Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 124 (2015)

Noncommutative Valiant's Classes: Structure and Complete Problems

V. Arvind[*] Pushkar S Joglekar[†] S. Raja[‡]

April 29, 2016

Abstract

In this paper 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.
- 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} (analogous to Ladner's theorem [Lad75]).
- 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 explicitly in this paper).

1 Introduction

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 [Val79], 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, $\mathrm{VP} \neq \mathrm{VNP}$ 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

[*]Institute of Mathematical Sciences, Chennai, India, email: arvind@imsc.res.in
[†]Vishwakarma Institute of Technology, Pune, India, email: joglekar.pushkar@gmail.com
[‡]Institute of Mathematical Sciences, Chennai, India, email: rajas@imsc.res.in

ISSN 1433-8092

complete for the class $\mathrm{VNP}_{\mathbb{F}}$ for any field $\mathbb{F}$ of characteristic different from 2. (Note: In this paper, 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 $X = \{x_{ij}\}_{1\le i,j\le n}$, is

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

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 [Val79], which allows variables or scalars to be substituted for variables).

The purpose of our paper is a closer study of the structure of the classes VP_{nc} and VNP_{nc} and its connections to formal language classes. Our main results show a rich structure within VNP_{nc} and VP_{nc} which nicely corresponds to properties of regular languages and context-free languages.

1.1 Main results of the paper

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

Definition 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 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 $g = (g_n) \in \mathrm{VP}_{nc}$ 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 paper, that this analogy goes further and shows up in the internal structure of VNP_{nc} and VP_{nc}.

1. We prove that the Dyck polynomials are complete for VP_{nc} w.r.t $\leq_{abp}$ 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 $\leq_{abp}$ 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 3 has the formal definitions and a discussion on this reducibility.

2. On the same lines 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 $\leq_{abp}$ reducibility, again by adapting the proof of the Chomsky-Schützenberger theorem.

3. 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.

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, 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 $\leq_{proj}$ reductions between VP_{nc} and VNP_{nc} and also incomparable p-families. To the best of our knowledge, obtaining an infinite hierarchy is open in the commutative case.

5. Within VP_{nc} we obtain a proper hierarchy w.r.t $\leq_{abp}$-reductions corresponding to the Dyck polynomials of bounded nesting depth. This roughly corresponds to the noncommutative VNC hierarchy within VP_{nc}.

P-family	Complexity Result	Remarks
$D_k, k \geq 2$	- VP$_{nc}$-Complete (Theorem 13) - VSKEW$_{nc}$-hard	w.r.t $\leq_{abp}$-reductions
PAL$_d$	VSKEW$_{nc}$-Complete (Theorem 19)	w.r.t. $\leq_{abp}$-reductions
ID_d	-not VNP$_{nc}$-Complete (Theorem 27) -not in VP$_{nc}$ [HWY10a]	$\leq_{proj}, \leq_{iproj}$-reductions assuming SOS_k conjecture
$f^{(i+1)}, i \geq 1$	VNP$_{nc}$-intermediate (Theorem 33)	$\leq_{proj}, \leq_{iproj}$-reductions - not reducible to $f^{(i)}$ assuming VP$_{nc} \neq$ VNP$_{nc}$
PER$^{*,\mathcal{X}}$	VNP$_{nc}$-Complete (Theorem 38)	w.r.t. $\leq_{abp}$-reductions
ID_n^*	VNP$_{nc}$-Complete (Theorem 28)	w.r.t. $\leq_{abp}$-reductions

Table 1: Summary of Results

Table 1 summarizes the results in this paper.

Organization: Section 2 contains some preliminary definitions. In Section 3 we define and compare the different reducibilities considered in this paper. In Section 4 we show the VP$_{nc}$-completeness of Dyck polynomials, and in Section 5 we show the VSKEW$_{nc}$-completeness of Palindrome polynomials. Section 6 contains our results analogous to Ladner's theorem. Sections 7 and 8 contain some more observations on VNP$_{nc}$-completeness and structure inside the VP$_{nc}$. Finally, in Section 9 we state some open problems.

2 Preliminaries

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 labelled 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.

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

A *noncommutative algebraic branching program* ABP ([Nis91], [RS05]) 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 $f \in \mathbb{F}\langle X\rangle$ defined as

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

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

2.1 Polynomials

We now define some p-families that are important for the paper.

Identity Polynomials:

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 will also consider some variants of this p-family in the paper.

