This is similar to the running time achieved in the commutative setting for sparse polynomials (see e.g., [BOT88, KS01]) and in particular it is better than our quasi-polynomial time algorithm in this particular regime. On the other hand our algorithm is more general and works for any non-commutative polynomial that is computed by a small ncABP.

Arvind, Mukhopadhyay and Srinivasan [AMS10] also showed how to deterministically learn sparse non-commutative polynomials in time polynomial in the number of variables, degree and sparsity. In contrast, for such polynomials our deterministic algorithm requires quasi-polynomial time. For general ncABPs Arvind-Mukhopadhyay-Srinivasan [AMS10] also obtained a deterministic polynomial time learning algorithm, but here they need to have the ability to query the ABP also at internal nodes and not just at the output node. Our deterministic algorithm runs in quasi-polynomial time.

124
time but it does not need to query the ABP at intermediate computations.

7.1.2 This Chapter

In this section, we show how to do black-box identity testing of non-commutative ABPs (defined in Definition 4.1.3) in quasi-polynomial time. As defined before, an ncABP is an ABP over the ring $\mathbb{F}\{\pi\}$, of polynomials in non-commuting variables. Nisan [Nis91] first explored this model, noting that any commutative polynomial $f$ can be computed non-commutatively (for example, by the complete monomial expansion) and thus non-commutative models of computation form a restricted class of computation we can explore. In the same work, he proved the first exponential lower bounds in the non-commutative ABP model for the (commutative) determinant and permanent polynomials (using the arguments we gave in Section 4.5). Later, Raz and Shpilka [RS05] gave a polynomial time PIT algorithm in the white-box model for this class.

In both of the above works, non-commutativity can be seen as “syntactic” in the sense that one can treat the variables as formally non-commutating free variables, and one doesn’t seek to substitute values for these variables. However, black-box PIT of non-commutative polynomials in $\mathbb{F}\{\pi\}$ by definition requires such a substitution. Such a substitution will occur in some ring $R$ that is an $\mathbb{F}$-algebra (that is, there is a (commutative) ring homomorphism $\mathbb{F} \rightarrow R$). Clearly, $R$ must be non-commutative in order to witness the non-identity $xy - yx$ in $\mathbb{F}\{\pi\}$. Trivially, we could choose “$R = \mathbb{F}\{\pi\}$” and get a black-box PIT algorithm that only requires a single query, but this simply pushes the complexity of the problem into the operations in $R$. That is, one would want the black-box PIT queries to be efficiently implementable given white-box access to the circuit. As such, it seems natural to ask that $R$ is a finite dimensional $\mathbb{F}$-vector space, and that the number of dimensions is polynomial in the relevant parameters. Further, another natural restriction is to take $R = \mathbb{F}^{[m] \times [m]}$, the algebra of matrices, and that is what we do here.

However, once we have moved from evaluations in $\mathbb{F}\{\pi\}$ to evaluations in $R$, there is the concern that we have lost information, in that $f \in \mathbb{F}\{\pi\}$ could vanish on all possible evaluations over $R$. Note that this is also a problem in the commutative case, as in the standard example of the polynomial $x^2 - x \in \mathbb{F}_2[x]$ which vanishes over the field $\mathbb{F}_2$. In the case of matrices, which is the ring we shall work over, there are such identities given by the Amitsur-Levitzki theorem [AL50]: the polynomial $\sum_{\sigma \in S_{2m}} \text{sgn}(\sigma) \prod_{i \in [2m]} A_{\sigma(i)}$, where $S_{2m}$ is the symmetric group on $2m$ elements, vanishes on every choice of $2m$ $m \times m$ matrices $A_i$, over any field.

However, recall that in the commutative case, (multivariate) polynomial interpolation states that for a polynomial $f \in \mathbb{F}[\pi]$ of (total) degree $< d$, $f$ cannot vanish on all evaluations over $\mathbb{F}$ as long as $|\mathbb{F}| \geq d$. Extending this, the Schwartz-Zippel lemma shows that if $|\mathbb{F}| \gg d$ then $f$ has a very low probability of vanishing on a random

\[1\] We need this homomorphism to make sense of how elements of $\mathbb{F}$ act on elements of $R$. That is, when evaluating a non-commutative polynomial $f$ in $\mathbb{F}\{\pi\}$ on elements of $R$, we replace the coefficients in $f$ with their images under this homomorphism.

\[2\] The relevant homomorphism $\mathbb{F} \rightarrow \mathbb{F}^{m \times m}$ maps $\pi \mapsto aI$, where $I$ is the identity matrix.
evaluation over $\mathbb{F}$. This result, applied via a union bound in a probabilistic argument, shows that efficient black-box PIT is (existentially) possible for small ABPs. In almost direct analogy, the Amitsur-Levitzki theorem [AL50] shows that polynomials of (total) degree $< 2m$ in $\mathbb{F}[x]$ cannot be identities over $\mathbb{F}^{m \times m}$. Bogdanov and Wee [BW05] observed that this result, in combination with the Schwartz-Zippel lemma, show that if $|\mathbb{F}|, m \gg d$ then a non-commutative polynomial $f$ has a very low probability of vanishing on a random evaluation over $\mathbb{F}^{m \times m}$. And thus, as in the commutative case, this establishes existence of small hitting sets for non-commutative polynomials computed by small ABPs. Given this existential argument, we now proceed to construct quasi-polynomial sized, explicit hitting sets, for non-commutative ABPs.

### 7.2 Reducing ncABPs to smABPs

We proceed, by giving a family of matrices with entries consisting of commutative variables, such that any non-zero non-commutative polynomial $f$ over these matrices induces a non-zero family of commutative polynomials over the commutative variables. Further, we show that if $f$ is computable by a small non-commutative ABP, then the resulting commutative polynomials are computable by small set-multilinear ABPs. We can then appeal to the hitting set for this class of polynomials, shown in Theorem 6.5.21. We first illustrate the construction of our matrices.

**Construction 7.2.1.** Define $\varphi : \mathbb{F}\{x_i\}_{i \in [n]} \rightarrow \mathbb{F}[x_{i,j}]_{i \in [n], j \in \mathbb{N}}$ to be the unique $\mathbb{F}$-linear map defined by $\varphi(1) = 1$, and

$$
\varphi(x_{i_1} x_{i_2} \cdots x_{i_d}) := x_{i_1,1} x_{i_2,2} \cdots x_{i_d,d}.
$$

For $i \in [n], \ D \geq 0$, define the upper-triangular matrix $X_{i,D} \in (\mathbb{F}[x_{i,j}]_{j \in \mathbb{N}})^{[D+1] \times [D+1]}$ by

$$(X_{i,D})_{k,\ell} := \begin{cases} 
  x_{i,\ell} & \text{if } k + 1 = \ell \\
  0 & \text{else}
\end{cases}.$$

Define the vector $X_D := (X_{i,D})_{i \in [n]}$.

Thus, pictorially, we have that the matrices $X_{i,D}$ look as follows.

$$
X_{i,D} = 
\begin{bmatrix}
0 & x_{i,1} & 0 & 0 & \cdots & 0 \\
0 & 0 & x_{i,2} & 0 & \cdots & 0 \\
0 & 0 & \ddots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & x_{i,D-1} & 0 \\
0 & 0 & \cdots & 0 & 0 & x_{i,D} \\
0 & 0 & \cdots & 0 & 0 & 0
\end{bmatrix}.
$$

These matrices are quite similar to the matrices used in Bogdanov-Wee [BW05] (see their Lemma 3.2), but they achieve a smaller dimension (in fact, matching the best possible, as seen by the Amitsur-Levitzki theorem [AL50]). However, their construction does not seem to give the reduction to set-multilinear computation that we need. We
now establish properties of our construction. In particular, we relate the evaluations of \( f \) on the matrices \( X_i \), to the commutative polynomial \( \varphi(f) \). Specifically, we related to the \textit{homogeneous parts} of \( \varphi(f) \), which the \( \ell \)-th part is denoted by \( H_\ell(\varphi(f)) \), following the conventions of Chapter 1.

**Lemma 7.2.2.** Assume the setup of Construction 7.2.1. Let \( f \in \mathbb{F}\{x_i\}_{i \in [n]} \) be a non-commutative polynomial. For \( D \geq \ell \geq 0 \),

\[
(f(X_D))_{0,\ell} = \varphi(H_\ell(f)).
\]

**Proof:** By \( \mathbb{F} \)-linearity (of \( \varphi \), \( H_\ell(\bullet) \), and \( (\bullet)_{0,\ell} \)), it is enough to prove the claim for monomials \( f(\bar{x}) = x_{i_1}x_{i_2} \cdots x_{i_d} \) for any \( d \geq 0 \). For \( d = 0 \), the only monomial is the identity matrix I, for which the claim is clear.

Now consider \( d > 0 \). Observe that

\[
(X_{i_1} \cdots X_{i_d})_{0,\ell} = \sum_{\ell_0=0,\ell_d=\ell}^{\ell_1,\ldots,\ell_{d-1} \in \{D+1\}} (X_{i_1})_{\ell_0,\ell_1} (X_{i_2})_{\ell_1,\ell_2} \cdots (X_{i_d})_{\ell_{d-1},\ell_d}
\]

and by construction, we see that if \( \ell_{j-1} + 1 \neq \ell_j \) then \( (X_i)_{\ell_{j-1},\ell_j} = 0 \), and so,

\[
= \sum_{\ell_0=0,\ell_d=\ell}^{\ell_1,\ldots,\ell_{d-1} \in \{D+1\}} (X_{i_1})_{\ell_0,\ell_1} (X_{i_2})_{\ell_1,\ell_2} \cdots (X_{i_d})_{\ell_{d-1},\ell_d}
\]

\[
\equiv \begin{cases} x_{i_{1,1}} \cdots x_{i_{d,d}} & d = \ell \\ 0 & \text{else} \end{cases} = \varphi(H_\ell(x_{i_1} \cdots x_{i_d}))
\]

as desired, where \((\dagger)\) follows from the observation that the only way a sequence of length \( d + 1 \) can start at 0 and end at \( \ell \), increasing 1 at each step, is if \( d = \ell \), and \( \ell_j = j \).

We now use this to give the black-box reduction from non-commutative polynomials to set-multilinear polynomials.

**Lemma 7.2.3.** Assume the setup of Construction 7.2.1. Let \( f \in \mathbb{F}\{x_i\}_{i \in [n]} \) be a non-commutative polynomial of degree \( \leq D \). Then for each \( \ell \in [D+1] \), the (commutative) polynomial \((f(X_D))_{0,\ell}\) is set-multilinear, and \( f \equiv 0 \) iff for all \( \ell \in [D+1] \), \((f(X_D))_{0,\ell} \equiv 0 \). Consequently, \( f \equiv 0 \) iff \( f(X_D) \equiv 0 \).

**Proof:** As \( \deg(f) \leq D \), it is clear that \( f \equiv 0 \) iff for all \( \ell \in [D+1] \), \( H_\ell(f) \equiv 0 \). Further, we see that \( \varphi \) is a bijection, so \( H_\ell(f) \equiv 0 \) iff \( \varphi(H_\ell(f)) \equiv 0 \), and that \( \varphi \) applied to any homogeneous non-commutative polynomial yields a set-multilinear polynomial. Finally, we use Lemma 7.2.2 to see that \( \varphi(H_\ell(f)) \equiv 0 \) iff \((f(X_D))_{0,\ell} \equiv 0 \). It follows then that \( f(\bar{x}) \equiv 0 \) iff \( f(X_D) \equiv 0 \). \( \square \)
Thus, this lemma says that to do black-box PIT of non-commutative polynomials, it is enough to do black-box PIT of set-multilinear polynomials, in order to hit each of the \((f(X_D))_{0,\ell}\). However, in order for this reduction to be useful, we must show that \((f(X_D))_{0,\ell}\) is not only set-multilinear, but is also computed by a small computational device, as otherwise no small hitting set may exist.

In the most general model, that of non-commutative circuits, it is not clear there is a simple bound on the complexity of \((f(X_D))_{0,\ell}\) in terms of the complexity of \(f\), as circuits have no global “order” in the computation. That is, in a non-commutative circuit each multiplication is ordered, and this local order is fixed by the circuit. However, as parts of the circuit may be reused, a partial computation in the circuit can appear in many different parts of the final computation. Concretely, in the circuit of the repeated squaring computation \(((x^2)^2)\), we cannot assign “\(x \rightarrow x_i\)” in any way compatible with the fact that \(\varphi(x^4) = x_1x_2x_3x_4\). Thus, it is not clear how to convert a non-commutative circuit for \(f\) into a circuit for \(\varphi(H_\ell(f))\).

However, for weaker models of computation such as ABPs, there is such an notion of ordering as each multiplication proceeds from the source to the sink, in a single direction. This allows us to relate the ABP complexity of a (non-commutative) polynomial \(f\) to the ABP complexity of the (commutative) set-multilinear polynomial \(\varphi(H_\ell(f))\). To do so, we first cite the following lemma, implicit in Nisan [Nis91] (see also Lemma 2 in Raz-Shpilka [RS05]).

**Lemma 7.2.4** (Nisan, [Nis91]). Let \(f \in \mathbb{F}\{\pi\}\) be a non-commutative polynomial computed by a non-commutative ABP of depth \(D\) and width \(r\). Then for \(1 \leq \ell \leq D\), \(H_\ell(f)\) is computed by a non-commutative ABP of depth \(\ell\) and width \(\leq r(D + 1)\). Further, each edge in this ABP is labeled with a homogeneous linear form.

We now show that for ABPs as the above lemma outputs (those with only homogeneous linear forms on the edges) there is a clear connection between the ABP complexity of \(f\) and \(\varphi(f)\).

**Lemma 7.2.5.** Let \(f \in \mathbb{F}\{\pi\}\) be a non-commutative polynomial computed by an ABP \(A\), whose edges are labelled with homogeneous linear forms. Let \(A'\) be the ABP with the same structure as \(A\), but for each edge from layer \(j - 1\) to layer \(j\) with label \(\sum_{i \in [n]} a_i x_i\), we replace the label with \(\sum_{i \in [n]} a_i x_{i,j}\). Then \(A'\) computes \(\varphi(f)\), and \(A'\) is a set-multilinear ABP.

**Proof:** By linearity, it is enough to prove the claim for a single path in the ABP \(A\). The path computes the product of its labels, and thus yields the product of linear forms
\[
g = \prod_j \sum_{i \in [n]} a_{i,j} x_i = \sum_{i_k \in [n]} (\prod_j a_{i,j}) (\prod_j x_{i,j}) .
\]

---

3In fact, the version that we use is slightly stronger than the version in [RS05], but the proof is the same and the only difference is that we use a less wasteful analysis, which is obvious from the proof there.

4The case when \(\ell = 0\) does not fit into this technicalities of this lemma, so will be treated separately in what follows.

5In fact, these ABPs can efficiently computed given the original ABP, but we do not use this fact.
Similarly, the same path in $A'$ computes

$$
\prod_j \sum_{i \in [n]} a_{i,j} x_{i,j} = \sum_{i_k \in [n]} (\prod_j a_{i,j}) (\prod_j x_{i,j})
$$

which equals $\varphi(g)$ by linearity, as desired. Finally, we see that the edges from layer $j-1$ to layer $j$ only involve the variables $x_{i,j}$, so the ABP is indeed set-multilinear. \qed

Combining the above results, we can now give the full reduction.

**Theorem 7.2.6.** Assume the setup of Construction 7.2.1. Let $\mathcal{H}$ be a hitting set for set-multilinear (commutative) polynomials with the (known) variable partition

$$
\bigsqcup_j \{x_{i,j} \mid i \in [n]\}
$$

computed by set-multilinear ABPs of width $\leq r(D+1)$ and depth $\leq D$. Define $\mathcal{H}' \subseteq (F^{[D+1] \times [D+1]}|^{[n]})$ by replacing each evaluation point in $\mathcal{H}$ with the corresponding evaluation point defined by the matrices $X_D$. Then $\mathcal{H}$ is a hitting set for non-commutative ABPs of width $r$ and depth $D$.

**Proof:** Let $f$ be computed by a non-commutative ABPs of width $r$ and depth $D$. Then, by Lemma 7.2.3, $f \equiv 0$ iff for all $\ell \in [D+1]$, the set-multilinear polynomial $(f(X_D))_{0,\ell} \equiv 0$. If $\ell = 0$, then we can observe that as the $X_i$ are strictly upper-triangular, it happens that $(f(X_D))_{0,0}$ is constant for any setting of $X_D$ (and in fact equals the constant term of $f$), so it is then clear that $(f(X_D))_{0,0} \equiv 0$ iff $(f(X_D))_{0,0}|_{X_D \in H'} \equiv 0$ as $|H'| > 0$.

If $\ell > 0$ then we use Lemma 7.2.2 and Lemma 7.2.3 to see that $(f(X_D))_{0,\ell} = \varphi(H_\ell(f))$ is a set-multilinear polynomial, and Lemma 7.2.4 and Lemma 7.2.5 imply that $\varphi(H_\ell(f))$ is computed by a width $\leq rD$, depth $\leq D$ set-multilinear ABP, and thus $(f(X_D))_{0,\ell} \equiv 0$ iff $(f(X_D))_{0,\ell}|_{X_D \in H'} \equiv 0$. \qed

Plugging in our hitting set for set-multilinear ABPs from Theorem 6.5.21, we obtain the following corollary.

**Corollary 7.2.7** (PIT for non-commutative ABPs). Let $\mathcal{NC}$ be the set of $n$-variate non-commutative polynomials computable by width $r$, depth $D$ ABPs. If $|F| \geq \text{poly}(n, r, D)$, then $\mathcal{NC}$ has a $\text{poly}(D, n, r)$-explicit hitting set over $(D+1) \times (D+1)$ matrices, of size $\leq \text{poly}(n, r, D)^{O(lg D)}$.

**Remark 7.2.8.** The use of $(D+1) \times (D+1)$ strictly upper-triangular matrices in the above results is optimal, in the sense that if we used $m \times m$ strictly upper-triangular matrices for $m < D+1$ then no such result is possible, as any such matrix $X$ has $X^D = 0$, and the monomial $x^D$ is computable by a depth $D$ ABP. \medskip
Chapter 8
Diagonal Circuits

Abstract: In this chapter, we define the (depth-4 and depth-3) diagonal circuit model of Saxena [Sax08]. We show that the depth-3 model is particularly weak using the partial derivative method of Nisan and Wigderson [NW96], and has a relatively simple quasipolynomial-size hitting set. The depth-4 model is not quite so weak, but we show that it is captured by the roABP model, and the PIT algorithms for this model from Chapter 6 apply to it.

8.1 Introduction

The model of diagonal circuits was first defined by Saxena [Sax08] in order to better understand depth-4 circuits, which as mentioned in Chapter 2 already capture a great deal of the complexity of general circuits. As also mentioned in the introduction, much of previous PIT research focused on small-depth circuits with restricted top fan-in. In contrast, the diagonal model allows unbounded top fan-in, but each multiplication gate only has few distinct factors (but the factors can have high multiplicity). This model is at its essence the sum of powers of linear forms model ($\Sigma \wedge \Sigma$), which is any polynomial that can be expressed as

$$f(\mathbf{x}) = \sum_{i \in [k]} (\alpha_{i,0} + \alpha_{i,1}x_1 + \cdots + \alpha_{i,n}x_n)^{d_i},$$

where $k, d$ are poly($n$). The full diagonal circuit model generalizes the above by allowing a small number of distinct factors in the multiplication, as opposed to a single factor. It also generalizes the above (which is computable in depth-3) to be depth-4 by allowing each factor to a sum of univariate polynomials as opposed to being a linear form.

Previous and Concurrent Work: In Saxena’s [Sax08] paper, he was able to show that the diagonal depth-4 model is in a sense non-commutative, in that he was able to reduce the white-box PIT question for diagonal depth-4 circuits to the white-box PIT problem for non-commutative ABPs “over algebras”, so that he could invoke an extension of the Raz-Shpilka [RS05] algorithm to solve white-box PIT for diagonal
circuits. Similarly, one can derive lower bounds for diagonal depth-4 circuits via lower bounds for non-commutative ABPs.

The key idea in his reduction is a notion of duality, where one converts a power such as \((x_1 + \cdots + x_n)^d\) into a sum of products of univariate polynomials. As developed by Saxena [Sax08], this trick only works in full generality over fields of polynomially-large characteristic.

In subsequent work, Kayal [Kay10] gave an alternate white-box PIT algorithm for diagonal depth-4 circuits. Saha, Saptharishi and Saxena [SSS13] generalized Saxena’s [Sax08] algorithm to work in the semi-diagonal model, which we do not define here. Saptharishi [Sap13] later rephrased Kayal’s [Kay10] algorithm in the language of evaluation dimension as defined in Subsection 4.5.4. From this perspective, as a polynomial \(f\) having small evaluation dimension implies it has a small roABP, the algorithm of Kayal [Kay10] and Saptharishi [Sap13] can be seen to be instantiating the Raz-Shpilka [RS05] algorithm in the specific case of diagonal depth-4 circuits, using that they have have small evaluation dimension to get the needed roABP-like properties.

Agrawal, Saha and Saxena [ASS13], in a paper concurrent and independent to Forbes-Shpilka [FS13b], gave a quasipolynomial-size hitting set for diagonal depth-4 circuits. Their work used a novel idea of rank concentration, which we will not define here. However, we will note that their hitting set, at least in the context of diagonal depth-3 circuits, is very similar to the hitting set presented in Corollary 8.5.7.

This Work: We divide this chapter into two distinct parts. The first part will only explore the depth-3 diagonal model, and the second part will discuss the depth-4 diagonal model.

The first part will be based on the paper of Forbes and Shpilka [FS13a] and will conclude with an explicit quasipolynomial-sized hitting set for the depth-3 diagonal model. At its core, it will extend the partial derivative technique of Nisan and Wigderson [NW96] from a lower-bound technique to a black-box PIT algorithm. The key insight in this translation is that the partial derivative method shows that for the above \(\Sigma \land \Sigma\) circuits to compute the full monomial \(x_1 \cdots x_n\) the size \(s\) must be exponentially large, so that \(s \geq \exp(n)\). This then can be scaled-down to show that a \(\Sigma \land \Sigma\) of size \(s\) computing a non-zero polynomial must compute a polynomial with a monomial of support \(\approx \log s\), at which brute-force methods can be applied to yield a quasipolynomial-size hitting set.

The second part of this chapter is based on Forbes and Shpilka [FS13b] (chronologically before Forbes and Shpilka [FS13a]), and shows that polynomials computed by diagonal depth-4 circuits are also efficiently computable by roABPs in any variable order (and thus have small evaluation dimension). It follows that the \(\text{poly}^{\log}\)-size hitting sets for roABPs (as given in Corollary 6.5.17) yield \(\text{poly}^{\log}\)-size hitting sets for diagonal depth-4 circuits.

Subsequent Work: While the diagonal depth-3 model (and the \(\Sigma \land \Sigma\) model in particular) are rather weak (in that they cannot even compute the monomial \(x_1 \cdots x_n\),
efficiently (Corollary 8.4.16)), understanding them has proven quite fruitful. Most interestingly, Gupta-Kamath-Kayal-Saptharishi [GKKS13b] used the duality idea to in showing that (arbitrary) depth-3 formulas capture (over characteristic zero), in a sense, the entire complexity of arbitrary algebraic circuits, and as such strengthened the earlier such results for depth-4.

More in the line with the work presented here, in a subsequent work, Forbes, Shpilka and Saptharishi [FSS14] combined the two hitting sets presented in this chapter to give a polyloglog-size hitting set for diagonal depth-3 circuits (but not depth-4). The ideas for this combination then lead to similar results for a slightly more general framework of combining the rank concentration ideas of Agrawal, Saha and Saxena [ASS13] with the generator for roABPs of Forbes-Shpilka [FS13b] presented in Section 6.5.

Chapter Organization: The chapter is organized as follows. We first give some elementary properties of the ×-norm (defined by ∥a∥ = Πi(ai + 1)), as this norm will be crucial for defining the notion of “size” of a diagonal circuit. We then define diagonal circuits (depth-3 and depth-4), and show that the model is complete (so can compute any polynomial, possibly with exponential size). Then, we setup the partial derivative method and show the desired structural weakness of diagonal depth-3 circuits. We conclude the first part of this chapter by giving quasipolynomial-size hitting sets for circuits with this structural weakness.

In the second part of the chapter, we give a version of Saxena’s [Sax08] duality trick over a field of arbitrary characteristic. This immediately shows that diagonal depth-4 circuits are computable by roABPs, which immediately implies the desired hitting set.

8.2 The ×-norm

This section is devoted to the properties of the ×-norm, for which we recall the definition from Chapter 1.

**Definition 8.2.1.** For a vector a ∈ Zn, we define ∥a∥ := Πi∈[n](ai + 1).

While it is not a norm, we shall still use this word. As this norm is not standard in the literature, we devote this section to comparing it with more familiar norms such as ℓ1, ℓ0 (sparsity, also not actually a norm) and ℓ∞.

First, we get a lower bound in terms of the 0-norm.

**Lemma 8.2.2.** Let a ∈ Nn. Then 2∥a∥0 ≤ ∥a∥×.

**Proof:**

\[ ∥a∥× = \prod_{i=1}^{n}(ai + 1) = \prod_{i ∈ \text{Supp}(a)}(ai + 1) \geq \prod(1 + 1) = 2∥a∥0. \]
Note that this can be tight, as for $\mathbf{T} \in \mathbb{N}^n$ has $\|\mathbf{T}\|_0 = n$ and $\|\mathbf{T}\|_\infty = 2^n$.
Similarly, we can get a lower bound in terms of the 1-norm.

**Lemma 8.2.3.** Let $\mathbf{a} \in \mathbb{N}^n$. Then $\|\mathbf{a}\|_1 + 1 \leq \|\mathbf{a}\|_\infty$.

**Proof:** By a simple induction.

$n = 1$: Then $\|\mathbf{a}\|_\infty = (a_1 + 1) = \|\mathbf{a}\|_1 + 1$.

$n > 1$:

$$
\|\mathbf{a}\|_\infty = \prod_{i=1}^n (a_i + 1)
$$
$$
= (a_n + 1) \cdot \prod_{i=1}^{n-1} (a_i + 1)
$$
by induction,

$$
\geq (a_n + 1) \cdot (a_1 + \cdots + a_{n-1} + 1)
$$
$$
\geq (a_n) \cdot (0 + 1) \cdot (a_1 + \cdots + a_{n-1} + 1)
$$
$$
= (a_1 + \cdots + n_{n-1} + a_n + 1).
$$

This is tight whenever $\|\mathbf{a}\|_\infty = 1$.

We now given upper bounds, again starting with the 0-norm.

**Lemma 8.2.4.** Let $\mathbf{a} \in \mathbb{N}^n$. Then $\|\mathbf{a}\|_\times \leq (\|\mathbf{a}\|_\infty + 1)^{\|\mathbf{a}\|_0}$.

**Proof:**

$$
\|\mathbf{a}\|_\times = \prod_{i=1}^n (a_i + 1) = \prod_{i \in \text{Supp}(\mathbf{a})} (a_i + 1)
$$
$$
\leq \prod_i ((\|\mathbf{a}\|_\infty + 1)^{\|\mathbf{a}\|_0}.
$$

This is tight whenever $\|\mathbf{a}\|_\infty = 1$, just as in **Lemma 8.2.3**.

We now get an upper bound in terms of the 1-norm. We begin with the following fact.

**Lemma 8.2.5.** For $i \geq 0$, $i + 1 \leq 2^i$.

**Proof:** By induction.

$i = 0$: $0 + 1 = 1 = 2^0$.

$i > 0$: $2^i = 2 \cdot 2^{i-1} \geq 2 \cdot ((i - 1) + 1) = 2i = i + i \geq i + 1$, where the last step uses that $i$ is integral.

We now get the desired upper bound on the $\times$-norm.

**Lemma 8.2.6.** Let $\mathbf{a} \in \mathbb{N}^n$. Then $\|\mathbf{a}\|_\times \leq 2^{\|\mathbf{a}\|_1}$.
Proof:

\[
\|\bar{a}\|_x = \prod_{i=1}^{n}(a_i + 1) \\
\leq \prod_{i=1}^{n}2^{a_i} \\
= 2\sum_{i}^{a_i} = 2\|\bar{a}\|_1.
\]

These results show that the $\times$-norm is “stronger” than the 0- and 1-norm, so that get upper bounds on the $\times$-norm is better than upper bounds on these other norms. We give an exact characterization of the $\times$-norm of a vector $\bar{a}$ as the number of monomials “below” $\bar{a}$.

Lemma 8.2.7. Let $\bar{a} \in \mathbb{N}^n$. Then $S = \{\bar{b} : \bar{b} \in \mathbb{N}^n, \bar{0} \leq \bar{b} \leq \bar{a}\}$ has $|S| = \|\bar{a}\|_\times$.

Proof: A vector $\bar{b}$ has “$\bar{0} \leq \bar{b} \leq \bar{a}$” iff $0 \leq b_i \leq a_i$ for all $i \in [n]$. Thus, each $b_i$ has $a_i + 1$ choices, so that there are $\prod_{i}(a_i + 1) = \|\bar{a}\|_\times$ such $\bar{b}$.

As a corollary, we can use this to derive yet another upper bound on the $\times$-norm.

Corollary 8.2.8. Let $\bar{a} \in \mathbb{N}^n$. Then $\|\bar{a}\|_\times \leq (\|\bar{a}\|_0 + \|\bar{a}\|_1)$.

Proof: By the above Lemma 8.2.7, $\|\bar{a}\|_\times$ is equal to the number of monomials with exponent vectors pointwise smaller than $\bar{a}$. All such monomials have degree and support smaller than that of $\bar{a}$, so are thus contained in the set of monomials on $\|\bar{a}\|_0$ variables with degree $\leq \|\bar{a}\|_1$, giving the inequality.

8.3 Diagonal Circuits

In this section we give the definition of the diagonal circuit model, in both the depth-3 and depth-4 version.

Definition 8.3.1 (Saxena [Sax08]). A polynomial $f(x_1, \ldots, x_n)$ is computable by a depth-3 diagonal circuit if

\[
f(\bar{x}) = \sum_{i=1}^{s} f_i(\bar{x})^\bar{a}_i,
\]

where each $f_i$ is a vector of degree $\leq 1$ polynomials. The size is $(n + 1) \sum_{i=1}^{s} \|\bar{a}_i\|_\times$.

A polynomial $f(x_1, \ldots, x_n)$ is computable by a depth-4 diagonal circuit if

\[
f(\bar{x}) = \sum_{i=1}^{s} f_i(\bar{x})^\bar{a}_i
\]

where each $f_i$ is a sum of univariate polynomials, so that $f_i(\bar{x}) = \sum_{j\in[n]} f_{i,j}(x_j)$ and where $deg f_i \leq d$. The size is $(nd + 1) \sum_{i=1}^{s} \|\bar{a}_i\|_\times$. 

\[
135
\]
Thus, it follows that the depth-3 model is the subclass of the depth-4 model when \( d = 1 \).

Unwrapping the vector notation for diagonal depth-4 circuits, the above is equivalent to,

\[
f(x) = \sum_{i=1}^{s} \prod_{j} \left( \sum_{k \in [n]} f_{i,j,k}(x_k) \right)^{a_{i,j}}.
\]

While a natural measure of size for the above expression counts each \( a_{i,j} \) in binary, the size measure used here is unary. The motivation for this is that this model is a restriction of arbitrary depth-3 circuits where many of the multiplication gates become powering gates. While powering of a single polynomial can be computed more efficiently than multiplication of many distinct polynomials (via repeated squaring), we study the size of these expressions with respect to the original size definition to explore how much “easier” this model is than the arbitrary depth-3 case.

Another reason for this is that in polynomial identity testing we would always like the size of the circuit to be commensurate with its degree, as this is the most reasonable regime of the Schwartz-Zippel lemma. As such, we disallow repeated squaring. In line with this, note that (by Lemma 8.2.3) that the degree of a diagonal depth-3 circuit is at most its size.

The main interest in this class is when each \( \bar{f}_i \) is actually a single affine polynomial, so that we can express \( f \) as a depth-3 powering circuit \((\Sigma \land \Sigma)\), that is,

\[
f(x) = \sum_{i=1}^{s} \left( \alpha_{i,0} + \alpha_{i,1} x_1 + \cdots + \alpha_{i,n} x_n \right)^{d_i}
\]

for \( \alpha_{i,j} \in \mathbb{F} \). Often, the term “depth-3 diagonal circuit” simply refers to this case.

