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

Noncommutative Valiant's Classes: Structure and Complete Problems

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

August 3, 2015

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. 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.
- Next, we consider the class VNP_{nc}. It is known [HWY10a] that, assuming the sum-of-squares conjecture, the noncommutative polynomial $\sum_{w\in\{x_0,x_1\}^n} ww$ requires exponential size circuits. We unconditionally show that $\sum_{w\in\{x_0,x_1\}^n} ww$ is not VNP_{nc}-complete under the projection reducibility. As a consequence, assuming the sum-of-squares conjecture, we exhibit a strictly infinite hierarchy of p-families under projections inside VNP_{nc} (analogous to Ladner's theorem [Lad75]). In the final section we discuss some new VNP_{nc}-complete problems under $\leq_{abp}$-reductions.
- Inside VP_{nc} too we show there is a strict hierarchy of p-families (based on the nesting depth of Dyck polynomials) under the $\leq_{abp}$ reducibility.

1 Introduction

Proving superpolynomial size lower bounds for arithmetic circuits that compute the permanent polynomial PER_n is a central open problem in computational complexity theory. This problem has a rich history in the field, starting with the work of Strassen on matrix multiplication [Str69].

In the late 1970's, 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 over a field $\mathbb{F}$ that have polynomial size circuits. The class VNP, which corresponds to NP (in fact #P to be precise) has a more technical definition which we will give later.

*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

Valiant showed that PER_n is VNP-complete w.r.t projection reductions. Thus, VP $\neq$ VNP iff PER_n requires superpolynomial in n size arithmetic circuits.

In 1990 paper [Nis91], Nisan explored the same question for *noncommutative* polynomials. The noncommutative polynomial ring $\mathbb{F}\langle x_1, \ldots, x_n\rangle$ consists of $\mathbb{F}$-linear combinations of words (we call them monomials) over the alphabet $X = \{x_1, \ldots, x_n\}$.

We can analogously define noncommutative arithmetic circuits for polynomials in $\mathbb{F}\langle X\rangle$ where the inputs to multiplication gates are ordered from left to right. A natural definition of the noncommutative PER_n is

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

over $X = \{x_{ij}\}_{1 \leq i,j \leq n}$.

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, in the noncommutative setting, 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] they have shown that PER_n is VNP_{nc}-complete w.r.t projections (the Valiant [Val79] notion which allows variables or scalars to be substituted for variables).

The purpose of our paper is to study 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 summarize our main results. Detailed definitions are given in the next section.

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.

- *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.*
- *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.*
- *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 $p(n)$*

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

where $q(n)$ is number of variables in f_n.

We note that the class VBP_{nc} is defined through polynomial-size algebraic branching programs (ABPs) which intuitively correspond to polynomial sized 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]. Moreover, arithmetic circuits can be seen as acyclic context-free grammars (CFGs) (if the coefficients come from the Boolean ring instead of a field).

It turns out, as we will see in this paper, the 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 proof is really an arithmetized version of the Chomsky-Schützenberger theorem [CS63] showing that the Dyck languages are the hardest CFLs.

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}, again by adapting the proof of the Chomsky-Schützenberger theorem.

3. 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}.

4. We examine the structure within VNP_{nc} assuming the sum-of-squares conjecture, under which Hrubes et al [HWY10a] have shown that the p-family $\{ID_d = \sum_{w\in\{x_0,x_1\}^d} w.w\}_d \notin \mathrm{VP}_{nc}$. We prove the following results about $\mathrm{VNP}_{nc} \setminus \mathrm{VP}_{nc}$.

   (a) 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.

   (b) Assuming the sum-of-squares conjecture we show that the polynomial ID_d is neither in VP_{nc} nor VNP_{nc}-complete w.r.t $\leq_{proj}$ reductions. This is analogous to Ladner's well-known theorem [Lad75].

   (c) It also turns out that under the sum-of-squares conjecture we have an infinite hierarchy w.r.t $\leq_{proj}$ reductions between VP_{nc} and VNP_{nc} and also incomparable p-families.

Table 1.1 summarizes the results in this paper.

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 is the set of indeterminates. 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

P-family	Complexity Result	Remarks
$D_k, k \geq 2$	- VP$_{nc}$-Complete (Theorem 12) - VSKEW$_{nc}$-hard (Theorem 27)	w.r.t $\leq_{abp}$-reductions
PAL$_d$	VSKEW$_{nc}$-Complete (Theorem 14)	w.r.t. $\leq_{abp}$-reductions
ID_d	-not VNP$_{nc}$-Complete (Theorem 21) -not VP$_{nc}$-hard (Theorem 23) -not in VP$_{nc}$ [HWY10a]	$\leq_{proj}, \leq_{iproj}$-reductions $\leq_{proj}, \leq_{iproj}$-reductions assuming SOS_k conjecture
PER$^{*,\mathcal{X}}$	VNP$_{nc}$-Complete (Theorem 33)	w.r.t. $\leq_{abp}$-reductions
ID_n^*	VNP$_{nc}$-Complete (Theorem 31)	w.r.t. $\leq_{abp}$-reductions

Table 1: Summary of Results

computed at its children. The circuit C computes the polynomial at the designated output node.

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 PAL $=$ (PAL$_n$) corresponds to the context-free language of palindromes.

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

Dyck Polynomials:

Let $X_i = \{(_1,)_1, ..., (_i,)_i\}$ for a fixed $i \in \mathbb{N}$. 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 (for all the i bracket types). The $D_{i,n}$ are Dyck polynomials of degree $2n$ over i different parenthesis. The corresponding p-family is denoted $D_i = (D_{i,n})$.

3 The Reducibilities

In the paper we consider 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 essentially Valiant's notion of reduction for 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] by using Valiant's original proof, the noncommutative PER_n p-family is VNP_{nc}-complete for $\leq_{proj}$-reducibility.

The indexed-projection reducibility

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 a monomial 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. It essentially amounts to matrix substitutions for variables, where the matrices have scalar or variable (we allow even constant-degree monomial) entries. In terms of complexity classes we have: $\mathrm{VBP}_{nc} \subsetneq \mathrm{VSKEW}_{nc} \subsetneq \mathrm{VP}_{nc} \subseteq \mathrm{VNP}_{nc}$. And $\leq_{abp}$-reductions correspond to the computational power of the class VBP_{nc}.