Palindrome Polynomials:

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

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

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_1v_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 $D_i = (D_{i,n})$.

3 The Reducibilities

In the paper we consider mainly three different notions of reducibility for our completeness results and for exploring the structure of the classes $\mathrm{VNP}_{nc}, \mathrm{VP}_{nc}$ and VSKEW_{nc}.

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 [Val79]. 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 $f \leq_{proj} g$ 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 $\leq_{proj}$-reducibility.

The indexed-projection reducibility

Let $[n] = \{1, 2, \cdots, n\}$. The indexed-projection is specific to the noncommutative setting. We say $f \leq_{iproj} g$ 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, $\leq_{iproj}$ is more powerful than $\leq_{proj}$ and we will show separations in this section.

The abp-reducibility

The $\leq_{abp}$ reducibility is the most general notion that we will consider. The $\leq_{abp}$ 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 $f \leq_{abp} g$ 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 $\leq_{abp}$ 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 $\leq_{abp}$ reducible to the noncommutative determinant.

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

Proposition 3. *Let $f, g, h \in \mathbb{F}\langle X\rangle$ such that $f \leq_{abp} g$ and $g \leq_{abp} h$ then $f \leq_{abp} h$.*

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 $f_n \leq_{abp} g_{p(n)}$ and $g_{p(n)} \leq_{abp} h_{p'(n)}$ 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. *Let $f, g \in \mathbb{F}\langle X\rangle$ and suppose $f \leq_{abp} g$. 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 $f \leq_{abp} g$, 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 $\times$ 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 5. *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 $g_n = y_1 y_2 \ldots y_n$ 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 $\leq_{abp}$ 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 6. *An arithmetic circuit is* weakly skew *if for every multiplication gate at least one of its input gates has fanout 1. Suppose $f \leq_{abp} g$ 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].*

3.1 Comparing the reducibilities

From the definition of the reducibilities it immediately follows that the $\leq_{abp}$ reduction is more powerful than $\leq_{iproj}$ reduction which is more powerful than $\leq_{proj}$ reduction. In fact, it is not difficult to show that the $\leq_{abp}$ and the $\leq_{iproj}$ reductions are *strictly* more powerful than the $\leq_{iproj}$ and the $\leq_{proj}$ reductions respectively. We summarize these simple observations below.

Proposition 7. *There exist p-families $f = (f_n)$ and $g = (g_n)$ such that:*

1. $f \leq_{iproj} g$ *but* $f \not\leq_{proj} g$.

2. $f \leq_{abp} g$ but $f \not\leq_{iproj} g$.

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, $f \leq_{iproj} g$ 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 $f \not\leq_{proj} g$ as the usual $\leq_{proj}$ 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 $\leq_{iproj}$-reducible to g as indexed projections cannot increase the number of monomials in g. To see that $f \leq_{abp} g$ 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, $f \leq_{abp} g$. ■

Remark 8. *A natural generalization of the projection reducibilities ($\leq_{proj}$ and $\leq_{iproj}$) is to consider indexed linear projections, denoted $\leq_{linproj}$. 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 $\leq_{linproj}$ is strictly more powerful than $\leq_{iproj}$. For example, in Proposition 7 we saw that $x + y$ is not $\leq_{iproj}$-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.*

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 6.

3.2 Matrix substitutions and $\leq_{abp}$ reductions

We describe ideas from [AJS09] that are useful for the present paper in connection with showing $\leq_{abp}$ 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 9. [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 $f(M_{x_1}, M_{x_2}, \ldots, M_{x_n})$ 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 $\sum_w f(w)g(w)w$.

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

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 10. *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 9.

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

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

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

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

Corollary 11. *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 $\leq_{abp}$ 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 9 and Corollary 10).

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 $\leq_{abp}$ reductions. We show that the p-family D_k (defined in Section 2.1), consisting of Dyck polynomials over k different types of brackets, is VP_{nc}-complete under $\leq_{abp}$ 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 $f \leq_{abp} D_k$ 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 3.2.

Theorem 12 (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 $D_k = \{D_{k,d}\}_{d\geq 0}$ is defined over $2k$ distinct variables $X_k = \{(_i,)_i | 1 \leq i \leq k\}$, 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 2.1).