We now note that this model of computation (even the depth-3 version) can compute any polynomial, given exponential size.

**Lemma 8.3.2.** Let \( f \in \mathbb{F}[x] \) be an \( n \)-variate, degree \( \leq d \) polynomial. Then \( f \) can be computed by a diagonal depth-3 circuit of size \( \leq (n + 1) \binom{n+d}{d}^2 \).

**Proof:** Note that any such polynomial \( f \) has at most \( \binom{n+d}{d} \) monomials, and as the diagonal depth-3 model is closed under addition it suffices to compute a single degree \( \leq d \) monomial \( c_\pi \bar{x}^\pi \) in size \( \leq (n+1) \binom{n+d}{d} \). This can be done by taking \( f_1(\bar{x}) = c_\pi x_1^{a_1} \), and \( f_i(\bar{x}) = x_i^{a_i} \) for \( i > 1 \), from which it is clear that \( \bar{f}(\bar{x}) = c_\pi \bar{x}^\pi \), and the fact that \( \|\bar{x}\|_\times \leq \binom{n+d}{d} \) follows from Corollary 8.2.8. \( \square \)

### 8.4 The Partial Derivative Method

In this section we develop the *partial derivative method* pioneered by Nisan and Wigderson [NW96] for lower bounds for homogeneous depth-3 circuits. Our focus here is to use this method to construct hitting sets for diagonal depth-3 circuits, but our hitting sets will actually hit any polynomial whose space of partial derivatives is low-dimensional. As we want our proof to work over arbitrary characteristic we
will make use of Hasse derivatives, which are a variant of (formal) partial derivatives but work better over finite fields. For completeness, we will derive various properties (such as linearity and the chain and product rules) of Hasse derivatives in Appendix C. However, we will state the relevant definitions here (that are actually simplifications/consequences of the definitions given in Appendix C).

Before diving into details, we give a brief overview of the proof. The starting point is to consider the monomial \( x_1 \cdots x_n \). If one considers all derivatives of this monomial, this yields the set of polynomials \( \left\{ \prod_{i \in S} x_i \right\}_{S \subseteq [n]} \), so that there are exactly \( 2^n \) of them, which in this context is “large”.

Now consider a power of a linear polynomial, \( f(\overline{x})^d \). As \( f \) is linear, the chain rule shows that the derivatives of \( f^d \) are all multiples of \( f_i \) for \( 0 \leq i \leq d \), and thus there are only \( d + 1 \) relevant derivatives (up to rescaling). As derivatives are linear, it then follows that a summation \( \sum_{i=1}^{s} f_i(\overline{x})^d \) where all of the \( f_i \) are linear, only has \( s(d + 1) \) relevant derivatives (“small”), which one can make precise by examining the dimension of these derivatives in the vector space \( \mathbb{F}[\overline{x}] \). It follows from this discussion then that for the monomial \( x_1 \cdots x_n \) to be computed in this model that \( s \geq 2^n/(d+1) \). This is a special case of the framework of Nisan and Wigderson [NW96] and was observed by Kayal, as reported in Saxena [Sax08].

In this work, we now scale down the above lower bound. That is, for non-zero \( f(\overline{x}) \) with \( f(\overline{x}) = \sum_{i=1}^{s} f_i(\overline{x})^d \), we wish to say that \( f(\overline{x}) \) is simple. Intuitively, if \( f(\overline{x}) \) could “compute” a monomial \( \prod_{i \in S} x_i \) in some sense, then we must have \( s \gtrsim 2^{|S|} \). Put another way, \( f(\overline{x}) \) must in some sense only be able to compute \( (\log s) \)-size monomials. We make this precise by looking at the leading monomial of the polynomial \( f \), and show that the support of the leading monomial must be \( \lesssim \log s \). In the next section, we will show how to construct hitting sets for polynomials of this form.

While the above proof outline is phrased in the partial derivative method of Nisan and Wigderson [NW96], it was originally conceived as an application of the hardness of representation approach of Shpilka and Volkovich [SV09]. This idea is to automatically translate lower bounds for the monomial \( x_1 \cdots x_n \) into explicit hitting sets, which is exactly what is done here.

We now proceed with the details. We begin with the definition of Hasse derivatives.

**Definition 8.4.1** (Lemma C.1.2, Lemma C.1.4). The Hasse derivative with respect to a monomial \( \overline{x}^\pi \), denoted \( \partial_{\overline{x}^\pi} \), is the linear operator that sends \( \overline{x}^\pi \mapsto \binom{\pi}{\overline{b}} \overline{x}^{\overline{b} - \pi} \).

Here, we will want to consider the space of all Hasse derivatives of a given polynomial, which we now define.

**Definition 8.4.2** (Definition C.2.1). Let \( \partial_{\overline{x}^{<\infty}} : \mathbb{F}[\overline{x}] \to 2^{\mathbb{F}[\overline{x}]} \) by the space of all (Hasse) derivatives (function), so that

\[
\partial_{\overline{x}^{<\infty}}(f) := \text{span} \left\{ \partial_{\overline{x}^\pi}(f) \right\}_{\pi \in \mathbb{N}^n}.
\]

Note that the above span is being taken in \( \mathbb{F}[\overline{x}] \) as an \( \mathbb{F} \)-vector space.

We now show bounds on the size of the partial derivative space of a monomial. First, we give an exact characterization of this dimension.
Lemma 8.4.3. Let \( \bar{x}^\pi \in \mathbb{F}[x] \) be a monomial. Then \( \dim \partial\bar{x}^{<\infty}(\bar{x}^\pi) \) is equal to the number of \( \bar{b} \) so that \( \partial\bar{x}^{<}(\bar{x}^\pi) \neq 0 \). Further, the non-zero \( \partial\bar{x}^{<}(\bar{x}^\pi) \) are distinct, even when ignoring non-zero scalar coefficients.

Proof: By Lemma C.1.2,

\[
\partial\bar{x}^{<}(\bar{x}^\pi) = \left( \frac{\pi}{\bar{b}} \right) \bar{x}^{\pi - \bar{b}},
\]

from which it follows that the non-zero derivatives \( \partial\bar{x}^{<}(\bar{x}^\pi) \) are (non-zero multiples of) distinct monomials, which are thus trivially linearly independent and span \( \dim \partial\bar{x}^{<\infty}(\bar{x}^\pi) \).

We now turn this into an alternate characterization.

Lemma 8.4.4. Let \( \bar{x}^\pi \in \mathbb{F}[x] \) be a monomial. Then \( \dim \partial\bar{x}^{<\infty}(\bar{x}^\pi) \) is equal to the number of \( \bar{b} \) such that \( \left( \frac{\pi}{\bar{b}} \right) \neq 0 \).

Proof: This follows from Lemma 8.4.3, by seeing that \( \partial\bar{x}^{<}(\bar{x}^\pi) = \left( \frac{\pi}{\bar{b}} \right) \bar{x}^{\pi - \bar{b}} = 0 \) iff \( \left( \frac{\pi}{\bar{b}} \right) = 0 \).

We now use this characterization to yield a concrete upper bound on the dimension.

Lemma 8.4.5. Let \( \bar{x}^\pi \in \mathbb{F}[x] \). Then \( \dim \partial\bar{x}^{<\infty}(\bar{x}^\pi) \leq \|\pi\|_\times \).

Proof: By the above Lemma 8.4.3, it suffices to count the number of \( \bar{b} \) such that \( \left( \frac{\pi}{\bar{b}} \right) \neq 0 \). Note that \( \left( \frac{\pi}{\bar{b}} \right) = 0 \) if \( \bar{b} \notin \bar{a} \) (point-wise), as \( \left( \frac{\pi}{\bar{b}} \right) = \prod_i \left( \frac{a_i}{b_i} \right) \). There are \( \|\pi\|_\times \) such \( \bar{b} \) (Lemma 8.2.7), yielding the claim.

Now, we give the lower bound.

Lemma 8.4.6. Let \( \bar{x}^\pi \in \mathbb{F}[x] \). Then \( \dim \partial\bar{x}^{<\infty}(\bar{x}^\pi) \geq 2\|\pi\|_0 \).

Proof: As in the upper bound, by Lemma 8.4.3, it suffices to count the number of \( \bar{b} \) such that \( \left( \frac{\pi}{\bar{b}} \right) \neq 0 \). Consider the set \( S = \{ \bar{b} : b_i \in \{0, a_i \}, i \in \text{Supp}(\pi) \} \), so that \( |S| = 2\|\pi\|_0 \). It follows that for \( \bar{b} \in S \), \( \left( \frac{\pi}{\bar{b}} \right) = \prod_i \left( \frac{a_i}{b_i} \right) = \prod_i 1 = 1 \neq 0 \), where we used that \( \left( \frac{a_i}{0} \right) = 1 \) and that \( b_i \in \{0, a_i \} \). Thus all \( \bar{b} \in S \) have \( \left( \frac{\pi}{\bar{b}} \right) \neq 0 \) yielding the claim.

Note that in large characteristic, the above lower bound can be improved to match the above upper bound, as we now observe.

Lemma 8.4.7. Let \( \bar{x}^\pi \in \mathbb{F}[x] \). If \( \text{char}(\mathbb{F}) > \|\pi\|_\times \), then \( \dim \partial\bar{x}^{<\infty}(\bar{x}^\pi) \geq \|\pi\|_\times \).

Proof: Again, by Lemma 8.4.4, it suffices to count the number of \( \bar{b} \) such that \( \left( \frac{\pi}{\bar{b}} \right) \neq 0 \). Consider the set \( S = \{ \bar{b} : b_i \in \{0, \ldots, a_i \}, i \in \text{Supp}(\pi) \} \), so that \( |S| = \|\pi\|_\times \). It follows that for \( \bar{b} \in S \), \( \left( \frac{\pi}{\bar{b}} \right) = \prod_i \left( \frac{a_i}{b_i} \right) \neq 0 \), where we used that \( \left( \frac{a_i}{b_i} \right) = \frac{a_i}{b_i (a_i - b_i)} \neq 0 \) for \( 0 \leq b_i \leq a_i \), as long as \( \text{char}(\mathbb{F}) > a_i \). Thus all \( \bar{b} \in S \) have \( \left( \frac{\pi}{\bar{b}} \right) \neq 0 \) yielding the claim.
However, in finite characteristic the first lower bound can be tight, as seen by $x^p$ in characteristic $p$, where $\dim \partial(x^p) = 2$ but $\|p\|_\times = p + 1$. However, the slack between the above two bounds does not qualitatively affect the results presented here.

We now use the above upper bounds to show that diagonal depth-3 circuits have a small dimensional partial derivative space.

**Lemma 8.4.8.** Let $f(x) = \sum_{i=1}^s f_i(x)a_i$ be a depth-3 diagonal circuit, where $f_i(x)$ is a vector of degree $\leq 1$ polynomials. Then $\dim \partial_{x^\leq\infty}(f) \leq \sum_{i=1}^s \|a_i\|_\times$.

**Proof:** First consider $\partial_{x^\leq\infty}(f)$. As $f_i$ is of degree $\leq 1$, the chain rule (more specifically, Corollary C.2.12) implies that $\dim \partial_{x^\leq\infty}(f) \leq \dim \partial_{y^\leq\infty}(y^{a_i})$ which by Lemma 8.4.5,

$$\leq \|a_i\|_\times .$$

The claim then follows by subadditivity of dimension, as derivatives are linear operators (formally, see Lemma C.2.3). 

The above results show that a diagonal depth-3 circuit of small size has a small partial derivative space, and that monomials with large support have a large partial derivative space. These facts are sufficient for the lower bound (as observed by Kayal, as reported in Saxena [Sax08]) showing that the monomial $x_1 \cdots x_n$ requires exponentially large diagonal depth-3 circuits to compute. However, we are seeking a hitting set, which intuitively is a stronger result than a lower bound. In particular, we need to “scale down” this lower bound so that it gives meaningful “structure” even when the circuit is only polynomially large. We now proceed to give such a structural result, showing that some monomial computed by a small diagonal depth-3 circuit must have small-support. In particular, we will show this for the leading monomial of such circuits, a notion that depends on a monomial order, which we now define (see Cox, Little and O’Shea [CLO07] for more on monomial orderings and leading monomials).

**Definition 8.4.9.** A monomial ordering is a total order $\prec$ on the non-zero monomials in $\mathbb{F}[\bar{x}]$ such that

- For all $\bar{a} \in \mathbb{N}^n \setminus \{\bar{0}\}$, $1 \prec \bar{x}^{\bar{a}}$.
- For all $\bar{a}, \bar{b}, \bar{c} \in \mathbb{N}^n$, $\bar{x}^{\bar{a}} \prec \bar{x}^{\bar{b}}$ implies $\bar{x}^{\bar{a} + \bar{c}} \prec \bar{x}^{\bar{b} + \bar{c}}$.

We will often abuse notation and extend this total order to non-zero scalar multiples of monomials, so that for $\alpha, \beta \in \mathbb{F} \setminus \{0\}$, $\alpha \bar{x}^{\bar{a}} \prec \beta \bar{x}^{\bar{b}}$ if $\bar{x}^{\bar{a}} \prec \bar{x}^{\bar{b}}$.

For $f \in \mathbb{F}[\bar{x}]$, the leading monomial of $f$ (with respect to a monomial order $\prec$), denoted $\text{LM}\(f\)$, is the largest monomial in $\text{Supp}\(f\) := \{\bar{x}^{\bar{a}} : \text{Coeff}_{\bar{x}^{\bar{a}}}(f) \neq 0\}$ with respect to the monomial order $\prec$. 

139
For concreteness, one can consider the lexicographic ordering on monomials, which is easily seen to be a monomial ordering (or see Cox, Little and O’Shea [CLO07]). We now observe that derivatives are monotonic with respect to monomial orderings, except when the characteristic prevents it. That is, over characteristic \( p \) we have \( x^{p-1} \prec x^p \) (in any ordering), but \( \partial_x(x^p) = 0 \), which is not included in the ordering, so that \( \partial_x(x^{p-1}) \not\prec \partial_x(x^p) \) as the latter term is zero.

**Lemma 8.4.10.** Let \( \prec \) be a monomial ordering on \( \mathbb{F}[x] \). Let \( x^\tau \prec x^\delta \) be monomials. Then for any \( \tau \), if \( \partial_{x^\tau}(x^\tau), \partial_{x^\tau}(x^\delta) \neq 0 \) then \( \partial_{x^\tau}(x^\tau) \prec \partial_{x^\tau}(x^\delta) \) (allowing for scalar multiples).

**Proof:** By basic properties of derivatives (Lemma C.1.2), the assumption that “\( \partial_{x^\tau}(x^\tau), \partial_{x^\tau}(x^\delta) \neq 0 \)” implies that \( \tau \leq \delta \), and that
\[
\partial_{x^\tau}(x^\tau) = \alpha x^{\tau-\tau}, \quad \partial_{x^\tau}(x^\delta) = \beta x^{\delta-\tau},
\]
where \( \alpha, \beta \neq 0 \) are constants from \( \mathbb{F} \). As the monomial ordering is total, and is monotonic over multiplication, it follows \( x^{\tau-\tau} \prec x^{\delta-\tau} \), and thus \( \alpha x^{\tau-\tau} \prec \beta x^{\delta-\tau} \), as desired.

Applying this, we see that derivatives not annihilating the leading monomial of a polynomial preserve that monomial as the leading monomial. Putting it another way, the LM-operator and the \( \partial_{x^\tau} \)-operator commute, at least in some cases.

**Corollary 8.4.11.** Let \( f \in \mathbb{F}[x] \) and let \( \prec \) be a monomial ordering on \( \mathbb{F}[x] \). If \( \partial_{x^\tau}(\text{LM}(f)) \neq 0 \), then \( \partial_{x^\tau}(\text{LM}(f)) \propto \text{LM}(\partial_{x^\tau}(f)) \), where ‘\( \propto \)’ indicates equality up to non-zero scalar coefficients.

**Proof:** Expand \( f \), so that
\[
f = c_{\text{LM}(f)} \cdot \text{LM}(f) + \sum_{x^\tau = \text{LM}(f)} c_\tau x^\tau.
\]
Taking derivatives,
\[
\partial_{x^\tau}(f) = c_{\text{LM}(f)} \cdot \partial_{x^\tau}(\text{LM}(f)) + \sum c_\tau \cdot \partial_{x^\tau}(x^\tau).
\]
Those \( \bar{\tau} \) where \( \partial_{x^\bar{\tau}}(x^\tau) = 0 \) do not contribute to the above sum. For those \( \bar{\tau} \) where \( \partial_{x^\bar{\tau}}(x^\tau) \neq 0 \), it follows from **Lemma 8.4.10** that \( \partial_{x^\tau}(x^\tau) \prec \partial_{x^\tau}(\text{LM}(f)) \), as \( \partial_{x^\tau}(\text{LM}(f)) \neq 0 \). Thus, the leading monomial of \( \partial_{x^\tau}(f) \) is the term \( \partial_{x^\tau}(\text{LM}(f)) \) stripped of its non-zero scalar coefficient, yielding the claim.

We now show that for any set of polynomials the dimension of their span in \( \mathbb{F}[x] \) is equal to the number of distinct leading monomials in their span. This can be easily seen by applying Gaussian elimination to these polynomials to yield the reduced row-echelon form, when each row is ordered by monomials in decreasing monomial order. However, for concreteness we give a self-contained proof.
Lemma 8.4.12. Let \( \prec \) be a monomial order on \( \mathbb{F}[x] \) and \( S \subseteq \mathbb{F}[x] \). Then \( \dim S = |\{\text{LM}(f) : f \in \text{span}(S)\}| \).

Proof: \( \geq \): Let \( f_1, \ldots, f_r \in \text{span}(S) \) have distinct leading monomials \( \text{LM}(f_1) \prec \cdots \prec \text{LM}(f_r) \), where \( r = |\{\text{LM}(f) : f \in \text{span}(S)\}| \). The claim follows from showing that the \( f_i \) are linearly independent. Consider any non-trivial linear combination \( \sum c_i f_i \).

Consider \( i_0 = \max \{i : c_i \neq 0\} \) and examine \( \text{Coeff}_{\text{LM}(f_{i_0})}(\sum c_i f_i) \).

As \( c_i = 0 \) for \( i > i_0 \), such \( f_i \) contribute nothing to this coefficient. Similarly, for \( i < i_0 \) as \( \text{LM}(f_i) \prec \text{LM}(f_{i_0}) \) it follows that all monomials with non-zero coefficients in \( f_i \) are below \( \text{LM}(f_{i_0}) \) and thus these \( f_i \) also do not contribute to \( \text{Coeff}_{\text{LM}(f_{i_0})}(\sum c_i f_i) \).

It follows then that only the polynomial \( f_{i_0} \) contributes to this coefficient, so that \( \text{Coeff}_{\text{LM}(f_{i_0})}(\sum c_i f_i) = c_{i_0} \cdot \text{Coeff}_{\text{LM}(f_{i_0})}(f_{i_0}) \). As \( \text{Coeff}_{\text{LM}(f_{i_0})}(f_{i_0}) \neq 0 \) by definition of the leading monomial, and that \( c_{i_0} \neq 0 \) by construction of \( i_0 \), it follows that \( \text{Coeff}_{\text{LM}(f_{i_0})}(\sum c_i f_i) \neq 0 \), and thus \( \sum c_i f_i \neq 0 \). Thus, any non-trivial linear combination of the \( f_i \) is non-zero, so the \( f_i \) are linearly independent.

\( \leq \): Let \( f_1, \ldots, f_r \in \text{span}(S) \) have distinct leading monomials so that \( \{\text{LM}(f_i) : f_i \in \text{span}(S)\} = \{\text{LM}(f) : f \in \text{span}(S)\} \), and we seek to show that \( \text{span}\{f_i\} = \text{span}(S) \). Consider any polynomial \( f \), and as \( \text{span}\{f_i\} \subseteq \text{span}(S) \) the claim follows from showing that if \( f \notin \text{span}\{f_i\} \), then \( f \notin \text{span}(S) \).

Thus, consider an \( f \notin \text{span}\{f_i\} \) and consider the set \( \{\text{LM}(f - g) : g \in \text{span}\{f_i\}\} \). Let \( \pi^0 \) be the minimal monomial (with respect to \( \prec \)) in this set. As \( f \notin \text{span}\{f_i\} \) this is a non-zero monomial. Now note that \( \pi^0 \neq \text{LM}(f_i) \) for any \( i \). For, if \( \pi^0 = \text{LM}(f - g) = \text{LM}(f_i) \) then writing \( f - g = c \cdot \pi^0 + \sum c_i \pi^0 \) and \( f_i = c' \cdot \pi^0 + \sum c_{i_0} \cdot \pi^0 \) it follows that

\[
    f - \left( g + \frac{c}{c'} \cdot f_i \right) = \left( c - \frac{c}{c'} \cdot c' \right) \pi^0 + \sum_{\pi \succ \pi^0} \left( c_{\pi} - \frac{c}{c'} \cdot c_{\pi} \right) \pi = \sum_{\pi \succ \pi^0} \left( c_{\pi} - \frac{c}{c'} \cdot c_{\pi} \right) \pi
\]

has a (non-zero) leading monomial as \( g + \frac{c}{c'} \cdot f_i \in \text{span}\{f_i\} \), and as this leading monomial must be \( \prec \pi^0 \) this contradicts the minimality of \( \pi^0 \).

Thus, there is some \( g \in \text{span}\{f_i\} \subseteq \text{span}(S) \) so that \( \text{LM}(f - g) \notin \{\text{LM}(f_i) : f \in \text{span}(S)\} \). Thus, \( f - g \notin \text{span}(S) \). As \( f \in \text{span}(S) \) implies \( f - g \in \text{span}(S) \), we have that \( f \notin \text{span}(S) \) as desired.

Note that the above only requires that the monomial order is a total order, and does not require that the order is compatible with multiplication. We now use the above results to lower bound the dimension of the partial derivative space of \( f \) in terms of the dimension of the partial derivative space of the leading monomial of \( f \).

Corollary 8.4.13. Let \( f \in \mathbb{F}[x] \) be a polynomial and let \( \prec \) be any monomial ordering on \( \mathbb{F}[x] \). Then \( \dim \partial_{\prec} f \geq \dim \partial_{\prec} (\text{LM}(f)) \).

Proof: Consider those \( b \) where \( \partial_{\prec} (\text{LM}(f)) \neq 0 \), so that \( \text{LM} \left( \partial_{\prec} (f) \right) \lohdash \partial_{\prec} (\text{LM}(f)) \) by Lemma 8.4.10, where ‘\( \lohdash \)’ means equality up to a non-zero scalar multiplication. By Lemma 8.4.3, it follows that the derivatives \( \partial_{\prec} (\text{LM}(f)) \) are all distinct, even after removing non-zero scalar coefficients, as long as \( \partial_{\prec} (\text{LM}(f)) \neq 0 \). Thus using the
proportionality, it follows that the leading monomials $\text{LM} \left( \partial_{x^B} (f) \right)$ are distinct as long as $\partial_{x^B} (\text{LM}(f)) \neq 0$.

Using these facts in counting dimensions, we see that
$$
\dim \partial_{x^B} \langle f \rangle \geq \dim \text{span} \left\{ \partial_{x^B} (f) \right\}
$$
and by applying Lemma 8.4.12 to see that the dimension of derivatives is lower bounded by the number of derivatives with distinct monomials,
$$
\geq \left| \{ \text{LM} \left( \partial_{x^B} (f) \right) \} \partial_{x^B} (\text{LM}(f)) \neq 0 \right|
$$
and by the above argument, these leading monomials are all distinct,
$$
= \left| \{ \beta : \partial_{x^B} (\text{LM}(f)) \neq 0 \} \right|
$$
and thus invoking Lemma 8.4.3,
$$
= \dim \partial_{x^B} \langle f \rangle .
$$

The above facts (Lemma 8.4.6, Lemma 8.4.7, Corollary 8.4.13) show that polynomials with large-support leading monomials have a large partial derivative space. Putting these together gets our desired structural result for any polynomial with a low dimensional partial derivative space.

**Corollary 8.4.14.** Let $f \in \mathbb{F}[x]$ be a polynomial, and let $\prec$ be any monomial ordering in $\mathbb{F}[x]$. Let $x^B = \text{LM}(f)$. Then $\|x^B\|_0 \leq \lg \dim \partial_{x^B} \langle f \rangle$. If $\text{char}(\mathbb{F}) > \deg f$ then $\|x^B\|_\times \leq \dim \partial_{x^B} \langle f \rangle$. \hfill \qed

Plugging in the dimension bound for diagonal depth-3 circuits (Lemma 8.4.8) we get the desired structural result for this class.

**Corollary 8.4.15.** Let $f(x) = \sum_{i=1}^s f_i(x)$ be a depth-3 diagonal circuit, where $f_i(x)$ is a vector of degree $\leq 1$ polynomials, thus of size $s = n \sum \|\pi_i\|_\times$. Let $x^B = \text{LM}(f)$. Then $\|x^B\|_0 \leq \lg (\sum \|x^B_i\|_\times) \leq \lg s$. If $\text{char}(\mathbb{F}) > \deg f$, then $\|x^B\|_\times \leq \sum \|x^B_i\|_\times \leq s$. \hfill \qed

In particular, this gives a lower bound for the size of diagonal depth-3 circuits computing the monomial $x_1 \cdots x_n$ as the leading monomial has support $n$. This result was first established by Kayal (see Saxena [Sax08]) using the partial derivative method, and the results of this section strengthened this lower bound to apply to leading monomials.

**Corollary 8.4.16** (Kayal, as reported in Saxena [Sax08]). Any diagonal depth-3 circuit computing $x_1 \cdots x_n$ must be of size $\geq 2^n$. \hfill \qed

Note that this lower bound does not apply to the diagonal depth-4 model, as that model computes the polynomial $(\sum_{i=1}^n x_i^2)^n$ in $\text{poly}(n)$ size despite having an exponentially large space of derivatives. However, as shown in Section 8.6, hitting sets for the diagonal depth-4 model can be achieved using other techniques.
8.5 Hitting Small-Support Monomials

In this section, we construct hitting sets for polynomials that contain some monomial involving few variables, and observe that (by the previous section) this subsumes the class of diagonal depth-3 circuits, and thus also produce hitting sets for that class. We begin with the definition.

Definition 8.5.1. The set of somewhere-support-(≤ ℓ) n-variate degree-(≤ d) polynomials in \( F[x_1, \ldots, x_n] \) is the set of polynomials of degree ≤ d which (when non-zero) have some support ≤ ℓ monomial with a non-zero coefficient.

Note that the small-support monomial that is guaranteed to have a non-zero coefficient can vary between polynomials in this class.

In particular, by Corollary 8.4.15 this class of polynomials subsumes diagonal depth-3 circuits.

Corollary 8.5.2. The class of somewhere-support-(≤ ⌊lg s⌋) n-variate degree-(< d) polynomials contains as subsets:

- The class of n-variate degree-(< d) diagonal depth-3 circuits of size ≤ s.
- The class of n-variate degree-(< d) polynomials \( f(\overline{x}) \) with \( \dim \partial_{\overline{x} < \infty}(f) \leq s \).

For \( \ell \ll n \), any non-zero \( f \) in this class must necessarily have a “small” monomial with a non-zero coefficient. Intuitively, this might suggest that these polynomials are simple. While this is somewhat true, we observe that there are in fact many such polynomials, even when \( \ell \) is small.

Lemma 8.5.3. Let \( \mathbb{F} \) be a finite field. For any \( \ell \leq d \), there are \( \geq |\mathbb{F}|^{(n+d)-1} \) somewhere-support-(≤ ℓ) n-variate degree-(≤ d) polynomials in \( \mathbb{F}[x] \).

Proof: As \( \ell \leq d \), there is some monomial \( \overline{x}^{a_0} \) of degree ≤ d and of support ≤ ℓ. Consider all polynomials \( f \) so that \( \text{Coeff}_{\overline{x}^{a_0}}(f) = 1 \). As there are \( \binom{n+d}{d} - 1 \) free coefficients of \( f \) remaining, this yields the bound.

As a corollary (by invoking Lemma 3.2.17), it follows that any interpolation set for this class must be large.

Corollary 8.5.4. Let \( \mathbb{F} \) be a finite field. For any \( \ell \leq d \), any interpolation set for the somewhere-support-(≤ ℓ) n-variate degree-(≤ d) polynomials in \( \mathbb{F}[x_1, \ldots, x_n] \) must \( \geq \binom{n+d}{d} - 1 \) points.

While the above lower bound might suggest that hitting sets for this class must also be large (that is, one might hope to use the relation between hitting sets and interpolation sets (Lemma 3.2.16)), this intuition is incorrect. First, the class of somewhere-support-(≤ ℓ) polynomials is not closed under subtraction, which prevents the moral equivalence of hitting sets and interpolation sets (Lemma 3.2.16) to be relevant. Second, there are small hitting sets for this class, as we now demonstrate. In some sense, this result thus “beats” the probabilistic-method/union-bound approach.

The intuition of the construction is as follows. Given any polynomial with some support-(≤ ℓ) monomial \( \overline{x}^\overline{a} \), we first guess the location of that monomial’s support,
which should take \(\approx n^\ell\) choices. We can then zero-out all variables not in this support. This will kill off many monomials, but the monomial \(x^\alpha\) will survive this process, so that we now have a low-degree polynomial in \(\ell\) variables. We can then use a brute-force (explicit) hitting set.

We start by stating the construction of the hitting set.

**Construction 8.5.5.** Let \(n, d, \ell \geq 1\). Let \(\mathbb{F}\) be a field of size \(\geq d\). Let \(S \subseteq \mathbb{F} \setminus \{0\}\) with \(|S| = d\). Define \(H_{n,d,\ell}^{ss} \subseteq \mathbb{F}^n\) by

\[
H_{n,d,\ell}^{ss} := \{\alpha : \alpha \in S^n, \|\alpha\|_0 = \ell\} \cup \{0\}_{i \in [d^\ell - n^\ell]} \cup \{\overline{0}\}
\]

so that the vector \(\overline{0}\) is taken with \(d^\ell \left(n^\ell - \binom{n}{\ell}\right)\) repetitions.

We now establish the desired properties of this construction.

**Theorem 8.5.6.** Assume the setup of Construction 8.5.5. Then \(H_{n,d,\ell}^{ss}\) is \(\text{poly}(n, d, \ell)\)-explicit hitting set for the class of somewhere-support-(\(\leq \ell\)) \(n\)-variate degree-(\(< d\)) polynomials, with size \(|H_{n,d,\ell}^{ss}| = (nd)^\ell\).

**Proof:** \(|H'| = (nd)^\ell\): As \(0 \notin S\), there is a bijection between \(\{\alpha : \alpha \in S^n, \|\alpha\|_0 = \ell\}\) and \(\binom{n}{\ell} \times S^\ell\), defined by separating \(\alpha\) into its support and the values in that support. This uses \(0 \notin S\) so that \(\alpha\) can be separated into its zero entries and those entries in \(S\). Thus \(|\{\alpha : \alpha \in S^n, \|\alpha\|_0 = \ell\}| = \binom{n}{\ell} d^\ell\), so that adding \(d^\ell \left(n^\ell - \binom{n}{\ell}\right)\) copies of \(\overline{0}\) yields \((nd)^\ell\) total points.

