On Structure and Lower Bounds in Restricted Models of Arithmetic Computations

By

S. Raja

MATH10201004009

The Institute of Mathematical Sciences, Chennai

A thesis submitted to the

Board of Studies in Mathematical Sciences

In partial fulfillment of requirements

For the Degree of

DOCTOR OF PHILOSOPHY

of

HOMI BHABHA NATIONAL INSTITUTE

April, 2017

Homi Bhabha National Institute

Recommendations of the Viva Voce Board

As members of the Viva Voce Committee, we certify that we have read the dissertation prepared by S. Raja entitled “On Structure and Lower Bounds in Restricted Models of Arithmetic Computations" and recommend that it maybe accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

_____ Date:

Chair - Prof. Meena Mahajan

_____ Date:

Guide/Convener - Prof. V. Arvind

_____ Date:

Member 1 - Prof. C. R. Subramanian

_____ Date:

Member 2 - Prof. Venkatesh Raman

_____ Date:

Examiner - Prof. Chandan Saha

Final approval and acceptance of this dissertation is contingent upon the candidate’s submission of the final copies of the dissertation to HBNI.

I hereby certify that I have read this thesis prepared under my direction and recommend that it may be accepted as fulfilling the thesis requirement.

Date:

Place: Guide

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at Homi Bhabha National Institute (HBNI) and is deposited in the Library to be made available to borrowers under rules of the HBNI.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgement of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the Competent Authority of HBNI when in his or her judgement the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

S. Raja

DECLARATION

I, hereby declare that the investigation presented in the thesis has been carried out by me. The work is original and has not been submitted earlier as a whole or in part for a degree / diploma at this or any other Institution / University.

S. Raja

List of Publications arising from the thesis

Journals

1. V. Arvind, **S. Raja** : Some Lower Bound Results for Set-Multilinear Arithmetic Computations. Chicago J. Theor. Comput. Sci. 2016 (6)

2. V. Arvind, Pushkar S. Joglekar, **S. Raja**: Noncommutative Valiant's Classes: Structure and Complete Problems. ACM Transactions on Computation Theory (ToCT) 9(1): 3:1-3:29 (2016)

Conference

1. V. Arvind, **S. Raja**, A. V. Sreejith: On lower bounds for multiplicative circuits and linear circuits in noncommutative domains. In Proc. CSR 2014, LNCS 8476, pages 65–76. Springer, 2014.

S. Raja

ACKNOWLEDGEMENTS

I am deeply indebted to my advisor V. Arvind. I thank him for his advice and encouragement throughout my stay at IMSc. I have learnt a lot about research in general and I admire him for his clarity in thinking and explaining new ideas. I thank him for all his advice and help both academic or otherwise. I am very fortunate for having him as my advisor.

I thank all my teachers: V. Arvind, Kamal Lodaya, Meena Mahajan, R. Ramanujam, Saket Saurabh, C.R. Subramanian, Venkatesh Raman, Vikram Sharma, Prahladh Harsha (TIFR) and Partha Mukhopadhyay (CMI) for their wonderful courses and encouragement. I thank Meena Mahajan for organizing many seminar series. I thank Vikram Sharma for many of his courses and I admire him for his teaching effort.

I thank my co-authors Pushkar Joglekar and A.V. Sreejith for fruitful discussions.

I thank all my friends at CMI, IIT-M and IMSc. I want to specially thank Anil Shukla, Anup, Joydeep, Ramchandra and Srinivasan for all their help and time.

I want to thank all the people in the IMSc administration for all their help and support. I want to thank both technical and non-technical staffs of IMSc (Canteen, Civil, Computer, Drivers, Electrical, Gardners, House keeping, Hostel, Library and Securities) for making IMSc a nice place.

Finally, I want to thank Amma, Ammayee, Mama and all my other relatives for their love and support.

Contents

Synopsis

In this thesis, we study structure and lower bounds questions for restricted models of arithmetic computations.

First, we study depth reduction and lower bounds questions for set-multilinear arithmetic computations. For depth reduction, we observe that set-multilinear arithmetic circuits of size s can be efficiently depth reduced, producing logarithmic depth set-multilinear circuits of size $O(s^3)$ with multiplication gates of fanin 2 and unbounded fanin addition gates, similar to depth reduction results of Valiant et al., [VSBR83] for general arithmetic circuits and Raz-Yehudayoff [RY08] for syntactic multilinear arithmetic circuits. For lower bound questions, we study restricted set-multilinear branching programs where type-width of the branching program is restricted. The *type-width* of a set-multilinear ABP at layer i is number of distinct sets labeling nodes at layer i. We show that *k-narrow* set-multilinear ABPs computing the Permanent polynomial PER_n (or determinant DET_n) require $2^{\Omega(k)}$ size. As a consequence, it follows that the sum of r read-once oblivious ABPs computing PER_n requires size $2^{\Omega(\frac{n}{r})}$. We also consider other restrictions like restricting total number of parse trees allowed in the circuit and requiring each monomial is generated by unique parse tree type, and show exponential lower bounds for these restricted set-multilinear circuits computing explicit polynomial families. We also show that set-multilinear branching programs are exponentially more powerful than *interval* multilinear circuits (where the index sets for each gate is restricted to be an interval w.r.t. some ordering), assuming the sum-of-squares conjecture [Sha00, HWY10a]. This further underlines the power of set-multilinear branching programs.

Next, we study the noncommutative analogues, VP_{nc} and VNP_{nc}, of Valiant's algebraic complexity classes, defined in [HWY10b]. We show that Dyck polynomials (defined from the Dyck languages of formal language theory) are complete for the class VP_{nc} under $\leq_{abp}$ reductions. Likewise, it turns out that PAL (Palindrome polynomials defined from palin-

dromes) are complete for the class VSKEW_{nc} (defined by polynomial-size skew circuits) under $\leq_{abp}$ reductions. The proof of these results is obtained by suitably adapting the classical Chomsky-Schützenberger theorem [CS63], which originally shows that Dyck languages are the hardest CFLs. Next, assuming $\text{VP}_{nc} \neq \text{VNP}_{nc}$, we exhibit a strictly infinite hierarchy of p-families, with respect to the projection reducibility, between the complexity classes VP_{nc} and VNP_{nc} (analogous to Ladner's theorem [Lad75]). We also show that, inside the class VP_{nc} there is a strict hierarchy of p-families (based on the nesting depth of Dyck polynomials) with respect to the $\leq_{abp}$-reducibility.

Finally, we study multiplicative and linear circuits over noncommutative domains. We show strong lower bounds on the size of multiplicative circuits computing a multi-output function over various noncommutative domains like free monoid over alphabet of size $\geqslant 2$, finite matrix semigroup, free group generated by $X = \{x_1, x_2, x_1^{-1}, x_2^{-1}\}$ and permutation groups. We then study size lower bound questions for linear circuits computing $\mathrm{A}\mathbf{y}$, where $\mathbf{y} = (y_1, y_2, \cdots, y_n)^T$ and A is a matrix of dimension $m \times n$ with entries from a noncommutative ring. We construct such an explicit matrix $A \in \mathbb{F}^{n\times n}\langle x_0, x_1\rangle$ such that $A\mathbf{y}$ cannot be computed by any circuit C with size $O(n)$ and depth $O(\log n)$. We prove this by suitably generalizing Valiant's matrix rigidity method [Val77], which is originally defined for matrices over fields. We next consider homogeneous depth 2 linear circuits over noncommutative rings. These are linear circuits of depth 2, in which each addition gate can have unbounded fanin. We exhibit an explicit matrix A and show that computing $A\mathbf{y}$ by a depth 2 homogeneous linear circuit (with unbounded fanin + gates) requires $\Omega(\frac{n^2}{\log n})$ wires. In contrast, for depth 2 linear circuits over fields only $\Omega(n \log^2 n/\log\log n)$ lower bound is known for explicit matrices [Lok09, Pud94], although for random matrices the lower bound is $\Omega(n^2/\log n)$.

Chapter 1

Introduction to the thesis

1.1 Introduction

In this chapter, we give an introduction to the thesis. In this section, we give a brief introduction to our field of study: arithmetic circuit complexity theory and in Section 1.2, we give an introduction to the main results of this thesis.

Arithmetic circuit complexity theory is a subfield of computational complexity theory. A central question here is determining number of additions and multiplications required to compute polynomials over set of indeterminates $X = \{x_1, x_2, \cdots, x_n\}$ in the polynomial ring $\mathbb{F}[X]$ where $\mathbb{F}$ is a field. Arithmetic circuits are a natural computational model for computing and describing polynomials.

Definition 1.1.1 (Arithmetic circuits)**.** *An* arithmetic circuit *C computing a polynomial $f \in \mathbb{F}[X]$, is a directed acyclic graph such that each in-degree* 0 *node of the graph is labeled with an element from $X \cup \mathbb{F}$. Each internal node v of C is of indegree* 2*, and is either a + gate or $\times$ gate. Each gate of the circuit inductively computes a polynomial in $\mathbb{F}[X]$: the polynomials computed at the input nodes are the labels; the polynomial computed at a + gate (resp. $\times$ gate) is the sum (resp. product) of the polynomials computed at its*

children. The output gate of C computes the polynomial f. The size *of the circuit C is the number of gates in it and its* depth *is the length of the longest path from an input gate to the output gate of C.*

Additionally, if every gate in a circuit C has outdegree bounded by 1 then C is an *arithmetic formula*.

Before giving examples of polynomials, we need the following definition of polynomial family (p-family).

Definition 1.1.2 (p-family)**.** *A polynomial family (p-family) $f = (f_n)$ is a sequence of polynomials such that number of variables and degree in the n-th polynomial grows polynomially.*

We study circuit families $C = (C_n)$ computing the polynomial family $f = (f_n)$ such that the circuit C_n computes the polynomial f_n.

Two important examples of polynomial family are the determinant $\mathrm{DET} = (\mathrm{DET}_n)$ and permanent $\mathrm{PER} = (\mathrm{PER}_n)$ polynomial families, which we define now.

$$\begin{aligned} \mathrm{DET}_n &= \sum_{\sigma \in S_n} \mathrm{sgn}(\sigma) x_{1,\sigma(1)} x_{2,\sigma(2)} \cdots x_{n,\sigma(n)} \\ \mathrm{PER}_n &= \sum_{\sigma \in S_n} x_{1,\sigma(1)} x_{2,\sigma(2)} \cdots x_{n,\sigma(n)} \end{aligned}$$

where S_n is the group of permutations on $[n]$ and for $\sigma \in S_n$, $sgn(\sigma)$ denote the sign of permutation σ. The determinant polynomial is ubiquitous in linear algebra and it can be computed by polynomial sized arithmetic circuits (see e.g., [Ber84]). On the other hand, the permanent of 0/1 matrices is #P-complete [Val79b], where #P corresponds to the counting class in the world of Boolean complexity classes. Thus, it is believed [Val79a, Bür00a] that, over fields of characteristic different from 2, the permanent $\mathrm{PER} = (\mathrm{PER}_n)$ p-family cannot be computed by any polynomial sized circuit family.

A central open problem of the field is proving superpolynomial size lower bounds for

arithmetic circuits that compute the permanent polynomial PER_n. Motivated by this problem, Valiant, in his seminal work [Val79a], defined the arithmetic analogues of P and NP: namely VP and VNP. Informally, VP consists of multivariate (commutative) polynomials that have polynomial size circuits, over the field of rationals. The class VNP (which corresponds to the counting class #P in the world of Boolean complexity classes) has a more technical definition which we will give later. Valiant showed that PER_n is VNP-complete w.r.t. projection reductions. Thus, VP $\neq$ VNP iff PER_n requires arithmetic circuits of size superpolynomial in n. Over any field $\mathbb{F}$ the classes $\mathrm{VP}_{\mathbb{F}}$ and $\mathrm{VNP}_{\mathbb{F}}$ are similarly defined. Indeed, Valiant's proof actually shows that PER_n is complete for the class $\mathrm{VNP}_{\mathbb{F}}$ for any field $\mathbb{F}$ of characteristic different from 2. We will drop the subscript and simply use VP and VNP to denote the classes as the field $\mathbb{F}$ will either not matter or will be clear from the context.

Another central question in arithmetic circuit complexity theory is to come up with an explicit family of polynomials $f = (f_n)$ such that it cannot be computed by any polynomial sized circuits family. There are many notions of explicit p-families [Bür00a]. We use the following notion: a p-family $f = (f_n)$ is explicit if there is a deterministic Turing machine which, when given as input n and a monic monomial $x_1^{e_1} x_2^{e_2} \cdots x_n^{e_n}$, computes the coefficient of this monomial in the polynomial f_n in time $\mathsf{poly}(n)$. For example, DET = (DET_n) and PER = (PER_n) are explicit p-families.

If we allow degree of the polynomial to be arbitrary then strong lower bounds can be shown (as one can see that computing the polynomial x^d requires at least $\log d$ gates). Thus, if allow large degree d, for example: doubly exponential, then the polynomial requires exponential sized arithmetic circuit to compute it. To exclude trivial solutions of this form, we use the notion of p-family [Bür00a], as defined in Definition 1.1.2.

Over any field $\mathbb{F}$, by a standard counting argument, there exists a polynomial f (in particular, with 0/1 coefficients) of degree d in n variables such that any circuit computing f has size at least $\Omega(\sqrt{\binom{n+d}{d}})$ (see e.g., [SY10]). But it is not known for any explicit p-family

which requires more than polynomial size arithmetic circuits to compute it.

We now give formal definitions of VP and VNP below.

Definition 1.1.3.

1. *The class* VP *consists of p-families* $f = (f_n)$ *such that each* f_n *has an arithmetic circuit of size bounded by* n^b *for some* $b > 0$ *depending on* f.

2. *A p-family* $f = (f_n)$ *is in the class* VNP *if there exists a p-family* $g = (g_n) \in$ VP *such that for some polynomial* $p(n)$

$$f_n(x_1, \ldots, x_{q(n)}) = \sum_{y_1,\ldots,y_{r(n)} \in \{0,1\}} g_{p(n)}(x_1, \ldots, x_{m(n)}, y_1, \ldots, y_{r(n)}).$$

   where $r(n)$ *is polynomially bounded.*

The determinant polynomial family DET is in VP (see e.g., [Ber84]) and the permanent polynomial family PER is in VNP [Val79a]. Valiant also defined a notion of reduction, called projections (where variables are substituted by a variable or a scalar), between polynomial families and defined notion of complete problems for these classes under projections. For example, over fields of characteristic different from 2, the permanent polynomial family PER = (PER_n) is VNP-complete under projection reductions [Val79a]. It is clear from definitions that VP $\subseteq$ VNP. The central question in arithmetic complexity theory is whether or not VP = VNP. In arithmetic complexity theory, Valiant's hypothesis is VP $\neq$ VNP.

The arithmetic circuits considered above are commutative. The best known lower bounds for arithmetic circuits for computing an explicit polynomial family of degree d in n variables, where depth is unbounded, is $\Omega(n \log d)$ by Baur and Strassen [BS83]. Given the limited progress in proving lower bounds for general arithmetic circuits, it is natural to study other interesting classes of arithmetic circuits. Some of the classes of arithmetic

circuits considered are non-commutative arithmetic circuits, set-multilinear circuits etc (see for example, survey by Shpilka and Yehudayoff [SY10]).

Noncommutative circuits

Nisan, in his 1990 paper [Nis91], explored the complexity of *noncommutative* arithmetic computations, in particular the complexity of computing the permanent with noncommutative computations. The noncommutative polynomial ring, denoted by $\mathbb{F}\langle X\rangle$, over a field $\mathbb{F}$ in noncommuting variables X, consists of noncommuting polynomials in X. These are just $\mathbb{F}$-linear combinations of words (we call them monomials) over the alphabet $X = \{x_1, \ldots, x_n\}$.

Non-commutative arithmetic circuit families computes non-commutative polynomial families in a non-commutative polynomial ring $\mathbb{F}\langle X\rangle$, where multiplication is non-commutative (i.e., for distinct $x, y \in X$, $xy \neq yx$). We now give formal definition of noncommutative arithmetic circuits.

Definition 1.1.4 (Noncommutative arithmetic circuits)**.** *A* noncommutative arithmetic circuit *C over a field $\mathbb{F}$ is a directed acyclic graph such that each in-degree* 0 *node of the graph is labeled with an element from $X \cup \mathbb{F}$, where $X = \{x_1, x_2, \ldots, x_n\}$ is a set of noncommuting variables. Each internal node has fanin two and is labeled by either $(+)$ or $(\times)$ – meaning a $+$ or $\times$ gate, respectively. Furthermore, each $\times$ gate has a designated left child and a designated right child. Each gate of the circuit inductively computes a polynomial in $\mathbb{F}\langle X\rangle$: the polynomials computed at the input nodes are the labels; the polynomial computed at a $+$ gate (resp. $\times$ gate) is the sum (resp. product in left-to-right order) of the polynomials computed at its children. The circuit C computes the polynomial at the designated output node.*

For example, by fixing row ordering in each monomial, we obtain a noncommutative permanent PER p-family. The first nontrivial lower bound for noncommutative arithmetic

computation was shown by Nisan [Nis91], who showed an exponential lower bound for non-commutative arithmetic formulas (circuits where underlying undirected circuit graph is a tree) computing the palindrome polynomial $\mathrm{PAL}_n = \sum_{w \in \{x_0, x_1\}^n} ww^R \in \mathbb{F}\langle x_0, x_1 \rangle$. But there is a linear sized non-commutative skew arithmetic circuits computing the polynomial PAL_n [Nis91]. In fact, Nisan [Nis91] showed lower bound for non-commutative algebraic branching programs which can simulate noncommutative arithmetic formulas efficiently. We now define noncommutative skew arithmetic circuit.

Definition 1.1.5 (Noncommutative skew arithmetic circuits)**.** *A noncommutative arithmetic circuit is said to be* skew *if for every multiplication gate one of its inputs is a scalar or an indeterminate $x \in X$.*

In Limaye et al., [LMS15], it is shown that the p-family $\mathrm{PAL} = (\mathrm{PAL}_n \mathrm{PAL}_n)$ requires exponential sized noncommutative skew circuits, but there is a linear sized noncommutative circuit for it. Thus, this result separates noncommutative skew circuits from noncommutative circuits.

For noncommutative circuits, we do not know any better lower bound than commutative arithmetic circuits [BS83]. Hrubes et al., [HWY10a] have related the well-known sum-of-squares (in short, SOS) problem (also see [Sha00]), which is a problem about commutative polynomials, to the lower bounds for noncommutative arithmetic circuits. In Hrubes et al., [HWY10a], it is shown that assuming sum-of-squares (SOS) conjecture, noncommutative circuits computing the permanent PER p-family requires exponential size. Consider expressing the commutative biquadratic polynomial

$$\mathrm{SOS}_k(x_1, \ldots, x_k, y_1, \ldots, y_k) = (\sum_{i \in [k]} x_i^2)(\sum_{i \in [k]} y_i^2)$$

as a sum of squares ($\sum_{i \in [s]} f_i^2$), where f_i are all homogeneous bilinear polynomials with the minimum s. We refer to the problem of determining smallest s as the *sum-of-squares problem*.

The SOS conjecture [Sha00, HWY10a] states that over complex numbers (or the algebraic closure of any field of characteristic different from 2), for all k we have the lower bound $s = \Omega(k^{1+\epsilon})$ for some constant $\epsilon > 0$ independent of k. Note that over $\mathbb{F}_2$, $s = 1$. Over other fields, the trivial bounds are $k \leqslant s \leqslant k^2$. Over $\mathbb{Z}$ and $\mathbb{R}$, superlinear lower bound (in k) is known for s [HWY10a].

Set-multilinear circuits and polynomials

An interesting class of polynomials in $\mathbb{F}[X]$ are set-multilinear polynomials.

Definition 1.1.6 (Set-multilinear polynomials)**.** *A set-multilinear polynomial in the disjoint variables $X = \sqcup_{i=1}^{d} X_i$, where X_i are set of variables, is a homogeneous polynomial f of degree d such that each non-zero monomial in f has exactly one variable from each set X_i where $i \in [d]$.*

Set-multilinear arithmetic circuits are a natural model for computing set-multilinear polynomials.

Definition 1.1.7 (Set-multilinear arithmetic circuits)**.** *A* set-multilinear arithmetic circuit *C computing f w.r.t. the above partition of X, is a directed acyclic graph such that each in-degree 0 node of the graph is labeled with an element from $X \cup \mathbb{F}$. Each internal node v of C is of indegree 2, and is either a $+$ gate or $\times$ gate. With each gate v of we can associate a subset of indices $I_v \subseteq [d]$ and the polynomial C_v computed by the circuit at v is set-multilinear over the variable partition $\sqcup_{i \in I_v} X_i$. If v is a $+$ gate then for each input u of v $I_u = I_v$, and v is a $\times$ gate with inputs v_1 and v_2 then $I_v = I_{v_1} \sqcup I_{v_2}$. Clearly, in a set-multilinear circuit every gate computes a set-multilinear polynomial (in a syntactic sense). The output gate of C is labeled by $[d]$ and computes the polynomial f. The* size *of C is the number of gates in it and its* depth *is the length of the longest path from an input gate to the output gate of C.*

Set-multilinear circuits are introduced in the work of [NW95]. It is easy to see that each set-multilinear polynomial can be computed by a set-multilinear arithmetic circuit. Some of the well known polynomial families like PER, DET, are set-multilinear. For set-multilinear formulas, super polynomial size lower bounds are known [Raz09].

Algebraic branching programs (ABP) are another computational model for computing polynomials.

Definition 1.1.8 (Algebraic branching programs)**.** *An* algebraic branching program *ABP is a layered directed acyclic graph (DAG) with one in-degree zero vertex s called the* source*, and one out-degree zero vertex t, called the* sink*. The vertices of the DAG are partitioned into layers $0, 1, \ldots, d$, and edges go only from level i to level $i+1$ for each i. 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 linear form in the variables X. The size of the ABP is the number of vertices.*

For any s-to-t directed path $\gamma = e_1, e_2, \ldots, e_d$, where e_i is the edge from level $i-1$ to level i, let ℓ_i denote the linear form labeling edge e_i. Let $f_\gamma = \ell_1 \cdot \ell_2 \cdots \ell_d$ be the product of the linear forms in that order. Then the ABP computes the degree d polynomial $f \in \mathbb{F}[X]$ defined as

$$f = \sum_{\gamma \in \mathcal{P}} f_\gamma,$$

where $\mathcal{P}$ is the set of all directed paths from s to t.

For any node u in the ABP A, the polynomial computed at u is the polynomial computed by the ABP, obtained from A, where start vertex is s and sink vertex is u. Set-multilinear ABPs are ABPs where each node in the ABP computes a set-multilinear polynomial with respect to some set $S \subseteq [d]$. Set-multilinear ABPs are a natural model for computing set-multilinear polynomials.

Definition 1.1.9 (Set-multilinear algebraic branching programs)**.** *A* set-multilinear alge-

braic branching program *is a layered directed acyclic graph (DAG) with one in-degree zero vertex s and one out-degree zero vertex t. The vertices of the graph are partitioned into layers* $0, 1, \ldots, d$*, and edges go only from layer i to* $i + 1$ *for each i. The source is the only vertex at level* 0 *and the sink is the only vertex at level d. We can associate an index set* $I_v \subseteq [d]$ *with each node v in the set multilinear ABP, and the polynomial computed at v is set-multilinear w.r.t. the partition* $\sqcup_{i \in I_v} X_i$*. For any edge* (u, v) *in the branching program labeled by a homogeneous linear form* ℓ*, we have* $I_v = I_u \sqcup \{i\}$ *for some* $i \in [d]$*, and* ℓ *is a linear form over variables* X_i*. The size of the ABP is the number of vertices.*

For any s-to-t directed path $\gamma = e_1, e_2, \ldots, e_d$, where e_i is the edge from level $i - 1$ to level i, let ℓ_i denote the linear form labeling edge e_i. Let $f_\gamma = \ell_1 \cdot \ell_2 \cdots \ell_d$ be the product of the linear forms in that order. Then the ABP computes the set-multilinear degree d polynomial:

$$f = \sum_{\gamma \in \mathcal{P}} f_\gamma,$$

where $\mathcal{P}$ is the set of all directed paths from s to t.

Linear circuits over noncommutative rings

Linear circuits are algebraic circuits consisting only of gates, of fanin 2, that compute linear combinations of their inputs. The size of a linear circuit L is defined to be the number of gates in L. This is a natural model for computing linear transformations. The problem is one of computing $A\mathbf{y}$ for an $n \times n$ matrix over $\mathbb{F}$ and $\mathbf{y} = (y_1, y_2, \cdots, y_n)^T$. At the expense of constant factors in size complexity, any arithmetic circuit computing a linear transformations over $\mathbb{C}$ can be turned into a linear circuit (see e.g., [Lok09]). One can see that $A\mathbf{y}$ can be computed by a linear circuit of size $O(n^2)$ and depth $O(\log n)$[1]. But there is no superlinear lower bound is known for this problem. This problem has a rich literature with many interesting developments. Morgenstern [Mor73] showed an $\Omega(n \log n)$ lower

[1] By $\log n$, throughout in this thesis we mean $\log_2 n$.

bound for the Hadamard matrix in the bounded coefficient model when $\mathbb{F} = \mathbb{C}$.

Valiant [Val77] developed matrix rigidity as a means to attack the problem in the case of logarithmic depth linear circuits. In spite of many interesting results and developments, superlinear size lower bounds remain elusive over any field $\mathbb{F}$ even for the special case of log-depth circuits (Lokam's monograph [Lok09] surveys most of the recent results).

Given the limited progress in proving lower bounds for linear circuits over commutative domains, we study the problem proving lower bounds for the case when entries of the matrix A comes from a noncommutative domain.

Now we introduce another kind of circuit called *multiplicative circuits*. Let $(S, \circ)$ be a semigroup, i.e., S is a set closed under the binary operation $\circ$ which is associative. A natural multi-output computational model (called *multiplicative circuit*) is a circuit over $(S, \circ)$.

Definition 1.1.10 (Multiplicative circuits)**.** *The multiplicative circuit C over $(S, \circ)$ is given by a directed acyclic graph with input nodes labeled $x_1, ..., x_n \in S$ of indegree 0 and output nodes $y_1, ..., y_m \in S$ of outdegree 0. The internal nodes of the circuit C is labeled by semigroup operation $\circ$.*

We assume that all gates have fanin 2. The size of the circuit is the number of nodes in it and it computes a function $f : S^n \to S^m$.

This provides a general setting to some well studied problems in circuit complexity. For example:

1. If $S = \mathbb{F}_2$ and $\circ$ is addition in $\mathbb{F}_2$, the problem is one of computing $A\mathbf{y}$ for an $m \times n$ matrix over $\mathbb{F}_2$ and $\mathbf{y} = (y_1, y_2, \cdots, y_n)^T$. The problem of giving an explicit A such that the size of any circuit for it is superlinear is a longstanding open problem. By means of counting arguments, we know that there exist such matrices A [Val77].

2. When $S = \{0, 1\}$ and $\circ$ is the boolean OR, this problem is also well studied and due

to its monotone nature it has explicit lower bounds of circuit size $n^{2-o(1)}$ (e.g. see section 3.4 in [JS13]).

A more restricted form is $S = (\mathbb{N}, +)$ called SUM circuits also well studied e.g. [JS13]. While for monotone settings (OR,SUM circuits) there are nontrivial lower bounds, in the commutative case for S we do not have strong lower bounds results.

Now we define linear circuits over arbitrary ring R.

Definition 1.1.11 (Linear circuit over rings)**.** *Let $(R, +, \cdot)$ be an arbitrary ring (possibly noncommutative). A* linear circuit *over R takes n inputs $y_1, y_2, \ldots, y_n$ labeling the indegree* 0 *nodes of a directed acyclic graph. The circuit has m output nodes (which have outdegree* 0*). Each edge of the graph is labeled by some element of the ring R. The indegree of each non-input node is two. Each node of the circuit computes a linear form $\sum_{i=1}^{n} \alpha_i y_i$ for $\alpha_i \in R$ as follows: the input node labeled y_i computes y_i. Suppose g is a node with incoming edges from nodes g_1 and g_2, and the edges (g_1, g) and (g_2, g) are labeled by α and β respectively. If g_1 and g_2 computes the linear forms ℓ_1 and ℓ_2 respectively, then g computes $\alpha\ell_1 + \beta\ell_2$. Thus, for an $m \times n$ matrix A over the ring R, the circuit computes $A\mathbf{y}$ at the m output gates.*

When R is a field we get the well-studied linear circuits model [Mor73, Val77, Lok09]. However, no explicit superlinear size lower bounds are known for this model over fields, except for some special cases like the bounded coefficient model [Mor73] or in the cancellation free case [BF13].

Depth of a linear circuit C is defined to be length of the longest leaf to root path in the directed acyclic graph of the circuit C. We now define depth 2 linear circuits.

Definition 1.1.12 (Depth-2 linear circuits)**.** *These are linear circuits of depth* 2*, where each addition gate can have unbounded fanin. More precisely, if g is an addition gate with inputs from $g_1, g_2, \ldots, g_t$ then the gate g computes $\sum_{i=1}^{t} \alpha_i g_i$, where each edge (g_i, g) is labeled by $\alpha_i \in \mathbb{F}$ such that $\alpha_i, 1 \leq i \leq t$.*

We again consider the problem of computing $A\mathbf{y}$ for $A \in \mathbb{F}^{n\times n}$. The goal is to lower bound the number of wires in the linear circuit. This problem is also well studied for linear circuits over fields and only an explicit $\Omega(n \log^2 n/ \log\log n)$ lower bound is known for it [Lok09, Pud94], although for random matrices the lower bound is $\Omega(n^2/\log n)$.

1.2 Results and organization of the thesis

Now, we describe the main results in each part of the thesis in detail. The technical contents of the thesis are organized in the following three parts.

1.2.1 Lower bounds results for set-multilinear arithmetic computations

We study depth reduction and lower bound questions for set-multilinear arithmetic computational models and show the following results.

1. For depth reduction, we observe that commutative set-multilinear arithmetic circuit can be efficiently depth reduced. Formally, we prove the following theorem in Chapter 2.

   Theorem 1.2.1 ([AR])**.** *Let Φ be a set-multilinear arithmetic circuit of size s and degree d over the field $\mathbb{F}$ and over the variable set X, partitioned as $X = X_1 \sqcup ... \sqcup X_d$, computing a polynomial $f \in \mathbb{F}[X]$. Then we can efficiently compute from Φ a set-multilinear arithmetic circuit Ψ, with multiplication gates of fanin 2 and unbounded fanin + gates, which is of size $O(s^3)$ and depth $O(\log d)$ computing the polynomial f.*

   The proof of this theorem is based on similar depth reduction theorems in [VSBR83, RY08]. It follows that polynomial size set-multilinear circuits have quasi-polynomial

size set-multilinear branching programs. Note that any non-commutative circuit C computing a non-commutative polynomial P can be efficiently converted into a set-multilinear circuit $\tilde{C}$ computing a corresponding set-multilinearized polynomial $\tilde{P}$ (i.e., each monomial $m = \prod_i x_{j_i}$ in the polynomial P with coefficient α_m appears in the polynomial $\tilde{P}$ as $\tilde{m} = \prod_i x_{i,j_i}$ with the same coefficient). Thus, proving strong enough lower bounds for set-multilinear formulas or set-multilinear ABPs implies super-polynomial lower bound for non-commutative circuits by the depth reduction result discussed above. This is one motivation for studying set-multilinear computational models.

2. For lower bound questions, we study restricted set-multilinear ABPs. These results are presented in Chapter 2.

   The *type width* of the ABP at layer k is the number $\mathrm{tw}(k)$ of *different* types labeling nodes at layer k of the ABP. The notion of type width is motivated by the fact that read-once oblivious ABPs (ROABPs defined in [FS13]) have type width 1 at each layer. It is well-known that Nisan's rank argument [Nis91] (originally used for lower bounding noncommutative ABP size) also yields exponential lower bounds for any ROABP computing PER_n. In particular, it implies an exponential lower bound for set-multilinear ABPs of type-width 1. This suggests that a natural first step to showing lower bounds for general set-multilinear ABPs/circuits is to first deal with set-multilinear ABPs with restrictions on its type width.

   A set-multilinear ABP computing a set-multilinear polynomial f of degree d is called k-narrow for $1 \leqslant k \leqslant d$, if $\min\{k \times \mathrm{tw}(k), (d-k) \times \mathrm{tw}(d-k)\} \leqslant d/2$. For example, ROABPs are a subclass of set-multilinear ABPs that are k-narrow for every k. As another example, we note that the sum of ℓ ROABPs is $\frac{d}{2\ell}$-narrow. We prove the following theorem.

   Theorem 1.2.2 ([AR]). *Any k-narrow set-multilinear ABP computing the permanent polynomial PER_n(or determinant DET_n) requires size 2^k.*

As a consequence, we show a lower bound for sum of ℓ read-once oblivious ABPs(ROABPs) computing PER_n.

Corollary 1.2.3. *Let $P_i, 1 \leq i \leq \ell$, be ROABPs such that $\sum_{i=1}^{\ell} P_i$ is the permanent polynomial PER_n (or the determinant polynomial DET_n). Then at least one of the P_i is of size $2^{\Omega(n/\ell)}$.*

In contrast, for the polynomial identity testing problem for sum of ROABPs, where one asks whether polynomial computed by a sum of c ROABPs is zero or not, we know *non-trivial* algorithms (both white and black box algorithms) only when $c = O(1)$ [GKST15].

3. We also study restrictions on the parse trees of a set-multilinear arithmetic circuit. These results are presented in Chapter 3. For an arithmetic circuit C, a *parse tree* for a monomial m is a multiplicative subcircuit of C rooted at the output gate defined by the following process starting from the output gate: at addition(+) gate retain exactly one of its input gates and at the multiplication gate ($\times$) retain both of its input gates. Retain all inputs that are reached by this process. The resulting subcircuit is multiplicative and computes the monomial m (with some coefficient). Let C be a set-multilinear circuit computing $f \in \mathbb{F}[X]$ for variable partition $X = \sqcup_{i=1}^{d} X_i$. A parse tree T for a monomial is, in fact, a *binary tree* with leaves labeled by variables (ignoring the leaves labeled by constants) and internal nodes labeled by gate names (the $\times$ gates of C occurring in the parse tree). By set-multilinearity, in each parse tree there is exactly one variable from each subset X_i, and each variable occurs at most once in a parse tree. With each parse tree T we can associate its *parse tree type* $\hat{T}$ which is a binary tree with d leaves. Each node v of T is labeled by an index set $I_v \subseteq [d]$: The root is labeled by $[d]$, each leaf is labeled by a distinct singleton set $[i], 1 \leq i \leq d$, and if v has children v_1 and v_2 in the tree then $I_v = I_{v_1} \sqcup I_{v_2}$.

   Thus, given a set-multilinear circuit C we can consider: (a) the set of parse tree types of the entire circuit C, and (b) the set of parse tree types of a given monomial.