Theorem 13. *The p-family $D_k = \{D_{k,d}\}_{d\geq 0}$ is VP_{nc}-complete under $\leq_{abp}$ reductions for $k \geq 2$.*

Proof. Let $f = (f_n)_{n>0}$ be a p-family in VP_{nc} and $\{C_n\}_{n\geq 0}$ 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 3.2. 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 14. $f \leq_{proj} f''$.

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 $\leq_{abp}$ 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 $\leq_{abp}$ 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 $\hat{D} \leq_{abp} D_k$ 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 $\{_j, \}_j$, $0 \leq j \leq r$ (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 $u = \{_1\}_1\{_2\}_2 \ldots \{_{i-1}\}_{i-1}\{_0\}_0$, $0 \leq i \leq r+1$ 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 3.2) with exactly $q(n)$ layers.

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

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

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 $0 \leq j < r$.
ii. $(\hat{s}, 2(r-j)) \to \{_0\}_0(s, 2(r-j+1))$, for $0 \leq j \leq r$, 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 15. *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:

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

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 labelled 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 11, 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, $f'' \leq_{abp} \hat{D}$.

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

Claim 16. *The p-family $\hat{D}$ is $\leq_{abp}$-reducible to D_2.*

Proof of Claim. Let $2^{p(n)-1} < t(n) \leq 2^{p(n)}$ for some $p(n) \in O(\log n)$. Consider the p-family $\hat{D}' = (D_{2^{p(n)},q(n)})_{n>0}$. Clearly, $\hat{D} \leq_{proj} \hat{D}'$, 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 $\leq_{abp}$ 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 $0 \leq i \leq 2^{p(n)} - 1$ we will encode $[_i,]_i$ by strings $(_{b_0}(_{b_1} \cdots (_{b_{p(n)-1}}$ and $)_{b_0})_{b_1} \cdots)_{b_{p(n)-1}}$ 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 16, it follows that D_2 is VP$_{nc}$-complete. As for any $r \geq 2$ we have $D_2 \leq_{abp} D_r$. Thus the Dyck polynomials D_r for all $r \geq 2$ are VP$_{nc}$-complete. This proves the theorem. ■

Remark 17. *In the commutative setting, Valiant has shown [Val79] 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 $\leq_{abp}$ reducibility does not make sense for* VP. *The commutative ABP model is very powerful:* DET *itself has polynomial-size commutative ABPs [Tod92].*

Remark 18. *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, $D_1 \in$* VBP$_{nc}$. *In fact, notice that $D_1 \leq_{abp}$* PAL *$\leq_{abp} D_2$ and $D_2 \not\leq_{abp}$* PAL *$\not\leq_{abp} D_1$. As* PAL *is not in* VBP$_{nc}$ *[Nis91], it follows that* PAL *$\not\leq_{abp} D_1$. Since* PAL^2 = (PAL$_n$PAL$_n$) *is in* VP$_{nc}$ *and does not have polynomial-size skew circuits [LMS15], it follows that D_2 is not $\leq_{abp}$-reducible to* PAL.

5 Palindrome Polynomials are VSKEW$_{nc}$-complete

In this section we show that the p-family PAL, consisting of palindrome polynomials (defined in Section 2.1), is complete for the class VSKEW$_{nc}$ w.r.t. $\leq_{abp}$ reductions. The proof is broadly similar to that of Theorem 13.

Theorem 19. *The p-family* PAL *is* VSKEW$_{nc}$*-complete under $\leq_{abp}$ reductions.*

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

Let $f = (f_n)_{n>0}$ be a p-family in VSKEW_{nc} and $\{C_n\}_{n\geq 0}$ 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, $f \leq_{proj} f'$ 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 $y_{j,L}, y_{j,R}$, $0 \le j \le r(n)$ (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

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

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 $X''_n = \{y_{j,L}, y_{j,R} \mid 0 \le j \le r(n)+1\} \cup X'_n$.

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 $f'' \le_{abp} \hat{\text{PAL}}$ and $\hat{\text{PAL}} \le_{abp} \text{PAL}$. As $f \le_{proj} f' \le_{proj} f''$, it will follow that $f \le_{abp} \text{PAL}$, completing the proof.

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

The $\le_{abp}$ 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 $(y_{1,L}y_{2,L}\cdots y_{r-j,L}y_{0,L})$. These transitions can be described as $(\hat{s}, i) \to y_{(i+1,L)}(\hat{s}, i+1))$, for $0 \le i < r$. 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) $(\hat{s}, j) \to y_{(0,L)}(s, j+1)$, where $0 \le j \le r$ 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:

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

(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:

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

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

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

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 $r+1 \le \ell < 2r+2$. 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 20. *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 11. 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 11 implies that $f'' \le_{abp} \hat{\mathrm{PAL}}$.

