CHICAGO JOURNAL OF THEORETICAL COMPUTER SCIENCE 2020, Article 02, pages 1–18
http://cjtcs.cs.uchicago.edu/

On Explicit Branching Programs for the Rectangular Determinant and Permanent Polynomials

V. Arvind Abhranil Chatterjee Rajit Datta Partha Mukhpadhyay

Received November 20, 2019; Revised July 30, 2020; Published August 5, 2020

Abstract: We study the arithmetic circuit complexity of some well-known families of polynomials through the lens of parameterized complexity. Our main focus is on the construction of explicit algebraic branching programs (ABPs) for the determinant and the permanent polynomials of the *rectangular* symbolic matrix in both commutative and noncommutative settings. The main results are:

- We show an explicit $O^*(\binom{n}{\downarrow k/2})$-size ABP construction for the noncommutative permanent polynomial of a $k \times n$ symbolic matrix. We obtain this via an explicit ABP construction of size $O^*(\binom{n}{\downarrow k/2})$ for $S^*_{n,k}$, the noncommutative symmetrized version of the elementary symmetric polynomial $S_{n,k}$.
- We obtain an explicit $O^*(2^k)$-size ABP construction for the commutative rectangular determinant polynomial of the $k \times n$ symbolic matrix.
- In contrast, we show that evaluating the rectangular noncommutative determinant with rational matrix entries is #W[1]-hard.

Key words and phrases: Rectangular Permanent, Rectangular Determinant.

1 Introduction

The complexity of arithmetic computations is usually studied in the model of arithmetic circuits and its various restrictions. An *arithmetic circuit* is a directed acyclic graph with each indegree-0 node (called an input gate) labeled by either a variable in $X = \{x_1, x_2, \ldots, x_n\}$ or a scalar from the field $\mathbb{F}$, and all other

DOI: 10.4086/cjtcs.2020.002

nodes (called gates) labeled as either a $+$ gate or a $\times$ gate. At a special node (designated the output gate), the circuit computes a multivariate polynomial in $\mathbb{F}[x_1, x_2, \ldots, x_n]$. We also use $\mathbb{F}[\mathrm{X}]$ to denote the polynomial ring $\mathbb{F}[x_1, x_2, \ldots, x_n]$.

Arithmetic computations are also considered in the noncommutative setting. The free noncommutative ring $\mathbb{F}\langle y_1, y_2, \ldots, y_n\rangle$ is usually denoted by $\mathbb{F}\langle \mathrm{Y}\rangle$.[1] In the ring $\mathbb{F}\langle \mathrm{Y}\rangle$, monomials are words in Y^* and polynomials in $\mathbb{F}\langle \mathrm{Y}\rangle$ are $\mathbb{F}$-linear combinations of words. We define noncommutative arithmetic circuits essentially as their commutative counterparts. The only difference is that at each product gate in a noncommutative circuit there is a prescribed left to right ordering of its inputs.

A more restricted model than arithmetic circuits is the algebraic branching program. An *algebraic branching program* (ABP) is a directed acyclic graph with one in-degree-0 vertex called *source*, and one out-degree-0 vertex called *sink*. The vertex set of the graph is partitioned into layers $0, 1, \ldots, \ell$, with directed edges only only between adjacent layers (i to $i+1$). The source and the sink are at layers zero and ℓ respectively. Each edge is labeled by a linear form over variables $x_1, x_2, \ldots, x_n$. The polynomial computed by the ABP is the sum over all source-to-sink directed paths of the product of linear forms that label the edges of the path. An ABP is *homogeneous* if all edge labels are homogeneous linear forms. ABPs are defined in both commutative and noncommutative settings. For this paper, the size of an ABP is always the total number of vertices and edges.

The main purpose of the current paper is to present new arithmetic complexity upper bound results, in the form of *optimal* algebraic branching programs, for some important polynomials in both the commutative and the noncommutative domains. These results are partly motivated by our recent work on an algebraic approach to design efficient parameterized and exact algorithms [2].

We now proceed to define the polynomials and explain the results obtained.

Elementary Symmetric Polynomial

We first recall the definition of k^{th} elementary symmetric polynomial $S_{n,k} \in \mathbb{F}[\mathrm{X}]$, over the n variables $\mathrm{X} = \{x_1, x_2, \ldots, x_n\}$,

$$S_{n,k}(\mathrm{X}) = \sum_{S \subseteq [n]: |S|=k} \prod_{i \in S} x_i.$$

It is well-known that $S_{n,k}(\mathrm{X})$ can be computed by an algebraic branching program of size $O(nk)$. In this paper, we consider the noncommutative symmetrized version $S^*_{n,k}$, in the ring $\mathbb{F}\langle \mathrm{Y}\rangle$, defined as:

$$S^*_{n,k}(\mathrm{Y}) = \sum_{T \subseteq [n]: |T|=k} \sum_{\sigma \in S_k} \prod_{i \in T} y_{\sigma(i)}.$$

The complexity of the polynomial $S^*_{n,k}$ is first considered by Nisan in his seminal work on noncommutative computation [11]. Nisan shows that any ABP for $S^*_{n,k}$ is of size $\Omega(\binom{n}{\downarrow k/2})$.[2] Furthermore, Nisan also shows the *existence* of an ABP of size $O(\binom{n}{\downarrow k/2})$ for $S^*_{n,k}$. However, it is not clear how to construct such an ABP in time $O(\binom{n}{\downarrow k/2})$. Note that an ABP of size $O^*(n^k)$ for $S^*_{n,k}$ can be directly constructed in $O^*(n^k)$ time by opening up the expression completely.

[1] Throughout the paper, we use X to denote a set of commuting variables, and use Y or Z for a set of noncommuting variables.

[2] We use $\binom{n}{\downarrow r}$ to denote $\sum_{i=0}^{r} \binom{n}{i}$.

Remark 1.1. Throughout the paper, we use the notation $O^*(\cdot)$ freely to suppress the terms $\mathrm{poly}(n,k)$ which are asymptotically smaller than the main terms.

The main upper bound question is whether we can achieve any constant factor saving of the parameter k in the exponent, in terms of ABP size and the run time of the construction. In this paper, we give such an explicit construction of size $O^*(\binom{n}{\downarrow k/2})$, and the construction takes $O^*(\binom{n}{\downarrow k/2})$ time. Note that Nisan's result [11] also rules out any $\mathrm{FPT}(k)$-size ABP for $S^*_{n,k}$, where by $\mathrm{FPT}(k)$ we mean a bound of the form $f(k)n^{O(1)}$.

Rectangular Permanent and Rectangular Determinant Polynomial

The next polynomial of interest in the current paper is the rectangular permanent polynomial. Given a $k \times n$ rectangular matrix $\mathrm{X} = (x_{i,j})_{1\le i\le k, 1\le j\le n}$ of commuting variables or a $k \times n$ rectangular matrix $\mathrm{Y} = (y_{i,j})_{1\le i\le k, 1\le j\le n}$ of noncommuting variables, the rectangular permanent polynomial in commutative and noncommutative domains are defined as follows

$$\mathrm{rPer}(\mathrm{X}) = \sum_{\sigma\in I_{k,n}} \prod_{i=1}^{k} x_{i,\sigma(i)}, \qquad\qquad \mathrm{rPer}(\mathrm{Y}) = \sum_{\sigma\in I_{k,n}} \prod_{i=1}^{k} y_{i,\sigma(i)}.$$

Here, $I_{k,n}$ is the set of all injections from $[k] \to [n]$. An alternative view is that $\mathrm{rPer}(\mathrm{X}) = \sum_{S\subset[n]:|S|=k} \mathrm{Per}(\mathrm{X}_S)$ where X_S is the $k \times k$ submatrix where the columns are indexed by the set S. Of course, such a polynomial can be computed in time $O^*(n^k)$ using a circuit of similar size, the main interesting issue is to understand whether the dependence on the parameter k can be improved. It is implicit in the work of Vassilevska and Williams [13] that the $\mathrm{rPer}(\mathrm{X})$ polynomial in the commutative setting can be computed by an algebraic branching program of size $O^*(2^k)$. This problem originates from its connection with combinatorial problems studied in the context of exact algorithm design [13].