For set-multilinear ABPs computing a degree d polynomial, parse trees of monomials are just simple paths of length d, and the corresponding parse tree types are also simple paths of length d. Furthermore, every ROABP have a unique parse tree type. As exponential lower bounds for ROABPs computing the permanent are known [Nis91] using the partial derivative method, a natural question is whether we can obtain lower bounds when more than one parse tree type is allowed in the set-multilinear circuit. We consider (1) Set-multilinear circuits with few parse tree types and, (2) *Unambiguous* set-multilinear circuits: i.e. circuits in which each monomial has a unique parse tree type (but the number of different parse tree types in the circuit is unbounded) and prove the following size lower bounds.

Theorem 1.2.4 ([AR])**.**

- *Let C be a set-multilinear circuit computing the permanent polynomial* PER_n *(or determinant* DET_n*) such that C has at most r distinct parse tree types. Then the size of C is* $\Omega(2^{\frac{n}{r \log n}})$.
- *Let C be an unambiguous set-multilinear circuit with variable partition $X = \sqcup_{i=1}^{n} X_i$, where $X_i = \{X_{ij} \mid 1 \leq j \leq n\}$, such that C computes the permanent polynomial* PER_n *(or determinant* DET_n*). Then C is of size* $2^{\Omega(n)}$.

4. Finally we study interval set-multilinear ABPs and these results are presented in Chapter 3. We first observe that a τ-interval set-multilinear circuit computing a set-multilinear polynomial in $\mathbb{F}[X]$, $X = \sqcup_{i=1}^{d} X_i$, is essentially like a noncommutative circuit computing a noncommutative polynomial over the variables X, whose monomials can be considered as words of the form $x_{i_1} x_{i_2} \ldots x_{i_d}$, where $x_{i_j} \in X_{\tau(j)}$ for $1 \leq j \leq d$. In [HWY10a], Hrubes et al have related the well-known sum-of-squares (in short, SOS) conjecture (also see [Sha00]) to lower bounds for *noncommutative arithmetic circuits*. Our results are based on their work. We prove the following theorem.

Theorem 1.2.5 ([AR]). *There is a set-multilinear polynomial $f \in \mathbb{F}[X]$, where $X = \sqcup_{i=1}^{D} X_i$ and $X_i = \{x_{0,i}, x_{1,i}\}, 1 \leq i \leq D$ such that f has an $O(D^2)$ size monotone set-multilinear ABP and, assuming the SOS conjecture, for any $\tau \in S_D$ any τ-interval multilinear circuit computing f has size $2^{\Omega(D)}$.*

A new aspect is that the polynomial we construct is only partially explicit. We use the probabilistic method to pick certain parameters that define the polynomial.

1.2.2 Noncommutative Valiant's classes: structure and complete problems

For the non-commutative arithmetic circuits model, we study noncommutative analogues of arithmetic complexity classes VP and VNP, called VP_{nc} and VNP_{nc} respectively. In [HWY10b], it is shown that PER_n is VNP_{nc}-complete w.r.t projections (this is the p-projection reducibility defined by Valiant [Val79a], which allows variables or scalars to be substituted for variables). We study the structure of the classes VP_{nc} and VNP_{nc} and its connections to formal language classes.

1. We prove that the Dyck polynomials (defined from the Dyck languages of formal language theory) are complete for VP_{nc} w.r.t $\leqslant_{abp}$ reductions. The result can be seen as an arithmetized version of the Chomsky-Schützenberger theorem [CS63] showing that the Dyck languages are the hardest CFLs. We note here that $\leqslant_{abp}$ reducibility is a generalization of the standard projection reducibility wherein instead of substitution by variables and scalars we allow substitutions by matrices (whose entries are variables/scalars).

   We define the polynomial $D_{i,n}$ over i different types of brackets $X_i = \{(_1,)_1, ..., (_i,)_i\}$ to be sum of all strings in X_i^{2n} which are well-balanced. The $D_{i,n}$ are Dyck polynomials of degree $2n$ over i different types of brackets. The corresponding p-family is denoted $D_i = (D_{i,n})$. We prove the following theorem in Chapter 4.

Theorem 1.2.6 ([AJR16]). *The p-family $D_k = \{D_{k,d}\}_{d\geqslant 0}$ is VP_{nc}-complete under $\leqslant_{abp}$ reductions for $k \geq 2$.*

The $\leqslant_{abp}$ reducibility is implicitly used in the works of [AJS09, AS10], where in [AJS09] it is used for polynomial identity testing of noncommutative ABPs and in [AS10] it is shown that the noncommutative determinant polynomial cannot have polynomial-size noncommutative circuits unless the noncommutative permanent has such circuits. Essentially the result shown is that PER_n is $\leqslant_{abp}$ reducible to the noncommutative determinant.

2. Let the class VSKEW_{nc} consists of p-families $f = (f_n)$ such that each f_n has a noncommutative *skew* arithmetic circuit of size bounded by n^b for some $b > 0$ depending on f. Let $PAL_n = \sum_{w\in\{x_0,x_1\}^n} w.w^R$. We prove the following theorem, in Chapter 4, by adapting the proof of the Chomsky-Schützenberger theorem.

   Theorem 1.2.7 ([AJR16]). *The p-family* $\mathrm{PAL} = (\mathrm{PAL_n})$ *is* VSKEW_{nc}*-complete under $\leqslant_{abp}$ reductions.*

3. We prove a transfer theorem, in Chapter 5, which essentially shows that if f is a VNP_{nc}-complete p-family under projections then an appropriately defined commutative version $f^{(c)}$ of f is complete under projections for the commutative VNP class.

4. Hrubes et al [HWY10a] have shown, assuming the sum-of-squares conjecture, that the p-family $ID = (ID_n)$, where $ID_n = \sum_{w\in\{x_0,x_1\}^n} w.w$ is not in VP_{nc}. Based on ID, we define a p-family ID^* and show in Chapter 5, assuming $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$, that ID^* is neither in VP_{nc} nor VNP_{nc}-complete.

   This is analogous to Ladner's well-known theorem [Lad75]. We note here that Bürgisser [Bür99] has proven an analogue of Ladner's theorem for commutative Valiant classes VP and VNP. That result requires an additional assumption about counting classes in the boolean setting. It also turns out that under $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$

we have an infinite hierarchy w.r.t $\leqslant_{proj}$ reductions between VP_{nc} and VNP_{nc} and also incomparable p-families. To the best of our knowledge, obtaining an infinite hierarchy under the assumption that $\mathsf{VP} \neq \mathsf{VNP}$, is open in the commutative case.

5. We also consider other hypotheses, like VP $\neq$ VNP and SOS conjecture [HWY10a], and show that each one gives rise to VNP_{nc}-intermediate problems under various reductions between polynomial families, like $\leqslant_{linproj}$-reductions where each variable is substituted by a affine linear form. Based on these intermediate polynomial families, we construct strict infinite hierarchy of polynomial families between VP_{nc} and VNP_{nc}. These results are presented in Chapter 5.

6. Within VP_{nc} we obtain a proper hierarchy w.r.t $\leqslant_{abp}$-reductions corresponding to the Dyck polynomials of bounded nesting depth. This roughly corresponds to the noncommutative VNC hierarchy within VP_{nc}. These results are presented in Chapter 5.

1.2.3 Lower bounds for multiplicative and linear circuits over noncommutative domains

We study multiplicative and linear circuits over various noncommutative domains in Chapter 6 and our results are the following:

1. Let $(S, \circ)$ be a semigroup, i.e., S is a set closed under the binary operation $\circ$ which is associative. A natural multi-output computational model is a circuit over $(S, \circ)$. The circuit is given by a directed acyclic graph with input nodes labeled $x_1, ..., x_n$ of indegree 0 and output nodes $y_1, ..., y_m$ of outdegree 0. The gates of the circuit all compute the monoid product. We assume that all gates have fanin 2. The size of the circuit is the number of nodes in it and it computes a function $f : S^n \to S^m$. This provides a general setting to some well studied problems in circuit complexity.

For example, if $S = \mathbb{F}_2$ and $\circ$ is addition in $\mathbb{F}_2$, the problem is one of computing $A\mathbf{x}$ for an $m \times n$ matrix over $\mathbb{F}_2$. The problem of giving an explicit A such that the size of any circuit for it is superlinear is a longstanding open problem. We study the case when the entries of A comes from a noncommutative domain and prove strong lower bounds for various semigroups.

We study the case when $(S, \circ)$ is noncommutative and manage to prove strong lower bounds in some cases. An interesting aspect is that the number of inputs can be restricted to just two: x_0, x_1. The explicit functions y_i, $1 \leqslant i \leqslant m$ are defined as words $y_i = y_{i1}y_{i2}...y_{in}$ where $y_{ij} \in \{x_0, x_1\}$ and $\{y_1, y_2, ..., y_m\}$ are explicitly defined. We show that any circuit $C : \{x_0, x_1\} \to \{y_1, y_2, ..., y_m\}$ is of size $\Omega(\frac{mn}{\log^2 n})$ in the following four settings:

(a) When $(S, \circ)$ is the free monoid X^* for X such that $|X| \geqslant 2$.

(b) When $(S, \circ)$ is the finite matrix semigroup over the boolean ring and matrices are of dimension $n^c \times n^c$ for some constant $c > 0$.

(c) When $(S, \circ)$ is the free group G_X generated by $X = \{x_1, x_2, x_1^{-1}, x_2^{-1}\}$.

(d) When $(S, \circ)$ is the permutation group where $S = S_N$ for $N = n^d$ for some constant $d > 0$.

2. We study a generalization of the linear circuits model, where we allow the coefficients come from *noncommutative rings*. In principle, we can expect lower bounds could be easier to prove in this model. The circuits are more constrained when coefficients come from a noncommutative ring as fewer cancellations can take place. However, we succeed in showing only some limited lower bounds.

   When the coefficients come from a *noncommutative* ring R, we prove lower bounds for certain restricted linear circuits. Suppose the coefficient ring is $R = \mathbb{F}\langle x_0, x_1\rangle$ consisting of polynomials over the field $\mathbb{F}$ in noncommuting variables x_0 and x_1. Let $A \in \mathbb{F}^{n\times n}\langle x_0, x_1\rangle$ where x_0, x_1 are noncommuting variables and $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T$

is a column vector of input variables. The first restriction we consider are *homogeneous* linear circuits over the ring $\mathbb{F}\langle x_0, x_1\rangle$ for computing $A\mathbf{y}$. The restriction is that for every gate g in the circuit, if g has its two incoming edges from nodes g_1 and g_2, then the edges (g_1, g) and (g_2, g) are labeled by α and β respectively, where $\alpha, \beta \in \mathbb{F}\langle x_0, x_1\rangle$ are restricted to be *homogeneous polynomials* of same degree in the variables x_0 and x_1. It follows, as a consequence of this restriction, that each gate g of the circuit computes a linear form $\sum_{i=1}^{n} \alpha_i y_i$, where the $\alpha_i \in \mathbb{F}\langle x_0, x_1\rangle$ are all homogeneous polynomials of the same degree. We prove the following theorem.

Theorem 1.2.8 ([ARS14]). *There exists an explicit matrix $A \in \mathbb{F}^{n\times n}\langle x_0, x_1\rangle$, computing $A\mathbf{y}$ by any homogeneous linear circuit C over the coefficient ring $\mathbb{F}\langle x_0, x_1\rangle$ requires either size $\omega(n)$ or depth $\omega(\log n)$.*

We prove this by suitably generalizing Valiant's matrix rigidity method [Val77].

3. We next consider homogeneous depth 2 linear circuits. These are linear circuits of depth 2, where each addition gate can have unbounded fanin. More precisely, if g is an addition gate with inputs from $g_1, g_2, \ldots, g_t$ then the gate g computes $\sum_{i=1}^{t} \alpha_i g_i$, where each edge (g_i, g) is labeled by $\alpha_i \in \mathbb{F}\langle x_0, x_1\rangle$ such that $\alpha_i, 1 \leq i \leq t$ are all homogeneous polynomials of the same degree. We again consider the problem of computing $A\mathbf{y}$ for $A \in \mathbb{F}^{n\times n}\langle x_0, x_1\rangle$. The goal is to lower bound the number of wires in the linear circuit. This problem is also well studied for linear circuits over fields and only an explicit $\Omega(n \log^2 n/\log\log n)$ lower bound is known for it [Lok09, Pud94], although for random matrices the lower bound is $\Omega(n^2/\log n)$.

   Theorem 1.2.9 ([ARS14]). *There exists an explicit matrix $A \in \mathbb{F}^{n\times n}\langle x_0, x_1\rangle$, computing $A\mathbf{y}$ by a depth 2 homogeneous linear circuit (with unbounded fanin) requires $\Omega(\frac{n^2}{\log n})$ wires.*

   In contrast, for depth 2 linear circuits over fields only an explicit $\Omega(n \log^2 n/\log\log n)$ lower bound is known [Lok09, Pud94], although for random matrices the lower bound is $\Omega(n^2/\log n)$.

Chapter 2

Some lower bound results for set-multilinear arithmetic computations

2.1 Introduction

In this chapter, we present our results on set-multilinear arithmetic computations. We study the structure of set-multilinear arithmetic circuits and set-multilinear branching programs with the aim of showing lower bound results. Superpolynomial lower bounds for the general case is still out of reach. We define some natural restrictions of these models for which we are able to show lower bound results. Some of our results extend existing lower bounds, while others are new and raise open questions. Specifically, our main results are the following:

- We observe that set-multilinear arithmetic circuits can be transformed into shallow set-multilinear circuits efficiently, following the work of Valiant et al., [VSBR83] (also Raz-Yehudayoff [RY08]). As a consequence, polynomial size set-multilinear circuits have quasi-polynomial size set-multilinear branching programs.

- We show that *k-narrow* set-multilinear ABPs computing the Permanent polynomial

PER_n (or determinant DET_n) require $2^{\Omega(k)}$ size. As a consequence, we show that sum of r read-once oblivious ABPs computing PER_n requires size $2^{\Omega(\frac{n}{r})}$.

2.2 Preliminaries

Let $\mathbb{F}$ be a field and $X = X_1 \sqcup X_2 \sqcup \cdots \sqcup X_d$ be a partition of the variable set X.

Definition 2.2.1. *(set-multilinear polynomial) A* set-multilinear polynomial $f \in \mathbb{F}[X]$ *w.r.t. this partition is a homogeneous degree d multilinear polynomial such that every nonzero monomial of f has exactly one variable from X_i, $1 \leq i \leq d$.*

Both the Permanent polynomial PER_n and the Determinant polynomial DET_n are set-multilinear polynomials. The variable set is $X = \{x_{ij}\}_{1 \leq i,j \leq n}$ and the partition can be taken as the row-wise partition of the variable set. I.e. $X_i = \{x_{ij} \mid 1 \leq j \leq n\}$ for $1 \leq i \leq n$.

In this chapter we will consider set-multilinear circuits (see Definition 1.1.7) and set-multilinear branching programs (see Definition 1.1.9) for computing set-multilinear polynomials. The model of set-multilinear branching programs that we consider is more general than related notions of branching programs recently studied in the literature, like the read-once oblivious branching programs (ROABPs) [FS13].

Remark 2.2.1. *Showing a superpolynomial lower bound for set-multilinear circuits and even for set-multilinear ABPs for computing the Permanent polynomial is an open problem. In this chapter we discuss some restricted versions of set-multilinear branching programs and show lower bounds.*

2.3 Summary of results

To begin with we observe that any set-multilinear arithmetic circuit of size s can be efficiently transformed into an $O(\log s)$ depth set-multilinear circuit with unbounded fanin +

gates and fanin 2 $\times$ gates of size polynomial in s. The proof exactly follows the depth-reduction results of [VSBR83, RY08] for general commutative circuits. The only extra point is our observation that set-multilinearity can be also preserved in the process. As a result, size s set-multilinear circuits have $s^{O(\log s)}$ size set-multilinear branching programs. This is stated and proved in Section 2.4. Our proof is based on the result in [RY08], about syntactic multilinear circuits. The proof in [RY08] follows general arithmetic circuits depth reduction result of Valiant et al., [VSBR83].

In Section 2.5 we consider *narrow* set-multilinear branching programs: The *typewidth* of a set-multilinear ABP at layer k is the number of distinct index types of the ABP in its k^{th} layer. Read-once oblivious ABPs (ROABPs) considered in [FS13] are the set-multilinear ABPs of typewidth 1 in each layer. Thus, it is natural to investigate lower bounds for set-multilinear ABPs of restricted typewidth in order to extend known lower bounds [Nis91] for ROABPs. We say that a set-multilinear ABP computing a degree d polynomial is k-*narrow* if either in layer $d-k$ or layer k the typewidth is bounded by $\frac{d}{2k}$. We show that k-narrow set-multilinear ABPs for PER_n require $2^{\Omega(k)}$ size. The proof follows standard techniques based on the rank of the partial derivative matrix. As a consequence, it follows that the sum of r ROABPs computing PER_n require size $2^{\Omega(\frac{n}{r})}$. Similarly, it follows that a set-multilinear ABP for PER_n, whose typewidth in all layers is bounded by $n^{1-\epsilon}$, requires size $2^{n^{\epsilon}}$. While these observations generalize the lower bound for ROABPs, it appears difficult to prove superpolynomial lower bounds for ABPs of larger typewidth (even $O(n)$). A lower bound result for general set-multilinear ABPs would imply, for instance, that the Permanent requires superpolynomial size noncommutative arithmetic circuits, which is an open problem for over two decades. We present these observations in Section 2.5.

Our lower bound proofs are applications of the well-known partial derivative method.

2.4 Depth Reduction of Set-Multilinear Circuits

We follow the standard method of depth reduction of commutative arithmetic circuits [VSBR83], and use the exposition from Shpilka and Yehudayoff's survey article [SY10]. The general depth reduction was adapted to syntactic multilinear circuits by Raz and Yehudayoff [RY08]. Our additional observation essentially is that the depth reduction procedure can be carried out while preserving set-multilinearity as well.

Given a commutative set-multilinear circuit C of size s computing a set-multilinear polynomial f of degree d in the input variable $X = X_1 \sqcup ... \sqcup X_d$, we show that there is another circuit C' of size $poly(s)$ and depth $O(\log d \log s)$ computing f.

The proof of the following theorem follows the same steps as in the depth reduction results in [VSBR83, RY08]. The only additional point we observe is that set-multilinearity is preserved by the construction.

Theorem 2.4.1. *Let Φ be a set-multilinear arithmetic circuit of size s and degree d over the field $\mathbb{F}$ and over the variable set X, partitioned as $X = X_1 \sqcup ... \sqcup X_d$, computing a polynomial $f \in \mathbb{F}[X]$. Then we can efficiently compute from Φ a set-multilinear arithmetic circuit Ψ, with multiplication gates of fanin 2 and unbounded fanin + gates, which is of size $O(s^3)$ and depth $O(\log d)$ computing the polynomial f.*

Proof. By definition, Φ is a homogeneous arithmetic circuit. We assume that Φ is non-redundant (i.e , for all gates v in Φ the polynomial f_v computed at v is nonzero). Since Φ is set-multilinear, at each gate v in Φ there is an associated index set $I_v \subseteq [d]$ such that the polynomial f_v is set-multilinear of degree $|I_v|$ over the variable set X_{I_v}, where

$$X_{I_v} = \sqcup_{i \in I_v} X_i.$$

We denote the subcircuit rooted at the gate v by Φ_v.

Partial Derivative of f_v by a gate w

Let v, w be any two gates in circuit Φ. Following the exposition in [SY10], let $\Phi_{w=y}$ denote the circuit obtained by removing any incoming edges at w and labeling w with a new input variable y and $f_{v,w}$ denote the polynomial (in $X \cup \{y\}$) computed at gate v in circuit $\Phi_{w=y}$. Define

$$\partial_w f_v = \partial_y f_{v,w}.$$

Note that $f_{v,w}$ is linear in y. Clearly, if w does not occur in Φ_v then $\partial_w f_v = 0$. If w occurs in Φ_v, since Φ is set-multilinear the polynomial $f_{v,w}$ is linear in y and is of the form

$$f_{v,w} = h_{v,w}y + g_{v,w}.$$

Therefore, $\partial_w f_v = h_{v,w}$. We make the following immediate observations from the set-multilinearity of Φ.

- Either $\partial_w f_v = 0$ or $\partial_w f_v$ is a homogeneous set-multilinear polynomial of degree $deg(v) - deg(w)$ over variable set $X \setminus X_{I_w}$.
- If $deg(v) < 2.deg(w)$ and v is a product gate with children v_1, v_2 such that $deg(v_1) \geqslant deg(v_2)$, then $\partial_w f_v = f_{v_2}.\partial_w f_{v_1}$.

For a positive integer m, let G_m denote the set of product gates t with inputs t_1, t_2 in Φ such that $m < deg(t)$ and $deg(t_1), deg(t_2) \leqslant m$. We observe the following claims (analogous to [SY10]) which are easily proved.

Claim 2.4.1. *Let Φ be a set-multilinear nonredundant arithmetic circuit over variable set $X = \sqcup_{i=1}^{d} X_i$. Let v be a gate in Φ such that $m < deg(v) \leqslant 2m$ for a positive integer m. Then $f_v = \sum_{t \in G_m} f_t.\partial_t f_v$.*

Claim 2.4.2. *Let Φ be a set-multilinear non-redundant arithmetic circuit over the field $\mathbb{F}$ and over the set of variables X. Let v and w be gates in Φ such that $0 < deg(w) \leqslant m <$*

$deg(v) < 2deg(w)$. *Then* $\partial_w f_v = \sum_{t \in G_m} \partial_w f_t . \partial_t f_v$

Construction of Ψ:

We now explain the construction of the depth-reduced circuit Ψ. The construction is done in stages. Suppose upto Stage i we have computed, for $1 \leq j \leq i$ the following:

- All polynomials f_v for gates v such that $2^{j-1} < deg(v) \leqslant 2^j$.
- All partial derivatives of the form $\partial_w f_v$ for gates v and w such that $2^{j-1} < deg(v) - deg(w) \leqslant 2^j$ and $deg(v) < 2deg(w)$.
- Furthermore, inductively assume that the circuit computed so far is set-multilinear of $O(i)$ depth, such that all product gates are fanin 2, sum gates are of unbounded fanin.

We now describe Stage $i + 1$ where we will compute all f_v for gates v such that $2^i < deg(v) \leqslant 2^{i+1}$ and also all partial derivatives of the form $\partial_w f_v$ for gates v and w such that $2^i < deg(v) - deg(w) \leqslant 2^{i+1}$ and $deg(v) < 2deg(w)$. Furthermore, we will do this by adding a depth of $O(1)$ to the circuit and $\mathsf{poly}(d, s)$ many new gates maintaining set-multilinearity.

Stage i+1: We describe the construction at this stage in two parts:

Computing f_v

Let v be a gate in Φ such that $2^i < deg(v) \leqslant 2^{i+1}$ and let $m = 2^i$. By Claim 2.4.1, we have

$$f_v = \sum_{t \in T} f_t \partial_t f_v = \sum_{t \in T} f_{t_1} f_{t_2} \partial_t f_v,$$

where T is the set of gates $t \in G_m$, with children t_1 and t_2 such that t is in Φ_v. Note that if t is not in Φ_v, then $\partial_t f_v = 0$. Let $t \in T$ be a gate with inputs t_1 and t_2. Thus.

$m < deg(t) \leqslant 2m$, $deg(t_1) \leqslant m$, $deg(t_2) \leqslant m$. Hence $deg(v) - deg(t) \leqslant 2^{i+1} - 2^i = 2^i$ and $deg(v) \leqslant 2^{i+1} < 2.deg(t)$. Therefore, f_{t_1}, f_{t_2} and $\partial_t f_v$ are already computed. Thus, in order to compute f_v we need $O(s)$ many $\times$ gates and $O(1)$ many $+$ gates. Overall, with $O(s^2)$ many new gates and $O(1)$ increase in depth we can compute all f_v such that $2^i < deg(v) \leqslant 2^{i+1}$. Furthermore, we note that f_{t_1}, f_{t_2} and $\partial_t f_v$ are all set-multilinear polynomials with disjoint index sets, and the union of their index sets is I_v for each $t \in T$. Thus, the new gates introduced all preserve set-multilinearity.

Computing $\partial_w f_v$

Let v and w be gates in Φ such that $2^i < deg(v) - deg(w) \leqslant 2^{i+1}$ and $deg(v) < 2deg(w)$. Let $m = 2^i + deg(w)$. Thus, $deg(w) \leqslant m < deg(v) < 2deg(w)$. Let T' denote the set of gates in Φ_v that are contained in G_m. Note that $\partial_t f_v = 0$ if $t \notin T'$. Hence by Claim 2.4.2 we can write

$$\partial_w f_v = \sum_{t \in T'} \partial_w f_t \partial_t f_v,$$

For a gate $t \in T'$, we have $deg(t) \leq deg(v) < 2deg(w)$. Suppose t_1 and t_2 are the gates input to t in the circuit Φ, and $\deg(t_1) \geq deg(t_2)$. Then we can write

$$\partial_w f_v = \sum_{t \in T'} f_{t_2} \partial_w f_{t_1} \partial_t f_v.$$

We claim that f_{t_2}, $\partial_w f_{t_1}$, and $\partial_t f_v$ are already computed.

- Since $deg(v) \leqslant 2^{i+1} + deg(w) \leq 2^{i+1} + deg(t_1) = 2^{i+1} + deg(t) - deg(t_2)$, we have $deg(t_2) \leqslant 2^{i+1} + deg(t) - deg(v) \leqslant 2^{i+1}$. Hence f_{t_2} is already computed (in first part of stage $i + 1$).
- Since $deg(t_1) - deg(w) \leq 2^i$, the polynomial $\partial_w f_{t_1}$ is already computed in an earlier stage.

- Since $deg(t) > m$, we have $deg(v) - deg(t) \leq deg(v) - m \leq 2^{i+1} - 2^i = 2^i$.
- Thus, since $deg(v) \leqslant 2^{i+1} + deg(w) \leqslant 2(2^i + deg(w)) < 2deg(t)$, the polynomial $\partial_t f_v$ is already computed in an earlier stage.

As before, for each such pair of gates w and v, we can compute $\partial_w f_v$ with $O(s)$ new gates (using the polynomials already computed in previous stages), and this increases the circuit depth by $O(1)$. Since we consider pairs of gates (w, v) such that $2^i < deg(v) - deg(w) \leqslant 2^{i+1}$ at the i^{th} stage, the total number of pairs (w, v) considered over all the stages is bounded by s^2. Hence the number of new gates added over all the stages is $O(s^3)$. Thus the size of the depth-reduced circuit obtained is $O(s^3)$ and its depth is $O(\log d)$. Furthermore, the new gates included clearly also have the set-multilinearity property. This completes the proof of the theorem. □

Remark 2.4.1. *We note that the size of the depth-reduced circuit for general multilinear circuits [RY08] turns out to be $O(s^3d^6)$ because of the homogenization step, which is not required for set-multilinear circuits.*

Set-Multilinear Circuits to ABPs

Theorem 2.4.2. *Given a set-multilinear arithmetic circuit of size s and degree d over the field $\mathbb{F}$ and over the variable set $X = \sqcup_{i=1}^{d} X_i$, computing $f \in \mathbb{F}[X]$, we can transform it, in time $s^{O(\log d)}$, into a set-multilinear ABP of size $(sd)^{O(\log d)}$ that computes f.*

Proof. The proof of this theorem is fairly straightforward consequence of the depth reduction result (Theorem 2.4.1 in the previous section). By Theorem 2.4.1 we can assume to have computed a set-multilinear circuit Ψ of size $O(s^3)$ and depth $O(\log d)$ for computing f. By a standard bottom-up procedure we can transform the circuit Ψ into a formula F: at a gate g with fanout t we make t copies of the circuit computed at g. The resulting circuit is a formula of size $(sd)^{O(\log d)}$, because at every level of the circuit there is a factor of s

increase in the size (as s bounds the fanout of all gates). The formula F thus constructed from C is clearly also homogeneous, set-multilinear, and of depth $O(\log d)$. The formula F is also semi-unbounded: the product gates are fanin 2 and plus gates have unbounded fanin.

Formula F could have subformulas that computes scalars. We perform the following transformation to remove such subformulas. Suppose g is a gate in F that evaluates to a scalar α, and g is input to gate g' in F. Then we remove the subformula at g and label the outgoing edge of g' by α. The transformed set-multilinear formula F' has only variables at the input gates and scalars labeling edges of the formula interpreted as follows: suppose g is a gate with g' as an input gate such that the edge from g' to g is labeled by scalar α. Then the contribution of gate g' to g is the polynomial αP, where P is the polynomial computed at g'.

Finally, we can apply a standard transformation (for e.g., see [Nis91]) to convert the formula F' into a homogeneous algebraic branching program (ABP). It is a bottom-up construction of the ABP: at a + gate we can do a "parallel composition" of the input ABPs to simulate the + gate. At a $\times$ gate it is a sequential composition of the two ABPs. Since the formula F is set-multilinear, the resulting ABP is also easily seen to be a set-multilinear ABP. □

2.5 A Lower Bound Result for Set-Multilinear ABPs

As we have shown in Theorem 2.4.1, we can simulate set-multilinear circuits of size s and degree d using set-multilinear ABPs of size $s^{O(\log d)}$. Thus, proving even a lower bound of $n^{\omega(\log n)}$ for set-multilinear ABPs computing the $n \times n$ Permanent polynomial PER_n would imply superpolynomial lower bounds for general set-multilinear circuits computing PER_n which is a long-standing open problem.

However, in this section we show a lower bound result for set-multilinear ABPs with

restricted *type width*, a notion that we now formally introduce.

Let P be a set-multilinear ABP computing a polynomial $f \in \mathbb{F}[X]$ of degree d with variable set $X = \sqcup_{i=1}^{d} X_i$. By definition, the ABP P is a layered directed acyclic graph with layers numbered $0, 1, \ldots, d$. Each node v in layer k of the ABP is labeled by an index set $I_v \subseteq [d]$, and a degree k set-multilinear polynomial f_v over variables $\sqcup_{i \in I_v} X_i$ is computed at v by the ABP. We refer to I_v as the *type* of node v. The *type width* of the ABP at layer k is the number $\mathrm{tw}(k)$ of *different* types labeling nodes at layer k of the ABP.

The notion of type width is motivated by the fact that read-once oblivious ABPs (ROABPs defined in [FS13]) have type width 1 (as noted in the following proposition).

Proposition 2.5.1. *Suppose P is a set-multilinear ABP computing a polynomial $f \in \mathbb{F}[X]$ of degree d with variable set $X = \sqcup_{i=1}^{d} X_i$ such that the type-width of P is 1 at each layer. Then P is in fact an ROABP which is defined by a suitable permutation on the index set $[d]$.*

Proof. As each layer of P has type width one, the list of type $I_0 = \emptyset \subset I_1 \subset \cdots \subset I_d$ gives an ordering of the index set, where the i^{th} index in the ordering is $I_i \setminus I_{i-1}$. W.r.t. this ordering clearly P is an ROABP. □

It is well-known that Nisan's rank argument [Nis91] (originally used for lower bounding noncommutative ABP size) also yields exponential lower bounds for any ROABP computing PER_n. In particular, it implies an exponential lower bound for set-multilinear ABPs of type-width 1. This suggests that a natural first step to showing lower bounds for general set-multilinear ABPs/circuits is to first deal with set-multilinear ABPs with restrictions on its type width.

2.5.1 Lower bounds for narrow set-multilinear ABPs

Definition 2.5.2. *A set-multilinear ABP computing a degree d polynomial in $\mathbb{F}[X]$ such that $X = \sqcup_{i=1}^{d} X_i$ is said to be* k-narrow*, for $1 \leq k \leq d$, if*

$$\min\{k.\mathrm{tw}(k), (d-k).\mathrm{tw}(k)\} \leq d/2.$$

For example, ROABPs are a subclass of set-multilinear ABPs that are k-narrow for every k. As another example, we note that the sum of ℓ ROABPs is $\frac{d}{2\ell}$-narrow.

Theorem 2.5.3. *Any k-narrow set-multilinear ABP computing the permanent polynomial PER_n requires size 2^k.*[1]

Proof. Let P be a k-narrow set-multilinear ABP computing PER_n, and suppose $k.\mathrm{tw}(k) \leq n/2$ (we note that if $(n-k).\mathrm{tw}(k) \leq n/2$ the proof proceeds analogously by reversing the roles of the source and sink nodes of the ABP). In the proof we shall assume that n is even in order to avoid floors and ceils.

Let V_k denote the set of nodes in the k^{th} layer of P. For each node $v \in V_k$ let I_v denote its index set. As $|I_v| = k$ and P is k-narrow, we have

$$|\bigcup_{v \in V_k} I_v| \leq k.\mathrm{tw}(k) \leq n/2.$$

We choose and fix a size $n/2$ subset A of $[n]$ such that $\bigcup_{v \in V_k} I_v \subseteq A$.

For any set-multilinear polynomial $f \in \mathbb{F}[X]$ w.r.t. the partition $X = \sqcup_{i=1}^{n} X_i$, where $X_i = \{X_{ij} \mid 1 \leq j \leq n\}$, we can define the matrix M_f whose rows are indexed by degree $n/2$ set-multilinear monomials m and columns by degree $n/2$ set-multilinear monomials m' such that m is a monomial in variables $\sqcup_{i \in A} X_i$ and m' in variables $\sqcup_{i \notin A} X_i$ and

[1] The same lower bound proof will work for the Determinant polynomial DET_n.

$$M_f(m, m') = f(mm'), \tag{2.1}$$

where $f(mm')$ denote the coefficient of monomial mm' in polynomial f.

For each node $v \in V_k$ let f_v and g_v denote the polynomials computed by P between start node s and sink node v, and start node v and sink node t, respectively. Since P computes PER_n, it follows that

$$M_{\text{PER}_n} = \sum_{v \in V_k} M_{f_v g_v}.$$

Let $A = \{i_1, i_2, \ldots, i_{n/2}\}$ and $\overline{A} = \{j_1, j_2, \ldots, j_{n/2}\}$ be the indices listed in, say, increasing order. We set some variables of PER_n to zero: We rename the rows and columns of the $n \times n$ matrix X_{ij} as $i_1, j_1, i_2, j_2, \ldots, i_{n/2}, j_{n/2}$. Along the principal diagonal we now have $n/2$ many 2×2 matrices, where the r^{th} such matrix is indexed by i_r, j_r in both rows and columns. We retain only these $4r = 2n$ many variables in PER_n and set all other variables X_{ij} to 0. Let the resulting polynomial in these $2n$ variables be denoted $\hat{\text{PER}}_n$. Notice that $\hat{\text{PER}}_n$ is set-multilinear of degree n with each set $X'_i \subset X_i$ in partition having exactly two variables in it. Furthermore, let $\hat{f}_v$ and $\hat{g}_v$ be polynomials obtained from f_v and g_v by the same substitution for each $v \in V_k$. We clearly have

$$M_{\hat{\text{PER}}_n} = \sum_{v \in V_k} M_{\hat{f}_v \hat{g}_v}.$$

Clearly,

$$rank(M_{\hat{\text{PER}}_n}) \leq \sum_{v \in V_k} rank(M_{\hat{f}_v \hat{g}_v}). \tag{2.2}$$

The following two claims immediately yield the claimed lower bound of 2^k on the size of the ABP P.

Claim 2.5.1. *rank*$(M_{\hat{\text{PER}}_n}) = 2^{n/2}$.