$\hat{\mathrm{PAL}}$ **is $\leq_{abp}$-reducible to** PAL

Finally, we note that $\hat{\mathrm{PAL}}$ is $\leq_{abp}$-reducible to PAL. The polynomial $\hat{\mathrm{PAL}}_n$ 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 $x_0x_1^jx_0$ 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 $x_0x_1^kx_0$. 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 $x_0x_1^kx_0$ for distinct choices of integer k. We will need only integers $k \leq |X''_n|$ which is polynomially bounded. Furthermore, this encoding is invertible and can be implemented by a polynomial-size substitution automaton. It now follows from Corollary 11 that $\hat{\mathrm{PAL}} \leq_{abp}$ PAL. ■

6 A Ladner's Theorem analogue for VNP$_{nc}$

In this section we explore the class VNP$_{nc}$ assuming VP$_{nc} \neq$ VNP$_{nc}$. We exhibit an explicit p-family in VNP$_{nc} \setminus$ 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 degrees between P and NP-complete. For commutative Valiant's classes, the existence of VNP-intermediate p-families is investigated by Bürgisser [Bür99]. The results there require an additional assumption about counting classes in the boolean setting.

Definition 21 (VNP$_{nc}$-intermediate)**.** *We say that a noncommutative p-family $f = (f_n)_{n\geq 0}$ is* VNP$_{nc}$*-intermediate if $f \notin$* VP$_{nc}$ *and f is not* VNP$_{nc}$*-complete w.r.t. $\leq_{iproj}$ 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 $\leq_{iproj}$ reductions. We prove it unconditionally using a simple "transfer" theorem, which allows us to transfer a VNP$_{nc}$-complete p-family w.r.t $\leq_{iproj}$ reductions to a commutative VNP-complete p-family w.r.t $\leq_{proj}$ reductions.

Definition 22. *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 *$f^{(c)} = (f^{(c)}_n)$ as follows: Suppose $f_n \in \mathbb{F}\langle X_n\rangle$. Let $Y_n = \bigcup_{1\leq i\leq d(n)} X_{n,i}$ 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 $f^{(c)}_n$ is defined as $f^{(c)}_n = \sum \alpha_m m'$, where if $m = x_{j_1}x_{j_2}\ldots x_{j_q}$ then $m' = x_{j_1,1}x_{j_2,2}\ldots x_{j_q,q}$.*

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

Lemma 23. *For any p-families f and g (in $\mathbb{F}\langle X\rangle$), if $f \leq_{iproj} g$ then $f^{(c)} \leq_{proj} g^{(c)}$.*

Proof. Since $f \leq_{iproj} g$, for every n there is a polynomial $p(n)$ and an indexed projection $\phi_n : [d_{p(n)}] \times X_{p(n)} \to (Y_{ij})_{1\leq i,j\leq n}$ 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 $1 \leq i, j \leq n$. 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 23.

Theorem 24 (Transfer theorem)**.** *Let $f = (f_n) \in \mathrm{VNP}_{nc}$ be a p-family that is VNP_{nc}-complete under $\leq_{iproj}$-reductions. Then $f^{(c)}$ is* VNP*-complete under $\leq_{proj}$-reductions.*

Proof. Since f is VNP_{nc}-complete and $\mathrm{PER} \in \mathrm{VNP}_{nc}$ we have $\mathrm{PER} \leq_{iproj} f$. It follows from Lemma 23 that $\mathrm{PER}_d^{(c)} \leq_{proj} f^{(c)}$, which means that $f^{(c)}$ is VNP-complete under $\leq_{proj}$-reductions. ■

Corollary 25. *If* $\mathrm{VP} \neq \mathrm{VNP}$ *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 $\leq_{iproj}$ reductions then, by Theorem 24, DET is VNP-complete under $\leq_{proj}$-reductions. However, DET is in VP, which contradicts $\mathrm{VP} \neq \mathrm{VNP}$. ■

Remark 26. *We note here that* $\mathrm{VP} \neq \mathrm{VNP}$ *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 $\leq_{iproj}$ reductions.