In the noncommutative setting, we start with the polynomial $S^*_{n,k}(\mathrm{Y})$ and then make the polynomial set-multilinear to obtain $\mathrm{rPer}(\mathrm{Y})$. More precisely, for each variable index i and position j we replace y_i at position j by the new variable $y_{j,i}$. With a slight abuse of notation, it is easy to see that the resulting polynomial is $\mathrm{rPer}(\mathrm{Y})$ where Y is a $k \times n$ symbolic matrix of noncommuting variables $y_{j,i}$. Since we already have an explicit ABP construction for $S^*_{n,k}(\mathrm{Y})$ polynomial which is of size $O^*(\binom{n}{\downarrow k/2})$, it suffices to make the ABP set-multilinear as described above to obtain an ABP for $\mathrm{rPer}(\mathrm{Y})$ in the *noncommutative setting*. Clearly the resulting ABP's size and the time required to construct it are both bounded by $O^*(\binom{n}{\downarrow k/2})$.

As in the usual commutative case, the noncommutative determinant polynomial of a symbolic matrix $\mathrm{Y} = (y_{i,j})_{1\le i,j\le k}$ is defined as follows (the variables in the monomials are ordered from left to right):

$$\mathrm{Det}(\mathrm{Y}) = \sum_{\sigma\in S_k} \mathrm{sgn}(\sigma)\, y_{1,\sigma(1)} \cdots y_{k,\sigma(k)}.$$

Nisan [11] has also shown that algebraic branching programs for the *noncommutative determinant* of a $k \times k$ symbolic matrix require size $\Omega(2^k)$. In this paper we give an explicit construction of such an ABP in time $O^*(2^k)$. Here too, the main point is that Nisan has also shown that the lower bound is tight, but we provide an explicit efficient construction.

Remark 1.2. The noncommutative determinant we have defined above is known as the *Cayley determinant*. There are other kinds of noncommutative determinant studied in the literature [1]. From a complexity-theoretic perspective, the Cayley determinant is perhaps the most interesting and well-studied, mainly for its connections to efficiently approximating the permanent (see e.g. [4]).

Motivated by the aforementioned result of Vassilevska and Williams [13], we also study the complexity of the rectangular determinant polynomial (in commutative domain) defined as follows.

$$\mathrm{rDet}(\mathrm{X}) = \sum_{S \in \binom{[n]}{k}} \mathrm{Det}(\mathrm{X}_S).$$

We prove that the rectangular determinant polynomial can be computed using $O^*(2^k)$-size explicit ABP.

Finally, we consider the problem of evaluating the noncommutative rectangular determinant over matrix algebras and show that it is #W[1]-hard for polynomial dimensional matrices. Hence the noncommutative rectangular determinant is unlikely to have an explicit $O^*(n^{o(k)})$-size ABP. Recently, we have shown the #W[1]-hardness of computing the noncommutative rectangular permanents over polydimensional rational matrices [2]. We note that the noncommutative $n \times n$ determinant over matrix algebras is well-studied, and computing it remains #P-hard even over 2×2 rational matrices [4, 8, 9]. Our proof technique is based on the Hadamard product of noncommutative polynomials [4]. However, the crucial difference is that, to show the #P-hardness of the noncommutative determinant, the proof in [4] shows a reduction from the evaluation of the commutative permanent to the noncommutative determinant; whereas, the #W[1]-hardness of the noncommutative rectangular determinant requires a different proof route because the commutative rectangular permanent is in FPT. In fact, we show that the rectangular determinant (and the rectangular permanent), whose entries are $r \times r$ matrices over any field, can be computed in time $O^*(2^k r^{2k})$.

Our Results

We first formally define what we mean by *explicit* circuit upper bounds.

Definition 1.3 (Explicit Circuit Upper Bound)**.** A family $\{f_n\}_{n>0}$ of degree-k polynomials in the commutative ring $\mathbb{F}[x_1, x_2, \ldots, x_n]$ (or the noncommutative ring $\mathbb{F}\langle y_1, y_2, \ldots, y_n\rangle$) is $q(n,k)$-*explicit* if there is an $O^*(q(n,k))$ time-bounded uniform algorithm that on input $\langle 0^n, k\rangle$ outputs a circuit C_n of size $O^*(q(n,k))$ computing f_n.

The same definition applies to the case of explicit ABP families as well. In this paper, we show the following explicit upper bound results.

Theorem 1.4.

1. *The family of symmetrized elementary polynomials* $\{S^*_{n,k}(\mathrm{Y})\}_{n>0}$ *has* $\binom{n}{\downarrow k/2}$*-explicit ABPs over any field.*

2. *The noncommutative rectangular permanent family* $\{\mathrm{rPer}(\mathrm{Y})\}_{n>0}$*, where* Y *is a* $k \times n$ *symbolic matrix of variables has* $\binom{n}{\downarrow k/2}$*-explicit ABPs.*

Remark 1.5. We note here that there is an algorithm of run time $O^*(\binom{n}{\downarrow k/2})$ for computing the rectangular permanent over rings and semirings [7]. Our contribution in the second part of Theorem 1.4 is that we obtain an $\binom{n}{\downarrow k/2}$-explicit ABP for it.

Theorem 1.6.

1. *The family of noncommutative determinants* $\{\mathrm{Det}(\mathrm{Y})\}_{k>0}$ *has* 2^k*-explicit ABPs over any field.*
2. *There is a family* $\{f_n\}$ *of noncommutative degree-k polynomials* f_n *such that* f_n *has the same support as* $S^*_{n,k}$*, and it has* 2^k*-explicit ABPs. This result holds over any field that has at least n distinct elements.*
3. *The commutative rectangular determinant family* $\{\mathrm{rDet}(\mathrm{X})\}_{k>0}$*, where* X *is a* $k \times n$ *matrix of variables has* 2^k*-explicit ABPs.*

Remark 1.7. It is interesting to compare the results stated in Theorem 1.4 and Theorem 1.6. In Theorem 1.4, the polynomial families are defined over two parameters n and k, and the size of the ABPs are bounded by $O^*(\binom{n}{\downarrow k/2})$. The first result stated in Theorem 1.6 considers a polynomial family with only one parameter k, and the size of the ABP is $O^*(2^k)$. However, the second and the third results consider the families of polynomials over two parameters n and k (like in Theorem 1.4) but the ABP sizes show parameterized dependence on the parameter k only.

We stress here that the constructive aspect of the above upper bounds is new. The *existence* of the ABPs claimed in the first two parts of Theorem 1.4 and the first part of Theorem 1.6 follows from Nisan's work [11] which shows a tight connection between optimal ABP-size for some $f \in \mathbb{F}\langle \mathrm{Y} \rangle$ and ranks of the matrices M_r whose rows are labeled by degree r monomials, columns by degree $k-r$ monomials and the $(m_1, m_2)^{th}$ entry is the coefficient of $m_1 m_2$ in f.

Remark 1.8. It is important to note that one can also make Nisan's results constructive using existing techniques. For example, it is possible to adapt the learning algorithm in [5] to design uniform algorithms which will construct the ABP families stated in Theorem 1.4. However, the constructions are not $\binom{n}{\downarrow k/2}$-explicit.

Next we describe the parameterized hardness result for rectangular determinant polynomial when we evaluate over matrix algebras.