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

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

Proposition 4. *Let $f, g \in \mathbb{F}\langle X\rangle$ and $f \leq_{abp} g$. Then if g has polynomial size ABP or a noncommutative arithmetic circuit or a noncommutative skew circuit then f has polynomial size ABP, a noncommutative arithmetic circuit, a noncommutative skew circuit respectively.*

3.1 Hadamard product of polynomials

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 5. [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 Hadamard product polynomial $\sum_w f(w)g(w)w$.

Remark 6. *A specific case of interest is when the ABP P is a deterministic finite automaton with start state s and sink t. In that case the polynomial g is the sum of all monomials that are accepted by the automaton (since it is acyclic, it accepts only finitely many). Let W denote the set of monomials accepted by the automaton P. Then the $(s,t)^{th}$ entry of the matrix $f(M_{x_1}, M_{x_2}, \ldots, M_{x_n})$ is the polynomial*

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

Remark 7. *It is useful to combine the construction described in the previous remark with* substitution maps. *As above, let the ABP P be a deterministic finite substitution automaton with q states accepting monomials of degree at most d over variables X with start state s and accept state t. The substitutions are defined as follows:*

For $1 \leq i,j \leq q$, $\psi_{ij} : X \to Y^*$ is a substitution map mapping variables in X to monomials over Y, where q is the number of nodes in the ABP P. For each $x \in X$ define the matrix M'_x as follows:

$$M'_x(i,j) = \psi_{ij}(x), 1 \leq i,j \leq q.$$

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

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

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

so that $\psi(w) \in Y^*$.

Let W denote the set of monomials accepted by the automaton P. Then the $(s,t)^{th}$ entry of the matrix $f(M'_{x_1}, M'_{x_2}, \dots, M'_{x_n})$ is the polynomial

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

From the above considerations it is clear that if $f \in \mathbb{F}\langle X\rangle$ has a polynomial-size circuit and P is a polynomial-size automaton then $\sum_{w\in W} f(w)\psi(w)$ has a polynomial-size circuit.

Comparing the reducibilities

Proposition 8. *For noncommutative p-families $f = (f_n)$ and $g = (g_n)$ we have,*

1. $f \leq_{proj} g \Rightarrow f \leq_{iproj} g$
2. $f \leq_{iproj} g \Rightarrow f \leq_{abp} g$

Theorem 9. *There are noncommutative p-families $f = (f_n)$ and $g = (g_n)$ such that $g \leq_{abp} f$ but $f \not\leq_{iproj} g$ and $g \not\leq_{iproj} f$.*

Proof. We define the p-families as follows: $g_n, f_n \in \mathbb{F}\langle x_1, x_2, \dots, x_n, y_1, \dots, y_n\rangle$ where $f_n = \prod_{i\in[n]}(x_i + y_i)$ and $g_n = x_1x_2\dots x_n + y_1y_2\dots y_n$. A key fact which is easy to check is that g_n is irreducible for all n, and f_n is a product of linear forms obviously. More crucially, g_n has only two monomials for all n, whereas f_n has 2^n nonzero monomials.

Now, if $f \leq_{iproj} g$ then for some polynomial $p(n)$ and substitution map ϕ we will have $g(\phi(X_{p(n)})) = f(X_n)$ where $X_n = \{x_1, \dots, x_n, y_1, \dots, y_n\}$ and $X_{p(n)} = \{x_1, \dots, x_{p(n)}, y_1, \dots, y_{p(n)}\}$. However, the substitution map cannot increase the number of monomials in $g(\phi(X_{p(n)}))$ whereas $f(X_n)$ has 2^n monomials. Hence $f \not\leq_{iproj} g$.

Also, $g \not\leq_{iproj} f$ because for all n, g_n is irreducible and f_n is a product of linear forms over $\mathbb{F}$.

Now, we claim $g \leq_{abp} f$, where the abp-reduction is defined by the following matrix substitutions which are given by the following DFA with start state s and final state t:

- In start state s, reading x_1 go to state 1 and reading y_1 go to state $1'$.
- In state i, reading x_{i+1} go to state $i+1$, $i < n-1$.
- In state i', reading y_{i+1} go to state $(i+1)'$, $i' < n-1$.
- In state $n-1$, reading x_n go to state t.
- In state $(n-1)'$, reading y_n go to state t.

For each variable in $\{x_1, \dots, x_n, y_1, \dots, y_n\}$ we substitute matrices of dimension $2n \times 2n$, corresponding to the above DFA, in the polynomial f to obtain polynomial g. ∎

Theorem 10. *There are p-families f and g s.t $f \leq_{iproj} g$ but $f \not\leq_{proj} g$.*

Proof. Let $f = \prod_{i\in[n]}(x_i + y_i)$ and $g = \prod_{i\in[n]}(z_0 + z_1)$. 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, the usual $\leq_{proj}$ reduction cannot increase the number of variables in g from two. Hence $f \not\leq_{proj} g$. ■

4 Dyck Polynomials are VP_{nc}-complete

4.1 VP_{nc}-Completeness

In this section we exhibit a *natural* p-family which is $\leq_{abp}$-complete for the complexity class VP_{nc}. We show that any homogeneous degree d polynomial $f \in \mathbb{F}\langle x_1, x_2, \ldots, x_n\rangle$ computed by a non-commutative arithmetic circuit of size $poly(n, d)$ is *abp*-reducible to the polynomials D_k for $k \geq 2$, where D_k refers to the Dyck polynomial over k different types of brackets. Our main Theorem in this section can be seen as an algebraic 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. More precisely,

Theorem 11 (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 $P \cup \overline{P}$.

We now show that the p-family $\{D_{k,d}\}_{d\geq 0}$ is VP_{nc}-complete for $\leq_{abp}$ reductions, where the p-family $\{D_{k,d}\}_{d\geq 0}$, denoted D_k, is over set of $2k$ distinct variables $\{(_i,)_i | 1 \leq i \leq k\}$ where $(_i$ and $)_i$ are matching parenthesis pairs. The polynomial $D_{k,d}$ consists of the sum of all monomials m which are well formed parenthesis strings of degree d over variables in X_k.

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

where $W_{k,d}$ is set of well formed parenthesis strings of degree d over X_k. The theorem we prove in this section is the following.