Proof of Claim. The degree $n/2$ monomials labeling the rows of matrix $M_{\hat{\text{PER}}_n}$ are of the form $m = \prod_{t=1}^{r} X_{i_t a_t}$, where $a_t \in \{i_t, j_t\}$. Likewise, the degree $n/2$ monomials labeling the columns of matrix $M_{\hat{\text{PER}}_n}$ are of the form $m' = \prod_{t=1}^{r} X_{j_t a_t}$, where $a_t \in \{i_t, j_t\}$. Clearly, for each such m there is a unique m' such that their product mm' is a nonzero monomial of $\hat{\text{PER}}_n$. Thus, the matrix $M_{\hat{\text{PER}}_n}$ is a permutation matrix and hence it has rank exactly $2^{n/2}$.

Claim 2.5.2. *For each* $v \in V_k$ *we have* *rank*$(M_{\hat{f}_v \hat{g}_v}) \leq 2^{n/2-k}$.

Proof of Claim. It suffices to show that we can write matrix $M_{\hat{f}_v \hat{g}_v}$ as a sum of $2^{n/2-k}$ rank 1 matrices. To this end, let $\hat{m}$ be a set-multilinear monomial of degree $n/2 - k$ whose variables come from $\sqcup_{i \in A \setminus I_v} X_i$ (where we recall that I_v is the index set of node v in the ABP). Let $M^{(\hat{m})}$ denote the $2^{n/2} \times 2^{n/2}$ matrix obtained from $M_{\hat{f}_v \hat{g}_v}$ as follows: for each degree k set-multilinear monomial m over $\sqcup_{i \in I_v} X_i$, the row labeled $m\hat{m}$ of $M_{\hat{f}_v \hat{g}_v}$ is the same for $M^{(\hat{m})}$. All other rows of $M^{(\hat{m})}$ are zero. Clearly,

$$M_{\hat{f}_v \hat{g}_v} = \sum_{\hat{m}} M^{(\hat{m})}.$$

Furthermore, the rank of $M^{\hat{m}}$ is at most 1 because it can be expressed as the product of a $2^{n/2} \times 1$ matrix and a $1 \times 2^{n/2}$ matrix (using the coefficient vectors of the polynomials $\hat{f}_v$ and $\hat{g}_v$).

Clearly, the above claims combined with Equation (2.2) imply that $|V_k| \geq 2^k$ which implies that the size of P is lower bounded by 2^k. □

As a consequence of Theorem 2.5.3 we immediately obtain the following lower bound on the size of a sum of r many ROABPs for computing the permanent.

Corollary 2.5.4.

- *If P is a set-multilinear ABP computing* PER_n *such that the type width of every layer is bounded by r then the size of P is $2^{\Omega(n/r)}$. A special case is the next part.*

- *Let $P_i, 1 \leq i \leq r$, be ROABPs such that $\sum_{i=1}^{r} P_i$ is the permanent polynomial* PER_n *(or the determinant polynomial* DET_n*). Then at least one of the P_i is of size $2^{\Omega(n/r)}$.*

Proof. The first part clearly implies that the type width at layer $\frac{n}{2r}$ is bounded by r and we can apply the above theorem. For the second part, it suffices to observe that the sum of r many ROABPs is an smABP whose type width at each layer is bounded by r. □

Thus, if $r = O(\frac{n}{\log^2 n})$, we get superpolynomial $(n^{\Omega(\log n)})$ lower bound for sum of $O(\frac{n}{\log^2 n})$-many ROABPs computing PER_n or DET_n.

Remark 2.5.1. *We note that there is a polynomial-time (white-box) identity testing algorithm for the sum of a* constant *number of ROABPs [GKST15]. Their black-box PIT is quasi-polynomial time. It remains an interesting problem to prove a superpolynomial lower bound for the sum of* $\mathsf{poly}(n)$ *many (or even $O(n)$ many) ROABPs computing* PER_n*. We also note the recent work by Anderson et al [AFS+16] showing lower bounds for read-k oblivious ABPs.*

2.6 Summary and open problems

We investigated lower bound questions for certain set-multilinear arithmetic circuits and ABPs. By imposing a restriction on the number of set types for set-multilinear ABPs, we could prove nontrivial lower bounds for the Permanent. Some interesting open questions arise from our work: can we show lower bounds for $f(n)$-narrow set-multilinear ABPs for $f(n) = O(n)$?. We believe that for set-multilinear ABPs/circuits the determinant and permanent are equally hard (like for noncommutative circuits [AS10]). Given a set-multilinear ABP for the determinant can we transform it into a set-multilinear ABP for

the permanent by somehow “removing” the signs of all the monomials?

Chapter 3

Lower bounds for some restricted set multilinear circuits

3.1 Introduction

In this chapter also, we continue our study of set-multilinear arithmetic computations. We study the structure of set-multilinear arithmetic circuits and set-multilinear branching programs with the aim of showing lower bound and separation results. We define some natural restrictions of these models for which we are able to show lower bound results. Our main results are the following:

- We show that set-multilinear branching programs are exponentially more powerful than *interval* multilinear circuits (where the index sets for each gate is restricted to be an interval w.r.t. some ordering), assuming the sum-of-squares conjecture. This further underlines the power of set-multilinear branching programs.
- Next, we examine a more semantic restriction. The semantic restriction we consider are restrictions on the number of parse trees of monomials. We show exponential lower bounds for set-multilinear circuits with restrictions on the number of parse

trees of monomials and prove exponential lower bounds results.

3.2 Preliminaries

For variable partition $X = \sqcup_{i=1}^{d} X_i$ let $f \in \mathbb{F}[X]$ be a set-multilinear polynomial.

Definition 3.2.1 (σ-interval multilinear circuit)**.** *For a permutation $\sigma \in S_d$, a* σ-interval multilinear circuit *C for computing f is a special kind of set-multilinear arithmetic circuit: for every gate of the circuit the corresponding index set is a* σ-interval *$\{\sigma(i), \sigma(i+1), \ldots, \sigma(j)\}$, $1 \leq i \leq j \leq d$.*

Definition 3.2.2 (σ-interval multilinear ABP)**.** *Similarly, a* σ-interval multilinear ABP *is a set-multilinear ABP such that the index set associated to every node is some σ-interval.*

Definition 3.2.3 (Parse Tree)**.** *For an arithmetic circuit C, a* parse tree *for a monomial m is a multiplicative subcircuit of C rooted at the output gate defined by the following process starting from the output gate:*

- *At each + gate retain exactly one of its input gates.*
- *At each $\times$ gate retain both its input gates.*
- *Retain all inputs that are reached by this process.*
- *The resulting subcircuit is multiplicative and computes the monomial m (with some coefficient).*

Definition 3.2.4 (Parse Tree Type)**.** *Let C be a set-multilinear circuit computing $f \in \mathbb{F}[X]$ for variable partition $X = \sqcup_{i=1}^{d} X_i$.*

- *A parse tree T for a monomial is, in fact, a* binary tree *with leaves labeled by variables (ignoring the leaves labeled by constants) and internal nodes labeled by*

gate names (the $\times$ gates of C occurring in the parse tree). By set-multilinearity, in each parse tree there is exactly one variable from each subset X_i, and each variable occurs at most once in a parse tree.

- *With each parse tree T we can associate its* parse tree type $\hat{T}$ *which is a binary tree with d leaves. Each node v of T is labeled by an index set $I_v \subseteq [d]$: The root is labeled by $[d]$, each leaf is labeled by a distinct singleton set $[i]$, $1 \leq i \leq d$, and if v has children v_1 and v_2 in the tree then $I_v = I_{v_1} \sqcup I_{v_2}$.*

3.3 Summary of results

In Section 3.4, we show that set-multilinear branching programs are exponentially more powerful than *interval* multilinear circuits (where the index sets for each gate is restricted to be an interval w.r.t. some ordering), assuming the sum-of-square conjecture [HWY10a]. This further underlines the power of general set-multilinear branching programs. A new aspect of the polynomial family which we define is that the homogeneous polynomial family $\{f_n\}_n$ we construct is only partially explicit. We use the probabilistic method to pick certain parameters that define the polynomial. We show that for any *interval* multilinear circuits computing the polynomial family $\{f_n\}_n$, we can always pick a set of indices S such that projecting the polynomial f_n on to these locations results in a polynomial f'_n which is hard to compute assuming the sum-of-square conjecture [HWY10a].

In Section 3.5 we investigate lower bounds for a different generalization of ROABPs based on the structure of parse trees of monomials. We can define parse tree types for a set-multilinear circuit which is a binary tree with index types labeling all its nodes. We note in an ROABP all monomials have the same parse tree type (a single path of length d). Thus, a natural generalization is to consider set-multilinear ABPs (and circuits) with restrictions on parse trees of monomials. It turns out that if a set-multilinear circuit for PER_n has only r many parse tree types *in all*, then we can use the results of Section 2.5

to show a $2^{\Omega(\frac{n}{r \log n})}$ lower bound on its size. Another restriction are set-multilinear circuits such that each monomial has at most one parse tree type (but the total number of parse tree types in the circuit is unbounded). For such circuits too we can prove exponential lower bounds. Our lower bound proofs are applications of well-known partial derivative method.

3.4 Interval multilinear circuits and ABPs

The aim of the present section is to compare the computational power of interval multilinear circuits with general set-multilinear circuits. Clearly, σ-interval multilinear circuits are restricted by the ordering. In essence, σ-interval multilinear circuits are restricted to compute like noncommutative circuits (with respect to the ordering prescribed by σ). This property needs to be exploited to prove the separations. We show the following result:

> Assuming the sum-of-squares conjecture [HWY10a], we show that there are set-multilinear polynomials $f \in \mathbb{R}[X]$ with monotone set-multilinear circuits (even ABPs) of size linear in d, but require $2^{\Omega(d)}$ size σ-interval multilinear circuits for every $\sigma \in S_d$.

A new aspect is that the polynomial we construct is only partially explicit. We use the probabilistic method to pick certain parameters that define the polynomial.

The polynomial construction

Let $X = \sqcup_{i=1}^{2d} X_i$ be the variable set, where

$$X_i = \{x_{0,i}, x_{1,i}\}, 1 \leq i \leq 2d.$$

For every binary string $b \in \{0,1\}^d$ we define the monomials:

$$w_b = \prod_{i=1}^{d} x_{b_i,i} \text{ and } w'_b = \prod_{i=1}^{d} x_{b_i,d+i}.$$

We define the set-multilinear polynomial

$$P = \sum_{\overline{b} \in \{0,1\}^d} w_b w'_b.$$

For any permutation $\sigma \in S_{2d}$ permuting the indices in $[2d]$, we define monomials:

$$\sigma(w_b) = \prod_{i=1}^{d} x_{b_i,\sigma(i)} \text{ and } \sigma(w'_b) = \prod_{i=1}^{d} x_{b_i,\sigma(d+i)},$$

and the corresponding polynomial $\sigma(P)$

$$\sigma(P) = \sum_{\overline{b} \in \{0,1\}^d} \sigma(w_b)\sigma(w'_b).$$

Definition 3.4.1. *In the polynomial $\sigma(P)$ we refer to the indices $\sigma(j)$ and $\sigma(d+j)$ as a* matched pair of indices*, since in the monomial $\sigma(w_b)\sigma(w'_b)$ it is required that the variables $x_{b_j,\sigma(j)}$ and $x_{b_j,\sigma(d+j)}$ have the same first index b_j.*

For $\sigma = id$, the matched pairs are $(j, d+j)$ for $1 \le j \le d$.

Lemma 3.4.2.

- *The set-multilinear polynomial $\sigma(P)$ can be computed by a monotone set-multilinear ABP of size $O(d)$.*

- *For any $\sigma_1, \sigma_2, \ldots, \sigma_s \in S_{2d}$ the polynomial $\sum_{j=1}^{s} \sigma_i(P)$ can be computed by a monotone set-multilinear ABP of size $O(sd)$.*

Proof. As

$$\sigma(P) = \prod_{i=1}^{d} (x_{0,\sigma(i)}x_{0,\sigma(d+i)} + x_{1,\sigma(i)}x_{1,\sigma(d+i)}),$$

clearly $\sigma(P)$ has an $O(d)$ monotone set-multilinear formula. Hence, it has a monotone set-multilinear ABP of size $O(d)$. The second part of the lemma is an immediate consequence.

□

We will show that there exist a set of d permutations $\sigma_1, \sigma_2, \ldots, \sigma_d \in S_{2d}$ with the following property: for any permutation $\tau \in S_{2d}$ there is a σ_i from this set such that any τ-interval multilinear circuit that computes $\sigma_i(P)$ requires size $2^{\Omega(d)}$.

Note that, in contrast, given a $\sigma(P)$ there is always a permutation $\tau \in S_{2d}$ such that $\sigma(P)$ can be computed by a small τ-interval multilinear circuit. Indeed, the ABP computing $\sigma(P)$ described in Lemma 3.4.2 is an interval-multilinear ABP w.r.t. the ordering $\sigma(1), \sigma(d+1), \sigma(2), \sigma(d+2), \ldots, \sigma(d), \sigma(2d)$.

Interval multilinear circuits and noncommutative circuits

We first observe that a τ-interval multilinear circuit computing a set-multilinear polynomial in $\mathbb{F}[X]$, $X = \sqcup_{i=1}^{d} X_i$, is essentially like a noncommutative circuit computing a noncommutative polynomial over the variables X, whose monomials can be considered as words of the form $x_{i_1}x_{i_2}\ldots x_{i_d}$, where $x_{i_j} \in X_{\tau(j)}$ for $1 \leq j \leq d$.

In [HWY10a], Hrubes et al have related the well-known sum-of-squares (in short, SOS) conjecture (also see [Sha00]) to lower bounds for *noncommutative arithmetic circuits*. Our results in this section are based on their work. We recall the conjecture.

The sum-of-squares (SOS) conjecture: Consider the question of expressing the bi-

quadratic polynomial

$$SOS_k(x_1, \ldots, x_k, y_1, \ldots, x_k) = (\sum_{i\in[k]} x_i^2)(\sum_{i\in[k]} y_i^2)$$

as a sum of squares ($\sum_{i\in[s]} f_i^2$) for the least possible s, where each f_i is a homogeneous bilinear polynomial. The conjecture states that $s = \Omega(k^{1+\epsilon})$ over the field of complex numbers $\mathbb{C}$ (or the algebraic closure of any field $\mathbb{F}$ such that $char(\mathbb{F}) \neq 2$).

The following lower bound is shown in [HWY10a] assuming the SOS conjecture.

Theorem 3.4.3. [HWY10a] *Assuming the SOS conjecture over field $\mathbb{F}$, any noncommutative circuit computing the polynomial $ID = \sum_{w\in\{x_0,x_1\}^d} ww$ in* noncommuting variables x_0 *and x_1 requires size $2^{\Omega(d)}$.*

The above theorem implies the following conditional lower bound for interval multilinear circuits.

Corollary 3.4.4. *Assuming the SOS conjecture, for any $\sigma \in S_{2d}$ a σ-interval multilinear circuit computing the set-multilinear polynomial $\sigma(P)$ requires size $2^{\Omega(d)}$.*

Proof. Let C be a size s σ-interval multilinear circuit for $\sigma(P)$. The SOS conjecture combined with the following claim clearly yields the corollary.

Claim 3.4.1. *From C we can obtain a size s homogeneous noncommutative circuit for the noncommutative polynomial $\sum_{w\in\{x_0,x_1\}^d} ww$.*

Proof of Claim. By definition, the index sets of every gate in circuit C is a σ-interval. We construct a noncommutative circuit C' of size s from C as explained below:

1. Replace variable $X_{b,\sigma(i)}$ by x_b for $1 \leqslant i \leqslant 2d$ and $b \in \{0, 1\}$ at the inputs.

2. Interpret all $\times$ gates as noncommutative product gate with inputs ordered left to right as per their respective interval locations.

By construction, each gate g in C' has an interval I_g in $[1-(2d)]$ associated with it. In particular, if a gate g of degree k in C has interval $\{\sigma(i), \sigma(i+1), \cdots, \sigma(i+k-1)\}$ for some $i \leqslant d-k$, then the corresponding gate g in C' has interval $I_g = \{i, i+1, \cdots, i+k-1\}$. We claim that the output gate of C' computes the polynomial $\sum_{w\in\{x_0,x_1\}^d} ww$. We prove this by an inductive argument on the circuit structure.

In particular, we show that gate g in C computes a monomial $\prod_j x_{b_j,\sigma(j)}$ with coefficient $\alpha \in \mathbb{F}$ iff the corresponding gate g in C' computes monomial $\prod_j x_{b_j}$ with coefficient $\alpha \in \mathbb{F}$. Let $\text{Coeff}_g(m')$ denote the coefficient of monomial m' in the polynomial computed by g in C'. Similarly, let $\text{Coeff}_g(m)$ denote the coefficient of monomial m in the polynomial computed by g in C.

- For the base case, suppose gate g in C' is an input gate with label x_{b_i} and interval $[i]$, then clearly gate g in C is an input gate with variable $x_{b_i,\sigma(i)}$.

- Suppose gate g in C' is a + gate of degree k, where $g = g_1 + g_2$, and all three gates have interval $I = [i, i+1, \ldots, i+k-1]$ labeling them. Further, suppose $m' = x_{b_i}x_{b_{i+1}}\cdots x_{b_{i+k-1}}$ is a monomial computed at gate g with coefficient $\alpha_{m'} \in \mathbb{F}$. Then $\alpha_{m'} = \text{Coeff}_{g_1}(m') + \text{Coeff}_{g_2}(m')$. By induction hypothesis, $\text{Coeff}_{g_1}(m')$, $\text{Coeff}_{g_2}(m')$ in C' equal $\text{Coeff}_{g_1}(m)$, $\text{Coeff}_{g_2}(m)$ in C respectively, where monomial $m = \prod_{j=i}^{i+k-1} x_{b_j,\sigma(j)}$. Thus, $\text{Coeff}_g(m') = \text{Coeff}_g(m)$.

- Suppose g is a $\times$ gate in C' of degree k. Let $g = g_1 \times g_2$, and $\deg(g_1) = k_1$. Let $I = I_1 \uplus I_2$ be the intervals corresponding to gates g, g_1, and g_2, and $I = [i, i+1, \ldots, i+k-1]$. Let $m' = \prod_{j=i}^{i+k-1} x_{b_j}$ be a monomial with coefficient $\alpha_{m'} \in \mathbb{F}$ computed at gate g'. Then we can write $\alpha_{m'} = \text{Coeff}_{g_1}(m'_1) \times \text{Coeff}_{g_2}(m'_2)$. By induction hypothesis, the coefficients $\text{Coeff}_{g_1}(m'_1)$ and $\text{Coeff}_{g_2}(m'_2)$ in C' are the same as the coefficients $\text{Coeff}_{g_1}(m_1)$, $\text{Coeff}_{g_2}(m_2)$ in C, respectively. Here, notice that the monomials m_1 and m_2 are uniquely defined from m'_1 and m'_2 and the intervals I_1 and I_2. It follows that $\text{Coeff}_g(m')$ in C' equals $\text{Coeff}_g(m)$ in C.

□

Remark 3.4.1. *In [HWY10a], the connection between the SOS conjecture and lower bounds for the noncommutative polynomial* $\sum_{w\in\{x_0,x_1\}^d} ww$ *is made by considering first writing it as* $\sum_{w_1,w_2\in\{x_0,x_1\}^{d/2}} w_1w_2w_1w_2$. *Then the noncommutative degree-*$d/2$ *monomials* w_1 *and* w_2 *are treated as single commuting variables which allows the polynomial to be written as a product of two quadratics* $(\sum w_1^2)(\sum w_2^2)$. *Here* w_1 *and* w_2 *run over two sets of* $2^{d/2}$ *distinct variables each.*

Now, the SOS conjecture applied to this biquadratic polynomial implies that writing $(\sum w_1^2)(\sum w_2^2)$ *as* $\sum_{i=1}^t f_i^2$, *for bilinear forms* f_i, *implies* $t = \Omega(2^{d/2(1+\epsilon)})$ *for a constant* $\epsilon > 0$. *In [HWY10a, Corollary 5.4] it is shown that any size* s *noncommutative circuit for* $\sum_{w\in\{x_0,x_1\}^d} ww$ *can be transformed into a sum of squares* $\sum_{i=1}^t f_i^2$ *such that* $t = O(d^3s2^{d/2})$ *which implies the lower bound of* $2^{\Omega(d)}$ *on the circuit size* s.

Motivated by the connection described in the above remark we obtain the following easy lemma.

Lemma 3.4.5. *For even* d, *partition* $[2d]$ *into four intervals of size* $d/2$ *each:* $I_1 = [1, 2, \ldots, d/2]$, $I_2 = [d/2 + 1, \ldots, d]$, $I_3 = [d + 1, \ldots, 3d/2]$, *and* $I_4 = [3d/2 + 1 \ldots 2d]$. *Let* $\sigma \in S_{2d}$ *be any permutation such that* $\sigma(I_j) = I_j, 1 \leq j \leq 4$. *Then, assuming the SOS conjecture, any id-interval multilinear circuit computing the polynomial* $\sigma(P)$ *requires size* $2^{\Omega(d)}$.

Proof. By definition, $\sigma(P) = \sum_{b\in\{0,1\}^d} \sigma(w_b)\sigma(w'_b)$. Since σ stabilizes each $I_j, 1 \leq j \leq 4$, we can write $\sigma(w_b) = w_1w_2$ and $\sigma(w'_b) = w'_1w'_2$, where the matched pairs are between w_1 and w'_1, and between w_2 and w'_2, respectively. Now, by substituting the *same* variable for each matched pair, we obtain the polynomial $(\sum w_1^2)(\sum w_2^2)$. As explained in Remark 3.4.1, we can treat this as a biquadratic polynomial, where w_1 and w_2 are single variables that run over variable sets of size $2^{d/2}$ each. The proof argument in [HWY10a]

can now be applied to yield that any id-interval circuit size computing $\sigma(P)$ has size $2^{\Omega(d)}$, assuming the SOS conjecture for the biquadratic polynomial $(\sum w_1^2)(\sum w_2^2)$. □

We will use the probabilistic method to show the existence of the set of permutations $\sigma_i, 1 \leq i \leq d$ in S_{2d}, such that for each $\tau \in S_{2d}$ there is some $\sigma_i(P), i \in [d]$ that requires size $2^{\Omega(d)}$ τ-interval multilinear circuits. We will require the following concentration bound.

Lemma 3.4.6. [DP09, Theorem 5.3, page 68] *Let $X_1, \cdots, X_n$ be any n random variables and let f be a function of $X_1, X_2 \ldots, X_n$. Suppose for each $i \in [n]$ there is $c_i \geq 0$ such that*

$$|\mathbb{E}[f|X_1, \cdots, X_i] - \mathbb{E}[f|X_1, \cdots, X_{i-1}]| \leq c_i.$$

Then for any $t > 0$, we have the bound $\text{Prob}[f < \mathbb{E}[f] - t] \leq exp(-\frac{2t^2}{c})$, *where* $c = \sum_{i \in [n]} c_i^2$.

Lemma 3.4.7. *Let $\sigma \in S_{2d}$ be a permutation picked uniformly at random. For any $\tau \in S_{2d}$, the probability that $\sigma(P)$ is computable by a τ-interval multilinear circuit of size $2^{o(d)}$ is bounded by $e^{-\Omega(d)}$, assuming the SOS conjecture.*

Proof. In the polynomial $P = \sum_{b \in \{0,1\}^d} w_b w'_b$ the *matched pairs*, as defined earlier, are the index pairs $(i, d+i), 1 \leq i \leq d$. As in Lemma 3.4.5, we partition the index set $[2d]$ into four consecutive $d/2$-size intervals $I_1 = [1, \ldots, d/2]$, $I_2 = [d/2+1, \ldots, d]$, $I_3 = [d+1, \ldots, 3d/2]$, and $I_4 = [3d/2+1, \ldots, 2d]$. Note that $d/2$ of the matched pairs are between I_1 and I_3 and the remaining $d/2$ are between I_2 and I_4. Consider the following two subsets of matched pairs of size $d/8$ each:

$$\begin{aligned} E_1 &= \{(i, d+i) \mid 1 \leq i \leq d/8\} \\ E_2 &= \{(d/2+i, 3d/2+i) \mid 1 \leq i \leq d/8\}. \end{aligned}$$

The pairs in E_1 are between I_1 and I_3 and pairs in E_2 are between I_2 and I_4. Let $\sigma \in S_{2d}$ be a permutation picked uniformly at random. We say $(i, d+i) \in E_1$ is *good* if $\sigma(i) \in I_1$ and $\sigma(d+i) \in I_3$. Similarly, $(d/2+i, 3d/2+i) \in E_2$ is called *good* if $\sigma(d/2+i) \in I_2$ and $\sigma(3d/2+i) \in I_4$. Let X_i, $1 \leq i \leq d/8$, be indicator random variables which take the value 1 iff the pair $(i, d+i) \in E_1$ is good. Similarly, define indicator random variables X_i' corresponding to pairs $((d/2+i, 3d/2+i) \in E_2, 1 \leq i \leq d/8$.

For each $i \in [d/8]$, $Y_i \in \{X_i, X_i'\}$ we have

$$\text{Prob}_{\sigma \in S_{2d}}[Y_i = 1] \geqslant \left(\frac{(d/2 - d/8)}{2d}\right)^2 = 9/256 > 1/64.$$

Let $f = \sum_{i=1}^{d/8} X_i$ and $f' = \sum_{i=1}^{d/8} X_i'$. Clearly, we have

$$\text{E}[f] \geq \frac{d}{8 \times 64} = \frac{d}{512}.$$

Furthermore, we also have for each $i : 1 \leq i \leq d/8$

$$|\mathbb{E}[f|X_1, X_2, \ldots, X_i] - \mathbb{E}[f|X_1, X_2, \ldots, X_{i-1}]| \leq 1$$

Applying Lemma 3.4.6, with $t = d/1024$ we deduce that

$$\text{Prob}_{\sigma \in S_{2d}}[f < \frac{d}{1024}] \quad \leq \quad e^{-\alpha d},$$

where $\alpha > 0$ is some constant independent of d. Similarly, we also have

$$\text{Prob}_{\sigma \in S_{2d}}[f' < \frac{d}{1024}] \leq e^{-\alpha d},$$

Combining the two bounds yields

$$\text{Prob}_{\sigma \in S_{2d}}[f \geq \frac{d}{1024} \text{ and } f' \geq \frac{d}{1024}] \geq 1 - 2e^{-\alpha d}.$$

Thus, with probability $1 - 2e^{-\alpha d}$ there are $d/1024$ pairs $(\sigma(i), \sigma(d+i))$ such that $\sigma(i) \in I_1$ and $\sigma(d+i) \in I_3$, and there are $d/1024$ pairs $(\sigma(d/2+i), \sigma(3d/2+i))$ such that $\sigma(d/2+i) \in I_2$ and $\sigma(3d/2+i) \in I_4$. If we set all other variables in the polynomial $\sigma(P)$ to 1, we can apply Lemma 3.4.5 to the resulting polynomial (with d replaced by $d/1024$ in the lemma) which will yield the lower bound of $2^{\Omega(d)}$ for any id-interval multilinear circuit computing $\sigma(P)$, for a random σ with probability $1 - 2e^{-\alpha d}$. For any τ-interval multilinear circuit too the same lower bound applies because $\tau\sigma$ is also a random permutation in S_{2d} with uniform distribution.

□

We are now ready to state and prove the main result of this section.

Theorem 3.4.8. *There is a set-multilinear polynomial $f \in \mathbb{F}[X]$, where $X = \sqcup_{i=1}^{\log d+2d} X_i$ and $X_i = \{x_{0,i}, x_{1,i}\}, 1 \leq i \leq \log d + 2d$ such that f has an $O(d^2)$ size monotone set-multilinear ABP and, assuming the SOS conjecture, for any $\tau \in S_{\log d+2d}$ any σ-interval multilinear circuit computing f has size $2^{\Omega(d)}$.*

Proof. Suppose the SOS conjecture holds. Let $\sigma \in S_{2d}$ be some permutation. By Lemma 3.4.7, if we pick permutations $\sigma_1, \sigma_2, \ldots, \sigma_d \in S_{2d}$ independently and uniformly at random, then the probability that each $\sigma_i(P)$ can be computed by a fixed σ-interval multilinear circuit is bounded by $(2e^{-\alpha d})^d = e^{-\Omega(d^2)}$. As $|S_{2d}| = (2d)!$, by the union bound

it follows that there exist permutations $\sigma_1, \sigma_2, \ldots, \sigma_d \in S_{2d}$ such that for any $\sigma \in S_{2d}$ at least one of the $\sigma_i(P)$ requires $2^{\Omega(d)}$ size σ-interval multilinear circuits.

We fix such a set of permutations $\sigma_1, \sigma_2, \ldots, \sigma_d \in S_{2d}$ and define the polynomial f by "interpolating" the $\sigma_i(P)$. To this end, we need the fresh $\log d$ variable sets. For each $c : 1 \leq c \leq d$, let its binary encoding also be denoted by c, where $c \in \{0,1\}^{\log d}$. Let u_c denote the monomial

$$u_i = \prod_{j=2d+1}^{2d+\log d} x_{c_j,j}.$$

Hence the monomial u_c can also be seen as an encoding of $c : 1 \leq c \leq d$. We define

$$f = \sum_{c \in \{0,1\}^{\log d}} u_c \sigma_c(P).$$

Clearly, $f \in \mathbb{F}[X]$ and for each 0-1 assignment $c \in \{0,1\}^d$ to the variables in $X_j, 2d+1 \leq j \leq 2d + \log d$, the polynomial f becomes $\sigma_c(P)$.

Let $\tau \in S_{\log d + 2d}$ be an arbitrary permutation. Let $\hat{\tau} \in S_{2d}$ denote the relative ordering of the indices in $[1, 2, \ldots, d]$ induced by τ.

Suppose f has a $2^{o(d)}$ size τ-interval multilinear circuit. Then, by different 0-1 assignments $c \in \{0,1\}^d$ to variables in $X_j, 2d+1 \leq j \leq 2d + \log d$ we will obtain a $2^{o(d)}$ size $\hat{\tau}$-interval multilinear circuit for each $\sigma_c(P)$ which is a contradiction to the choice of the σ_i. □

Finally, we show for monotone circuits that the analogue of Theorem 3.4.8 holds unconditionally.

Theorem 3.4.9. *There is a set-multilinear polynomial $f \in \mathbb{F}[X]$, where $X = \sqcup_{i=1}^{\log d + 2d} X_i$ and $X_i = \{x_{0,i}, x_{1,i}\}, 1 \leq i \leq \log d + 2d$ such that for any $\tau \in S_{\log d + 2d}$ any monotone τ-interval multilinear circuit computing f has size $2^{\Omega(d)}$.*

Proof. Let

$$f = \sum_{c \in \{0,1\}^{\log d}} u_c \sigma_c(P),$$

be the polynomial defined in Theorem 3.4.8. By Corollary 3.4.2, f has small monotone set-multilinear ABPs. Let C be any monotone τ-interval multilinear circuit of size s computing f.

The *bilinear complexity* of $(\sum_{i=1}^{k} x_i^2)(\sum_{j=1}^{k} y_j^2)$ [HWY10a, Section 1.3] is the minimum t such that $(\sum_{i=1}^{k} x_i^2)(\sum_{j=1}^{k} y_j^2)$ equals $\sum_{i=1}^{t} f_i f_i'$, for bilinear forms f_i and f_i' (both f_i and f_i' are bilinear in the variable sets $\{x_1, x_2, \ldots, x_k\}$ and $\{y_1, y_2, \ldots, y_k\}$). It is shown in [HWY10a, Theorem 1.6] that a lower bound of $\Omega(k^{1+\epsilon})$ for constant $\epsilon > 0$ yields a $2^{\Omega(d)}$ size lower bound for noncommutative circuits computing $ID = \sum_{w \in \{x_0, x_1\}^d} ww$ (over $\mathbb{F}$ algebraically closed such that $char(\mathbb{F}) \neq 2$). The connection to the SOS conjecture comes (for fields of characteristic different 2) because the bilinear complexity is related by a constant factor to the minimum sum-of-squares expression.

When f_i and f_i' are monotone bilinear forms we have the following.

Claim 3.4.2. *If $(\sum_{i=1}^{k} x_i^2)(\sum_{j=1}^{k} y_j^2)$ equals $\sum_{i=1}^{t} f_i f_i'$, for monotone bilinear forms f_i and f_i' then $t \geq k^2$.*

Proof of Claim. By assumption, the f_i and f_i' are all monotone bilinear in variable sets $\{x_1, x_2, \ldots, x_k\}$ and $\{y_1, y_2, \ldots, y_k\}$. For each $i \in [t]$, all nonzero terms in the product $f_i f_i'$ must be of the form $x_j^2 y_\ell^2$ because there are no cancellations. Therefore, if f_i has a nonzero term of the form $x_j y_\ell$ then f_i' can have only $x_j y_\ell$ as a nonzero term and no other terms, because the coefficients are nonnegative in f_i and f_i' and there are no cancellations. Thus, each product $f_i f_i'$ involves precisely one pair (x_j, y_ℓ) for some $j, \ell \in [k]$. This forces $t \geq k^2$.

It follows from the proof of [HWY10a, Corollary 5.4] and the above claim that any monotone noncommutative circuit for $ID = \sum_{w \in \{x_0, x_1\}^d} ww$ is of size $2^{\Omega(d)}$. We observe that Corollary 3.4.4, Lemma 3.4.5, Lemma 3.4.7, and Theorem 3.4.8 all hold for monotone

interval multilinear circuits unconditionally, because the SOS conjecture is true in the monotone case by the above claim. This completes the proof. □

3.5 Parse tree restrictions on set-multilinear circuits

In this section we investigate lower bounds for set-multilinear circuits computing the Permanent that satisfy some "semantic" restrictions on the parse trees of monomials.

Thus, given a set-multilinear circuit C we can consider: (a) the set of parse tree types of the entire circuit C, and (b) the set of parse tree types of a given monomial.

For set-multilinear ABPs computing a degree d polynomial, parse trees of monomials are just simple paths of length d, and the corresponding parse tree types are also simple paths of length d. Furthermore, every ROABP have a unique parse tree type. As exponential lower bounds for ROABPs computing the permanent are known [Nis91] using the partial derivative method, a natural question is whether we can obtain lower bounds when more than one parse tree type is allowed in the set-multilinear circuit. We consider the following two restrictions and prove lower bounds.

1. Set-multilinear circuits with few parse tree types.
2. *Unambiguous* set-multilinear circuits: i.e. circuits in which each monomial has a unique parse tree type (but the number of different parse tree types in the circuit is unbounded).

3.5.1 Set-multilinear circuits with few parse tree types

Let C be a set-multilinear circuit computing a degree-n polynomial $f \in \mathbb{F}[X]$ for variable partition $X = \sqcup_{i=1}^{n} X_i$ such that *the total number* of parse tree types in the circuit is bounded, say, by a polynomial in n. Can we prove superpolynomial lower bounds for such circuits?