Theorem 1.9. *For any fixed* $\varepsilon > 0$*, evaluating the* $k \times n$ *rectangular determinant polynomial over* $n^\varepsilon \times n^\varepsilon$ *rational matrices is* #W[1]*-hard, treating k as fixed parameter.*

However, we can easily design an algorithm of run time $O^*(2^k r^{2k})$ for computing the rectangular permanent and determinant polynomials with $r \times r$ matrix entries over any field.

Organization

The paper is organized as follows. In Section 2, we provide the necessary background. The proofs of Theorem 1.4 and Theorem 1.6 are given in Section 3 and Section 4 respectively. We prove Theorem 1.9 in Section 5.

2 Preliminaries

We provide some background results from noncommutative computation. Given a commutative circuit C, we can naturally associate a noncommutative circuit C^{nc} by prescribing an input order at each multiplication gate. This is captured in the following definition.

Definition 2.1. Given a commutative circuit C computing a polynomial in $\mathbb{F}[x_1, x_2, ..., x_n]$, the noncommutative version of C, C^{nc} is the noncommutative circuit obtained from C by fixing an ordering at each product gate in C (e.g. from left to right) and replacing x_i by the noncommuting variable $y_i : 1 \leq i \leq n$.

Let $f \in \mathbb{F}[\mathrm{X}]$ be a homogeneous degree-k polynomial computed by a circuit C, and let $\hat{f}(\mathrm{Y}) \in \mathbb{F}\langle \mathrm{Y} \rangle$ be the polynomial computed by C^{nc}. Let X_k denote the set of all degree-k monomials over X. As usual, Y^k denotes all degree-k noncommutative monomials (i.e., words) over Y. Each monomial $m \in \mathrm{X}_k$ can appear as different noncommutative monomials $\hat{m}$ in $\hat{f}$. We use the notation $\hat{m} \to m$ to denote that $\hat{m} \in \mathrm{Y}^k$ will be transformed to $m \in X_k$ by substituting x_i for $y_i, 1 \leq i \leq n$. We use the notation $[m]f$ (resp. $[\hat{m}]\hat{f}$) to denote the coefficient of m in the polynomial f (resp. in the polynomial $\hat{f}$). Then, we observe the following

$$[m]f = \sum_{\hat{m} \to m} [\hat{m}]\hat{f}.$$

For each monomial $\hat{m} = y_{i_1} y_{i_2} \cdots y_{i_k}$, the permutation $\sigma \in S_k$, where S_k is the symmetric group of all $k!$ permutations on $\{1, 2, \ldots, k\}$, maps $\hat{m}$ to the monomial $\hat{m}^\sigma$ defined as $\hat{m}^\sigma = y_{i_{\sigma(1)}} y_{i_{\sigma(2)}} \cdots y_{i_{\sigma(k)}}$. By linearity, $\hat{f} = \sum_{\hat{m} \in \mathrm{Y}^k} [\hat{m}]\hat{f} \cdot \hat{m}$ is mapped by σ to the polynomial, $\hat{f}^\sigma = \sum_{\hat{m} \in \mathrm{Y}^k} [\hat{m}]\hat{f} \cdot \hat{m}^\sigma$. This gives rise to the following definition.

Definition 2.2. The *symmetrized polynomial* of f, f^* is the degree-k homogeneous polynomial $f^* = \sum_{\sigma \in S_k} \hat{f}^\sigma$.

Next, we recall the definition of Hadamard product of two polynomials.

Definition 2.3. Given polynomials f, g, their Hadamard product is defined as

$$f \circ g = \sum_m ([m]f \cdot [m]g) \cdot m.$$

In the commutative setting, computing the Hadamard product is intractable in general. This is readily seen as the Hadamard product of the determinant polynomial with itself yields the permanent polynomial. However, in the noncommutative setting the Hadamard product of two ABPs can be computed efficiently [3].

Theorem 2.4. [3] *Given a noncommutative ABP of size S' for a degree k polynomial $f \in \mathbb{F}\langle y_1, y_2, \ldots, y_n \rangle$ and a noncommutative ABP of size S for another degree k polynomial $g \in \mathbb{F}\langle y_1, y_2, ..., y_n \rangle$, we can compute a noncommutative ABP of size SS' for $f \circ g$ in deterministic $SS' \cdot \mathrm{poly}(n, k)$ time.*

Let C be a circuit and B an ABP computing homogeneous degree-k polynomials $f, g \in \mathbb{F}\langle \mathrm{Y} \rangle$ respectively. Then their Hadamard product $f \circ g$ has a noncommutative circuit of polynomially bounded size which can be computed efficiently [3].

Furthermore, if C is given by black-box access then $f \circ g(a_1, a_2, \ldots, a_n)$ for $a_i \in \mathbb{F}, 1 \leq i \leq n$ can be evaluated by evaluating C on matrices defined by the ABP B [4] as follows: For each $i \in [n]$, the transition matrix $M_i \in M_s(\mathbb{F})$ are computed from the noncommutative ABP B (which is of size s) that encode layers. We define $M_i[k,\ell] = [x_i]L_{k,\ell}$, where $L_{k,\ell}$ is the linear form on the edge (k,ℓ). Now to compute $(f \circ g)(a_1, a_2, \ldots, a_n)$ where $a_i \in \mathbb{F}$ for each $1 \leq i \leq n$, we compute $C(a_1M_1, a_2M_2, \ldots a_nM_n)$. The value $(f \circ g)(a_1, a_2, \ldots, a_n)$ is the $(1,s)^{th}$ entry of the matrix $f(a_1M_1, a_2M_2, \ldots, a_nM_n)$.

Lemma 2.5. [4] *Given a circuit C and a ABP B computing homogeneous noncommutative polynomials f and g in $\mathbb{F}\langle \mathrm{Y} \rangle$, the Hadamard product $f \circ g$ can be evaluated at any point $(a_1, \ldots, a_n) \in \mathbb{F}^n$ by evaluating $C(a_1M_1, \ldots, a_nM_n)$ where $M_1, \ldots, M_n$ are the transition matrices of B, and the dimension of each M_i is the size of B.*

3 The Proof of Theorem 1.4

In this section, we present the construction of explicit ABPs for $S^*_{n,k}(\mathrm{Y})$ and noncommutative rPer(Y).

3.1 The construction of ABP for $S^*_{n,k}(\mathrm{Y})$

The construction of the ABP for $S^*_{n,k}(\mathrm{Y})$ is inspired by an inclusion-exclusion based dynamic programming algorithm for the disjoint sum problem [6].

Proof of Theorem 1.4.1. Let us denote by F the family of subsets of $[n]$ of size exactly $k/2$. Let $\downarrow F$ denote the family of subsets of $[n]$ of size at most $k/2$. For a subset $S \subset [n]$, we define $m_S = \prod_{j \in S} y_j$ where the product is taken in the natural order. We define the polynomial

$$f_S = \sum_{\sigma \in S_{k/2}} \prod_{j=1}^{k/2} y_{i_{\sigma(j)}},$$

for $S \in F$ and $S = \{i_1, i_2, \ldots, i_{k/2}\}$. For subsets $S \notin F$, we define $f_S = 0$. Note that, for each $S \in F$, the polynomial f_S is the symmetrization of the monomial m_S.

For each $S \in \downarrow F$, we define $\hat{f}_S = \sum_{S \subseteq A} f_A$ where $A \in F$. We now show, using the inclusion-exclusion principle, that we can express $S^*_{n,k}$ with an appropriate combination of these symmetrized polynomials for different subsets.