Theorem 12. *The Dyck polynomial $D_2 = \{D_{2,d}\}_{d\geq 0}$ is VP_{nc}-complete under $\leq_{abp}$-reductions and hence $D_k = \{D_{k,d}\}_{d\geq 0}$ for $k \geq 2$ is VP_{nc}-complete under $\leq_{abp}$-reductions.*

Proof. Let $\{C_n\}_{n\geq 0}$ be a polynomial sized polynomial degree circuit family computing polynomials (by abuse of notation, also denoted by) C_n in $\mathbb{F}\langle x_1, \ldots, x_n\rangle$. Let $s(n)$ and $d(n)$ be polynomials bounding the size and degree of C_n, respectively. 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 large polynomial bound $t(n)$. These matrices have the property that polynomial $D_{t(n),q(n)}$, in which we substitute M_i for $(_i$ and M'_i for $)_i$, evaluates 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 C_n.

The idea underlying this construction is from the proof of the Chomsky-Schützenberger theorem (ours is an arithmetic version of it) : the matrices $M_1, M'_1, \ldots, M_t, M'_t$ actually correspond to the transitions of a deterministic finite state substitution automaton which will 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) 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 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 over $\mathbb{F}\langle X' \rangle$ where

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

   We make a further substitution: we replace every variable $y \in X$ by the degree-2 monomial $[_y]_y$ and every variable z_a for constants a appearing in C_n by $[_{z_a}]_{z_a}$ to obtain the arithmetic circuit C''_n.

   With these substitutions it is clear, by abuse of notation, that (C''_n) is a p-family. Furthermore, by construction C''_n is a polynomial whose monomials are certain properly balanced parenthesis strings over the above parentheses set. It is not homogeneous, but clearly its degree bounded by a polynomial in $(s(n) + d(n))$. Furthermore, $C_n \leq_{abp} C''_n$ because we can recover C_n by substituting 1 for the parenthesis and y for the term $[_y]_y$ and the scalar a for $[_{z_a}]_{z_a}$.

2. The next step is the crucial part of the proof. We describe the reduction from C''_n to $D_{t(n)}$ for suitably chosen $t(n)$. Indeed, $t(n)$ is already the number of parentheses type used by C''_n, along with some additional parenthesis. Let the degree of polynomial C''_n be $2r$. Thus, monomials of 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) and consider the polynomial $D_{t(n),q(n)}$ where $q(n) = 2r + 2$ and $t(n)$ is $(r + 1)$ plus the number of parenthesis types occurring in C''_n.

The reduction will map all degree $2j$ monomials in C''_n to monomials in $D_{t,q}$ of the form $m' = \{_1\}_1\{_2\}_2 \dots \{_{r-j}\}_{r-j}\{_0\}_0 m$ where m is a degree $2j$ monomial over the other parentheses types. Now m' is of degree $2r + 2$ for all choices of j and it is clear that monomials which were distinct before the reduction remains distinct after the reduction.

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 $u = \{_1\}_1\{_2\}_2 \dots \{_{i-1}\}_{i-1}\{_0\}_0$, $0 \le i \le r+1$ and v is a well-balanced string over remaining parentheses type. This automaton is essentially the one defined in the proof of the Chomsky-Schützenberger theorem. We outline its description. The automaton runs only on monomials of $D_{t,q}$ and hence can be seen as a layered DAG with exactly $q(n)$ layers.

(a) The start state of the automaton is $(\hat{s}, 0)$. The automaton first looks for prefix $\{_1\}_1\{_2\}_2 \dots \{_{r-j}\}_{r-j}\{_0\}_0$. 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 $\{_l, \}_l$, $l \in [r] \cup \{0\}$ occur later they are substituted by 0 (to kill that monomial).

(b) The automaton will substitute $[_x]_x$ by x (if $[_x$ is not immediately followed by $]_x$ then it substitutes 0 for $[_x$). Similarly, the automaton substitutes $[_a]_a$ by a (if $[_a$ is not followed by $]_a$ then it substitutes 0 for it).

(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 $D < 2r + 2$. We have the following transitions:

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

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

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

   iv. Include the transition $(g, D) \to (_h(h_l, D + 1)$, 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 $(h, D) \to [_z]_z)_g(g_r, D{+}3)$, if $D + 3 < 2r + 2$, where g_r is the right child of the internal product gate g, and h stands for any internal gate in C''_n.
   If $D + 3 = 2r + 2$ then the automaton instead includes the transition $(h, D) \to [_z]_z)_g(t, 2r{+}2)$, where $(t, 2r{+}2)$ is the unique accepting state of the automaton.

   Note that the interpretation of the transition

$$(h, D) \to [_z]_z)_g(g_r, D + 3)$$

   is as follows: The automaton reads the degree-3 monomial $[_z]_z)_g$ and goes from state (h, D) to $(g_r, D + 3)$.

We now describe the matrices that we substitute for each parenthesis. Let M_p be the matrix we substitute for parenthesis p its whose rows and columns are labelled by nodes of the ABP.

We define the matrix M_p for parenthesis p as follows:

$$m_{u,v} = M_p[u,v] = \begin{cases} 1 & \text{if } p \in U \text{ and } \exists e = (u,v) \in \mathrm{E(A)} \text{ and label of } e \text{ is } p \\ z & \text{if } p =]_z \text{ and } \exists e = (u,v) \in \mathrm{E(A)} \text{ and label of } e \text{ is } p \end{cases}$$

where z denotes a variable in the circuit C''_n and E(A) is the edge set of the automaton A and

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

where s' denotes the number of product gates in the circuit C_n.

It is clear that after substituting these matrices for the variables in the polynomial D_k, where k denotes the number of parenthesis types in C''_n, the top right corner entry of the resulting matrix is polynomial computed by the given circuit C. It is easy to see that $D_2 \leq_{abp} D_k$ for all $k > 2$. Furthermore, we can show for any $k > 2$ that $D_k \leq_{abp} D_2$, by suitably encoding different types of brackets into two types. Thus, it follows that the p-family D_k, for any $k \geq 2$, is VP_{nc}-complete under $\leq_{abp}$-reductions. ■

Remark 13. *We note that $D_1 \leq_{abp}$ PAL $\leq_{abp} D_2$ and $D_2 \not\leq_{abp}$ PAL $\not\leq_{abp} D_1$. To see this the first one, observe that we have a DFA (of growing size) for D_1. Hence D_1 is in VBP_{nc} which trivially implies that D_1 is $\leq_{abp}$-reducible to* PAL. *As* PAL *is not in* VBP_{nc} *[Nis91], it follows that* PAL $\not\leq_{abp} D_1$. *We show in theorem 27 that D_2 is not $\leq_{abp}$-reducible to* PAL.