We are able to show non-trivial lower bounds when the number of parse trees are small enough. More precisely, suppose C is a set-multilinear circuit that computes PER_n and C has at most r parse tree types. Then we show that C is of size $2^{\Omega(\frac{n}{r \log n})}$. Here is a brief outline of the proof: We decompose C into a sum of r many set-multilinear formulas $C_i, 1 \leq i \leq r$, such that each C_i has a unique proof tree type. Next, we convert each such C_i into an ROABP A_i. Thus, the sum of these ROABPs $A_i, 1 \leq i \leq r$ computes the permanent PER_n and we can apply Corollary 2.5.4 to the sum of these A_i's and obtain the claimed lower bound. We now present the details.

Lemma 3.5.1. *Let C be a set-multilinear circuit of size s computing a degree-d polynomial $P \in \mathbb{F}[X]$. If all parse trees in C have the same parse tree type T, then C can be efficiently transformed into a set-multilinear formula C' of size $s^{O(\log d)}$ such that in C' too all parse trees have the same parse tree type T', where T' depends only on T (and not on the circuit C).*

Proof. The proof is a standard depth reduction argument as, for example Hyafil's result [Hya79]. The only extra work we need to do is argue about the parse tree type.

We prove the lemma by induction on the size of the index set of the output gate of C (i.e., degree of P). At the input gates, where index set is a singleton set, it clearly holds. Suppose the index set of the output gate is of size at least 2. Let T_C denote the unique parse tree type for all parse trees in C. Each node v of T_C is labeled by its index set $I_v \subseteq [d]$. As T_C is a binary tree, there is a vertex u such that $\frac{d}{3} \leqslant |I_u| \leqslant \frac{2d}{3}$. Let $S_u = \{v \in C \mid I_v = I_u\}$. Let $\hat{C}_v$ denote the set-multilinear circuit obtained from C by (i) setting to zero all the gates in $S_u \setminus \{v\}$, and (ii) replacing the gate v by the constant 1. Let Q_v denote the polynomial computed at the output gate of $\hat{C}_v$. Its index set is $[d] \setminus I_v$. Let P_v denote the polynomial computed at a gate v of C. Then we can clearly write

$$P = \sum_{v \in S_u} P_v Q_v.$$

Let C_v denote the subcircuit of C with output gate v. Note that

$$\frac{d}{3} \leqslant deg(P_v), deg(Q_v) \leqslant \frac{2d}{3}.$$

Thus, for each $v \in S_u$ both P_v and Q_v are set-multilinear polynomials computed by set-multilinear circuits (C_v and $\hat{C}_v$, respectively) of size at most s. Furthermore, these circuits also have the property that all parse trees has the same parse tree type (otherwise, C would not have the property).

By induction hypothesis, for each $v \in S_u$ we have set-multilinear formulas F_v and $\hat{F}_v$ such that:

- F_v and $\hat{F}_v$ compute P_v and Q_v, respectively.
- The size of F_v as well as $\hat{F}_v$ is bounded by $s^{O(\log \frac{2d}{3})}$.
- All parse trees in F_v have a unique parse tree type. All parse trees in $\hat{F}_v$ have a unique parse tree type.

Furthermore, the circuit C has the following stronger property: suppose v and v' are two gates with the same index set $I_v = I_{v'}$. Then the unique parse tree type associated with subcircuit C_v is the same as the unique parse tree type for subcircuit $C_{v'}$. Otherwise, the circuit C would not have a unique parse tree type associated with it.

Since each subcircuit $C_v, v \in S_u$ has the same index set and, thus, the same parse tree type associated to it, it follows by induction hypothesis that all the formulas $F_v, v \in S_u$ also have the same unique parse tree type. The same property holds for $\hat{C}_v, v \in S_u$ and hence $\hat{F}_v, v \in S_u$.

Therefore, each of the product polynomials $P_v Q_v, v \in S_u$, computed by the formulas $F_v \times \hat{F}_v, v \in S_u$, with a $\times$ output gate, all have the same parse tree type. Thus, since $|S_u| \leqslant s$ the polynomial $P = \sum_{v \in S_u} P_v Q_v$ has a set multilinear formula C' of size $\leqslant s(2s^{O(\log \frac{2d}{3})}) \leqslant$

$s^{O(\log d)}$ and all the parse trees of C' have the same parse tree type T'. Furthermore, it is clear that T' depends only on T_C. This completes the proof of the lemma. □

Lemma 3.5.2. *Let C be a set-multilinear formula of size s computing a degree d polynomial P, such that C has a unique parse tree type T. Then C can be transformed into a set-multilinear ABP that has a unique parse tree type T' which depends only on T and not on the formula C.*

Proof. We proof the lemma by induction on the size of formula C. Suppose the output gate of C is a + gate. Let C_1 and C_2 be the two subformulas. Since C has a unique parse tree type T, both subcircuits C_1 and C_2 have the same unique tree type T. By induction hypothesis the two subformulas C_1 and C_2 of C with same parse tree T can be converted into set-multilinear ABPs A_1 and A_2, respectively, such that both A_1 and A_2 have the same unique parse tree type T'. The set-multilinear ABP A for their sum is obtained by "parallel composition" of the two ABPs A_1 and A_2. Clearly, A has the same unique parse tree type T'.

Next, suppose output gate of formula C is a $\times$ gate. Since C has unique proof tree type T, the subcircuits C_1 and C_2 of C have unique parse tree types T_1 and T_2, respectively. Let T_1 be the left subtree of T and T_2 be the right subtree. By induction hypothesis both C_1 and C_2 have ABPs A_1 and A_2 with unique parse tree types T_1 and T_2 respectively. In order to compute their product, we can take the "series composition" of A_1 and A_2, which yields the desired set-multilinear ABP with unique parse tree type. □

Lemma 3.5.3. *Let C be a set-multilinear circuit of size s computing polynomial $P \in \mathbb{F}[X]$ of degree d such that C has r distinct parse tree types. Then from C we can construct set-multilinear circuits $C_i, 1 \leq i \leq r$ such that $\sum_{i\in[r]} C_i$ computes polynomial P, each C_i is of size bounded by s, and each C_i has a unique parse tree type.*

Proof. Let the parse tree types of C be $T_1, T_2, \cdots, T_r$. We describe the construction of circuit C_i from C corresponding to parse tree type T_i. In C_i, we label 0 for all the outgoing

edges of gates v in C whose index set $I_v \subseteq [n]$ is not equal to any of the index sets of parse tree T_i. Clearly, the parse trees of C_i are precisely all parse trees of parse tree type T_i present in circuit C and with the same coefficients. Therefore, $\sum_{i=1}^{r} C_i$ computes polynomial P. This completes the proof. □

As a consequence of the above lemmas we obtain the following.

Theorem 3.5.4. *Let C be a set-multilinear circuit computing the permanent polynomial PER_n (or determinant DET_n) such that C has at most r distinct parse tree types. Then the size of C is $\Omega(2^{\frac{n}{r \log n}})$.*

Proof. Let C be of size s. The idea is to convert C into a narrow set multilinear ABPs and apply the lower bound for narrow set multilinear ABPs (Corollary 2.5.4). First, by Lemma 3.5.3 we compute set-multilinear circuits $C_i, 1 \leq i \leq r$, of size s each, such that C_i has unique parse tree type T_i. Next, by Lemma 3.5.1, each C_i can be converted into a set-multilinear formula C_i' of size $s^{O(\log n)}$, also of unique parse tree type. Finally, by Lemma 3.5.2 C_i' can be transformed into a homogeneous d-layer set-multilinear ABP A_i of size $s^{O(\log n)}$ which has unique parse tree type. Their sum, $\sum_{i=1}^{r} A_i$, obtained by "parallel composition", is a set-multilinear ABP A with at most r many parse tree types. Clearly, it follows that at each layer $i \in [n]$ of the ABP A, the typewidth is bounded by r.

By Lemma 3.5.1, the size of ABP A is bounded by $rs^{O(\log n)}$. As A computes PER_n (or DET_n), by Corollary 2.5.4 the size of A is lower bounded by $\Omega(2^{\frac{n}{r}})$. Thus, $rs^{O(\log n)} = \Omega(2^{\frac{n}{r}})$, which implies that $s = \Omega(2^{\frac{n}{r \log n}})$. □

3.5.2 Unambiguous set-multilinear circuits

Definition 3.5.5. *A set-multilinear circuit C computing a degree d polynomial $f \in \mathbb{F}[X]$, with variable partition $X = \sqcup_{i=1}^{d} X_i$, is said to be* unambiguous *if for every monomial $m \in X^d$ has a* unique *parse tree type in circuit C.*

In unambiguous circuits different monomials are allowed to have different parse tree types. Furthermore, each monomial can have many parse trees, only the parse tree types have to be all identical. Unambiguous boolean circuits and unambiguous computation in general are well-studied in complexity theory.

Clearly, ROABPs are a special case of unambiguous set-multilinear ABPs. Also, unambiguous set-multilinear circuits can have many parse tree types, unlike what we considered in Section 3.5.1.

Theorem 3.5.6. *Let C be an unambiguous set-multilinear circuit with variable partition $X = \sqcup_{i=1}^{n} X_i$, where $X_i = \{X_{ij} \mid 1 \leq j \leq n\}$, such that C computes the permanent polynomial PER_n. Then C is of size $2^{\Omega(n)}$.*[1]

Proof. Suppose C is an unambiguous size set-multilinear circuit of size s computing the permanent polynomial PER_n. Let $G_{n/3}$ denote the set of all product gates g in C such that $deg(g) > n/3$ and $deg(g_1) \leq n/3$ and $deg(g_2) \leq n/3$, where g_1 and g_2 are the gates that are input to g. It follows that $n/3 < deg(g) \leq 2n/3$. Furthermore, every parse tree of the circuit C has at least one gate from $G_{n/3}$ and at most two gates from $G_{n/3}$.

Since C is unambiguous, every monomial of PER_n has a unique parse tree type. Consequently, by the pigeon-hole principle there is an index set $I \subseteq [n]$ of size $n/3 < |I| \leq 2n/3$ with the following property: for at least $n!/s$ many monomials m of PER_n, every parse tree of m has a gate in $G_{n/3}$ of index set I. Let

$$\hat{G} = \{g \in G_{n/3} \mid \text{ index set of } g \text{ is } I\}.$$

We will lower bound $|\hat{G}|$. For $g \in \hat{G}$ let C_g be the subcircuit of C rooted at the gate g. Let $\partial_g C$ denote the partial derivative of the output gate of C w.r.t. gate g as defined in Section 2.4. A circuit for $\partial_g C$ can be obtained from C as follows:

[1] The same lower bound result holds for DET_n.

- For each gate $h \in \hat{G}$ such that $h \neq g$, label by 0 all outgoing edges from h.
- Replace gate g with constant 1.
- For all gates $h \in G_{n/3}$ such that $I_h \cap I \neq \emptyset$ label by 0 all outgoing edges of h.

Now, consider the circuit

$$C' = \sum_{g \in \hat{G}} C_g \partial_g C. \tag{3.1}$$

Since C_g and $\partial_g C$ are both set-multilinear circuits of size at most s, clearly the size of the set-multilinear circuit C' is bounded by $2s^2$.

Let M denote the set of monomials m of PER_n such that every parse tree of m has a gate in $\hat{G}$. By choice of I

$$|M| \geq \frac{n!}{s}.$$

Claim 3.5.1. *Let $f \in \mathbb{F}[X]$ denote the polynomial computed by C'. Then*

- *The coefficient of every monomial M in f is* 1.
- *The coefficient of any monomial not in M is* 0 *in f.*

Proof of Claim. Let $m \in X^n$ be any monomial. Since C is an unambiguous circuit, if some parse tree of m in C has a gate in $\hat{G}$ then every parse tree of m has a gate in C. Hence every parse tree of such a monomial m is accounted for in the circuit C'. Thus the net contribution of any such monomial m is the coefficient of m in PER_n. In particular, monomials in M have coefficient 1 and all other monomial in X^n have coefficient 0.

Similar to Equation 2.1, for the set-multilinear polynomial $f \in \mathbb{F}[X]$ with variable partition $X = \sqcup_{i=1}^{n} X_i$, where $X_i = \{X_{ij} \mid 1 \leq j \leq n\}$, we define matrix M_f whose rows

are indexed by degree $|I|$ set-multilinear monomials m_1 and columns by degree $n - |I|$ set-multilinear monomials m_2 such that m_1 is a monomial in variables $\sqcup_{i\in I} X_i$ and m_2 in variables $\sqcup_{i\notin I} X_i$ and

$$M_f(m_1, m_2) = f(m_1 m_2).$$

For any degree n set-multilinear monomial $m \in X^n$ we can uniquely write it as $m = m_1 m_2$, where m_1 is a monomial in variables $\sqcup_{i\in I} X_i$ and m_2 in variables $\sqcup_{i\notin I} X_i$. By the above claim

$$M_f(m_1, m_2) = \begin{cases} 1, & \text{if } m_1 m_2 \in M, \\ 0, & \text{otherwise.} \end{cases}$$

Furthermore, notice that for any other factorization $m = m_1' m_2'$ of m, the entry $M_f(m_1', m_2') = 0$, because the circuit C is unambiguous.

Claim 3.5.2. $s \geq rank(M_f) \geq \frac{\binom{n}{n/3}}{s}$.

Proof of Claim. To see that $s \geq rank(M_f)$ it suffices to note from Equation 3.1 that

$$M_f = \sum_{g\in\hat{G}} M_{C_g \partial_g C},$$

where each $M_{C_g \partial_g C}$ is a rank 1 matrix. Thus, $s \geq |\hat{G}| \geq rank(M_f)$.

Now we show the other inequality in the claim. For each subset $S \in \binom{[n]}{|I|}$ we group together the rows of matrix M_f indexed by monomials $m_1 = X_{i_1 j_1} X_{i2 j_2} \ldots X_{i_k j_k}$, where $I = \{i_1, i_2, \ldots, i_k\}$ and $S = \{j_1, j_2, \ldots, j_k\}$. Likewise, for each subset $T \in \binom{[n]}{n-|I|}$ we group together the columns indexed by monomials $m_2 = X_{i_1 j_1} X_{i2 j_2} \ldots X_{i_\ell j_\ell}$, where $[n] \setminus I = \{i_1, i_2, \ldots, i_\ell\}$ and $T = \{j_1, j_2, \ldots, j_\ell\}$.

The matrix M_f consists of different (S, T) blocks, corresponding to subsets $S \in \binom{[n]}{|I|}$ and $T \in \binom{[n]}{n-|I|}$. For each such S, only the $(S, [n] \setminus S)$ block has nonzero entries. All other blocks in the row corresponding to S or the columns corresponding to $[n] \setminus S$ are

zero. Furthermore, we note that the number of entries in each $(S, [n] \setminus S)$ block is clearly bounded by $(|I|!)(n - |I|)!$. Therefore, as there are $n!/s$ many 1's in matrix M_f, there are at least

$$\frac{n!}{s(n-|I|)!|I|!} = \frac{\binom{n}{|I|}}{s}$$

many *nonzero* $(S, [n] \setminus S)$ blocks, each of which contributes at least 1 to the rank of M_f. Hence,

$$rank(M_f) \geq \frac{\binom{n}{|I|}}{s}.$$

As $\binom{n}{|I|} \geq \binom{n}{n/3} = 2^{\Omega(n)}$, it follows that $s = 2^{\Omega(n)}$, which completes the lower bound proof. □

Remark 3.5.1. *It suffices to assume that the "top half" of the parse tree types are unambiguous for each monomial. More precisely, a* truncated parse tree type *is obtained from a parse tree type by deleting all nodes v such that $|I_v| \leq d/3$. Let C be a set-multilinear circuit computing* PER_n *such that C has the following property: each monomial $m \in X^n$ has at most one truncated parse tree type. The above proof yields the same lower bounds on the size of C.*

3.6 Summary and open problems

In this chapter we investigated lower bound questions for certain set-multilinear arithmetic circuits and ABPs. By imposing a restriction on the number of parse trees in set-multilinear circuits, we could prove nontrivial lower bounds for the Permanent. We also showed a separation between set-multilinear circuits and interval multilinear circuits, assuming the SOS conjecture. An open problem is to show the separation unconditionally.

Another interesting open question is proving lower bounds for set-multilinear circuits with polynomially (or even $O(n)$) many parse trees computing PER_n.

Chapter 4

Complete problems for the classes VP_{nc} & VSKEW_{nc}

4.1 Introduction

In this chapter we explore the noncommutative analogues, VP_{nc} and VNP_{nc}, of Valiant's algebraic complexity classes and show some striking connections to classical formal language theory. Our main results are the following:

- We show that Dyck polynomials (defined from the Dyck languages of formal language theory) are complete for the class VP_{nc} under $\leq_{abp}$ reductions. To the best of our knowledge, these are the first *natural* polynomial families shown to be VP_{nc}-complete.

- Likewise, it turns out that PAL (Palindrome polynomials defined from palindromes) are complete for the class VSKEW_{nc} (defined by polynomial-size skew circuits) under $\leq_{abp}$ reductions. The proof of these results is by suitably adapting the classical Chomsky-Schützenberger theorem showing that Dyck languages are the hardest CFLs.

The field of arithmetic complexity has a rich history, starting with the work of Strassen on matrix multiplication [Str69]. A central open problem of the field is proving superpolynomial size lower bounds for arithmetic circuits that compute the permanent polynomial PER_n. Motivated by this problem, Valiant, in his seminal work [Val79a], defined the arithmetic analogues of P and NP: namely VP and VNP. Informally, VP consists of multivariate (commutative) polynomials that have polynomial size circuits, over the field of rationals. The class VNP (which corresponds to the counting class #P in the world of Boolean complexity classes) has a more technical definition which we will give later. Valiant showed that PER_n is VNP-complete w.r.t. projection reductions. Thus, VP $\neq$ VNP if and only if PER_n requires arithmetic circuits of size superpolynomial in n. Over any field $\mathbb{F}$ the classes $\mathrm{VP}_{\mathbb{F}}$ and $\mathrm{VNP}_{\mathbb{F}}$ are similarly defined. Indeed, Valiant's proof actually shows that PER_n is complete for the class $\mathrm{VNP}_{\mathbb{F}}$ for any field $\mathbb{F}$ of characteristic different from 2. (Note: In this chapter, we will drop the subscript and simply use VP and VNP to denote the classes as the field $\mathbb{F}$ will either not matter or will be clear from the context.)

Nisan, in his 1990 paper [Nis91], explored the complexity of *noncommutative* arithmetic computations, in particular the complexity of computing the permanent with noncommutative computations. The noncommutative polynomial ring $\mathbb{F}\langle x_1, \ldots, x_n\rangle$ over a field $\mathbb{F}$ in noncommuting variables $x_1, x_2, \ldots, x_n$, consists of noncommuting polynomials in $x_1, x_2, \ldots, x_n$. These are just $\mathbb{F}$-linear combinations of words (we call them monomials) over the alphabet $X = \{x_1, \ldots, x_n\}$.

Analogous to commutative arithmetic circuits, we can define noncommutative arithmetic circuits for computing polynomials in $\mathbb{F}\langle X\rangle$. The only difference is that in noncommutative circuits inputs to multiplication gates are ordered from left to right. A natural definition of the noncommutative permanent polynomial PER_n, over $X = \{x_{ij}\}_{1\leqslant i,j\leqslant n}$, is

$$\mathrm{PER}_n = \sum_{\sigma\in S_n} x_{1,\sigma(1)}x_{2,\sigma(2)}\ldots x_{n,\sigma(n)}.$$

Can we show that PER_n requires superpolynomial size noncommutative arithmetic circuits? One would expect this problem to be easier than in the commutative setting. Indeed, for the model of noncommutative algebraic branching programs (ABPs), Nisan [Nis91] showed exponential lower bounds for PER_n (and even the determinant polynomial DET_n). Unlike in the commutative world, where ABPs are nearly as powerful as arithmetic circuits, Nisan [Nis91] could show an exponential separation between noncommutative circuits and noncommutative ABPs. However, showing that PER_n requires superpolynomial size noncommutative arithmetic circuits remains an open problem.

Analogous to VP and VNP, the classes VP_{nc} and VNP_{nc} can be defined, as has been done by Hrubes et al [HWY10b]. In [HWY10b] it is shown that PER_n is VNP_{nc}-complete w.r.t projections (this is the p-projection reducibility defined by Valiant [Val79a], which allows variables or scalars to be substituted for variables).

4.1.1 Main results

We begin with some formal definitions needed to state and explain the main results.

Definition 4.1.1. *A sequence $f = (f_n)$ of noncommutative multivariate polynomials over a field $\mathbb{F}$ is called a* polynomial family *(abbreviated as p-family henceforth) if both the number of variables in f_n and the degree of f_n are bounded by n^c for some constant $c > 0$.*

Definition 4.1.2.

1. *The class VBP_{nc} consists of p-families $f = (f_n)$ such that each f_n has an algebraic branching program (ABP) of size bounded by n^b for some $b > 0$ depending on f.*

2. *The class VP_{nc} consists of p-families $f = (f_n)$ such that each f_n has an arithmetic circuit of size bounded by n^b for some $b > 0$ depending on f.*

3. *A p-family $f = (f_n)$ is in the class VNP_{nc} if there exists a p-family $g = (g_n) \in \text{VP}_{nc}$*

such that for some polynomial $p(n)$

$$f_n(x_1, \ldots, x_{q(n)}) = \sum_{y_1,\ldots,y_{r(n)} \in \{0,1\}} g_{p(n)}(x_1, \ldots, x_{m(n)}, y_1, \ldots, y_{r(n)}).$$

where $r(n)$ is polynomially bounded.

4. *The class* VSKEW_{nc} *consists of p-families $f = (f_n)$ such that each f_n has a* skew *arithmetic circuit of size bounded by n^b for some $b > 0$ depending on f.*

We note that the class VBP_{nc} is defined through algebraic branching programs (ABPs) which intuitively correspond to acyclic finite automata. In fact noncommutative ABPs are also studied in the literature as multiplicity automata [BBB+00], and Nisan's rank lower bound argument [Nis91] is related to the rank of Hankel matrices in formal language theory [BR11]. Similarly, arithmetic circuits correspond to acyclic context-free grammars.

It turns out, as we will see in this chapter as well as in the next chapter, that this analogy goes further and shows up in the internal structure of VNP_{nc} and VP_{nc}. Our main results of this chapter are the following:

1. In Section 4.4, we prove that the Dyck polynomials are complete for VP_{nc} w.r.t $\leqslant_{abp}$ reductions. The result can be seen as an arithmetized version of the Chomsky-Schützenberger theorem [CS63] showing that the Dyck languages are the hardest CFLs. We note here that $\leqslant_{abp}$ reducibility is a generalization of the standard projection reducibility wherein instead of substitution by variables and scalars we allow substitutions by matrices (whose entries are variables/scalars). Section 4.3 has the formal definitions and a discussion on this reducibility.

2. On the same lines, in Section 4.5, we show that the Palindrome polynomials $PAL_n = \sum_{w \in \{x_0, x_1\}^n} w.w^R$ are complete for VSKEW_{nc} under $\leqslant_{abp}$ reducibility, again by adapting the proof of the Chomsky-Schützenberger theorem.

The following table summarizes the main results in this chapter.

P-family	Complexity Result	Remarks
$D_k, k \geqslant 2$	- VP_{nc}-Complete (Theorem 4.4.2) - VSKEW_{nc}-hard	w.r.t $\leq_{abp}$-reductions
PAL_d	VSKEW_{nc}-Complete (Theorem 4.5.1)	w.r.t. $\leq_{abp}$-reductions

Table 4.1: Summary of Results

4.2 Preliminaries

4.2.1 Polynomials

We define some p-families that are important for this chapter.

Palindrome Polynomials:

The p-family $\mathrm{PAL} = (\mathrm{PAL}_n)$ corresponds to the context-free language of even length palindromes.

$$\mathrm{PAL}_n = \sum_{w \in \{x_0, x_1\}^n} w w^R,$$

where w^R denotes the string obtained by reversing the string w.

Dyck Polynomials:

Let $X_i = \{(_1,)_1, ..., (_i,)_i\}$ for a fixed $i \in \mathbb{N}$ denote the set of i different types of matching left and right bracket pairs. The set of all well-balanced strings over alphabet X_i is inductively defined as below.

The empty string ϵ is well-balanced.

For each well-balanced string v over X_i, the strings $({}_j v)_j$ are well-balanced for $j \in \{1, 2, \ldots, i\}$.

For any two well-balanced strings v_1, v_2, their concatenation $v_1 v_2$ is well-balanced.

We define the polynomial $D_{i,n}$ over the variable set X_i to be sum of all strings in X_i^{2n} which are well-balanced. The $D_{i,n}$ are Dyck polynomials of degree $2n$ over i different types of brackets. The corresponding p-family is denoted $D_i = (D_{i,n})$.

4.3 The Reducibilities

In this Chapter as well as in the Chapter 5, we consider mainly three different notions of reducibility for our completeness results and for exploring the structure of the classes VNP_{nc}, VP_{nc} and VSKEW_{nc}.

4.3.1 The projection reducibility

The projection is basically Valiant's classical notion of reductions between p-families using which he showed VNP-completeness for PER_n and other p-families in his seminal work [Val79a]. Let $f = (f_n)$ and $g = (g_n)$ be two noncommutative p-families over a field $\mathbb{F}$, where $\forall n$ $f_n \in \mathbb{F}\langle X_n \rangle$ and $g_n \in \mathbb{F}\langle Y_n \rangle$. We say $f \leqslant_{proj} g$ if there are a polynomial $p(n)$ and a substitution map $\phi : Y_{p(n)} \to X_n \cup \mathbb{F}$ such that $\forall n$

$$f(X_n) = g(\phi(Y_{p(n)})).$$

As shown in [HWY10b], based on Valiant's original proof, the noncommutative PER_n p-family is VNP_{nc}-complete for $\leqslant_{proj}$-reducibility.

4.3.2 The indexed-projection reducibility

Let $[n] = \{1, 2, \cdots, n\}$. The indexed-projection is specific to the noncommutative setting. We say $f \leqslant_{iproj} g$ for p-families $f = (f_n)$ and $g = (g_n)$, where $deg(f_n) = d_n$, $deg(g_n) = d'_n$, $f_n \in \mathbb{F}\langle X_n\rangle$, and $g_n \in \mathbb{F}\langle Y_n\rangle$, if there are a polynomial $p(n)$ and indexed projection map

$$\phi : [d'_{p(n)}] \times Y_{p(n)} \to X_n \cup \mathbb{F},$$

such that on substituting $\phi(i, y)$ for variable $y \in Y_{p(n)}$ occurring in the i^{th} position in monomials of $g_{p(n)}$ we get polynomial f_n.

Clearly, $\leqslant_{iproj}$ is more powerful than $\leqslant_{proj}$ and we will show separations in this section.

4.3.3 The abp-reducibility

The $\leqslant_{abp}$ reducibility is the most general notion that we will consider. The $\leqslant_{abp}$ reduction essentially amounts to matrix substitutions for variables, where the matrices have scalar or variable entries (we can even allow constant-degree monomial entries). Formally, let $f_n \in \mathbb{F}\langle X_n\rangle$ and $g_n \in \mathbb{F}\langle Y_n\rangle$ as before. We say $f \leqslant_{abp} g$ if there are polynomials $p(n), q(n)$ and the substitution map $\phi : Y_{p(n)} \to M_{q(n)}(X_n \cup \mathbb{F})$ where $M_{q(n)}(X_n \cup \mathbb{F})$ stands for $q(n) \times q(n)$ matrices with entries from $X_n \cup \mathbb{F}$, with the property that $f(X_n)$ is the $(1, q(n))$-th entry of $g(\phi(Y_{p(n)}))$.

The $\leqslant_{abp}$ reducibility is implicitly used in [AS10], where it is shown that the noncommutative determinant polynomial cannot have polynomial-size noncommutative circuits unless the noncommutative permanent has such circuits. Essentially the result shown is that PER_n is $\leqslant_{abp}$ reducible to the noncommutative determinant.

We note that $\leqslant_{abp}$ is transitive.

Proposition 4.3.1. *Let $f, g, h \in \mathbb{F}\langle X\rangle$ such that $f \leqslant_{abp} g$ and $g \leqslant_{abp} h$ then $f \leqslant_{abp} h$.*

Proof. Let X_n, Y_n, Z_n denote the variable sets of f_n, g_n, h_n respectively. Let $\phi : Y_{p(n)} \to M_{q(n)}(X_n \cup \mathbb{F})$ and $\phi' : Z_{p'(n)} \to M'_{q'(n)}(Y_{p(n)} \cup \mathbb{F})$ be the substitution maps corresponding to reductions $f_n \leqslant_{abp} g_{p(n)}$ and $g_{p(n)} \leqslant_{abp} h_{p'(n)}$ respectively. The substitution map ψ for the abp-reduction from f to h is defined in the following way. For $z \in Z_{p'(n)}$, let $\psi(z)$ denotes a $r(n) \times r(n)$ matrix ($r(n) = q(n) \cdot q'(n)$) obtained from $\phi'(z)$ by substituting every scalar α in $\phi'(z)$ by $\alpha \cdot I_{q(n)}$ where $I_{q(n)}$ is $q(n) \times q(n)$ identity matrix and every variable y appearing in $\phi'(z)$ by $q(n) \times q(n)$ matrix $\phi(y)$. It is easy to see that if we substitute matrices $\psi(z)$ for variables z in $h_{p'(n)}$ we obtain polynomial f at $(1, r(n))^{th}$ entry of the resulting matrix. □

Proposition 4.3.2. *Let $f, g \in \mathbb{F}\langle X \rangle$ and suppose $f \leqslant_{abp} g$. Then*

$g \in \mathrm{VBP}_{nc}$ *implies* $f \in \mathrm{VBP}_{nc}$.

$g \in \mathrm{VP}_{nc}$ *implies* $f \in \mathrm{VP}_{nc}$.

$g \in \mathrm{VSKEW}_{nc}$ *implies* $f \in \mathrm{VSKEW}_{nc}$.

Proof. As $f \leqslant_{abp} g$, for every variable y of g we have polynomial sized matrix $\phi(y)$ such that on substituting $\phi(y)$ for variables y in g the top right corner entry of the resulting matrix is f.

Suppose g has a polynomial sized algebraic branching program P. W.l.o.g. assume that edges in P are labeled either with scalars or variable $y \in Var(g)$ where $Var(g)$ is set of variables of the polynomial g. To get polynomial sized ABP P' for f, we replace each edge of P with non-scalar label y by a small ABP with two layers, each layer containing k nodes where k is the size of matrix $\phi(y)$. An edge from i^{th} node in first layer to j^{th} node in the second layer is labeled with $\phi(y)(i, j)$. Clearly P' will compute f and has polynomial size.

Now suppose g has a polynomial sized arithmetic circuit C. The polynomial sized circuit C' for f is obtained simply by replacing + and × gates of C by small sub-circuits computing sum and product of two polynomial sized matrices respectively. If C is a skew circuit

so is C'. □

Remark 4.3.1. *We could as well have called abp-reductions as matrix-reductions, since the reductions are a generalization of projections with matrix substitutions instead of only scalars and variables. However, abp-reducibility seems to us a more appropriate name because the matrix-valued variable substitutions really captures the power of noncommutative ABPs. To see this, let $g = (g_n)$ be a p-family where $g_n = y_1 y_2 \dots y_n$ consists of a single degree-n monomial for each n. Now, a p-family f is in* VBP_{nc} *if and only if $f \leq_{abp} g$.*

Another point about the definition of $\leqslant_{abp}$ is that the choice of the $(1, q(n))$-th entry of $g(\phi(Y_{p(n)}))$ is arbitrary. We could have chosen any specific entry of the matrix $g(\phi(Y_{p(n)}))$ or its trace or the sum of all entries. These would all yield equivalent definitions.

Remark 4.3.2. *An arithmetic circuit is* weakly skew *if for every multiplication gate g, there is at least one input f to g such that, either f is an input gate or removing the edge from f to g makes the underlying undirected graph disconnected. That is, the entire subcircuit rooted at f is used only to compute g; none of the gates in this subcircuit are reused elsewhere.*

Suppose $f \leqslant_{abp} g$ and g has polynomial-size weakly-skew circuits then we do not know if f has polynomial-size weakly skew circuits. In the noncommutative case we note that weakly-skew circuits are strictly more powerful than skew circuits. The polynomial family $PAL_n PAL_n$ has polynomial-size weakly skew circuits but skew-circuits require exponential size [LMS15]. In contrast, for the commutative case polynomial-size weakly-skew circuits are equivalent to polynomial-size skew circuits [Tod92].

4.3.4 Comparing the reducibilities

From the definition of the reducibilities it immediately follows that the $\leqslant_{abp}$ reduction is more powerful than $\leqslant_{iproj}$ reduction which is more powerful than $\leqslant_{proj}$ reduction. In

fact, it is not difficult to show that the $\leqslant_{abp}$ and the $\leqslant_{iproj}$ reductions are *strictly* more powerful than the $\leqslant_{iproj}$ and the $\leqslant_{proj}$ reductions respectively. We summarize these simple observations below.

Proposition 4.3.3. *There exist p-families $f = (f_n)$ and $g = (g_n)$ such that:*

1. *$f \leqslant_{iproj} g$ but $f \not\leqslant_{proj} g$.*
2. *$f \leqslant_{abp} g$ but $f \not\leqslant_{iproj} g$.*

Proof. For the first part define p-families $f_n \in \mathbb{F}\langle x_1, x_2, \ldots, x_n, y_1, \ldots, y_n\rangle$ and $g_n \in \mathbb{F}\langle z_0, z_1\rangle$ as $f_n = \prod_{i\in[n]}(x_i + y_i)$ and $g_n = \prod_{i\in[n]}(z_0 + z_1)$. Clearly, $f \leqslant_{iproj} g$ where the indexed projection will substitute x_i for z_0 and y_i for z_1 in the i-th linear factor $(z_0 + z_1)$ of g. However $f \not\leqslant_{proj} g$ as the usual $\leqslant_{proj}$ reduction cannot increase the number of variables.

For the second part define p-families $f_n, g_n \in \mathbb{F}\langle x, y\rangle$ as $f_n = x + y$ and $g_n = xy$ for every n. Clearly f is not $\leqslant_{iproj}$-reducible to g as indexed projections cannot increase the number of monomials in g. To see that $f \leqslant_{abp} g$ we define 2×2 substitution matrices: $M_x = \begin{bmatrix} 1 & x \\ 0 & 0 \end{bmatrix}$ and $M_y = \begin{bmatrix} 0 & y \\ 0 & 1 \end{bmatrix}$. Clearly, the $(1,2)^{th}$ entry of $g(M_x, M_y) = M_x M_y$ is $x + y$. Hence, $f \leqslant_{abp} g$. □

Remark 4.3.3. *A natural generalization of the projection reducibilities ($\leqslant_{proj}$ and $\leqslant_{iproj}$) is to consider indexed linear projections, denoted $\leqslant_{linproj}$. Namely, we allow for each variable occurring in a given position, substitution by either a scalar or a linear form. It is easy to see that $\leqslant_{linproj}$ is strictly more powerful than $\leqslant_{iproj}$. For example, in Proposition 4.3.3 we saw that $x + y$ is not $\leqslant_{iproj}$-reducible to xy. However, $x + y$ is trivially reducible by a linear projection: in the polynomial xy we can substitute 1 for y and $x + y$ for x.*

We can also show that there are p-families f and g such that $f \leqslant_{abp} g$ but $f \not\leqslant_{linproj} g$. To see this, define p-family $f_n \in \mathbb{F}\langle x, y, z, w\rangle$ and $g_n \in \mathbb{F}\langle w, x\rangle$ as $f_n = wx + yz$ and $g_n = wx$

for every n. Since f_n is irreducible for each n and g_n is reducible for each n, clearly f is not $\leqslant_{linproj}$-reducible to g. To see that $f \leqslant_{abp} g$ we define 2×2 substitution matrices: $M_w = \begin{bmatrix} w & y \\ 0 & 0 \end{bmatrix}$ *and* $M_x = \begin{bmatrix} 0 & x \\ 0 & z \end{bmatrix}$. *Clearly, the* $(1,2)^{th}$ *entry of* $g(M_w, M_x) = M_w M_x$ *is* $wx + yz$. *Hence,* $f \leqslant_{abp} g$.