Lemma 3.1.

$$S^*_{n,k} = \sum_{S \in \downarrow F} (-1)^{|S|} \hat{f}^2_S.$$

Proof. Note that $S^*_{n,k} = \sum_{A \in F} \sum_{B \in F} (A \cap B = \emptyset) f_A f_B$, where for any proposition P we use (P) to denote

its truth-value (namely, 1 if P is true and 0 if P is false). By the inclusion-exclusion principle:

$$\begin{aligned} S^*_{n,k} &= \sum_{A\in F}\sum_{B\in F}(A\cap B=\emptyset)f_A f_B \\ &= \sum_{A\in F}\sum_{B\in F}\sum_{S\in\downarrow F}(-1)^{|S|}(S\subseteq A\cap B)f_A f_B \\ &= \sum_{S\in\downarrow F}(-1)^{|S|}\sum_{A\in F}\sum_{B\in F}(S\subseteq A)(S\subseteq B)f_A f_B \\ &= \sum_{S\in\downarrow F}(-1)^{|S|}\left(\sum_{A\in F}(S\subseteq A)f_A\right)^2 = \sum_{S\in\downarrow F}(-1)^{|S|}\hat{f}_S^2. \end{aligned}$$

□

Now we describe two ABPs where the first ABP simultaneously computes f_A for each $A\in F$ and the second one simultaneously computes $\hat{f}_S$ for each $S\in\downarrow F$.

Lemma 3.2. *There is an $\binom{n}{\downarrow k/2}$-explicit multi-output ABP B_1 that outputs the collection $\{f_A\}$ for each $A\in F$.*

Proof. The ABP consists of $(k/2+1)$ layers where layer $\ell\in\{0,1,\ldots,k/2\}$ has $\binom{n}{\ell}$ many nodes indexed by ℓ size subsets of $[n]$. In $(\ell+1)^{th}$ layer, the node indexed by the set T is connected to the nodes $T\setminus\{j\}$ in the previous layer with an edge label y_j for each $j\in T$. Clearly, in the last layer, the sink node indexed by the set A computes the polynomial f_A. □

Lemma 3.3. *There is an $\binom{n}{\downarrow k/2}$-explicit multi-output ABP B_2 that outputs the collection $\{\hat{f}_S\}$ for each $S\in\downarrow F$.*

Proof. To construct such an ABP, we use ideas from [6]. We define $\hat{f}_{i,S}=\sum_{S\subseteq A}f_A$ where $S\subseteq A$ and $A\cap[i]=S\cap[i]$. Note that, $\hat{f}_{n,S}=f_S$ and $\hat{f}_{0,S}=\hat{f}_S$. From the definition, it is clear that $\hat{f}_{i-1,S}=\hat{f}_{i,S}+\hat{f}_{i,S\cup\{i\}}$ if $i\notin S$ and $\hat{f}_{i-1,S}=\hat{f}_{i,S}$ if $i\in S$. Hence, we can take a copy of ABP B_1 from Lemma 3.2, and then simultaneously compute $\hat{f}_{i,S}$ for each $S\in\downarrow F$ and i ranging from n to 0. Clearly, the new ABP B_2 consists of $(n+k/2+1)$ many layers and at most $\binom{n}{\downarrow k/2}$ nodes at each layer. The number of edges in the ABP is also linear in the number of nodes. □

Let $f=\sum_{m\in Y^k}[m]f\cdot m$ be a noncommutative polynomial of degree k in $\mathbb{F}\langle Y\rangle$. The *reverse* of f is defined as the polynomial

$$f^R=\sum_{m\in Y^k}[m]f\cdot m^R,$$

where m^R is the reverse of the word m.

Lemma 3.4. *[Reversing an ABP] Suppose B is a multi-output ABP with r sink nodes where the i^{th} sink node computes $f_i\in\mathbb{F}\langle Y\rangle$ for each $i\in[r]$. We can construct an ABP of twice the size of B that computes the polynomial $\sum_{i=1}^r f_i\cdot L_i\cdot f_i^R$ where L_i are affine linear forms.*

Proof. Suppose B has ℓ layers, then we construct an ABP of $2\ell+1$ layers where the first ℓ layers are the copy of ABP B and the last ℓ layers are the *mirror image* of the ABP B, call it B^R. More precisely, the ABP B^R is a r-source and single sink ABP, where the polynomial computed between i^{th} source and the sink is the polynomial f_i^R. In the $(\ell+1)^{th}$ layer we connect the i^{th} sink node of ABP B to the i^{th} source node of B^R by an edge with edge label L_i. Note that, B^R has r source nodes and one sink node and the polynomial computed between ith source node and sink is f_i^R. □

Now, applying the construction of Lemma 3.4 to the multi-output ABP B_2 of Lemma 3.3 with $L_S = (-1)^{|S|}$ we obtain an ABP that computes the polynomial $\sum_S (-1)^{|S|} \hat{f}_S \cdot \hat{f}_S^R$. Since $\hat{f}_S$ is a symmetrized polynomial, we note that $\hat{f}_S^R = \hat{f}_S$ and using Lemma 3.1 we conclude that this ABP computes $S^*_{n,k}$. The ABP size is $O(k\binom{n}{\downarrow k/2})$. □

Remark 3.5. Given an arithmetic circuit C of size s computing a degree-k polynomial $f \in \mathbb{F}[\mathrm{X}]$, it is possible to compute the sum of the coefficients of all the multilinear terms present in f in $O(n^k)$ time. This can be done by opening up the circuit completely. In [10], Koutis and Williams raise as an open problem whether one can beat this brute-force run time. A solution of this problem is given in [2] by designing an algorithm of run time $O^*(\binom{n}{\downarrow k/2} \cdot n^{c \cdot \log k})$ (where c is a constant). The key observation is that $f \circ S_{n,k}(1)$ is the sum of the coefficients of all the multilinear terms. It is also proved that for any $a \in \mathbb{F}^n : f \circ S_{n,k}(a) = C^{nc} \circ S^*_{n,k}(a)$. Then the algorithm reduces the computation of $C^{nc} \circ S^*_{n,k}(1)$ to a computation of a rectangular permanent over matrix inputs and appeal to the algorithm in [7]. Using Theorem 1.4, one can give a simple solution. Let A be the ABP obtained for $S^*_{n,k}$ from Theorem 1.4. Using the standard circuit to ABP conversion method [12], one can first construct an ABP B of size $s^{O(\log k)}$ for f. Now using Theorem 2.4, one can compute $B^{nc} \circ A(1)$ in time $O^*(\binom{n}{\downarrow k/2} \cdot n^{O(\log k)})$.

3.2 The construction of ABP for rPer(Y)

Proof of Theorem 1.4.2. A $\binom{n}{\downarrow k/2}$-explicit ABP for the rectangular permanent polynomial can be obtained easily from the $\binom{n}{\downarrow k/2}$-explicit ABP for $S^*_{n,k}(\mathrm{Y})$ by making the ABP set-multilinear.

More precisely, let A be the $\binom{n}{\downarrow k/2}$-explicit ABP for $S^*_{n,k}(\mathrm{Y})$ obtained above. Recall that,

$$S^*_{n,k}(\mathrm{Y}) = \sum_{T \subseteq [n]:|T|=k} \sum_{\sigma \in S_k} \prod_{i \in T} y_{\sigma(i)}.$$

To make the ABP set-multilinear, we simply rename the variables $y_i : 1 \le i \le n$ at the position $1 \le j \le k$ by $y_{j,i}$. It is immediate that the resulting polynomial is rPer(Y). □

4 The Proof of Theorem 1.6

We prove the three parts of Theorem 1.6 in three subsections. More precisely, in this section we show the claimed explicit ABPs for the noncommutative determinant polynomial, for a polynomial having the same support as $S^*_{n,k}$, and for the commutative rectangular determinant polynomial.

4.1 A 2^k-explicit ABP for $k \times k$ noncommutative determinant

We now present an optimal explicit ABP construction for the noncommutative determinant polynomial $\mathrm{Det}(\mathrm{Y})$ for the square $k \times k$ symbolic matrix Y.