5 Palindrome Polynomials are VSKEW_{nc}-complete

Theorem 14. *The p-family* PAL *is* VSKEW_{nc}*-complete for $\leq_{abp}$ reductions.*

Proof. The proof is along the same lines as that of Theorem 12. We will show for any p-family in VSKEW_{nc} is $\leq_{abp}$-reducible to PAL.

Let $\{C_n\}_{n\geq 0}$ be a polynomial sized skew circuit family of polynomial degree $d(n)$ computing polynomials (by abuse of notation, also denoted by) C_n in $\mathbb{F}\langle x_1, \ldots, x_n\rangle$. Let $s(n)$ and $d(n)$ be polynomials bounding the size and degree of C_n, respectively. We will construct a collection of $2t(n)$ matrices $M_1, M'_1, \ldots, M_{t(n)}, M'_{t(n)}$ whose entries are either field elements or monomials in variables $\{x_{1,L}, x_{1,R}, \ldots, x_{n,L}, x_{n,R}\}$ for a suitably large polynomial bound $t(n)$. These matrices have the property that polynomial $\mathrm{PAL}_{t(n)}$, in which we substitute M_i for $x_{i,L}$ and M'_i for $x_{i,R}$, evaluates to a matrix $M = \mathrm{PAL}_t(M_1, M'_1, \ldots, M_{t(n)}, M'_{t(n)})$ whose top right corner entry is precisely the polynomial C_n.

As in the proof of Theorem 12, the basic idea is from the Chomsky-Schützenberger theorem: the matrices $M_1, M'_1, \ldots, M_t, M'_t$ will correspond to the transitions of a deterministic finite state (substitution) automaton which will transform monomials of $\mathrm{PAL}_{t(n)}$ into monomials of C_n so that M's top right entry (corresponding to the accept state) contains the polynomial C_n. We now give a structured description of the reduction.

W.l.o.g we can assume the skew circuit C_n is homogeneous. At the input level, we replace variables x by $x_L x_R$.

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 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 transformations:

   (a) For each left-skew product gate $f = xh$ in the circuit C_n (similarly for the right-skew gate $f = hx$), where x is an input variable and h a gate in the circuit, let $e = (h, f)$ denote the directed edge in the circuit C_n (seen as a directed acyclic graph). We convert it to the gates

   $$\begin{aligned} f' &= hx_{(e,h,R)} \\ f'' &= x_{(e,h,L)}f', \end{aligned}$$

   where $x_{(e,h,L)}, x_{(e,h,R)}$ are fresh variables.

   (b) For each product gate $f = ah$ in the circuit C_n for $a \in \mathbb{F}$ and $e = (h, f)$ is the edge in the circuit we convert it to gates

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

   where $a_{(e,h,L)}, a_{(e,h,R)}$ are fresh variables.

   Let C'_n denote the resulting circuit. It computes a polynomial over $\mathbb{F}\langle X'\rangle$ where the variable set X' is:

   $$\begin{aligned} X' &= \{x_{(e,h,R)}, x_{(e,h,R)} | x \in X, e \in E\} \\ &\cup \ \{a_{(e,h,L)}, a_{(e,h,R)} | a \text{ is a constant appearing in the edge } e \in E\ \}. \end{aligned}$$

   Here E is set of all edges e in the given circuit C_n.

   Clearly, (C'_n) is a p-family, and C'_n is a polynomial whose nonzero monomials m are palindrome monomials in the following sense: in a monomial m of degree $2d$, for all $i \in [d]$ and for any edge e and gate g at position i we have variable $x_{(e,g,L)}$ and at position $2d - i + 1$ we have variable $x_{(e,g,R)}$.

   We also have the reduction $(C_n) \leq_{abp} (C'_n)$ because we can recover C_n from C'_n by substituting x for either $x_{e,h,L}$ or $x_{e,h,R}$ (and 1 for the other variable) and the scalar a for either $a_{e,h,L}$ or $a_{e,h,R}$ (and 1 for the other variable). Notice that the number of variables in C'_n and the degree of C'_n are polynomially bounded by a suitable function of n (but we are not specifying it for ease of notation).

2. Let the degree of polynomial C'_n be $2r$. Thus monomials of C'_n are of even degree bounded by $2r$. Like in Theorem 12, we will introduce $r+1$ *new* variable pairs $y_{j,L}, y_{j,R}$, $0 \leq j \leq r$ (to be used as prefix and suffix padding in order to get homogeneity). The

reduction will map a degree $2j$ monomial m in C'_n to monomial m' in PAL_{r+1} of the following form:

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

Now, m' is of degree $2r+2$ for all choices of j and it is clear that monomials which were distinct before the reduction remains distinct after the reduction. Let C''_n denote this resulting new circuit.

3. Like in Theorem 12, we construct automaton A from this modified circuit C''_n. We construct automaton which (apart from accepting many non-palindrome monomials) accepts only palindrome monomials ww^R such that the first half w is "compatible" with the circuit structure of C''_n (and monomials whose first half is non-compatible are not accepted by the automaton A). Now the matrices of the automaton have only to effect substitutions in a careful manner to convert these m' into a monomial of C''_n of degree $2j$. The automaton is a layered DAG with exactly $2r+2$ layers.

   (a) The start state of the automaton is $(\hat{s}, 0)$. The automaton first looks for a prefix $(y_{1,L}y_{2,L}\dots y_{r-j,L}y_{0,L})$. As it reads these variables, one by one, it steps through states $(\hat{s}, i)$, substitutes 1 for each of them, and reaches state $(s, (r-j+1))$ 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 they are substituted by 0 (to kill that monomial).

   (b) Now we describe the transitions of the automaton continuing from state $(s, (r-j+1))$. Here the automaton has to use the structure of the circuit C''_n to define further transitions. At this point the automaton is looking for a degree $2j$ monomial. Let $D < 2r+2$. We have the following transitions:

      i. $(\hat{s}, j) \to y_{(j+1,L)}(\hat{s}, j+1))$, where $0 \le j < r$ (as already described above).

      ii. $(\hat{s}, j) \to y_{(0,L)}(s, j+1)$, where $0 \le j \le r$ and s is the output gate in the circuit C''_n.

      iii. In state $(s, j+1)$ if the automaton reads variable $x_{e,g,L}$ (or variable $a_{e,g,L}$) then 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 transition made is:

$$(s, j+1) \to x_{(e,g,L)}(g, j+2).$$

      We have a similar transition when the automaton reads variable $a_{e,g,L}$.

      iv. In general, when the automaton is in state (g, D) for a left-skew multiplication gate g in the circuit and it reads variable $x_{e,h,L}$ (or $a_{e,h,L}$) then it moves to state $(h, D+1)$ if the gate h is left-skew occurring in the circuit, and the directed path from h to g has only $+$ gates or right-skew multiplication gates in it. Formally, the transition made is:

$$(g, D) \to x_{(e,h,L)}(h, D+1).$$

      We have a similar transition for variable $a_{e,h,L}$.

v. Proceeding thus, when the automaton reaches a state $(g, r+1)$ for some left-skew multiplication gate it makes only transitions of the form:

$$(g, D) \to x_{(e,h,R)}(t, D+1),$$

for all variables $x_{e,h,R}$ and for all $D < 2r+2$. The state $(t, 2r+2)$ is the unique accepting state of the automaton.