It turns out that the proofs of some of our results shown for projections carry over to indexed linear projections but other results do not. We will return to this point at the end of Section 5.3.

4.3.5 Matrix substitutions and $\leqslant_{abp}$ reductions

We describe ideas from [AJS09] that are useful in connection with showing $\leqslant_{abp}$ reductions between p-families. Consider an ABP P computing a noncommutative polynomial $g \in \mathbb{F}\langle X \rangle$. Suppose the ABP P has q nodes with source s and and sink t.

For each variable $x \in X$ we define a $q \times q$ matrix M_x, whose $(i,j)^{th}$ entry $M_x(i,j)$ is the coefficient of variable x in the linear form labeling the directed edge (i,j) in the ABP P.[1]

Consider a degree d polynomial $f \in \mathbb{F}\langle X \rangle$, where $X = \{x_1, \cdots, x_n\}$. For each monomial $w = x_{j_1} \cdots x_{j_k}$ we define the corresponding matrix product $M_w = M_{x_{j_1}} \cdots M_{x_{j_k}}$. When each indeterminate $x \in X$ is substituted by the corresponding matrix M_x then the polynomial $f \in \mathbb{F}\langle X \rangle$ evaluates to the matrix

$$\sum_{f(w) \neq 0} f(w) M_w,$$

where $f(w)$ is the coefficient of monomial w in the polynomial f.

Theorem 4.3.4. [AJS09] *Let C be a noncommutative arithmetic circuit computing a polynomial* $f \in \mathbb{F}\langle x_1, x_2, \ldots, x_n \rangle$. *Let P be an ABP (with q nodes, source node s and sink node t) computing a polynomial* $g \in \mathbb{F}\langle x_1, x_2, \ldots, x_n \rangle$. *Then the* $(s,t)^{th}$ *entry of the matrix*

[1] If (i,j) is not an edge in the ABP then the coefficient of x is taken as 0.

$f(M_{x_1}, M_{x_2}, \ldots, M_{x_n})$ is the polynomial

$$\sum_w f(w)g(w)w.$$

where $f(w), g(w)$ are coefficients of monomial w in f and g respectively. Hence there is a circuit of size polynomial in n, size of C and size of P that computes the noncommutative polynomial $\sum_w f(w)g(w)w$.

Acyclic Automata and ABPs

Let P be a deterministic finite automaton with q states that accepts a finite language $W \subseteq X^d$. There is an equivalent automaton P' with $O(qd)$ states with the following properties: it has a start state labeled s and a unique final state labeled t. The state transition graph for P' is a layered directed acyclic graph with d layers, and each transition edge in P' is labeled by a variable from X.

Clearly, we can also interpret P' as an ABP, and the polynomial g that it computes is the sum of all monomials that are accepted by P. I.e.

$$g = \sum_{w \in W} w.$$

Corollary 4.3.5. *Suppose $f \in \mathbb{F}\langle X \rangle$ is a homogeneous degree d polynomial computed by a noncommutative circuit C and $W \subseteq X^d$ has a finite automaton P. Then the polynomial $\sum_{w \in W} f(w)w$ can be computed by a noncommutative circuit whose size is polynomially bounded in d, size of C and the size of the automaton P.*

The above corollary follows directly from Theorem 4.3.4.

It is useful to combine the construction described in the previous remark with *substitution maps*. For this purpose we consider *substitution automata*. A finite *substitution automaton* is a finite automaton P along with a substitution map $\psi : Q \times X \to Q \times Y \cup \mathbb{F}$, where

Q is the set of states of P, and Y is a set of (noncommuting) variables. If $\psi(i, x) = (j, u)$ it means that when the automaton P in state i reads variable x it replaces x by $u \in Y \cup \mathbb{F}$ and makes a transition to state $j \in Q$.

Now, for each $x \in X$ we can define the matrix M'_x as follows:

$$M'_x(i, j) = u, 1 \leq i, j \leq q, \text{ where } \psi(i, x) = (j, u).$$

For every monomial $w = x_{j_1} x_{j_2} \dots x_{j_d}$ accepted by P, there is a unique s-to-t path $\gamma = (s, i_1), (i_1, i_2), \dots, (i_{d-1}, t)$ along which it accepts. This defines the substitution map ψ extended to monomials accepted by P as

$$\psi(w) = \psi_{s,i_1}(x_{j_1})\psi_{i_1,i_2}(x_{j_2}) \dots \psi_{i_{d-1},t}(x_{j_d}),$$

so that $\psi(w) \in Y^*$. It follows that the $(s, t)^{th}$ entry of matrix $f(M'_{x_1}, M'_{x_2}, \dots, M'_{x_n})$ is the polynomial

$$\sum_{w \in W} f(w)\psi(w).$$

Corollary 4.3.6. *Suppose $f = (f_n)$ is a p-family computed by a circuit family $(C_n)_{n>0}$, where $f_n \in \mathbb{F}\langle X_n \rangle$ is a homogeneous degree $d(n)$ polynomial for each n. Suppose P_n is a polynomial (in n) size substitution automaton accepting a subset $W_n \subseteq X_n^{d(n)}$ with substitution map ψ_n for each n. Then the polynomial family $g = (g_n)$, where*

$$g_n = \sum_{w \in W_n} f_n(w)\psi_n(w),$$

is $\leqslant_{abp}$ reducible to f. In particular, it follows that g_n has a noncommutative circuit whose size is polynomial in $d(n)$ and the sizes of C_n and P_n.

The above corollary follows directly from the definition of a substitution automaton, Theorem 4.3.4 and Corollary 4.3.5).

4.4 Dyck Polynomials are VP_{nc}-complete

In this section we exhibit a *natural* p-family which is complete for the complexity class VP_{nc} under $\leqslant_{abp}$ reductions. We show that the p-family D_k (defined in Section 4.2.1), consisting of Dyck polynomials over k different types of brackets, is VP_{nc}-complete under $\leqslant_{abp}$ reductions. This result can be understood as an arithmetic analogue of the Chomsky-Schützenberger representation theorem [CS63] (also see [DSW94, pg. 306]), which says that every context-free language is a homomorphic image of intersection of a language of balanced parenthesis strings over suitable number of different types of parentheses and a regular language.

The overall idea is as follows: Given a VP_{nc} p-family f, in order to show $f \leqslant_{abp} D_k$ we first construct a deterministic finite automaton which filters out suitable monomials from monomials of D_k. Then we do appropriate scalar substitutions for certain suitably chosen variables in D_k in order to obtain the polynomial f. These two steps together can be seen as doing matrix substitutions for variables in D_k. Here the matrices are polynomial-sized, and are defined using the required scalar substitutions and the transition function of the automaton, as explained in Section 4.3.5.

Theorem 4.4.1 (Chomsky-Schützenberger)**.** *A language L over alphabet Σ is context free iff there exist*

1. *a matched alphabet $P \cup \overline{P}$ (P is set of k different types of opening parentheses $P = \{(_1, (_2, \ldots, (_k\}$ and $\overline{P}$ is the corresponding set of matched closing parentheses $\overline{P} = \{)_1,)_2, \ldots,)_k\}$),*

2. *a regular language R over $P \cup \overline{P}$,*

3. *and a homomorphism $h : (P \cup \overline{P})^* \mapsto \Sigma^*$*

such that $L = h(D \cap R)$, where D is the set of all balanced parentheses strings over $P \cup \overline{P}$.

Recall that the p-family $D_k = \{D_{k,d}\}_{d\geqslant 0}$ is defined over $2k$ distinct variables $X_k = \{(_i,)_i | 1 \leqslant i \leqslant k\}$, where $(_i$ and $)_i$ are matching parenthesis pairs.

$$D_{k,d} = \sum_{m \in W_{k,d}} m,$$

where $W_{k,d}$ is all degree-d well-balanced parenthesis strings over X_k (defined in Section 4.2.1).

Theorem 4.4.2. *The p-family $D_k = \{D_{k,d}\}_{d\geqslant 0}$ is VP_{nc}-complete under $\leqslant_{abp}$ reductions for $k \geq 2$.*

Proof. Let $f = (f_n)_{n>0}$ be a p-family in VP_{nc} and $\{C_n\}_{n\geqslant 0}$ be a circuit family such that C_n computes polynomial $f_n \in \mathbb{F}\langle X_n\rangle$ for each n. Let $s(n)$ and $d(n)$ be polynomials bounding the size and syntactic degree of circuit C_n, respectively. We do the following preprocessing on C_n.

- Suppose g is a gate in C_n with input gates g_1 and g_2. If the subcircuit rooted at either g_1 or g_2 consists only of scalars at the input level then we replace this subcircuit by the actual scalar value computed by the subcircuit. We perform this preprocessing for the entire circuit.

We can assume without loss of generality that the above preprocessing is already done on C_n for each n. For each n we will construct a collection of $2t(n)$ many matrices $M_1, M'_1, \ldots, M_{t(n)}, M'_{t(n)}$ whose entries are either field elements or monomials in variables $\{x_1, \ldots, x_n\}$ for a suitably chosen polynomial bound $t(n)$. These matrices will have the following property: consider the Dyck polynomial $D_{t(n),q(n)}$, where $q(n)$ is a polynomial to be suitably chosen later in the proof. When we substitute M_i for variable $(_i$ and M'_i for variable $)_i$ in $D_{t(n),q(n)}$, it will evaluate to a matrix $M = D_{t,q}(M_1, M'_1, \ldots, M_{t(n)}, M'_{t(n)})$ whose top right corner entry is precisely the polynomial f_n computed by C_n.

The idea underlying the construction is from the proof of the Chomsky-Schützenberger theorem. Our proof is essentially an arithmetic version. We need to additionally take care of coefficients of monomials and polynomial size bounds. The matrices $M_1, M_1', \ldots, M_t, M_t'$ correspond to the transitions of a deterministic finite state substitution automaton, in the sense explained in Section 4.3.5. The substitution automaton will be designed to transform monomials of $D_{t(n),q(n)}$ into monomials of C_n so that M's top right entry (corresponding to the accept state of the automaton) contains the polynomial C_n. We now give a structured description of the reduction.

1. Firstly, we do not directly work with the circuit C_n because we need to introduce a parsing structure to the monomials of C_n. We also need to make the circuit initially constant-free by introducing new variables. We will substitute back the constants for the new variables in the matrices. To this end, we will carry out the following modifications to the circuit C_n:

   (a) For each product gate $f = gh$ in the circuit, we convert it to the product gate computing $f = ({}_f g)_f h$, where ${}_f$ and $)_f$ are new variables.

   (b) We replace each input constant a of the circuit C_n by a degree-3 monomial ${}_a z_a)_a$, where ${}_a,)_a, z_a$ are new variables.

   Let C_n' denote the resulting arithmetic circuit after the above transformations applied to the gates. The new circuit C_n' computes a polynomial in the ring $\mathbb{F}\langle X_n' \rangle$, where

$$\begin{aligned} X_n' &= X_n \cup \{({}_g,)_g \mid g \text{ is a } \times \text{ gate in } C_n\} \\ &\cup \{({}_a,)_a \mid a \text{ is a constant in } C_n\} \\ &\cup \{z_a \mid a \text{ is a constant in } C_n\}. \end{aligned}$$

   We make a further substitution: we replace every variable $y \in X_n$ by the degree-2

monomial $[_y]_y$ and every variable z_a for constants a appearing in C_n by $[_{z_a}]_{z_a}$. Let the resulting arithmetic circuit be C''_n and the expanded variable set be denoted X''_n.

It is clear that the resulting family of circuits $(C''_n)_{n>0}$ computes a p-family $f'' = (f''_n)_{n>0}$, where $f''_n \in \mathbb{F}\langle X''_n\rangle$ is the polynomial computed by C''_n. Furthermore, by construction, C''_n is a polynomial whose monomials are certain properly balanced parenthesis strings over the parentheses set defined above. The circuit C''_n is not homogeneous. Clearly, its degree is bounded by a polynomial in $(s(n) + d(n))$. A *multiplicative subcircuit* of C''_n is defined by the following procedure starting at the output gate of C''_n: At each + gate retain exactly one of its input gates. At each $\times$ gate retain both input gates. In general, each multiplicative subcircuit computes a monomial with some coefficient. In the case of C''_n notice that distinct multiplicative subcircuits compute distinct monomials (guaranteed because of the new gate variables introduced). Furthermore, as C''_n is constant-free, the coefficients of all monomials is 1. We have the following simple claim.

Claim 4.4.1. $f \leqslant_{proj} f''$.

The above claim follows because we can recover the circuit C_n from C''_n by substituting 1 for the parenthesis variables $(_g,)_g$ occurring in C''_n for each gate g of C_n. Then substituting variable y for the term $[_y]_y$ in C''_n, and substituting the scalar a for $[_{z_a}]_{z_a}$ in C''_n.

f'' is $\leqslant_{abp}$ reducible to D_k

This is the main part of the proof. We describe the reduction in two steps. We first show that f'' is $\leqslant_{abp}$ reducible to the p-family $\hat{D} = (D_{t(n),q(n)})_{n>0}$. Here $t(n)$ is a polynomial bounding the number of parenthesis types used in C''_n along with some additional parentheses types. The polynomial bound $q(n)$ will be specified below. We then show that $\hat{D} \leqslant_{abp} D_k$ for any $k \geq 2$.

Let the syntactic degree of polynomial C''_n be $2r$. By construction, all nonzero monomials in the polynomial computed by C''_n are of even degree bounded by $2r$. We introduce $r+1$ *new* parenthesis types $\{_j, \}_j$, $0 \leqslant j \leqslant r$ (to be used as prefix padding in order to get homogeneity). Now, consider the polynomial $D_{t(n),q(n)}$ where $q(n) = 2r+2$ and $t(n) = (r+1) + p_n$, where p_n is the number of parenthesis types occurring in C''_n.

The reduction will map all degree $2j$ monomials in C''_n to prefix-padded monomials in $D_{t,q}$ of the form $m' = \{_1\}_1\{_2\}_2 \ldots \{_{r-j}\}_{r-j}\{_0\}_0 m$, where m is a degree $2j$ monomial over the parentheses types of C''_n. As a consequence m' is of degree $2r+2$ for all choices of j.

Now the matrices of the automaton have to effect substitutions in order to convert these m' into a monomial of C''_n of degree $2j$. The strings accepted by this automaton is of the form uv, where $u = \{_1\}_1\{_2\}_2 \ldots \{_{i-1}\}_{i-1}\{_0\}_0$, $0 \leqslant i \leqslant r+1$ and v is a well-balanced string over remaining parentheses types. This automaton is essentially based on the one defined in the proof of the Chomsky-Schützenberger theorem. We give its description below. The automaton runs only on monomials of $D_{t(n),q(n)}$ and hence can be seen as a layered acyclic DAG (as explained in Section 4.3.5) with exactly $q(n)$ layers.

(a) The start state of the automaton is $(\hat{s}, 0)$. The automaton first looks for prefix

$$\{_1\}_1\{_2\}_2 \ldots \{_{r-j}\}_{r-j}\{_0\}_0.$$

As it reads these variables, one by one, it steps through states $(\hat{s}, i)$, substitutes 1 for each of them, and reaches state $(s, 2(r-j+1))$ when it reads $\}_0$, where s is the name of the output gate of circuit C''_n. If any of the variables $\{_l, \}_l$, $l \in [r] \cup \{0\}$ occur later they are substituted by 0 (to kill such monomials).

(b) The automaton will substitute the substring $[_x]_x$ by variable x. If $[_x$ is not immediately followed by $]_x$ then it substitutes 0 for variable $[_x$ (to kill such

monomials). Similarly, the automaton substitutes $[_a]_a$ by a (if $[_a$ is not immediately followed by $]_a$ then it substitutes 0 for $[_a$).

(c) Now, we describe the crucial transitions of the automaton continuing from state $(s, 2(r-j+1))$, where s is the output gate of circuit C_n''. The transitions are defined using the structure of the circuit C_n''. At this point the automaton is looking for a degree $2j$ monomial. Let $\ell < 2r+2$.

In order to simplify our notation, we describe the transitions of the automaton on reading certain (short) monomials rather than symbol by symbol. The transition denoted

$$(h, \ell) \to m(g, \ell + |m|)$$

means that the automaton in state (h, ℓ) reads the monomial m and goes from state (h, ℓ) to state $(g, \ell + |m|)$, where $|m|$ is the degree of the monomial m.

The automaton is defined by the following transitions:

i. $(\hat{s}, 2j) \to \{_{j+1}\}_{j+1}(\hat{s}, 2(j+1))$, for $0 \leqslant j < r$.

ii. $(\hat{s}, 2(r-j)) \to \{_0\}_0(s, 2(r-j+1))$, for $0 \leqslant j \leqslant r$, where s is the output gate in the circuit C_n''.

iii. $(g, \ell) \to (_g(g_l, \ell+1)$, where g is an internal product gate in circuit C_n'' and g_l is its left child.

iv. Include the transition $(g, \ell) \to (_h(h_l, \ell+1)$, if g is an internal + gate in circuit C_n'', h is an internal product gate such that there is a directed path of + gates from h to g. As before, h_l denotes the left child of h.

v. For each input variable, say z, in the circuit C_n'' and for each product gate g in the circuit C_n'', the automaton includes the transition $(h, \ell) \to [_z]_z)_g(g_r, \ell+3)$, if $\ell + 3 < 2r+2$, where g_r is the right child of the internal product gate g, and h stands for any internal gate in C_n''. If $\ell + 3 = 2r+2$ then the automaton instead includes the transition $(h, \ell) \to [_z]_z)_g(t, 2r+2)$, where $(t, 2r+2)$ is the unique accepting state of the automaton.

This completes the definition of the automaton. The important property about monomials accepted by the automaton is summarized in the following claim.

Claim 4.4.2. *The above automaton accepts only strings from* $(X''_n)^{q(n)}$*, where* $q(n) = 2r(n)+2$*. Furthermore, if the input to the automaton is a monomial* m' *of* $D_{t(n),q(n)}$ *then the automaton accepts* m' *iff* $m' = \{_1\}_1\{_2\}_2 \ldots \{_{r-j}\}_{r-j}\{_0\}_0 m$ *for some* j*, where* m *is a nonzero degree* $2j$ *monomial of* f''_n*.*

The claim follows directly from the automaton's construction. Notice that the automaton could accept some arbitrary monomials in $(X''_n)^{q(n)}$ which are not monomials of $D_{t(n),q(n)}$. However, that is not a problem for our reduction.

Now that we have specified the transitions of the automaton, we can define the substitution automaton, by describing the matrices that we substitute for each parenthesis. We define U as the following subset of X''_n:

$$\begin{aligned} U &= \{[_z \mid z \text{ is a variable in } C''_n\} \\ &\bigcup \{(_g,)_g \mid g \in \mathcal{G}\} \\ &\bigcup \{\{_j, \}_j \mid j \in [r] \cup \{0\}\} \end{aligned}$$

where $\mathcal{G}$ denotes the set of all product gates in the circuit C_n. Let M_v be the matrix we substitute for variable $v \in U$. The rows and columns of matrix M_v are labeled by the states of the automaton. Matrix M_v is defined as follows:

$$m_{i,j} = M_v[i,j] = \begin{cases} 1 & \text{if } v \in U \text{ and } (i,j) \text{ is a transition labeled } v \\ z & \text{if } p =]_z \text{ and } (i,j) \text{ is a transition labeled } v \end{cases}$$

where z denotes a variable in the circuit C''_n.

From the above construction and by Corollary 4.3.6, it follows that upon substituting these matrices for the variables in the polynomial $D_{t(n),q(n)}$ the top right corner entry of the resulting matrix is the polynomial computed by the circuit C''_n. Therefore, $f'' \leqslant_{abp} \hat{D}$.

We now complete the proof by showing that $\hat{D} \leqslant_{abp} D_k$ for any $k \geq 2$.

Claim 4.4.3. *The p-family $\hat{D}$ is $\leqslant_{abp}$-reducible to D_2.*

Proof of Claim. Let $2^{p(n)-1} < t(n) \leqslant 2^{p(n)}$ for some $p(n) \in O(\log n)$. Consider the p-family $\hat{D}' = (D_{2^{p(n)},q(n)})_{n>0}$. Clearly, $\hat{D} \leqslant_{proj} \hat{D}'$, where the projection reduction will substitute 1 for variables $(_j,)_j$ when $t(n) < j \leq 2^{p(n)}$. Thus, it suffices to show $\hat{D}'$ is $\leqslant_{abp}$ reducible to D_2.

Let $\{[_0, [_1, \ldots, [_{2^{p(n)}-1}\} \cup \{]_0,]_1, \ldots,]_{2^{p(n)}-1}\}$ be the variable set for $D_{2^{p(n)},q(n)}$. For $0 \leqslant i \leqslant 2^{p(n)} - 1$ we will encode $[_i$, $]_i$ by strings $(_{b_0}(_{b_1} \ldots (_{b_{p(n)-1}}$ and $)_{b_0})_{b_1} \ldots)_{b_{p(n)-1}}$ respectively, where the tuple $\langle b_0, \ldots, b_{p(n)-1}\rangle$ is the binary encoding of index i. We can easily design a finite automaton which on input a degree $p(n)q(n)$ monomial m of $D_{2,p(n)q(n)}$, checks if m is a valid encoding of some monomial of $D_{2^{p(n)},q(n)}$. So by using the transition function of this automaton we can define appropriate matrix substitutions for the variables of $D_{2,p(n)q(n)}$ (namely the four variables $(_0, (_1,)_0,)_1$), so that the top right corner entry of the resulting matrix obtained after this substitution in $D_{2,p(n)q(n)}$ is the polynomial $D_{2^{p(n)},q(n)}$. This proves the claim.

Clearly every Dyck polynomial can be computed by a polynomial-size non-commutative arithmetic circuit so from Claim 4.4.3, it follows that D_2 is VP_{nc}-complete. As for any $r \geqslant 2$ we have $D_2 \leqslant_{abp} D_r$. Thus the Dyck polynomials D_r for all $r \geqslant 2$ are VP_{nc}-complete. This proves the theorem. □

Remark 4.4.1. *In the commutative setting, Valiant has shown [Val79a] that the determinant* DET *is* VP*-complete, but only under* quasipolynomial *projections. The problem of finding natural* VP*-complete p-families that are complete under p-projections is not yet satisfactorily settled in the commutative case. Perhaps one needs to consider a more*

flexible reducibility than projections. However, the $\leqslant_{abp}$ reducibility does not make sense for VP. *The commutative ABP model is very powerful:* DET *itself has polynomial-size commutative ABPs [Tod92].*

Remark 4.4.2. *We note that D_1 is not VP_{nc}-complete. Indeed, it is easy to see that each $D_{1,n}$ has a polynomial in n size ABP. Therefore, $D_1 \in \mathrm{VBP}_{nc}$. In fact, notice that $D_1 \leqslant_{abp}$ PAL $\leqslant_{abp} D_2$ and $D_2 \not\leqslant_{abp}$ PAL $\not\leqslant_{abp} D_1$. As* PAL *is not in VBP_{nc} [Nis91], it follows that* PAL $\not\leqslant_{abp} D_1$. *Since* $\mathrm{PAL}^2 = (\mathrm{PAL}_n\mathrm{PAL}_n)$ *is in VP_{nc} and does not have polynomial-size skew circuits [LMS15], it follows that D_2 is not $\leqslant_{abp}$-reducible to* PAL.

4.5 Palindrome Polynomials are VSKEW_{nc}-complete

In this section we show that the p-family PAL, consisting of palindrome polynomials (defined in Section 4.2.1), is complete for the class VSKEW_{nc} w.r.t. $\leqslant_{abp}$ reductions. The proof is broadly similar to that of Theorem 4.4.2.

Theorem 4.5.1. *The p-family* PAL *is VSKEW_{nc}-complete under $\leqslant_{abp}$ reductions.*

Proof. We show that any p-family in VSKEW_{nc} is $\leqslant_{abp}$-reducible to PAL.

Let $f = (f_n)_{n>0}$ be a p-family in VSKEW_{nc} and $\{C_n\}_{n\geqslant 0}$ be a skew circuit family computing f. Suppose $f_n \in \mathbb{F}\langle X_n\rangle$ for each n. Let $s(n)$ and $d(n)$ be polynomials bounding the size and syntactic degree of C_n, respectively.

We need to construct matrices corresponding to the transitions of a substitution automaton which will transform monomials of $\mathrm{PAL}_{t(n)}$, for a suitably large polynomial $t(n)$, into monomials of C_n. More precisely, after substitutions, the top right entry of the resulting matrix contains the polynomial f_n.

We will modify circuit C_n in order to introduce a parsing structure to the monomials it computes. We apply the following transformations to C_n:

1. For each left-skew product gate $g = xh$ in the circuit C_n where x is an input variable and h a gate in the circuit, let (h, g) be the directed edge in the circuit C_n from gate g to gate h. We convert the gate into the two skew gates

$$\begin{aligned} g' &= h\mathbf{x}_{(h,g,R)} \\ g'' &= \mathbf{x}_{(h,g,L)}g', \end{aligned}$$

   where $\mathbf{x}_{(h,g,L)}$ and $\mathbf{x}_{(h,g,R)}$ are fresh variables. Right-skew gates $g = hx$ are transformed analogously using fresh variables $x^{(r)}_{(h,g,L)}$ and $x^{(r)}_{(h,g,R)}$. Here we use superscripts ℓ and r for the variables to keep track of whether the original $\times$ gate was left skew or right skew.

2. For each product gate $g = ah$ in the circuit C_n, for some scalar $a \in \mathbb{F}$ we convert it to two skew gates

$$\begin{aligned} g' &= ha_{(h,g,R)} \\ g'' &= a_{(h,g,L)}g' \end{aligned}$$

   where $a_{(h,g,L)}$ and $a_{(h,g,R)}$ are again fresh variables.

Let C'_n denote the resulting skew circuit after transformation. It computes a polynomial f'_n in $\mathbb{F}\langle X'_n\rangle$ where the variable set X'_n is the collection of all the $a_{(h,g,L)}$, $a_{(h,g,R)}$, $\mathbf{x}_{(h,g,L)}$, $\mathbf{x}_{(h,g,R)}$, $x^{(r)}_{h,g,L}$, and $x^{(r)}_{h,g,R}$ defined above.

The transformation ensures that distinct multiplicative subcircuits of C'_n compute distinct monomials because all the new variables introduced carry the gate names. As C'_n is constant-free, the coefficients of all nonzero monomials in f'_n is 1.

Furthermore, the nonzero monomials in f'_n are all palindrome monomials in the variable set X'_n: in a monomial m of degree $2d$ occurring in f'_n, for all $i \in [d]$, variable $\mathbf{x}_{(h,g,L)}$ occurs at position i if and only if at position $2d - i + 1$ we have the matching variable $\mathbf{x}_{(h,g,R)}$. Similarly, for the variable pairs $a_{h,g,L}$ and $a_{h,g,R}$, as well as $x^{(r)}_{h,g,L}$ and $x^{(r)}_{h,g,R}$. This property is easy to check inductively by the transformation at all $\times$ gates in C'_n.

Clearly, $f \leqslant_{proj} f'$ because we can recover f_n from f'_n by the following substitutions:

- Variable x for $\mathbf{x}_{h,g,L}$ and 1 for $\mathbf{x}_{h,g,R}$.
- Variable x for $x^{(r)}_{h,g,R}$ and 1 for $x^{(r)}_{h,g,L}$.
- Scalar a for variable $a_{h,g,L}$ and 1 for $a_{h,g,R}$.

Clearly, the number of variables and degree of f'_n are polynomially bounded in n.

Let the degree of polynomial f'_n be $2r(n)$. The nonzero monomials computed by C'_n are of even degree bounded by $2r(n)$. We introduce $r(n) + 1$ *new* variable pairs $y_{j,L}, y_{j,R}$, $0 \leqslant j \leqslant r(n)$ (to be used as prefix and suffix padding in order to get homogeneity). For a degree $2j$ monomial m in f'_n define the palindrome monomial

$$m' = (y_{1,L}y_{2,L} \dots y_{r-j,L}y_{0,L})m(y_{0,R}y_{r-j,R} \dots y_{2,R}y_{1,R}).$$

Now, m' is of degree $2r(n) + 2$ for all choices of j. Let C''_n denote a new circuit obtained from C'_n as follows: From C'_n we find skew circuits for each of its homogeneous components. For the degree $2j$ homogeneous component we apply the appropriate prefix/suffix padding (of length $2r(n)+2-j$) as described above. We add the resulting circuits for the different homogeneous components to obtain C''_n. Let f''_n denote the polynomial computed by C''_n and $f'' = (f''_n)_{n>0}$ the corresponding p-family. Clearly, f''_n computes

a homogeneous degree $2r(n) + 2$ polynomial. All the monomials of f''_n are palindrome monomials over the variable pairs in the set $X''_n = \{y_{j,L}, y_{j,R} \mid 0 \leq j \leq r(n) + 1\} \cup X'_n$.

Let $\hat{\text{PAL}} = (\hat{\text{PAL}}_n)_{n>0}$ denote the p-family, where $\hat{\text{PAL}}_n$ consists of all degree $2r(n) + 2$ palindrome monomials over the variable set X''_n.

In the next steps, we will show that $f'' \leqslant_{abp} \hat{\text{PAL}}$ and $\hat{\text{PAL}} \leqslant_{abp} \text{PAL}$. As $f \leqslant_{proj} f' \leqslant_{proj} f''$, it will follow that $f \leqslant_{abp} \text{PAL}$, completing the proof.

f'' is $\leqslant_{abp}$-reducible to $\hat{\text{PAL}}$

The $\leqslant_{abp}$ reduction is effected by a substitution automaton which accepts precisely those palindrome monomials ww^R such that the first half w is "compatible" with the circuit structure of C''_n (although it also accepts many non-palindrome monomials). The transition matrices of the automaton, when substituted for variables in $\hat{\text{PAL}}_n$, ensure that only monomials of C''_n survive in the final polynomial obtained as the top right entry of the resulting matrix. The automaton is a layered DAG with exactly $2r + 2$ layers.

1. The start state of the automaton is $(\hat{s}, 0)$. The automaton first looks for a prefix $(y_{1,L}y_{2,L} \ldots y_{r-j,L}y_{0,L})$. These transitions can be described as $(\hat{s}, i) \rightarrow y_{(i+1,L)}(\hat{s}, i+1))$, for $0 \leqslant i < r$. As the automaton reads these variables it steps through states $(\hat{s}, i)$, substitutes 1 for each of them, and reaches state $(s, (r - j + 1))$ when it reads $y_{0,L}$, where s is the name of the output gate of circuit C''_n. If any of $y_{l,L}$, $l \in [r] \cup \{0\}$ occur later the automaton will substitute 0 for it (in order to kill that monomial).

2. Now we describe the transitions of the automaton continuing from state $(s, (r - j + 1))$. The automaton will use the circuit C''_n. At this point the automaton is looking for a degree $2j$ monomial. Let $\ell < 2r + 2$. The automaton has the following transitions:

   (a) $(\hat{s}, j) \rightarrow y_{(0,L)}(s, j + 1)$, where $0 \leqslant j \leqslant r$ and s is the output gate in the circuit

C''_n.

(b) In state $(s, j+1)$ if the automaton reads variable $\mathbf{x}_{h,g,L}$ (or $x^{(r)}_{h,g,L}$ or $a_{e,g,L}$) it moves to state $(g, j+2)$ if the gate g is a left-skew multiplication occurring in the circuit C''_n, and the directed path from g to s in the circuit has only + gates or right-skew multiplication gates in it. Formally, the transitions made are:

$$\begin{aligned}(s, j+1) &\rightarrow \mathbf{x}_{(h,g,L)}(g, j+2),\\ (s, j+1) &\rightarrow x^{(r)}_{(h,g,L)}(g, j+2),\\ (s, j+1) &\rightarrow a_{(h,g,L)}(g, j+2).\end{aligned}$$

(c) In general, when the automaton is in state (g, ℓ) for a left-skew multiplication gate g in the circuit and it reads variable $\mathbf{x}_{g_1,g_2,L}$ ($x^{(r)}_{g_1,g_2,L}$ or $a_{g_1,g_2,L}$) then it moves to state $(g_2, \ell+1)$ if the gate g_2 is left-skew occurring in the circuit, and the directed path from g_2 to g has only + gates or right-skew multiplication gates in it. The transitions are:

$$\begin{aligned}(g, \ell) &\rightarrow \mathbf{x}_{(g_1,g_2,L)}(g_2, \ell+1),\\ (g, \ell) &\rightarrow x^{(r)}_{(g_1,g_2,L)}(g_2, \ell+1),\\ (g, \ell) &\rightarrow a_{(g_1,g_2,L)}(g_2, \ell+1).\end{aligned}$$

(d) After the automaton reaches a state $(g, r+1)$ for some left-skew multiplication gate g it makes only transitions of the form:

$$\begin{aligned}(g, \ell) &\rightarrow z(t, \ell+1),\\ (t, \ell) &\rightarrow z(t, \ell+1),\end{aligned}$$

for all choices of $z \in \{a_{(h,g,R)}, \mathbf{x}_{(h,g,R)}, x^{(r)}_{h,g,R} \mid g \text{ and } h \text{ gates in } C_n\}$, and for $r+1 \leq \ell < 2r+2$. The state $(t, 2r+2)$ is the unique accepting state of the automaton.

Transitions (a)-(d) ensures that the automaton accepts a monomial $ww' \in (X''_n)^{2r(n)+2}$, where $|w| = r(n) + 1$, if and only if ww^R is a nonzero monomial in the polynomial f''_n computed by C''_n. Thus, the transitions in (a)-(d) ensure the following claim.

Claim 4.5.1. *The automaton defined above accepts a palindrome monomial $ww^R \in (X''_n)^{2r(n)+2}$ iff ww^R is a nonzero monomial in f''_n.*

Note that there may be many other monomials ww' also accepted by the automaton. However, that does not affect the reduction.

We can now apply Corollary 4.3.6. Let $W \subset (X''_n)^{2r(n)+2}$ denote the set of strings accepted by the above automaton, and ψ denote its substitution map (which replaces the variables $y_{j,L}$ and $y_{j,R}$ by 1 and leaves other variables unchanged). Then Corollary 4.3.6 implies that $f'' \leqslant_{abp}$ PÂL.