\(H_{n,d,\ell}^{ss}\) is explicit: One can construct such an \(S\) in \(\text{poly}(d)\) steps. Further, given an index in \([\binom{n}{\ell}]\) one can efficiently convert this to an \((\ell, \overline{j}) \in [n]^\ell \times [d]^\ell\). If \(\overline{j} \in [n]^\ell\) is a sequence of strictly decreasing unique numbers, then it represents an \(\ell\)-subset of \([n]\) and one can output a \(\alpha\) with support from this subset and can create \(\alpha\) so that the elements from its support correspond with an element from \(S^\ell\) as given by \(\overline{j} \in [d]^\ell \equiv S^\ell\). If \(\overline{j}\) is not strictly decreasing then one can output \(\overline{0}\). It is clear that this runs in the required time, and outputs \(H_{n,d,\ell}^{ss}\).

\[f \equiv 0 \implies f|_{H_{n,d,\ell}^{ss}} \equiv 0;\] This is clear.

\[f \not\equiv 0 \implies f|_{H_{n,d,\ell}^{ss}} \not\equiv 0;\] By definition \(f\) has a non-zero coefficient on a monomial \(x^\alpha\) with \(\|\alpha\|_0 \leq \ell\), and let \(T \supseteq \text{Supp}(\alpha)\) so that \(|T| = \ell\). Let \(\overline{x}_T\) be the vector of variables indexed by \(T\) within \(x\), and define \(f_T(\overline{x}_T)\) as \(f(\overline{x})\) under the substitution that \(x_i \leftarrow 0\) for \(i \notin T\). As this substitution simply takes those monomials in \(f\) supported entirely in \(T\), and there are such monomials (eg, \(x^\alpha\)), it follows that \(f_T(\overline{x}_T)\) is of degree-(\(< d\)) and that \(f_T(\overline{x}_T) \not\equiv 0\).

Now observe that \(H_{n,d,\ell}^{ss}|_T\) contains the brute-force hitting set of Lemma 3.2.12 as by construction \(H_{n,d,\ell}^{ss}|_T\) contains all tuples in \(S^T\) as \(|T| = \ell\), and \(|S| \geq d\). Thus, \(f_T(\overline{x}_T)|_{H_{n,d,\ell}^{ss}|_T} \not\equiv 0\). Thus, there is some \(\alpha \in \mathbb{F}^n\) with \(\alpha|_T \in H_{n,d,\ell}^{ss}|_T\) and \(\alpha|_{[n]\setminus T} = \overline{0}\) so that \(f_T(\overline{\alpha}_T) \not\equiv 0\). But by construction of \(H_{n,d,\ell}^{ss}\) and the substitution used to create \(f_T\) this implies that \(\alpha \in H_{n,d,\ell}^{ss}\) and \(f(\alpha) \not\equiv 0\), as desired.

As is clear from the proof, the repeated \(\overline{0}\) in Construction 8.5.5 is not needed for the hitting set property, but is only used to make the algorithm for indexing this set
somewhat simpler. One can remove these vectors given an efficient algorithm that indexes $\ell$-subsets of $[n]$, which is not hard to devise based on Pascal’s Identity.

Combining this with inclusion of diagonal depth-3 circuits into somewhere-small-support polynomials (Corollary 8.5.2), we get hitting sets of quasipolynomial size for diagonal depth-3 circuits.

**Corollary 8.5.7.** Let $\mathbb{F}$ be a field with $\geq d$ elements. Then the classes of:

- $n$-variate degree-$(< d)$ diagonal depth-3 circuits of size $\leq s$
- $n$-variate degree-$(< d)$ polynomials $f(x)$ with $\dim \partial_{x<\infty}(f) \leq s$

have $\text{poly}(n, d, \lg s)$-explicit hitting sets of size $(nd)^{\lfloor \lg s \rfloor}$.

---

### 8.6 Reducing Diagonal Depth-4 to roABPs

In this section, we show that small diagonal depth-4 circuits are also computable by small roABPs, and that this conversion is efficient. Using the white-box and black-box PIT algorithms for roABPs of Chapter 6, we can then immediately conclude a polynomial-time white-box PIT and quasipolynomial-time black-box PIT algorithm for diagonal-depth 4 circuits.

The above reduction and black-box conclusion is from Forbes-Shpilka \[FS13b\], but in Saxena’s \[Sax08\] original paper, he was able to conclude a polynomial-time white-box PIT algorithm, although in a non-modular fashion. That is, he proved the following result.

**Theorem 8.6.1** (Saxena \[Sax08\]). Let $\mathbb{F}$ be a field with $|\mathbb{F}| \geq \text{poly}(n, s)$. Then there is an efficient deterministic polynomial time algorithm for white-box PIT of $n$-variate, size-$s$ diagonal depth-4 circuits, in the unit cost model over $\mathbb{F}$.

The proof of the above was via a white-box reduction from diagonal depth-4 circuits to non-commutative PIT over algebras, and then an extension of the Raz-Shpilka \[RS05\] algorithm to this case was applied. Our derivation of the above result will be similar, but more modular. That is, we rederive the above reduction and strengthen it in two ways. The first aspect is that establish the reduction in arbitrary characteristic using an application of the Frobenius automorphism. The second improvement is that we explicitly transform the diagonal depth-4 circuit to a roABP instead of a non-commutative ABP over a large algebra. That is, we follow Saxena’s \[Sax08\] duality idea, but “pull” the complexity of the algebras he works with into the roABP. This allows the standard Raz-Shpilka \[RS05\] algorithm (presented in Section 6.3) to be applied, as well as the hitting set of Corollary 6.5.17 for roABPs, which incurs only a quasipolynomial overhead over the white-box results.

Our reduction will be based on the duality idea of Saxena \[Sax08\] (see his Lemma 2). However, the prior duality statements were cumbersome over fields of positive characteristic. Saxena \[Sax08\] only alluded briefly to this case, and later Saha, Saptharishi and Saxena \[SSS13\] gave a description, arguing that duality holds if one can move from $\mathbb{Z}_p$ to $\mathbb{Z}_{p^N}$ for some large $N$. This is not suitable for black-box reductions, as we cannot change the characteristic in a black-box way. Thus, we
develop a unified duality statement that works for all fields, and also works in the
black-box setting. This duality improves on that of Saxena [Sax08] by examining
individual degree as opposed to total degree, and using this technical improvement we
can extend the result to positive characteristic by using the Frobenius automorphisms.

We note that, as mentioned at the beginning of this chapter, that Kayal [Kay10]
gave an alternate white-box PIT algorithm for diagonal depth-4 circuits, and this was
rephrased by Saptharishi [Sap13] in the language of evaluation dimension as defined in
Subsection 4.5.4. Further, this perspective is agnostic to the characteristic of the
underlying field, so is somewhat cleaner.

In this section, we will make heavy use of the exponential power series 
\[ \exp(x) = \sum_{n \geq 0} \frac{x^n}{n!}, \]
and the truncated exponential denoted by \[ \exp_k(x) = \sum_{n=0}^{k} \frac{x^n}{n!}, \] recalling
the notation of Chapter 1. Further, recall our meaning, stated in Chapter 1, for
extracting coefficients of polynomials, and its meaning when we take coefficients “over
\( \mathbb{F}[\pi][\gamma] \)” (as opposed to “over \( \mathbb{F}[\pi, \gamma] \)).

In particular, we will need the following fact about formal power series (see
Shoup [Sho09] for more on formal power series). Strictly speaking, we will only use
this fact in a truncated form, so one could work entirely with polynomials. However,
this would loose some of the intuition.

**Lemma 8.6.2.** Let \( R \) be a commutative ring, and let \( R[\pi] \) be its multivariate power
series. Then if \( f, g \in R[\pi] \) have no constant terms, then \( \exp(f), \exp(g), \exp(f + g) \)
are defined in \( R[\pi] \), and \( \exp(f) \exp(g) = \exp(f + g) \).

**Proof:** By definition,

\[ \exp(f) = \sum_{n \geq 0} \frac{f^n}{n!}. \]

Thus, consider any monomial \( \pi^n \) in the above expression,

\[ \text{Coeff}_{\pi^n} \exp(f) = \text{Coeff}_{\pi^n} \sum_{n \geq 0} \frac{f^n}{n!} \]
as \( f \) has no constant term, the minimal degree term in \( f^n \) is \( n \),

\[ = \text{Coeff}_{\pi^n} \sum_{0 \leq n \leq \|\pi\|_1} \frac{f^n}{n!}, \]

and this last expression is defined in \( R \), as the summation is finite.

Now consider

\[ \text{Coeff}_{\pi^n} \exp(f + g) = \text{Coeff}_{\pi^n} \sum_{n \geq 0} \frac{(f + g)^n}{n!} \]
and truncating as above, as \( f + g \) has no constant term,

\[
\begin{align*}
&= \text{Coeff}_{x^\pi} \sum_{0 \leq n \leq \|a\|_1} \frac{(f + g)^n}{n!} \\
&= \text{Coeff}_{x^\pi} \sum_{0 \leq n \leq \|a\|_1} \frac{1}{n!} \sum_{i + j = n} \binom{n}{i} f^i g^j \\
&= \text{Coeff}_{x^\pi} \sum_{0 \leq n \leq \|a\|_1} \sum_{i + j = n} \frac{f^i g^j}{i! j!} .
\end{align*}
\]

Thus, if we consider each coefficient of \( \exp(f) \exp(g) \),

\[
\text{Coeff}_{x^\pi} (\exp(f) \exp(g)) = \text{Coeff}_{x^\pi} \left( \left( \sum_{0 \leq n \leq \|a\|_1} \frac{f^n}{n!} \right) \left( \sum_{0 \leq n \leq \|a\|_1} \frac{g^n}{n!} \right) \right)
\]

and discarding high degree terms, as above,

\[
= \text{Coeff}_{x^\pi} \left( \left( \sum_{0 \leq n \leq \|a\|_1} \frac{f^n}{n!} \right) \left( \sum_{0 \leq n \leq \|a\|_1} \frac{g^n}{n!} \right) \right) ,
\]

So that considering the difference \( \exp(f + g) - \exp(f) \exp(g) \), and invoking the above,

\[
\begin{align*}
&\text{Coeff}_{x^\pi} \left( \exp(f) \exp(g) - \exp(f + g) \right) \\
&= \text{Coeff}_{x^\pi} (\exp(f) \exp(g)) - \text{Coeff}_{x^\pi} (\exp(f + g)) \\
&= \text{Coeff}_{x^\pi} \left( \left( \sum_{0 \leq n \leq \|a\|_1} \frac{f^n}{n!} \right) \left( \sum_{0 \leq n \leq \|a\|_1} \frac{g^n}{n!} \right) \right) - \text{Coeff}_{x^\pi} \sum_{0 \leq n \leq \|a\|_1} \sum_{i + j = n} \frac{f^i g^j}{i! j!} \\
&= \text{Coeff}_{x^\pi} \left( \sum_{0 \leq i, j \leq \|a\|_1} \frac{f^i g^j}{i! j!} - \sum_{0 \leq n \leq \|a\|_1} \sum_{i + j = n} \frac{f^i g^j}{i! j!} \right) \\
&= \text{Coeff}_{x^\pi} \left( \sum_{0 \leq i, j \leq \|a\|_1} \frac{f^i g^j}{i! j!} - \sum_{0 \leq i, j \leq \|a\|_1} \frac{f^i g^j}{i! j!} \right) \\
&= \text{Coeff}_{x^\pi} \left( \sum_{0 \leq i, j \leq \|a\|_1} \frac{f^i g^j}{i! j!} \right) \\
&= 0 ,
\end{align*}
\]

where the last line uses that \( f \) and \( g \) have no constant term, so that the summation has degree > \( \|a\|_1 \). Thus \( \exp(f + g) - \exp(f) \exp(g) \) has all zero coefficients, so \( \exp(f + g) = \exp(f) \exp(g) \).

\[\square\]

Note that one cannot define \( \exp(1) \in \mathbb{Q}[x] \) without dealing with convergence issues, so that the above insistence that \( f, g \) have no constant terms is needed.

We now use the above fact to give the details of the reduction from diagonal depth-4 circuits to roABPs, using our extension of Saxena’s [Sax08] duality trick. We
first establish duality over \( \mathbb{Q} \), treating the coefficients \( \bar{c} \) of our polynomials as formal variables.

**Lemma 8.6.3** (Duality over \( \mathbb{Q} \)). Let \( \overrightarrow{f} \) be a vector of polynomials from \( \mathbb{Z}[\overline{x}, \overline{c}] \), where \( f_j(\overline{x}, \overline{c}) = \sum_m f_{j,m}(x_m, \overline{c}) \). Then, taking coefficients in \( \mathbb{Q}[\overline{x}, \overline{c}][z] \),

\[
\frac{1}{a!} \overrightarrow{f} = \text{Coeff}_{z^a} \left( \prod_m \prod_j \text{[exp]}_{a_j} (f_{j,m}(x_m, \overline{c})z_j) \right).
\]

**Proof:** We will work over the power series \( \mathbb{Q}[\overline{x}, \overline{c}][z] \). In particular, for \( f \in \mathbb{Q}[\overline{x}, \overline{c}] \), we have the identity

\[
\frac{1}{a!} f(\overline{x}, \overline{c})^a = \text{Coeff}_{z^a} \exp(f(\overline{x}, \overline{c})z).
\]

Note that this is a formal identity, that is, there is no notion of convergence needed as each coefficient in the power series of \( \exp(f(\overline{x}, \overline{c})z) \) is the sum of finitely many non-zero terms, as argued above in **Lemma 8.6.2**. It follows then that

\[
\frac{1}{a!} \overrightarrow{f}(\overline{x}, \overline{c}) = \prod_j \text{Coeff}_{z^j} \exp(f_j(\overline{x}, \overline{c})z_j)
\]

using that taking variable disjoint coefficients can be merged (using a power series analogue of **Lemma B.1.3**),

\[
= \text{Coeff}_{\overrightarrow{z}} \prod_j \exp(f_j(\overline{x}, \overline{c})z_j)
\]

using the identity \( \exp(f + g) = \exp(f) \exp(g) \) in \( \mathbb{Q}[\overline{x}, \overline{c}][z] \) from **Lemma 8.6.2**, as these polynomials do not have constant terms in \( z \),

\[
= \text{Coeff}_{\overrightarrow{z}} \exp \left( \sum_j f_j(\overline{x}, \overline{c})z_j \right) = \text{Coeff}_{\overrightarrow{z}} \exp \left( \sum_j \sum_m f_{j,m}(x_m, \overline{c})z_j \right)
\]

reversing the order of the (finite) summation,

\[
= \text{Coeff}_{\overrightarrow{z}} \exp \left( \sum_m \sum_j f_{j,m}(x_m, \overline{c})z_j \right)
\]

\[
= \text{Coeff}_{\overrightarrow{z}} \prod_m \prod_j \exp (f_{j,m}(x_m, \overline{c})z_j)
\]

and now discarding terms of degree \( > a_j \) in \( z_j \), as that do not affect the coefficient of \( z^a \),

\[
= \text{Coeff}_{\overrightarrow{z}} \prod_m \prod_j \text{[exp]}_{a_j} (f_{j,m}(x_m, \overline{c})z_j)
\]

\[\square\]
Even though the above proof used the exponential function, which has no exact analogue in positive characteristic, the final statement is just one of polynomials, and thus transfers to any field of sufficiently large characteristic. In particular, the characteristic must be > ||π||\_∞, the maximum exponent, as then all constants in the above duality are defined. We now transfer this duality to any field. We will abuse notation and treat the characteristic zero case as working over characteristic \( p \), where \( p = \infty \). Note that the “base-∞” expansion of any integer is just that integer itself, followed by zeroes.

**Lemma 8.6.4** (Duality over any \( \mathbb{F} \)). Let \( \mathbb{F} \) be a field of characteristic \( p \) (for characteristic zero, we abuse notation and use \( p = \infty \)). Let \( \overline{f^x} \in \mathbb{F}[\overline{x}] \), where \( f_j(\overline{x}) = \sum_m f_{j,m}(x_m) \).

Write \( a_j \) in its base-\( p \) expansion, so that \( a_j = \sum_{k \geq 0} b_{j,k} p^k \) with \( b_{j,k} \in \mathbb{Z} \). Denote \( g_{j,k}(\overline{x}) := \sum_m f_{j,m}(x_m) p^k \) and \( \overline{g^x} := \prod_j \prod_k b_{j,k} \). Then, \( \overline{f^x} = \overline{g^x} \) and

\[
\frac{1}{\partial |\overline{g^x}|} = \text{Coeff}_{\overline{z^x}} \left( \prod_{m,j,k} \left[ \exp \left( \frac{f_{j,m}(x_m) p^k}{b_{j,k}} \right) \right] \right),
\]

and ||\overline{b}||_\infty \leq ||\overline{\pi}||_\infty.

**Proof:** \( \overline{f^x} = \overline{g^x} \). This follows from the Frobenius automorphism, that \( (x + y)^p = x^p + y^p \) for any \( k \geq 0 \), in any field of characteristic \( p \). That is,

\[
\overline{f^x} = \prod_j f_j^{a_j} = \prod_j f_j^{\sum_{k \geq 0} b_{j,k} p^k} = \prod_j \prod_k f_j^{b_{j,k} p^k} = \prod_j \prod_k (f_j^p)^{b_{j,k}}
\]

\[
= \prod_j \prod_k \left( \sum_m f_{j,m}(x_m) \right)^{b_{j,k}} = \prod_j \prod_k g_{j,k} = \overline{g^x}.
\]

(8.6.5): Note that the identity of Lemma 8.6.3 applied to an exponent \( \overline{b} \) only involves rational numbers whose denominators \( \leq ||\overline{b}||_\infty \), the maximum exponent. Thus, as ||\overline{b}||_\infty < p by construction, Lemma 8.6.3 is also an identity over \( \mathbb{Z}_p[\overline{\pi}, \overline{z}] \) when applied with the exponent vector \( \overline{b} \), and any \( g \) polynomials. In particular, we can substitute values for the constants \( \overline{z} \), implying that we can take the \( g \) polynomials to be polynomials in \( \overline{\pi} \) with constants from any field extending \( \mathbb{Z}_p \), such as \( \mathbb{F} \). Thus, the identity can be applied to \( \overline{g^x} \) over \( \mathbb{F} \), yielding the result.

\( ||\overline{b}||_\infty \leq ||\overline{\pi}||_\infty \): As \( ||\overline{\pi}||_\infty = \prod_j (a_j + 1) \) and \( ||\overline{b}||_\infty = \prod_j \prod_{k \geq 0} (b_{j,k} + 1) \), it is sufficient to prove that \( \prod_{k \geq 0} (b_k + 1) \leq 1 + a \), for \( a = \sum_{k \geq 0} b_k p^k \) with \( b_k \in \mathbb{Z} \). To see this, first define \( T := \{ a' : 0 \leq a' \leq a \} \) and note that \( |T| = 1 + a \). Second, let \( S = \{ a' : a' = \sum b'_k p^k, 0 \leq b'_k \leq b_k \} \). Observe that \( S \subseteq T \), and \( |S| = \prod_k (b_k + 1) \), so \( \prod_k (b_k + 1) = |S| \leq |T| = 1 + a \). □
The above results show how to take a diagonal depth-4 circuit, such as \((f_1(x_1) + \cdots + f_n(x_n))^d\), and convert it into (the coefficient of) a product where the \(x_i\)'s only interact via variable disjoint products. As roABPs inherently compute in a variable disjoint manner, this is making substantial progress toward the desired reduction to roABPs. To complete this reduction, we need to show how one can extract coefficients from roABPs in an efficient manner, which we now establish in a generic way.

**Lemma 8.6.6.** Let \(R\) be a non-commutative ring. Let \(d \geq 1\). Let \(L_1, \ldots, L_d \subseteq R[z]\). Let \(F \in R[z]^{n \times m}\) be a computed by a width-\(w\) layered \(R[z]\)-ABP with labels \(L_1, \ldots, L_d\). Then for any exponent vector \(\pi\), \(\text{Coeff}_{\pi} (F)\) is computable by a width-(\(\leq w \cdot \|\pi\|_x\)) layered \(R\)-ABP with labels \(L'_1, \ldots, L'_d\), where \(L'_i := \{\text{Coeff}_{\pi}(\ell) \mid b \leq a, \ell \in L_i\}\), where we use the natural partial order on vector exponents.

Further, the layered \(R\)-ABP can be computed from the layered \(R[z]\)-ABP in \(\text{poly}(\|\pi\|_x, d, w)\) operations in \(O(\log(\|\pi\|_x wd))\) parallel time, when we assume the labels \(L'_i\) are provided as input, and where labels in \(R\) can be manipulated only by copying (without inspection), by setting to the constant 0, or testing of a label is 0.

**Proof:** By Lemma 4.2.4, it follows that there are matrices \(M_i \in (L_i \cup \{0\})^{w \times w}\) such that in block notation,

\[
\begin{bmatrix}
F & 0 \\
0 & 0
\end{bmatrix} = \prod_{i \in [d]} M_i,
\]

where 0 is the zero matrix with the appropriate dimensions in each case.

Let \(S_{\pi} := \{|b| \mid b \leq b \leq \pi\}\), so that \(|S_{\pi}| = \|\pi\|_x\). Define the block-matrix \(M'_i \in ((L'_i \cup \{0\})^{w \times w})^{S_{\pi} \times S_{\pi}}\) as follows,

\[
(M'_i)_{b, b'} := \begin{cases}
\text{Coeff}_{\pi_{\pi} \pi'}(M_i) & \pi_{\pi} \geq b' \\
0_{w \times w} & \pi_{\pi} \nleq b'
\end{cases}
\]

noting that “\(\geq\)” is not a total order in this usage. From this construction, it follows that \(M'_i\) are \((w\|\pi\|_x) \times (w\|\pi\|_x)\) matrices over \(L'_i \cup \{0\}\).

Now consider \(\prod_i M'_i\), where we analyze this product in block-matrix notation. By construction, we have that

\[
\left(\prod_i M'_i\right)_{\pi_0, \pi_d} = \sum_{\pi_0, \pi_1, \ldots, \pi_d} (M'_i)_{\pi_0, \pi_i} \cdots (M'_d)_{\pi_{d-1}, \pi_d}
\]

as \((M'_i)_{\pi_{i-1}, \pi_i} = 0\) if \(\pi_i \nleq \pi_{i-1}\),

\[
= \sum_{\pi_0 \leq \pi_1 \leq \cdots \leq \pi_d} (M'_i)_{\pi_0, \pi_1} \cdots (M'_d)_{\pi_{d-1}, \pi_d}
\]

\[
= \sum_{\pi_0 \leq \pi_1 \leq \cdots \leq \pi_d} \text{Coeff}_{\pi_1 - \pi_0} (M_1) \cdots \text{Coeff}_{\pi_d - \pi_{d-1}} (M_d)
\]

150
and re-indexing, via \( \overline{b}_i = \overline{a}_i - \overline{a}_{i-1} \geq 0 \),

\[
\begin{align*}
\sum_{\overline{b}_1 + \cdots + \overline{b}_d = \overline{a}_d - \overline{a}_0} \text{Coeff}_{\overline{z}_1} (M_1) \cdots \text{Coeff}_{\overline{z}_d} (M_d) \\
= \text{Coeff}_{\overline{z}_d - \overline{a}_0} (M_1 \cdots M_d) .
\end{align*}
\]

Thus, it follows that

\[
\left( \prod_i M'_i \right)_{\overline{a}, \overline{a}} = \text{Coeff}_{\overline{z}} (M_1 \cdots M_d) = \begin{bmatrix} \text{Coeff}_{\overline{z}} (F) & 0 \\ 0 & 0 \end{bmatrix} ,
\]

from which it follows that \( \text{Coeff}_{\overline{z}} (F) \) is a submatrix of \( \prod_i M'_i \), so appealing to Lemma 4.2.4 shows that \( \text{Coeff}_{\overline{z}} (F) \) is computable by the layered \( R \)-ABP as desired.

**Explicitness:** This is clear from construction, as each label in the layered \( R \)-ABP is constructed from some \( \text{Coeff}_{\overline{z}} (\ell) \) for some label \( \ell \) in the layered \( R[\overline{z}] \)-ABP, where \( \overline{b} \leq \overline{a} \), and this construction has the needed uniformity.

We now apply this structural result, along with Saxena’s dual form, to show that diagonal depth-4 circuits can be computed by small roABPs.

**Lemma 8.6.7** (Diagonal Depth-4 reduces to roABP). Let \( \mathbb{F} \) be any field. Let \( f(x_1, \ldots, x_n) \) be computed by a diagonal depth-4 circuit, so that

\[
f(\overline{x}) = \sum_{i=1}^a \overline{f}_i(\overline{x})^{a_i} ,
\]

where each \( \overline{f}_i \) is a sum of univariate polynomials, so that \( \overline{f}_i(\overline{x}) = \sum_{j \in [n]} \overline{f}_{i,j}(x_j) \), and where \( \deg \overline{f}_i \leq d \).

Then for any variable order \( \pi : [n] \to [n] \), \( f \) is computed by a \( \text{poly}(\sum_{i \in [a]} \|\overline{a_i}\|_x, n, d) \)-explicit width-\( \left( \sum_{i \in [a]} \|\overline{a_i}\|_x \right) \) roABP, of individual degree \( \leq (d \max_{i \in [a]} \|\overline{a_i}\|_1) \), in the variable order \( \pi \).

In particular, \( \dim \text{Eval}(f) \leq \sum_{i \in [a]} \|\overline{a_i}\|_x \).

**Proof:** We prove the claim when the variable order is the identity permutation, as the definition of a diagonal depth-4 circuit is invariant under permutation, so other permutations are handled symmetrically.

Let \( p \) denote the characteristic of \( \mathbb{F} \), with the convention that \( p = \infty \) for characteristic zero fields. By the dual form, Lemma 8.6.4, we have that for each \( i \),

\[
\overline{f}_i^{\overline{a}_i} = \overline{b}_i! \cdot \text{Coeff}_{\overline{z}_i} \left( \prod_m h_{i,m}(x_m, \overline{z}) \right) ,
\]

where \( \overline{b}_i \) is the base-\( p \) expansion of \( \overline{a}_i \) (so that \( \overline{b}_i! \neq 0 \), and \( \|\overline{b}_i\|_x \leq \|\overline{a}_i\|_x \)), and \( h_{i,m} \in \mathbb{F}[x_m, \overline{z}] \) has \( \deg x_m h_{i,m} \leq d \|\overline{a}_i\|_1 \). It follows that \( \prod_m h_{i,m}(x_m, \overline{z}) \) can be computed using a width-1 roABP over \( \mathbb{F}[\overline{z}][\overline{z}] \) with individual degree \( \leq (d \|\overline{a}_i\|_1) \), in the standard variable order \( x_1 < \cdots < x_m \).
Viewing this roABP for \( \prod_m h_{i,m} \) as a layered \( R[z]-\text{ABP} \) over where \( R = \mathbb{F}[\pi] \), we can then apply Lemma 8.6.6 to extract the coefficient \( z^{\pi_i} \) of \( \prod_m h_{i,m} \). In this layered \( R[z]-\text{ABP} \), the label set for the \( m \)-th layer are polynomials \( \mathbb{F}[x_m][z] \) of \( x_m \)-degree \( \leq d \| \pi_i \|_1 \). Thus, the claim produces a layered \( R \)-ABP for \( f_{\pi_i} \) of width \( \| \pi_i \|_x \), where the label set of the \( m \)-th layer are polynomials in \( \mathbb{F}[x_m] \) of degree \( \leq d \| \pi_i \|_1 \). Thus, \( f_{\pi_i} \) has a width-\( \| \pi_i \| \times \text{roABP} \) of individual degree \( \leq d \| \pi_i \|_1 \) in the variable order \( x_1 < \cdots < x_n \).

Applying that roABPs (in a fixed variable order) are closed (allowing suitable parameter growth) under addition (Corollary 4.4.4) yields the claim.

evaluation dimension: This follows from the characterization of evaluation dimension as the maximization over the width needed to compute a polynomial as roABP over all variable orders (Corollary 4.5.14).

explicitness: This is clear from construction (and using that \( \| \pi_i \|_1 \leq \| \pi_i \|_x \) (Lemma 8.2.3)), as the dual form (Lemma 8.6.4) has the desired explicitness, as does coefficient extraction (Lemma 8.6.6).

The above shows that diagonal depth-4 circuits can efficiently be transformed to roABPs of essentially the same size. As an immediate corollary, we can transfer the lower bounds for roABPs (Corollary 4.5.10) to lower bounds for diagonal depth-4 circuits, noting that such a result was already essentially observed by Saxena [Sax08]. For clarity, we state the result in terms of the size of the diagonal depth-4 circuit.

**Corollary 8.6.8** (Lower Bound for Diagonal Depth-4 Circuits). Consider \( f(\pi, \bar{y}) \in \mathbb{F}[x_1, \ldots, x_n, y_1, \ldots, y_n] \) defined by \( f(\pi, \bar{y}) = \prod_i (x_i + y_i) \). Then computing \( f \) as a diagonal depth-4 circuit requires size \( \geq 2^n \).

We now invoke the relevant PIT results for roABPs to conclude PIT for diagonal depth-4 circuits. First, by invoking the white-box algorithm of Raz-Shpilka [RS05] presented in Section 6.3, we rederive Saxena’s [Sax08] result for white-box PIT of diagonal depth-4 circuits, but now we obtain a result over any field.

**Corollary 8.6.9** (Poly-time White-Box PIT for Diagonal Depth-4 Circuits). Let \( \mathbb{F} \) be any field. There is an efficient deterministic polynomial time algorithm for white-box PIT of \( n \)-variate, size-\( s \) diagonal depth-4 circuits, in the unit cost model over \( \mathbb{F} \).

Similarly, we can invoke the hitting sets for roABPs (Corollary 6.5.17) to obtain hitting sets for diagonal depth-4 circuits. Again, we state the result in terms of size (and now use that the size, as defined, bounds the size of the outputted roABP).

**Corollary 8.6.10** (Quasipolynomial-time White-Box PIT for Diagonal Depth-4 Circuits). Let \( \mathbb{F} \) be any field. Let \( n, s \geq 1 \). Let \( \mathcal{C} \) be the set of polynomials \( f(\pi) \in \mathbb{F}[x_1, \ldots, x_n] \) computable by a size-\( s \) diagonal depth-4 circuit. If \( |\mathbb{F}| > \text{poly}(n, s) \), then \( \mathcal{C} \) has a \( \text{poly}(n, s) \)-explicit hitting set \( \mathcal{H} \subseteq \mathbb{F}^n \) of size \( \text{poly}(n, s)^{[\lg n] + 1} \).

**Remark 8.6.11.** We remark here that the work of Agrawal, Saha and Saxena [ASS13] (concurrent and independent to Forbes-Shpilka [FS13b]) also obtain a quasi-polynomial hitting set for diagonal depth-4 circuits (when they assume each product gate is over
a constant number of factors), but only over large characteristic, as they rely on the
duality statements of Saxena [Sax08] (and later exposited by Saha, Saptharishi, and
Saxena [SSS13]) which only hold over large characteristic. Over small characteristic,
Saxena [Sax08] and Saha-Saptharishi-Saxena [SSS13] gave more cumbersome duality
statements working over prime-power characteristic, and Agrawal-Saha-Saxena [ASS13]
do not extend their work to this case.

Our duality statements work any characteristic, and as such can show that the
work of Agrawal-Saha-Saxena [ASS13] also implies results for diagonal circuits over
any characteristic. In particular, instead of using Lemma 8.6.4 to construct a roABP,
one can interpolate the coefficient $\mathbf{z}^5$ in Lemma 8.6.4, and this can be done in depth-3,
although the resulting formula is inherently larger than the resulting roABP would be.
Chapter 9

Explicit Noether Normalization for Simultaneous Conjugation

Abstract: While the rest of this thesis was about developing PIT algorithms (and black-box algorithms in particular), this chapter is about applying them. In particular, Mulmuley [Mul12a] made an interesting connection between black-box PIT and the Noether Normalization Lemma from commutative algebra. That lemma is a form of dimension reduction for varieties, so that low-dimensional varieties in a large-dimensional ambient space can be mapped into an ambient space matching their dimension. Noether Normalization refers to the process of finding such a dimension reduction, and in fact, a random linear map suffices. Mulmuley [Mul12a] then raised the challenge of finding explicit dimension reduction maps computationally, especially when the variety is explicit. Further, he observed that this task, in full generality, is equivalent to a (strong) form of black-box PIT.

Even further, he suggested that explicit Noether Normalization, even for some extremely simple varieties, might be challenging, due to some superficial similarity with wild problems from representation. In this chapter we give the results of Forbes-Shpilka [FS13a], and consider the primary example he raised, simultaneous conjugation of matrices, and show that this variety does not need the full power of PIT for its Noether Normalization, and rather that PIT for roABPs suffices. By applying the hitting sets of Corollary 6.5.17 for roABPs, we obtain explicit Noether Normalization for this variety (albeit of quasipolynomial size). This shows that this specific variety is not a hard case of Noether Normalization (while leaving the rest of the problem untouched).