Transitions (i-iv) reads the first half of any input monomial which are compatible with the structure of the circuit C''_n. By construction of the transitions in (i-iv) the following claim holds.

Claim 15. *The DFA defined above accepts a palindrome string $uv \in (X')^{2r+2}$ iff the palindrome uv is a nonzero monomial in the polynomial computed by C''_n.*

4. We can convert this automaton into a homogeneous ABP A computing the homogeneous polynomial of degree $2r+2$. We now describe matrices we substitute for each variable. Let M_z be the matrix we substitute for a variable z where rows and columns of M_z are labelled by nodes of the ABP.

   We set entries of the matrix M_z for a variable z as follows:

   - If the variable $z = a_{(e,h,L)}$ where a is a scalar appearing on the edge e in the circuit C_n, then we set $m_{u,v} = M_z[u,v] = a$ iff the automaton reaches the state v from the state u when it reads z.
   - Else, if the variable $z = a_{(e,h,R)}$ where a is a scalar appearing on the edge e in the circuit C_n, then we set $m_{u,v} = M_z[u,v] = 1$ iff the automaton reaches the state v from the state u when it reads z.
   - Else, if $z = x_{(e,g,L)}$, where $x \in X$, e is an edge in the circuit C_n, g is some gate in C_n, then
     - If the actual variable for z occurs as left multiplication on the edge e, then we set $m_{u,v} = x$ iff the automaton reaches the state v from the state u when it reads z.
     - Else, if $m_{u,v} = 1$ (i.e., the actual variable for z occurs as right multiplication)
   - Else, if $z = x_{(e,g,R)}$, where $x \in X$, e is an edge in the circuit C_n, g is some gate in C_n, then
     - If the actual variable for z occurs as right multiplication on the edge e, then we set $m_{u,v} = x$ iff the automaton reaches the state v from the state u when it reads z.
     - Else, if $m_{u,v} = 1$ (i.e., the actual variable for z occurs as left multiplication)
   - Else, if the variable $z = y_{(j,L)}$ or $z = y_{(j,R)}$, $0 \le j \le r$ then we set $m_{u,v} = M_z[u,v] = 1$ iff the automaton reaches the state v from the state u when it reads z.
   - Else, we set $m_{u,v} = 0$.

It is clear that on substituting these matrices for the variables in PAL_{r+1} , we get the polynomial computed by the given circuit C_n in the top right corner entry of the resulting matrix. This completes the proof. ■

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

In this section we explore the structure of VNP_{nc} assuming the sum-of-squares conjecture. The sum-of-squares conjecture implies that the p-family ID (which is in VNP_{nc}) is not in VP_{nc} [HWY10a]. In particular, the conjecture implies that $\mathrm{VP}_{nc} \neq \mathrm{VNP}_{nc}$. A natural question that arises is whether this conjecture implies that there are p-families in $\mathrm{VNP}_{nc} \setminus \mathrm{VP}_{nc}$ that are not VNP_{nc}-complete.

This is similar in spirit to the well-known Ladner's Theorem that shows, assuming $\mathrm{P} \neq \mathrm{NP}$, 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.

Conjecture 16 (SOS_k Conjecture)**.** *Consider the question of expressing the biquadratic polynomial*

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

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

The SOS_k conjecture states that over the field of complex numbers $\mathbb{C}$, for all k we have the lower bound $s = \Omega(k^{1+\epsilon})$.

In [HWY10a], it is shown that the SOS_k-conjecture implies that the p-family $ID = \{ID_d\}_{d\geq 0}$ where $ID_d(x_0, x_1) = \sum_{w\in\{x_0,x_1\}^d} ww$ is not in VP_{nc}. In fact, they prove exponential circuit size lower bounds for ID_d assuming the conjecture. We need the following definition.

Definition 17 (VNP_{nc}-intermediate)**.** *We say that a noncommutative p-family $f = (f_n)_{n\geq 0}$ is VNP_{nc}-intermediate if $f \notin \mathrm{VP}_{nc}$ and f is not VNP_{nc}-complete w.r.t. $\leq_{iproj}$ reductions.*

In this section, we show the SOS_k conjecture actually yields much more inside VNP_{nc}. We prove the following results.

1. That ID is a VNP_{nc}-intermediate polynomial assuming SOS_k conjecture.
2. There are infinitely many p-families $f^{(i)}$, $i = 1, 2, \ldots$ in VNP_{nc} such that for all i, $f^{(i)} \leq_{iproj} f^{(i+1)}$ and $f^{(i+1)} \not\leq_{iproj} f^{(i)}$.

We do not have similar results for the stronger $\leq_{abp}$ reducibility.

The proof of the first result is by 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 18. *Let $f = (f_n)$ be a p-family in VNP_{nc}, where each f_n is a homogeneous polynomial of degree $d(n)$. We define the* commutative version *$f^{(c)} = (f_n^{(c)})$ as follows: Suppose $f_n \in \mathbb{F}\langle X_n\rangle$. Let $Y_n = \bigcup_{1\leq i\leq d(n)} X_{n,i}$ 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 $m \in X_n^{d(n)}$ is a monomial, the polynomial $f_n^{(c)}$ is defined as $f_n^{(c)} = \sum \alpha_m m'$, where if $m = x_{j_1} x_{j_2} \ldots x_{j_d}$ then $m' = x_{j_1,1} x_{j_2,2} \ldots x_{j_d,d}$.*

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

Lemma 19. *For any p-families f and g, 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 theorem is a corollary of Lemma 19.