PÂL is $\leqslant_{abp}$-reducible to PAL

Finally, we note that PÂL is $\leqslant_{abp}$-reducible to PAL. The polynomial PÂL_n consists of palindromes over a large (polynomial size) variable set X''_n. We can transform each such palindrome into a palindrome over two variables $\{x_0, x_1\}$ with simple encoding: we can substitute $y_{j,L}$ and $y_{j,R}$ by $x_0 x_1^j x_0$ for each j. Similarly, for $\mathbf{x}_{h,g,L}$ and $\mathbf{x}_{h,g,R}$ we use a distinct integer k and encode them both as $x_0 x_1^k x_0$. Likewise, we encode each variable pair $x^{(r)}_{h,g,L}$ and $x^{(r)}_{h,g,R}$, or $a_{h,g,L}$ and $a_{h,g,R}$ as $x_0 x_1^k x_0$ for distinct choices of integer k. We will need only integers $k \leqslant |X''_n|$ which is polynomially bounded. Furthermore, this encoding is invertible and can be implemented by a polynomial-size substitution automaton. It now follows from Corollary 4.3.6 that PÂL $\leqslant_{abp}$ PAL. □

4.6 Concluding remarks and open problems

We have shown that Dyck polynomials are VP_{nc}-complete under $\leqslant_{abp}$ reductions. Finding natural VP_{nc}-complete p-families under $\leqslant_{iproj}$ reductions appears to be a challenging problem, given that finding natural VP-complete p-families under projections does not have a satisfactory answer yet. In the commutative case, it would be nice to show that DET is VP-complete under a more general reducibility (projections are probably too restricted).

Chapter 5

Structure inside the classes VP_{nc} and VNP_{nc}

5.1 Introduction

In this chapter we continue our study of the noncommutative analogues, VP_{nc} and VNP_{nc}, of Valiant's algebraic complexity classes. Our main results are the following:

- Assuming $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$, we exhibit a strictly infinite hierarchy of p-families, with respect to the projection reducibility, between the complexity classes VP_{nc} and VNP_{nc} This is analogous to the well-known Ladner's theorem [Lad75] that shows, assuming P $\neq$ NP, that there is an infinite hierarchy of polynomial-time many-one degrees between P and NP-complete. For commutative Valiant's classes, the existence of VNP-intermediate p-families is investigated by Bürgisser [Bür99].

- Inside VP_{nc} too we show there is a strict hierarchy of p-families (based on the nesting depth of Dyck polynomials) with respect to the $\leq_{abp}$-reducibility (defined in the Chapter 4).

5.2 Summary of main results

In this chapter, we study the structure of the classes VP_{nc} and VNP_{nc}. Our main results show that there is a rich structure within the classes VNP_{nc} and VP_{nc}.

To state our results, we need the following definition. We define the p-family $ID = (ID_n)$ which corresponds to the familiar context-sensitive language $\{ww \mid w \in \Sigma^*\}$.

$$ID_n = \sum_{w \in \{x_0, x_1\}^n} ww.$$

We call the polynomial ID_n as identity polynomial.

Now we state our main results of this chapter.

1. We prove a transfer theorem which essentially shows that if f is a VNP_{nc}-complete p-family under projections then an appropriately defined commutative version $f^{(c)}$ of f is complete under projections for the commutative VNP class. This is in Section 5.3.

2. Hrubes et al [HWY10a] have shown, assuming the sum-of-squares conjecture, that the p-family $ID = (ID_n)$, where $ID_n = \sum_{w \in \{x_0, x_1\}^n} w.w$ is not in VP_{nc}. Based on the p-family ID, we define a p-family ID^* and show, assuming $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$, that ID^* is neither in VP_{nc} nor VNP_{nc}-complete under projections. This is analogous to Ladner's well-known theorem [Lad75]. Such an analogue of Ladner's theorem for commutative Valiant classes VP and VNP is already shown by Bürgisser [Bür99]. These results are also in Section 5.3.

3. Within VP_{nc} we obtain a proper hierarchy w.r.t $\leqslant_{abp}$-reductions corresponding to the Dyck polynomials of bounded nesting depth. This roughly corresponds to the noncommutative VNC hierarchy within VP_{nc}. These results are in Section 5.5.

The following table summarizes the results in this chapter.

P-family	Complexity Result	Remarks
ID_d	-not VNP_{nc}-Complete (Theorem 5.3.6) -not in VP_{nc} [HWY10a]	$\leqslant_{proj}, \leqslant_{iproj}$-reductions assuming SOS_k conjecture
$f^{(i+1)}, i \geqslant 1$	VNP_{nc}-intermediate (Theorem 5.3.12)	$\leqslant_{proj}, \leqslant_{iproj}$-reductions - not reducible to $f^{(i)}$ assuming $\text{VP}_{nc} \neq \text{VNP}_{nc}$
$\text{PER}^{*,\chi}$	VNP_{nc}-Complete (Theorem 5.4.3)	w.r.t. $\leq_{abp}$-reductions
ID_n^*	VNP_{nc}-Complete (Theorem 5.3.7)	w.r.t. $\leq_{abp}$-reductions

Table 5.1: Summary of Results

5.3 A Ladner's Theorem analogue for VNP_{nc}

In this section we explore the class VNP_{nc} assuming $\text{VP}_{nc} \neq \text{VNP}_{nc}$. We exhibit an explicit p-family in $\text{VNP}_{nc} \setminus \text{VP}_{nc}$ that is not VNP_{nc}-complete. Based on this p-family we construct *strictly infinite* hierarchy of p-families under indexed projections between VP_{nc} and VNP_{nc}. This is similar in spirit to the well-known Ladner's Theorem [Lad75] that shows, assuming P $\neq$ NP, that there is an infinite hierarchy of polynomial-time many-one degrees between P and NP-complete. For commutative Valiant's classes, the existence of VNP-intermediate p-families is investigated by Bürgisser [Bür99].

Bürgisser [Bür99] has shown, under Valiant's hypothesis, that any countable poset can be embedded inside the poset of p-families in $VNP \setminus VP$ under c-reductions (a notion of reduction between p-families defined there). We call a p-family f is a c-reduction of p-family g, denoted by $f \leqslant_c g$, iff the map $L^{g_{t(n)}}(f_n)$ is polynomial bounded where $t(n)$ is polynomial bounded function and $L^{g_{t(n)}}(f_n)$ is, called oracle complexity of f_n with respect to the oracle polynomial $g_{t(n)}$, the minimum number of arithmetic operations $(+, \times)$ and evaluations of $g_{t(n)}$ (at previously computed values) that are sufficient to compute f_n from the input variables and constants from a field.

In particular, Bürgisser [Bür99] result implies the existence of an infinite hierarchy of

VNP-intermediate p-families w.r.t. c-reductions. We note that these VNP-intermediate families in [Bür99] are constructed using diagonalization. Bürgisser also shows a natural and explicit VNP-intermediate p-family, but the proof of intermediateness for that family requires an additional hardness assumption about counting classes in the boolean setting. In the noncommutative setting, we give an infinite hierarchy of explicit and natural VNP_{nc}-intermediate p-families assuming $\mathrm{VNP}_{nc} \neq \mathrm{VP}_{nc}$.

Definition 5.3.1 (VNP_{nc}-intermediate)**.** *We say that a noncommutative p-family $f = (f_n)_{n\geqslant 0}$ is VNP_{nc}-intermediate if $f \notin \mathrm{VP}_{nc}$ and f is not VNP_{nc}-complete w.r.t. $\leqslant_{iproj}$ reductions.*

For any set of noncommuting variables X with $|X| \geq 2$, we define the p-family $ID = (ID_n)$, where $ID_n = \sum_{w\in X^n} ww$. As the monomials of ID_n can be recognized and their coefficients computed in polynomial time the p-family ID is in VNP_{nc} [HWY10b].

We first show that ID is not VNP_{nc}-complete under $\leqslant_{iproj}$ reductions. We prove it unconditionally using a simple "transfer" theorem, which allows us to transfer a VNP_{nc}-complete p-family w.r.t $\leqslant_{iproj}$ reductions to a commutative VNP-complete p-family w.r.t $\leqslant_{proj}$ reductions.

Definition 5.3.2. *Let $f = (f_n)$ be a p-family in VNP_{nc}, where each f_n is a polynomial of degree $d(n)$. We define the* commutative version *$f^{(c)} = (f_n^{(c)})$ as follows: Suppose $f_n \in \mathbb{F}\langle X_n\rangle$. Let $Y_n = \bigcup_{1\leqslant i\leqslant d(n)} X_{n,i}$ be a new variable set, where $X_{n,i} = \{x_{ji} | \forall x_j \in X_n\}$ is a copy of the variable set X_n for the i^{th} position. If the polynomial $f_n = \sum \alpha_m m$ where $\alpha_m \in \mathbb{F}$ and $m \in X_n^{\leq d(n)}$ is a monomial, the polynomial $f_n^{(c)}$ is defined as $f_n^{(c)} = \sum \alpha_m m'$, where if $m = x_{j_1}x_{j_2}\dots x_{j_q}$ then $m' = x_{j_1,1}x_{j_2,2}\dots x_{j_q,q}$.*

Clearly, $f_n^{(c)} \in \mathbb{F}[X]$ and is a polynomial of degree $d(n)$.

Lemma 5.3.3. *For any p-families f and g (in $\mathbb{F}\langle X\rangle$), if $f \leqslant_{iproj} g$ then $f^{(c)} \leqslant_{proj} g^{(c)}$.*

Proof. Since $f \leqslant_{iproj} g$, for every n there is a polynomial $p(n)$ and an indexed projection $\phi_n : [d_{p(n)}] \times X_{p(n)} \to (Y_{ij})_{1\leqslant i,j\leqslant n}$ s.t. $f_n(Y_n) = g(\phi_n(X_{p(n)}))$ where $d_{p(n)}$ is the degree of the

polynomial $g_{p(n)}$. Define $\phi'_n : \bigcup_{i\in[d(n)]} X_{p(n),i} \to Y_n$ as $\phi'_n(x_{ji}) = \phi_n(i, x_j)$ for $1 \leqslant i, j \leqslant n$. Clearly, $f^{(c)}$ is reducible to $g^{(c)}$ via this projection reduction. This completes the proof. □

The following observation is an easy consequence of Lemma 5.3.3.

Theorem 5.3.4 (Transfer theorem)**.** *Let $f = (f_n) \in \mathrm{VNP}_{nc}$ be a p-family that is VNP_{nc}-complete under $\leqslant_{iproj}$-reductions. Then $f^{(c)}$ is* VNP*-complete under $\leqslant_{proj}$-reductions.*

Proof. Since f is VNP_{nc}-complete and $\mathrm{PER} \in \mathrm{VNP}_{nc}$ we have $\mathrm{PER} \leqslant_{iproj} f$. It follows from Lemma 5.3.3 that $\mathrm{PER}_d^{(c)} \leqslant_{proj} f^{(c)}$, which means that $f^{(c)}$ is VNP-complete under $\leqslant_{proj}$-reductions. □

Corollary 5.3.5. *If* VP $\neq$ VNP *then the noncommutative determinant* $\mathrm{DET} = (\mathrm{DET}_n)$ *is* VNP_{nc}*-intermediate.*

Proof. If the noncommutative determinant $\mathrm{DET} = (\mathrm{DET}_n)$ is VNP_{nc}-complete under $\leqslant_{iproj}$ reductions then, by Theorem 5.3.4, DET is VNP-complete under $\leqslant_{proj}$-reductions. However, DET is in VP, which contradicts VP $\neq$ VNP. □

Remark 5.3.1. *We note here that* VP $\neq$ VNP *is a* stronger *assumption as it implies* $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$*. However, in this section we show existence of* VNP_{nc}*-intermediate polynomials under the* weaker *assumption that* $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$*.*

We first note that the p-family ID is not VNP_{nc}-complete under $\leqslant_{iproj}$ reductions.

Theorem 5.3.6. *The p-family ID is not VNP_{nc}-complete under $\leqslant_{iproj}$-reductions.*

Proof. Consider $ID = (ID_n)$ with ID_n defined over variable set $X = \{x_1, x_2, \ldots, x_{m(n)}\}$. Then then commutative polynomial $ID_n^{(c)}$ is

$$ID_n^{(c)} = \prod_{j=1}^{n} \Big(\sum_{i=1}^{m(n)} x_{i,j}x_{i,n+j}\Big).$$

All irreducible factors of $ID_n^{(c)}$ have degree 2. If g is a p-family such that $g \leqslant_{iproj} ID$, then by Lemma 5.3.3 we have $g^{(c)} \leqslant_{proj} ID^{(c)}$. As $g^{(c)}$ is obtained by projection from $ID_n^{(c)}$ (for some n), it follows that all irreducible factors of $g^{(c)}$ also have degree at most 2. Now, define the p-family $g = (g_n)$, where $g_n = x_1x_2x_3 + x_4x_5x_6$ for all n. Clearly, $g \in \text{VNP}_{nc}$ and $g_n^{(c)}$ is irreducible of degree 3. Therefore, g is not $\leqslant_{iproj}$-reducible to ID. □

Thus, the p-family ID is not VNP_{nc}-complete w.r.t. $\leq_{iproj}$ reductions unconditionally. As ID is not known to be in VP_{nc}, that makes it a candidate for being VNP_{nc}-intermediate. If we could show that ID is VNP_{nc}-complete w.r.t. $\leq_{abp}$ reductions it would follow that ID is not in VP_{nc} assuming $\text{VP}_{nc} \neq \text{VNP}_{nc}$. Motivated by this observation, we consider a generalized version of ID which we call ID^* which turns out to be VNP_{nc}-complete under $\leqslant_{abp}$ reductions but not VNP_{nc}-complete under $\leqslant_{iproj}$ reductions.

For each positive integer n, let X_n be a variable set such that $|X_n| = n^2$. Let W_n denote the set of all degree n monomials over X_n and define the polynomial

$$ID_n^* = \sum_{w \in W_n} \underbrace{ww \dots w}_{n^2-times}.$$

Clearly, the p-family $ID^* = (ID_n^*)$ is in VNP_{nc} as we can recognize the monomials of ID_n^* in time polynomial in n for each n. We show the following completeness result for ID^*.

Theorem 5.3.7. *The p-family ID^* is VNP_{nc}-complete under $\leqslant_{abp}$ reductions.*

Proof. Consider the permanent polynomial PER_n and the ID_n^* polynomials, both defined on the variable set $V_n = \{x_{ij} \mid 1 \leq i, j \leq n\}$.

We design a deterministic finite state automaton A with the following properties:

1. The automaton A takes as input strings of length n^3 over alphabet V_n. We can write each such string as $w_1w_2 \dots w_{n^2}$, where each w_i is of length n.

2. It checks that each w_i is a monomial of the form $w = X_{1i_1} \dots X_{ni_n}$. I.e. the automaton checks that the first index of the variables in monomial w_i is strictly increasing from 1 to n.

3. For the i^{th} block w_i, since $1 \leq i \leq n^2$, we can consider the index i as a pair (j,k), $1 \leq j,k \leq n$. While reading the i^{th} block $w_i = X_{1i_1} \dots X_{ni_n}$ the automaton checks that $i_j \neq i_k$ if $j \neq k$.

The automaton A can be easily realized as a DAG with n^3 layers. The first layer has the start state s and the last layer has one accepting state t and one rejecting state t'. The transitions of automaton A are only between adjacent layers of this DAG. We group the adjacent layers of this DAG into blocks of size n. Let these layer blocks be denoted $B_1, B_2, \dots, B_{n^2}$. In block B_i, the transitions of the automaton will check if $i_j \neq i_k$ holds in w_i assuming $j \neq k$, where $i = (j,k)$ and the entire input is $w_1, w_2 \dots w_{n^2}$. The automaton will have the indices j and k hardwired in the states corresponding to block B_i and can easily check this condition. If for any block B_i, the indices $i_j = i_k$ then the automaton stores this information in its state and in the end makes a transition to the rejecting state t'.

Finally, the matrices of the automaton have to effect substitutions in order to convert monomials of ID_n into monomials of PER_n. The matrices will replace x_{ij} by the same variable x_{ij} in the first block B_1 and by 1 in all subsequent blocks. The polynomial ID_n^* when evaluated on these matrices will have the permanent polynomial PER_n in the $(s,t)^{th}$ entry of the resulting matrix. This completes the proof of the theorem. □

Theorem 5.3.8. *Assuming* $\text{VP}_{nc} \neq \text{VNP}_{nc}$*, the p-family* ID^* *is* VNP_{nc}*-intermediate under* $\leqslant_{iproj}$ *projections.*

Proof. If ID^* is VNP_{nc}-complete under $\leqslant_{iproj}$ reductions, then $\text{PER} \leqslant_{iproj} ID^*$ where

$d \leqslant p(n)$ for a polynomial p. By Theorem 5.3.4 it follows that $\mathrm{PER}^{(c)} \leqslant_{proj} ID^{*(c)}$. Now,

$$ID_d^{*(c)} = \prod_{i=1}^{d^2} \sum_{j=1}^{d^2} \prod_{k=1}^{d^2} x_{j,d(k-1)+i}$$

Thus, each irreducible factor of $ID_d^{*(c)}$ is of degree d^2 and has d^2 monomials. On the other hand, for each n $\mathrm{PER}_n^{(c)}$ is irreducible with $n!$ monomials. Thus, $\mathrm{PER}_n^{(c)}$ can be obtained as a projection of $ID_d^{*(c)}$ only if $d^2 = \Omega(n!)$, which contradicts $\mathrm{PER}^{(c)} \leqslant_{proj} ID^{*(c)}$.

Finally, note that ID^* is not in VP_{nc} under the assumption $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$, as ID^* is VNP_{nc}-complete under $\leqslant_{abp}$ reductions by Theorem 5.3.7. □

5.3.1 A strict $\leqslant_{iproj}$ hierarchy in VNP_{nc}

We give an infinite hierarchy of p-families under $\leqslant_{iproj}$ reductions *between* VP_{nc} and VNP_{nc} using the p-families ID^* and D_2.

We define p-families $f^{(i)}$ in $\mathrm{VNP}_{nc} \setminus \mathrm{VP}_{nc}$ such that $f^{(i)} \leqslant_{iproj} f^{(i+1)}$ but $f^{(i+1)} \not\leqslant_{iproj} f^{(i)}$, for each positive integer $i \in \mathbb{N}$. Let $ID^* = (ID_n)$ where ID_n^* are degree n^3, and $D_2 = (D_{2,n})_{n \geqslant 0}$ where $D_{2,n}$ are degree $2n$. Each $f^{(i)} = (f_n^{(i)})$, where

$$\begin{aligned} f_n^{(1)} &= ID_n^*, \\ f_n^{(2)} &= D_{2,n} ID_n^*, \\ f_n^{(i)} &= f_n^{(i-1)} D_{2,n} ID_n^*, \end{aligned}$$

for each $i, n \in \mathbb{N}$. It is easy to verify that $f^{(i)} \in \mathrm{VNP}_{nc}$ for all i. The degree of $f_n^{(i)}$ is $i(n^3 + 2n)$.

Proposition 5.3.9. *For every i, $f^{(i)} \leqslant_{iproj} f^{(i+1)}$, where the $f^{(i)}$ are the p-families defined*

above.

Proof. The indexed projection that gives a reduction from $f_n^{(i)}$ to $f_n^{(i+1)}$ will simply substitute 1 for the variables (occurring in positions $1 \leq i \leq n$, and 1 for the variables) occurring in positions $n+1 \leq i \leq 2n$. For all other occurrences of the variables of $D_{2,n}$ in the positions $1 \leq j \leq 2n$, the indexed projection substitutes 0. This substitution picks out the following unique degree-$2n$ monomial in the first copy of $D_{2,n}$

$$\underbrace{(((\cdots((}_{n-times}\underbrace{))\cdots)))}_{n-times}$$

in the polynomial $D_{2,n}$ and gives it the value 1, and it zeros out the remaining monomials of $D_{2,n}$.

For positions $2n+1 \leq j \leq n^3+2n$, the indexed projection will substitute 1 for variable x_1 and 0 for all other variables, which will pick out the unique monomial $x_1^{n^3}$ from ID_n^* and give it value 1 and zero out all other monomials in the first copy of ID_n^*.

Finally, the indexed projection substitutes x for x, for each variable x occurring in positions after $2n+n^3$. □

It remains to show that $f^{(i+1)} \not\leq_{iproj} f^{(i)}$ assuming $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$. First, we observe that ID^* and D_2 are incomparable under $\leqslant_{iproj}$ reductions, assuming $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$. In order to show this we need to show that $D_{2,n}^{(c)}$ is irreducible.

Lemma 5.3.10. *The polynomial $D_{2,n}^{(c)}$ is irreducible for each n.*

Proof. Suppose $D_{2,n}^{(c)} = g.h$ is a nontrivial factorization. Notice that $D_{2,n}^{(c)}$ is set-multilinear of degree $2n$ since the i-th location is allowed only one variable from the set $\{(_i,)_i, [_i,]_i\}$. It follows that g and h are both homogeneous and multilinear, and their variable sets are disjoint.

Thus, every nonzero monomial m of f has a unique factorization $m = m_1 m_2$, where m_1

occurs in g and m_2 in h. There are no cancellations of terms in the product gh. Hence, it also follows that both g and h are set-multilinear, where the set of locations $[2n]$ is partitioned as S for g and $[2n] \setminus S$ for h. The monomials of g are over variables in $\{(_i,)_i, [_i,]_i \mid i \in S\}$ and monomials of h are over variables in $\{(_i,)_i, [_i,]_i \mid i \in [2n] \setminus S\}$.

Now, there are monomials m occurring in $D^{(c)}_{2,n}$ such that the projection of m onto positions in S does not give a string of matched brackets. Let m' be any such monomial. Then we have the factorization $m' = m'_1.m'_2$, where m'_1 and m'_2 are monomials that occur in g and h respectively. Let the monomial m'' be obtained from m' by swapping $(_i$ with $[_i$ and $)_i$ with $]_i$. Notice that m'' occurs in $D^{(c)}_{2,n}$. Let $m'' = m''_1 m''_2$, where m''_1 and m''_2 occur in g and h, respectively. Now, since there are no cancellations in the product gh, the monomial $m'_1 m''_2$ (which is not a properly matched bracket string) must also occur in gh and hence in $D^{(c)}_{2,n}$, which is a contradiction. This completes the proof. □

Lemma 5.3.11. *1. If* $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$ *then* $ID^* \not\leqslant_{iproj} D_2$.

2. $D_2 \not\leqslant_{iproj} ID^*$.

Proof. The first part follows from the VNP_{nc}-completeness of ID^* shown in Theorem 5.3.7. For the second part, if $D_2 \leqslant_{iproj} ID^*$ then by Lemma 5.3.3 it follows that $D^{(c)}_2 \leqslant_{proj} ID^{*(c)}$. By Lemma 5.3.10 $D^{(c)}_{2,n}$ is irreducible for each n. Moreover, the number of monomials of $D^{(c)}_{2,n}$ is $2^{\Omega(n)}$. On the other hand, each irreducible factor of $ID^{*(c)}_d$ has only d^2 monomials. Hence, $D^{(c)}_2 \not\leqslant_{proj} ID^{*(c)}$. □

We now show that $f^{(i)}$ form a strictly infinite hierarchy under $\leqslant_{iproj}$ reductions in $\mathrm{VNP}_{nc} \setminus \mathrm{VP}_{nc}$.

Theorem 5.3.12. *If* $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$ *then for each* i $f^{(i+1)} \not\leqslant_{iproj} f^{(i)}$.

Proof. Suppose $f^{(i+1)} \leqslant_{iproj} f^{(i)}$. Then there are a polynomial $p(n)$ and indexed projection map ϕ_n s.t. $f^{(i)}_{p(n)}(\phi_n(X^{(i)}_{p(n)})) = f^{(i+1)}_n(X^{(i+1)}_n)$, where $X^{(i)}_{p(n)} = Var(f^{(i)}_{p(n)})$ and $X^{(i+1)}_n = Var(f^{(i+1)}_n)$. By definition we have

- $f^{(i)}_{p(n)} = \underbrace{D_{2,p(n)}ID^*_{p(n)} \dots D_{2,p(n)}ID^*_{p(n)}}_{i-times}$
- $f^{(i+1)}_{n} = \underbrace{D_{2,n}ID^*_{n} \dots D_{2,n}ID^*_{n}}_{(i+1)-times}$

By Lemma 5.3.11, $ID^*_n \not\leq_{iproj} D_{2,n}$ and $D_{2,n} \not\leq_{iproj} ID^*_n$. Therefore, $D_{2,n}ID^*_n \not\leq_{iproj} D_{2,p(n)}$ and $D_{2,n}ID^*_n \not\leq_{iproj} ID^*_{2,p(n)}$. Hence $D_{2,n}ID^*_n$ must get mapped by the projection ϕ_n to the product $D_{2,p(n)}ID^*_{p(n)}$ or $ID^*_{p(n)}D_{2,p(n)}$, overlapping both factors. But $f^{(i+1)}_n$ has $(i+1)$ such factors $D_{2,n}ID^*_n$. Hence, at least one of these factors $D_{2,n}ID^*_n$ must map wholly to $ID^*_{p(n)}$ or $D_{2,p(n)}$ by the indexed projection ϕ_n, which is a contradiction to Lemma 5.3.11. Hence $f^{(i+1)} \not\leq_{iproj} f^{(i)}$. This proves the theorem. □

In Theorem 5.3.12 we have shown that there is infinite hierarchy of explicit p-families between VP_{nc} and VNP_{nc}. Next we prove that for each k we can construct k explicit p-families in $\text{VNP}_{nc} \setminus \text{VP}_{nc}$ that are incomparable w.r.t. $\leqslant_{iproj}$ reductions.

Definition 5.3.13. *Let k be a positive integer. For each $i \in [k]$ define the product polynomial*

$$g^{(i)}_{k,n} = P_{1,i,n}P_{2,i,n} \cdots P_{k,i,n}$$

where $P_{i,i,n} = ID^*_n$ and for $j \neq i$ $P_{j,i,n} = D_{2,n}$.

Theorem 5.3.14. *The p-families defined as $g^{(i)}_k = (g^{(i)}_{k,n})_n$, for $1 \leq i \leq k$, are pairwise incomparable w.r.t. $\leqslant_{iproj}$ reductions.*

Proof. Suppose to the contrary that $g^{(i)}_k \leqslant_{iproj} g^{(i')}_k$ for $i \neq i'$. Then for each n there are a polynomial $p(n)$ and indexed projection map ϕ_n s.t. $g^{(i')}_{k,p(n)}(\phi_n(X^{(i')}_{p(n)})) = g^{(i)}_{k,n}(X^{(i)}_n)$, where $X^{(i')}_{p(n)} = Var(g^{(i')}_{k,p(n)})$ and $X^{(i)}_n = Var(g^{(i)}_{k,n})$. By definition

$g^{(i')}_{k,p(n)} = P_{1,i',p(n)}P_{2,i',p(n)} \cdots P_{k,i',p(n)}$.

For $1 \leq j \leq k$, let f_j denote the polynomial obtained from $P_{j,i',p(n)}$ after applying the

substitution map ϕ_n. Hence we have

$$P_{1,i,n}P_{2,i,n}\cdots P_{k,i,n} = f_1 \cdot f_2 \ldots f_k.$$

These are two factorizations of the same noncommutative polynomial. Notice that each $P_{j,i,n}$ is a homogeneous polynomial. Hence their product (which is the left hand side of the equality) is also a homogeneous polynomial. Since that homogeneous polynomial has also the product of f_j's as its factorization, given by the right hand side, it forces that each $f_j, 1 \leq j \leq k$ is homogeneous. Furthermore, both $D_{2,n}$ and ID^*_n are irreducible polynomials. Consequently, the left hand side of the above equality has exactly k irreducible factors. Now, homogeneous noncommutative polynomials have unique factorization into irreducible factors, upto scalar multiples (e.g. see [AJR16]). Therefore, it follows that $P_{j,i,n}$ and f_j are equal upto scalar multiplication for $1 \leq j \leq k$.

In particular, for $j = i$ it follows that the index projection is a reduction that maps $P_{i,i,n} = ID^*_n$ to a scalar multiple of $f_i = \phi_n(P_{i,i',p(n)}) = D_{2,p(n)}$. This implies ID^* is $\leqslant_{iproj}$ reducible to (a scalar multiple of) D_2. More precisely, if we define the p-family $(zD_{2,n})_n$, where z is a new variable, then ID^* is $\leqslant_{iproj}$ reducible to $(zD_{2,n})_n$, where the variable z can be substituted by a suitable scalar to cancel the scalar multiple introduced above. Since $(zD_{2,n})_n$ is clearly in VP_{nc}, it is a contradiction to Lemma 5.3.11. This proves the theorem.

□

Finally, we exhibit an infinite collection of p-families in $\mathrm{VNP}_{nc} \setminus \mathrm{VP}_{nc}$ that consist of explicit polynomials and these p-families are incomparable under $\leqslant_{proj}$ reductions.

For this purpose we consider a variant of ID^* (which we still denote by ID^*). For each positive integer n, let $X_n = \{x_0, x_1\}$. Let W_n denote the set of all degree n monomials over

X_n and define the polynomial

$$ID^*_n = \sum_{w \in W_n} \underbrace{ww \dots w}_{n^2-times}.$$

Clearly, under this modified definition too the p-family $ID^* = (ID^*_n)$ is in VNP_{nc}. Moreover, this modified ID^* remains VNP_{nc}-complete under $\leqslant_{abp}$ reductions by making small changes to the proof of Theorem 5.3.7. Following the proof of Theorem 5.3.8, we can see that this modified ID^* is also VNP_{nc}-intermediate p-family w.r.t. $\leqslant_{iproj}$, assuming $\text{VP}_{nc} \neq \text{VNP}_{nc}$.

Theorem 5.3.15. *The p-families $f_i = (ID^*_n x^i)_n$ for $i = 1, 2, 3, \dots$ are pairwise incomparable under $\leqslant_{proj}$ reductions.*

Proof. Suppose to the contrary that $f_i \leqslant_{proj} f_j$ for $i \neq j$. Now, for each n, ID^*_n is a polynomial defined over variables x_0, x_1. The reduction is defined by a substitution map ϕ_n and polynomial $p(n)$ such that on substituting x_0, x_1, x in $ID^*_{p(n)} x^j$ by $\phi_n(x_0), \phi_n(x_1), \phi_n(x)$, respectively, we obtain the polynomial $ID^*_n x^i$. Notice that $\phi_n(x)$ cannot be a scalar. Because the polynomial $ID^*_n x^i$ has three variables, and with $\phi_n(x)$ set to a scalar, the polynomial $\phi_n(ID^*_{p(n)})$ has at most two variables which is impossible. As the rightmost factor of $ID^*_n x^i$ is x, we must have $\phi_n(x) = x$.

Clearly, $i < j$ is not possible, because the substitution ϕ_n cannot get rid of x^{j-i} in $ID^*_{p(n)} x^j$ by any substitutions to x_0 and x_1. If $i > j$ then we want to obtain the polynomial $ID^*_n x^{i-j}$ (defined over three variables) from the polynomial $ID^*_{p(n)}$ (defined over 2 variables) using projections which is again impossible. □

Remark 5.3.2. *Theorem 5.3.15 exhibits the weakness of $\leqslant_{proj}$ reductions when applied to noncommutative polynomials with a constant number of variables. Specifically, for the p-family ID^* that is defined as above in two variables x_0 and x_1, although we cannot reduce $ID^* x^i$ to ID^* under projections, with minor modifications to the proof of Theorem 5.3.7 we can show that it remains complete for VNP_{nc} under $\leqslant_{abp}$ reductions. In contrast, by*

exploiting the occurrence of many distinct variables, projections are strong enough to prove the VNP_{nc}*-completeness of the noncommutative Permanent [HWY10b].*

5.3.2 Discussion

Proving the existence of VNP_{nc}-intermediate p-families *under* $\leqslant_{abp}$ *reductions*, assuming $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$, remains open. At present we do not see an approach. However, in Section 4.3.4 we briefly discussed $\leqslant_{linproj}$, which we termed linear indexed reducibility. Unfortunately, our proof that PER $\not\leqslant_{iproj}$ ID^* (see Theorem 5.3.8) does not generalize to $\leqslant_{linproj}$. Specifically, our proof is based on counting the number of monomials in the irreducible factors of $ID_n^{*(c)}$ and PER_n, which does not carry over to linear indexed projections. Indeed, it is easy to note that $\leqslant_{linproj}$ does not, in general, preserve the number of monomials in irreducible factors.

However, a plausible stronger assumption than $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$ implies the existence of VNP_{nc}-intermediate p-families under $\leqslant$ *linproj* reductions.

Conjecture 5.3.16 (SOS_k Conjecture)**.** *Consider expressing the biquadratic polynomial*

$$SOS_k(x_1, \ldots, x_k, y_1, \ldots, y_k) = (\sum_{i\in[k]} x_i^2)(\sum_{i\in[k]} y_i^2)$$

as a sum of squares ($\sum_{i\in[s]} f_i^2$)*, where* f_i *are all homogeneous bilinear polynomials with the minimum* s*.*

The SOS_k *conjecture states that over complex numbers (or the algebraic closure of any field of characteristic different from* 2*), for all* k *we have the lower bound* $s = \Omega(k^{1+\epsilon})$ *for some constant* $\epsilon > 0$ *independent of* k*.*

In [HWY10a], it is shown that the SOS_k-conjecture implies that the p-family ID is not in VP_{nc}. In fact, they prove exponential circuit size lower bounds for ID_d assuming the conjecture.

It is easy to see that unconditionally PER $\not\leqslant_{linproj}$ ID. We can apply the argument of counting monomials in the irreducible factors of ID_d, which is also used in the proof of Theorem 5.3.8. The reason is that the irreducible factors of ID_d are of degree 2 and even with linear substitutions the number of monomials in each factor remains polynomially bounded. As PER_n is irreducible with exponentially many monomials it follows that PER $\not\leqslant_{linproj}$ ID.

Now, since the SOS_k conjecture implies that $ID \notin \text{VP}_{nc}$, it follows that ID is a VNP_{nc}-intermediate p-family assuming the SOS_k conjecture.

Finally, exactly as in Section 5.3.1, we can combine ID and D_2 to define an infinite hierarchy of p-families $g^{(i)}\text{q}$ within $\text{VNP}_{nc} \setminus \text{VP}_{nc}$ under $\leqslant_{linproj}$ reductions. We omit the proof details. The p-families $g^{(i)}$ are defined as:

$$g_n^{(1)} = ID_n, \quad g_n^{(2)} = D_{2,n}ID_n, \quad g_n^{(i)} = g_n^{(i-1)}D_{2,n}ID_n.$$

Theorem 5.3.17. *Assuming SOS_k conjecture we have for every i:*

1. $g^{(i)} \leqslant_{linproj} g^{(i+1)}$.
2. $g^{(i+1)} \not\leqslant_{linproj} g^{(i)}$.

5.4 More on VNP_{nc}-Completeness

By the transfer theorem (Theorem 5.3.4) we know that if f is a VNP_{nc}-complete p-family under $\leqslant_{iproj}$ reductions, then in the commutative setting $f^{(c)}$ is VNP-complete under $\leqslant_{proj}$-reductions.