Proof of Theorem 1.6.1. The ABP B has $k+1$ layers with $\binom{k}{\ell}$ nodes at the layer ℓ for each $0 \leq \ell \leq k$. The source of the ABP is labeled $\emptyset$ and the nodes in layer ℓ are labeled by the distinct size ℓ subsets $S \subseteq [k]$, $1 \leq \ell \leq k$, hence the sink is labeled $[k]$. From the node labeled S in layer ℓ, there are $k-\ell$ outgoing edges $(S, S \cup \{j\})$, $j \in [k] \setminus S$.

Define the sign $\mathrm{sgn}(S, j)$ as $\mathrm{sgn}(S, j) = (-1)^{t_j}$, where t_j is the number of elements in S larger than j. Equivalently, treating S as a sorted list, t_j is the number of swaps required to insert j in the correct position, by starting with j appended to the right of the sorted list for S.

We connect the set S in the i^{th} layer to a set $S \cup \{j\}$ in the $(i+1)^{th}$ layer with the edge label $\mathrm{sgn}(S, j) \cdot y_{i+1,j}$ The source to sink paths in this ABP are in 1-1 correspondence to the node labels on the paths which give subset chains $\emptyset \subset T_1 \subset T_2 \subset \cdots \subset T_k = [k]$ such that $|T_i \setminus T_{i-1}| = 1$ for all $i \leq k$. Such subset chains are clearly in 1-1 correspondence with permutations $\sigma \in S_k$ listed as a sequence: $\sigma(1), \sigma(2), \ldots, \sigma(k)$, where $T_i = \{\sigma(1), \sigma(2), \ldots, \sigma(i)\}$. The following claim spells out the connection between the sign $\mathrm{sgn}(\sigma)$ of σ and the $\mathrm{sgn}(S, j)$ function defined above.

Claim 4.1. *For each $\sigma \in S_k$ and $T_i = \{\sigma(1), \sigma(2), \ldots, \sigma(i)\}$, we have*

$$\mathrm{sgn}(\sigma) = \prod_{i=1}^{k} \mathrm{sgn}(T_{i-1}, \sigma(i)).$$

Proof. We first note that $\mathrm{sgn}(\sigma) = (-1)^t$, if there are t transpositions $(r_i\ s_i), 1 \leq i \leq t$ such that $\sigma \cdot (r_1\ s_1) \cdot (r_2\ s_2) \cdots (r_t\ s_t) = 1$. Equivalently, interpreting this as sorting the list $\sigma(1), \sigma(2), \ldots, \sigma(k)$ by swaps $(r_i\ s_i)$, applying these t swaps will sort the list into $1, 2, \ldots, k$. As already noted, $\mathrm{sgn}(T_{i-1}, \sigma(i)) = (-1)^{t_i}$, where t_i is the number of swaps required to insert $\sigma(i)$ in the correct position into the sorted order of T_{i-1} (where $\sigma(i)$ is initially placed to the right of T_{i-1}). Hence, $\sum_{i=1}^{k} t_i$ is the total number of swaps required for this insertion sort procedure to sort $\sigma(1), \sigma(2), \ldots, \sigma(k)$. It follows that $\prod_{i=1}^{k} \mathrm{sgn}(T_{i-1}, \sigma(i)) = (-1)^{\sum_i t_i} = \mathrm{sgn}(\sigma)$, which proves the claim. □

Now, it is easy to see that the ABP computes $\mathrm{Det}(\mathrm{Y})$. For each $\sigma \in S_k$, Claim 4.1 guarantees the correct sign $\mathrm{sgn}(\sigma)$ for the monomial $\prod_{i=1}^{k} y_{i,\sigma(i)}$, where the monomial itself is defined through the edge labels as described above. □

Remark 4.2. Using subsets to label the nodes of the above ABP construction suggests itself from Nisan's work [11]. The new idea in the above result is the computation of the sign of the monomials of the noncommutative determinant.

4.2 A 2^k-explicit ABP weakly equivalent to $S^*_{n,k}$

A polynomial $f \in \mathbb{F}[\mathrm{X}]$ (resp. $\mathbb{F}\langle \mathrm{Y} \rangle$) is said to be *weakly equivalent* to a polynomial $g \in \mathbb{F}[\mathrm{X}]$ (resp. $\mathbb{F}\langle \mathrm{Y} \rangle$), if for each monomial m over X, $[m]f = 0$ if and only if $[m]g = 0$. For the construction of an ABP computing a polynomial weakly equivalent to $S^*_{n,k}$, we will suitably modify the ABP construction described above.

Proof of Theorem 1.6.2. Let $\alpha_i, 1 \leq i \leq n$ be distinct elements from $\mathbb{F}$. For each $j \in [k] \setminus S$, the edge $(S, S \cup \{j\})$ is labeled by the linear form $\text{sgn}(S,j) \cdot \sum_{i=1}^{n} \alpha_i^j y_i$, where $y_i, 1 \leq i \leq n$ are noncommuting variables. This gives an ABP B of size $O^*(2^k)$.

We show that the polynomial computed by ABP B is weakly equivalent to $S^*_{n,k}$. Clearly, B computes a homogeneous degree k polynomial in the variables $y_i, 1 \leq i \leq n$. We determine the coefficient of a monomial $y_{i_1} y_{i_2} \cdots y_{i_k}$. As noted, each source to sink path in B corresponds to a permutation $\sigma \in S_k$. Along that path the ABP compute the product of linear forms

$$\text{sgn}(\sigma) L_{\sigma(1)} L_{\sigma(2)} \cdots L_{\sigma(k)}, \text{ where } L_{\sigma(q)} = \sum_{i=1}^{n} \alpha_i^{\sigma(q)} y_i,$$

where the sign is given by the previous claim. The coefficient of the monomial $y_{i_1} y_{i_2} \cdots y_{i_k}$ in the above product is given by $\text{sgn}(\sigma) \prod_{q=1}^{k} \alpha_{i_q}^{\sigma(q)}$. Thus, the coefficient of $y_{i_1} y_{i_2} \cdots y_{i_k}$ in the ABP is given by $\sum_{\sigma \in S_k} \text{sgn}(\sigma) \prod_{q=1}^{k} \alpha_{i_q}^{\sigma(q)}$, which is the determinant of the $k \times k$ Vandermonde matrix whose q^{th} column is $(\alpha_{i_q}, \alpha_{i_q}^2, \ldots, \alpha_{i_q}^k)^T$. That determinant is non-zero if and only if the monomial $y_{i_1} y_{i_2} \cdots y_{i_k}$ is multilinear. Clearly the proof works for any field that contains at least n distinct elements. □

Remark 4.3. A polynomial $f \in \mathbb{Q}\langle \text{Y} \rangle$ is *positively weakly equivalent* to $S^*_{n,k}$, if for each multilinear monomial $m \in \text{Y}^k$, $[m]f > 0$. In the above proof, let g be the polynomial computed by ABP B that is weakly equivalent to $S^*_{n,k}$. Clearly, $f = g \circ g$ is positively weakly equivalent to $S^*_{n,k}$, and f has a 4^k-explicit ABP, since B is 2^k-explicit. This follows from Theorem 2.4. We leave open the problem of finding a 2^k-explicit ABP for some polynomial that is positively weakly equivalent to $S^*_{n,k}$. Such an explicit construction would imply a deterministic $O^*(2^k)$ time algorithm for k-path which is a longstanding open problem [10].

4.3 A 2^k-explicit ABP for $k \times n$ commutative rectangular determinant

We now present our ABP construction for the rectangular determinant polynomial $\text{rDet}(\text{X})$, where X is a $k \times n$ symbolic matrix.