9.1 Introduction

Many results in mathematics are non-constructive, in the sense that they establish that certain mathematical objects exist, but do not give an efficient or explicit construction of such objects, and often further work is needed to find constructive arguments. Motivated by the recent results of Mulmuley [Mul12a] (referred to in this chapter as “Mulmuley”, but theorem and page numbering will refer to the third arXiv
version, Mulmuley [Mul12b]), this paper studies constructive versions of the Noether Normalization Lemma from commutative algebra. The lemma, as used in this paper, can be viewed as taking a commutative ring $R$, and finding a smaller subring $S \subseteq R$ such that $S$ captures many of the interesting properties of $R$ (see Subsection 9.1.1 for a formal discussion). Like many arguments in commutative algebra, the usual proof of this lemma does not focus on the computational considerations of how to find (a concise representation of) the desired subring $S$. However, the area of computational commutative algebra (eg, [DK02,CLO07]) offers methods showing how many classic results can be made constructive, and in certain cases, the algorithms are even efficient.

While constructive methods for Noether Normalization are known using Gröbner bases (cf. [CLO07]), the Gröbner basis algorithms are not efficient in the worst-case (as show by Mayr and Meyer [MM82]), and are not known to be more efficient for the problems we consider. Diverging from the Gröbner basis idea, Mulmuley recently observed that a constructive version of Noether Normalization is really a problem in derandomization. That is, given the ring $R$, if we take a sufficiently large set of “random” elements from $R$, then these elements will generate the desired subring $S$ of $R$. Indeed, the usual proof of Noether Normalization makes this precise, with the appropriate algebraic meaning of “random”. This view suggests that random sampling from $R$ is sufficient to construct $S$, and this sampling will be efficient if $R$ itself is explicitly given. While this process uses lots of randomness, the results of the derandomization literature in theoretical computer science (eg, [IW97, IKW02, KI04]) give strong conjectural evidence that randomness in efficient algorithms is not necessary. Applied to the problems here, there is thus strong conjectural evidence that the generators for the subring $S$ can be constructed efficiently, implying that the Noether Normalization Lemma can be made constructive.

Motivated by this connection, Mulmuley explored what are the minimal derandomization conjectures necessary to imply an explicit form of Noether Normalization. The existing conjectures come in two flavors. Most derandomization hypotheses concern boolean computation, and as such are not well-suited for algebraic problems (for example, a single real number can encode infinitely many bits), but Mulmuley does give some connections in this regime. Other derandomization hypotheses directly concern algebraic computation, and using them Mulmuley gives an explicit Noether Normalization Lemma, for some explicit rings of particular interest. In particular, Mulmuley proves that it would suffice to derandomize the polynomial identity testing (PIT) problem in certain models (see Section 3.2 for background), in order to obtain a derandomization of the Noether Normalization Lemma. Mulmuley actually views this connection as an evidence that derandomizing PIT for these models is a difficult computational task (Mulmuley, p. 3) and calls this the GCT chasm. Although Mulmuley conjectures that it could be crossed he strongly argues that this cannot be achieved with current techniques (Mulmuley, p. 3):

On the negative side, the results in this article say that black-box derandomization of PIT in characteristic zero would necessarily require proving, either directly or by implication, results in algebraic geometry that seem impossible to prove on the basis of the current knowledge.
Mulmuley supports this intuition with his Theorem 1.6, which gives an equivalence between the general PIT problem (considered difficult in complexity theory) and explicit Noether Normalization for explicit varieties (considered difficult in algebraic geometry). However, he also focuses on Noether Normalization of specific varieties, such as the ring of invariants of matrices under simultaneous conjugation. The conditional derandomization of Noether Normalization for this ring is the content of his Theorems 1.1 and 1.2. Explicit Noether Normalization for this ring is simpler as there are explicitly known generators for the ring of invariants (as given in previous work, cited in Theorem 9.1.3), and these generators are computationally very simple. For general explicit varieties, obtaining such explicit generators is an open problem, and even if found, the generators would not likely be as computational simple as the generators of Theorem 9.1.3.

However, Mulmuley has suggested that even the results in his Theorems 1.1 and 1.2 are beyond current techniques, because of their superficial similarity to the “wild” classification problem of matrix tuples under simultaneous conjugation (see Mulmuley, p. 5, and the public presentation [Mul12c]). Indeed, the general Noether Normalization problem for explicit varieties was not discussed in the public presentation, as the specific case of simultaneous conjugation was itself implied to contain the “seemingly impossible” problems mentioned above.

In this work, we obtain (via existing techniques) an unconditional derandomization of Noether’s Normalization Lemma for simultaneous conjugation, as conditionally established in Mulmuley’s Theorems 1.1 and 1.2. While we have no results for explicit Noether Normalization of general explicit varieties, we believe that our work suggests that problems cannot be assumed to be difficult just because they originated in algebraic-geometric setting, and that one has to consider the finer structure of the problem. In particular, our work suggests that the similarity of the problem we consider to the above wild problem is indeed superficial.

In addition, just as Mulmuley’s techniques extend to Noether Normalization of arbitrary quivers (Mulmuley’s Theorem 4.1), our results also extend to this case, but we omit the straightforward details.

9.1.1 Noether Normalization for the Invariants of Simultaneous Conjugation

Mulmuley showed that when $R$ is a particular ring, then the problem of finding the subring $S$ given by Noether Normalization can be reduced to the black-box PIT problem, so that explicit hitting sets (of small size) would imply a constructive version of Noether Normalization for this ring. The ring considered here and in Mulmuley’s Theorems 1.1 and 1.2 is the ring of invariants of matrices, under the action of simultaneous conjugation.

**Remark 9.1.1.** Mulmuley considers conjugation by matrices with determinant 1, that is, by $\text{SL}_n$, whereas we consider conjugation in $\text{GL}_n$. However, they are the same, as seen by the following argument.

Note that a matrix $M \in \text{GL}_n$ iff $1/\det M \cdot M \in \text{SL}_n$. Two matrices $M, N \in \text{GL}_n$ are
conjugate in \( \text{GL}_n \) iff there is a \( P \in \text{GL}_n \) so that \( PMP^{-1} = N \). Further, by taking \( P \leftarrow 1/\det P \cdot P \), we can assume \( P \in \text{SL}_n \).

In particular, that \( M \) and \( N \) in \( \text{GL}_n \) are conjugate implies that \( \det M = \det N \). Thus, \( M, N \in \text{GL}_n \) are conjugate in \( \text{GL}_n \) iff \( 1/\det M \cdot M \) and \( 1/\det N \cdot N \) are conjugate in \( \text{SL}_n \).

We now define the action of simultaneous conjugation.

**Definition 9.1.2.** Let \( \mathbf{X} \) denote a vector of \( r \) matrices, each \( n \times n \), whose entries are distinct variables. Consider the action of \( \text{GL}_n(\mathbb{F}) \) by simultaneous conjugation on \( \mathbf{X} \), that is,

\[
(X_1, \ldots, X_r) \mapsto (PX_1P^{-1}, \ldots, PX_rP^{-1}).
\]

Define \( \mathbb{F}[\mathbf{X}]^{\text{GL}_n(\mathbb{F})} \) to be the subring of \( \mathbb{F}[\mathbf{X}] \) consisting of polynomials in the entries of \( \mathbf{X} \) that are invariant under the action of \( \text{GL}_n(\mathbb{F}) \). That is,

\[
\mathbb{F}[\mathbf{X}]^{\text{GL}_n(\mathbb{F})} := \{ f \mid f(\mathbf{X}) = f(P\mathbf{X}P^{-1}), \forall P \in \text{GL}_n(\mathbb{F}) \}.
\]

Thus, \( \mathbb{F}[\mathbf{X}] \) is a ring of polynomials in \( rn^2 \) variables. \( \mathbb{F}[\mathbf{X}]^{\text{GL}_n(\mathbb{F})} \) is itself a ring (as can be checked), and thus a subring of \( \mathbb{F}[\mathbf{X}] \). While \( \mathbb{F}[\mathbf{X}] \) treats the variables simply as un-grouped variables, \( \mathbb{F}[\mathbf{X}]^{\text{GL}_n(\mathbb{F})} \) only contains polynomials that “recognize” that the variables in \( \mathbf{X} \) are grouped into matrices.

When \( \mathbb{F} \) has characteristic zero, the following result gives an explicit set of generators for the ring of invariants. When \( \mathbb{F} \) has positive characteristic, the result is known not to hold (see [KP00, §2.5]) so we will only discuss characteristic zero fields.

**Theorem 9.1.3** (Generators for the Ring of Invariants of Simultaneous Conjugation [Pro76, Raz74, For86]). Let \( \mathbb{F} \) be a field of characteristic zero. Let \( \mathbf{X} \) denote a vector of \( r \) matrices, each \( n \times n \), whose entries are distinct variables. The ring \( \mathbb{F}[\mathbf{X}]^{\text{GL}_n(\mathbb{F})} \) of invariants is generated by the invariants \( \mathcal{I}_{n,r}^{\text{PRF}} := \{ \text{tr}(X_{i_1} \cdots X_{i_\ell}) \mid \ell \in [r], \ell \in [n^2] \} \).

Further, the ring \( \mathbb{F}[\mathbf{X}]^{\text{GL}_n(\mathbb{F})} \) is not generated by the invariants \( \{ \text{tr}(X_{i_1} \cdots X_{i_\ell}) \mid \ell \in [r], \ell \in \lceil n^2/8 \rceil \} \).

That is, every invariant can be represented as a (multivariate) polynomial, with coefficients in \( \mathbb{F} \), in the above generating set. Note that it is straightforward to see that these trace polynomials are elements of \( \mathbb{F}[\mathbf{X}]^{\text{GL}_n(\mathbb{F})} \), as we now record as a lemma.

**Lemma 9.1.4.** Let \( \mathbf{X} \) denote a vector of \( r \) matrices, each \( n \times n \). Let \( f(\mathbf{X}) = \text{tr}(X_{i_1} \cdots X_{i_\ell}) \in \mathbb{F}[\mathbf{X}] \), for some \( \ell \geq 1 \) and \( \ell \in [r] \). Then for any \( P \in \text{GL}_n(\mathbb{F}) \), \( f(\mathbf{X}) = f(P\mathbf{X}P^{-1}) \), so that \( f \in \mathbb{F}[\mathbf{X}]^{\text{GL}_n(\mathbb{F})} \).

**Proof:**

\[
f(P\mathbf{X}P^{-1}) = \text{tr}((PX_{i_1}P^{-1}) \cdot (PX_{i_2}P^{-1}) \cdots (PX_{i_\ell}P^{-1}))
\]

canceling \( P^{-1}P \) terms,

\[
= \text{tr}(P \cdot X_{i_1} \cdot X_{i_2} \cdots X_{i_\ell} \cdot P^{-1})
\]

158
as the trace is cyclic, so that \( \text{tr} ABC = \text{tr} BCA \) for any matrices \( A, B, C \),

\[
\text{tr}(P^{-1} \cdot P \cdot X_{i_1} \cdot X_{i_2} \cdots X_{i_\ell})
\]

and canceling \( P^{-1}P \) terms again,

\[
\text{tr}(X_{i_1} \cdot X_{i_2} \cdots X_{i_\ell}) = f(\overline{X}) \quad \square
\]

In particular, this verifies that \( \mathcal{I}_{n,r}^{\text{PRF}} \subseteq \mathbb{F}[\overline{X}]^{\text{GL}_n(\mathbb{F})} \).

The above theorem yields explicit generators in two senses. The first sense is that all involved field constants can be efficiently constructed. The second is that for any \( f \in \mathcal{I}_{n,r}^{\text{PRF}} \) and \( A \),

\[
f(A) = \text{tr}(X_{i_1} \cdot X_{i_2} \cdots X_{i_\ell})
\]

Lemma 9.1.5. Let \( \overline{X} \) denote a vector of \( r \) matrices, each \( n \times n \). Let \( \overline{i} \in [r]^\ell \), for some \( \ell \geq 1 \), and let \( f_\overline{i}(\overline{X}) := \text{tr}(X_{i_1} \cdots X_{i_\ell}) \in \mathbb{F}[\overline{X}] \). Then \( f_\overline{i}(\overline{X}) \) can be computed by a \( \text{poly}(n, \ell, r) \)-explicit layered ABP of \( \mathbb{F}[\overline{X}] \) with depth-\( \ell \) and width \( \leq n^2 \).

Proof: Note that \( f_\overline{i} = \sum_{j=1}^{n} (X_{i_1} \cdots X_{i_\ell})_{jj} \). By Corollary 4.2.8, it follows that each matrix \( (X_{i_1} \cdots X_{i_\ell})_{jj} \) can be computed by a width-\( n \), depth-\( \ell \) layered ABP that is \( \text{poly}(n, \ell, r) \)-explicit. By Corollary 4.2.9 it then follows that their sum \( f_\overline{i} \) is computable by a width-\( n^2 \), depth-\( \ell \) layered ABP that is \( \text{poly}(n, \ell, r) \)-explicit. \( \square \)

It follows from the above that the set \( \mathcal{I}_{n,r}^{\text{PRF}} \) has \( \text{poly}(n, r) \)-explicit circuits (as layered ABPs have small circuits, Fact 4.2.10), as each polynomial in \( \mathcal{I}_{n,r}^{\text{PRF}} \) is indexed by an element of \([r]^\ell\) for \( \ell \leq n^2 \).

We encapsulate these notions in the following definition.

Definition 9.1.6. Let \( \mathcal{C} \) be a class of (possibly restricted) algebraic circuits. A set \( \mathcal{P} \subseteq \mathbb{F}[x_1, \ldots, x_n] \) of polynomials has \( t(n) \)-explicit \( \mathcal{C} \)-circuits, if there is an algorithm such that given an index into \( \mathcal{P} \), a circuit \( C \in \mathcal{C} \) can be computed in \( t(n) \)-time, assuming unit cost arithmetic in \( \mathbb{F} \), such that \( C \) computes the indexed \( f \in \mathcal{P} \).

Remark 9.1.7. As this chapter is about applications in mathematics, it is important to contrast the above notion of explicit with the vague notion of being mathematically explicit. For example, a non-trivial \( n \)-th root of unity is mathematically explicit, but is not computationally explicit from the perspective of the rational numbers. In particular, such an \( n \)-th root cannot be computed efficiently in the unit-cost model over \( \mathbb{C} \), as defined in Appendix A.

Conversely, the lexicographically first function \( f : \{0, 1\}^{\lceil \log \log n \rceil} \to \{0, 1\} \) with the maximum possible boolean circuit complexity among all functions on \( \lceil \log \log n \rceil \) bits, is computationally explicit, but arguably not mathematically explicit.

We will restrict ourselves the notion of computational explicitness, as it has a well-defined notion. However, all of the results of this work are also arguably mathematically explicit. \( \diamond \)
In particular, the above definition implies that the resulting circuits $C$ have size at most $t(n)$. The class of circuits $\mathcal{C}$ can be the class of all algebraic circuits, or some restricted notion, such as algebraic branching programs (defined in Section 4.1). Thus, in the language of the above definition, the set of generators $\mathcal{I}_{n,r}^{\text{PRF}}$ has $\text{poly}(n, r)$-explicit algebraic circuits.

However, the above result is unsatisfactory in that the set of generators $\mathcal{I}_{n,r}^{\text{PRF}}$ has size $\exp(\text{poly}(n, r))$, which is unwieldy from a computational perspective. One could hope to find a smaller subset of generators, but the lower bound in the above theorem rules out one approach that direction. The number of generators is relevant here, as we will consider three computational problems where these generators are useful, but because of their number the resulting algorithms will be exponential-time, where one could hope for something faster. To define these problems, we first give the following standard definition from commutative algebra.

**Definition 9.1.8.** Let $R$ be a commutative ring, and $S$ a subring. Then $R$ is integral over $S$ if every element in $R$ satisfies some monic polynomial with coefficients in $S$. ♦

As an example, the algebraic closure of $\mathbb{Q}$ (the algebraic numbers) is integral over $\mathbb{Q}$. In this chapter the rings $R$ and $S$ will be rings of polynomials, and it is not hard to see that all polynomials in $R$ vanish at a point iff all polynomials in $S$ vanish at that point. This can be quite useful, especially if $S$ has a small list of generators. The statement of Noether Normalization is exactly that of providing such an $S$ with a small list of generators. The question we consider here is how to find an explicit such $S$ for the ring of invariants under simultaneous conjugation, where $S$ should be given by its generators.

**Question 9.1.9 (Explicit Noether Normalization).** Let $\mathbb{F}$ be an algebraically closed field of characteristic zero. Is there a small set of polynomials $\mathcal{I}_{n,r} \subseteq \mathbb{F}[X]^{\text{GL}_n(\mathbb{F})}$ with explicit algebraic circuits, such that $\mathbb{F}[X]^{\text{GL}_n(\mathbb{F})}$ is integral over the subring $S$ generated by $\mathcal{I}_{n,r}$? ♦

We will in fact mostly concern ourselves with the next problem, which has implications for the first, when $\mathbb{F}$ is algebraically closed. We first give the following definition, following Derksen and Kemper [DK02].

**Definition 9.1.10.** A subset $\mathcal{I}_{n,r} \subseteq \mathbb{F}[X]^{\text{GL}_n(\mathbb{F})}$ is a set of separating invariants if for all $\mathcal{A}, \mathcal{B} \in (\mathbb{F}[X]^{[n \times n]})^{[r]}$ there exists an $f \in \mathbb{F}[X]^{\text{GL}_n(\mathbb{F})}$ such that $f(\mathcal{A}) \neq f(\mathcal{B})$ iff there exists an $f' \in \mathcal{I}_{n,r}$ such that $f'(\mathcal{A}) \neq f'(\mathcal{B})$. ♦

In words, whenever there is an $f \in \mathbb{F}[X]^{\text{GL}_n(\mathbb{F})}$ separating $\mathcal{A}$ and $\mathcal{B}$ (that is, $f(\mathcal{A}) \neq f(\mathcal{B})$), there is also an $f' \in \mathcal{I}_{n,r}$ separating them. As before, we will ask whether we can find an explicit construction.

**Question 9.1.11.** Let $\mathbb{F}$ have characteristic zero. Is there a small set of separating invariants $\mathcal{I}_{n,r} \subseteq \mathbb{F}[X]^{\text{GL}_n(\mathbb{F})}$ with explicit algebraic circuits? ♦

Mumlpuley used the tools of geometric invariant theory [MFK94], as done in Derksen and Kemper [DK02], to note that a construction of separating invariants automatically gives the desired integral subring. We only state this in our specific case, but the result does hold more generally.
Theorem 9.1.12 (Separating invariants yield small integral subring, Theorem 2.3.12, Derksen and Kemper [DK02], stated by Mulmuley in Theorem 2.11). Let \( \mathbb{F} \) be an algebraically closed field of characteristic zero. Let \( \mathcal{I}_{n,r} \subseteq \mathbb{F}[\mathbf{X}]^{\text{GL}_n(\mathbb{F})} \) be a finite set of homogeneous separating invariants. Then \( \mathbb{F}[\mathbf{X}]^{\text{GL}_n(\mathbb{F})} \) is integral over the subring \( S \) generated by \( \mathcal{I}_{n,r} \).

Thus, any positive answer to Question 9.1.11 will give a positive answer to Question 9.1.9. Hence, we will focus on constructing explicit separating invariants (over any field of characteristic zero).

Note that relaxations of Question 9.1.11 can be answered positively. If we only insist on explicit separating invariants (relaxing the insistence on having few invariants), then the exponentially-large set of generators \( \mathcal{I}_{PRF}^{n,r} \) given in Theorem 9.1.3 suffices as these polynomials have small circuits and as they generate the ring of invariants, they have the required separation property. In contrast, if we only insist of a small set of separating invariants (relaxing the explicitness), then Noether Normalization essentially shows that a non-explicit set of separating invariants \( \mathcal{I}_{n,r} \) of size \( \text{poly}(n,r) \) exists, basically by taking a random \( \mathcal{I}_{n,r} \). More constructively, Mulmuley observed that Gröbner basis techniques can construct a small set of separating invariants \( \mathcal{I}_{n,r} \), but this set is still not explicit as such algorithms take exponential-space, so are far from efficient. In the particular case of \( \mathbb{F}[\mathbf{X}]^{\text{GL}_n(\mathbb{F})} \), Mulmuley showed that the construction can occur in \( \text{PSPACE} \) unconditionally, or even \( \text{PH} \), assuming the Generalized Riemann Hypothesis. Thus, while there are explicit sets of separating invariants, and there are small sets of separating invariants, existing results do not achieve these two properties simultaneously.

The third problem is more geometric, as opposed to algebraic. Given a tuple of matrices \( \overline{A} \), we can consider the orbit of \( \overline{A} \) under simultaneous conjugation as a subset of \( (\mathbb{F}^{n \times n})^r \). A natural computational question is to decide whether the orbits of \( \overline{A} \) and \( \overline{B} \) intersect. However, from the perspective of algebraic geometry it is more natural to ask of the orbit closures intersect. That is, we now consider \( \overline{A} \) and \( \overline{B} \) as lying in \( (\mathbb{F}^{n \times n})^r \), where \( \mathbb{F} \) is the algebraic closure of \( \mathbb{F} \). Then, we consider the orbit closures of \( \overline{A} \) and \( \overline{B} \) in this larger space, where this refers to taking the orbits in \( (\mathbb{F}^{n \times n})^r \) and closing them with respect to the Zariski topology. This yields the following question.

**Question 9.1.13.** Let \( \mathbb{F} \) be a field of characteristic zero. Is there an efficient deterministic algorithm, in the unit-cost model over \( \mathbb{F} \), so that given \( \overline{A}, \overline{B} \in (\mathbb{F}^{n \times n})^r \), can decide whether the orbit closures of \( \overline{A} \) and \( \overline{B} \) under simultaneous conjugation have an empty intersection? ♦

Mulmuley observed that by the dictionary of geometric invariant theory [MFK94], this question also reduces to that of finding separating invariants.

**Theorem 9.1.14** (Separating invariants characterize orbit closures, [MFK94, Mul12a]). Let \( \mathcal{I}_{n,r} \subseteq \mathbb{F}[\mathbf{X}]^{\text{GL}_n(\mathbb{F})} \) be a set of separating invariants. Then the orbit closures of \( \overline{A}, \overline{B} \in (\mathbb{F}^{n \times n})^r \) have an empty intersection if and only if they are separated by some \( f \in \mathcal{I}_{n,r} \).

Thus, any explicit set \( \mathcal{I}_{n,r} \) of separating invariants, would answer the above question, as one could test if \( f \) agrees on \( \overline{A} \) and \( \overline{B} \) (as \( f \) is easy to compute, as it has a small
circuit), for all $f \in \mathcal{I}_{n,r}$. Thus, as before, Question 9.1.13 can be solved positively by a positive answer to Question 9.1.11.

The main results of this paper provide a positive answer to Question 9.1.11, and thus by the above connections a positive answer to Questions 9.1.9 and 9.1.13.

### 9.1.2 Mulmuley’s Results

Having introduced the above questions, we now summarize Mulmuley’s results that show that these questions can be solved positively if one assumes that there exist explicit hitting sets for a certain subclass of algebraic circuits, which we now define.

**Remark 9.1.15.** We note that Mulmuley defines this model using linear, and not affine functions. However, we define the model using affine functions as this allows the model to compute any polynomial (and not just homogeneous polynomials), potentially with large size. However, this is without loss of generality, as derandomizing PIT is equally hard in the linear and the affine case, via standard homogenization techniques.

**Definition 9.1.16** (Mulmuley). A polynomial $f(x_1, \ldots, x_n)$ is computable by a width $w$, depth $d$, trace of a matrix power if there exists a matrix $A(x) \in \mathbb{F}[x]^{w \times w}$ whose entries are degree $\leq 1$ polynomials in $x$ such that $f(x) = \text{tr}(A(x)^d)$. The size of a trace of a matrix power is $(n+1)w^2d$.

**Remark 9.1.17.** One could consider the size to be $(n+1)w^2 \log d$ because of the repeated squaring algorithm. However, as in this work $d \leq \text{poly}(n)$ (see Remark 3.2.2) this is essentially $nw^2d$ anyhow. Further, defining the size to be linear in $d$ is more in line with this model being a restriction of the layered ABP model, where all of the layers are the same. This is akin to the discussion of the size of a diagonal circuit in Chapter 8.

As matrix multiplication and trace both have small algebraic circuits, it follows that traces of matrix powers have small circuits. Further, as a restricted class of algebraic circuits, we can seek to deterministically solve the PIT problem for them, and the hypothesis that this is possible is potentially weaker than the corresponding hypothesis for general algebraic circuits. However, this hypothesis, in its black-box form, is strong enough for Mulmuley to derive implications for the above questions.

**Theorem** (Part of Mulmuley’s Theorems 1.1 and 3.6). Let $\mathbb{F}$ be a field of characteristic zero. Assume that there is a $t(m)$-explicit hitting set, of size $s(m)$, for traces of matrix power over $\mathbb{F}$ of size $m$. Then there is a set $\mathcal{I}_{n,r}$ of separating invariants, of size $\text{poly}(s(\text{poly}(n,r)))$, with $\text{poly}(t(\text{poly}(n,r)))$-explicit traces of matrix powers. Further, for algebraically closed $\mathbb{F}$, $\mathbb{F}[X]^{\text{GL}_n(\mathbb{F})}$ is integral over the ring generated by $\mathcal{I}_{n,r}$.

**Proof Idea:** We briefly summarize the proof idea. It is clear from the definitions that the generating set $\mathcal{I}_{n,r}^{\text{PRF}}$ for $\mathbb{F}[X]^{\text{GL}_n(\mathbb{F})}$ from Theorem 9.1.3 is a set of separating invariants, albeit larger than desired. The proof will recombine these invariants into a smaller set, by taking suitable linear combinations. Specifically, suppose $\mathcal{A}$ and $\mathcal{B}$ are separable, and thus $\mathcal{T}_{n,r}(\mathcal{A}) \neq \mathcal{T}_{n,r}(\mathcal{B})$, where $\mathcal{T}_{n,r}(\mathcal{A})$ denotes the sequence of evaluations $(f(\mathcal{A}))_{f \in \mathcal{I}_{n,r}^{\text{PRF}}}$. Standard arguments about inner-products show then
that $\langle T_{n,r}(A), \overline{\alpha} \rangle \neq \langle T_{n,r}(B), \overline{\alpha} \rangle$ for random values of $\overline{\alpha}$. As a linear combination of invariants is also an invariant, it follows that $f_{\overline{\alpha}}(M) = \langle T_{n,r}(M), \overline{\alpha} \rangle$ is an invariant, and will separate $A$ and $B$ for random $\overline{\alpha}$. Intuitively, one can non-explicitly derandomize this to yield a set of separating invariants by taking sufficiently many choices of $\overline{\alpha}$ and union bounding over all $\overline{A}$ and $\overline{B}$.

Note that finding such $\overline{\alpha}$ is equivalent to asking for a hitting set for the class of polynomials $\{f_{\overline{\alpha}}(A) - f_{\overline{\alpha}}(B) : \overline{A}, \overline{B} \in (\mathbb{F}_{\overline{n}}^{\overline{n}})^* \}$, so explicitly derandomizing PIT would give an explicit set of separating invariants, as was desired. However, as is, the above reduction is unsatisfactory in two ways. Primarily, the resulting set of separating invariants would still be exponentially large, as one cannot, by a counting argument, construct small hitting sets for the above class of polynomials unless one can exploit structure in vectors $T_{n,r}(A)$. Second, the resulting invariants $f_{\overline{\alpha}}(M)$ will not have small circuits, unless, as before, one can exploit the structure of $T_{n,r}(A)$, but now using the structure to compute the exponentially-large sum $\langle T_{n,r}(A), \overline{\alpha} \rangle$ in sub-exponential time. Both of these problems can be overcome by showing that indeed the vector $T_{n,r}(A)$ does have structure, in particular that it can be encoded into the coefficients of a small circuit. The circuit class that Mulmuley uses is the trace of matrix powers model.

**Proof Idea via Communication Complexity:** Another perspective on this proof idea comes from communication complexity. Asking whether $\overline{A}$ and $\overline{B}$ can be separated is equivalent to asking whether the two exponentially-long vectors $T_{n,r}(A), T_{n,r}(B)$ differ. From a communication standpoint, standard arguments (see [KN97]) show that deterministic equality testing of $N$-bit vectors requires $\geq N$ bits of communication, but that $\approx \lg N$ bits of communication suffice for a randomized equality testing procedure. One (randomness-efficient) method for this is to pack these bits into the coefficients of a degree $\leq N$ univariate polynomial over a finite field of size $\approx 2^N$, and then to use that different polynomials must different in $\approx 1/2$ of the points, and this can be seen as using a brute-force hitting set for low-degree univariate polynomials.

This uses about $\approx \lg N$ bits of randomness so that derandomizing this yields a $\text{poly}(N)$-bit deterministic protocol, as one would expect given the above lower bound. However, the key observation is that the vectors $T_{n,r}(A), T_{n,r}(B)$ are not arbitrary, so that one could expect to beat the above bounds by exploiting their structure. At the very least, these vectors are $\text{polylog}(N)$-explicit, as given an index the indicated value can be computed in $\text{polylog}(N)$ time (as $\overline{\text{PRF}}_{n,r}$ has $\text{poly}(n,r)$-explicit circuits, as argued above in Lemma 9.1.5).

As the above protocol for arbitrary vectors packs the vectors into a low-degree univariate polynomial, one could hope that upon performing this embedding the resulting polynomial would be of low complexity, as now one can replace the brute-force hitting set for low-degree polynomials with one that is smaller (assuming such hitting sets can be found explicitly). Mulmuley observed that this is indeed true (when embedding into multivariate polynomials).
Assuming various plausible conjectures and using the requisite results in derandomization literature, Mulmuley showed that the requisite small explicit hitting sets exist, removing the need to outright conjecture the existence of such hitting sets. This thus established the conjectural existence of small explicit sets of separating invariants by the above theorem. We list one such conditional result here, noting that all such conditional results Mulmuley derived gave sets of separating invariants of quasi-polynomial size, or worse.

**Theorem (Part of Mulmuley’s Theorems 1.2 and 5.1).** Let $\mathbb{F}$ be a field of characteristic zero. Suppose there is a multilinear polynomial (family) $f(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$ with coefficients containing at most $\text{poly}(n)$ bits, such that $f(\overline{x})$ can be computed in $\exp(n)$-time, but $f$ cannot be computed by an algebraic circuit of size $O(2^{\epsilon n})$ and depth $O(n')$ for some $\epsilon > 0$. Then $\mathbb{F}[\overline{X}]^{\text{GL}_n(\mathbb{F})}$ has a $\text{poly}(n, r)^{\text{polylog}(n, r)}$-size set of separating invariants, with $\text{poly}(n, r)^{\text{polylog}(n, r)}$-explicit traces of matrix powers.

While the above result is conditional, unconditional results can also be derived, if randomness is allowed. That is, by exploiting the connection between separating invariants and closed orbit intersections mentioned above, and using that PIT can be solved using randomness, Mulmuley obtains the following randomized algorithm.

**Theorem (Mulmuley’s Theorem 3.8).** Let $\mathbb{F}$ be a field of characteristic zero. There is an algorithm, running in randomized $\text{polylog}(n, r)$-time using $\text{poly}(n, r)$-processors (RNC), in the unit cost model over $\mathbb{F}$, such that given $\overline{A}, \overline{B} \in (\mathbb{F}^{n \times n})^r$, one can decide whether the orbit closures of $\overline{A}$ and $\overline{B}$ under simultaneous conjugation have an empty intersection.

Using the just mentioned conjectures, Mulmuley can also partially derandomize the above algorithm, but not to within polynomial time.

### 9.1.3 Our Results

We study further the connection raised by Mulmuley regarding the construction of separating invariants and the black-box PIT problem. In particular, we more carefully study the classes of algebraic circuits arising in the reduction from Noether Normalization to PIT, as well as the reduction itself.