Theorem 27. *The p-family ID is not* VNP_{nc}*-complete under $\leq_{iproj}$-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

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

All irreducible factors of $ID_n^{(c)}$ have degree 2. If g is a p-family such that $g \leq_{iproj} ID$, then by Lemma 23 we have $g^(c) \leq_{proj} ID^{(c)}$. 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, $g \in \mathrm{VNP}_{nc}$ and $g_n^{(c)}$ is irreducible of degree 3. Therefore, g is not $\leq_{iproj}$-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 $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$. Motivated by this observation, we consider a generalized version of ID which we call ID^* which turns out to be VNP_{nc}-complete under $\leq_{abp}$ reductions but not VNP_{nc}-complete under $\leq_{iproj}$ 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

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

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 28. *The p-family ID^* is* VNP_{nc}*-complete under $\leq_{abp}$ 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 $w_1 w_2 \dots w_{n^2}$, 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 29. *Assuming* $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$*, the p-family ID^* is* VNP_{nc}*-intermediate under $\leq_{iproj}$ projections.*

Proof. If ID^* is VNP_{nc}-complete under $\leq_{iproj}$ reductions, then $\mathrm{PER} \leq_{iproj} ID^*$ where $d \leq p(n)$ for a polynomial p. By Theorem 24 it follows that $\mathrm{PER}^{(c)} \leq_{proj} ID^{*(c)}$. 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 $\mathrm{PER}^{(c)} \leq_{proj} ID^{*(c)}$.

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 $\leq_{abp}$ reductions by Theorem 28. ■

6.1 A strict $\leq_{iproj}$ hierarchy in VNP_{nc}

We give an infinite hierarchy of p-families under $\leq_{iproj}$ 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 $f^{(i)} \leq_{iproj} f^{(i+1)}$ but $f^{(i+1)} \not\leq_{iproj} f^{(i)}$, for each positive integer $i \in \mathbb{N}$. Let $ID^* = (ID_n)$ where ID_n^* are degree n^3, and $D_2 = (D_{2,n})_{n\geq 0}$ 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 30. *For every i, $f^{(i)} \leq_{iproj} f^{(i+1)}$, 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 $\leq_{iproj}$ 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 31. *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 $D_{2,n}^{(c)}$ 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 $D_{2,n}^{(c)}$. 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 $D_{2,n}^{(c)}$, which is a contradiction. This completes the proof. ∎

Lemma 32. *1. If* $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$ *then* $ID^* \not\leq_{iproj} D_2$.

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

Proof. The first part follows from the VNP_{nc}-completeness of ID^* shown in Theorem 28. For the second part, if $D_2 \leq_{iproj} ID*$ then by Lemma 23 it follows that $D_2^{(c)} \leq_{proj} ID^{*(c)}$. By Lemma 31 $D_{2,n}^{(c)}$ is irreducible for each n. Moreover, the number of monomials of $D_{2,n}^{(c)}$ is $2^{\Omega(n)}$. On the other hand, each irreducible factor of $ID_d^{*(c)}$ has only d^2 monomials. Hence, $D_2^{(c)} \not\leq_{proj} ID^{*(c)}$. ∎

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

Theorem 33. *If* $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$ *then for each* i $f^{(i+1)} \not\leq_{iproj} f^{(i)}$.

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

- $f_{p(n)}^{(i)} = \underbrace{D_{2,p(n)} ID_{p(n)}^* \dots D_{2,p(n)} ID_{p(n)}^*}_{i-times}$
- $f_n^{(i+1)} = \underbrace{D_{2,n} ID_n^* \dots D_{2,n} ID_n^*}_{(i+1)-times}$

By Lemma 32, $ID_n^* \not\leq_{iproj} D_{2,n}$ 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 $f_n^{(i+1)}$ 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 32. Hence $f^{(i+1)} \not\leq_{iproj} f^{(i)}$. This proves the theorem. ∎

6.2 Discussion

Proving the existence of VNP_{nc}-intermediate p-families *under* $\leq_{abp}$ *reductions*, assuming $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$, remains open. At present we do not see an approach. However, in Section 3.1 we briefly discussed $\leq_{linproj}$, which we termed linear indexed reducibility. Unfortunately, our proof that $\mathrm{PER} \not\leq_{iproj} ID^*$ (see Theorem 29) does not generalize to $\leq_{linproj}$. 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 $\leq_{linproj}$ 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 $\leq linproj$ reductions.

Conjecture 34 (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 $\not\leq_{linproj} ID$. We can apply the argument of counting monomials in the irreducible factors of ID_d, which is also used in the proof of Theorem 29. 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 $\not\leq_{linproj} ID$.