Proof of Theorem 1.6.3. We start with modifying the ABP construction presented in Section 4.1. The changes are only to the edge labels. In the modified ABP, the linear form labeling the edge $(S, S \cup \{j\})$ is $\text{sgn}(S,j) \cdot (\sum_{i=1}^{n} x_{j,i} z_i)$, where the $z_i : 1 \leq i \leq n$ are fresh *noncommuting variables*, and the $x_{j,i} : 1 \leq j \leq k, 1 \leq i \leq n$ are the (commuting) variable entries of the symbolic matrix X. Clearly, as there is no change in the vertices or edges of the construction, the size of the new ABP is the same as the size of the ABP obtained in Section 4.1, which is $O^*(2^k)$.

Now, by an argument on the same lines as in Section 4.1, we note that the coefficient of the monomial $z_{i_1} z_{i_2} \ldots z_{i_k}$ where $i_1 < i_2 < \ldots < i_k$ is given by $\sum_{\sigma \in S_k} \text{sgn}(\sigma) x_{\sigma(1),i_1} \ldots x_{\sigma(k),i_k}$. Consider a fixed permutation $\sigma \in S_k$. Corresponding to σ we define the injection map $\tau^\sigma : [k] \to [n]$ as

$$\tau^\sigma(j) = i_{\sigma^{-1}(j)} \text{ for } 1 \leq j \leq k.$$

We define $\text{sgn}(\tau^\sigma)$ as the sign of the permutation π that sorts the list $(\tau^\sigma(1), \tau^\sigma(2), \ldots, \tau^\sigma(k))$ in increasing order. By definition, the list $(\tau^\sigma(1), \tau^\sigma(2), \ldots, \tau^\sigma(k))$ is

$$(i_{\sigma^{-1}(1)}, i_{\sigma^{-1}(2)}, \ldots, i_{\sigma^{-1}(k)}),$$

which is sorted in increasing order by the permutation σ. Hence $\mathrm{sgn}(\tau^\sigma) = \mathrm{sgn}(\sigma)$.

Let (j_1, j_2) be an index pair that is an inversion in σ. That is, $j_1 < j_2$ and $\sigma(j_1) > \sigma(j_2)$. Let $\ell_1 = \sigma(j_1)$ and $\ell_2 = \sigma(j_2)$. Then $i_{\tau^\sigma(\ell_1)} = i_{\sigma^{-1}(\ell_1)} = i_{j_1}$ and $i_{\tau^\sigma(\ell_2)} = i_{\sigma^{-1}(\ell_2)} = i_{j_2}$. Thus, $i_{\tau^\sigma(\ell_1)} < i_{\tau^\sigma(\ell_2)}$. Hence:

$$\sum_{\sigma \in S_k} \mathrm{sgn}(\sigma) x_{\sigma(1),i_1} \ldots x_{\sigma(k),i_k} = \sum_{\tau^\sigma \in I_{k,n}} \mathrm{sgn}(\tau^\sigma) x_{1,\tau^\sigma(1)} \ldots x_{k,\tau^\sigma(k)}.$$

Finally, the idea is to filter out only the good monomials $z_{i_1} z_{i_2} \ldots z_{i_k}$ where $i_1 < i_2 < \ldots < i_k$ from among all the monomials in the polynomial computed by the ABP. This can be done by taking the Hadamard product (using Theorem 2.4) with the following polynomial,

$$S^{nc}_{n,k}(\mathrm{Z}) = \sum_{S=\{i_1<i_2<\ldots<i_k\}} z_{i_1} z_{i_2} \ldots z_{i_k}.$$

Clearly, $S^{nc}_{n,k}$ has a $\mathrm{poly}(n,k)$-sized ABP which is just the noncommutative version (see Definition 2.1) of the well-known ABP for commutative $S_{n,k}$. Finally, we substitute each $z_i = 1$ to get the desired $O^*(2^k)$-size ABP for $\mathrm{rDet}(\mathrm{X})$. □

5 Hardness of Evaluating Rectangular Determinant Over Matrix Algebras

In this section, we prove a hardness result for evaluating the rectangular determinant over matrix algebras. More precisely, if A is a $k \times n$ matrix whose entries A_{ij} are $n^\varepsilon \times n^\varepsilon$ rational matrices for a fixed $\varepsilon > 0$, then it is #W[1]-hard to compute $\mathrm{rDet}(A)$. We show this by a reduction from the #W[1]-complete problem of counting the number of simple k-paths in directed graphs.

Let $G(V,E)$ be a directed graph with n vertices where $V(G) = \{v_1, v_2, \ldots, v_n\}$. A k-walk is a sequence of k vertices $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$ where $(v_{i_j}, v_{i_{j+1}}) \in E$ for each $1 \le j \le k-1$. A k-path is a k-walk where no vertex is repeated. Let A be the adjacency matrix of G, and let $z_1, z_2, \ldots, z_n$ be noncommuting variables. Define the $n \times n$ matrix B as follows:

$$B[i,j] = A[i,j] \cdot z_i, \quad 1 \le i, j \le n.$$

Let $\vec{1}$ denote the all 1's vector of length n. Let $\vec{z}$ be the length n vector defined by $\vec{z}[i] = z_i$. The polynomial $C_G \in \mathbb{F}\langle \mathrm{Z} \rangle$ is defined as

$$C_G(z_1, z_2, \ldots, z_n) = \vec{1}^T \cdot B^{k-1} \cdot \vec{z}.$$

Let W be the set of all k-walks in G. The following observation is folklore.

Observation 1.

$$C_G(z_1, z_2, \ldots, z_n) = \sum_{v_{i_1} v_{i_2} \ldots v_{i_k} \in W} z_{i_1} z_{i_2} \cdots z_{i_k}.$$

Hence, G contains a k-path if and only if the polynomial C_G contains a multilinear term.

5.1 The Proof of Theorem 1.9

As usual, let $[s]^{[r]}$ denote the set of all functions $f : [r] \to [s]$ and $I_{r,s}$ denote the set of injections $f : [r] \to [s]$. Now, for integers n and k we define the following subset of $[2n]^{[2k]}$:

$$S := \{f \in [2n]^{[2k]} | \exists g \in [n]^{[k]} \text{ such that } \forall i \in [k], f(2i-1) = g(i); f(2i) = n + g(i)\}.$$

Clearly, there is a bijection between S and $[n]^{[k]}$. We denote each $f \in S$ by f_g, where $g \in [n]^{[k]}$ is the corresponding function.

Also, note that $f_g \in S$ is an injection if and only if the corresponding g is an injection. Let $I \subset S$ denote the set of all injections in S. We note that $f_g \mapsto g$ is also a bijection from I to $I_{k,n}$.

Let $f \in I$. Recall that $\text{sgn}(f)$ is the sign of the permutation in S_{2k} that sorts the list $(f(1), f(2), \ldots, f(2k))$ in increasing order. Equivalently, we can sort this list by separately sorting the sublist of odd positions $(f(1), f(3), \ldots, f(2i-1), \ldots)$, and the sublist of even positions $(f(2), f(4), \ldots, f(2i), \ldots)$, and then moving the sorted sublist of odd positions to the first k positions. The two sublists of odd and even positions, which are of length k, are sorted by the same permutation $\pi \in S_k$. To complete the sorting, we apply the permutation $\varphi \in S_{2k}$ that moves the odd position elements to the first k positions in the list. Therefore, $\text{sgn}(f) = \text{sgn}(\pi)\text{sgn}(\pi)\text{sgn}(\varphi) = \text{sgn}(\varphi)$. By a simple counting argument, we observe that $\text{sgn}(\varphi) = (-1)^{1+2+\cdots+k-1} = (-1)^{\frac{k(k-1)}{2}}$. Hence, we have the following.

Observation 2. For each $f \in I, \text{sgn}(f) = (-1)^{\frac{k(k-1)}{2}}$.