For the reverse direction, suppose f is a commutative p-family which is VNP-complete under $\leqslant_{proj}$-reductions. There are several examples starting with the permanent, the p-family HC (corresponding to Hamiltonian circuits) and so on [Val79a]. Is there an associated noncommutative p-family that is VNP_{nc}-complete under $\leqslant_{iproj}$ reductions? In this

section we formulate an answer to this question and make some related observations.

Suppose $f = (f_n)$ is a commutative p-family that is VNP-complete. Since f is VNP-complete, suppose $\text{PER}_n \leqslant_{proj} f_{r(n)}$ for each n, where $r(n)$ is polynomially bounded.

Suppose the polynomial $f_{r(n)} \in \mathbb{F}[X_n]$ is of degree $d(n)$. Let $X_n = \{x_1, x_2, \ldots, x_{q(n)}$ ordered by increasing indices. The monomials of $f_{r(n)}$ are of the form $m = x_1^{e_1} x_2^{e_2} \ldots x_{q(n)}^{e_{q(n)}}$, where the sum of the exponents e_i is at most $d(n)$. Letting β_m denote the coefficient of monomial m in $f_{r(n)}$, we can write

$$f_{r(n)} = \sum_m \beta_m m.$$

Now, consider the *noncommutative* p-family $f^* = (f_n^*)$ where

$$f_n^* = \sum_m \beta_m \underbrace{mm \ldots m}_{n-times}.$$

Note that $f_n^* \in \mathbb{F}\langle X_n \rangle$ for each n.

Proposition 5.4.1. *If f is* VNP*-complete under $\leqslant_{proj}$ reductions then f^* is* VNP_{nc}*-complete under $\leqslant_{iproj}$ reductions.*

Proof. Denote PER_n's variables by $X_{jk}, 1 \leq j, k \leq n$. Let $i \in [q(n)]$. Suppose the $\leqslant_{proj}$ reduction from PER_n to $f_{r(n)}$ does the substitution

$$x_i \leftarrow X_{jk},$$

then in the noncommutative case the $\leqslant_{iproj}$ reduction from PER_n to f_n^* substitutes X_{jk} for the x_i in the $((j-1)n + k)^{th}$ copy of m and substitutes 1 for x_i in all other copies of m. If the reduction does the substitution

$$x_i \leftarrow \alpha,$$

for a scalar α, then in the noncommutative case the $\leqslant_{iproj}$ reduction substitutes α for x_i in the 1^{st} copy of m and substitutes 1 for x_i in all other copies of m. It is easy to verify that

this trick of repeated copies ensures that the projection transforms f_n^* to PER_n, where all the monomials of PER_n are ordered as $X_{1i_1}X_{2i_2}\ldots X_{ni_n}$ as per its definition. This completes the proof. □

5.4.1 A generalized permanent

We next address a different question regarding the permanent polynomial. Let $\chi : S_n \to \mathbb{F} \setminus \{0\}$ be any polynomial-time computable function assigning nonzero values to each permutation in S_n. We define a generalized permanent polynomial

$$\mathrm{PER}_n^{\chi} = \sum_{\sigma \in S_n} \chi(\sigma) x_{1\sigma(1)} x_{2\sigma(2)} \ldots x_{n\sigma(n)}.$$

Clearly $\mathrm{PER}^{\chi} = (\mathrm{PER}_n^{\chi})$ is a p-family that is in VNP_{nc}. For which functions χ is PER^{χ} a VNP_{nc}-complete p-family? In other words, does the hardness of the *noncommutative* permanent depend only on the nonzero monomial set (and the coefficients are not important)?

In the commutative setting, a related well-studied question is the complexity of immanents. For each *Young diagram* λ the immanent polynomial is defined as

$$Imm_{\lambda}(X) = \sum_{\pi \in S_n} \chi_{\lambda}(\pi) \prod_{i=1}^{n} X_{i\pi(i)}.$$

The λ are basically (ordered) partitions of n, and we can draw a staircase like diagram (known as the Ferrers diagram) to represent them. The two extreme cases are when the diagram is a single column (then the immanent is the determinant) and a single row (the immanent is the permanent in this case). Intermediate cases are algorithmically well studied with many interesting results [Bür00b, Bar90, BÖ0, Har85, MM13]. Notably, the immanent polynomial is efficiently solvable when the Ferrers diagram is concentrated on the leftmost column (but for a constant number of entries) [Bür00b, Bar90]. Furthermore,

the character χ_λ itself is known to be #P-hard to compute for arbitrary partitions λ [Hep94,].

In the noncommutative setting, as already shown in [AS10], PER $\leqslant_{abp}$ DET. Thus, it is quite plausible that Imm_λ is a hard polynomial for each partition λ, although we have not been able to answer this question. The main technical difficulty is the complexity of computing χ_λ.

However, to the question regarding the complexity of PER^χ, defined above, for arbitrary but easily computable functions χ, we are able to give a partial answer. Define

$$\mathrm{PER}^* = \sum_{\sigma \in S_n} \underbrace{\overline{X}_\sigma \overline{X}_\sigma \ldots \overline{X}_\sigma}_{n-times}, \text{ where } \overline{X}_\sigma \text{ is the monomial } x_{1\sigma(1)} \ldots x_{n\sigma(n)}.$$

Proposition 5.4.2. PER^* *is* VNP_{nc}*-complete.*

The above proposition is easy to prove: PER^* is in VNP_{nc} because the coefficient of any given monomial is polynomial-time computable. Furthermore, PER is $\leq_{iproj}$-reducible to PER^* by substituting 1 for all except the first n variables in every monomial.

Now, consider the polynomial

$$\mathrm{PER}^{*,\chi} = \sum_{\sigma \in S_n} \chi(\sigma) \underbrace{\overline{X}_\sigma \overline{X}_\sigma \ldots \overline{X}_\sigma}_{n-times}.$$

We prove the following theorem about PER^χ and $\mathrm{PER}^{*,\chi}$ under assumptions about the function χ.

Theorem 5.4.3. *Suppose the function* χ *is such that* $|\chi(S_n)| \leq p(n)$ *for some polynomial* $p(n)$ *and each* n*. Then*

- *If* χ *is computable by a* 1*-way logspace Turing machine then* PER $\leqslant_{abp}$ PER^χ.

- *If χ is computable by a logspace Turing machine then* $\mathrm{PER} \leqslant_{abp} \mathrm{PER}^{*,\chi}$.

Proof. We explain the second part of the theorem. The first part follows from the proof of the second. The idea is to construct an automaton from the given logspace machine such that for a given $\sigma \in S_n$, the automaton computes $\frac{1}{\chi(\sigma)}$ in the field $\mathbb{F}$.

Let T be a logspace Turing machine which uses space $s = O(\log n)$, computing χ. Thus, total running time of T is bounded by $P(n)$, where $P(n)$ is some fixed polynomial in n. Since the range of χ is $p(n)$ bounded in size, we can encode in a state of the automaton the following:

- Input head position,
- Content of working tape, and
- Content of output tape.

The number of states is bounded by a polynomial in n. We can convert this log-space machine T on input σ into a one-way log-space machine T' on a modified input as follows:

- The input to T' is the concatenation of $P(n)$ copies of σ. Thus the input to T' is of the form $\sigma\sigma\ldots\sigma$, with $P(n)$ many σ.
- At a step i, T' reads from the i^{th} copy.

The difference between machine T' and T is that T' is a *1-way* logspace machine whose input head moves always to the right. For $\sigma \in S_n$, we can convert T' into a deterministic automaton with $\mathsf{poly}(n)$ many states as follows: there are only polynomially many instantaneous descriptions of T'. This consists of the input head position, the work tape contents and head position, and the current output string (which is a prefix of some element in the range $\chi(S_n)$). When this automaton completes reading the input, suppose the

state q contains the output element $\alpha = \chi(\sigma)$. The automaton has a transition from q to the unique final state t labeled by scalar $1/\chi(\sigma)$.

Finally, we can modify this automaton to work on the monomials $\overline{X}_\sigma \overline{X}_\sigma \ldots \overline{X}_\sigma$, where it replaces all but the first block of variables by 1.

When the polynomial $\mathrm{PER}^{*,\chi}$ is evaluated on the matrices corresponding to the above automaton (with the substitutions), the $(s,t)^{th}$ entry of the output matrix will be the permanent polynomial PER_n. □

Remark 5.4.1. *We note that the sign of a permutation can be computed by a logspace Turing machine, which implies that* DET^* *(which is* $\mathrm{PER}^{*,\chi}$ *where* $\chi(\pi)$ *is the sign of* π*) is* VNP_{nc}*-complete under* $\leqslant_{abp}$ *reductions. As the above theorem is for any logspace computable* χ*, it is not strong enough to imply the hardness of* DET*. The hardness proof of* DET *shown in [AS10] uses a different strategy.*

5.5 Inside VP_{nc}

In the boolean complexity setting, the sub-classes of P are the parallel complexity classes NC^i defined by boolean circuits with bounded fanin gates of polynomial size and $log^i n$ depth for length n inputs. On the other hand, we have no such hierarchy of algebraic complexity classes inside the commutative Valiant class VP because VP coincides with VNC^2. The reason for it is that commutative arithmetic circuits of polynomial degree can be transformed to logarithmic depth with only a polynomial increase in size.

In this section we briefly examine the structure within VP_{nc}. It follows easily from Nisan's rank argument [Nis91] that the corresponding VNC_{nc} classes form a strict infinite hierarchy within VP_{nc}. Furthermore, by considering Dyck polynomials with $\log^i n$ nesting depth we obtain a strict hierarchy under $\leqslant_{abp}$ reductions which roughly corresponds to the VNC_{nc} hierarchy.

Definition 5.5.1. *A p-family $f = (f_n)$ is in VNC^i_{nc} if there is a family of circuits (C_n) for f such that each C_n is of polynomial size and degree, and is of $\log^i n$ depth. The class VNC_{nc} is the union $\cup_i \mathrm{VNC}^i_{nc}$.*

The classes $\mathrm{VNC}^i_{nc}, i = 1, 2, \ldots$ are clearly contained in VP_{nc}. Furthermore, Nisan's rank argument directly implies that $\mathrm{VNC}^i_{nc}, i = 1, 2, \ldots$ form a strict hierarchy. Specifically, for each i, palindromes of length $\log^{i+1} n$ over variables $\{x_0, x_1\}$ have circuits of depth $\log^{i+1} n$ and size $O(\log^{i+1} n)$. However, circuits of depth $\log^i n$ for it require superpolynomial size.

5.5.1 Dyck depth hierarchy inside VP_{nc}

We now show that the nesting depth of Dyck polynomials yields a strict hierarchy of p-families within VP_{nc}. This hierarchy roughly corresponds to the VNC_{nc} hierarchy.

Definition 5.5.2 (Nesting depth)**.** *The nesting depth of a string in D_2 is defined as follows:*

- () *and* [] *have depth 1.*
- *If u_1 has depth d_1 and u_2 has depth d_2, u_1u_2 has depth $max\{x_1, d_2\}$ and $(u_1), [u_1]$ have depth $d_1 + 1$.*

Let $W^{(k)}_{2,n}$ denote the set of all monomials in $D_{2,n}$ of depth at most k and degree $2n$. We define the polynomial $D^{(k)}_{2,n} = \sum_{u \in W^{(k)}_{2,n}} u$ and denote the corresponding p-family as $D^{(k)}_2$. In this definition we allow k to be a growing function $k(n)$ of n, where $D^{(k)}_2 = (D^{(k)}_{2,n})_{n \geqslant 0}$.

We next observe that the Dyck polynomial of nesting depth $\log^{i+1} n$ lies strictly between VNC^i_{nc} and VNC^{i+2}_{nc}. We need following definition.

Definition 5.5.3. *A p-family $f = (f_n)$ is in VAC^i_{nc} if there is a family (C_n) of circuits with* unbounded *fanin gates such that each C_n is of polynomial size and degree, and $\log^i n$ depth. The class VAC_{nc} is the union $\bigcup_i \mathrm{VAC}^i_{nc}$.*

We note that $\mathrm{VNC}^i_{nc} \subseteq \mathrm{VAC}^i_{nc} \subseteq \mathrm{VNC}^{i+1}_{nc}$, because we can simulate an unbounded fanin with a subcircuit of $O(\log n)$ depth and polynomial in n size, with fanin two gates.

Theorem 5.5.4. *For any $i \geqslant 0$, the Dyck polynomial of depth $\log^{i+1} n$ satisfy the following:*

1. $D_2^{(\log^{i+1} n)}$ *is hard for* VNC^i_{nc} *for* $\leqslant_{abp}$ *reductions.*
2. $D_2^{(\log^{i+1} n)} \in \mathrm{VAC}^{i+1}_{nc} \setminus \mathrm{VNC}^i_{nc}$.
3. $D_2^{(\log^{i+1} n)}$ *is not hard for* VAC^{i+1}_{nc} *for* $\leqslant_{abp}$ *reductions.*

Proof. It follows from inspection of the proof of the Theorem 4.4.2 that it scales down and yields part (1) of the theorem.

We now show part (2). Assume that $D_2^{(\log^{i+1} n)} \in \mathrm{VNC}^i_{nc}$. Then $D_2^{(\log^{i+1} n)}$ has an algebraic branching program of size $2^{O(\log^i n)} \cdot poly(n)$. The $\leqslant_{abp}$ reduction from PAL to D_2 can be easily modified to show that $\mathrm{PAL}_{\log^{i+1} n} \leqslant_{abp} D_{2,n}^{(\log^{i+1} n)}$. It follows that $\mathrm{PAL}_{\log^{i+1} n}$ too has an ABP of size $2^{O(\log^i n)} \cdot poly(n)$, which contradicts Nisan's result [Nis91] that $\mathrm{PAL}_{\log^{i+1} n}$ requires ABPs of size $2^{\Omega(\log^{i+1} n)}$. Hence, $D_2^{(\log^{i+1} n)} \notin \mathrm{VNC}^i_{nc}$.

Now we show that $D_2^{(\log^{i+1} n)} \in \mathrm{VAC}^i_{nc}$. More generally, consider the polynomial $\hat{D}^{k(n)}_{l(n),d(n)} = \sum_{p=1}^{k(n)} D^p_{l(n),d(n)}$ where $l(n), d(n)$ and $k(n)$ are polynomial functions in n. Clearly,

$$\hat{D}^k_{l,d} = \sum_{i=1}^{l} ({}_i\hat{D}^{k-1}_{l,d-2})_i + \sum_{i,j\in[l]} \sum_{p=0}^{d-4} ({}_i\hat{D}^{k-1}_{l,p})_i ({}_j\hat{D}^{k-1}_{l,d-4-p})_j.$$

Using the above recursive description, we can construct an unbounded fanin polynomial size arithmetic circuit of depth $O(k(n))$ for $\hat{D}^{k(n)}_{l(n),d(n)}$ recursively. Clearly, the resulting circuit will have depth $O(k(n))$. From this we can obtain a polynomial size arithmetic circuit of depth $O(k(n))$ for the largest degree homogeneous part which will be $\hat{D}^{k(n)}_{l(n),d(n)}$. Applying this for $l = 2$ yields an unbounded fanin polynomial size circuit for $D_2^{(\log^{i+1} n)}$ of depth $O(\log^{i+1} n)$. Hence, $D_2^{(\log^{i+1} n)} \in \mathrm{VAC}^i_{nc}$.

In order to prove (3), we exhibit a polynomial f in VAC^{i+1}_{nc} such that $f \not\leqslant_{abp} D_2^{(\log^{i+1} n)}$. Let $f = D^{(\log^{i+1} n)}_{\log^{i+1} n}$. We know that $f \in \mathrm{VAC}^{i+1}_{nc}$ from the above recursive description. Now, if $f \leqslant_{abp} D_2^{(\log^{i+1} n)}$ then f has an algebraic branching program of size $2^{O(\log^{i+1} n)} \cdot poly(n)$. Applying Nisan's rank argument to the polynomial f, we can see that any ABP for f must have size at least $(\log^{i+1} n)!$ in the $\log^{i+1} n^{th}$ layer of the ABP. Hence any algebraic branching program for f is of size $2^{\omega(\log^{i+1} n)}$ which is a contradiction. □

5.6 Concluding remarks and open problems

Several open questions arise from the results in this chapter. We list the important ones below:

- Assuming $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$, analogous to Ladner's theorem, we have given an infinite hierarchy within VNP_{nc} under $\leqslant_{iproj}$ reductions. Likewise, we have shown infinitely many p-families that are incomparable under $\leqslant_{proj}$ reductions and arbitrarily many under $\leqslant_{iproj}$ reductions. Similar results for the more powerful $\leqslant_{abp}$ reducibility will require substantially new techniques. It is also interesting to further compare the strengths of the three hypotheses considered in this chapter: $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$, VP ≠ VNP, and the SOS_k conjecture. As explained in Section 5.3.2, the first hypothesis is the weakest of the three. Does the SOS_k conjecture imply VP ≠ VNP?

- Suppose $f = (f_n)$ is a p-family such that f_n has the same nonzero monomial set as PER_n for each n. When the coefficients of f_n are 1-way logspace computable from their corresponding monomials, we have shown f is VNP_{nc}-complete under $\leqslant_{abp}$ reductions. Can we prove any hardness result for f in general?

- We have seen that $ID \not\leqslant_{iproj} D_2$. Showing that $ID \not\leqslant_{abp} D_2$ would imply superpolynomial circuit size lower bounds for ID. It would be interesting to show this in the special case when the $\leqslant_{abp}$ reductions are allowed only 2×2 matrix substitutions.

- The complexity of the noncommutative immanent discussed in Section 5.4 remains open for different Young diagrams.

Chapter 6

Linear circuits over noncommutative domains

6.1 Introduction

In this chapter, we present our results on lower bounds for *multiplicative circuits and linear circuits over noncommutative domains*. We start by summarizing results in this chapter and then formally define multiplicative and linear circuits over noncommutative domains. The main results are the following.

- Let $(S, \circ)$ be a semigroup. We show that there exists a list of elements $y_1, y_2, ..., y_m \in S$, which can be generated using two elements $x_0, x_1 \in S$ by semigroup operation $\circ$. Let the explicit functions y_i, $1 \leqslant i \leqslant m$ are defined as words $y_i = y_{i1}y_{i2}...y_{in}$ where $y_{ij} \in \{x_0, x_1\}$ and $\{y_1, y_2, ..., y_m\}$ are explicitly defined. We show that the size, the number of semigroup operations $\circ$ required for computing $y_1, y_2, ..., y_m \in S$ using x_0, x_1 as input, of any circuit $C : \{x_0, x_1\} \rightarrow \{y_1, y_2, ..., y_m\}$ computing $y_1, y_2, ..., y_m \in S$ is $\Omega(\frac{mn}{\log^2 n})$ in the following four noncommutative domains.

  1. When $(S, \circ)$ is the free monoid X^* for X such that $|X| \geqslant 2$.

2. When $(S, \circ)$ is the finite matrix semigroup over the boolean ring and matrices are of dimension $n^c \times n^c$ for some constant $c > 0$.

3. When $(S, \circ)$ is the free group G_X generated by $X = \{x_1, x_2, x_1^{-1}, x_2^{-1}\}$.

4. When $(S, \circ)$ is the permutation group where $S = S_N$ for $N = n^d$ for some constant $d > 0$.

The complexity measure we use is the number of distinct substrings of certain length of a given string. If number of distinct substrings are large then we show that circuit size is also large.

- We study a generalization of the linear circuits model (see e.g., [Lok09]), where we allow the coefficients come from *noncommutative rings*. We show that there exists an explicit matrix $A \in \mathbb{F}^{n\times n}\langle x_0, x_1\rangle$ such that computing $A\mathbf{y}$ by any homogeneous linear circuit C over the coefficient ring $\mathbb{F}\langle x_0, x_1\rangle$ requires either size $\omega(n)$ or depth $\omega(\log n)$. We prove this by suitably generalizing Valiant's matrix rigidity method [Val77].

- We next consider homogeneous depth 2 linear circuits. These are linear circuits of depth 2, where each addition gate can have unbounded fanin. We show that there exists an explicit matrix $A \in \mathbb{F}^{n\times n}\langle x_0, x_1\rangle$ such that computing $A\mathbf{y}$ by a depth 2 homogeneous linear circuit (with unbounded fanin) requires $\Omega(\frac{n^2}{\log n})$ wires.

6.2 Lower bounds for Multiplicative Circuits

6.2.1 Motivation and Our Results

To state our results, we need the following definition. Let $(S, \circ)$ be a semigroup, i.e., S is a set closed under the binary operation $\circ$ which is associative. A natural multi-output computational model is a circuit over $(S, \circ)$.

Definition 6.2.1 (Multiplicative Circuit)**.** *The multiplicative circuit C over $(S, \circ)$ is given by a directed acyclic graph with input nodes labeled $x_1, ..., x_n \in S$ of indegree 0 and output nodes $y_1, ..., y_m \in S$ of outdegree 0. The internal nodes of the circuit C is labeled by semigroup operation $\circ$.*

The gates of the circuit all compute the monoid product. We assume that all gates have fanin 2. The size of the circuit is the number of nodes in it and it computes a function $f : S^n \to S^m$.

This multi-output multiplicative circuit provides a general setting to some well studied problems in circuit complexity. For example,

1. If $S = \mathbb{F}_2$ and $\circ$ is addition in $\mathbb{F}_2$, the problem is one of computing Ax for an $m \times n$ matrix over $\mathbb{F}_2$. The problem of giving an explicit A such that size of any circuit for it is superlinear is a longstanding open problem. By means of counting arguments, we know that there exist such matrices A [Val77].

   This problem has a rich literature with many interesting developments. Morgenstern [Mor73] showed an $\Omega(n \log n)$ lower bound for the Hadamard matrix in the bounded coefficient model when $\mathbb{F} = \mathbb{C}$. Valiant [Val77] developed matrix rigidity as a means to attack the problem in the case of logarithmic depth circuits. In spite of many interesting results and developments, superlinear size lower bounds remain elusive over any field $\mathbb{F}$ even for the special case of log-depth circuits (Lokam's monograph [Lok09] contains most of the recent results).

2. When $S = \{0, 1\}$ and $\circ$ is the boolean OR, this problem is also well studied and due to its monotone nature it has explicit lower bounds of circuit size $n^{2-o(1)}$ (e.g., see section 3.4 in [JS13]).

   A more restricted form is $S = (\mathbb{N}, +)$ called SUM circuits also well studied e.g., [JS13]. While for monotone settings (OR,SUM) there are nontrivial lower bounds, in the commutative case for S we do not have strong lower bounds results.

We explored the case when $(S, \circ)$ is noncommutative. In principle, we can expect lower bounds could be easier to prove in this model. The circuits are more constrained when computing an element in S as fewer ways to compute an element in S. An interesting aspect is that the number of inputs can be restricted to just two: x_0, x_1.

Our Results

Let the explicit functions y_i, $1 \leqslant i \leqslant m$ are defined as words $y_i = y_{i1}y_{i2}...y_{in}$ where $y_{ij} \in \{x_0, x_1\}$ and $\{y_1, y_2, ..., y_m\}$ are explicitly defined. We show that any circuit $C : \{x_0, x_1\} \to \{y_1, y_2, ..., y_m\}$ is of size $\Omega(\frac{mn}{\log^2 n})$ in the following four settings:

1. When $(S, \circ)$ is the free monoid X^* for X such that $|X| \geqslant 2$.

2. When $(S, \circ)$ is the finite matrix semigroup over the boolean ring and matrices are of dimension $n^c \times n^c$ for some constant $c > 0$.

3. When $(S, \circ)$ is the free group G_X generated by $X = \{x_1, x_2, x_1^{-1}, x_2^{-1}\}$.

4. When $(S, \circ)$ is the permutation group where $S = S_N$ for $N = n^d$ for some constant $d > 0$.

6.2.2 Circuits over free monoids

We consider the free monoid X^* where X is a finite alphabet and the monoid operation is concatenation with the empty string ϵ as identity.

Notice that when X is a singleton set $X = \{1\}$ then $(1^*, \circ)$ is essentially the semigroup $(\mathbb{N}, +)$. We consider the simplest noncommutative setting with $X = \{0, 1\}$. In the problem, we consider circuits that take the "generating set" X as input and the m outputs $y_1, y_2, ..., y_m \in X^n$ (where n is the "input" parameter).

Since each y_i is of length n, clearly n gates are sufficient to compute each y_i and hence $O(mn)$ is an obvious upper bound for the circuit size. We will give an explicit set $y_1, y_2, ..., y_m \in \{0,1\}^n$ so that $\Omega(\frac{mn}{\log^2 n})$ is the circuit size lower bound. We will let $m = n$ in the construction and it can be suitably generalized to larger values of m. We now explain the construction of the set $S = \{y_1, y_2, ..., y_m\} \subseteq \{0,1\}^n$.

Construction of S

Consider the set $[n^2]$ of the first n^2 natural numbers. Each $i \in [n^2]$ requires $2 \log n$ bits to represent in binary. Let $D = [n^2]$.

for $i = 1, ..., n$ do

pick the first $\frac{n}{2 \log n}$ numbers from current D, concatenate their binary

representation to obtain y_i and remove these numbers from D.

end for

This defines the set $S = \{y_1, y_2, ..., y_n\}$. Each y_i constructed has the property that y_i has $\geqslant \frac{n}{2 \log n}$ distinct substrings of length $2 \log n$. We show the following two result about these strings:

- For each $y_i \in S$ any concatenation circuit that generates y_i from input $X = \{0,1\}$ requires size $\Omega(\frac{n}{\log^2 n})$.
- Any concatenation circuit that takes $X = \{0,1\}$ as input and outputs $S = \{y_1, y_2, ..., y_n\}$ at n output gates requires size $\Omega(\frac{n^2}{\log^2 n})$.

Lemma 6.2.2. *Let $s \in X^n$ be any string where $|X| \geqslant 2$, such that the number of distinct substrings of s of length l is N. Then any concatenation circuit for s will require $\Omega(\frac{N}{l})$*

gates.

Proof. Note the case not covered by the lemma: $|X| = 1$. In that case we know that every string of length n (for every n) has a concatenation circuit of size $\leqslant 2 \log n$ and the circuit exploits the fact that for each length l there is a unique string. Now we prove the lemma. Let C be any circuit that computes the string s. Now each gate g of C computes some string s_g. Suppose $g = g_1 \circ g_2$ is a gate whose inputs are gates g_1, g_2.

Suppose s_{g_1} has k_1 distinct substrings of length l and s_{g_2} has k_2 distinct substrings of length l. Now, in s_g notice that the *new* substrings of length l (not occurring in s_{g_1} or s_{g_2}) could only arise as a concatenation of some suffix of s_{g_1} and prefix of s_{g_2} such that neither of them is the empty string. The number of such substrings is at most l.

Hence, s_g can have at most $k_1 + k_2 + l$ distinct substrings of length l. Thus, each new gate of C can generate at most l new substrings of length l. Since the output string s has N distinct length l substrings, it follows that number of gates in C is $\Omega(\frac{N}{l})$. □

Now we show that any concatenation circuit that takes $X = \{0, 1\}$ as input and outputs $S = \{y_1, y_2, ..., y_n\}$ at n output gates requires size $\Omega(\frac{n^2}{\log^2 n})$.

Theorem 6.2.3. *Let $S \subseteq \{0, 1\}^n$ be the explicit set of n strings defined above. Any concatenation circuit that takes $X = \{0, 1\}$ as input and outputs S at its n output gates will require size $\Omega(\frac{n^2}{\log^2 n})$.*

Proof. Let $S = \{y_1, y_2, ..., y_n\}$ as defined above and let C be any concatenation circuit that takes $X = \{0, 1\}$ as inputs and at its n output gates generates $y_1, y_2, ..., y_n$ respectively. Let C' be a concatenation circuit obtained from C by adding $n - 1$ new gates such that C' outputs the concatenation $y = y_1 y_2 ... y_n$. By construction $size(C') = size(C) + n - 1$. The number of distinct length $2 \log n$ strings in the string y is, by construction, $\geqslant \frac{n^2}{2 \log n}$. This is because each y_i has $\geqslant \frac{n}{2 \log n}$ distinct substrings and these are disjoint for different y_i. Hence by Lemma 6.2.2, $size(C') = \Omega(\frac{n^2}{\log^2 n})$ which implies $size(C) = \Omega(\frac{n^2}{\log^2 n})$. □

6.2.3 Circuits over matrix semigroups

The setting now is that of a finite monoid $(M, \circ)$ where M consisting of $p(n)\times p(n)$ matrices whose entries come from the boolean semiring $\{0, 1, \vee, \wedge\}$. We will modify the lower bound of the previous section to make it work over $(M, \circ)$ which is a finite monoid.

Recall we constructed $S = \{y_1, y_2, ..., y_n\} \subseteq \{0,1\}^n$. Let D_l be the set of all length l substrings of each $y_i \in S$. Let $D = \bigcup_{l=0}^{n} D_l$. Clearly $|D| = \sum_{l=0}^{n} |D_l| \leqslant n^3$. The matrices in M are $|D| \times |D|$. We now define two functions $f_0, f_1 : D \to D$ corresponding to the generating set $X = \{0, 1\}$ of the free monoid. For $b \in \{0, 1\}$, define

$$f_b(s) = \begin{cases} s \circ b & s \circ b \in D \\ \epsilon & \text{otherwise} \end{cases}$$

These give rise to two matrices M_b, $b \in \{0, 1\}$. The rows and columns of M_b are indexed by elements of D and $M_b(s, s \circ b) = 1$ if $s \circ b \in D$ and $M_b(s, s') = 0$ if $s \circ b \neq s'$. If $s \circ b \notin D$ then $M_b(s, \epsilon) = 1$ and $\forall s' \neq \epsilon$, $M_b(s, s') = 0$.

Thus, we have defined a morphism, $\Phi : (X^*, \circ) \to (M, \circ)$ which maps $b \to M_b$, $b \in \{0, 1\}$ and by natural extension maps a string $s \in X^*$ to M_s. In particular, the set $S = \{y_1, y_2, ..., y_n\}$ defined in section 6.2.2 is mapped to $\hat{S} = \{M_{y_1}, M_{y_2}, ..., M_{y_n}\}$.

Theorem 6.2.4. *Any circuit over $(M, \circ)$ that takes M_0, M_1 as input and computes $\{M_{y_i} | y_i \in S\}$ at its n output gates is of size $\Omega(\frac{n^2}{\log^2 n})$.*

Proof. Let C be a circuit over $(M, \circ)$ computing M_{y_i}, $1 \leqslant i \leqslant n$ at the n output gates and input M_0, M_1. Consider the corresponding circuit C' over the free monoid X^* with input $X = \{0, 1\}$. Let g_i be the output gate of C computing M_{y_i}, $1 \leqslant i \leqslant n$. In C' let $w_i \in X^*$ be the word computed at g_i. We know that $M_{w_i} = M_{y_i}$ for $1 \leqslant i \leqslant n$. That means $M_{w_i}(\epsilon, y_i) = 1$. By definition of the matrices M_b, the only way this can happen is when $w_i = y_i \circ z_i$ for some $z_i \in X^*$ for each i. Now, let C'' be a new circuit obtained from C' that

outputs the concatenation of $w_1, w_2, ..., w_n$ in that order. Then $size(C^{''}) \leqslant size(C^{'}) + n - 1$. The output string by $C^{''}$ is of the form $y_1 \circ z_1 \circ y_2 \circ z_2 \circ ... \circ y_n \circ z_n$. Since the number of distinct substrings of length $2 \log n$ in $\{y_1, y_2, ..., y_n\}$ we know is $\geqslant \frac{n^2}{\log^2 n}$, it follows by Lemma 6.2.2 that $size(C^{''}) = \Omega(\frac{n^2}{\log^2 n})$. Consequently, $size(C) = size(C^{'}) = \Omega(\frac{n^2}{\log^2 n})$. This completes the proof. □

6.2.4 Circuits over free groups

We consider the free group G_X generated by the set $X = \{x_1, x_2, x_1^{-1}, x_2^{-1}\}$ consisting of x_1, x_2 and their inverses. The group operation is concatenation with the empty string ϵ as identity and the only cancellation rules we can repeatedly use are $x_i x_i^{-1} = x_i^{-1} x_i = \epsilon$ for $i \in \{1, 2\}$. Given a word $w \in X^*$ we can repeatedly apply these rules and obtain a *normal form* $w' \in G_X$ from it which cannot be simplified further. This normal form, by Church-Rosser property, is unique and independent of how we apply the rules.

Recall the set of binary strings we constructed in Section 6.2.2. Replacing 0 by x_1 and 1 by x_2 we obtain $S = \{y_1, y_2, ..., y_n\} \subseteq \{x_1, x_2\}^n \subseteq G_X$. Each word y_i constructed has the property that y_i has $\geqslant \frac{n}{2 \log n}$ distinct subwords of length $2 \log n$. These words are already in their normal forms.

Lemma 6.2.5. *Let $w \in G_X$ be any word where $X = \{x_1, x_2, x_1^{-1}, x_2^{-1}\}$, such that the number of distinct subwords of length l in its normal form w' is N. Then any concatenation circuit for w will require size $\Omega(\frac{N}{l})$ gates.*

Proof. Let C be any circuit that computes the word w. Now each gate g of C computes some word w_g and, as above, $w_g^{'}$ denotes its normal form.

Suppose $g = g_1 \circ g_2$ is a gate whose inputs are gates g_1, g_2. Then, by the Church-Rosser property of cancellations, the normal form for w_g satisfies

$$w'_g = (w'_{g_1} \circ w'_{g_2})'.$$

Suppose w'_{g_1} has k_1 distinct subwords of length l and w'_{g_2} has k_2 distinct subwords of length l. Now, in w'_g notice that the *new* subwords of length l (not occurring in w'_{g_1} or w'_{g_2}) could only arise as a concatenation of some suffix of word w'_{g_1} and prefix of word w'_{g_2} such that neither of them is the empty string. The number of such new subwords is at most l. Hence, w'_g can have at most $k_1 + k_2 + l$ distinct subwords of length l.

Now, since the normal form w' for the output word w has N distinct length l subwords, it follows that number of gates in C is $\Omega(\frac{N}{l})$. □

Now we show that computing $S \subseteq \{x_1, x_2\}^n \subseteq G_X$, defined as above, by any concatenation circuit that takes X as input and outputs S at its output gates will require size $\Omega(\frac{n^2}{\log^2 n})$.

Theorem 6.2.6. *Let $S \subseteq \{x_1, x_2\}^n \subseteq G_X$ be the explicit set of n words defined above. Any concatenation circuit that takes $X = \{x_1, x_2, x_1^{-1}, x_2^{-1}\}$ as input and outputs S at its n output gates will require size $\Omega(\frac{n^2}{\log^2 n})$.*

Proof. Let $S = \{y_1, y_2, ..., y_n\}$ as defined above and let C be any concatenation circuit that takes $X = \{x_1, x_2, x_1^{-1}, x_2^{-1}\}$ as inputs and at its n output gates generates $y_1, y_2, ..., y_n$ respectively. Let C' be a concatenation circuit obtained from C by adding $n-1$ new gates such that C' outputs the concatenation $y = y_1 y_2 ... y_n$. By construction $size(C') = size(C) + n - 1$. The number of distinct length $2\log n$ words in the words y is, by construction, $\geqslant \frac{n^2}{2\log n}$. This is because each y_i has $\geqslant \frac{n}{2\log n}$ distinct subwords and these are disjoint for different y_i. Hence by Lemma 6.2.5, $size(C') = \Omega(\frac{n^2}{\log^2 n})$ which implies $size(C) = \Omega(\frac{n^2}{\log^2 n})$. □

Remark 6.2.1. *Let* $\mathbf{M_0} = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$, $\mathbf{M_1} = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$ *be* 2×2 *matrices. Consider the infinite group G generated by these elements and their inverses over the field of rationals* $\mathbb{Q}$. *It is well known (e.g. see [LZ77] for a nice complexity theoretic application) that the group G is isomorphic to the free group G_X, where the isomorphism is defined by $x_1 \to M_0$ and $x_2 \to M_1$. It follows that Theorem 6.2.6 also applies to the group G by setting $x_1 = M_0$*

and $x_2 = M_1$.