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

Finally, exactly as in Section 6.1, we can combine ID and D_2 to define an infinite hierarchy of p-families $g^{(i)}$q within $\mathrm{VNP}_{nc} \setminus \mathrm{VP}_{nc}$ under $\leq_{linproj}$ reductions. We omit the proof details. The p-families $g^{(i)}$ are defined as:
$g_n^{(1)} = ID_n, \;\; g_n^{(2)} = D_{2,n} ID_n, \;\; g_n^{(i)} = g_n^{(i-1)} D_{2,n} ID_n.$

Theorem 35. *Assuming SOS_k conjecture we have for every i:*

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

7 More on VNP_{nc}-Completeness

By the transfer theorem (Theorem 24) we know that if f is a VNP_{nc}-complete p-family under $\leq_{iproj}$ reductions, then in the commutative setting $f^{(c)}$ is VNP-complete under $\leq_{proj}$-reductions.

For the reverse direction, suppose f is a commutative p-family which is VNP-complete under $\leq_{proj}$-reductions. There are several examples starting with the permanent, the p-family HC (corresponding to Hamiltonian circuits) and so on [Val79]. Is there an associated noncommutative p-family that is VNP_{nc}-complete under $\leq_{iproj}$ 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 $\mathrm{PER}_n \leq_{proj} f_{r(n)}$ 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 36. *If f is* VNP*-complete under $\leq_{proj}$ reductions then f^* is* VNP$_{nc}$*-complete under $\leq_{iproj}$ reductions.*

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

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

then in the noncommutative case the $\leq_{iproj}$ 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 $\leq_{iproj}$ 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. ∎

7.1 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

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

Clearly PER$^{\chi}$ = (PER$_n^{\chi}$) is a p-family that is in VNP$_{nc}$. For which functions χ is PER$^{\chi}$ 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 [BÖ0b, Bar90, BÖ0a, 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) [BÖ0b, 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 $\leq_{abp}$ 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$^\chi$, 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 37. 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$^\chi$ and PER$^{*,\chi}$ under assumptions about the function χ.

Theorem 38. *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 $\leq_{abp}$ PER$^\chi$.
- *If χ is computable by a logspace Turing machine then* PER $\leq_{abp}$ 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 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 39. *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* $\leq_{abp}$ *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.*

8 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 $\leq_{abp}$ reductions which roughly corresponds to the VNC_{nc} hierarchy.

Definition 40. *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.

8.1 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 41 (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 $W_{2,n}^{(k)}$ denote the set of all monomials in $D_{2,n}$ of depth at most k and degree $2n$. We define the polynomial $D_{2,n}^{(k)} = \sum_{u \in W_{2,n}^{(k)}} u$ and denote the corresponding p-family as $D_2^{(k)}$. In this definition we allow k to be a growing function $k(n)$ of n, where $D_2^{(k)} = (D_{2,n}^{(k)})_{n\geq 0}$.

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

Definition 42. *A p-family $f = (f_n)$ is in VAC_{nc}^i 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 $\bigcup_i \mathrm{VAC}_{nc}^i$.*

We note that $\mathrm{VNC}_{nc}^i \subseteq \mathrm{VAC}_{nc}^i \subseteq \mathrm{VNC}_{nc}^{i+1}$, 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 43. *For any $i \geq 0$, the Dyck polynomial of depth $\log^{i+1} n$ satisfy the following:*

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

Proof. It follows from inspection of the proof of the Theorem 13 that it scales down and yields part (1) of the theorem.

We now show part (2). Assume that $D_2^{(\log^{i+1} n)} \in \mathrm{VNC}_{nc}^i$. Then $D_2^{(\log^{i+1} n)}$ has an algebraic branching program of size $2^{O(\log^i n)} \cdot poly(n)$. The $\leq_{abp}$ reduction from PAL to D_2 can be easily modified to show that $\mathrm{PAL}_{\log^{i+1} n} \leq_{abp} D_{2,n}^{(\log^{i+1} n)}$. 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, $D_2^{(\log^{i+1} n)} \notin \mathrm{VNC}_{nc}^i$.