Theorem 20 (Transfer theorem)**.** *Let $f = (f_n) \in \mathrm{VNP}_{nc}$ be a homogeneous p-family that is VNP_{nc}-complete for $\leq_{iproj}$-reductions. Then $f^{(c)} \in \mathrm{VNP}$ is VNP-complete for $\leq_{proj}$-reductions.*

Proof. Since $\mathrm{PER} \leq_{iproj} f$, by Lemma 19 $\mathrm{PER}_d^{(c)} \leq_{proj} f^{(c)}$. This completes the proof of the theorem. ∎

Theorem 21. *The polynomial ID is not VNP_{nc}-complete under $\leq_{iproj}$-reductions.*

Proof. Suppose, to the contrary that ID is VNP_{nc}-complete w.r.t $\leq_{iproj}$-reductions. Then $\mathrm{PER} \leq_{iproj} ID$. Define the noncommutative p-family $ID' = (ID'_n)_{n\geq 0}$, where $ID'_n \in \mathbb{F}\langle X_n\rangle$ where $X_n = \{x_{0,1}, x_{0,2}, \ldots, x_{0,n}, x_{1,1}, x_{1,2}, \ldots, x_{1,n}\}$ and

$$ID'_n = \sum_{z_i\in\{x_{0,i},x_{1,i}\},i\in[n]} z_1 z_2 \ldots z_n z_1 \ldots z_n.$$

Clearly, $ID \leq_{iproj} ID'$. Hence $\mathrm{PER} \leq_{iproj} ID'$. Applying the transfer theorem (Theorem 20), we have that $\mathrm{PER} \leq_{proj} ID'^{(c)}$ in the commutative setting. However, $ID'^{(c)} = \prod_{i\in[n]}(x_{0,i}x_{0,n+i} + x_{1,i}x_{1,n+i})$. Thus, $ID'^{(c)}$ is a reducible polynomial with factors of degree 2. Since PER_n is irreducible for all n, it follows that PER cannot be $\leq_{proj}$ reducible to ID'. ∎

Assuming the SOS_k conjecture, Theorem 21 implies that ID is a VNP_{nc}-intermediate polynomial.

Corollary 22. *Assuming SOS_k conjecture, $ID \notin \mathrm{VP}_{nc}$ and ID is not VNP_{nc}-complete under $\leq_{iproj}$-reductions.*

Now we will show that there are infinitely many p-families $f^{(i)}$ such that $f^{(i)} \leq_{iproj} f^{(i+1)}$ but for all i $f^{(i+1)} \not\leq_{iproj} f^{(i)}$. For that we need the following observation that ID is not even VP_{nc}-hard w.r.t. $\leq_{iproj}$-reductions.

Theorem 23. *The p-family ID is not VP_{nc}-hard w.r.t $\leq_{iproj}$-reductions.*

Proof. We will prove that the Dyck p-family D_2 is not $\leq_{iproj}$-reducible to ID. Suppose $D_2 \leq_{iproj} ID$. Since the reduction is an indexed projection it follows that the polynomial family $\hat{D}_2$ defined below is also $\leq_{iproj}$-reducible to ID by essentially the same reduction. $\hat{D}_2 = (\hat{D}_{2,n})$, where $\hat{D}_{2,n}$ is a homogeneous degree $2n$ polynomial on variable set of size $4n$ $\{(_i,)_i, [_i,]_i | i \in [n]\}$ where $(_i,)_i, [_i$ and $]_i$ are variables that can occur only in i-th position. The polynomial $\hat{D}_{2,n}$ is defined as an indexed projection of $D_{2,n}$ obtained by replacing the i-th occurrence of a bracket $b \in \{(,), [,]\}$ by its indexed version $b_i \in \{(_i,)_i, [_i,]_i\}$. We observe that the p-families $\hat{D}_2$ and D_2 are $\leq_{iproj}$-reducible to each other.

Now, by assumption $\hat{D}_2 \le_{iproj} ID \le_{iproj} ID'$ which means that, by the transfer theorem (Theorem 21), that the commutative version $\hat{D}_2^{(c)} \le_{proj} ID'^{(c)}$. Now, we know for all n that $ID_n'^{(c)} = \prod_{i\in[n]}(x_{0,i}x_{0,n+i} + x_{1,i}x_{1,n+i})$. We show in the following claim that the commutative polynomials $\hat{D}_{2,n}^{(c)}$ are irreducible which rules out $\hat{D}_2^{(c)} \le_{proj} ID'^{(c)}$, and hence completes the proof by contradiction.

Claim 24. *The polynomial $\hat{D}_{2,n}^{(c)}$ is irreducible for each n.*

Proof of Claim: Suppose $\hat{D}_{2,n}^{(c)} = g.h$ is a nontrivial factorization. We will derive a contradiction. First, note that $\hat{D}_{2,n}^{(c)}$ is set-multilinear of degree $2n$ where the i-th location is allowed only variables from the set $\{(_i,)_i, [_i,]_i\}$. Since $\hat{D}_{2,n}^{(c)}$ is multilinear, it follows that both g and h are homogeneous multilinear and $Var(g) \cap Var(h) = \emptyset$, where $Var(g), Var(h)$ are the variables sets of g and h respectively.

Thus, every nonzero monomial m of f has a unique factorization $m = m_1m_2$, 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 and $[2n] \setminus S$ and the monomials of g are over variables in $\{(_i,)_i, [_i,]_i | i \in S\}$ and h's monomials are over variables in $\{(_i,)_i, [_i,]_i | i \in [2n] \setminus S\}$. Now, there are monomials m occurring in $\hat{D}_{2,n}^{(c)}$ 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 $]_i$. Then $m' = m_1'm_2'$, 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 $\hat{D}_{2,n}^{(c)}$, which is a contradiction. This completes the proof of the claim and hence the theorem. ∎

We have shown that ID is VNP_{nc}-intermediate assuming SOS_k conjecture. On the other hand, $D_2 \not\le_{iproj} ID$ unconditionally. Our aim is to use D_2 and ID to create an infinite collection $f^{(i)}$ of p-families in VNP_{nc} such that $f^{(i)} \le_{iproj} f^{(i+1)}$ but $f^{(i+1)} \not\le_{iproj} f^{(i)}$.

Let $ID = (ID_n)$ where ID_n are degree $2n$, and $D_2 = (D_{2,n})_{n\ge 0}$ where $D_{2,n}$ are degree $2n$.

- Define $f^{(1)} = ID$.
- $f^{(2)} = (f_n^{(2)})$ where $f_n^{(2)} = D_{2,n}ID_n$.
- $f^{(i)} = (f_n^{(i)}) = (D_{2,n}ID_n \dots D_{2,n}ID_n)$, where $f_n^{(i)} = f_n^{(i-1)}D_{2,n}ID_n$ for all i and n.