**The computational power of traces of matrix powers:** In Section 4.3 we studied the trace of matrix power model over arbitrary rings. When specialized to computation over $\mathbb{F}[\overline{x}]$ when the matrix entries are affine forms, we recover Definition 9.1.16 from above. Mulmuley showed that derandomizing PIT for this class has implications for derandomizing Noether Normalization. In particular, as this model is a restricted class of algebraic circuits, we can ask: how restricted is it? If this model was particularly simple, it would suggest that derandomizing PIT for this class would be a viable approach to derandomizing Noether Normalization. In contrast, if this model of computation is sufficiently general, then given the difficulty of derandomizing PIT for such general models, using Mulmuley’s reduction to unconditionally derandomize...
Noether Normalization could be a formidable challenge. By the results of Section 4.3 (instantiated over $\mathbb{F}[\tau]$ with affine matrix entries) we observe that it is the latter case, proving the following observation.

**Corollary (Lemma 4.3.4).** The computational models of algebraic branching programs and traces of matrix powers are equivalent, up to polynomial blow up in size.

**Derandomizing Noether Normalization via an improved reduction to PIT:** Section 9.2 contains the main results of this chapter. Given the lack of progress on derandomizing PIT in such general models such as traces of matrix powers and ABPs, it might seem challenging to derandomize Noether Normalization for simultaneous conjugation by constructing explicit hitting sets. However, we show this is not true, by showing that derandomization of black-box PIT for (known-order) roABPs (defined in Section 4.4) suffices for derandomizing Noether Normalization for simultaneous conjugation. The roABP model is strictly weaker (sometimes exponentially so) than the ABP model, as discussed in Section 4.5.

By then invoking the hitting sets for roABPs from Section 6.5, we establish the following theorems, giving quasi-affirmative answers to Questions 9.1.9, 9.1.11 and 9.1.13. Furthermore, our results are proved unconditionally and are at least as strong as the conditional results Mulmuley obtains by assuming strong conjectures such as the Generalized Riemann Hypothesis or strong lower bound results.

Specifically, we prove the following theorem which gives an explicit set of separating invariants (see Question 9.1.11).

**Theorem (Corollary 9.2.12).** Let $\mathbb{F}$ be a field of characteristic zero. Then there is a $\text{poly}(n, r)^{O(\log(n))}$-sized set $I_{n,r}^{FS}$ of separating invariants, with $\text{poly}(n, r)$-explicit ABPs. That is, $I_{n,r}^{FS} \subseteq \mathbb{F}[^{X}_{GL_n(\mathbb{F})}]$, and for any $\overline{A}, \overline{B} \in (\mathbb{F}^{n \times n})^r$, $f(\overline{A}) \neq f(\overline{B})$ for some $f \in \mathbb{F}[^{X}_{GL_n(\mathbb{F})}]$ iff $f'(\overline{A}) \neq f'(\overline{B})$ for some $f' \in I_{n,r}^{FS}$.

As a consequence of Corollary 9.2.12 and Theorem 9.1.12 we obtain the following corollary that gives a positive answer to Question 9.1.9. In particular, it provides a derandomization of Noether Normalization Lemma for the ring of invariants of simultaneous conjugation.

**Theorem (Corollary 9.2.13).** Let $\mathbb{F}$ be an algebraically closed field of characteristic zero. Let $I_{n,r}^{FS}$ be the set guaranteed by Corollary 9.2.12. Then, $\mathbb{F}[^X_{GL_n(\mathbb{F})}]$ is integral over the subring generated by $I_{n,r}^{FS}$.

For deciding intersection of orbit closures, Question 9.1.13, the natural application of Corollary 9.2.12 with Theorem 9.1.14 would yield a quasi-polynomial-time algorithm for deciding orbit closure intersection. However, by replacing the black-box PIT results for roABPs of Section 6.5 with the white-box PIT results by Raz and Shpilka [RS05] (or the subsequent work of Arvind, Joglekar and Srinivasan [AJS09]) as presented in Section 6.3, we can obtain the following better algorithm for deciding orbit closure intersection, proving a strong positive answer to Question 9.1.13.
Theorem (Corollary 9.2.14). Let $\mathbb{F}$ be a field of characteristic zero. There is an algorithm, running in deterministic $\text{polylog}(n,r)$-time using $\text{poly}(n,r)$-processors (NC), in the unit cost model over $\mathbb{F}$, such that given $\overline{A}, \overline{B} \in (\mathbb{F}^{n \times n})^r$, one can decide whether the orbit closures of $\overline{A}$ and $\overline{B}$ under simultaneous conjugation have an empty intersection.

Mulmuley also gets results for Noether Normalization of arbitrary quivers (Mulmuley’s Theorem 4.1) in a generalization of the results on simultaneous conjugation. The main difference is a generalization of the list of generators given in Theorem 9.1.3 to arbitrary quivers, as given by Le Bruyn and Procesi [LBP90]. Our improved reduction to PIT, involving roABPs instead of ABPs, also generalizes to this case, so analogous results to the above three theorems are readily attained. However, to avoid discussing the definition of quivers, we do not list the details here.

Deciding (non-closed) orbit membership via PIT: The results mentioned thus far have taken an algebro-geometric approach to studying the orbits of tuples of matrices under simultaneous conjugation, as they take this geometric action and study the algebraic structure of its invariants. This perspective, by the very continuous nature of polynomials, can only access the orbit closures under this group action. For example, working over $\mathbb{C}$, define

$$A_\epsilon := \begin{bmatrix} 1 & \epsilon \\ 0 & 1 \end{bmatrix}, \quad P_\delta := \begin{bmatrix} \delta & 0 \\ 0 & 1 \end{bmatrix},$$

and note that for any $\epsilon$ and for any $\delta \neq 0$, $P_\delta A_\epsilon P_\delta^{-1} = A_\delta$. It follows that for any polynomial $f$ invariant under simultaneous conjugation of $2 \times 2$ matrices, that $f(A_\epsilon)$ is independent of $\epsilon$, as $f$ is continuous and we can take $\epsilon \to 0$. However, for any $\epsilon \neq 0$, $A_\epsilon$ is not conjugate to $A_0 = I_2$, the identity matrix, or equivalently, $A_\epsilon$ and $A_0$ are not in the same orbit. Thus, from the perspective of invariants $A_\epsilon$ and $A_0$ are the same, despite being in different orbits.

One can ask the analogue of Question 9.1.13, but for orbits as opposed to orbit closures. Note that by the invertibility of the group action, two orbits must either be disjoint, or equal. Thus, we equivalently ask for an algorithm to the orbit membership problem for simultaneous conjugation. That is, given $\overline{A}, \overline{B} \in (\mathbb{F}^{n \times n})^r$ is there an invertible $P \in \mathbb{F}^{n \times n}$ such that $\overline{B} = P \overline{A} P^{-1}$.

Several interesting cases of this problem were solved: Chistov, Ivanyos and Karpinski [CIK97] gave a deterministic polynomial time algorithm over finite fields and over algebraic number fields; Sergeichuk [Ser00] gave a deterministic algorithm over any field, that runs in polynomial time when supplied with an oracle for finding roots of polynomials. Chistov-Ivanyos-Karpinski also mentioned that a randomized polynomial-time algorithm for the problem follows from the work of Schwartz and

---

1In his paper Sergeichuk gives credit for the algorithm to Belitskii [Bel83].

2This is not how the result is stated in [Ser00], but this is what one needs to make the algorithm efficient. In fact, to make the algorithm of Sergeichuk run in polynomial space one needs to make another assumption that would allow writing down the eigenvalues of all matrices involved in polynomial space. For example, one such assumption would be that they all belong to the some polynomial degree extension field (Grochow [Gro13]).
Zippel [Sch80, Zip79]. In conversations with Yekhanin [Yek13], we also discovered this randomized algorithm, showing that this problem is reducible to PIT for ABPs. Because of its close relation to the rest of this work, we include for completeness the proof of the following theorem.

**Theorem** *(Lemma 9.3.1).* Let \( F \) be a field of size \( \geq \text{poly}(n) \). Then the orbit membership problem, for simultaneous conjugation, is reducible to polynomial identity testing for ABPs. In particular, this problem has a randomized, parallel, polynomial-time algorithm.

**PIT for Diagonal Depth-3 Circuits:** Mulmuley’s Theorem 1.4 showed that Noether Normalization for representations of \( \text{SL}_m(F) \) can be reduced, when \( m \) is constant, to black-box PIT of *diagonal depth-3 circuits*. We discuss hitting sets for this model more in Chapter 8.

### 9.2 The Improved Reduction to roABPs

In this section we construct a small set of explicit separating invariants for simultaneous conjugation. We do so by constructing a single roABP that encodes the entire generating set \( I_{n,r}^{\text{PRF}} \) for \( F[\mathbf{GL}_n(F)] \), as given by Theorem 9.1.3. We then use hitting sets for roABPs to efficiently extract the separating invariants from this roABP.

We begin with the construction of the roABP.

**Remark 9.2.1.** There are some slightly better versions of this construction, as well as a way to more efficiently use the hitting sets of Corollary 6.5.17. However, these modifications make the presentation slightly less modular, and do not improve the results by more than a polynomial factor, so we do not pursue these details. ♦

**Construction 9.2.2.** Let \( n, r, \ell \geq 1 \). Let \( X \) denote a vector of \( r \) matrices indexed by \([r]\), each \( n \times n \), whose entries are distinct variables. Define \( X(x) := \sum_{i \in [r]} X_i x^i \), and, for the \( \ell \) variables \( \overline{x} \), define

\[
f_\ell(X, \overline{x}) := \text{tr}(X(x_1) \cdots X(x_\ell)) .
\]

The polynomials \( f_\ell(X, \overline{x}) \) lie in the ring \( F[X, \overline{x}] \), but we also want to consider them in the ring \( F[X]/[\overline{x}] \), that is when the matrices \( X \) are considered as constants and \( \overline{x} \) are variables. In this case, the following lemma shows that these polynomials \( f_\ell(X, \overline{x}) \) can be computed by small roABPs.

**Lemma 9.2.3.** Assume the setup of Construction 9.2.2. Let \( \overline{A}, \overline{B} \in (F^{n \times n})^{[r]} \). Then \( f_\ell(\overline{A}, \overline{x}) - f_\ell(\overline{B}, \overline{x}) \) can be computed by a \( \text{poly}(n, r) \)-explicit roABP with width \( 2n^2 \), depth \( \ell \), and individual degree \( < r \), where the variable order is \( x_1 < \cdots < x_\ell \). Further, this roABP can also be computed in \( \text{poly}(n, r) \) operations in \( O(\log nr) \) parallel time.
Proof: Observe that $f_\ell(A, \pi) = \text{tr}(A(x_1) \cdots A(x_\ell)) = \sum_{i \in [n]} (\prod_{j=1}^\ell A(x_j))_{(i,i)}$. By applying a permutation of indices, and appealing to the equivalence of roABPs and disjoint variable products (Corollary 4.4.2), we see that $(\prod_{j=1}^\ell A(x_j))_{(i,i)}$ is computable by roABP in the desired variable order with depth $\ell$, width $n$, and individual degree $< r$, for each $i$. Thus, appealing to closure of roABPs under addition (Corollary 4.4.4) completes the claim, noting that these claims give the desired explicitness.

Alternatively, we can view these $f_\ell(X, \pi)$ as lying in the ring $\mathbb{F}[\pi][X]$, so that the $\pi$ are constant, and the matrices $X$ are variable. In this case, the following lemma shows that $f_\ell(X, \pi)$ can be computed by a small layered ABP.

**Lemma 9.2.4.** Assume the setup of Construction 9.2.2. Let $\alpha \in \mathbb{F}_\ell$. Then $f_\ell(X, \alpha)$ can be computed poly(n, r, $\ell$)-explicit layered ABP with width $n^2$ and depth $\ell$.

**Proof:** Observe that $f_\ell(X, \alpha) = \text{tr}(X(\alpha_1) \cdots X(\alpha_\ell))$, and that each $X(\alpha_i)$ is an $n \times n$ matrix with entries that are affine forms in the $\alpha$ that can be explicitly computed given $\pi$. Thus, in analogue to the above Lemma 9.2.3, we can compute this trace in a layered ABP of width $n^2$ and depth $\ell$ by appealing to the equivalence of layered ABPs and affine matrix products (Corollary 4.2.8) and closure of layered ABPs under addition (Corollary 4.2.9). It is straightforward to observe that the construction of this layered ABP runs in the desired time bounds, as the above lemmas are constructive.

Our next two lemmas highlight the connection between the polynomials in Construction 9.2.2 and the generators of the ring of invariants provided by Theorem 9.1.3. Namely, they show that the generators in the set $\mathcal{I}_{n,r}^{\text{PRF}}$ of Theorem 9.1.3 are faithfully encoded as coefficients of the polynomial $f_\ell(X, \pi)$, when viewing this polynomial as lying in the ring $\mathbb{F}[X][\pi]$. Note here that we use the Coeff notation as defined in Chapter 1.

Given these two lemmas the proof of the reduction is similar to the one done in Mulmuley’s paper. Nevertheless, for completeness we give the entire reduction.

**Lemma 9.2.5.** Assume the setup of Construction 9.2.2. Then for $\ell \in \mathbb{N}^I$, taking coefficients in $\mathbb{F}[X][\pi]$,

$$\text{Coeff}_{\pi^I}(f_\ell(X, \pi)) = \begin{cases} \text{tr}(X_{i_1} \cdots X_{i_\ell}) & \text{if } \ell \in [r]^I \\ 0 & \text{else} \end{cases}.$$  

**Proof:** Consider $f_\ell(X, \pi)$ as a polynomial in $\mathbb{F}[X][\pi]$. Taking coefficients, we see that

$$\text{Coeff}_{\pi^I}(f_\ell(X, \pi)) = \text{Coeff}_{\pi^I}(\text{tr}(X(x_1) \cdots X(x_\ell)))$$

$$= \text{Coeff}_{\pi^I}(\text{tr} \left( \left( \sum_{j_i \in [r]} X_{j_i} x_i^{j_i} \right) \cdots \left( \sum_{j_\ell \in [r]} X_{j_\ell} x_\ell^{j_\ell} \right) \right))$$

$$= \text{Coeff}_{\pi^I}(\text{tr} \left( \sum_{j_1, \ldots, j_\ell \in [r]} X_{j_1} \cdots X_{j_\ell} x_1^{j_1} \cdots x_\ell^{j_\ell} \right)).$$
by linearity of the trace,

\[
\text{Coeff}_{x^r} \left( \sum_{j \in [r]} \text{tr}(X_j, \ldots, X_{i_\ell}) x^r \right) = \begin{cases} 
\text{tr}(X_{i_1}, \ldots, X_{i_\ell}) & \text{if } \overline{r} \in [r]^\ell \\
0 & \text{else} \end{cases}
\]

As the above lemma shows that \( f_\ell(X, \overline{x}) \) encodes all of the generators \( T_{n,r}^{\text{PRF}} \) of Theorem 9.1.3, it follows that \( \overline{A} \) and \( \overline{B} \) agree on the generators \( T_{n,r}^{\text{PRF}} \) iff they agree on \( f_\ell(X, \overline{x}) \).

**Lemma 9.2.6.** Assume the setup of Construction 9.2.2. Let \( \overline{A}, \overline{B} \in (F^{n \times n})^{|r|} \) and \( \ell \geq 1 \). Then \( \text{tr}(A_{i_1}, \ldots, A_{i_\ell}) = \text{tr}(B_{i_1}, \ldots, B_{i_\ell}) \) for all \( \overline{r} \in [r]^\ell \) iff \( f_\ell(\overline{A}, \overline{x}) = f_\ell(\overline{B}, \overline{x}) \), where this second equality is as polynomials in the ring \( F[\overline{x}] \).

**Proof:** The two polynomials \( f_\ell(\overline{A}, \overline{x}) \), \( f_\ell(\overline{B}, \overline{x}) \) are equal iff all of their coefficients are equal. By Lemma 9.2.5, this is exactly the statement that \( \text{tr}(A_{i_1}, \ldots, A_{i_\ell}) = \text{tr}(B_{i_1}, \ldots, B_{i_\ell}) \) for all \( \overline{r} \in [r]^\ell \). \( \square \)

Hence, the polynomials \( f_\ell(X, \overline{x}) \) capture the generators \( T_{n,r}^{\text{PRF}} \) of \( F[X]^{\text{GL}_n(F)} \) thus in a sense capturing the entire ring \( F[X]^{\text{GL}_n(F)} \) also, by Theorem 9.1.3.

**Corollary 9.2.7.** Assume the setup of Construction 9.2.2. Let \( F \) be a field of characteristic zero. Let \( \overline{A}, \overline{B} \in (F^{n \times n})^{|r|} \). Then \( f(\overline{A}) = f(\overline{B}) \) for all \( f \in F[X]^{\text{GL}_n(F)} \) iff \( f_\ell(\overline{A}, \overline{x}) = f_\ell(\overline{B}, \overline{x}) \) for all \( \ell \in [n^2] \), where the second equality is as polynomials in the ring \( F[\overline{x}] \).

**Proof:** By Lemma 9.2.6, \( f_\ell(\overline{A}, \overline{x}) = f_\ell(\overline{B}, \overline{x}) \) for \( \ell \in [n^2] \) iff \( g(\overline{A}) = g(\overline{B}) \) for all \( g \) of the form \( g(X) = \text{tr}(X_{i_1}, \ldots, X_{i_\ell}) \) for \( \overline{r} \in [r]^\ell \) and \( \ell \in [n^2] \). This set of \( g \) is exactly the set \( T_{n,r}^{\text{PRF}} \) of Theorem 9.1.3, and by that result \( T_{n,r}^{\text{PRF}} \) generates \( F[X]^{\text{GL}_n(F)} \), which implies that the polynomials in \( T_{n,r}^{\text{PRF}} \) agree on \( \overline{A} \) and \( \overline{B} \) iff all the polynomials in \( F[X]^{\text{GL}_n(F)} \) agree on \( \overline{A} \) and \( \overline{B} \). Thus \( f_\ell(\overline{A}, \overline{x}) = f_\ell(\overline{B}, \overline{x}) \) for \( \ell \in [n^2] \) iff \( f(\overline{A}) = f(\overline{B}) \) for all \( f \in F[X]^{\text{GL}_n(F)} \). \( \square \)

Thus having reduced the question of whether \( F[X]^{\text{GL}_n(F)} \) separates \( \overline{A} \) and \( \overline{B} \), to the question of whether some \( f_\ell(X, \overline{x}) \) separates \( \overline{A} \) and \( \overline{B} \), we now seek to remove the need for the indeterminates \( \overline{x} \). Specifically, we will replace them by the evaluation points of a hitting set, as shown in the next construction.

**Construction 9.2.8.** Assume the setup of Construction 9.2.2. Let \( H \subseteq F^{n^2} \) be a \( t(n, r) \)-explicit hitting set for roABPs of width \( \leq 2n^2 \), depth \( n^2 \), individual degree \( < r \). Define \( T_{n,r}^H := \{ f_\ell(X, \overline{\alpha}) \mid \overline{\alpha} \in H, \ell \in [n^2] \} \), where if \( \ell < n^2 \) we use the first \( \ell \) variables of \( \overline{\alpha} \) for the values of \( \overline{x} \) in the substitution. \( \square \)
We now prove the main theorems, showing how to construct small sets of explicit separating invariants. We first do this for an arbitrary hitting set, then plug in the hitting set stated in Corollary 6.5.17.

**Theorem 9.2.9.** Assume the setup of Construction 9.2.8. Let $F$ be a field of characteristic zero. Then $I^H_{n,r}$ is a set of size $n^2|H|$ of homogeneous separating invariants with $\text{poly}(t(n, r), n, r)$-explicit layered ABPs. That is, $I^H_{n,r} \subseteq F[X]^{\text{GL}_n(F)}$, and for any \(A, B \in (F^{n \times n})^t\), \(f(A) \neq f(B)\) for some \(f \in F[X]^{\text{GL}_n(F)}\) iff \(f'(A) \neq f'(B)\) for some \(f' \in I^H_{n,r}\), and each such \(f'\) is computed by an explicit layered ABP.

**Proof:** \(I^H_{n,r}\) is explicit: We can index \(I^H_{n,r}\) by \(\ell \in [n^2]\) and an index in \(H\). Given an index in \(H\) we can, by the explicitness of \(H\), compute the associated \(\bar{\sigma} \in H\) in \(t(n, r)\) steps. By Lemma 9.2.4 we can compute a layered ABP for \(f_\ell(X, \bar{\sigma})\) in \(\text{poly}(n, r)\) steps, as \(\ell \leq n^2\), as desired.

\[f \in I^H_{n,r} \text{ are homogeneous: This is clear by construction.}\]

\[|I^H_{n,r}| = n^2|H|: \text{For each } \ell \in [n^2], \text{we use one invariant per point in } H.\]

\[I^H_{n,r} \subseteq F[X]^{\text{GL}_n(F)}: \text{For any } \ell, \text{ Lemma 9.2.5 shows that the coefficients of } f_\ell(X, \bar{\sigma}) \text{ (when regarded as polynomials in } F[X][\bar{\sigma}]\text{) are traces of products of the matrices in } X, \text{ so these coefficients are invariant under simultaneous conjugation of } X \text{ (Lemma 9.1.4).}\]

It follows then for any \(\bar{\sigma}, f_\ell(X, \bar{\sigma})\) is a linear combination of invariants, and thus is an invariant. As \(I^H_{n,r}\) consists of exactly such polynomials, it is contained in the ring of all invariants.

\[F[X]^{\text{GL}_n(F)} \text{ separates } \iff I^H_{n,r} \text{ separates: By Corollary 9.2.7, we see that for any } A \text{ and } B, \text{ there is an } f \in F[X]^{\text{GL}_n(F)} \text{ with } f(A) \neq f(B) \text{ iff there is some } \ell \in [n^2] \text{ such that } f_\ell(A, \bar{\sigma}) - f_\ell(B, \bar{\sigma}) \neq 0. \text{ By Lemma 9.2.3, } f_\ell(A, \bar{\sigma}) - f_\ell(B, \bar{\sigma}) \text{ is computable by a roABP with width } \leq 2n^2, \text{ depth } \ell, \text{ individual degree } < r, \text{ which by the addition of } n^2 - \ell \text{ dummy variables (say, to the end of } \bar{\sigma}), \text{ can be considered as a depth } n^2 \text{ roABP. That is, the matrix-product form for roABPs (Corollary 4.4.2) shows that we can pad identity matrices (regarded as matrices in } F[\bar{\sigma}]^{2n^2 \times 2n^2} \text{) to the matrix product to enlarge the number of variables while keeping the computation the same.}\]

Thus, as \(H\) is a hitting set for roABPs of the relevant size, for any \(\ell \in [n^2], f_\ell(A, \bar{\sigma}) - f_\ell(B, \bar{\sigma}) \neq 0 \text{ iff there is some } \bar{\sigma} \in H \text{ such that } f_\ell(A, \bar{\sigma}) - f_\ell(B, \bar{\sigma}) \neq 0, \text{ and thus iff there is an } \bar{\sigma} \in H \text{ such that the invariant } f'(X) := f_\ell(X, \bar{\sigma}) \in I^H_{n,r} \text{ separates } A \text{ and } B. \]
Corollary 9.2.11. Assume the setup of Construction 9.2.8. Let $\mathbb{F}$ be a field of characteristic zero. There is an algorithm, running in deterministic $\text{poly}(n, r, t(n, r), |H|)$-time, in the unit cost arithmetic model, such that given $\overline{A}, \overline{B} \in (\mathbb{F}^{n \times n})^r$, one can decide whether the orbit closures of $\overline{A}$ and $\overline{B}$ under simultaneous conjugation have an empty intersection. Further, this algorithm is “black-box”, as it simply compares $f(\overline{A})$ and $f(\overline{B})$ for various polynomials $f$.

Proof: As observed in Mulmuley’s Theorem 3.8, the orbit closures of $\overline{A}$ and $\overline{B}$ intersect iff all invariants in $\mathbb{F}[X]^{\text{GL}_n(\mathbb{F})}$ agree on $\overline{A}$ and $\overline{B}$. Our Theorem 9.2.9 shows this question can be answered by testing if $\overline{A}$ and $\overline{B}$ agree with respect to all $f \in \mathcal{I}_{n,r}$, and this can be tested in $\text{poly}(n, r, |H|, t(n, r))$-time, as ABPs can be evaluated quickly (for example, they have small circuits (Fact 4.2.10) and circuits can be evaluated efficiently (Fact 3.1.7)).

Plugging in the hitting set results for roABPs of Section 6.5 (Corollary 6.5.17), we obtain from Theorem 9.2.9 and Corollary 9.2.10 the following results, which respectively give positive answers to Question 9.1.11 and Question 9.1.9.

Corollary 9.2.12. Let $\mathbb{F}$ be a field of characteristic zero. There is a $\text{poly}(n, r)$-sized set $\mathcal{I}_{n,r}^{\text{FS}}$ of separating invariants, with $\text{poly}(n, r)$-explicit ABPs. That is, $\mathcal{I}_{n,r}^{\text{FS}} \subseteq \mathbb{F}[X]^{\text{GL}_n(\mathbb{F})}$, and for any $\overline{A}, \overline{B} \in (\mathbb{F}^{n \times n})^r$, $f(\overline{A}) \neq f(\overline{B})$ for some $f \in \mathbb{F}[X]^{\text{GL}_n(\mathbb{F})}$ iff $f'(\overline{A}) \neq f'(\overline{B})$ for some $f' \in \mathcal{I}_{n,r}^{\text{FS}}$.

Corollary 9.2.13. Let $\mathbb{F}$ be an algebraically closed field of characteristic zero. Let $\mathcal{I}_{n,r}^{\text{FS}}$ be the set guaranteed by Corollary 9.2.12. Then, $\mathbb{F}[X]^{\text{GL}_n(\mathbb{F})}$ is integral over the subring generated by $\mathcal{I}_{n,r}^{\text{FS}}$.

However, using the hitting set of Corollary 6.5.17 does not allow us to deduce the efficient algorithm for orbit closure intersection claimed in Corollary 9.2.14 as the hitting set is too large. Note that even a polynomial-size hitting set would not actually yield the desired result either, as we need a hitting set that is computable in NC, not just in P. This is also enough as ABPs have low-depth circuits, so can be evaluated efficiently in parallel. Indeed, Mulmuley used that a random $\mathcal{H}$ is such a hitting set to get his RNC algorithm for testing orbit closure intersection (cited above as his Theorem 3.8). For our NC algorithm we observe that hitting sets are somewhat stronger than needed, so that instead of using black-box PIT algorithms we can use white-box PIT algorithms. In particular, we can invoke the NC white-box algorithm presented Section 6.3 (Theorem 6.3.7), due to Raz and Shpilka [RS05].

Corollary 9.2.14. Let $\mathbb{F}$ be a field of characteristic zero. There is an algorithm, running in deterministic $O((\log^3 nr))$-time using $\text{poly}(n, r)$-processors (NC3), in the unit cost arithmetic model, such that given $\overline{A}, \overline{B} \in (\mathbb{F}^{n \times n})^r$, one can decide whether the orbit closures of $\overline{A}$ and $\overline{B}$ under simultaneous conjugation have an empty intersection.

Proof: We use that disjoint orbit closures are characterized by separating invariants (Theorem 9.1.14), as used in Mulmuley’s Theorem 3.8: the orbit closures of $\overline{A}$ and $\overline{B}$ intersect iff all invariants in $\mathbb{F}[X]^{\text{GL}_n(\mathbb{F})}$ agree on $\overline{A}$ and $\overline{B}$. Our results, Corollary 9.2.7
and Lemma 9.2.3, show there is a non-empty intersection iff an \( n^2 \)-sized set of \( \text{poly}(n, r) \)-size roABPs all compute the zero polynomial.

It follows from the white-box PIT algorithm of Raz and Shpilka [RS05], given as Theorem 6.3.7 that we can decide in NC whether each of these roABPs \( f_i(\mathbf{A}, \mathbf{x}) - f_i(\mathbf{B}, \mathbf{x}) \) are zero. Further, we can do each of the \( n^2 \) checks in parallel, and return the “and” of each of these tests, to decide if there is a non-empty intersection. Thus, there is an NC algorithm for testing if orbit closures intersect. 

\[ \square \]

### 9.3 Orbit Intersection Reduces to PIT

In this section, we study the (non-closed) orbit intersection problem, as compared with the orbit closure intersection problem studied in Section 9.2. Unlike with orbit closures, the orbits with non-empty intersections must be equal, because the group action is invertible. Thus, the orbit intersection problem is equivalent to the orbit membership problem. As mentioned before, Chistov, Ivanyos, and Karpinski [CIK97] observed a randomized algorithm for the orbit membership problem, based on the Schwartz-Zippel lemma [Sch80, Zip79]. In conversations with Yekhanin [Yek13], we also discovered this result, which we include for completeness, given its closeness to the other questions studied in this work.

**Lemma 9.3.1.** Let \( \mathbb{F} \) be a field of size \( > n \). There is a reduction, running in deterministic \( \text{polylog}(n, r) \)-time using \( \text{poly}(n, r) \)-processors in the unit cost model over \( \mathbb{F} \), from the orbit membership problem of \((\mathbb{F}^{n \times n})^r\) under simultaneous conjugation, to polynomial identity testing of ABPs. In particular, the orbit membership problem can be solved in randomized \( \text{polylog}(n, r) \)-time with \( \text{poly}(n, r) \)-processors (RNC), in the unit cost arithmetic model.

**Proof:** Let \( \mathbf{A}, \mathbf{B} \in (\mathbb{F}^{n \times n})^r \). There exists an invertible \( P \) such that \( \mathbf{B} = P \mathbf{A} P^{-1} \) iff there is an invertible \( P \) such that \( \mathbf{B} P = P \mathbf{A} \). This second equation is a homogeneous linear equation in \( P \), and thus the set of solutions is a vector space. Gaussian elimination can efficiently (Theorem 6.3.3) find a basis \( \{P_i\}_{i=1}^\ell \) for this vector space, with \( \ell \leq n^2 \). It follows then that such an invertible \( P \) exists iff \( \{P_i\}_{i=1}^\ell \) contain an invertible matrix in their \( \mathbb{F} \)-span. As the non-vanishing of the determinant characterizes invertible matrices, it follows that such a \( P \) exists iff \( f(x_1, \ldots, x_\ell) := \det(\sum_{i=1}^\ell P_i x_i) \) has a non-zero point in \( \mathbb{F}^\ell \).

Clearly \( f(\mathbf{x}) \) having a non-zero point in \( \mathbb{F}^\ell \) implies that \( f(\mathbf{x}) \) is non-zero as a polynomial. Conversely, as the total degree of \( f(\mathbf{x}) \) is \( \leq n \), and the field \( \mathbb{F} \) has size \( > n \), polynomial interpolation (Corollary B.2.9) implies that if \( f(\mathbf{x}) \) is a non-zero polynomial then it has a non-zero points in \( \mathbb{F}^\ell \). Thus, we see that there is a \( P \) such that \( \mathbf{B} = P \mathbf{A} P^{-1} \) iff \( f(\mathbf{x}) \) is a non-zero polynomial, where by the results of Berkowitz [Ber84], \( f(\mathbf{x}) \) is computable by a \( \text{poly}(n, r) \)-size ABP. Thus, we have reduced orbit-membership to PIT of ABPs, and done so in parallel. Using that ABPs can be evaluated efficiently in parallel (Fact 4.2.10 and Fact 3.1.7), and that we can solve PIT via random evaluations (Lemma 3.2.6, and using that operations in small extension
fields can be done efficiently in parallel), we get the corresponding randomized parallel algorithm for orbit membership.
Appendix A

Computational Background

We assume a basic background in the theory of computation (see for example Sipser [Sip06]). While the theory of computation is often discussed over bits, our results will often hold in fields that are sufficiently large, such as the complex numbers. As complex numbers can encode an infinite number of bits, the Turing machine model cannot manipulate arbitrary complex numbers. Thus, we will work in a “mixed” model of computation, where the usual Turing machine is augmented with the ability to manipulate field elements in a limited way. This model is essentially that of Blum-Shub-Smale [BSS89] for arbitrary fields, which we make precise here.

Definition A.0.1. The unit-cost algebraic computation model over a field \( F \) is the model of computation that augments a (possibly randomized) Turing Machine with the ability to manipulate elements of \( F \) using only the following operations:

- Storing a field element, in unit space.
- Adding two elements of \( F \), in unit time.
- Multiplying two elements of \( F \), in unit time.
- Producing the constants 0, 1 and \(-1\), in unit time.
- Testing if an element of \( F \) is zero, in unit time.
- Inverting non-zero elements of \( F \), in unit time.