Let Y denote the $2k \times 2n$ symbolic matrix with noncommuting variables $y_{i,j}$ as its $(i,j)^{th}$ entry. By abuse of notation, we use Y to also denote the set of variables $y_{i,j}$. That is, $\text{Y} = \{y_{1,1}, y_{1,2}, \ldots, y_{2k,2n}\}$. Now, each function $f \in [2n]^{[2k]}$ can be encoded as the noncommutative monomial $m_f = \prod_{i=1}^{2k} y_{i,f(i)}$.

Lemma 5.1. *There is an ABP B of* $\text{poly}(n,k)$ *size that computes the homogeneous degree-$2k$ polynomial $F \in \mathbb{F}\langle \text{Y} \rangle$:*

$$F = \sum_{f \in S} m_f.$$

Proof. The homogeneous ABP B we construct has $2k+1$ layers, labelled $\{0, 1, \ldots, 2k\}$. For each even index $2i, 0 \le i \le k$, there is exactly one node q_i in level $2i$ of the ABP. For each odd $2i-1, i \in [k]$, there are n nodes $p_{i,1}, p_{i,2}, \ldots, p_{i,n}$ at level $2i-1$. We now describe the edges of B. For each $i < k$ and $j \in [n]$, there is an edge from q_i to $p_{i+1,j}$ labelled $y_{i+1,j}$, and there is an edge from $p_{i,j}$ to q_{i+1} labelled $y_{i+1,n+j}$.

Let F be the polynomial computed by B. Notice, from the above construction, that the product of edge labels on each directed path from source to sink in the ABP B is a distinct monomial m_f for an $f \in S$. Furthermore, the coefficient of each monomial is 1, because distinct source to sink paths give rise to distinct monomials m_f. Thus, the polynomial computed by the ABP B is as claimed in the lemma. □

Clearly, the ABP B of the above proof can be computed in $\text{poly}(n,k)$ time. Suppose Y is a $2k \times 2n$ matrix where the $(i,j)^{th}$ entry is $y_{i,j}$. By Observation 2 and Lemma 5.1,

$$\text{rDet}(\text{Y}) \circ F(\text{Y}) = \sum_{f_g \in I} \text{sgn}(f_g) m_{f_g} = (-1)^{\frac{k(k-1)}{2}} \sum_{g \in I_{k,n}} m_{f_g}.$$

Let $Z = \{z_1, \ldots, z_n\}$ be a set of noncommuting variables. Define for each $g \in I_{k,n}$, $m'_g = \prod_{i=1}^{k} z_{g(i)}$. Define a map τ such that $\tau : y_{i,j} \mapsto z_j$ if i is odd, and $\tau : y_{i,j} \mapsto 1$ for even i. In other words, $\tau(m_{f_g}) = m'_g$. Notice that,

$$\begin{aligned} \mathrm{rDet}(\mathrm{Y}) \circ F(\mathrm{Y})|_\tau &= (-1)^{\frac{k(k-1)}{2}} \sum_{g \in I_{k,n}} m_{f_g}|_\tau \\ &= (-1)^{\frac{k(k-1)}{2}} \sum_{g \in I_{k,n}} m'_g \\ &= (-1)^{\frac{k(k-1)}{2}} S^*_{n,k}(\mathrm{Z}). \end{aligned}$$

Given a directed graph G on n vertices, we first construct an ABP for the noncommutative graph polynomial C_G over rationals. From the definition, it follows that C_G has a polynomial size ABP. Notice that, $((\mathrm{rDet}(\mathrm{Y}) \circ F(\mathrm{Y})|_\tau) \circ C_G(\mathrm{Z}))(\vec{1}) = S^*_{n,k}(\mathrm{Z}) \circ C_G(\mathrm{Z})(\vec{1})$ counts the number of directed k-paths in the graph G, and hence evaluating this term is #W[1]-hard. Let us modify the ABP for graph polynomial $C_G(\mathrm{Z})$ by replacing each edge labeled by z_j at i^{th} layer by two edges where the first edge is labeled by $y_{2i-1,j}$ and second one is labeled by $y_{2i,n+j}$. Let $C'_G(\mathrm{Y})$ is the new polynomial computed by the ABP. Notice that, each monomial of the modified graph polynomial looks like $\prod_{i=1}^{2k} y_{i,f(i)}$ for some $f : [2k] \mapsto [2n]$. More importantly, for each k-path $v_{i_1} v_{i_2} \ldots v_{i_k}$, if $g \in I_{k,n}$ is the corresponding injection, then $\prod_{i=1}^{k} z_{g(i)}$ is converted to $\prod_{i=1}^{2k} y_{i,f_g(i)}$ for $f_g \in S$. Notice that, $(\mathrm{rDet}(\mathrm{Y}) \circ F(\mathrm{Y})|_\tau) \circ C_G(\mathrm{Z}) = (\mathrm{rDet}(\mathrm{Y}) \circ F(\mathrm{Y}) \circ C'_G(\mathrm{Y}))|_\tau$ and hence, evaluating $(\mathrm{rDet}(\mathrm{Y}) \circ F(\mathrm{Y}) \circ C'_G(\mathrm{Y}))(\vec{1})$ is #W[1]-hard.

Now, assume to the contrary that we have an FPT algorithm to evaluate $\mathrm{rDet}(\mathrm{Y})$ over matrix inputs. As polynomials $C'_G(\mathrm{Y})$ and $F(\mathrm{Y})$ are computed by ABPs, we can efficiently obtain an ABP B' for computing their Hadamard product $F(\mathrm{Y}) \circ C'_G$. From ABP B', we construct the $t \times t$ transition matrices $M_{i,j}, i \in [2k], j \in [2n]$ corresponding to the variables $y_{i,j}$, where t is the size of the ABP B'. By Lemma 2.5, evaluating $(\mathrm{rDet}(\mathrm{Y}) \circ F(\mathrm{Y}) \circ C'_G(\mathrm{Y}))(\vec{1})$ (which is shown #W[1]-hard above) is equivalent to evaluating the rectangular determinant $\mathrm{rDet}(\mathrm{Y})$ on the matrix tuple $(M_{1,1}, \ldots, M_{2k,2n})$ which can be done by invoking the assumed algorithm on the $2k \times 2n$ matrix with matrix entries $M_{i,j}$, $i \in [2k], j \in [2n]$. This completes the proof of #W[1]-hardness of rDet with matrix entries.

By a simple reduction we can strengthen it to show #W[1]-hardness of evaluating rDet with entries from the $n^\varepsilon \times n^\varepsilon$ dimensional matrix algebras for any fixed $\varepsilon > 0$. □

5.2 Computing the rectangular permanent and the determinant over small dimensional algebras

Finally, in contrast to the above hardness result, we give a simple algorithm of run time $O^*(2^k r^{2k})$ to evaluate the rectangular permanent and the rectangular determinant of size $k \times n$ over $r \times r$ matrix algebra. Thus, over constant-dimensional matrix algebras these computations are fixed parameter tractable.