Clearly, $f^{(i)} \in \mathrm{VNP}_{nc}$ for all i.

Proposition 25. *For every i, $f^{(i)} \le_{iproj} f^{(i+1)}$, where the $f^{(i)}$ are the p-families defined above.*

Proof. We explain the easy proof for $f^{(1)} \le_{iproj} f^{(2)}$ which can be easily extended to all i. The indexed projection that gives a reduction from $f_n^{(1)}$ to $f_n^{(2)}$ will simply substitute 1 for the variables (occurring in positions $1 \le i \le n$, and 1 for the variables) occurring in positions

$n+1 \leq i \leq 2n$. For all other occurrences of the variables of $D_{2,n}$ the indexed projection substitutes 0. This substitution picks out the following unique degree-$2n$ monomial in $D_{2,n}$

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

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

Finally, the indexed projection substitutes x for x, for each variable x occurring in the polynomial ID_n. ■

Theorem 26. *Assuming the SOS_k-conjecture, for every i, we have $f^{(i+1)} \not\leq_{iproj} f^{(i)}$.*

Proof. Suppose to the contrary that $f^{(i+1)} \leq_{iproj} f^{(i)}$. Then there is a polynomial $p(n)$ and indexed projection map ϕ_n s.t $f^{(i)}_{p(n)}(\phi_n(X^{(i)}_{p(n)})) = f^{(i+1)}_n(X^{(i+1)}_n)$, where $X^{(i)}_{p(n)} = Var(f^{(i)}_{p(n)})$ and $X^{(i+1)}_n = Var(f^{(i+1)}_n)$. Now, we will derive a contradiction from this. We have:

- $f^{(i)}_{p(n)} = \underbrace{D_{2,p(n)}ID_{p(n)}\ldots D_{2,p(n)}ID_{p(n)}}_{i-times}$
- $f^{(i+1)}_n = \underbrace{D_{2,n}ID_n\ldots D_{2,n}ID_n}_{(i+1)-times}$

Since $ID_n \not\leq_{iproj} D_{2,n}$ (by [HWY10a] assuming SOS_k-conjecture), we have $D_{2,n}ID_n \not\leq_{iproj} D_{2,p(n)}$ and $D_{2,n}ID_n \not\leq_{iproj} ID_{2,p(n)}$ because of irreducibility of $\hat{D}^{(c)}_{2,n}$ (as shown in Theorem 21). Hence $D_{2,n}ID_n$ must get mapped by the projection ϕ_n to the product $D_{2,p(n)}ID_{p(n)}$ or $ID_{p(n)}D_{2,p(n)}$, overlapping both factors. But $f^{(i+1)}_n$ has $(i+1)$ such factors $D_{2,n}ID_n$. Hence, at least one of these factors $D_{2,n}ID_n$ must map wholly to $ID_{p(n)}$ or $D_{2,p(n)}$ by the indexed projection ϕ_n. If $D_{2,n}ID_n$ maps to $ID_{p(n)}$ that contradicts Theorem 23. If $D_{2,n}ID_n$ maps to $D_{2,p(n)}$ then ID_n must be in VP_{nc}, which is not true assuming the SOS_k conjecture. ■

7 Inside VP_{nc}

We first show that D_2 is strictly harder than PAL w.r.t $\leq_{abp}$-reductions.

Theorem 27. PAL $\leq_{abp} D_2$ *but* $D_2 \not\leq_{abp}$ PAL.

Proof. As PAL has polynomial size circuit, clearly PAL $\leq_{abp} D_2$ since D_2 is VP_{nc}-complete. For clarity, we give a direct reduction below. Consider $\mathrm{PAL}_n = \sum_{w\in\{x_0,x_1\}^n} w.w^R$ and $D_{2,n}$. The idea is to encode monomial ww^R by encoding x_0 as (and x_1 as [for position $i \in [n]$ and x_0 as) and x_1 as] for position $i \in [n+1, 2n]$. We can easily design an automaton with $O(n)$ states that replaces (in i-th position by x_0 and [in i-th position by x_1 for $i \in [n]$ and if it sees a closing bracket in any positions $i \in [n]$ it replaces it by 0. Similarly, the position from $n+1, \ldots, 2n$ are handled by replacing) in i-th position by x_0 and] in i-th position by x_1 and anything else by 0. The matrices defining these substitutions give the desired abp-reduction, which we explain now.

As in Theorem 12, we convert this automaton into a ABP A computing the homogeneous polynomial of degree $2n$. We now describe matrices we substitute for each parenthesis. Let

M_p be the matrix we substitute for parenthesis p whose rows and columns are labelled by nodes of this ABP A.

We define the matrix M_p for parenthesis p as follows:

$$m_{u,v} = M_p[u,v] = \begin{cases} x_0 & \text{if } p \in \{(,)\} \text{ and } \exists e = (u,v) \in \mathrm{E(A)} \text{ and label of } e \text{ is in } \{(,)\} \\ x_1 & \text{if } p \in \{[,]\} \text{ and } \exists e = (u,v) \in \mathrm{E(A)} \text{ and label of } e \text{ is in } \{[,]\} \end{cases}$$

where E(A) is the edge set of the automaton A.

We now turn to the converse problem. In fact, we only need to observe that PAL^2 is also $\leq_{abp}$-reducible to D_2, where PAL^2 is the square of the Palindrome polynomial. I.e. $\mathrm{PAL}^2 = (\mathrm{PAL}_n\mathrm{PAL}_n)_{n\geq 0}$. We can easily reduce $\mathrm{PAL}_n\mathrm{PAL}_n$ to $D_{2,2n}$ by repeating the automaton construction giving PAL_n from $D_{2,n}$ twice. The automaton will zero out all monomials of $D_{2,2n}$ except those of the form $u_1.u_2$ where u_1 has an equal number of (and) and equal number of [and] and similarly u_2.

Furthermore, while reading u_1 the automaton will do exactly as the reductions of PAL_n to $D_{2,n}$ and also for u_2 the same. This will yield the polynomial $\mathrm{PAL}_n\mathrm{PAL}_n$. Hence $\mathrm{PAL}^2 \leq_{abp} D_2$. However, $\mathrm{PAL}^2 \not\leq_{abp}$ PAL because, as shown in [LMS15], skew circuits computing PAL^2 require exponential size. This completes the proof sketch. ■

7.1 Dyck depth hierarchy inside VP_{nc}

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

Definition 28. *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 classes $\mathrm{VNC}^i_{nc}, i = 1, 2, \ldots$ are contained in VP_{nc}. Furthermore, it is easy to show using Nisan's rank argument that $\mathrm{VNC}^i_{nc}, i = 1, 2, \ldots$ form a strict hierarchy.[2]

It turns out that Dyck polynomials of nesting depth $\log^{i+1} n$ are hard for VNC^i_{nc} w.r.t. $\leq_{abp}$ reductions. Indeed, this follows from inspection of the proof of Theorem 12.