- There is a function \( \varphi : \mathbb{N} \to F \) enumerating elements in \( F \), so that \( \text{im} \varphi \) is either infinite if \( F \) is infinite, or \( \text{im} \varphi = F \) if \( F \) is finite. Computing \( \varphi(i) \) costs \( i \) steps in time.

Note that this model allows access to elements of \( F \) only through those operations a field is promised to have. In particular, when \( F = \mathbb{R} \), this model is not given direct access to the (infinitely-long) bit representation.

Remark A.0.2. When \( F \) has characteristic zero, it is convenient to consider \( \varphi(i) := i \). With this definition of \( \varphi \), any algorithm in this model computing over \( \mathbb{Q} \) will only manipulate rational numbers.
Remark A.0.3. The enumeration of $\mathbb{F}$ given by $\varphi$ takes $i$ steps to compute the $i$-th element. In many interesting situations (such as $\mathbb{Q}$ or any finite field), the $i$-th element could be computed in $\text{polylog}(i)$ steps. However, this increased efficiency will not be relevant for the algorithms here.

Remark A.0.4. While this model has access to random bits (as normal Turing machines do), it does not have any notion of a “random element of $\mathbb{F}$”. However, this notion can be simulated as follows. First, use the enumeration $\varphi$ to construct some explicit $S \subseteq \mathbb{F}$ in $\text{poly}(|S|)$ steps. Then, a random element of $S$ can be sampled using $\text{polylog}(|S|)$ random bits.

This model of computation is not given any initial power as compared to a normal Turing machine (in contrast, giving the model constants such as an $n$-th root of 2 would give such power). Rather, this model should only be consider unrealistic in its ability to compute in $\mathbb{F}$ at unit cost. When $\mathbb{F}$ is a small finite field, this model is not unreasonable as all algebraic operations will stay within this field, and thus each field operation will take some fixed polynomial time. When $\mathbb{F}$ is a large finite field or $\mathbb{Q}$, then there is the possibility that the sizes of the field elements grows exponentially fast. For example $2^{2^n} \in \mathbb{Q}$ can be computed in $n$ steps in this model, but this would take exponential time for a Turing machine to even write down. However, this model most cleanly captures the algorithms presented, so this is the models in which the algorithms will be presented. Further, the algorithms presented can likely be straight-forwardly shown to not have the above bad-behavior of rapidly growing bit-complexity, as these algorithms are based on linear algebra and compute low-degree polynomials.

We give a lemma here showing that in the above model one can find elements of polynomially-large multiplicative order efficiently.

Lemma A.0.5. Let $\mathbb{F}$ be a field with $|\mathbb{F}| > n \geq 1$. Then one can in $\text{poly}(n)$ steps in the unit cost model over $\mathbb{F}$ produce an $\omega \in \mathbb{F}$ with multiplicative order $\geq n$.

Proof: First, we note that one can test in $\text{poly}(n)$ steps whether a given $\alpha \in \mathbb{F}$ has order $\geq n$ by enumerating $1, \alpha, \alpha^2, \ldots, \alpha^{n-1}$ and ensuring that these elements are distinct. Thus, it suffices so show that such an $\omega$ exists in some explicit enumeration of elements in $\mathbb{F}$. There are two cases, based on the size of $\mathbb{F}$.

$\mathbb{F} > n^2$: Note that there are $\leq k$ non-zero elements of order $\leq k$, as those elements satisfy the polynomial $x^k - 1$, which can have $\leq k$ roots in $\mathbb{F}$ (Lemma B.2.10). Thus, there are at most $\sum_{k=1}^{n-1} k \leq (n-1)^2$ non-zero elements with order $< n$. So in any subset $S \subseteq \mathbb{F}$ of size $> n^2 \geq (n-1)^2 + 1$ there must be a non-zero element of order $\geq n$, as there are only $(n-1)^2$ non-zero elements of order $< n$ and only 1 zero element. Thus, by enumerating elements of $\mathbb{F}$ we can find the desired $\omega$ in $\text{poly}(n)$ steps.

$\mathbb{F} \leq n^2$: As the non-zero elements in $\mathbb{F}$ form a cyclic group (see Shoup [Sho09]) there is a non-zero element of $\mathbb{F}$ with order $|\mathbb{F}| - 1 \geq n$. Thus, again we can enumerate $\mathbb{F}$ to find such an $\omega$. 


Appendix B

Algebra Background

Abstract: In this appendix, we give various algebraic facts that we will need. The first (small) section is about how to extract coefficients from polynomials, and how this operation is linear. The second (larger) section is about polynomial interpolation, and includes all of the standard results such as the Schwartz-Zippel [Sch80,Zip79] Lemma. The final (smallest) section gives a proof of the Cauchy-Binet formula.

B.1 Coefficients

In this section we give properties of coefficients of polynomials that will be used frequently. We first show that taking coefficients is linear.

Lemma B.1.1. Consider the map Coeff: \( \mathbb{F}[x] \to \mathbb{F} \) defined by \( f \to \text{Coeff}_x(f) \). Then Coeff is \( \mathbb{F} \)-linear.

Proof: Let \( f, g \in \mathbb{F}[x] \), where \( f = \sum x^i c_i \) and \( g = \sum d_i x^i \) then

\[
\text{Coeff}_x(\alpha f + \beta g) = \text{Coeff}_x(\sum_{i} (\alpha c_i + \beta d_i) x^i) \\
= \alpha \cdot c_0 + \beta \cdot d_0 \\
= \alpha \cdot \text{Coeff}_x(f) + \beta \cdot \text{Coeff}_x(g). \quad \Box
\]

As we will often “take coefficients in \( \mathbb{F}[\overline{y}][x] \)” we record here that linearity can be over the field of rational functions \( \mathbb{F}(\overline{y}) \).

Corollary B.1.2. Consider the map Coeff: \( \mathbb{F}[\overline{y}][x] \to \mathbb{F}[\overline{y}] \), when taking coefficients in \( \mathbb{F}[\overline{y}][x] \). Then Coeff is \( \mathbb{F}(\overline{y}) \)-linear, and thus \( \mathbb{F} \)-linear. \( \Box \)

We now observe that when taking coefficients of polynomials that factorize into variable disjoint factors, we can take coefficients separately.

Lemma B.1.3. Consider the map Coeff taking coefficients in \( \mathbb{F}[x, \overline{y}] \). Then for \( f(x) \in \mathbb{F}[x] \) and \( g(\overline{y}) \in \mathbb{F}[\overline{y}] \), \( \text{Coeff}_{x,\overline{y}}(f \cdot g) = \text{Coeff}_x(f) \cdot \text{Coeff}_{\overline{y}}(g) \).
Proof: Express \( f = \sum \alpha_i x^i \) and \( g = \sum \beta_j y^j \). Then,

\[
\text{Coeff}_{x^a y^b}(f \cdot g) = \text{Coeff}_{x^a y^b} \left( \sum \alpha_i \beta_j x^i y^j \right) = \alpha_i \beta_j = \text{Coeff}_{x^a}(f) \cdot \text{Coeff}_{y^b}(g) .
\]

\( \square \)

### B.2 Polynomial Interpolation

The process of polynomial evaluation is the process of moving from the coefficients of a polynomial to the evaluation of this polynomial at a specified point. This section is all about polynomial interpolation, which is the inverse of polynomial evaluation, in that we wish to convert a set of evaluations of a single polynomial back to the coefficients of that polynomial. We will give the standard results in this area, such as the fact that a large enough “cube” of evaluation points facilitates interpolation, and we will also prove the Schwartz-Zippel [Sch80, Zip79] Lemma.

We begin with a key construction, that of the Lagrange Interpolation Polynomial, which act as indicator functions on a specified set. As such, we will use a modification of our indicator function notation (which, recalling Chapter 1, is \( \mathbbm{1}_P(x) \) for the \( \{0, 1\}\)-function returning 1 iff \( x \) has property \( P \)).

**Construction B.2.1.** Let \( \mathbb{F} \) be a field, let \( S_1, \ldots, S_n \subseteq \mathbb{F} \), and denote \( S := S_1 \times \cdots \times S_n \). For any \( \overline{\alpha} \in S \), define the Lagrange Interpolation Polynomial at \( \overline{\alpha} \) with respect to \( S \), denoted \( \mathbbm{1}_{\overline{\alpha}, S}(\overline{x}) \), to be

\[
\mathbbm{1}_{\overline{\alpha}, S}(\overline{x}) := \prod_{i=1}^n \prod_{\overline{\beta} \in S_i \setminus \{\alpha_i\}} \frac{x_i - \beta_i}{\alpha_i - \beta_i} .
\]

Note that these polynomials are well-defined as the denominator \( \alpha_i - \beta_i \) is never zero.

The interpretation of the above notation “\( \mathbbm{1}_{\overline{\alpha}, S}(\overline{x}) \)” is that this is an indicator function for “being \( \overline{\alpha} \)”, but that this is only valid when \( x \in S \). We now establish this indicator-like behavior and other properties of these polynomials. First, it is easily seen that the construction gives low-degree polynomials.

**Lemma B.2.2.** Assume the setup of **Construction B.2.1.** Then \( \deg_{x_i} \mathbbm{1}_{\overline{\alpha}, S} = |S_i| - 1. \) \( \square \)

Second, we observe that these polynomials are somewhat explicit.

**Lemma B.2.3.** Assume the setup of **Construction B.2.1.** Given \( S_1, \ldots, S_n \) explicitly, and given any \( \overline{\alpha} \in S \), an algebraic circuit for \( \mathbbm{1}_{\overline{\alpha}, S}(\overline{x}) \) can be computed in \( \text{poly}(|S|, n) \)-steps in the unit-cost model over \( \mathbb{F} \). \( \square \)

Now we observe that indeed the Lagrange interpolation polynomials act as indicator functions on the set \( S \).
Lemma B.2.4. Assume the setup of Construction B.2.1. Then for any \( \alpha, \gamma \in S \)

\[
1_{\pi,S}(\gamma) = \begin{cases} 
1 & \gamma = \alpha \\
0 & \gamma \in S \setminus \alpha
\end{cases}.
\]

Proof: \( n = 1 \): Then,

\[
1_{\alpha,S}(x) = \prod_{\beta \in S \setminus \{\alpha\}} \frac{x - \beta}{\alpha - \beta},
\]

from which it follows that

\[
1_{\alpha,S}(\alpha) = \prod_{\beta \in S \setminus \{\alpha\}} \frac{\alpha - \beta}{\alpha - \beta} = 1,
\]

and

\[
1_{\alpha,S}(\gamma) = \prod_{\beta \in S \setminus \{\alpha, \gamma\}} \frac{\gamma - \beta}{\alpha - \beta} = \frac{\gamma - \gamma}{\alpha - \gamma} \cdot \prod_{\beta \in S \setminus \{\alpha, \gamma\}} \frac{\gamma - \beta}{\alpha - \beta} = 0.
\]

\( n > 1 \): Now observe that \( 1_{\pi,S}(\pi) = \prod_i 1_{\alpha_i,S_i}(x_i) \). When \( \gamma = \alpha \), the \( n = 1 \) case shows that \( 1_{\pi,S}(\alpha) = \prod_i 1_{\alpha_i,S_i}(x_i) = 1 \). When \( \gamma \neq \alpha \), there is some \( i_0 \) so \( \gamma_{i_0} \neq \alpha_{i_0} \), so we can again use the \( n = 1 \) case to see that \( 1_{\pi,S}(\gamma) = 1_{\alpha_{i_0},S_{i_0}}(\gamma_{i_0}) \cdot \prod_{i \neq i_0} 1_{\alpha_i,S_i}(\gamma_i) = 0 \cdot \prod_{i \neq i_0} 1_{\alpha_i,S_i}(\gamma_i) = 0 \), as desired. \( \square \)

We now use the Lagrange interpolation polynomials to understand the map from a polynomial to its evaluations.

Proposition B.2.5. Assume the setup of Construction B.2.1. Consider the vector space \( F \subseteq \mathbb{F}[x_1, \ldots, x_n] \) such that for all \( f \in F \), \( \deg_{x_i} f < |S_i| \) for all \( i \in [n] \). Define \( \varphi : F \to \mathbb{F}^S \) by \( f \mapsto (f(\pi))_{\pi \in S} \). Then \( \varphi \) is an isomorphism of vector spaces, and its inverse is given by \( \psi : \mathbb{F}^S \to F \), taking \( \tau \in \mathbb{F}^S \) to

\[
\sum_{\pi \in S} \tau_\pi \cdot 1_{\pi,S}(\pi).
\]

Proof: As evaluation maps are linear maps, and \( \varphi \) is a coordinate-wise evaluation map, it follows that \( \varphi \) is linear. We now show \( \varphi \) is surjective. Observe that the above map \( \psi \) is indeed a linear map, and by Lemma B.2.2 \( \psi \) indeed maps into \( F \). Further, it follows from Lemma B.2.4 that for \( \tau \in \mathbb{F}^S \) and \( \pi \in S \), \( \psi(\pi)(\pi) = \tau_\pi \), from which it follows that \( \varphi \circ \psi \) is the identity map on \( \mathbb{F}^S \), implying that \( \varphi \) is surjective.

Now observe that both \( F \) and \( \mathbb{F}^S \) are vector spaces of dimension \( |S| \). Thus, \( \varphi \) surjective implies that \( \varphi \) is injective, and thus an isomorphism. Thus, \( \varphi \) has some inverse \( \varphi^{-1} \). As \( \varphi \circ \psi \) is the identity on \( \mathbb{F}^S \), it implies that \( \psi = \varphi^{-1} \circ \varphi \circ \psi = \varphi^{-1} \) as maps on \( \mathbb{F}^S \), so \( \psi = \varphi^{-1} \) as desired. \( \square \)

The above result shows that there are two fundamental ways of defining polynomials: in terms of their coefficients, and in terms of their evaluations (on a large enough cube). While these different perspectives are isomorphic (as shown above), going from one representation to another often has fruitful outcomes.
In particular, as a corollary we now observe how to extract coefficients from the evaluations of a polynomial.

**Lemma B.2.6.** Assume the setup of Construction B.2.1. Fix \( \pi \in \mathbb{N}^n \) with \( a_i < |S_i| \). Then there exist a \( \text{poly}(n, \sum_i |S_i|) \)-explicit family of coefficients \((c_{\pi, \pi})_{\pi \in S} \), defined by

\[
c_{\pi, \pi} := \prod_{i=1}^{n} \text{Coeff}_{x_i^{a_i}} \left( \prod_{\beta_i \in S_i \setminus \{\alpha_i\}} \frac{x_i - \beta_i}{\alpha_i - \beta_i} \right),
\]

so that for any \( f \in \mathbb{F}[x_1, \ldots, x_n] \) with \( \deg_{x_i} f < |S_i| \),

\[
\text{Coeff}_{\pi}(f) = \sum_{\pi \in S} c_{\pi, \pi} f(\pi).
\]

**Proof:** correctness: By Proposition B.2.5, it follows that

\[
f(\pi) = \sum_{\pi \in S} f(\pi) \cdot 1_{\pi, S}(\pi).
\]

Taking coefficients of both sides, and invoking linearity, it follows that,

\[
\text{Coeff}_{\pi}(f) = \sum_{\pi \in S} f(\pi) \cdot \text{Coeff}_{\pi} (1_{\pi, S}(\pi))
\]

\[
= \sum_{\pi \in S} f(\pi) \cdot \text{Coeff}_{\pi} \left( \prod_{i=1}^{n} \prod_{\beta_i \in S_i \setminus \{\alpha_i\}} \frac{x_i - \beta_i}{\alpha_i - \beta_i} \right)
\]

Invoking that coefficients of variable disjoint formulas can be taken disjointly, which is the content of Lemma B.1.3,

\[
= \sum_{\pi \in S} f(\pi) \cdot \prod_{i=1}^{n} \text{Coeff}_{x_i^{a_i}} \left( \prod_{\beta_i \in S_i \setminus \{\alpha_i\}} \frac{x_i - \beta_i}{\alpha_i - \beta_i} \right).
\]

**Explicitness:** This is clear from the definition, appealing to the unit-cost model over \( \mathbb{F} \).

Note that the above is only non-trivial when \( \pi \) has \( a_i < |S_i| \) always, as otherwise \( \text{Coeff}_{\pi}(f) = 0 \).

We now turn to using the isomorphism between coefficients and evaluations for polynomial identity testing. In particular, this isomorphism was for a cube of a certain size, we now study how the size of this cube interacts with the number of non-zero evaluations of a non-zero polynomial. We start with the cube from the above isomorphism.

**Corollary B.2.7.** Let \( \mathbb{F} \) be a field, let \( S_1, \ldots, S_n \subseteq \mathbb{F} \), and denote \( S := S_1 \times \cdots \times S_n \). Let \( f \in \mathbb{F}[x_1, \ldots, x_n] \) be a polynomial with \( \deg_{x_i} f < |S_i| \). Then \( f \equiv 0 \) iff \( f|_S \neq 0 \).
Proof: In the language of Proposition B.2.5, $0 \neq f \in \mathcal{F}$. As $\varphi : \mathcal{F} \to \mathbb{F}^S$ is a linear isomorphism, it follows that $f \neq 0$ iff $\varphi(f) \neq 0$, where $\varphi(f) \neq 0$ is equivalent to $f|_S \neq 0$.

Remark B.2.8. Note that the above result is tight, in that omitting a single evaluation point from $S$ will yield a polynomial in $\mathcal{F}$ without a non-zero evaluation, as seen by taking the appropriate Lagrange polynomial $l_{\pi,S}$ for some $\pi \in S$.

One can also see the above result as a corollary of Corollary 6.2.10, where the $\mathbb{F}$-algebra $R$ is simply $R = \mathbb{F}$. In this case, the span of coefficients or evaluations is simply either $\{0\}$ (when the polynomial is zero) or $\mathbb{F}$ (when the polynomial is zero). Corollary 6.2.10 shows that the $\mathbb{F}$-span is preserved between coefficients and evaluations over the cube, so in particular non-zeroness is preserved.

We now state a simple corollary of the above which is looser in the parameters but more convenient for applications.

Corollary B.2.9. Let $\mathbb{F}$ be a field, let $S \subseteq \mathbb{F}$. Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a non-zero polynomial with $\deg f < |S|$. Then $f|_{S^n} = 0$.

The above result shows that the polynomial $f$ is non-zero on some point $S^n$, but one can obtain many non-zero points if $\deg f \ll |S|$. We first establish the univariate case, which is just showing that a low-degree univariate polynomial has few roots.

Lemma B.2.10. Let $\mathbb{F}$ be a field, and let $f \in \mathbb{F}[x]$ be a non-zero univariate polynomial. Then $f$ has $\leq \deg f$ roots in $\mathbb{F}$.

Proof: Suppose $f$ is a (potentially zero) univariate polynomial with $> \deg f$ roots in $\mathbb{F}$. Let $S \subseteq \mathbb{F}$ be these roots, so $|S| > \deg f$. Thus, $f|_S = 0$ from which Corollary B.2.9 implies $f$ cannot be non-zero, thus $f$ is zero.

Note that the above fact is more traditionally proven using the division algorithm (or unique factorization) in $\mathbb{F}[x]$, and this proof can be found in Shoup [Sho09]. We now use this as the inductive base case to obtain a similar result for multivariate polynomials, yielding the following well-known lemma.

Lemma B.2.11 (Schwartz-Zippel Lemma, [Sch80, Zip79]). Let $\mathbb{F}$ be a field, let $S \subseteq \mathbb{F}$. Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a non-zero polynomial. Then

$$\Pr_{\pi \leftarrow S^n}[f(\pi) = 0] \leq \frac{\deg f}{|S|^n}.$$ 

Proof: By induction.

$n = 1$: This follows from Lemma B.2.10.

$n > 1$: Rename the variables so that $\pi = (\bar{y}, z)$. Expand $f(\bar{y}, z)$ as a polynomial in $\mathbb{F}[\bar{y}][z]$ so that

$$f(\bar{y}, z) = \sum_{0 \leq i \leq d} f_i(\bar{y})z^i,$$

where as $f(\bar{y}, z) \neq 0$ we can assume $d$ is chosen $f_d(\bar{y}) \neq 0$. Note that this is still true if $\deg_z f = 0$, as then $d = 0$. Note that as this decomposition is simply a way of
rewriting $f$, it must be that $\deg f_d + \deg z^d \leq \deg f$ so that
$$\deg f_d \leq \deg f - d.$$ Breaking $\overline{\alpha}$ into $(\overline{\beta}, \gamma)$, we analyze the probability by conditioning on the event that “$f_d(\overline{\beta}) = 0$”.

$$\Pr_{(\overline{\beta}, \gamma) \leftarrow S^n}[f(\overline{\beta}, \gamma) = 0] = \Pr_{\overline{\beta}, \gamma}[f(\overline{\beta}, \gamma) = 0 | f_d(\overline{\beta}) = 0] \cdot \Pr_{\overline{\beta}}[f_d(\overline{\beta}) = 0]$$
$$+ \Pr_{\overline{\beta}, \gamma}[f(\overline{\beta}, \gamma) = 0 | f_d(\overline{\beta}) \neq 0] \cdot \Pr_{\overline{\beta}}[f_d(\overline{\beta}) \neq 0]$$

As probabilities are at most 1,
$$\leq 1 \cdot \Pr_{\overline{\beta}}[f_d(\overline{\beta}) = 0] + \Pr_{\overline{\beta}, \gamma}[f(\overline{\beta}, \gamma) = 0 | f_d(\overline{\beta}) = 0] \cdot 1$$

Applying the inductive claim to $f_d$,
$$\leq 1 \cdot \frac{\deg f - d}{|S|} + \Pr_{\overline{\beta}, \gamma}[f(\overline{\beta}, \gamma) = 0 | f_d(\overline{\beta}) \neq 0] \cdot 1$$

Applying the $n = 1$ case to $f(\overline{\beta}, z)$, as $f_d(\overline{\beta}) \neq 0$ implies $f(\overline{\beta}, z)$ is a non-zero univariate polynomial in $z$ of degree $d$,
$$\leq 1 \cdot \frac{\deg f - d}{|S|} + \frac{d}{|S|} \cdot 1$$
$$= \frac{\deg f}{|S|} .$$

Remark B.2.12. This lemma is tight, as consider the polynomial $f(\overline{x}) = \prod_{\beta \in T} (x_1 - \beta)$ for $T \subseteq S$. Then $\Pr_{\overline{x} \leftarrow S^n}[f(\overline{x}) = 0] = |T|/|S|$, exactly meeting the above bound. Even though this polynomial seems “inherently” univariate, one can also apply a linear change of variables to $\overline{x}$ to get less-obviously univariate examples that meet this upper bound.

We note that this lemma has several different forms. The one above is due to Schwartz [Sch80], and gives a probability based on the total degree of a polynomial. Zippel [Zip79] gave a version that bounds the probability in terms of the individual degree of the polynomial. Both versions are tight with respect to their parameters, which are different regimes. DeMillo and Lipton [DL78] also gave a version of this lemma, and the bound is in terms of the total degree, but is not tight.

### B.3 The Cauchy-Binet Formula

For completeness we give the proof of the Cauchy-Binet formula here.
Lemma B.3.1 (Cauchy-Binet Formula). Let \( m \geq n \geq 1 \). Let \( M \in \mathbb{F}^{n \times m} \), \( N \in \mathbb{F}^{m \times n} \). Then,

\[
\det(MN) = \sum_{S \in \binom{[m]}{n}} \det(M_{\bullet,S}) \det(N_{S,\bullet}).
\]

Proof: Let \( \Lambda(\pi) \) be an \( m \times m \) diagonal matrix with the variables \( x_1, \ldots, x_m \) on the diagonal. Define the polynomial \( f(x_1, \ldots, x_m) := \det(M \cdot \Lambda(\pi) \cdot N) \), so that \( f(1, \ldots, 1) = \det(MN) \). Every entry of \( M \Lambda(\pi) N \) is a degree \( \leq 1 \) polynomial in \( \pi \), which, as the determinant is degree \( n \) implies that \( f \) has degree \( \leq n \). Let \( S \in \binom{[m]}{n} \) and consider all (not necessarily multilinear) monomials only containing variables in \( \{x_i\}_{i \in S} \). Each monomial of degree \( \leq n \) (and thus each monomial with non-zero coefficient in \( f \)) must be associated with at least one such \( S \).

Define \( \rho_S \) to be the vector of variables when the substitution \( x_i \mapsto 0 \) is performed for \( i \notin S \). By block matrix multiplication, it follows then that \( f(\rho_S) = \det(M_{\bullet,S} \cdot \Lambda(\pi)|_{S,S} \cdot N_{S,\bullet}) = \det(M_{\bullet,S}) \det(N_{S,\bullet}) \prod_{i \in S} x_i \), where the last equality follows as \( M_{\bullet,S}, N_{S,\bullet} \) and \( \Lambda(\pi)|_{S,S} \) are all \( n \times n \) matrices. By the above reasoning, this implies that the only monomials with non-zero coefficients in \( f \) are monomials of the form \( \prod_{i \in S} x_i \) and such monomials have coefficient \( \det(M_{\bullet,S}) \det(N_{S,\bullet}) \). Thus \( f = \sum_{S \in \binom{[m]}{n}} \det(M_{\bullet,S}) \det(N_{S,\bullet}) \prod_{i \in S} x_i \), and so \( \det(MN) = f(1, \ldots, 1) = \sum_{S \in \binom{[m]}{n}} \det(M_{\bullet,S}) \det(N_{S,\bullet}) \), yielding the claim. \( \square \)
Appendix C

Hasse Derivatives

Abstract: This appendix gives the details Hasse derivatives, which are a notion of formal derivative that work better in arbitrary characteristic. In particular, we give the chain rule for Hasse derivatives, which is somewhat intricate. We then use Hasse derivatives to define the complexity measure of shifted Hasse derivatives, as introduced by Kayal [Kay10] and whose power was demonstrated by Gupta-Kamath-Kayal-Saptharishi [GKKS13a].

C.1 Hasse Derivatives

In this section we define and prove basic properties of Hasse derivatives, which are an algebraic (as opposed to analytic) notion of derivative. Hasse derivatives are a variant of the “usual” formal partial derivative that works better in small characteristic. In particular, $k$-th order partial derivatives simply iterate the first-order derivative $k$ times. However, this process annihilates too much information, as the $p$-th derivative of $x^p$ in characteristic zero is 0. For Hasse derivatives, the $p$-th derivative of $x^p$ is 1. To achieve this, Hasse derivatives define multiple derivatives from the beginning, and not from iterating the first-order derivative. This difference is key for understanding multiplicities, and turns out to generally be more useful in small characteristic fields.

We begin with the definition of Hasse derivatives. Note that we define these derivatives over any commutative ring, of which fields will be the main application.

Definition C.1.1. Let $R$ be a commutative ring and $R[x_1, \ldots, x_n]$ a polynomial ring. For a vector $\bar{v} \in R^n$ and $a \geq 0$, define $\partial_{\bar{v}}^a(f) : R[\bar{x}] \to R[\bar{x}]$, the $a$-th Hasse derivative of $f$ in direction $\bar{v}$, by $\partial_{\bar{v}}^a(f) := \text{Coeff}_{y^a}(f(\bar{x} + \bar{v}y)) \in R[\bar{x}]$, where coefficients of $f(\bar{x} + \bar{v}y)$ are taken in $R[\bar{x}][y]$.

If $\bar{v} = e_i$, then we use $\partial_{x_i}^a$ to denote $\partial_{e_i}^a$, and will call this the $a$-th Hasse derivative with respect to the variable $x_i$. For $\bar{v} \in \mathbb{N}^n$, we will define $\partial_{\bar{x}\bar{v}}^a := \partial_{x_{i_1}}^{e_i} \cdots \partial_{x_{i_n}}^{e_i}$. ♦

While most often we will consider Hasse derivatives with respect to monomials, directional Hasse derivatives will prove useful as a proof technique, such as in the chain rule (Lemma C.1.12).

The intuition for the above definition comes from analysis, where the Taylor
expansion of a function \( f \), around the point \( \bar{x} \), yields coefficients based on the derivatives of that function. For polynomials, this relation is especially simple as the Taylor series are finite. As such, we will use these coefficients as the definition of the derivative, as in this setting there is no topology for differential analysis to use.

This Taylor-series perspective of taking coefficients with respect to the infinitesimal \( y \) can also be defined simply by the action of this map on each monomial. While equivalent, a definition based on the action of each monomial is more cumbersome to work with.

One could hope to also define the notion over \( R\{\bar{x}\} \), the ring of non-commuting variables. However, one can then not do Taylor series as that requires commutativity. Even if one defined the derivative monomial-wise, the non-zero non-commutative polynomial \( f = xy - yx \) would then have \( \partial_x(f) = \partial_y(f) = 0 \), which seems perverse.

We first show the action of a Hasse derivative on a single monomial.

**Lemma C.1.2.** For any \( i \in [n] \), \( b \geq 0 \), and \( a_i \geq 0 \),

\[
\partial_{\bar{x}_i^b}(\bar{x}^\alpha) = \partial_{\bar{x}_i^b}(x_1^{a_1} \cdots x_i^{a_i} \cdots x_n^{a_n}) = \left( \frac{a_i}{b} \right) x_1^{a_1} \cdots x_{i-1}^{a_{i-1}} x_i^{a_i-b} x_{i+1}^{a_{i+1}} \cdots x_n^{a_n} = \left( \frac{a_i}{b} \right) \bar{x}^\alpha \cdot x_i^{-b},
\]

where the third and fourth terms are computed in the ring of Laurent polynomials (formally \( R[x_1, \ldots, x_n, x_1^{-1}, \ldots, x_n^{-1}] \)), but lie in \( R[x_1, \ldots, x_n] \).

**Proof:** We do the case where \( i = 1 \), as the general case is symmetric. Thus we want to understand the coefficient of \( y^b \) in \( (x_1 + y)^{a_1} x_2^{a_2} \cdots x_n^{a_n} \). The binomial theorem tells us the coefficient of \( y^b \) in \( (x_1 + y)^{a_1} \) is \( \left( \frac{a_1}{b} \right) x_1^{a_1-b} \) (even in the case that \( b > a_1 \), interpreted correctly). Plugging this into the rest of the monomial yields the result. \( \square \)

Generalizing this via linearity to derivatives on more than one variable, we get the following immediate corollary.

**Corollary C.1.3.**

\[
\partial_{\bar{x}}(\bar{x}^\alpha) = \left( \frac{a}{b} \right) \bar{x}^{\alpha-b},
\]

where the last statement is computed in the ring of Laurent polynomials \( R[\bar{x}, \bar{x}^{-1}] \), but lies in \( R[\bar{x}] \). \( \square \)

We now use the definition to establish some basic properties of the Hasse derivative. Note that while Hasse derivatives are linear operators on \( R[\bar{x}] \), they are not in general linear in the direction \( \bar{\pi} \) of the derivative, and so several of these properties will be stated for more than two terms.

**Lemma C.1.4.** For \( f, g \in R[x_1, \ldots, x_n], \bar{\pi}, \bar{\nu} \in R^n, \alpha, \beta \in R, \) and \( a, b, a_1, \ldots, a_m \geq 0 \), then

1. \( \partial_{\bar{\pi}}(f) = f \)
2. \( \partial_{\bar{\pi}}(\alpha f + \beta g) = \alpha \partial_{\bar{\pi}}(f) + \beta \partial_{\bar{\pi}}(g) \)
3. \( f(\bar{x} + \bar{\pi}_1 y_1 + \cdots + \bar{\pi}_m y_m) = \sum_{a_1, \ldots, a_m \geq 0} [\partial_{\bar{\pi}^{a_1}_1} \cdots \partial_{\bar{\pi}^{a_m}_m}(f)] y_1^{a_1} \cdots y_m^{a_m} \)
4. $\partial_{\pi_1} \cdots \partial_{\pi_m} f = \text{Coeff}_{y_1^a \cdots y_m^a} (f(x + \pi_1 y_1 + \cdots + \pi_m y_m))$

5. $\partial_{\pi_1} \partial_{\pi_2} (f) = \partial_{\pi_1} \partial_{\pi_2} (f)$

6. $\partial \left( \sum_{j=1}^{m} \alpha_j \pi_j \right)^a (f) = \sum_{a_1 + \cdots + a_m = a} \left( \prod_{j} \alpha_j^a \right) \partial_{\pi_1} \cdots \partial_{\pi_m} (f)$

7. $\partial_{\alpha \pi + \beta \pi} (f) = \alpha \partial_{\pi} (f) + \beta \partial_{\pi} (f)$

8. $\partial_{\pi_1} \cdots \partial_{\pi_m} (f) = \left( \begin{array}{c} a_1 + \cdots + a_m \\ a_1, \ldots, a_m \end{array} \right) \partial_{\pi_1} \cdots \partial_{\pi_m} (f)$

9. $\partial_{\pi_1} \cdots \partial_{\pi_m} (f) = \left( \begin{array}{c} \pi_1 + \cdots + \pi_m \\ \pi_1, \ldots, \pi_m \end{array} \right) \partial_{\pi_1} \cdots \partial_{\pi_m} (f)$

**Proof:**

1: By regrouping terms, we can express $f(x + \pi y) = \sum a f_a(x) y^a$. Setting $y = 0$ we then get $f(x) = f_0(x)$, which is the claim.