Theorem 5.2. *Let $\mathbb{F}$ be any field and $\mathcal{A}$ be an r-dimensional algebra over $\mathbb{F}$ with basis $e_1, e_2, \ldots, e_r$. Let $\{A_{ij}\}_{\substack{1 \le i \le k \\ 1 \le j \le n}}$ be a $k \times n$ matrix with $A_{ij} \in \mathcal{A}$. Then $\mathrm{rPer}(A)$ and $\mathrm{rDet}(A)$ can be computed in deterministic $O^*(2^k r^k)$ time.*

Proof. We present the proof for the rectangular permanent. The proof for the rectangular determinant is identical. The proof follows easily from expressing each entry $A_{i,j} \in \mathcal{A}$ in the given basis $e_1, e_2, \ldots, e_r$ for $\mathcal{A}$ over $\mathbb{F}$. First we note that

$$\begin{aligned}
\mathrm{rPer}(A) &= \sum_{f \in I_{k,n}} \prod_{i=1}^{k} A_{if(i)} \\
&= \sum_{f \in I_{k,n}} \prod_{i=1}^{k} \sum_{\ell=1}^{r} A_{if(i)}^{(\ell)} e_\ell \\
&= \sum_{f \in I_{k,n}} \sum_{(t_1,t_2,\ldots,t_k) \in [r]^k} \prod_{i=1}^{k} A_{if(i)}^{(t_i)} \prod_{i=1}^{k} e_{t_i} \\
&= \sum_{(t_1,t_2,\ldots,t_k) \in [r]^k} \left(\sum_{f \in I_{k,n}} \prod_{i=1}^{k} A_{if(i)}^{(t_i)} \right) \prod_{i=1}^{k} e_{t_i}.
\end{aligned} \tag{5.1}$$

Now we observe that

$$\sum_{f \in I_{k,n}} \prod_{i=1}^{k} A_{if(i)}^{(t_i)} = \mathrm{rPer}(A^{(t_1,t_2,\ldots,t_k)}),$$

where $A^{(t_1,t_2,\ldots,t_k)}$ is the $k \times n$ matrix defined as $A_{ij}^{(t_1,t_2,\ldots,t_k)} = A_{ij}^{(t_i)}$. Thus we have

$$\mathrm{rPer}(A) = \sum_{(t_1,t_2,\ldots,t_k) \in [r]^k} \mathrm{rPer}(A^{(t_1,t_2,\ldots,t_k)}) \prod_{i=1}^{k} e_{t_i}. \tag{5.2}$$

For a fixed $(t_1, t_2, \ldots, t_k) \in [r]^k$ the value $\mathrm{rPer}(A^{(t_1,t_2,\ldots,t_k)})$ can be computed in $O^*(2^k)$ time using the rectangular permanent algorithm [13]. Now we can compute $\mathrm{rPer}(A)$ by computing r^k many such rectangular permanents and putting them together according to equation 5.2. This gives a deterministic $O^*(2^k r^k)$ time algorithm for computing $\mathrm{rPer}(A)$. □

As a direct corollary we get the following.

Corollary 5.3. *Let $\mathbb{F}$ be any field and let A be a $k \times n$ matrix with $A_{ij} \in \mathbb{M}_{r \times r}(\mathbb{F})$. Then* $\mathrm{rPer}(A)$ *and* $\mathrm{rDet}(A)$ *can be computed in $O^*(2^k r^{2k})$ time.*

6 Conclusion

In this paper, we have presented the construction of explicit algebraic branching programs for the noncommutative symmetrized elementary symmetric polynomial, the noncommutative rectangular permanent polynomial, and the commutative rectangular determinant polynomial. Additionally, we have given an explicit algebraic branching program for the noncommutative square determinant polynomial. The constructions are essentially optimal, as they match the lower bounds of Nisan [11]. We have also shown that evaluating the rectangular determinant polynomial over matrix algebras is #W[1]-hard.

The paper opens further avenues of research. A very interesting problem is to tightly classify the complexity of computing the commutative rectangular $k \times n$ determinant polynomial. Is it computable in $\text{poly}(n,k)$ time? If not, can one show a complexity-theoretic hardness of evaluating the commutative determinant polynomial? We feel that the main obstacle is to interpret the rectangular determinant computation combinatorially. Another open question is whether or not there is an explicit algebraic branching program for the *noncommutative* rectangular determinant of size $O^*(n^{ck})$ for some $c < 1$, similar to our construction for the noncommutative rectangular permanent.

Acknowledgement

We thank the reviewers for their invaluable comments and suggestions.

References

[1] HELMUT ALASKEN: Quaternionic determinant. *Mathematical Intelligencer*, 18(3):57–65, 1996. [doi:10.1007/BF02404401] 4

[2] VIKRAMAN ARVIND, ABHRANIL CHATTERJEE, RAJIT DATTA, AND PARTHA MUKHOPADHYAY: Fast exact algorithms using Hadamard product of polynomials. In *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2019, December 11-13, 2019, Bombay, India*, volume 150 of *LIPIcs*, pp. 9:1–9:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. [doi:10.4230/LIPIcs.FSTTCS.2019.9] 2, 4, 9

[3] VIKRAMAN ARVIND, PUSHKAR S. JOGLEKAR, AND SRIKANTH SRINIVASAN: Arithmetic circuits and the hadamard product of polynomials. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2009, December 15-17, 2009, IIT Kanpur, India*, pp. 25–36, 2009. 6

[4] VIKRAMAN ARVIND AND SRIKANTH SRINIVASAN: On the hardness of the noncommutative determinant. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pp. 677–686, 2010. [doi:10.1145/1806689.1806782] 4, 7

[5] AMOS BEIMEL, FRANCESCO BERGADANO, NADER H. BSHOUTY, EYAL KUSHILEVITZ, AND STEFANO VARRICCHIO: Learning functions represented as multiplicity automata. *J. ACM*, 47(3):506–530, 2000. [doi:10.1145/337244.337257] 5

[6] ANDREAS BJÖRKLUND, THORE HUSFELDT, PETTERI KASKI, AND MIKKO KOIVISTO: Counting paths and packings in halves. In AMOS FIAT AND PETER SANDERS, editors, *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pp. 578–586. Springer, 2009. [doi:10.1007/978-3-642-04128-0_52] 7, 8

[7] ANDREAS BJÖRKLUND, THORE HUSFELDT, PETTERI KASKI, AND MIKKO KOIVISTO: Evaluation of permanents in rings and semirings. *Inf. Process. Lett.*, 110(20):867–870, 2010. [doi:10.1016/j.ipl.2010.07.005] 5, 9

[8] MARKUS BLÄSER: Noncommutativity makes determinants hard. *Information and Computation*, 243:133 – 144, 2015. 40th International Colloquium on Automata, Languages and Programming (ICALP 2013). [doi:https://doi.org/10.1016/j.ic.2014.12.010] 4

[9] STEVE CHIEN, PRAHLADH HARSHA, ALISTAIR SINCLAIR, AND SRIKANTH SRINIVASAN: Almost settling the hardness of noncommutative determinant. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pp. 499–508, 2011. [doi:10.1145/1993636.1993703] 4

[10] IOANNIS KOUTIS AND RYAN WILLIAMS: LIMITS and applications of group algebras for parameterized problems. *ACM Trans. Algorithms*, 12(3):31:1–31:18, 2016. [doi:10.1145/2885499] 9, 11

[11] NOAM NISAN: Lower bounds for non-commutative computation (extended abstract). In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pp. 410–418, 1991. [doi:10.1145/103418.103462] 2, 3, 5, 10, 15

[12] LESLIE G. VALIANT, SVEN SKYUM, S. BERKOWITZ, AND CHARLES RACKOFF: Fast parallel computation of polynomials using few processors. *SIAM J. Comput.*, 12(4):641–644, 1983. [doi:10.1137/0212043] 9

[13] VIRGINIA VASSILEVSKA WILLIAMS AND RYAN WILLIAMS: Finding, minimizing, and counting weighted subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013. [doi:10.1137/09076619X] 3, 4, 15

AUTHORS

V. Arvind
Professor
The Institute of Mathematical Sciences (HBNI), Chennai
arvind@imsc.res.in
https://www.imsc.res.in/~arvind/

Abhranil Chatterjee
Research Scholar
The Institute of Mathematical Sciences (HBNI), Chennai
abhranilc@imsc.res.in
https://www.imsc.res.in/abhranil_chatterjee

Rajit Datta
Research Scholar
Chennai Mathematical Institute
rajit@cmi.ac.in
https://www.cmi.ac.in/~rajit

Partha Mukhopadhyay
Associate Professor
Chennai Mathematical Institute
partham@cmi.ac.in
https://www.cmi.ac.in/~partham/