6.2.5 Circuits over permutation groups

We now present a lower bound in the setting of groups. We will transform our free monoid construction to this setting. Recall the set of binary strings S we constructed in Section 6.2.2. To this end, we will define two permutations $\pi_0, \pi_1 \in S_N$ (where $N = poly(n)$ will be defined later). These permutations correspond to $X = \{0, 1\}$ and by multiplication the target output permutations are defined:

$G_S = \{\pi_{y_i} = \Pi_{j=1}^{n} \pi_{y_i[j]} | y_i \in S\}$, where $y_i[j]$ is the j-th bit of string y_i.

Definition of π_0, π_1:

We pick r primes $p_1, p_2, ..., p_r$ where $r = n^2$ such that $n < p_1 < p_2 < ... < p_r < n^4$. The permutation π_0 is defined as the product of $r + 1$ disjoint cycles, $\pi_0 = C_0.C_1...C_r$ where C_0, C_1 are of length p_1 and for $i \geqslant 2$, C_i is of length p_i. Similarly, $\pi_1 = C_0^{'}.C_1^{'}...C_r^{'}$ is a product of $r + 1$ disjoint cycles with $C_0^{'}$ and $C_1^{'}$ of length p_1 and for $i \geqslant 2$, $C_i^{'}$ is of length p_i. Let supp(C) denote the set of points moved by C for a cycle C (i.e., if we write $C = (i_1 i_2 ... i_p)$ it means C maps i_1 to i_2 and so on i_p to i_1 and moves no other element of the domain. Hence, supp(C) = $\{i_1, i_2, ..., i_p\}$). In the construction above we pick the cycles C_i and $C_i^{'}$, $0 \leqslant i \leqslant r$ such that supp(C_0) $\cap$ supp($C_0^{'}$) = $\{1\}$ and $\forall (i, j) \neq (0, 0)$ supp(C_0) $\cap$ supp($C_0^{'}$) = ϕ. The domain $[N]$ on which these permutations are defined is $\bigcup_{i=0}^{r}$(supp(C_i) $\cup$ supp($C_i^{'}$)). Note that $N \leqslant 4p_1 + 2\sum_{i=2}^{r} p_i = O(n^6)$. Thus, the problem we consider is that of designing a circuit over S_N that takes as input x_0, x_1 and outputs at the n output gates $\pi_{y_i} = \Pi_{j=1}^{n} x_{y_i[j]}$ where $y_i[j]$ is the j-th bit of string y_i for each $y_i \in S$.

Theorem 6.2.7. *Any circuit over the group* $(S_N, \circ)$ *that takes as input* π_0, π_1 *and computes* $G_S = \{\pi_{y_i} | y_i \in S\}$ *as output is of size* $\Omega(\frac{n^2}{\log^2 n})$.

Proof. Let C be the circuit that solves this problem of computing G_S from x_0, x_1. We fix the input as $x_0 = \pi_0$ and $x_1 = \pi_1$. Now, consider the corresponding concatenation circuit $C^{'}$ with input $x_0, x_1 \in X$. At each output gate g_i, $1 \leqslant i \leqslant m$, circuit $C^{'}$ computes some word $w_i \in X^*$ such that $\forall i, \pi_{w_i} = \pi_{y_i}$ where π_{w_i} is the permutation in S_N obtained by putting $x_0 = \pi_0$ and $x_1 = \pi_1$ in w_i. If $w_i = y_i$ for all i then, in fact $C^{'}$ as a concatenation circuit computes the set S at its output gates. This implies by Theorem 6.2.3 that $size(C^{'}) = \Omega(\frac{n^2}{\log^2 n})$ and $size(C) = \Omega(\frac{n^2}{\log^2 n})$.

Suppose $w_i \neq y_i$ at some output gate g_i. We can write $w_i = u \circ b_2 \circ s$ and $y_i = v \circ b_1 \circ s$ where $b_1 \neq b_2$. Assume, without loss of generality, that $b_1 = 0$ and $b_2 = 1$. Since $\pi_{w_i} = \pi_{y_i}$, we know $\pi_u \pi_{b_2} \pi_s = \pi_v \pi_{b_1} \pi_s$ (i.e., $\pi_u \pi_1 \pi_s = \pi_v \pi_0 \pi_s$). Let $\alpha \in [N]$ such that $\pi_s(\alpha) = 1$. In $\pi_{y_i} = \pi_v \pi_0 \pi_s$, the permutation π_0 will map 1 to $\beta \in C_0 \backslash \{1\}$, whereas in $\pi_{w_i} = \pi_u \pi_1 \pi_s$ the permutation π_1 maps 1 to $\gamma \in C_0^{'} \backslash \{1\}$. Since $|v| < n$ the point β cannot be moved back to 1 and subsequently to $C_0^{'} \backslash \{1\}$. This is because $p_1 > n$ and length of cycle $C_0^{'}$ is p_1. Therefore by π_{y_i} the point α is mapped to some point in $C_0 \backslash \{1\}$. Since π_{w_i} must map α to the same point and $\pi_1 \pi_s$ has mapped α to a point in $\gamma \in C_0^{'} \backslash \{1\}$, π_u must have at least $p_1 > n$ occurrences of π_1 in it to move γ to 1 and subsequently to the final point in $C_0 \backslash \{1\}$ (using some π_0 applications). We will now argue that this forces w_i to be a long string.

Pick any tuple of points $(\alpha_1, \alpha_2, ..., \alpha_r)$ where $\alpha_i \in C_i^{'}$, $1 \leqslant i \leqslant r$. Notice that only π_1 moves this tuple because α_i, $1 \leqslant i \leqslant r$ do not belong to $\text{supp}(\pi_0)$. Since $p_1, ..., p_r$ are distinct primes, the permutation π_1 maps $(\alpha_1, \alpha_2, ..., \alpha_r)$ to a set of $\Pi_{i=1}^{r} p_i - 1$ distinct r-tuples before returning to $(\alpha_1, \alpha_2, ..., \alpha_r)$. Suppose there are l occurrences of π_1 in π_{y_i}, $l < n$. Thus, if $\pi_{y_i}(\alpha_1, \alpha_2, ..., \alpha_r) = (\beta_1, \beta_2, ..., \beta_r)$ then $\pi_1^l(\alpha_1, \alpha_2, ..., \alpha_r) = (\beta_1, \beta_2, ..., \beta_r)$. Then $\pi_{w_i}(\alpha_1, \alpha_2, ..., \alpha_r) = (\beta_1, \beta_2, ..., \beta_r)$. However we know number of occurrences of π_1 in π_{w_i} is some $k > n^2$ which means $\pi_{w_i}(\alpha_1, \alpha_2, ..., \alpha_r) = (\beta_1, \beta_2, ..., \beta_r) = \pi_1^k(\alpha_1, \alpha_2, ..., \alpha_r)$.

It follows that $\pi_1^{k-l}(\alpha_1, \alpha_2, ..., \alpha_r) = (\alpha_1, \alpha_2, ..., \alpha_r)$ which implies $k - l$ is a multiple of $\Pi_{i=1}^{r} p_i$. Hence $|w_i| \geqslant \Pi_{i=1}^{r} p_i$. This implies that the circuit needs at least $\log \Pi_{i=1}^{r} p_i$ multiplication gates to compute w_i. This gives, $size(C) \geqslant \log \Pi_{i=1}^{r} p_i \geqslant \log 2^{n^2} = n^2$.

Putting it together $size(C) = \Omega(\frac{n^2}{\log^2 n})$ in any case. This completes the proof. □

6.3 Lower bounds for Linear Circuits over Rings

In this section we consider a generalization of the linear circuits model. In this generalization we allow the coefficients to come from *noncommutative rings*. In principle, we can expect lower bounds could be easier to prove in this model. The circuits are more constrained when coefficients come from a noncommutative ring as fewer cancellations can take place. This is in the same spirit as Nisan's [Nis91] work on lower bounds for noncommutative algebraic branching programs. However, we succeed in showing only some limited lower bounds. We leave open problems that might be more accessible than the notorious problems for linear circuits over fields.

6.3.1 Preliminaries

We start with the definition of linear circuits over arbitrary ring R.

Definition 6.3.1. *Let $(R, +, \cdot)$ be an arbitrary ring (possibly noncommutative). A* linear circuit *over R takes n inputs $y_1, y_2, \ldots, y_n$ labeling the indegree 0 nodes of a directed acyclic graph. The circuit has m output nodes (which have outdegree 0). Each edge of the graph is labeled by some element of the ring R. The indegree of each non-input node is two. Each node of the circuit computes a linear form $\sum_{i=1}^{n} \alpha_i y_i$ for $\alpha_i \in R$ as follows: the input node labeled y_i computes y_i. Suppose g is a node with incoming edges from nodes g_1 and g_2, and the edges (g_1, g) and (g_2, g) are labeled by α and β respectively. If g_1 and g_2 computes the linear forms ℓ_1 and ℓ_2 respectively, then g computes $\alpha\ell_1 + \beta\ell_2$. Thus, for an $m \times n$ matrix A over the ring R, the circuit computes $A\mathbf{y}$ at the m output gates.*

When R is a field we get the well-studied linear circuits model [Mor73, Val77, Lok09]. However, no explicit superlinear size lower bounds are known for this model over fields

(except for some special cases like the bounded coefficient model [Mor73] or in the cancellation free case [BF13]).

6.3.2 Our Results

When the coefficients to come from a *noncommutative* ring R we prove lower bounds for certain restricted linear circuits. Suppose the coefficient ring is $R = \mathbb{F}\langle x_0, x_1\rangle$ consisting of polynomials over the field $\mathbb{F}$ in noncommuting variables x_0 and x_1.

Let $M \in \mathbb{F}^{n\times n}\langle x_0, x_1\rangle$ where x_0, x_1 are noncommuting variables and $Y = (y_1, y_2, \ldots, y_n)^T$ is a column vector of input variables. We show the following results.

1. There is a $M \in \mathbb{F}^{n\times n}\langle x_0, x_1\rangle$ such that any homogeneous linear circuit C over the coefficient ring $\mathbb{F}\langle x_0, x_1\rangle$ computing MY requires either size $\omega(n)$ or depth $\omega(\log n)$. We prove this by suitably generalizing Valiant's matrix rigidity method [Val77] as explained below.

2. We next consider homogeneous depth 2 linear circuits. We show that for the explicit matrix M as defined above, computing MY by a depth 2 homogeneous linear circuit (with unbounded fanin) requires $\Omega(\frac{n^2}{\log n})$ wires.

3. We show that any linear circuit over $\mathbb{F}\langle x_0, x_1\rangle$ computing MY, where edge labels are restricted to be constant-degree polynomials, requires size $\Omega(\frac{n^2}{\log n})$.

4. We show that any linear circuit, whose edge labels are restricted to be either a homogeneous degree $4 \log n$ polynomial or a scalar, computing MY requires $\Omega(n^2)$ size, where M is the palindrome matrix.

6.3.3 Lower bounds for homogeneous linear circuits

The first restriction we consider are *homogeneous* linear circuits over the ring $\mathbb{F}\langle x_0, x_1\rangle$ for computing MY. The restriction is that for every gate g in the circuit, if g has its two incoming edges from nodes g_1 and g_2, then the edges (g_1, g) and (g_2, g) are labeled by α and β respectively, where $\alpha, \beta \in \mathbb{F}\langle x_0, x_1\rangle$ are restricted to be *homogeneous polynomials* of same degree in the variables x_0 and x_1. It follows, as a consequence of this restriction, that each gate g of the circuit computes a linear form $\sum_{i=1}^{n} \alpha_i y_i$, where the $\alpha_i \in \mathbb{F}\langle x_0, x_1\rangle$ are all homogeneous polynomials of the same degree. Our goal is to construct an explicit matrix $M \in \mathbb{F}^{n\times n}\langle x_0, x_1\rangle$ such that MY can not be computed by any circuit C with size $O(n)$ and depth $O(\log n)$. We prove this by suitably generalizing Valiant's matrix rigidity method [Val77] as explained below.

Consider $n \times n$ matrices $\mathbb{F}^{n\times n}$ over field $\mathbb{F}$. The *support* of a matrix $A \in \mathbb{F}^{n\times n}$ is the set of locations $\text{supp}(A) = \{(i, j) \mid A_{ij} \neq 0\}$.

Definition 6.3.2. *Let $\mathbb{F}$ be any field. The rigidity $\rho_r(\mathcal{A})$ of a deck of matrices $\mathcal{A} = \{A_1, A_2, \ldots, A_N\} \subseteq \mathbb{F}^{n\times n}$ is the smallest number t for which there are a set of t positions $S \subseteq [n]\times[n]$ and a deck of matrices $\mathcal{B} = \{B_1, B_2, \ldots, B_N\}$ such that for all i:* $\text{supp}(B_i) \subseteq S$ *and the rank of $A_i + B_i$ is bounded by r. A collection $\mathcal{A} = \{A_1, A_2, \ldots, A_N\} \subseteq \mathbb{F}^{n\times n}$ is a* rigid deck *if $\rho_{\epsilon\cdot n}(\mathcal{A}) = \Omega(n^{2-o(1)})$.*

Notice that for $N = 1$ this is precisely the notion of rigid matrices. We are interested in constructing *explicit* rigid decks: I.e. a deck $\mathcal{A}$ such that for each $k \in [N]$ and each $1 \leq i, j \leq n$ there is a polynomial (in n) time algorithm that outputs the $(i, j)^{th}$ entry of A_k. We describe an explicit deck of size $N = 2^{n^2}$ over any field $\mathbb{F}$ and use it to prove our first lower bound result. It is convenient write the deck as $\mathcal{A} = \{A_m \mid m \in \{x_0, x_1\}^{n^2}\}$ with matrices A_m indexed by monomials m of degree n^2 in the noncommuting variables x_0 and x_1. The matrix A_m is defined as follows:

$$A_m[i,j] = \begin{cases} 1 & \text{if } m_{ij} = x_1 \\ 0 & \text{if } m_{ij} = x_0 \end{cases}$$

Note that all the matrices A_m in the deck $\mathcal{A}$ are in $\mathbb{F}^{n\times n}$. Clearly, $\mathcal{A}$ is an explicit deck. We prove that it is a rigid deck.

Lemma 6.3.3. *The deck $\mathcal{A} = \{A_m \mid m \in \{x_0, x_1\}^{n^2}\}$ is an explicit rigid deck for any field $\mathbb{F}$.*

Proof. Valiant [Val77] showed that almost all $n \times n$ 0-1 matrices over any field $\mathbb{F}$ have rigidity $\Omega(\frac{(n-r)^2}{\log n})$ for target rank r. In particular, for $r = \epsilon \cdot n$, over any field $\mathbb{F}$, there is a 0-1 matrix R for which we have $\rho_r(R) \geq \frac{\delta \cdot n^2}{\log n}$ for some constant $\delta > 0$ depending on ϵ.

We claim that for the deck $\mathcal{A}$ we have $\rho_{\epsilon n}(\mathcal{A}) \geq \frac{\delta \cdot n^2}{\log n}$. To see this, suppose let $E = \{E_m \in \mathbb{F}^{n\times n} | m \in \{x_0, x_1\}^{n^2}\}$ be any collection of matrices such that $|\text{supp}(E_m)| \leqslant \frac{\delta n^2}{\log n}$ for each m. Since the deck $\mathcal{A}$ contains all 0-1 matrices, in particular $R \in \mathcal{A}$ and $R = A_m$ for some monomial m. From the rigidity of R we know that the rank of $R + E_m$ is at least ϵn. This proves the claim and the lemma follows. □

We now turn to the lower bound result for homogeneous linear circuits where the coefficient ring is $\mathbb{F}\langle x_0, x_1\rangle$. We define an explicit $n \times n$ matrix M, whose (i, j)-th entry M_{ij} is the polynomial,

$$M_{ij} = (x_0 + x_1)^{(i-1)n+j-1} \cdot x_1 \cdot (x_0 + x_1)^{n^2-((i-1)n+j)}. \tag{6.1}$$

It is easy to see that we can express the matrix M as $M = \sum_{m\in\{x_0,x_1\}^{n^2}} A_m m$, where $\mathcal{A} = \{A_m \mid m \in \{x_0, x_1\}^{n^2}\}$ is the deck defined above.

Theorem 6.3.4. *Any homogeneous linear circuit C over the coefficient ring $\mathbb{F}\langle x_0, x_1\rangle$ computing MY, for M defined above, requires either size $\omega(n)$ or depth $\omega(\log n)$.*

Proof. Assume to the contrary that C is a homogeneous linear circuit of size $O(n)$ and

depth $O(\log n)$ computing MY. We know that by Valiant's graph-theoretic argument (see e.g. [Lok09]) that in the circuit C there is a set of gates V of cardinality $s = \frac{c_1 n}{\log\log n} = o(n)$ such that at least $n^2 - n^{1+\delta}$, for $\delta < 1$, input-output pairs have all their paths going through V. Thus, we can write $M = B_1 B_2 + E$ where $B_1 \in \mathbb{F}^{n\times s}\langle x_0, x_1\rangle$ and $B_2 \in \mathbb{F}^{s\times n}\langle x_0, x_1\rangle$ and $E \in \mathbb{F}^{n\times n}\langle x_0, x_1\rangle$, and $|\text{supp}(E)| \leq n^{1+\delta}$. By collecting the matrix coefficient of each monomial we can express M and E as

$$M = \sum_{m\in\{x_0,x_1\}^{n^2}} A_m m, \text{ and } E = \sum_{m\in\{x_0,x_1\}^{n^2}} E_m m,$$

where A_m are already defined and $|\cup_{m\in\{x_0,x_1\}^{n^2}} \text{supp}(E_m)| \leq n^{1+\delta}$. Now consider the matrix $B_1 B_2$. By collecting matrix coefficients of monomials we can write $B_1 B_2 = \sum_{m\in\{x_0,x_1\}^{n^2}} B_m m$.

We now analyze the matrices B_m. Crucially, by the homogeneity condition on the circuit C, we can partition $V = V_1 \cup V_2 \cup \ldots V_\ell$, where each gate g in V_i computes a linear form $\sum_{j=1}^{n} \gamma_j y_j$ and $\gamma_j \in \mathbb{F}\langle x_0, x_1\rangle$ is a homogeneous degree d_i polynomial. Let $s_i = |V_i|, 1 \leq i \leq \ell$. Then we have $s = s_1 + s_2 + \ldots s_\ell$. Every monomial m has a unique prefix of length d_i for each degree d_i associated with the gates in V. Thus, we can write $B_m = \sum_{j=1}^{\ell} B_{m,j,1} B_{m,j,2}$, where $B_{m,j,1}$ is the $n \times s_j$ matrix corresponding to the d_j-prefix of m and $B_{m,j,2}$ is the $s_j \times n$ matrix corresponding to the $n^2 - d_j$-suffix of m. It follows that for each monomial m the rank of B_m is bounded by s. Putting it together, for each monomial m we have $A_m = B_m + E_m$, where B_m is rank s and $|\cup_{m\in\{x_0,x_1\}^{n^2}} \text{supp}(E_m)| \leq n^{1+\delta}$. This contradicts the fact that $\mathcal{A}$ is a rigid deck. □

Remark 6.3.1. *For the matrix $M = (M_{ij})$, as defined above, it does not seem that Shoup-Smolensky dimension method [SS96] can be used to prove a similar lower bound. To see this, suppose $\Gamma_M(n)$ is the set of all monomials of degree n in $\{m_{ij}\}$ and let $D_M(n)$ be the dimension of the vector space over $\mathbb{F}$ spanned by the set $\Gamma_M(n)$. The upper bound for $D_M(n)$ that we can show for a depth d and size $O(n)$ linear circuit over the ring $\mathbb{F}\langle x_0, x_1\rangle$ is as large as $(\frac{O(n)}{d})^{dn}$. This bound, unfortunately, is much larger than the bounds*

obtainable for the commutative case [SS96]. On the other hand, the lower bound for $D_M(n)$ is only $n^{\Theta(n)}$. Thus, we do not get a superlinear size lower bound for the size using Shoup-Smolensky dimensions when the coefficient ring is $\mathbb{F}\langle x_0, x_1\rangle$.

6.3.4 Lower bounds for homogeneous depth 2 linear circuits

We next consider homogeneous depth 2 linear circuits. These are linear circuits of depth 2, where each addition gate can have unbounded fanin. More precisely, if g is an addition gate with inputs from $g_1, g_2, \ldots, g_t$ then the gate g computes $\sum_{i=1}^{t} \alpha_i g_i$, where each edge (g_i, g) is labeled by $\alpha_i \in \mathbb{F}\langle x_0, x_1\rangle$ such that $\alpha_i, 1 \leq i \leq t$ are all homogeneous polynomials of the same degree. We again consider the problem of computing MY for $M \in \mathbb{F}^{n\times n}\langle x_0, x_1\rangle$. The goal is to lower bound the number of wires in the linear circuit. This problem is also well studied for linear circuits over fields and only an explicit $\Omega(n \log^2 n/\log\log n)$ lower bound is known for it [Lok09, Pud94], although for random matrices the lower bound is $\Omega(n^2/\log n)$.

We show that for the explicit matrix M as defined above, computing MY by a depth 2 homogeneous linear circuit (with unbounded fanin) requires $\Omega(\frac{n^2}{\log n})$ wires.

Theorem 6.3.5. *Let $M \in \mathbb{F}_2^{n\times n}\langle x_0, x_1\rangle$ as defined in Equation 6.1. Any homogeneous linear circuit C of depth 2 computing MY requires $\Omega(\frac{n^2}{\log n})$ wires.*

Proof. Let C be a homogeneous linear circuit of depth 2 computing MY. Let $w(C)$ denote the number of wires in C. Let s be the number of gates in the middle layer of C. We can assume without loss of generality that, all input to output paths in C are of length 2 and hence pass through the middle layer. A *level 1* edge connects an input gate to a middle-layer gate and a *level 2* edge is from middle layer to output. Thus, we can factorize $M = M' * M''$ where the matrix M' is in $\mathbb{F}^{n\times s}\langle x_0, x_1\rangle$ and M'' is in $\mathbb{F}^{s\times n}\langle x_0, x_1\rangle$, and the complexity of C is equivalent to total number of nonzero entries in M' and M''. As before, write $M = \sum_{m\in\{x_0,x_1\}^{n^2}} A_m m$.

Given A_m for $m \in \{x_0, x_1\}^{n^2}$, we show how to extract from C a depth-2 linear circuit over the *field* $\mathbb{F}$, call it $C^{(m)}$, that computes A_m such that the number of wires in $C^{(m)}$ is at most the number of wires in C. Indeed, we do not add any new gate or wires in obtaining $C^{(m)}$ from C.

For each gate g in the middle layer, there are at most n incoming edges and n outgoing edges. As C is a homogeneous linear circuit we can associate a degree d_g to gate g. Each edge (i, g) to g is labeled by a homogeneous degree-d_g polynomial $\alpha_{i,g}$ in $\mathbb{F}\langle x_0, x_1\rangle$. Likewise, each edge (g, j) from g to the output layer is labeled by a degree-$n^2 - d_g$ homogeneous polynomial $\beta_{g,j}$. Let $m = m_1 m_2$, where m_1 is of degree d_g and m_2 of degree $n^2 - d_g$. For each incoming edge (i, g) to g we keep as label the coefficient of the monomial m_1 in $\alpha_{i,g}$ and for outgoing edge (g, j) from g we keep as label the coefficient of the monomial m_2 in $\beta_{g,j}$. We do this transformation for each gate g in the middle layer of C. This completes the description of the depth-2 circuit $C^{(m)}$. By construction it is clear that $C^{(m)}$ computes A_m and the number of wires $w(C^{(m)})$ in $C^{(m)}$ is bounded by $w(C)$ for each monomial $m \in \{x_0, x_1\}^{n^2}$. However, $\{A_m \mid m \in \{x_0, x_1\}^{n^2}\}$ is the set of all 0-1 matrices over $\mathbb{F}$ and it is known that there are $n \times n$ 0-1 matrices A_m such that any depth-2 linear circuit for it requires $\Omega(\frac{n^2}{\log n})$ wires (e.g. see [Lok09]). Hence, the number of wires in C is $\Omega(\frac{n^2}{\log n})$. □

6.3.5 More Lower bounds for homogeneous linear circuits

If we restrict the edge labels in the linear circuit computing MY to only *constant-degree polynomials* then we can obtain much stronger lower bounds using Nisan's [Nis91] lower bound technique for noncommutative algebraic branching programs. We can define the matrix M as follows. Let $M_{ij} = w_{ij} w_{ij}^R$, where $w_{ij} \in \{x_0, x_1\}^{2 \log n}$ and $1 \leqslant i, j \leqslant n$ are all distinct monomials of degree $2 \log n$. We refer to M as a palindrome matrix. All entries of M are distinct and note that each entry of MY can be computed using $O(n \log n)$ gates.

Theorem 6.3.6. *Any linear circuit over $\mathbb{F}\langle x_0, x_1\rangle$ computing MY, where edge labels are*

restricted to be constant-degree polynomials, requires size $\Omega(\frac{n^2}{\log n})$.

Proof. Let C be such a linear circuit computing MY. Since edges can be labeled by constant-degree polynomials, we can first obtain a linear circuit C' computing MY such that each edge is labeled by a *homogeneous linear form*. The size $\text{size}(C') = O(\text{size}(C) \log n)$. From C', we can obtain a noncommutative algebraic branching program $\hat{C}$ that computes the palindrome polynomial $\sum_{w \in \{x_0, x_1\}^{2 \log n}} ww^R$ such that $\text{size}(\hat{C}) = O(\text{size}(C'))$. By Nisan's lower bound [Nis91] $\text{size}(\hat{C}) = \Omega(n^2)$, which implies $\text{size}(C) = \Omega(\frac{n^2}{\log n})$. □

If we restrict edge labels to either homogeneous degree $4 \log n$ polynomial or a scalar then we can show the following lower bound.

Theorem 6.3.7. *Any linear circuit, whose edge labels are restricted to be either a homogeneous degree* $4 \log n$ *polynomial or a scalar, computing* MY *requires* $\Omega(n^2)$ *size, where* M *is the palindrome matrix. Moreover, there is a matching upper bound.*

Proof. Let C be any linear circuit computing MY. Each entry m_{ij} of the matrix M can be written as sum of products of polynomials $m_{ij} = \sum_{\rho_{ij}} \prod_{e \in \rho_{ij}} l(e)$ where ρ_{ij} is a path from input y_j to output gate i in C and $l(e)$ is the label of edge e in C. Let S be set of all edge labels in C with degree $4 \log n$ polynomial. Thus, each m_{ij} is a linear combinations of elements in the set S over $\mathbb{F}$. This implies that $m_{ij} \in S\,pan(S)$ where $i \leqslant i, j \leqslant n$. Since all m_{ij} are distinct, $|S| \geqslant n^2$. Since fan in is 2, $size(C) \geqslant \frac{n^2}{2} = \Omega(n^2)$.

For upper bound, we use n^2 edges (n edges starting from each input y_i) each labeled by a corresponding monomial in M (of degree $4 \log n$) and then we add relevant edges to get the output gates. Thus, upper bound is $O(n^2)$ for computing MY. □

Note that, since we have not used noncommutativity in the proof, Theorem 6.3.7 also holds in the commutative settings (we require all the entries of M to be distinct).

6.4 Discussion

For multiplicative circuits we could prove lower bounds only for large monoids and large groups. The main question here is whether we can prove lower bounds for an explicit function $f : S^n \to S^m$, for some constant size nonabelian group or monoid S.

We introduced the notion of rigidity for decks of matrices, but the only explicit example we gave was the trivial one with a deck of size 2^{n^2}. A natural question is to give explicit constructions for smaller rigid decks of $n \times n$ matrices, say of size $n!$ or less. Or is the construction of rigid decks of smaller size equivalent to the original matrix rigidity problem?

Bibliography

[AFS+16] Matthew Anderson, Michael A. Forbes, Ramprasad Saptharishi, Amir Shpilka, and Ben Lee Volk. Identity testing and lower bounds for read-k oblivious algebraic branching programs. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 30:1–30:25, 2016.

[AJR16] Vikraman Arvind, Pushkar S. Joglekar, and S. Raja. Noncommutative valiant's classes: Structure and complete problems. *ACM Transactions on Computation Theory (ToCT)*, 9(1):3:1–3:29, 2016.

[AJS09] Vikraman Arvind, Pushkar S. Joglekar, and Srikanth Srinivasan. Arithmetic circuits and the hadamard product of polynomials. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2009, December 15-17, 2009, IIT Kanpur, India*, pages 25–36, 2009.

[AR] Vikraman Arvind and S. Raja. Some lower bound results for set-multilinear arithmetic computations. *Chicago Journal of Theoretical Computer Science*, 2016(6).

[ARS14] Vikraman Arvind, S. Raja, and A. V. Sreejith. On lower bounds for multiplicative circuits and linear circuits in noncommutative domains. In *Computer Science - Theory and Applications - 9th International Computer Sci-*

ence Symposium in Russia, CSR 2014, Moscow, Russia, June 7-11, 2014. Proceedings, pages 65–76, 2014.

[AS10] Vikraman Arvind and Srikanth Srinivasan. On the hardness of the noncommutative determinant. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 677–686, 2010.

[BÖ00] Peter Bürgisser. The computational complexity of immanants. 30(3):1023–1040, 2000. Preliminary version in FPSAC'98.

[Bar90] Alexander I. Barvinok. Computational complexity of immanents and representations of the full linear group. *Functional Analysis and its Applications*, 24(2):144–145, 1990.

[BBB+00] Amos Beimel, Francesco Bergadano, Nader H. Bshouty, Eyal Kushilevitz, and Stefano Varricchio. Learning functions represented as multiplicity automata. *Journal of the ACM*, 47(3):506–530, 2000.

[Ber84] Stuart J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Inf. Process. Lett.*, 18(3):147–150, 1984.

[BF13] Joan Boyar and Magnus Gausdal Find. Cancellation-free circuits in unbounded and bounded depth. In *FCT*, pages 159–170, 2013.

[BR11] J. Berstel and C. Reutenauer. *Noncommutative Rational Series with Applications*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2011.

[BS83] Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theor. Comput. Sci.*, 22:317–330, 1983.

[Bür99] Peter Bürgisser. On the structure of valiant's complexity classes. *Discrete Mathematics & Theoretical Computer Science*, 3(3):73–94, 1999.

[Bür00a] Peter Bürgisser. *Completeness and reduction in algebraic complexity theory*. Algorithms and computation in mathematics. Springer, Berlin, New York, 2000.

[Bür00b] Peter Bürgisser. The computational complexity to evaluate representations of general linear groups. 30(3):1010–1022, 2000. Preliminary version in FPSAC'98.

[CS63] Noam Chomsky and Marcel Paul Schützenberger. The Algebraic Theory of Context-Free Languages. In P. Braffort and D. Hirshberg, editors, *Computer Programming and Formal Systems*, Studies in Logic, pages 118–161. North-Holland Publishing, 1963.

[DP09] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[DSW94] Martin D. Davis, Ron Sigal, and Elaine J. Weyuker. *Computability, Complexity, and Languages (2Nd Ed.): Fundamentals of Theoretical Computer Science*. Academic Press Professional, Inc., San Diego, CA, USA, 1994.

[FS13] Michael A. Forbes and Amir Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 243–252, 2013.

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

[Har85] Werner Hartmann. On the complexity of immanants. *Linear and Multilinear Algebra*, 18(2):127–140, 1985.

[Hep94] Charles Thomas Hepler. The complexity of computing characters of a finite group. 1994.

[HWY10a] Pavel Hrubes, Avi Wigderson, and Amir Yehudayoff. Non-commutative circuits and the sum-of-squares problem. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 667–676, 2010.

[HWY10b] Pavel Hrubes, Avi Wigderson, and Amir Yehudayoff. Relationless completeness and separations. In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, June 9-12, 2010*, pages 280–290, 2010.

[Hya79] Laurent Hyafil. On the parallel evaluation of multivariate polynomials. *SIAM J. Comput.*, 8(2):120–123, 1979.

[JS13] Stasys Jukna and Igor Sergeev. Complexity of linear boolean operators. *Foundations and Trends in Theoretical Computer Science*, 9(1):1–123, 2013.

[Lad75] Richard E. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22(1):155–171, 1975.

[LMS15] Nutan Limaye, Guillaume Malod, and Srikanth Srinivasan. Lower bounds for non-commutative skew circuits. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:22, 2015.

[Lok09] Satyanarayana V Lokam. Complexity lower bounds using linear algebra. *Foundations and Trends in Theoretical Computer Science*, 4(1-2):1–155, 2009.

[LZ77] Richard J Lipton and Yechezkel Zalcstein. Word problems solvable in logspace. *Journal of the ACM (JACM)*, 24(3):522–526, 1977.

[MM13] Stephan Mertens and Cristopher Moore. The complexity of the fermionant and immanants of constant width [note]. *Theory of Computing*, 9:273–282, 2013.

[Mor73] Jacques Morgenstern. Note on a lower bound on the linear complexity of the fast fourier transform. *Journal of the ACM (JACM)*, 20(2):305–306, 1973.

[Nis91] Noam Nisan. Lower bounds for non-commutative computation (extended abstract). In *STOC*, pages 410–418, 1991.

[NW95] Noam Nisan and Avi Wigderson. Lower bounds for arithmetic circuits via partial serivatives (preliminary version). In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, 23-25 October 1995*, pages 16–25, 1995.

[Pud94] P Pudlak. Large communication in constant depth circuits. *Combinatorica*, 14(2):203–216, 1994.

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

[RY08] Ran Raz and Amir Yehudayoff. Balancing syntactically multilinear arithmetic circuits. *Computational Complexity*, 17(4):515–535, 2008.

[Sha00] D. B. Shapiro. *Composition of Quadratic Forms*. W. de Gruyter Verlag, 2000.

[SS96] Victor Shoup and Roman Smolensky. Lower bounds for polynomial evaluation and interpolation problems. *Computational Complexity*, 6(4):301–311, 1996.

[Str69] Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, 1969.

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

[Tod92] Seinosuke Toda. Classes of arithmetic circuits capturing the compl exity of computing the determinant. In *IEICE Transactions on Informations and Systems,E75-D , 116âĂŞ124, 1992*, pages 116–124, 1992.

[Val77] Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *MFCS*. Springer, 1977.

[Val79a] Leslie G. Valiant. Completeness classes in algebra. In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 249–261, 1979.

[Val79b] Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

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