2: This follows immediately from 1, taking Coeff_{y^a} (\cdot), and so

$\text{Coeff}_{y^a} (\alpha f(x + \pi y) + \beta g(x + \pi y)) = \alpha \text{Coeff}_{y^a} (f(x + \pi y)) + \beta \text{Coeff}_{y^a} (g(x + \pi y))$.

3: This will be proven by induction on $m$.

$m = 1$: This is the definition of the Hasse derivative.

$m > 1$: By induction, we have that

$$f(x + \pi_1 y_1 + \cdots + \pi_m y_m) = \sum_{a_2, \ldots, a_m \geq 0} [\partial_{\pi_2} \cdots \partial_{\pi_m} (f)](x) y_2^{a_2} \cdots y_m^{a_m}.$$  

Making the substitution $x \leftarrow x + \pi_1 y_1$ we obtain

$$f(x + \pi_1 y_1 + \pi_2 y_2 + \cdots + \pi_m y_m) = \sum_{a_2, \ldots, a_m \geq 0} [\partial_{\pi_2} \cdots \partial_{\pi_m} (f)](x + \pi_1 y_1) y_2^{a_2} \cdots y_m^{a_m},$$

and so by expanding $\partial_{\pi_2} \cdots \partial_{\pi_m} (f)$ into its Hasse derivatives, we obtain

$$f(x + \pi_1 y_1 + \pi_2 y_2 + \cdots + \pi_m y_m) = \sum_{a_1, a_2, \ldots, a_m \geq 0} [\partial_{\pi_1} \partial_{\pi_2} \cdots \partial_{\pi_m} (f)](x) y_1^{a_1} y_2^{a_2} \cdots y_m^{a_m}.$$  

as desired.

4: This is a restatement of 3.

5: This follows immediately from 4, taking $m = 2$, as Coeff_{y^2} (f(x + \pi z + \pi y)) = Coeff_{y^2} (f(x + \pi y + \pi z)).$

6: By 3 we have that

$$f(x + \pi_1 y_1 + \cdots + \pi_m y_m) = \sum_{a_1, \ldots, a_m \geq 0} [\partial_{\pi_1} \cdots \partial_{\pi_m} (f)](x) y_1^{a_1} \cdots y_m^{a_m},$$

and applying the substitution $y_j \leftarrow \alpha_j y$ we obtain

$$f(x + (\alpha_1 \pi_1 + \cdots + \alpha_m \pi_m) y) = \sum_{a_1, \ldots, a_m \geq 0} \alpha_1^{a_1} \cdots \alpha_m^{a_m} [\partial_{\pi_1} \cdots \partial_{\pi_m} (f)](x) y^{a_1 + \cdots + a_m},$$

187
and thus taking the coefficient of \( y^a \) yields the result.

(7): This is a special case of (6) with \( m = 2 \) and \( a = 1 \).

(8): By (4), we see that

\[
\partial_{\pi_1} \cdots \partial_{\pi_m} (f) = \text{Coeff}_{y_1^{a_1} \cdots y_m^{a_m}} (f(\pi_1 y_1 + \cdots + \pi y_m)) \\
= \text{Coeff}_{\pi}(f(\pi_1 y_1 + \cdots + y_m))
\]

Expanding \( f \) in Hasse derivatives using \( y_1 + \cdots + y_m \) as a single variable,

\[
= \text{Coeff}_{\pi} \left( \sum_a \partial_{\pi}^a (f)(\pi) \cdot (y_1 + \cdots + y_m)^a \right) \\
= \sum_a \partial_{\pi}^a (f)(\pi) \cdot \text{Coeff}_{\pi}((y_1 + \cdots + y_m)^a) \\
= \left( a_1 + \cdots + a_m \right) \partial_{\pi_1^{a_1} \cdots \pi_m^{a_m}} (f) .
\]

(9): This follows from the above properties and expanding the definition.

\[
\partial_{\pi_1} \cdots \partial_{\pi_m} (f) \\
= \left( \partial_{\pi_1}^{a_1,1} \cdots \partial_{\pi_1}^{a_1,n} \right) \cdots \left( \partial_{\pi_1}^{a_m,1} \cdots \partial_{\pi_1}^{a_m,n} \right)
\]

Regrouping using commutativity ((5)),

\[
= \left( \partial_{\pi_1}^{a_1,1} \cdots \partial_{\pi_1}^{a_m,1} \right) \cdots \left( \partial_{\pi_1}^{a_1,n} \cdots \partial_{\pi_1}^{a_m,n} \right)
\]

Combining iterated derivatives ((8)),

\[
= \left( \left( a_1 + \cdots + a_m,1 \right) \partial_{\pi_1}^{a_1,1,1} \cdots \partial_{\pi_1}^{a_1,m,1} \right) \cdots \left( \left( a_1 + \cdots + a_m,n \right) \partial_{\pi_1}^{a_1,n,1} \cdots \partial_{\pi_1}^{a_1,n,m} \right)
\]

Rewriting in vector notation,

\[
= \left( \vec{a}_1 + \cdots + \vec{a}_m \right) \partial_{\pi_1 + \cdots + \pi_m} (f) .
\]

Notice that these properties show that the derivative \( \partial_{\pi} \) does not depend on the ordering of the variables. These properties also allow us to compare the Hasse derivative with the “usual” formal multiple partial derivative.

**Lemma C.1.5.** Let \( \pi \in \mathbb{R}^n \). Then as operators on \( R[\pi] \), \( \partial_{\pi}^a = a! \partial_{\pi}^a \), where \( \partial_{\pi}^a \) is the operator \( \partial_{\pi} \) iterated a times.

**Proof:** For \( a = 0 \), this is trivial, as both operators are the identity operator (for partials, this is as we apply the partial operator 0 times, for Hasse derivatives this is
by Lemma C.1.4.1), and 0! = 1. For a > 1 this is just a special case of Lemma C.1.4.8, using that \((1,\ldots,1) = a!\).

We now show that taking derivatives decreases the degree of a polynomial, where we take the convention that \(\deg 0 = -\infty\).

**Lemma C.1.6.** Let \(f \in R[\bar{x}]\). Then \(\deg \partial_{\bar{x}}^\pi(f) \leq \deg f - \deg \bar{x} = \deg f - \|\bar{x}\|_1\).

**Proof:** Let \(f = \sum b_\bar{x} \bar{x}^\pi\). By linearity (Lemma C.1.4) and the action of derivatives on monomials (Corollary C.1.3), we see that
\[
\partial_{\bar{x}}(f) = \sum b_{\bar{x}} \bar{x}^{\bar{b} - \bar{x}},
\]
so that
\[
\deg \partial_{\bar{x}}(f) = \max_{\bar{x} \neq 0} \|\bar{b} - \bar{\pi}\|_1
\]
As \((\bar{b} / \bar{\pi}) \neq 0 \implies \bar{b} \geq \bar{\pi},
\[
= \max_{\|\bar{b}\|_1 - \|\bar{\pi}\|_1}
\]
\[
\leq \max_{\|\bar{b}\|_1 - \|\bar{\pi}\|_1}
\]
\[
= \deg f - \|\bar{x}\|_1,
\]
where the last step is an equality if the maximization is not empty, and reduces to the equality \(\neg \infty = \neg \infty - \|\bar{x}\|_1\) otherwise.

Note that, by the convention that \(\deg 0 = -\infty\), this inequality is always valid. Note that this inequality can be strict as over characteristic \(p\), \(\partial_{x^p} = 0\). This can also be a strict inequality over characteristic zero, as \(\partial_{x^2}(xy^{10} + x^2) = 1\).

We can now use the above properties to establish the product rule for Hasse derivatives.

**Lemma C.1.7 (Product Rule).** For \(f_1, \ldots, f_m \in R[x_1, \ldots, x_n]\), \(\bar{u} \in R^n\),
\[
\partial_{\bar{u}^m}(f_1 \cdots f_m) = \sum_{a_1 + \cdots + a_m = a} \partial_{\bar{u}^1}(f_i) \cdots \partial_{\bar{u}^m}(f_m).
\]
Proof:

\[(f_1 \cdots f_m)(x + uy) = \prod_{i=1}^{m} f_i(x + uy)\]

\[= \prod_{i} \left( \sum_{a_i} \partial_{\pi a_i} (f_i) y^{a_i} \right)\]

\[= \sum_{a} \sum_{a_1+\cdots+a_m=a} \prod_{i} \partial_{\pi a_i} (f_i) \cdot y^{a},\]

and result follows by taking the coefficient of \(y^a\). \(\square\)

We now record two specializations of this as corollaries.

**Corollary C.1.8 (Product Rule).** For \(f, g \in R[x_1, \ldots, x_n], \pi \in R^n\),

\[\partial_{\pi a} (fg) = \sum_{b+c=a} \partial_{\pi b} (f) \partial_{\pi c} (g).\]

This corollary becomes particularly simple (and familiar) when we take first-order derivatives.

**Corollary C.1.9 (Liebnitz Rule).** For \(f, g \in R[x_1, \ldots, x_n], \pi \in R^n\),

\[\partial_{\pi a} (fg) = \partial_{\pi a} (f) g + f \partial_{\pi a} (g).\]

We remark here on the uniqueness of the first-order Hasse derivative, considering the univariate case \(R[x]\). If we require that the map \(\partial : R[x] \to R[x]\) obeys linearity, the Liebnitz rule, \(\partial(x) = 1\) and \(\partial(1) = 0\), then the map is completely determined. For these properties enforce that \(\partial(x^n) = \partial(x)x^{n-1} + x\partial(x^{n-1})\), which by induction gives \(nx^{n-1}\), as we expect, where the base case is \(\partial(x) = 1\). The action on polynomials is then determined by linearity. The uniqueness for Hasse derivatives over multivariate polynomials follows from the univariate case, as by Lemma C.1.4, the directional derivatives are determined the derivatives \(\partial_{x_i}\), which are essentially the single-variable Hasse derivatives.

We can also use the Liebnitz rule to get the quotient rule. As we will need division we restrict this discussion to fields.

**Lemma C.1.10 (Quotient Rule).** Let \(f, g, h \in F[x_1, \ldots, x_n]\) and \(\pi \in F^n\) such that \(fh = g\), or \(f = g/h\). Then \(\frac{\partial_{\pi a}(g)h - g\partial_{\pi a}(h)}{h^2} = \partial_{\pi a}(f) \in F[\pi]\), where the last expression is computed in the fraction field \(F(\pi)\), but lies in \(F[\pi]\).

**Proof:** By the Liebnitz rule, we see that \(\partial_{\pi a}(g) = \partial_{\pi a}(fh) = \partial_{\pi a}(f) h + f \partial_{\pi a}(h)\). Solving for \(\partial_{\pi a}(f)\), we see that \(\partial_{\pi a}(f) = \frac{\partial_{\pi a}(g)h - g\partial_{\pi a}(h)}{h^2}\). Substituting in \(f = g/h\), we get the result. The fact that the Hasse derivative is a polynomial follows from that \(f\) is a polynomial (even though we performed some of these manipulations over the field of rational functions, the result is a polynomial). \(\square\)

**Remark C.1.11.** The above quotient rule begs the question as to whether we could have discussed derivatives over the field of rational functions in the variables \(\pi\). That is
indeed possible, but seemingly more cumbersome, as the above definition of a derivative requires a Taylor expansion. Such expansions are always finite for polynomials, but can be power series for rational functions. As such, it may seem more natural to define derivatives for power series. However, then the situation becomes more complicated as the notion of substitution (such as in the chain rule) is not valid in all situations as there are power series such as $1 + x + x^2 + \cdots$ where substitutions such as $x = 1$ yield infinities (when $\mathbb{F}$ has characteristic zero).

If one wanted only to stick to rational functions over a field $\mathbb{F}$, the substitution problem still exists (as the denominator can become zero), and furthermore it is more work to show the derivative is well defined. One option is to construct a $\mathbb{F}$-basis for $\mathbb{F}[x]$, and this can be done using the partial fraction decomposition given by the Chinese Remainder Theorem. Alternatively, one can show that the derivative over $\mathbb{F}[x]$ can be lifted to a derivative over $\mathbb{F}(x)$ in a well-defined manner by carrying the derivative through the fraction field construction. However, the gains of these generalizations seems unclear, so avoid them here. ♦

We now will establish the chain rule for Hasse derivatives. As Hasse derivatives necessarily take multiple derivatives all at once, the chain rule will be for multiple derivatives and thus necessarily more complicated than the “usual” chain rule for first derivatives, which we derive as a corollary. This formula, and its variants, are sometimes called Fàà di Bruno’s formula.

Note that in the below we will make heavy use of vector notation, in particular

$$(\partial_{\mathbf{u}}(\mathbf{g})(\mathbf{x}))^{\mathbf{b}} := \prod_i (\partial_{u_i}(g_i)(x))^{b_i},$$

where vector notation is used three times. The first use takes “$\partial_{u_i}(\mathbf{g})$” as applying the operator $\partial_{u_i}$ to the vector $\mathbf{g}$ point-wise, thus producing a vector of polynomials. This vector is then applied to the point $\mathbf{x}$ point-wise. Finally, we take this vector and raise it to the exponent $\mathbf{b}$, which exponentiates each coordinate of the vector and then takes the product, thus producing a single polynomial, following the convention of Chapter 1.

**Lemma C.1.12** (Chain Rule). For $f \in R[y_1, \ldots, y_m]$ and $\mathbf{g} \in (R[x_1, \ldots, x_n])^m$ and $\mathbf{u} \in R^n$,

$$\partial_{\mathbf{u}}((f \circ \mathbf{g})(\mathbf{x})) = \sum_{\sum_{j=1}^m b_j = a} \left( \prod_{j=1}^a (\partial_{u_j}(g_j)(x))^{b_j} \right) \left( \frac{\partial_{\mathbf{x}} f}{\partial_{\mathbf{x}}} \right)^{\sum_{j=1}^a b_j} (\mathbf{g}(\mathbf{x}))$$

where the $\mathbf{b}$ are in $\mathbb{N}^m$. 191
Proof:

\[ \partial_{\pi^a}(f \circ g) = \text{Coeff}_{z^a}((f \circ g)(\pi + \pi z)) \]

\[ = \text{Coeff}_{z^a}\left(f \left(\sum_j \partial_{\pi^j}(g)(\pi) z^j\right)\right) \]

As we can discard powers of \( z \) that are > \( a \),

\[ = \text{Coeff}_{z^a}\left(f \left(\overline{g}(\pi) + \partial_{\pi^a}(g)(\pi) z + \cdots + \partial_{\pi^a}(g)(\pi) z^a\right)\right) . \]

We now seek to take derivatives “in the direction of \( \partial_{\pi^j}(g)(\pi) \)” from the point \( \overline{y} := \overline{g}(\pi) \), treating each \( j \) as a different direction and each \( z^j \) as a different variable. However, this is a subtle operation, as up until now we have taken the directions of our derivatives as independent of the point of derivation. To make this subtlety clear, we now study the above equation, by “undoing” the substitutions “\( \overline{y} \leftarrow \overline{g}(\pi) \)”, and “\( z^j \leftarrow z^j \)”, and working with derivatives in the ring \( R[\pi][\overline{y}] \). We will then later “redo” these substitutions. A simpler form of this logic was used to establish Lemma C.1.4.6. We start about by applying Lemma C.1.4.3 to expand out \( f \) in the directions \( \partial_{\pi^j}(g)(\pi) \).

\[ f\left(\overline{y} + \partial_{\pi^a}(g)(\pi) z_1 + \cdots + \partial_{\pi^a}(g)(\pi) z_a\right) \]

\[ = \sum_{\substack{b_1, \ldots, b_a \geq 0}} \left[ \partial_{\partial_{\pi^a}(g)(\pi)}^{b_1} \cdots \partial_{\partial_{\pi^a}(g)(\pi)}^{b_a} (f) \right] (\overline{y}) \cdot z_1^{b_1} \cdots z_a^{b_a} \]

and by Lemma C.1.4.6, we can decompose the derivative \( \partial_{\pi^j}(g)(\pi) \) as linear combinations of the \( \partial_{y_k} \) derivatives, so that as operators on \( R[\pi][\overline{y}] \),

\[ \partial\left[\partial_{\pi^j}(g)(\pi)\right]^{b_j} = \sum_{b_{j,1} + \cdots + b_{j,m} = b_j} \left(\prod_{i=1}^{m} \left[ \partial_{\pi^j}(g_i)(\pi) \right]^{b_{j,i}}\right) \partial_{y_{i,1}} \cdots \partial_{y_{i,m}} \]
Or in vector notation,

\[ \partial_{\mathbf{\nu}_j} (\mathbf{g})(\mathbf{x}) = \sum_{\|b_j\|_1 = b_j} (\partial_{\mathbf{g}}(\mathbf{g})(\mathbf{x})) b_j \partial_{\mathbf{y}} b_j \]

As these operators are acting on \( \mathbb{R}^{[\mathbf{x}][\mathbf{y}]} \) we can treat all polynomials in \( \mathbf{x} \) as constants, in particular the terms involving the \( \partial_{\mathbf{g}}(g_i)(\mathbf{x}) \) above. This allows us to move all of the operators past the terms involving the \( g_i \), to obtain,

\[ = \sum_{\|b_j\|_1 = b_j} \left( \prod_{j=1}^{a} (\partial_{\mathbf{g}}(\mathbf{g})(\mathbf{x})) b_j \right) \left[ \partial_{\mathbf{y}} b_1 \cdots \partial_{\mathbf{y}} b_a \right] (\mathbf{g}) \cdot \mathbf{z} b_j \]

By invoking Lemma C.1.4.9 we can combine iterated derivatives,

\[ = \sum_{\|b_j\|_1 = b_j} \left( \prod_{j=1}^{a} (\partial_{\mathbf{g}}(\mathbf{g})(\mathbf{x})) b_j \right) \cdot \left( \frac{\partial}{\partial y_1} + \cdots + \frac{\partial}{\partial y_a} \right) \left[ \partial_{\mathbf{y}} \sum_{j=1}^{a} b_j f \right] (\mathbf{g}) \cdot \mathbf{z} \]

We now “redo” the substitutions \( \mathbf{g} \leftarrow \mathbf{g}(\mathbf{x}) \), and \( z_j \leftarrow z^j \), to obtain that,

\[ \partial_{\mathbf{\nu}_j} ((f \circ g)(\mathbf{x})) \]

\[ = \text{Coeff}_{z^a} \left( f \left( \mathbf{g}(\mathbf{x}) + \partial_{\mathbf{\nu}_j}(\mathbf{g})(\mathbf{x}) z + \cdots + \partial_{\mathbf{\nu}_j}(\mathbf{g})(\mathbf{x}) z^a \right) \right) \]

\[ = \text{Coeff}_{z^a} \left( \sum_{\|b_j\|_1 = b_j} \left( \prod_{j=1}^{a} (\partial_{\mathbf{g}}(\mathbf{g})(\mathbf{x})) b_j \right) \left( \frac{\partial}{\partial y_1} + \cdots + \frac{\partial}{\partial y_a} \right) \left[ \partial_{\mathbf{y}} \sum_{j=1}^{a} b_j f \right] (\mathbf{g}(\mathbf{x})) \cdot z \sum_{j=1}^{a} b_j f \right) \]

\[ = \sum_{\sum_{j=1}^{a} j \cdot \|b_j\|_1 = a} \left( \prod_{j=1}^{a} (\partial_{\mathbf{g}}(\mathbf{g})(\mathbf{x})) b_j \right) \left( \frac{\partial}{\partial y_1} + \cdots + \frac{\partial}{\partial y_a} \right) \left[ \partial_{\mathbf{y}} \sum_{j=1}^{a} b_j f \right] (\mathbf{g}(\mathbf{x})) \cdot \mathbf{z} \]

We now recover the chain rule for first derivatives by specializing the above.

**Corollary C.1.13** (Chain Rule, Revisited). For the polynomial \( f \in \mathbb{R}[y_1, \ldots, y_m] \) and vector of polynomials \( \mathbf{g} \in (\mathbb{R}[x_1, \ldots, x_n])^m \), \( \mathbf{u} \in \mathbb{R}^n \),

\[ \partial_{\mathbf{\nu}} (f \circ \mathbf{g}) = \sum_{i=1}^{m} \partial_{y_i} (f)(\mathbf{g}) \cdot \partial_{\mathbf{\nu}} (g_i) . \]

**Proof:** Take \( a = 1 \) in the above Lemma C.1.12. It follows that there is only one
$m$-dimensional vector $\overline{b}_1$, and the summation that ranges over \(\sum_{j=1}^n j \cdot \|\overline{b}_j\|_1 = a\) indicates that the summation is over all vectors $\overline{b}_1$ with $\|\overline{b}_1\|_1 = 1$, that is, it sums over the $m$-dimensional indicator vectors $\overline{e}_1, \ldots, \overline{e}_m$. Further, the multinomial coefficient \(\binom{\overline{b}_1 + \cdots + \overline{b}_a}{\overline{b}_1} = \binom{\overline{b}_1}{\overline{b}_1} = 1\). Thus

$$
\partial_\pi(f(g_1, \ldots, g_m)) = \sum_{j=1}^m \left[ \prod_{i=1}^m \partial_{\pi}(g_i)(\overline{\pi})_{j_i} \right] \cdot 1 \cdot \left[ \partial_{\pi}(f)(\overline{\pi}) \right]
$$

As \((\overline{\pi}_j)_i\) is the indicator function $\mathbb{1}_j(i)$,

$$
= \sum_{j=1}^m \partial_{\pi}(g_j)(\overline{\pi}) \cdot 1 \cdot \left[ \partial_{\pi}(f)(\overline{\pi}) \right]
$$

$$
= \sum_{i=1}^m \partial_{g_i}(f)(\overline{\pi}(\overline{\pi})) \cdot \partial_{\pi}(g_i)(\overline{\pi}) .
$$

\[\square\]

### C.2 Derivatives as a Complexity Measure

In this section we define several complexity measures for polynomials all based on Hasse derivatives. These measures will take a polynomial and associate a vector space with this polynomial, so that simple polynomials are associated with low-dimensional vector spaces. The first measure is based on low-order derivatives and was introduced by Nisan and Wigderson [NW96], and will be used more in Section 8.4. The second measure, as introduced by Kayal [Kay10] and Gupta-Kamath-Kayal-Saptharishi [GKKS13a], extends this first measure in a robust way.

**Definition C.2.1.** Let $\partial_{x \leq k} : F[x] \rightarrow 2^F[x]$ be the space of $k$-th order (Hasse) derivatives (function), so that

$$
\partial_{x \leq k}(f) := \text{span} \left\{ \partial_x(f) \right\}_{\|\overline{b}_i\|_1 \leq k} .
$$

Define $\partial_{x = k}$, and $\partial_{x < k}$ similarly, where $\partial_{x < \infty}$ considers the space of all Hasse derivatives.

The second measure is based on multiplication with low-degree monomials, and was introduced by Kayal [Kay12] to augment the first measure.

**Definition C.2.2.** Let $\overline{x} \leq \ell : F[\overline{x}] \rightarrow 2^F[\overline{x}]$ be the space of $\ell$-th order shifts (function), so that

$$
\overline{x} \leq \ell(f) := \text{span} \left\{ \overline{x} \cdot f \right\}_{\|\overline{x}\|_1 \leq \ell} .
$$

and define $\overline{x} = \ell$ similarly.

Note that the operator $\overline{x} \cdot F[\overline{x}] \rightarrow F[\overline{x}]$ defined by $f \mapsto \overline{x} \cdot f$ is a linear operator. Further, we allow the above measures to be composed in the natural way, so that $\overline{x} \leq \ell \partial_{x \leq k}(f)$ denotes $\text{span} \left\{ \overline{x} \partial_{x \leq k}(f) \right\}_{\|\overline{x}\|_1 \leq \ell, \|\overline{b}_i\|_1 \leq k}$.

In a sense, these measures are “super-operators” in that they collect operators on $F[\overline{x}]$ together to map elements of $F[\overline{x}]$ to subspaces of $F[\overline{x}]$. One could imagine devel-
Aiming to develop a theory of such super-operators, to the extent that addition and multiplication of vector spaces is defined in $\mathbb{F}[\mathbf{x}]$, but we will not do so here. Instead, we will note the relevant facts in a more concrete fashion. We begin with showing that the above measures interact nicely with addition by linearity.

**Lemma C.2.3.** Let $\{\varphi_i : \mathbb{F}[\mathbf{x}] \to \mathbb{F}[\mathbf{x}]\}_i$ be a (potentially infinite) collection of linear operators. Let $\varphi : \mathbb{F}[\mathbf{x}] \to 2^{\mathbb{F}[\mathbf{x}]}$ be defined by $\varphi(f) := \text{span}\{\varphi_i(f)\}_i$.

Then for any $f, g \in \mathbb{F}[\mathbf{x}]$, $\varphi(f + g) \subseteq \varphi(f) + \varphi(g)$, where this addition is of subspaces in $\mathbb{F}[\mathbf{x}]$. In particular, $\dim(\varphi(\sum_i f_i)) \leq \sum_i \dim(\varphi(f_i))$.

**Proof:** By linearity (Lemma C.1.4)

$$\varphi(f + g) = \text{span}\{\varphi_i(f + g)\}_i = \text{span}\{\varphi_i(f) + \varphi_i(g)\}_i \subseteq \text{span}\{\varphi_i(f) + \varphi_i'(g)\}_{i,i'} = \text{span}\{\varphi_i(f)\}_i + \text{span}\{\varphi_i'(g)\}_{i'} = \varphi(f) + \varphi(g).$$

The second part of the claim follows from induction on the first, and taking the dimension. $\square$

We now study, following Gupta-Kamath-Kayal-Saptharishi [GKKS13a], how these measures interact with function composition. In particular, this requires use of the above chain rule. We begin with a very basic simplification of the chain rule.

**Corollary C.2.4.** Let $f \in \mathbb{F}[y_1, \ldots, y_m]$. Suppose $\mathbf{g} \in (\mathbb{F}[x_1, \ldots, x_n])^m$ and $\mathbf{u} \in \mathbb{F}^n$, then

$$\partial_{\mathbf{u}}(f \circ \mathbf{g})(\mathbf{x}) \in \text{span}\left\{ \left( \prod_{j=1}^a (\partial_{\mathbf{u}}(\mathbf{g})(\mathbf{x})) \right)^{\mathbf{b}_j} \cdot \left[ \partial_{\mathbf{y}} f \right] (\mathbf{g}(\mathbf{x})) \right\} \sum_{i=1}^m \sum_{\sum_{j=1}^a \mathbf{b}_j = a} \mathbf{b}_j.$$

**Proof:** Using the above chain rule (Lemma C.1.12), we get

$$\partial_{\mathbf{u}}(f \circ \mathbf{g})(\mathbf{x}) = \sum_{\sum_{j=1}^a \mathbf{b}_j = a} \left( \prod_{j=1}^a (\partial_{\mathbf{u}}(\mathbf{g})(\mathbf{x})) \right)^{\mathbf{b}_j} \left[ \partial_{\mathbf{y}} \sum_{j=1}^a \mathbf{b}_j f \right] (\mathbf{g}(\mathbf{x}))$$

$$\in \text{span}\left\{ \left( \prod_{j=1}^a (\partial_{\mathbf{u}}(\mathbf{g})(\mathbf{x})) \right)^{\mathbf{b}_j} \left[ \partial_{\mathbf{y}} \sum_{j=1}^a \mathbf{b}_j f \right] (\mathbf{g}(\mathbf{x})) \right\} \sum_{j=1}^a \sum_{\mathbf{b}_j = a} \mathbf{b}_j,$$
Taking $\bar{b} = \sum_j \bar{b}_j$ and using that $\sum_j j\|\bar{b}_j\|_1 = a,$
\[
\subseteq \text{span} \left\{ \left( \prod_{j=1}^a (\partial_{\pi_j}(\bar{g}(\bar{x}))^{\bar{b}_j} \right) \left[ \partial_{\bar{g}} f \right] (\bar{g}(\bar{x})) \right\}_{\sum_{j=1}^a j\|\bar{b}_j\|_1 = a, \|\bar{b}\|_1 \leq a}
\]
Rewriting that $\sum_j j\|\bar{b}_j\|_1 = a$ as $\sum_i \sum_j j\bar{b}_{j,i} = a$ for all $i,$
\[
\subseteq \text{span} \left\{ \left( \prod_{j=1}^a (\partial_{\pi_j}(\bar{g}(\bar{x}))^{\bar{b}_j} \right) \left[ \partial_{\bar{g}} f \right] (\bar{g}(\bar{x})) \right\}_{\sum_{i=1}^m \sum_{j=1}^a j\bar{b}_{j,i} = a, \|\bar{b}\|_1 \leq a}
\]

In the case that the $g_i$ are low-degree, we can more concisely relate $\partial_{\pi^a}((f \circ \bar{g})(\bar{x}))$ to the above measures with respect to $f$. In particular, the measure take Hasse derivatives of $f$ (with respect to $\bar{g}$) to some order, then evaluates these derivatives on the point $\bar{g}(\bar{x})$, then shifts by some monomial in $\bar{x}$.

**Corollary C.2.5.** Let $f \in \mathbb{F}[y_1, \ldots, y_m]$. Suppose $\bar{g} \in (\mathbb{F}[x_1, \ldots, x_n])^m$ and $\bar{\pi} \in \mathbb{F}^n$, where each $g_i$ is of degree $\leq d$. Then
\[
\partial_{\bar{\pi}^a}((f \circ \bar{g})(\bar{x})) \in \mathbb{F}^{\leq a(d-1)} \left( \left[ \partial_{\bar{g}} f \right] (\bar{g}(\bar{x})) \right)
\]

**Proof:** We begin with the above corollary (Corollary C.2.4)
\[
\partial_{\bar{\pi}^a}((f \circ \bar{g})(\bar{x}))
\]
\[
\subseteq \text{span} \left\{ \left( \prod_{j=1}^a (\partial_{\pi_j}(\bar{g}(\bar{x}))^{\bar{b}_j} \right) \left[ \partial_{\bar{g}} f \right] (\bar{g}(\bar{x})) \right\}_{\sum_{i=1}^m \sum_{j=1}^a j\bar{b}_{j,i} = a, \|\bar{b}\|_1 \leq a}
\]
Now note that as $\deg g_i \leq d$ then $\deg \partial_{\pi_j} g_i \leq d - j$ (Lemma C.1.6), from which it follows that (unwrapping the vector notation),
\[
\deg \prod_i \prod_j [\partial_{\pi_j}(g_i)(\bar{x})]^{b_{j,i}} \leq \sum_i \sum_j b_{j,i}(d - j) = d \sum_i \sum_j b_{j,i} - \sum_i \sum_j j b_{j,i} \leq d \cdot a - a
\]
where we used that $\sum_{i=1}^m \sum_{j=1}^a j b_{j,i} = a$ and thus $\sum_{i=1}^m \sum_{j=1}^a b_{j,i} \leq a$, as $j \geq 1$ in the summation. By splitting $\prod_i \prod_j [\partial_{\pi_j}(g_i)(\bar{x})]^{b_{j,i}}$ into its monomials it follows that
\[
\prod_i \prod_j [\partial_{\pi_j}(g_i)(\bar{x})]^{b_{j,i}} \in \text{span}\{\bar{x}^{\bar{T}}\}_{\|\bar{T}\| \leq a(d-1)}
\]
which combined with the above equation (as multiplication is bilinear) yields the claim. \qed

**Remark C.2.6.** Note that the above statement is also true with $d = 0$ so that $f \circ \bar{g}$ is a constant, with the convention that $\deg 0 = -\infty$. That is, if $a = 0$ then (for any
\(d \geq 0\) the statement yields
\[
(f \circ \overline{g})(x) = \partial_{x^n}((f \circ \overline{g})(x)) \\
\in \text{span}\left\{ x^\alpha \cdot \left[ \partial_{y^\beta} f \right](\overline{g}(x)) \right\}_{\|\alpha\|_1 \leq d, \|\beta\|_1 \leq \alpha}
\]
\[
= \text{span}\{ 0 \cdot \left[ \partial_{y^\beta} f \right](\overline{g}(x)), 1 \cdot \left[ \partial_{y^\beta} f \right](\overline{g}(x)) \}
\]
\[
= \text{span}\{(f \circ \overline{g})(x)\},
\]
which is true. If \(d = 0\) and \(a > 0\) then
\[
0 = \partial_{x^n}((f \circ \overline{g})(x)) \\
\in \text{span}\left\{ x^\alpha \cdot \left[ \partial_{y^\beta} f \right](\overline{g}(x)) \right\}_{\|\alpha\|_1 \leq a-1, \|\beta\|_1 \leq \alpha}
\]
\[
= \text{span}\{ 0 \cdot \left[ \partial_{y^\beta} f \right](\overline{g}(x)) \}_{\|\beta\|_1 \leq a}
\]
\[
= \text{span}\{0\},
\]
which is true as the only monomials of degree \(\leq -a\) is the monomial 0.
\(\diamondsuit\)

We now extend the above corollary by induction to derivatives in multiple directions.