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

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

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 $f \not\leq_{abp} D_2^{(\log^{i+1} n)}$. 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 $f \leq_{abp} D_2^{(\log^{i+1} n)}$ 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. ■

9 Concluding remarks and open problems

Several open questions arise from our work. We list the important ones below:

- We have shown that Dyck polynomials are VP_{nc}-complete under $\leq_{abp}$ reductions. Finding natural VP_{nc}-complete p-families under $\leq_{iproj}$ 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 more general reducibility (projections are probably too restricted).

- Assuming $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$, analogous to Ladner's theorem, we have given an infinite hierarchy within VNP_{nc} under $\leq_{iproj}$ reductions. A similar result for the more powerful $\leq_{abp}$ reducibility will require substantially new techniques. It remains open whether we can show the existence of infinitely many p-families that are incomparable under $\leq_{proj}$ reductions (Lemma 32 shows the existence of two such p-families). It is also interesting to further compare the strengths of the three hypotheses considered in this paper: $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$, $\mathrm{VP} \neq \mathrm{VNP}$, and the SOS_k conjecture. As explained in Section 6.2, the first hypothesis is the weakest of the three. Does the SOS_k conjecture imply $\mathrm{VP} \neq \mathrm{VNP}$?

- 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 $\leq_{abp}$ reductions. Can we prove any hardness result for f in general?

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

- The complexity of the noncommutative immanent discussed in Section 7 remains open for different Young diagrams.

Acknowledgments. We are very grateful to the referees for their insightful and detailed reports. Their comments and suggestions have been greatly helpful to us in sharpening several statements and improving the overall presentation.

References

[AJS09] Vikraman Arvind, Pushkar S. Joglekar, and Srikanth Srinivasan, *Arithmetic circuits and the hadamard product of polynomials*, IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2009, December 15-17, 2009, IIT Kanpur, India, 2009, pp. 25–36.

[AS10] Vikraman Arvind and Srikanth Srinivasan, *On the hardness of the noncommutative determinant*, Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010, 2010, pp. 677–686.

[B0̈0a] Peter Bürgisser, *The computational complexity of immanants*, no. 3, 1023–1040, Preliminary version in FPSAC'98.

[B0̈0b] ———, *The computational complexity to evaluate representations of general linear groups*, no. 3, 1010–1022, 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** (1990), no. 2, 144–145.

[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** (2000), no. 3, 506–530.

[BR11] J. Berstel and C. Reutenauer, *Noncommutative rational series with applications*, Encyclopedia of Mathematics and its Applications, Cambridge University Press, 2011.

[Bür99] Peter Bürgisser, *On the structure of valiant's complexity classes*, Discrete Mathematics & Theoretical Computer Science **3** (1999), no. 3, 73–94.

[CS63] Noam Chomsky and Marcel Paul Schützenberger, *The Algebraic Theory of Context-Free Languages*, Computer Programming and Formal Systems (P. Braffort and D. Hirshberg, eds.), Studies in Logic, North-Holland Publishing, 1963, pp. 118–161.

[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.

[Har85] Werner Hartmann, *On the complexity of immanants*, Linear and Multilinear Algebra **18** (1985), no. 2, 127–140.

[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*, Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010, 2010, pp. 667–676.

[HWY10b] ______, *Relationless completeness and separations*, Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, June 9-12, 2010, 2010, pp. 280–290.

[Lad75] Richard E. Ladner, *On the structure of polynomial time reducibility*, J. ACM **22** (1975), no. 1, 155–171.

[LMS15] Nutan Limaye, Guillaume Malod, and Srikanth Srinivasan, *Lower bounds for non-commutative skew circuits*, Electronic Colloquium on Computational Complexity (ECCC) **22** (2015), 22.

[MM13] Stephan Mertens and Cristopher Moore, *The complexity of the fermionant and immanants of constant width [note]*, Theory of Computing **9** (2013), 273–282.

[Nis91] Noam Nisan, *Lower bounds for non-commutative computation (extended abstract)*, STOC, 1991, pp. 410–418.

[RS05] Ran Raz and Amir Shpilka, *Deterministic polynomial identity testing in non-commutative models*, Computational Complexity **14** (2005), no. 1, 1–19.

[Str69] Volker Strassen, *Gaussian elimination is not optimal*, Numerische Mathematik **13** (1969), no. 4, 354–356.

[Tod92] Seinosuke Toda, *Classes of arithmetic circuits capturing the compl exity of computing the determinant*, IEICE Transactions on Informations and Systems,E75-D , 116–124, 1992, 1992, pp. 116–124.

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

ECCC ISSN 1433-8092

http://eccc.hpi-web.de