Definition 29 (Nesting depth)**.** *The nesting depth of a string in D_2 is defined as follows:*

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

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

Theorem 30. *Let $k_1 = \omega(\log n)$ and $k_2(n) \geq \omega(k_1(n))$ for all n. Then $D^{k_2}_2 \not\leq_{abp} D^{k_1}_2$ but $D^{k_1}_2 \leq_{abp} D^{k_2}_2$.*

[2] Palindromes of length $\log^{i+1} n$ have circuits of depth $\log^{i+1} n$ and polynomial in $\log^{i+1} n$ size. However, circuits of depth $\log^i n$ for it require superpolynomial size.

Proof. Suppose $D_2^{(k_2)} \leq_{abp} D_2^{(k_1)}$. Then there are polynomials $p(n)$ and $q(n)$ such that there is a matrix substitution ϕ_n for the variables X of $D_{2,p(n)}^{(k_1)}$ with the property that

$$D_{2,p(n)}^{(k_1)}(\phi_n(X))(1, q(n)) = D_{2,n}^{(k_2)},$$

where ϕ_n is a $q(n) \times q(n)$ matrix substitution for each variable in X. Now, the polynomial $D_{2,p(n)}^{(k_1(n))}$ has an ABP of size $2^{k_1(n)}.poly(n)$ (this ABP can be constructed by keeping the stack content as part of the DFA state for stack size at most $k_1(n)$). Combined with the matrix substitutions ϕ_n, we obtain a $2^{k_1(n)}.\text{poly}(n)$ size ABP for the polynomial $D_{2,n}^{(k_2(n))}$.

Furthermore, the reduction from PAL to D_2 (Theorem 27) can be easily modified to show that $\text{PAL}_{k_2} \leq D_{2,n}^{(k_2(n))}$ (the reduction will work only with the prefixes of length $2k_2(n)$ of $D_{2,n}^{(k_2(n))}$ and substitute rest by 1, if the prefix has same number of left and right brackets and 0 otherwise).

But by Nisan's [Nis91] rank argument PAL_{k_2} requires $2^{\Omega(k_2)}$ size ABPs contradicting the above $2^{k_1(n)}.poly(n)$ size ABP.

We now show the reduction $D_2^{k_1} \leq_{abp} D_2^{k_2}$. We design a DFA with $O(n.k_1(n))$ states that takes strings u of length $2n$ over $\{(,),[,]\}$ with an equal number of (&) and an equal number of [&] s.t in every prefix s of u, the number of left brackets exceed the number of right brackets by at most $k_1(n)$.

Corresponding to this DFA we can create matrix substitutions which replace each variable $x \in \{(,),[,]\}$ by itself if the string is accepted and otherwise, the $(2n)$-th variable by 0. Let ϕ_n define this matrix substitution. Then $D_{2,n}^{k_2(n)}\phi_n(X)) = D_{2,n}^{k_1(n)}$, where $X = Var(D_{2,n}^{k_2(n)})$. This completes the proof. ■

8 More on VNP_{nc}-Completeness

Apart from the polynomial family PER_d, we know from [AS10] that the polynomial family DET_d is VNP_{nc}-complete for $\leq_{abp}$-reductions. In this section we show some new VNP_{nc}-complete p-families w.r.t. $\leq_{abp}$ reductions and raise some open questions. In Theorem 21 we saw that ID is not VNP_{nc}-complete w.r.t. $\leq_{iproj}$ reductions. However, we do not know if ID is VNP_{nc}-complete w.r.t. $\leq_{abp}$ reductions.

Motivated by this question we consider a generalized version of ID which we call ID^* defined as follows:

For each positive integer n, let W_n denote the set of all degree n monomials of the form $x_{1,i_1} \ldots x_{n,i_n}$, over the variable set $\{x_{ij} \mid 1 \leq i, j \leq n\}$.

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

Theorem 31. $\text{PER} \leq_{abp} ID_n^*$.

Proof. Consider the permanent polynomial PER_n defined on the variable set $V_n = \{x_{ij} \mid 1 \leq i, j \leq n\}$. We design a polynomial in n sized deterministic automaton A with the following properties:

1. It takes inputs $w_1 w_2 \ldots w_{n^2}$ over alphabet V_n, where each w_i is of length n.

2. It checks that each w_i is a monomial of the form $w = X_{1i_1} \ldots X_{ni_n}$. I.e. the first index of the variables 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 $w_i = X_{1i_1} \ldots X_{ni_n}$ 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'. Transitions are only between adjacent layers, from i to $i+1$ for each i. Layers are grouped into blocks of size n. Let the blocks be $B_1, B_2, \ldots, B_{n^2}$. In block B_i, the transitions of the automaton will check if $i_j \neq i_k$ assuming $j \neq k$, where $i = (j,k)$. The automaton can have the indices j and k hardwired in the states corresponding to block B_i and 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 P into monomials of PER. 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. ■

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

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

Clearly $\mathrm{PER}^\chi = (\mathrm{PER}_n^\chi)$ is a p-family that is in VNP_{nc}. For which functions χ is PER^χ VNP_{nc}-complete? In other words, does the hardness of the noncommutative permanent depend only on the nonzero monomial set (and the coefficients are not important)? We give a partial answer to this question. 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 32. PER^* *is* VNP_{nc}*-complete.*

The above proposition is easy to prove: PER^* is in VNP_{nc} because coefficients of each monomial is polynomial-time computable from the monomial [HWY10b]. Furthermore, PER is $\leq_{iproj}$-reducible to PER^* by substituting 1 for all except the first n variables in every monomial.

Now, consider the polynomial

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

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

Theorem 33. *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^{χ}*.*
- *If χ is computable by a logspace Turing machine then* PER $\leq_{abp}$ $\mathrm{PER}^{*,\chi}$*.*

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

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

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

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

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

The difference between machine T' and T is that T' is a *1-way* logspace machine whose input head moves always to the right. For $\sigma \in S_n$, we can convert T' into a deterministic automaton with 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. ■

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.

[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.

[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.

[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.

[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