To keep notation clear, we restrict to directions \(\overline{e}_i\) so that the derivatives are respect to some (multivariate) monomial.

**Corollary C.2.7.** Let \(f \in \mathbb{F}[y_1, \ldots, y_m]\). Suppose \(\overline{g} \in (\mathbb{F}[x_1, \ldots, x_n])^m\) and \(x^\alpha \in \mathbb{F}[x] \) a non-zero monomial, where each \(g_i\) is of degree \(\leq d\). Then
\[
\partial_\overline{g}((f \circ \overline{g})(x)) \in \text{span}\left\{ x^\alpha \cdot \left[ \partial_{y^\beta} f \right](\overline{g}(x)) \right\}_{\|\alpha\|_1 \leq \|\overline{g}\|_1, \|\beta\|_1 \leq \|\overline{g}\|_1}
\]

**Proof:** Write \(x^\alpha = x_1^{a_1} \cdots x_n^{a_n}\). Let \(j \in [n]\) be the maximum \(j\) so \(a_j \neq 0\). The claim is by induction on \(j\).

\(j = 1\): Then \(x^\alpha = x_1^{a_1}\) so \(\partial_{x_1^{a_1}} = \partial_{x_1^{a_1}},\) so that by the above corollary (Corollary C.2.5)
\[
\partial_{x_1^{a_1}}(f \circ g) \in \text{span}\left\{ x^\alpha \cdot \left[ \partial_{y^\beta} f \right](\overline{g}) \right\}_{\|\alpha\|_1 \leq a_1, \|\beta\|_1 \leq a_1}
\]
and using that \(a_1 = \|\overline{g}\|_1\) yields the claim.

\(j > 1\): Then \(x^\alpha = x_1^{a_1} \cdots x_{j-1}^{a_{j-1}} x_j^{a_j}\), so that by commutativity of derivatives we have that \(\partial_{x^\alpha} = \partial_{x_1^{a_1}} \cdots \partial_{x_{j-1}^{a_{j-1}}} \partial_{x_j^{a_j}}\). By induction,
\[
\partial_{x_1^{a_1} \cdots x_{j-1}^{a_{j-1}}}(f \circ \overline{g}) \in \text{span}\left\{ x^\alpha \cdot \left[ \partial_{y^\beta} f \right](\overline{g}) \right\}_{\|\alpha\|_1 \leq (a_1 + \cdots + a_{j-1} - 1), \|\beta\|_1 \leq (a_1 + \cdots + a_{j-1} - 1)}
\]
As \(\partial_{x_j^{a_j}}\) is a linear operator, it follows then that
\[
\partial_{x_j^{a_j}} \left[ \partial_{x_1^{a_1} \cdots x_{j-1}^{a_{j-1}}}(f \circ \overline{g}) \right] \in \text{span}\left\{ \partial_{x_j^{a_j}} \left( x^\alpha \cdot \left[ \partial_{y^\beta} f \right](\overline{g}) \right) \right\}_{\|\alpha\|_1 \leq (a_1 + \cdots + a_{j-1} - 1), \|\beta\|_1 \leq (a_1 + \cdots + a_{j-1} - 1)}\] (C.2.8)
so now it suffices to understand these latter polynomials, which we can do first using
the product rule (Lemma C.1.7),
\[ 
\partial_{x_j}^a [\overline{\varpi} \cdot \overline{\partial\varpi f}(\overline{y})] = \sum_{r+s=a_j} \partial_{x_j}^r [\overline{\varpi}] \cdot \partial_{x_j}^s [\overline{\partial\varpi f}(\overline{y})]
\]

Applying the derivative to the monomial (Corollary C.1.3),
\[ 
\sum_{r+s=a_j} \left( \frac{c_j}{r} \overline{\varpi} x_j^{-r} \cdot \partial_{x_j}^s [\overline{\partial\varpi f}(\overline{y})] \right) 
\in \text{span} \left\{ \overline{\varpi} x_j^{-r} \cdot \partial_{x_j}^s [\overline{\partial\varpi f}(\overline{y})] \right\}_{r \leq c_j, s \leq a_j}.
\]

Continuing (C.2.9) yields
\[ 
\partial_{x_j}^a [\overline{\varpi} \cdot \overline{\partial\varpi f}(\overline{y})] \in \text{span} \left\{ \overline{\varpi} x_j^{-r} \cdot \partial_{x_j}^s [\overline{\partial\varpi f}(\overline{y})] \right\}_{r \leq c_j, s \leq a_j}
\subseteq \text{span} \left\{ \overline{\varpi} x_j^{-r} \cdot \overline{\varpi}^r [\partial_{\overline{\varpi} f}(\overline{y})] \right\}_{r \leq c_j, s \leq a_j}
\]

Plugging this into (C.2.8),
\[ 
\partial_{x_j}^a [\partial_{x_1^{a_1} \cdots x_{j-1}^{a_{j-1}}} f(\overline{y})] \in \text{span} \left\{ \partial_{x_j}^a \left( \overline{\varpi} \cdot \overline{\partial\varpi f}(\overline{y}) \right) \right\}_{\|\overline{\varpi}\|_1 \leq (a_1 + \cdots + a_{j-1})(d-1) \atop \|\overline{\varpi}\|_1 \leq a_1 + \cdots + a_{j-1}}
\subseteq \text{span} \left\{ \overline{\varpi} x_j^{-r} \cdot \overline{\varpi}^r [\partial_{\overline{\varpi} f}(\overline{y})] \right\}_{\|\overline{\varpi}\|_1 \leq (a_1 + \cdots + a_{j-1})(d-1) \atop \|\overline{\varpi}\|_1 \leq a_1 + \cdots + a_{j-1}}
\]

Renaming variables \( \overline{\varpi} \leftarrow \overline{\varpi} + r \overline{\varpi} \) and \( \overline{\varpi} \leftarrow \overline{\varpi} + \overline{\varpi} \) so that \( \|\overline{\varpi}\|_1 \leq (a_1 + \cdots + a_{j-1})(d-1) + s(d-1) - r \leq (a_1 + \cdots + a_{j-1})(d-1) + a_j(d-1) \) and \( \|\overline{\varpi}\|_1 \leq a_1 + \cdots + a_{j-1} + s \),
\[ 
\subseteq \text{span} \left\{ \overline{\varpi} [\partial_{\overline{\varpi} f}(\overline{y})] \right\}_{\|\overline{\varpi}\|_1 \leq (a_1 + \cdots + a_{j-1} + a_j)(d-1) \atop \|\overline{\varpi}\|_1 \leq a_1 + \cdots + a_{j-1} + a_j}
\]

which is the inductive claim as \( \|\overline{\varpi}\|_1 = a_1 + \cdots + a_{j-1} + a_j \).

Noticing that we can multiply on arbitrary monomials of \( \overline{\varpi} \) on both sides of the
above, the following corollary is immediate, and was established by Gupta, Kamath,
Kayal, and Saptharishi [GKKS13a].

**Corollary C.2.10 ([GKKS13a]).** Let $f \in \mathbb{F}[y_1, \ldots, y_m]$. Suppose $\mathbf{g} \in (\mathbb{F}[x_1, \ldots, x_n]^m$, where each $g_i$ is of degree $\leq d$. Then for any $t \geq 0$,

$$\mathbf{x}^{\leq \ell} \partial_{\mathbf{y}^{\leq t}} (f \circ \mathbf{g}) \subseteq \mathbf{x}^{\leq t(d-1) + \ell} \left( [\partial_{\mathbf{y}^{\leq t}} (f)] (\mathbf{g}) \right).$$

As in the above remark, these corollaries still makes sense when $d = 0$ assuming the convention of deg 0 = $-\infty$.

The above is most useful when applied to counting dimensions of the spaces of polynomials. Unlike the above, we will now need to assume $d \geq 1$, and also will need to analyze bounded and unbounded $t$ separately. We first analyze the dimensions for unbounded $t$.

**Corollary C.2.11 ([GKKS13a]).** Let $d \geq 1$ and let $f \in \mathbb{F}[y_1, \ldots, y_m]$. Suppose $\mathbf{g} \in (\mathbb{F}[x_1, \ldots, x_n]^m$, where each $g_i$ is of degree $\leq d$. Then for any $0 \leq t < \infty$,

$$\dim \mathbf{x}^{\leq \ell} \partial_{\mathbf{y}^{\leq t}} (f \circ \mathbf{g}) \leq \begin{pmatrix} n + t(d-1) + \ell \\ t(d-1) + \ell \end{pmatrix} \cdot \dim [\partial_{\mathbf{y}^{\leq t}} (f)] (\mathbf{g}) \leq \begin{pmatrix} n + t(d-1) + \ell \\ t(d-1) + \ell \end{pmatrix} \cdot \begin{pmatrix} m + t \\ t \end{pmatrix}.$$

**Proof:** This follows from Corollary C.2.10 and counting the number of elements in the spanning set. That is, (when $d \geq 1$) there are at $\begin{pmatrix} n + t(d-1) + \ell \\ t(d-1) + \ell \end{pmatrix}$ shifts $\mathbf{x}^\mathbf{a}$ with $\|\mathbf{a}\|_1 \leq t(d-1) + \ell$, as $\mathbf{x}$ has $n$ variables. Similarly, there are $\begin{pmatrix} m + t \\ t \end{pmatrix}$ possible derivatives $\mathbf{y}^\mathbf{b}$ with $\|\mathbf{b}\|_1 \leq t$. \qed

We now consider unbounded $t$, but only when $d = 1$. This is because otherwise the expression “$t(d-1)$” would become too large.

**Corollary C.2.12 ([GKKS13a]).** Let $f \in \mathbb{F}[y_1, \ldots, y_m]$. Suppose $\mathbf{g} \in (\mathbb{F}[x_1, \ldots, x_n]^m$, where each $g_i$ is of degree $\leq 1$. Then for any $t \geq 0$,

$$\dim \mathbf{x}^{\leq \ell} \partial_{\mathbf{y}^{\leq t}} (f \circ \mathbf{g}) \leq \begin{pmatrix} n + \ell \\ \ell \end{pmatrix} \cdot \dim [\partial_{\mathbf{y}^{\leq t}} (f)] (\mathbf{g}) \leq \begin{pmatrix} n + \ell \\ \ell \end{pmatrix} \cdot \begin{pmatrix} m + t \\ t \end{pmatrix}.$$

In particular,

$$\dim \mathbf{x}^{\leq \ell} \partial_{\mathbf{y}^{< \infty}} (f \circ \mathbf{g}) \leq \begin{pmatrix} n + \ell \\ \ell \end{pmatrix} \cdot \dim [\partial_{\mathbf{y}^{< \infty}} (f)] (\mathbf{g}) \cdot$$

**Proof:** As $d = 1$, Corollary C.2.11 can be applied with the observation that for any finite $t$, $t(d-1) = 0$. Thus, as above, we can simply count the number of shifts (which is $\begin{pmatrix} n + \ell \\ \ell \end{pmatrix}$), as well as the number of derivatives (which is $\begin{pmatrix} m + t \\ t \end{pmatrix}$, when $t$ is finite). The
last statement follows from letting $t$ be larger than the degrees of all polynomials involved, so that any further derivatives simply yield zero.
Appendix D

Counting Subspaces

Abstract: In this chapter we establish estimates for the number of low-dimensional subspaces in a high-dimensional space, over finite fields, as well as giving good estimates for the probability a random matrix is full-rank. While it is easy to give somewhat-loose estimates for these quantities, we derive fairly tight bounds here so that we can get a fairly tight bound for the number of seeds in the non-explicit construction of a rank-condenser given in Lemma 5.3.3.

We first define \( q \)-analogues of usual combinatorial notions such as factorials and binomial coefficients. We then show that these numbers exactly count low-dimensional subspaces, and then give more useful approximations for these numbers.

D.1 \( q \)-Analogues

Counting subsets of size \( k \) in \([n]\) can be done with the usual binomial coefficients. In this section, we define the \( q \)-analogue of this notion, which will count \( k \)-dimensional subspaces of \( \mathbb{F}_q^n \). In particular, we generalize familiar notions (integers, factorials, binomial coefficients) to depend on a parameter \( q \) such that as \( q \to 1 \) the usual notion is recovered, but for \( q > 1 \) we obtain counting results for \( \mathbb{F}_q \). We begin by defining the \( q \)-analogue of a non-negative integer.

Definition D.1.1. For \( n \geq 0 \), the \( q \)-analogue of \( n \) is \( [n]_q := q^{n-1} + \cdots + q^0 \). \( \diamondsuit \)

Note that for \( n = 0 \) the above summation is empty, so \( [0]_q = 0 \). It is then straightforward to see that as \( q \to 1 \) we recover the usual notion of a natural number.

Lemma D.1.2. \( \lim_{q \to 1} [n]_q = [n]_1 = n \) \( \square \)

Often we will consider \( q > 1 \), and consider \( q \to 1 \) as a limiting statement, so that “\( q = 1 \)” never occurs. In this situation, we have the following equivalent definition of \( [n]_q \).

Lemma D.1.3. For \( q \neq 1 \), \( [n]_q = \frac{q^n - 1}{q - 1} \). \( \square \)

We now define \( q \)-analogues of factorials and binomial coefficients.

Definition D.1.4. The \( q \)-factorial \( n!_q \) is defined by \( n!_q := [n]_q \cdots [1]_q \). For \( 0 \leq k \leq n \), the \( q \)-binomial coefficient is defined by \( \binom{n}{k}_q := \frac{[n]_q}{[k]_q [n-k]_q} \). We have the following identities:

\[
\binom{n}{k}_q = \frac{q^n}{q^k q^{n-k}} = \binom{n}{k}
\]

where \( \binom{n}{k} \) is the usual binomial coefficient.
n, the q-binomial \( \binom{n}{k}_q \) is defined by

\[
\binom{n}{k}_q := \frac{n!_q}{k!_q (n-k)!_q},
\]

and is zero otherwise.

From the symmetry in the definition of \( \binom{n}{k}_q \) the following is immediate.

**Lemma D.1.5.**

\[
\binom{n}{k}_q = \binom{n}{n-k}_q.
\]

\(\square\)

When \( q \to 1 \) we recover the usual factorials and binomial coefficients, as it follows from the similar property of the \( q \)-integers (Lemma D.1.2).

**Corollary D.1.6.** \( \lim_{q \to 1} n!_q = n!_1 = n! \) and \( \lim_{q \to 1} \binom{n}{k}_q = \binom{n}{k}_1 = \binom{n}{k} \).

\(\square\)

## D.2 Counting Subspaces

The previous section introduced the q-binomial coefficients and showed that for \( q \to 1 \) they recover the usual notion of binomial coefficients. As the binomial coefficient \( \binom{n}{k} \) counts size-k subsets of \([n]\), we now show a similar property for \( \binom{n}{k}_q \) — that is, \( \binom{n}{k}_q \) counts the number of dimension \( k \) subspaces of \( \mathbb{F}_q^n \).

As \( q > 1 \) in this discussion, we will make use of expressions only valid in this case (eg, Lemma D.1.3), in particular the following lemma.

**Lemma D.2.1.** For \( q > 1 \),

\[
\binom{n}{k}_q = \frac{(q^n - q^0) \cdots (q^n - q^{k-1})}{(q^k - q^0) \cdots (q^k - q^{k-1})}.
\]

**Proof:**

\[
\binom{n}{k}_q = \frac{n!_q}{k!_q (n-k)!_q} = \frac{[n]_q \cdots [n-k+1]_q [n-k]_q \cdots [1]_q}{[k]_q \cdots [1]_q} \cdot \frac{[n]_q \cdots [n-k+1]_q}{[k]_q \cdots [1]_q} \cdot \frac{(q-1)^k}{(q-1)^k} = \frac{[n]_q \cdots [n-k+1]_q}{[k]_q \cdots [1]_q} \cdot \frac{(q-1)^k}{(q-1)^k}.
\]
By Lemma D.1.3,
\[
(q^n - 1) \cdots (q^{n-k+1} - 1) \\
(q^k - 1) \cdots (q^1 - 1)
\]
\[
= \prod_{i=0}^{k-1} q^{n-i} - 1 = \prod_{i=0}^{k-1} q^{n-i} - 1 \cdot \frac{q^i}{q^i} = \prod_{i=0}^{k-1} q^n - q^i
\]
\[
= \frac{(q^n - q^0) \cdots (q^n - q^{k-1})}{(q^k - q^0) \cdots (q^k - q^{k-1})}.
\]
\[\square\]

With the above in hand, we now turn to counting matrices.

**Lemma D.2.2.** The number of matrices \(\mathbb{F}_q^{n \times k}\) of rank \(k\) is equal to \((q^n - q^0) \cdots (q^n - q^{k-1})\).

**Proof:** Consider a matrix \(A \in \mathbb{F}_q^{n \times k}\), denoting its \(k\) columns as \(v_1, \ldots, v_k \in \mathbb{F}_q^n\). Then \(A\) is rank \(k\) iff \(v_i \not\in \text{span}\{v_1, \ldots, v_{i-1}\} =: V_{i-1}\) for all \(i \in [k]\). Thus, \(A\) is rank \(k\) iff for all \(i \in [k]\), \(V_{i-1}\) is of rank \(i - 1\) and thus \(|V_{i-1}| = q^{i-1}\). Note that \(V_0 = \text{span}\{0\}\), so \(|V_0| = q^0 = 1\). Thus, for a rank \(k\) matrix there are \(q^n - |V_{i-1}| = q^n - q^{i-1}\) choices for \(v_i\). Thus there are \((q^n - q^0) \cdots (q^n - q^{k-1})\) choices for all of the \(v_i\), and thus that many choices for \(A\).
\[\square\]

We briefly digress on counting matrices, although this section is ostensibly about counting subspaces. That is, the above exactly gives the probability that a \(k \times n\) matrix is full-rank, but this quantity is somewhat numerically opaque. While one can given fairly easy loose upper bounds (eg, Lemma 5.3.2), these loose bounds are terrible when \(k = n\) and \(q = 2\). As such, we now give an asymptotically more useful version of the above formula that is fairly tight when \(k = n\), which we base on the scribe notes of Williams [Wil13]. We first give an approximation of the logarithm that will be needed.

**Lemma D.2.3.** Let \(x \in [x_0, 0]\) for \(x_0 \in (-1, 0)\). Then \(\ln(1 + x) \geq \frac{\ln(1+x_0)}{x_0} \cdot x\).

**Proof:** Take \(f(x) := \ln(1 + x)\). Then \(f'(x) = \frac{1}{1+x}\) and \(f''(x) = -\frac{1}{(1+x)^2}\), so that \(f''(x) < 0\) for \(x > -1\). Thus, \(\ln(1+x)\) is concave. By taking \(\lambda = x/x_0 \in [0, 1]\), concavity implies that \(\ln(1+x) = f(\lambda x_0) = f(\lambda x_0 + (1-\lambda)0) \geq \lambda f(x_0) + (1-\lambda)f(0) = \lambda f(x_0) = x/x_0 \ln(1 + x_0)\), as desired.
\[\square\]

We now use this approximation to estimate the probability a random matrix is full rank.

**Lemma D.2.4.** Let \(n \geq k\). The probability a random matrix in \(\mathbb{F}_q^{n \times k}\) has rank \(k\) is \((q^n - q^0) \cdots (q^n - q^{k-1})/q^{kn} > \left(1 - \frac{1}{q^{n-k+1}}\right)^{\frac{n}{k}}\).
Proof: The first part of the bound follows from Lemma D.2.2. We now use analysis to get the second part.

\[
\frac{(q^n - q^0) \cdots (q^n - q^{k-1})}{q^{kn}} = \prod_{i \in [k]} \frac{q^n - q^i}{q^n} = \prod_{i} 1 - q^{i-n} = \prod_{j=n-k+1}^{n} 1 - q^{-j} = \exp \left( \sum_{j} \ln \left( 1 - q^{-j} \right) \right)
\]

using that \( \ln(1 - q^{-j}) \geq \frac{\ln(1-q^{-(n-k+1)})}{q^{-(n-k+1)}} \cdot -q^{-j} \) for \( j \geq n - k + 1 \) (Lemma D.2.3),

\[
\geq \exp \left( \sum_{j} \frac{\ln(1-q^{-(n-k+1)})}{q^{-(n-k+1)}} \cdot -q^{-j} \right) = \exp \left( \frac{\ln(1-q^{-(n-k+1)})}{q^{-(n-k+1)}} \cdot \sum_{j} q^{-j} \right)
\]

as \( \ln(1 - q^{-(n-k+1)}) < 0, \)

\[
> \exp \left( \frac{\ln(1-q^{-(n-k+1)})}{q^{-(n-k+1)}} \cdot \sum_{j=n-k+1}^{\infty} q^{-j} \right) = \exp \left( \ln(1-q^{-(n-k+1)}) \cdot \frac{q^{-(n-k+1)}}{1-q^{-1}} \right) = \left(1 - \frac{1}{q^{n-k+1}}\right)^{\frac{q}{q-1}} \]

\[\Box\]

Remark D.2.5. Note that one could also use the bound \( \ln(1 + x) \geq \frac{x}{1+x} \) for \( x > -1 \), instead of Lemma D.2.3. However, this would yield worse results as this alternate bound, while better when \( x \approx 0 \), is worse when \( x \gg 0 \), which will happen in the above application for small \( q \).

Remark D.2.6. Note that when \( n = k \) we get \( (1 - 1/q)^{\frac{q}{q-1}} \). As \( q \to \infty \), \( 1-1/q \to e^{-1/q} \) so that \( (1 - 1/q)^{\frac{q}{q-1}} \to \exp(-1/(q - 1)) \), and as \( q \to \infty \) this approaches 1.

Specializing the above for \( \mathbb{F}_2 \) we obtain the following corollary.

Corollary D.2.7. The probability a random matrix in \( \mathbb{F}_2^{k \times k} \) has rank < \( k \) is < 3/4. \[\Box\]

We now return to counting subspaces.

Lemma D.2.8. The number of dimension \( k \) subspaces of \( \mathbb{F}_q^n \) is \( \binom{n}{k}_q \).
Proof: Consider the map from rank $k$ matrices in $\mathbb{F}^{n \times k}$ to $k$-dimensional subspaces of $\mathbb{F}_q^n$ defined by $A \mapsto \text{col-span}(A)$. This map is well-defined and surjective. Consider then the pre-image of a rank $k$ subspace $V \subseteq \mathbb{F}_q^n$. Each of these matrices is related by an invertible change of basis given by an invertible $k \times k$ matrix. Thus, the pre-images of this map are of size $(q^k - q^0) \cdots (q^k - q^{k-1})$ by Lemma D.2.2. The claim follows by applying double-counting, using Lemma D.2.2 to see that there are $(q^n - q^0) \cdots (q^n - q^{k-1})$ rank $k$ matrices, and using Lemma D.2.1 to relate the resulting quantity to the $q$-binomial coefficient.

The above result allows us to show the symmetry in $q$-binomial coefficients between $k$ and $n-k$ in a different manner than that of Lemma D.1.5.

Corollary D.2.9. $(\binom{n}{k})_q = (\binom{n}{n-k})_q$

Proof: Note that there is a bijection between $k$-dimensional subspaces and $(n-k)$-dimensional subspaces, via the dual space map $V \mapsto V^\perp$. The claim then follows from the above Lemma D.2.8.

While the above counting result is exact, the following cruder bounds are more useful in approximations. We begin with the lower bound.

Lemma D.2.10. For $q > 1$ and for $n \geq k$, $(\binom{n}{k})_q \geq q^{k(n-k)}$.

Proof:

\[
\binom{n}{k}_q = \frac{(q^n - 1)(q^n - q)(q^n - q^2) \cdots (q^n - q^{k-1})}{(q^k - 1)(q^k - q) \cdots (q^k - q^{k-1})}
= \prod_{i=0}^{k-1} \frac{q^n - q^i}{q^k - q^i} = \prod_{i=0}^{k-1} \frac{q^{n-i} - 1}{q^{k-i} - 1}
\]

As $\frac{x-1}{y-1} \geq \frac{x}{y}$ when $x \geq y$,

\[
\geq \prod q^{n-k} = q^{k(n-k)}.
\]

We now establish a very weak upper bound.

Lemma D.2.11. For $q > 1$ and for $n \geq k$, the number of dimension $\leq k$ subspaces of $\mathbb{F}_q^n$ is $\sum_{i=0}^{k} \binom{n}{i}_q \leq q^{kn}$.

Proof: That $\sum_{i=0}^{k} \binom{n}{i}_q$ counts the number of subspaces follows from Lemma D.2.8. The upper bound follows from the observation that the map from $n \times k$ matrices to dimension $\leq k$ subspaces is surjective.

We follow up with another bound using the symmetry between $k$ and $n-k$.

Corollary D.2.12. For $q > 1$ and for $n \geq k$, the number of dimension $k$ subspaces of $\mathbb{F}_q^n$ is $\leq \min\{q^{kn}, q^{(n-k)n}\}$. Further, this inequality is strict for $k \in (0, n)$.
Proof: That \( \binom{n}{k}_q \leq q^{kn} \) follows from the above Lemma D.2.11, while the \( q^{(n-k)n} \) upper bound follows from the symmetry between \( k \) and \( n - k \) from Lemma D.1.5. The strictness of the inequality for \( k \in (0,n) \) also follows from Lemma D.2.11 as we can always discard the zero dimensional subspace.

We now get a sharper upper bound.

Lemma D.2.13. For \( n \geq k \) and \( q > 1 \), \( \binom{n}{k}_q \leq e^{\frac{k}{q-1}} \cdot q^{k(n-k)}. \)

Proof:

\[
\binom{n}{k}_q = \frac{(q^n - q^0) \cdots (q^n - q^{k-1})}{(q^k - q^0) \cdots (q^k - q^{k-1})} = \prod_{i=0}^{k-1} \frac{q^n - q^i}{q^k - q^i} = \prod \frac{q^{n-i} - 1}{q^{k-i} - 1}
\]

As \( i \leq k - 1 \), \( 1 \leq q^{k-i-1} \),

\[
\leq \prod \frac{q^{n-i}}{q^{k-i} - q^{k-i-1}} = \prod \frac{q}{q - 1} \frac{q^{n-i-1}}{q^{k-i-1}} = \left( \frac{q}{q - 1} q^{n-k} \right)^k = \left( 1 + \frac{1}{q - 1} \right)^k q^{k(n-k)}
\]

As \( 1 + x \leq e^x \),

\[
\leq e^{\frac{k}{q-1}} q^{k(n-k)}. \]

Remark D.2.14. At first glance, the above bound is somewhat counter-intuitive, as one might expect the number to be closer to \( \binom{n}{k}_q \cdot q^{k(n-k)}. \) This number is more in line with one expects by analogy to counting sparse vectors in \( \mathbb{F}^n_q \), as one must pick the sparsity pattern as well as the values for those entries.

Note that this number is an upper bound, as it upper bounds the number of matrices in reduced row-echelon form, which are a canonical form for matrices. That is, reduced row-echelon form for rank-\( k \) \( k \times n \) matrices consists of choosing \( k \) pivot columns out of the \( n \) columns, placing an identity matrices in these columns, and then filling in those columns that are larger than the first pivot column using arbitrary values from \( \mathbb{F}_q \). There are \( \binom{n}{k} \) choices of pivot columns, and there are at most \( q^{k(n-k)} \) choices for the other entries as there are at most \( n - k \) columns not used as a pivot column.

While the above bound works, and might at first feel tight, the above bound is inherently loose. That is, once we pick the first pivot column as column \( i \), the first \( i - 1 \) columns will then become zero. Further, on average one expects \( i \approx n/k \), so that much of the time there are only \( \approx k(n - k - n/k) \) arbitrary entries to fill in after choosing the pivot columns. As \( q^{k(n-k-n/k)} \ll q^{k(n-k)} \), especially when \( q \) is large, this shows that not all choices of pivot columns contribute to the number of subspaces equally, so the above
bound is quite loose. In particular, one expects a sort of geometric sum so that the number of subspaces is approximately equal to the number \( q^{k(n-k)} \) of subspaces formed from having the first \( k \) columns be pivot columns, and this is indeed what happens. ♦

This shows that as \( q \to \infty \) the number of \( k \)-dimensional subspaces of \( \mathbb{F}_q^n \) is \( \approx q^{k(n-k)} \).

As a simply corollary, we can symmetrize the above bound with respect to \( k \) and \( n - k \) (by Lemma D.1.5).

**Corollary D.2.15.** For \( n \geq k \) and \( q > 1 \), \( \binom{n}{k}_q \leq e^{\frac{\min(k,n-k)}{q-1}} \cdot q^{k(n-k)} \).

We now show that the second bound subsumes the first, for \( q \geq 3 \). We start with the following approximation.

**Lemma D.2.16.** For \( q \geq 3 \), \( e^{1 \over q-1} \leq q \).

*Proof:* As \( q \geq 3 \), it follows that \( q - 1 \geq 2 \) and \( q \geq e \), so that \( e^{1 \over q-1} \leq e^{1 \over 1} \leq q = q \).

It is evident that the approximation is false for \( q = 2 \), so this will have to be excluded in the below usage. We now use this approx to get a approximate version of the above bound.

**Corollary D.2.17.** For \( n \geq k \) and \( q \geq 3 \), \( \binom{n}{k}_q \leq \min\{q^{k(n-k+1)}, q^{(k+1)(n-k)}\} \).

*Proof:* Using the above bound (Lemma D.2.13), it suffices to show that \( e^{k \over q-1} q^{k(n-k)} \leq q^{k(n-k+1)} \) and appeal to the symmetry between \( k \) and \( n - k \) (Lemma D.1.5). This bound is equivalent to showing that \( e^{k \over q-1} \leq q^k \), which follows from \( e^{1 \over q-1} \leq q \) given that \( k \geq 0 \), as given from Lemma D.2.16.

Noting that \( k(n-k+1) = k(n-k) + k \leq k(n-k) + k^2 = kn \) and symmetrically that \( (k+1)(n-k) \leq (n-k)n \), it follows that Corollary D.2.17 is stronger than Corollary D.2.12.
Bibliography


An introduction to computational algebraic geometry and commutative algebra. 7, 139, 140, 156


[HS80] Joos Heintz and Claus-Peter Schnorr. Testing polynomials which are easy to compute (extended abstract). In *Proceedings of the 12th Annual ACM


[JQS10] Maurice J. Jansen, Youming Qiao, and Jayalal M. N. Sarma. Deterministic black-box identity testing π-ordered algebraic branching programs. 214


[Kay12] Neeraj Kayal. An exponential lower bound for the sum of powers of bounded degree polynomials. Electronic Colloquium on Computational Complexity (ECCC), 19(81), 2012. 194


